Norwegian University of
Science and Technology

Python Gamut Library

Forfattere

Jakob Michael Voigt
Lars Michael Niebuhr
Nawar Maher Behenam
Sahand Lahafdoozian

Bachelor i ingeniør - data
20 ECTS
Institute for Datateknikk og Informatikk
Norges teknisk-naturvitenskapelige universitet,

15.05.2017

Veileder Marius Pedersen

Sammendrag av Bacheloroppgaven

Tittel:	**Python Gamut Bibliotek**
Dato:	15.05.2017
Deltakere:	Jakob Michael Voigt Lars Michael Niebuhr Nawar Maher Behenam Sahand Lahafdoozian
Veileder:	Marius Pedersen
Oppdragsgiver:	Norwegian University of Science and Technology
Kontaktperson:	Ivar Farup, ivar.farup@ntnu.no, 61135227
Nøkkelord:	Bachelor, Latex, IE, IMT, NTNU, Gamut, Python, Colorspace, Fargebehandling, Algoritmer
Antall sider:	188
Antall vedlegg:	
Tilgjengelighet:	Åpen

Sammendrag:

Colour er et programmeringsbibliotek utviklet av prosjektets oppdragsgiver Ivar Farp. Bibliotekets formål er å forenkle forskning innen fargevitenskap og fargebildeteknologi, ved å tilby et enkelt API å jobbe med. Biblioteket har funksjonalitet for håndtering av fargemålinger og fargeromstransformasjoner. Oppdragsgiver ønsket å utvidde bibliotekt med funksjonalitet for håndtering av ICC-profiler, fargegamuter og utvikling av gamut mapping algoritmer. Hovedoppgaven vår har vært å utvikle en ny klasse for biblioteket som håndterer gamutrelatert arbeid. Klassen kan beregne gamuters overflater ved å benytte én av to utvalgte metoder. Gamutene kan visualiseres i brukerdefinere aksesystemer. Den største delen har vært å utvikle metoder for grunnoperasjoner som inngår i gamut mapping algoritmer. Vi har også implementert to slike algoritmer, HpminDE og minDE, ved å bruke klassens grunnoperasjoner. Ved å lese denne oppgaven får du innblikk i utviklingsprosessen, en teoretisk innføring i fagområdet avgrenset til hva som er relevant for vår utvikling, samt detaljerte forklaringer av algoritmene som benyttes. Biblioteket er tilgjengelig på github [1]

Summary of Graduate Project

Title:	**Python Gamut Library**
Date:	15.05.2017
Authors:	Jakob Michael Voigt Lars Michael Niebuhr Nawar Maher Behenam Sahand Lahafdoozian
Supervisor:	Marius Pedersen
Employer:	Norwegian University of Science and Technology
Contact Person:	Ivar Farup, ivar.farup@ntnu.no, 61135227
Keywords:	Thesis, Bachelor, Latex, IE, IMT, NTNU, Gamut, Python, Color space, Color management, Algorithms
Pages:	188
Attachments:	
Availability:	Open

Abstract:

Colour is a programming library developed by the project commissioner Ivar Farp. The purpose of the library is to simplify research within color science and color imaging technology, by offering a simple API to work with. The library has functionality for handling color metrics and color space transformations. The project commissioner wanted to expand the library with functionality for handling ICC profiles, color gamuts and development of gamut mapping algorithms. Our main task has been to develop a new class for the library that manages gamut-related work. The class can calculate gamut surfaces by using one of two developed methods. Gamutes can be visualized wtih user-defined axis. The largest part of the project been to develop methods for the basic operations used in gamut mapping algorithms. We have also implemented two such algorithms, HPminDE and minDE, using these operations. By reading this assignment you get insight into the development process, a theoretical introduction to the subject area, limited to what is relevant to our development, as well as detailed explanations of the algorithms used. The library is available at github [1]

Innhold

Figurer

Listings

1 Innledning

1.1 Forrord

Prosjektgruppen består av fire dataingeniørstudenter ved NTNU Gjøvik, samt oppdragsgiver Ivar Farup og veileder Marius Pedersen. Gjennom studiet har studentene blitt introdusert til emner om matematikk, algoritmiske metoder, programmering og systemutvikling. I valg av oppgave ønsket vi (studentene) å arbeide med en oppgave som relaterer seg til disse fagene. Blant alle oppgavene vi kunne velge mellom var det kun denne oppgaven som appellerte til oss alle. Oppgaven har vært meget utfordrene og vi har lært mye gjennom arbeid med dette prosjektet.

Vi vil spesielt takke følgende personer:

- Takk til Dr. Marius Pedersen for din fremragende innsats i rollen som gruppens veileder. Vi setter stor pris på hvor tilgjengelig du har vært, og kompetansen du har brakt inn i prosjektgruppen.
- Takk til Dr. Ivar Farup for alt du har gjort som prosjektets oppdragsgiver. Det har vært en fornøyelse å ha deg som oppdragsgiver. Du har involvert deg meget aktivt i hele utviklingsprosessen, med god hjelp og tydelig formulering av ønsker.

1.2 Bakgrunn

I 2014 startet oppdragsgiver utvikling av biblioteket colour for å forenkle arbeidet innenfor fargevitenskap og fargebildeteknologi [1]. Biblioteket benytter seg av et objekt orientert beregningsrammeverk for å behandle fargedata og metriske tensorer for fargerommet. Gjennom å objektorientere komponentene i prosessen og utvikle metoder som utfører de vanligste transformasjonene kan arbeid som før kunne ta dager, gjøres ved hjelp av noen få kodelinjer. I en artikkel skrevet av Ivar Farup i 2016 om biblioteket [2] står det « "*Future extensions could include ... computation and representation of colour gamuts (Bakke, Farup Hardeberg, 2010), as well as gamut mapping algorithms (Alsam Farup, 2009) in any colour space, and under any colour metric.*» . Det eksisterer altså et reelt behov for det denne oppgaven, og resultatet vil aktivt bli benyttet til forskning ved NTNU.

Et tilgjengelig verktøy for arbeid på gamuter, ICC3D(Interactive color correction in 3 dimensions)[3], ble utviklet i 2002 i samarbeid med The Norwegian Colour and Visual Computing Laboratory (fargelaben). ICC3D ble utviklet for å hjelpe forskere og fagfolk til å forstå og møte utfordringene i farge- og bildegjengivelse, spesielt for gamut mapping. Med tiden har ICC3D blitt utdatert da den baserer seg på en eldre versjon av biblioteket. Dette gjør at APIet er unødvendig tungvint å bruke.

1.3 Oppgavedefinisjon

1.3.1 Fagfelt / Problemområdet

Oppdragsgivers bibliotek mangler i dag funksjonalitet for gamut relatert arbeid. Han ønske å utvide biblioteket med verktøy for å beregne, representere og visualisere gamuter tilhørende forskjellige enheter og bilder. Han ønsker også et grensesnitt for implementering av gamut mapping algoritme(GMA).

1.3.2 Effektmål

- Redusere antall programmeringsbibliotek brukergruppen må forholde seg til ved arbeid med gamuter.
- Forenkle prosessen med å sammenligne resultater fra forskjellige GMA, ved bruk av kun colour biblioteket.
- Redusere tiden det tar å anvende GMAer.
- Redusere tiden det tar å tilføye nye GMAer i colour biblioteket.

1.3.3 Læringsmål

- Økte programmeringsferdigheter.
- Økte matteferdigheter.
- Reell erfaring med større utviklingsprosjekt.
- Økt kompetanse innen algoritmiske metoder og tolkning av komplekse konsepter.

1.3.4 Oppgavebeskrivelse

Prosjektgruppens oppgave er å videreutvikle Python biblioteket colour, ved å legge til modulen colour.gamut. For å gi en oversikt kan modulen dele i tre.

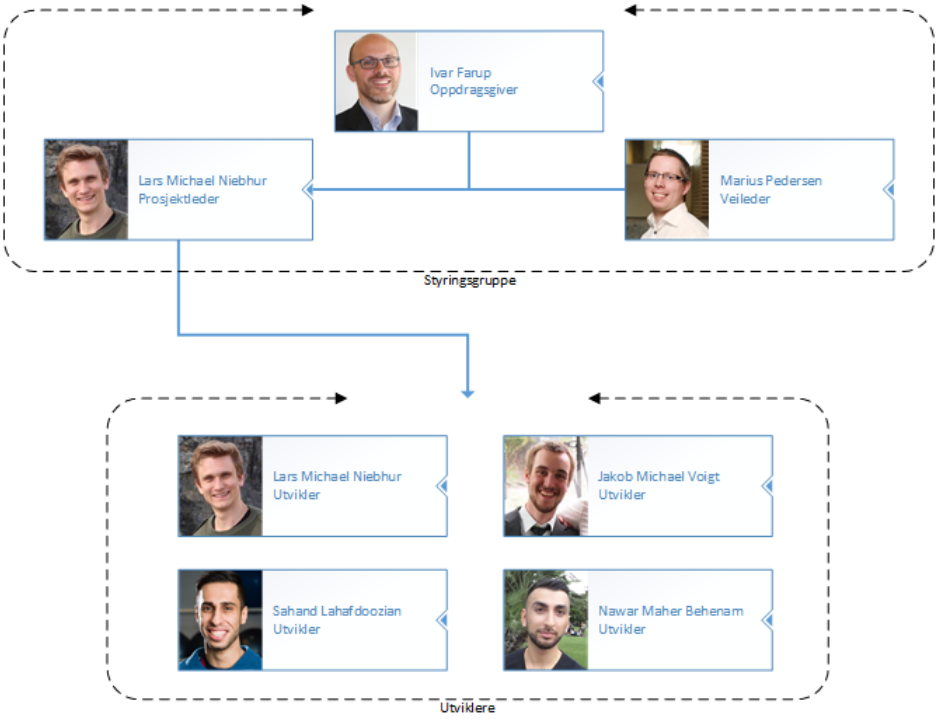
- Gamutoverflater: Modulen skal kunne beregne gamuter utfra bilder, ICC-profiler[4] og CSV-filer[5]. Når gamuten er beregnet skal den lagres og representeres som et objekt. Biblioteket skal kunne visualisere gamutens overflate som et 3D objekt.
- GMA: GMAene HPminde og SGCK skal implementeres gjennom bruk av kjerneoperasjonene. Dersom utviklerene får tid kan flere GMAer implementeres.
- Kjerneoperasjoner: Biblioteket skal tilrettelegge et sett med kjerneoperasjoner, se punktlisten under, på en slik måte at de kan settes sammen for å konstruere GMAer.
    - Ha funksjonalitet som lar brukeren kartlegge gamuter utfra bilder, ICC-profiler og CSV-filer.
    - Lagre kartlagt informasjon om gamuter.
    - La brukeren visualisere gamuten som et 3D objekt, ved bruk av vektorgrafikk.
    - Tilby et grensesnitt som kan utføre følgende kjerneoperasjoner på et sett med fargedatapunkter.
        - Svare på om et fargedatapunkt er innenfor eller utenfor en gitt gamut.
        - Finne et fargedatapunkts nærmeste punkt i en gitt gamut.
        - Finne avstanden fra et fargedatapunkt til gamutoverflaten, målt i et spesifisert fargerom.
        - Finne fargedatapunktets nærmeste punkt innenfor et gitt plan og gamuten. Disse planene skal kunne være plan parallelle med to koordinat akser i et kartesisk rom, eller et plan med en konstant vinkel ved bruk av sylinderkoordinater.

- Finne fargedatapunktets nærmeste punkt i en gitt gamut og på en gitt linje. Linjen skal kunne være en linje mellom to valgte punkter.
- Finne sentrum til en gitt gamut.
- Finne fargedatapunktets nærmeste punkt langs cuspen.
- Finne lysheten til cuspen.
- Sigmoidal lightness mapping.

1.4 Prosjektorganisering

Ansvarsforhold og roller

Lars Michael Niebhur, Jakob Michael Voigt, Nawar Maher Behenam og Sahand Lahafdoozian er prosjektgruppens utviklere. I tillegg til utviklerrollen er Lars prosjektleder, og Jakob referent. Ivar Farup er prosjektets oppdragsgiver, og har særskilt ansvar for å være en ressurs for teknisk kompetanse innen oppgavens fagfelt. Marius Pedersen er prosjektgruppens veileder, og har ansvar for generell veiledning av bacheloroppgaven. Han har også god kompetanse innen fargebildeteknologi. Se Avsnitt 1.7.3 for roller definert av utviklingsmodellen. I Figur 1 vises et organisasjonskart av prosjektgruppen.



Figur 1: Prosjektets organisering delt i styringsgruppe og utviklere.

Rutiner og regler i gruppen

Utviklerne har utarbeidet noen rutiner og regler som medlemmene skal følge. Reglene tar for seg blant annet hva som legges i feriedag, og fravær samt hvordan vi behandler eventuelle sykefravær.

Timelister

Alle utviklere i prosjektgruppen skal føre timer i hver sitt dokument se Vedlegg R.

Forventet arbeidsinnsats

Får å måle arbeidsinnsatsen til utviklerne, er det besluttet et innputt basert mål og et output basert mål [6].

- Input: Utviklerne skal arbeide minst 30 timer i gjennomsnitt per uke, og skal aldri arbeide mindre en halvparten av den forventede arbeidsmengden.
- Output: Dersom en utvikler en uke ikke ferdigstiller oppgaver tilsvarende mer enn 25 timer estimert tidsbruk skal det skriftlig rapporteres hvorfor. Gruppen kan kreve at vedkommende ferdigstiller mer arbeid.

Sykdom, fravær og feriedager

Ved sykdom trekkes det ifra 3 timer per sykedag på antall forventede timer. Dersom en utvikler ikke kan møte en dag der det er planlagt at gruppen skal arbeide fra samme lokasjon, skal dette meldes inn minst én dag før til prosjektleder. I tillegg til planlagt ferie 8. april til 17. april, disponerer utviklerne enkeltvis 3 feriedager.

1.5 Målgruppe

Prosjektet har to målgrupper. Den første er fagpersoner som ønsker en introduksjon eller mer kunnskap innen fargevitenskap og metoder innen gamut mapping. Den andre målgruppen er forskere som ønsker å benytte colour biblioteket og gamut modulen til forskning innen GMAer.

1.5.1 Målgruppe rapport

Vi ønsker å skrive en rapport som er forståelig og interessant å lese for de som for de som har lite eller ingen kunnskap for fagområdet, samt de som er godt kjent med faget. Derfor har vi valgt å skrive på et generelt høyere nivå, med en implementasjon av en teoridel for lesere som er nye til fagområdet. Teoridelen er lettleselig og forbereder leseren med manglende forkunnskaper til de deler av rapporten som krever en mer omfattende matematisk og faglig forståelse.

1.5.2 Målgruppe colour bibliotek

Målgruppen til den leverte koden er i utgangspunktet oppdragsgiver. Derunder er fargelaben en viktig målgruppe ettersom dens forskere aktivt kan benytte biblioteket i deres arbeid rundt utvikling av nye GMAer. Da biblioteket er lisensiert av GPL-3.0 og dermed er tilgjengelig for alle som ønsker å benytte hele eller deler av biblioteket er de også en indirekte målgruppe.

1.6 Utviklerne

Vi er en gruppe bestående av fire dataingeniør studenter ved NTNU Gjøvik. Vi har litt forskjellige fordypningsfag, og derfor er det noen som har større kompetanse innenfor matematikk enn programmering. Variasjon i fagkunnskaper har vært svært nyttig for samarbeidet i prosjektet.

Vi har benyttet Python i et mindre prosjekt og noen oppgaver fra et fordypningfag som alle i gruppen har deltatt i. Dette har gitt oss en innføring i Python som et språk og rundt utvikling metodikken.

En stor utfordring for oss har vært manglende erfaring med så store og omfattende

prosjekter innenfor Python og fargegamuter. Fagfeltet er nytt for oss og det er dermed blitt brukt mye tid å sette seg inn i begreper og konsepter innen fargebehandling og forskningen rundt dette.

Algoritmene og metodene som skal bli implementert i biblioteket krever komplisert matematikk som vi kun delvis har kjennskap til fra før. Alle i gruppen har bestått faget matematikk 2, én av fire har også fordypning i faget matematikk 3. En stor del av matematikken i prosjektet er basert på matematikk 3. Det er noe som har vært utfordrende for oss som gruppe og har krevd mye tid for den enkelte for å sette seg inn i matematikken og for kunne løse problemene. Mye av matematikken og flere av fagbegrepene som blir brukt i dette prosjektet er helt nytt for oss. Dette utgjør en signifikant del av læringsutbytte.

1.7 Utviklingsmodell

Vi presenterer vårt valg av utviklingsmodell, hvilke modifiseringer vi har valgt å gjøre og de roller som inngår i modellen.

1.7.1 Valg av utviklingsmodell

Prosjektgruppen har valgt å benytte Scrum som utviklingsmodell. For en grundig vurderingsprosess se forrapporten Vedlegg B avsnitt 4.

1.7.2 Forklaring av vår utviklingsmodell

Scrum var den foretrekkende utviklingsmodellen for hele prosjektgruppen. Vi har valgt å innsnevre og modifisere utviklingsmodellen noe utfra våre behov og ønsker. Innsnevringer er hovedsakelig gjort for å spare tid i utviklingen. Modifiseringer er gjort for å ta med elementer fra andre utviklingsmodeller som passet spesielt godt.

For oppgaver i produkt backlog(PBI) som blir registrert i Jira[7], er det utarbeidet en Definition of done(DoD)[8]. Denne er vedlagt (Vedlegg H) og gjelder kun for PBIer som krever skrevene kodelinjer. For de øvrige PBIene benyttes kun tolkningen av beskrivingen gitt i JIRA. Tolkningen, og DoDen dersom PBIen krever kodelinjer, brukes som en sjekkeliste for programmeringspar før en PBI flyttes til review. En utvikler fra et annet programmeringspar forholder seg til det samme da han inspiserer og flytter fra review til done.

Sprinter vil ha varighet på ti arbeidsdager med noen avvik, se GANTT diagrammet Vedlegg D. Vi har valgt en litt kortere sprintlengde for å kunne få ofte innspill fra oppdragsgiver gjennom review- og planning møter. Dette synes utviklerne passer godt, ettersom de har lite erfaring. Veileder vil også bli brukt aktivt, og det vil derfor holdes flere møter enn hva scrum i utgangspunktet tilrettelegger for. Disse ekstra møtene holdes i ukene som vi ikke har review/planning møter, og brukes som veiledningsmøter for rapportskrivning og andre spørsmål angående fargevitenskap.

Det er blitt besluttet at parprogrammering skal benyttes. Med parprogrammering ønsker vi å oppnå at utviklerne kommer raskere i gang med kompleske oppgaver og skriver mer effektiv og veldokumentert kode. Bruken av parprogrammering evalueres underveis, og kan fjernes dersom utviklerne uttrykker ønske om det.

Det er kun utviklerne som er med på retrospective møter. Tilbakemeldinger på prosessen fra oppdragsgiver og veileder kan gis i veiledningsmøtene. Review- og planning

møter legges til samme møte annenhver tirsdag.

Vi har under Avsnitt 12 beskrevet i detalj hvordan vi har benyttet oss av de scrum spesifikke møtene. Vi har også reflektert over bruken av scrum og hvordan vi har fordelt arbeidsoppgaver før og under sprinter.

1.7.3 Roller i valgt utviklingsmodell

Vi legger frem de scrum roller som inngår i prosjektet og hva deres oppgaver er.

Scrum master

Prosjektleder får rollen som scrum master. Han har ansvaret for å fordele arbeidsoppgaver og godkjenne kvalitetskravene som definert i Avsnitt 12. Scrum master har også hovedansvar for kommunikasjon mellom oppdragsgiver, veileder og utviklere. En viktig del av oppgavene til scrum master er å motivere utviklerne og lede diverse møter som retrospektive møter.

Produkt eier

Produkteier har ansvaret for å fylle backloggen i JIRA slik han mener er mest hensiktsmessig, og prioritere sprint backlogg etter estimering.

Utvikler

Utviklernes oppgave er å utarbeide alt av kode og dokumentasjon, samt skrive bacheloroppgave og holde presentasjon.

1.8 Rapport og layout

Rapporten følger NTNU i Gjøvik sin latex mal for bacheloroppgaver, utarbeidet av Simon McCallum og Ivar Farup [9]. Denne malen har vi importert inn i skrive verktøyet sharelatex som vi benytter til rapportskriving. I rapporten vil vi referere til kapitler og seksjoner med henholdsvis navnet på kapittelet og seksjonen. Terminologien punkt vil bli benyttet for punktlister eller andre listformer. Vi vil også referere til vedlegg som er vedlagt sist i rapporten, det er ledige sider mellom hvert vedlegg som NTNU malen foreslår og som er vanlig for forskningsbaserte rapporter.

1. **Innledning:** Innledningen til prosjektoppgaven omtaler blant annet oppgaven, organisering og valg av utviklingsmodell.

2. **Teori:** Generell teori om forkunnskaper leseren burde ha for å forstå hovedkapittelet.

3. **Oversikt over oppgaver:** I dette kapittelet gir vi innsikt i alle utviklingsoppgaver og presenterer strukturen og designet rundt hvordan vi har beskrevet de enkelte oppgaver.

4-11 Utvalgte oppgaver: Her presenterer vi utvalgte programmerings oppgaver vi har løst gjennom prosjektet.

12. **Kvalitetssikring:** Kvalitetssikrende tiltak vi har gjort under utvikling og dokumentering av bacheloroppgaven.

13. **Design og struktur:** Under dette kapittelet gir vi innblikk i hvordan biblioteket er strukturert som helhet og hvordan vi har valgt å designe modulene vi har opprettet.

14. **Avslutning:** Her gjøres avsluttende konklusjoner og refleksjoner. Vi drøfter også videre arbeid med oppgaven og mulighet for fremtidige bacheloroppgaver.

15. **Konklusjon:** Et konkluderende kapittel der vi drøter om vi har møtt de satte mål for bacheloroppgaven.

Bibliografi: Alle kilder vi har benyttet til under prosjektrapporten, inkluderer litteratursøk, nettsider, linker til verktøy og annet .

Vedlegg: Ordforklaring(C), forprosjekt(B), brukermanual(F), produkt backlog(E) o.l.

Vi har tatt strukturvalg i noen utvalgte kapitler og seksjoner disse har vi forklart nærmere under de aktuelle kapitlene. Strukturerende grep ble gjort for å gi leseren en mer ryddig og sammenhengende leseropplevelse.

2 Teori

I dette kapittelet vil vi forsøke å gi leseren den nødvendige teoretiske kompetansen for å forstå resten av rapporten. Det vil hovedsakelig dreie seg om en avgrenset del av fargevitenskap, samt en redegjørelse for det pre-eksisterende innholdet i colour biblioteket. Avslutningsvis presenteres en definisjonsliste med ord som går igjen flere steder i rapporten.

2.1 Farger og fargerom

For å uttrykke farger på en standardisert måte, er det utviklet et stort antall fargerom som på hver sin måte representerer fargeverdier. I dette delkapittelet vil vi først forklare de ulike attributtene som brukes for å definere farger, for deretter å forklare hvordan disse attributtene relateres til fargerom. Til slutt gis et par eksempler på fargerom.

2.1.1 Fargeattributter

Følgende attributter brukes for å definere farger [10, p.13]. Mer utfyllende informasjon kan leses i Gamut Color Mapping [10, p.13], men detaljenivået presentert der er ikke nødvendig for å forstå denne rapporten.

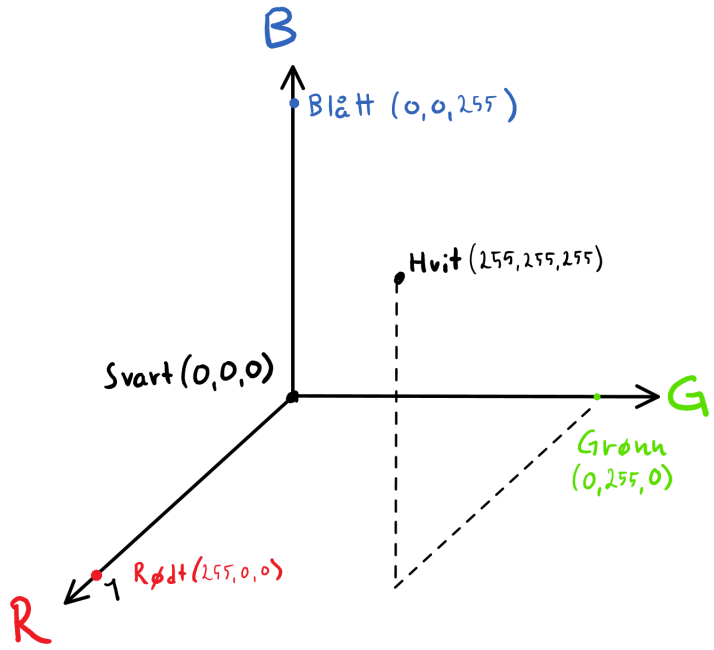
- **Fargetone:** Fargetonen beskriver fargens proporsjoner av rødt, gult, grønt og blått. Eksempelvis kan fargetonen til en appelsin beskrives som 60% rød, og 40% gul.
- **Lyshet:** Dette beskriver hvor lys fargen oppfattes å være. Vi kan bruke appelsineksempelet igjen. Deler av appelsinen som treffes av en tenkt lyskilde vil ha høyere lyshet enn deler av appelsinen som ligger i sin egen skygge.
- **Kroma:** Kroma representerer hvor sterkt fargetonen uttrykkes, eller hvor langt fargen er fra en gråfarge med samme lyshet. Dette ligner på fargemetthet, men forskjellen er at fargemetthet ser på hvor forskjellig fargen er fra svart, i motsetning til en gråfarge med samme lyshet. Et eksempel på reduksjon av kroma kan være et bilde som har blitt eksponert til lys over lengre tid, slikt at fargene oppfattes å være svekket i forhold til da det ble først ble laget.

2.1.2 Fargerom

Et fargerom kan visualiseres som et tredimensjonalt rom. Det finnes avvik fra dette, men de er ikke relevante for denne rapporten. Hver akse relaterer til et fargeattributt og de mest vanlige enhetene er rød, gul, blå, grønn og lyshet. Farger kan da defineres ut ifra et punkts posisjon i rommet og hvilket fargerom som benyttes. Gjennomgående bruker vi relativ kolorimetri, altså at alle fargene et relative til hvitfargen. Hva som er 100% hvitt vil variere ut ifra mediet fargene fremstilles med.

2.1.3 Eksempler på fargerom

I Figur 2 vises en illustrasjon av sRGB fargerommet, med noen utvalgte punkter tegnet inn. Fargetonen defineres ut ifra forholdet mellom koordinatenes verdi. Vi ser at punktet Rødt(255,0,0) som er en ren rødfarge, ligger på rødaksen. Lysheten i dette fargerom-



Figur 2: Illustrasjon av sRGB fargerommet, med noen utvalgte fargepunkter.

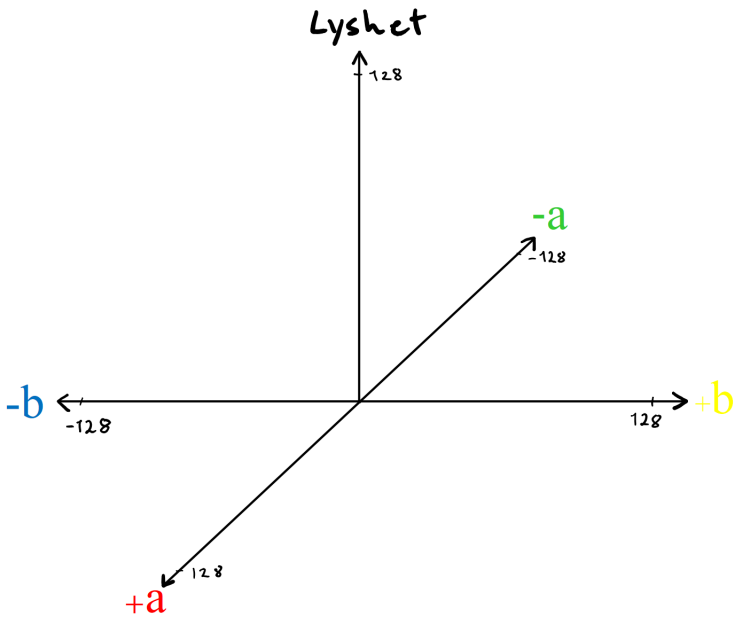
met er definert slik at mørkere farger er nærmere (0,0,0) og lysere farger er nærmere(255,255,255). Mørk oransje har koordinater (255,140,0)[11]

Neste eksempel er fargerommet CIELAB, illustrert i Figur 3. I dette fargerommet kalles aksene *L*, *a* og b. L aksen går fra 0 til 128 og representerer fargens lyshetsverdi. Høyere verdier betyr lysere farger. *a* går fra -128 til 128 der negative tall indikerer grønt, og positive indikerer rødt. *b* aksen går fra -128 til 128, og her indikerer negative verdier blått, og positive verdier gult. For å forklare fargeattributtene i dette fargerommet kan vi tenke på sylinderkoordinater. Vinkelen er fargetonen, radien er kroma, og z-aksen er lyshet. Koordinatene har er da på formatet (L-verdi, a-verdi, b-verdi). Fargene fra forrige eksempler har følgende koordinater i CIELAB:

- **Blå:** (32, 79, -107)
- **Gul:** (97, -21, 94)
- **Rød:** (53, 80, 67)
- **Hvitt:** (100, 0, 0)
- **Svart:** (0, 0, 0)
- **Mørk oransje:** (69, 36, 75)

2.2 Gamut

En gamut eller en fargegamut, er et sett med farger i et gitt fargerom. Gamuter kan visualiseres som et polyeder i et tredimensjonalt rom definert av alle ytterpunktene i settet av farger. Det finnes flere metoder for å beregne disse ytterpunktene. Ofte brukte metoder er *Convex hull*, *Modified convex hull*, *Segment maxima*, og *Alpha shapes*[10]. Gamutene regnes som kontinuerlig, det betyr at alle punkter innenfor gamutens overflate er en del av gamuten. Hva en gamut representer kan variere. De kan representere hvilke farger



Figur 3: Illustrasjon av CIELAB fargerommet.

som er definert i et fargerom. De kan også brukes for å representere hvilke farger som er tilstede i et bilde, og gamuten kalles gjerne da en bildegamut. Enheter som printere og skjermer har en enhetsgamut som beskriver hvilke farger enheten kan gjengi.

Alle enheter har naturligvis ikke alltid tilgang til de samme fargene, og for å tilpasse et bilde en ny gamut benyttes *gamut mapping*. Ján Morovič skriver i sin bok *Color gamut mapping* følgende om gamut [10]

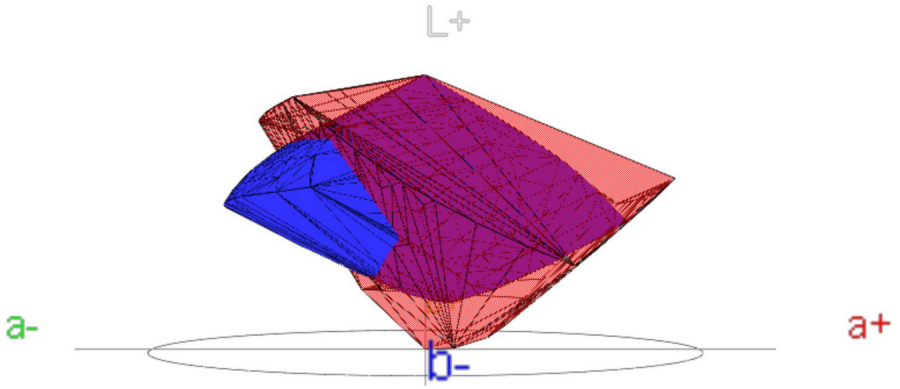
> mapping of the colour-space coordinates of the elements of a source image to colour-space coordinates of the elements of a reproduction to compensate for differences in the source and output medium colour gamut capability.

I Figur 4 ser vi en enhetsgamut illustrert med rødt, og en bildegamut illustrert med blått. Enheten har ikke tilgang til alle bildets farger, og det må først gjennomføres gamut mapping før bildet kan vises på den nye enheten. Det finnes mange metoder for dette, disse kalles *Gamut mapping algoritmer*. Et eksempel på en velkjent algoritme er *Hue-angle preserving minimum*Δ *E Clipping* (HPminDe)[10].

2.3 Colour biblioteket

Colour pakken inneholder diverse moduler Figur 5. Inni hver enkel modul inneholder det Python kode som oppdragsgiver har jobbet med. Vi leste gjennom hver modul for å få godt forståelse for hva hver modul inneholder.

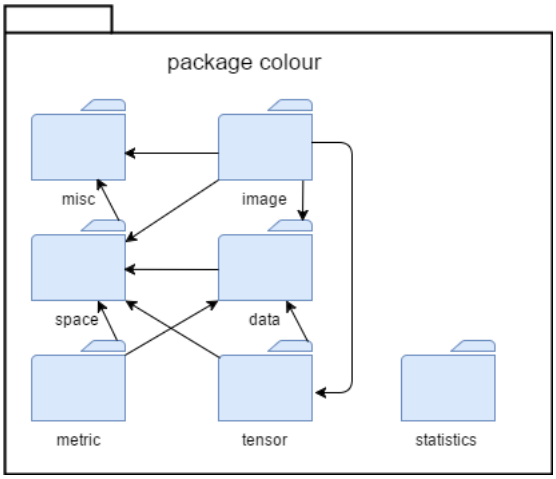
I colour biblioteket [1] er det lagt ved en readme(wiki) fil der det står hvilke moduler som er tilgjengelig, og en forklaring på hvordan ta i bruk modulene og den generelle strukturen i biblioteket. I denne readmefilen er det referert til en artikkel [2] som oppdragsgiver har publisert i tidsskriftet PeerJ Computer Science [12]. I denne artikkelen beskriver oppdragsgiver hvordan hvert moduler er bygd opp og avhengighet mellom



Figur 4: To gamuter illustrert i CIELAB. Generert med ICC3D.

modulene samt diverse klassediagrammer.

Mange av testene som var skrevet for modulene i biblioteket var utdaterte og ikke lenger brukbare. Testene var samlet i en modul og var ustrukturert med deler av testene liggende i samme modul som klassene de testet.



Figur 5: Figuren viser de indre avhengigheter i colour pakken.

Data: Modulen inneholder klasser som holder fargedata i forskjellige fargerom og former. Vi benytter klasse med å hente data punktene i ndarray og sette dataene i ønsket fargerom.

Image: Image modulen er underklasse av data modulen for bildeformede data.

Metrics: Modulen inneholder det beregninger mellom to objekter i samme dimensjon.

Misc: Misc modulen inneholder diverse støttefunksjoner. For eksempel Auxiliary functions og finite differences.

Space: Space modulens oppgave er å transformere fra et fargerom til et annet fargerom (eks. fra XYZ til CIELAB fargerom).

Statistics: Statistics modulen inneholder metoder for beregning av ulike statistikker for colour metric data.

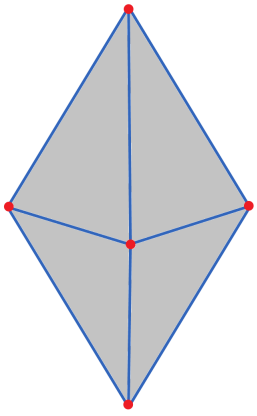
Tensor: Tensor modulen har metoder for beregning av metric tensorer som tilsvarer beregningene i metric pakken.

Vi benytter ingen av metodene direkte som finnes i image, metric, misc, space, statistics og tensor modulen i vårt gamut modul siden vi ikke ser nytten av dette. Alt arbeid blir gjort av data modulen når vi benytter den for å hente og sette data punkter.

2.4 Definisjoner

I dette avsnittet defineres en liste med ord som inngår i rapporten. Ord som kun brukes i en avgrenset del defineres lokalt.

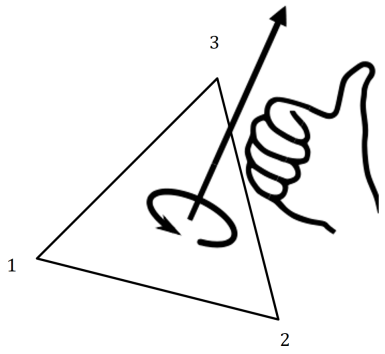
- **Polytop:** Et polytop er en geometrisk figur med flate sider, og finnes i alle antall dimensjoner. Figurene kan skrives på formatet *n-polytop* og da betegner n dimensjonen til polytopet. Et polytop med én dimensjon er en linje, og kalles et 1-polytop. Et polytop med to dimensjoner er en polygon og kalles et 2-polytop. Et polytop med 3 dimensjoner er et polyeder og betegnes 3-polytop.[13]. Et 0-polytop er et punkt.
- **Polyeder:** Et polyeder er et geometrisk objekt med plane overflater og rette kanter. Ordet er variasjon av polyhedron [14]. Overflaten til et polyeder er altså sammensatt av flate polygoner. Figur 6 er et eksempel på et polyeder.
- **Hjørne:** Et hjørne er i geometri et punkt ytterst på en geometrisk figur. I en todimensjonal figur møtes to kanter i hvert hjørne. I et tredimensjonalt legeme møtes tre eller flere kanter og sider i hvert hjørne. Det er alltid like mange kanter som sider [15]. Dette er illustrert rødt i Figur 6.
- **Kant:** En kant er linjestykket mellom to hjørner. Illustrert i blått i Figur 6.
- **Fasett:** En av de plane overflatene i et polyeder. I denne rapporten brukes det eksklusivt om trekanter i R^3. Overflaten til gamutene vi behandler i denne rapporten er konstruert av slike fasetter. Illustrert i grått i Figur 6.



Figur 6: Illustrasjon av et enkelt polyeder. Hjørner(rødt), kanter(Blått) og fasetter(Grått).

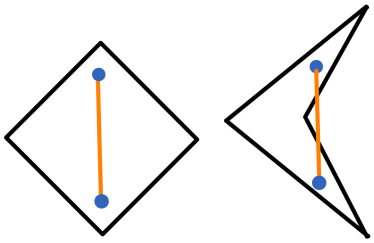
- **Orientering:** For å definere hvilken side av en fasett som tilhører utsiden av gamuten, bruker vi høyrehåndsregelen[16]. Fasettens normal vektor peker likt som tom-

melen, og peker ut av gamuten. Kryssprodukt følger også denne reglen. se Figur 7 for illustrasjon.



Figur 7: Illustrasjon av høyrehåndsregelen. Tallene indikerer hjørnenes rekkefølge.

- **Lokalt origo:** S er et sett med punkter i R^3. S(L) er det punktet vi ønsker å gjøre til lokal origo for S. V er vektoren fra origo til S(L). Ved å subtrahere V fra alle punkter i S vil S(L) ende opp i orgio, da S(L) - S(L) = O (0,0,0). Alle punkter i S vil fortsatt være plassert likt iforhold til hverandre da alle punkter flyttes like mye.
- **Randen:** Randen brukes her om kantene til en fasett, altså de aller ytterste punktene som skiller mellom punkter som er inne i trekanten og utenfor.
- **konveks, ikke-konveks:** Et polytop er konvekst dersom det mellom alle punkter i polytopet kan trekkes en rettlinje som ikke går på utsiden av polytopet. Se Figur 8 for illustrasjon.



Figur 8: Konveks til venste, ikke-konveks til høyre.

- **ndarray:** Et n-dimensjonalt matriseformat for lagring av da utgikk av numpy[17].
- **Normalvektor(3x3):** En vektor som står normalt på et plan. Det vil si at det er en 90 graders vinkel mellom normalvektoren og planet, uansett hvordan du måler vinkelen. I denne rapporten har alle normalvektorene lengde 1. Ofte kalles slike normalvektorer *normalenhetsvektor*, men for leselighets skyld bruker vi kun normalvektor.

3 Oversikt over oppgaver

Under utviklingen har vi benyttet JIRA [7] som organiseringsverktøy for oppgaver og fordeling av arbeid. I Jira blir alle oppgaver tildelt navn og nummer. Prosjekt vårt heter Python Gamut Library, og derfor har alle oppgavene fått tildelt navnet PGL. Denne forkortelsen vil bli benyttet svært mye i det videre forløpet av rapporten, og refererer altså til en utviklings oppgave.

3.1 PGL arbeidsoppgaver

I denne seksjonen har vi listet alle PGLer vi har arbeidet med og en kortfattet forklaring rundt innholdet i oppgaven. Grunnen til at oppgavene starter på PGL-32 er fordi vi har benyttet Jira til forrapporten og hadde en del oppgaver forbundet med den.

PGL-32 Gamut modul:
Opprettelsen av en gamut modul i biblioteket, slik at ved import av colour biblioteket skal gamut modulen lastes inn på samme måte som de øvrige modulene. Det skal også opprettes selve gamut klasse i denne modulen.

PGL-33 Beregne convex hull-gamut:
En metode for å opprette en konveks hull representasjon av en gamut i et oppgitt fargerom. Metoden er en konstruktor til gamut klassen.

PGL-34 Plot metode:
Implementer en plot metode som setter inn alle hjørner og fasetter til en gamut i et gitt fargerom og dermed visualisere gamuten som et 3D objekt.

PGL-35 Modified convex hull:
Alternativ konstruktor til gamut klassen, skal ta hensyn til konkavheter i en polyhedra. Oppretter en representasjon av en konkav gamut i et ønsket fargerom.

PGL-36 FeitoTorres:
Metode som avgjør om fargepunkter er innenfor eller utenfor en enhetsgamut. Denne metoden benyttes i alle GMAer aktuelle for dette prosjektet.

PGL-37 Clip Nearest:
Metoden skal ta imot datapunkter som er utenfor en enhetsgamut. Ved beregning skal metoden returnere nærmeste punkt på overflaten i en hvilken som helst retning mot en gamut i 3D.

PGL-39 Komprimerer til gamut i én dimensjon:
Metoden skal kunne komprimere alle fargepunkter lineært gitt en akse, og returnere et data objekt med de komprimerte punktene. Punktene skal returneres i samme format som punktene ble sendt med.

PGL-40 Nærmeste punkt i en retning:
Metoden skal returnere nærmeste punkter på overflaten langs en rettlinje fra hvert enkelt punkt til et gitt referansepunkt. En rettlinje mellom datapunkter utenfor gamut objektet og referansepunktet som er innenfor gamut objekte.

PGL-45 Nærmeste punkt på plan:
Metoden skal mappe alle fargepunkter til gamutens overflate gitt en vinkel definert av en akse. Aksen vil sammen med punktet lage et plan som benyttes til å finne nærmeste punkt i.

PGL-48 HPminde:
Metoden tar inn fargepunkter og mapper alle farger som ligger utenfor en enhetsgamut, til nærmeste farge med samme fargetone i fargerommet CIELAB.

PGL-54 Gamut testmodul:
Opprette en testmodul som inneholder alle tester som benyttes til å teste colour bibliotekets moduler. Det skal opprettes separate filer for hver modul i colour.

PGL-57 Optimalisering:
Undersøk om Cython eller Numba kan implementeres for å øke hurtigheten til metoder, og hvor mye tid som kan spares.

PGL-58 Flere punkter i clip nearest:
Utvid PGL-40 med støtte for mer enn ett punkt. Punktene vil bli medsendt i et colour.data.Data objekt og har ukjent form.

PGL-59 Traversering:
Test om PGL-36 kan optimaliseres ved å benytte en annen metode enn traversering. Det må tas hensyn til en problemstilling der fargepunkter kommer i form av data med ukjent dimensjon.

PGL-60 Refactoring:
Gå gjennom alle skreven kode og gjør optimaliseringer der det er mulig. Standardiser variabler og fjern kode som ikke blir brukt. Merk også metoder som ikke skal benyttes av bruker direkte og gjennomgå kommentering.

PGL-61 minDE:
Metoden tar inn fargepunkter og mapper alle farger som ligger utenfor en enhetsgamut til nærmeste farge i fargerommet CIELAB.

3.2 Struktur og design

For bedre oversikt har vi strukturert hver PGL rapport som følgende:

1. Teori
2. Kravspesifikasjon
   2.1. Kravspesifikasjon fra oppdragsgiver
   2.2. Tolkning
3. Arbeidsprosess
4. Resultat

5. Testing

I teori har vi valgt å forklare matematikken som behøves for å forstå PGLen. Vi kan da enkelt referere til definisjonene i den videre teksten.

Under kravspesifikasjon har vi først lagt inn den tekstlige beskrivelsen oppdragsgiver har gitt oss i Jira på oppgaven. Vi har så utarbeidet en tolkning som vi har brukt som grunnlag under arbeidet med oppgaven og utviklingen.

Under arbeidsprosess viser vi til litteratursøk, lesing av artikler og hvilke metoder vi har funnet og testet. Videre beskriver vi nærmere hvordan vi gikk frem for å løse oppgaven. Valg og diskusjoner underveis i prosessen vil belyses her.

I resultat presenterer vi den endelige løsningen på oppgaven. Vi har utarbeidet skisser og figurer der vi ønsker å grafisk vise et resultat eller når vi forklarer de mer komplekse metoder.

Testing forklarer hvordan vi har testet PGLen for å sikre oss at den virker som ønsket og gir de resultater som er forventet.

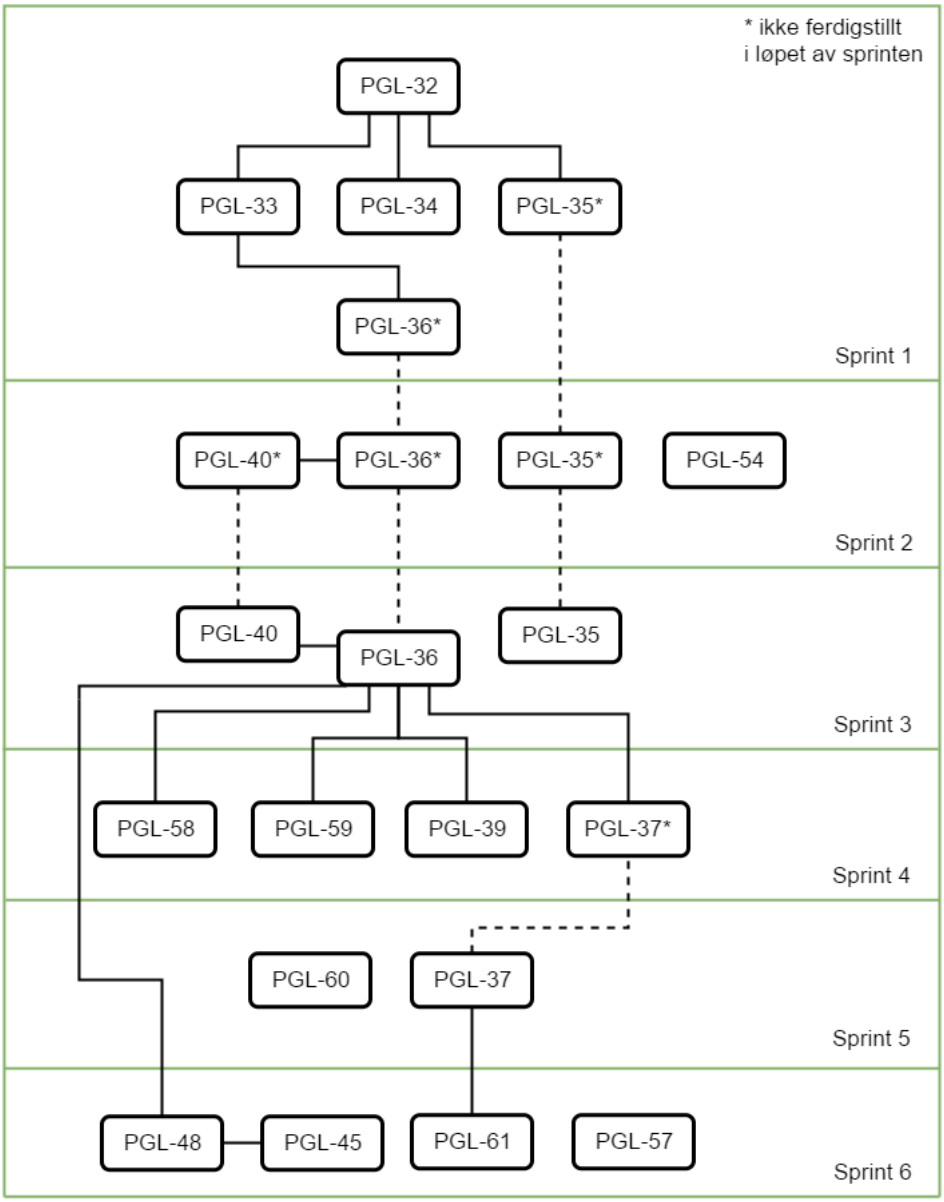
3.3 Utvalgte PGL rapporter

I de neste kapitelene presenterer vi utvalgte PGLer som vi har arbeidet med under utvikling. Utvalget av PGLer baserer seg på hva vi mener har vært de viktige i forhold til oppgavens videre gang, tidsbruk og kompleksitet. Dette er oppgavens hovedkapittel og skal gjenspeile den arbeidsmengden og arbeidsinnsatsen vi har lagt inn for å tilfredsstille oppdragsgiver sine krav og ønsker til den kodemessige siden av bachelorprosjekt.

I Figur 9 har vi laget en oversikt over de PGLer vi har valgt å skrive rapporter for. Bilde er delt i sprinter der sprint 1 er første utviklingssprint. Dette innebærer at sprint 1 i figuren er sprint 2 i GANTT diagrammet se Vedlegg D.

Avhengigheten er illustrert med heltrukne svarte linjer der en høyere nummerert PGL er avhengig av en lavere nummerert. Med avhengighet menes at en metode behøver/benytter beregninger eller informasjon som gis av en annen metode.

I figuren er det markert med stjerne der en PGL ikke ble ferdigstilt i løpet av den aktuelle sprinten. Stiplet svart linje er så benyttet for å vise at PGLen ble prioritert med i den påfølgende sprinten.



Figur 9: PGL struktur og avhengighet. For beskrivelse av hver PGL se Avsnitt 3.1.

4 PGL-34 Plot metode

Implementer en plot metode som setter inn alle hjørner og fasetter til en gamut i et gitt fargerom og dermed visualisere gamuten som et 3D objekt.

4.1 Kravspesifikasjon

I dette avsnittet viser vi først oppgaven som den var gitt av oppdragsgiver og så hvordan vi tolket oppgaven.

4.1.1 Kravspesifikasjon fra oppdragsgiver

Et gamut objekt inneholder mye (punkter, kanter, etc.), men ikke nødvendigvis i et format som er direkte kompatibelt med visualiseringsfunksjonene i matplotlib. Ønsker en metode i gamut-klassen som returnerer en representasjon som kan sendes mer eller mindre direkte videre til matplotlibs funksjoner. Pseudokode for syntaks Listing 4.1

Listing 4.1: Der g vil være gamut objektet og alle punkter blir lagt til plot metode.

```
1    g = colour.gamut.Gamut(...)
2    plt.plot_3dsurfaceXXX(g.get_surface(colour.space.XXX))
```

4.1.2 Tolkning

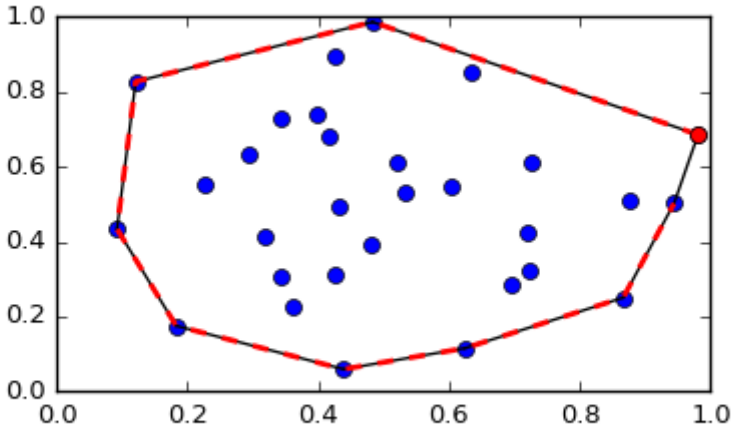
Det skal skrives en metode under gamut modul klassen som skal visualisere et gamut objekt. Dette kan gjøres ved å hente alle hjørnene i riktig fargerom og plotte dem i en 3D figur. Deretter skal det tegnes en linje mellom alle hjørnene og returnerer en illustrasjon av en gamut objekt i 3D. Det må lages et aksepunkt system som passer for selve gamut objektet. Metoden må kunne håndtere å at imot en akse og et fargerom.

Akse figuren må opprettes ved hjelp av matplotlib sin funksjon kalt figure() og deretter sett figuren til å være en 3D figur. Punktene hentes fra colour.data.Data objektet i riktig fargerom.

4.2 Arbeidsprosess

Gruppen fikk tips av oppdragsgiver om at det fantes metoder som kan visualisere en gamut objekt. Det ble gjort grundig litteratursøk på alle plot metoder som finnes i matplotlib, og de aktuelle er surface og tri-surface plot metode [18]. Disse to ble valgt fordi de kan ha data verdier i xyz-akse og setter på fargekontraster. Testing av metodene viste at matplot_surface ikke kunne brukes grunnen er at vi sendt med x, y og z akse, men matplot_surface krever en ekstra z akse som ikke blir sendt med, dermed er det ikke mulig å benytte matplot_surface metoden. Vi får ikke benyttet matplot_trisurf på grunn av at den ikke bruker våre gamuter, men bruker egne algoritmer som igjen endrer på formene og det medfører at gamuten ikke stemmer med det oppdragsgiver ønsker. Dette ble diskutert og bekreftet av oppdragsgiver. Med dette kom vi frem til at vi må implementere egen algoritme for å kunne visualisere en gamut objekt.

Vi startet med å lage egen metode som tar imot en akse og en fargedata. Videre vil metoden ha alle hjørner får å kunne sett punktene inn. Med litt forklaring av oppdragsgiver så må det implementere egen algoritme for å kunne visualisere en gamut objekt. Dette ble gjort ved å gå gjennom alle konveks hull fasetter sine hjørner og tegne linje i figuren. Figur 10 viser hvordan dette ser ut i 2D.



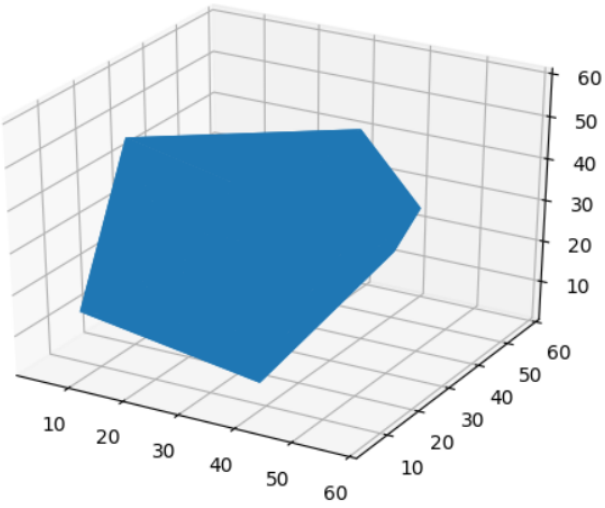
Figur 10: Den røde linjen viser randen til et polygon etter konveks beregning av punktenes hjørner.[19].

I tillegg til å lage plot metoden vil den ta imot en akse. Dermed må det implementeres en metode på hvordan aksene xyz skal settes. Metoden som ble implementert er enkelt ved at metoden henter alle x-ene, y-ene, z-ene sine verdier og sortere dem i en liste. Det ble diskutert i gruppen hvordan gå frem for å sortere akse verdiene. Vi startet med å bruke for-løkke for å sortere aksene, men etter ha tatt en prat med oppdragsgiver anbefalt han å bruke en funksjon som allerede finnes i Python som heter sort(). Fordelen med sort funksjonen er at det blir mindre kode og lesbart kode.

Første gang akse verdiene ble implementert, ble akse satt til å være faste verdier fra -100 til 100. Problemet var hvis gamut objektet var liten så ville den få store akse verdier og en liten gamut objekt på skjermen. Videre ble akse verdiene satt til å være eksakte verdien, men da ville gamut objektet legge seg inntil aksen. Med hjelp av oppdragsgiver på hvordan sette akse verdiene, mente han enkleste måten var om hver akse settes til å være like lang som 120% av sin respektive maks verdi. Eneste problemet er om akse verdien er 0, da vil gamut objektet legge seg inntil aksen.

4.3 Resultat

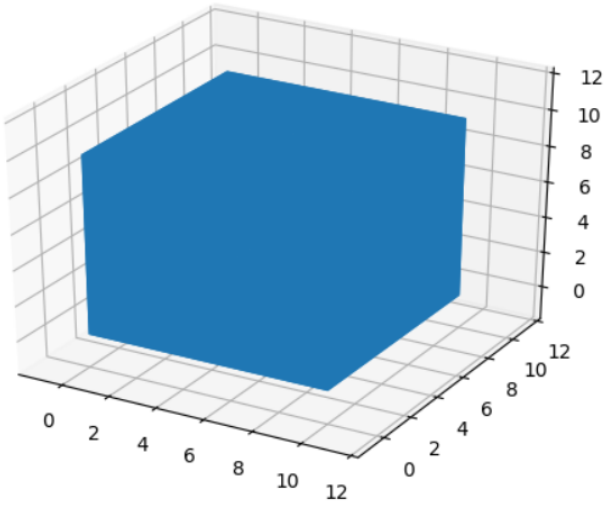
For å kunne visualisere gamut objektet må det brukes en matplotlib funksjon som heter show(). Vi lager en gamut objekt i et fargerom og sender den til metoden med aksen. Metoden vil da visualisere en gamut objektet som ønsket. Figur 11 viser gamut objektet i 3D.



Figur 11: Visualisering av et gamut objekt i sRGB fargerom (Polyhedron), ved bruk av gamut.plot_surface().

4.4 Testing

Vi testet plot metoden ved å lage punkter som ville visualisere en kube. Figur 12 viser at plot metoden fungerer som ønsket.



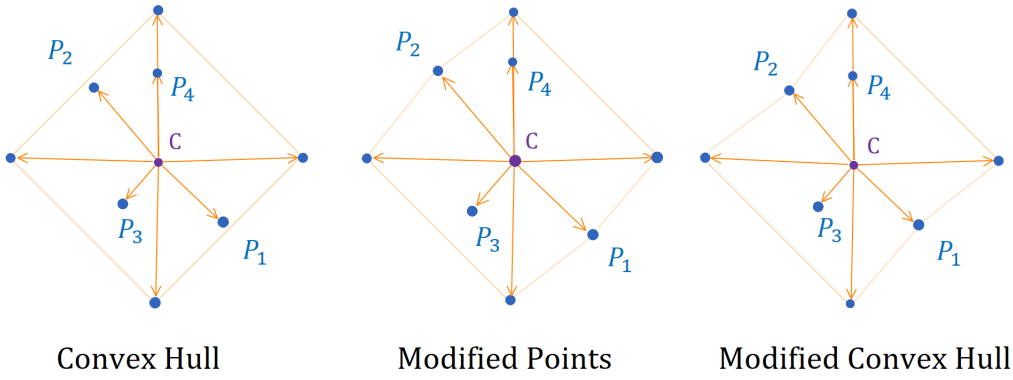
Figur 12: Visualisering av et gamut objekt i sRGB fargeromet (kube) ved bruk av gamut.plot_surface().

5 PGL-33 Beregne convex hull-gamut PGL-35 Modified Convex Hull

I dette kapittelet presenteres hovedsakelig arbeidet for PGL-35, men vi vil i resultat presentere gamut klassens konstruktor som et resultat av PLG-35, PGL-33 og endring gjort under PGL-36. Vi har valgt å ikke skrive et eget kapittel for PGL-33, da dette var en meget liten oppgave.

5.1 Teori

I teoridelen for dette kapittelet vil vi forklare konseptet til modified convex hull metoden. Se Figur 13 for en behjelpelig illustrasjon tilhørende denne forklaringen. Først er punktene i en umodefisert plassering, og vi ser at P_1 og P_3 ikke er hjørner i det konvekse hullet. I neste steg modifiseres punktene, og vi ser at P_1 og P_2 er hjørner for de modifiserte punktenes konvekse hull. Til slutt endres punktene tilbake til sine originale posisjon, men vi bevarer de samme hjørnene. Hullet tar da hensyn til noen konkavheter, men ikke alle. Punktene ble ikke modifisert nok til at P_3 ble et hjørnet, og det vil aldri være mulig for P_4 å bli et hjørne.



Figur 13: Illustrasjon av hvordan punkter endrer sin plassering realtivt til hverandre i modified convex hull metoden. C er senter. Punkter er illustrert i blått, noen utvalgte er navngitt.

5.2 Kravspesifikasjon

I dette avsnittet viser vi først oppgaven som den var gitt av oppdragsgiver og så hvordan vi tolket oppgaven.

5.2.1 Kravspesifikasjon fra oppdragsgiver

Alternativ konstruktor (eller en parameter til den eksisterende) for beregning av gamut vha. modified convex hull i henhold til Bala et al., se Bakkes artikkel om evaluering av gamut boundary descriptors. Skal ta fargerom for beregning av gamuten, gamma-verdien

til modified convex hull og senter for ekspansjonen som parametere. Pseudokode for syntaks Listing 5.1

Listing 5.1: Der d er colour.data.Data punkter og g er beregningen av gamut ved hjelp av modified convex hull.

```
1    d = colour.data.Data(...)
2    g = colour.gamut.Gamut(data, colour.space.XXX, gamma=0.2, center=XXX)
```

Dette innebærer også å finne en god måte å representere senter på, f.eks. vha. gitte konstanter. Vanligste sentere er:

- Midt på gråaksen (0.5, 0.5, 0.5) i RGB eller (50, 0, 0) i Lab
- Midt på den delen av gråaksen som er innenfor gamut
- I tyngdepunktet av convex hull-gamuten, eventuelt for iterativ forbedring

Ivar Farup skrev ny kommentar - 24/Jan/17 23:45:
Tror vi dropper siste definisjonen av senter, og eventuelt lager en egen sak på dette senere om det skulle være aktuelt. I stedet kan man ha muligheten for at brukeren selv oppgir et punkt i fargerommet som skal brukes som sentrum for ekspansjonen.

5.2.2 Tolkning

Denne oppgaven går ut på å lese oss opp på modified convex hull metoden, og implementere den. Vi må også ta vurdere hva som blir beste måten å strukturere løsningen på, altså om vi skal ha to konstruktorerer eller én.

5.3 Arbeidsprosses

For denne oppgaven valgte vi å splitte programmeringsparet. Det var flere grunner til dette. Ut ifra tidligere lesing og forklaring av oppdragsgiver så vi ikke på oppgaven som veldig kompleks også derfor mindre behov for å sette to utviklere på denne oppgaven. Samtidig var det på tide å integrere arbeidet vårt så langt med oppdragsgiver sin master branch PGL-54, og dette var også en oppgave vi ikke så behov for to personer for å løse. Derfor valgte vi å sette Jakob på PGL-35 og Lars på PGL-54(Avsnitt 3.1).

Tidligere i utviklingsprosessen har vi lest i Ján Morovič sin bok Color Gamut Mapping for å tilegne oss grunnleggende forståelse om fargevitenskap, gamuter og gamut mapping. Vi visste derfor at denne boken inneholdt en forklaring av modified convex hull metoden[10, s.146]. Etter grundig gjennomlesning sendte vi noen oppklarende spørsmål til oppdragsgiver per e-post. Når spørsmålene var besvart var jeg klar for å begynne implementasjon.

Med bakgrunn i forarbeid vi valgte å gjøre noen endringer til metoden Ján Morovič foreslår. Han foreslår å konvertere til spheriske koordinater, for så å endre deres tredje koordinat som da er radiusen. Vi kan enkelt regne ut radiusen ved bruk av numpy.linalg.norm og slipper da denne konverteringen. Formelen for endring av radius som foreslås er ny $radius = \alpha \cdot (Radius/Maksradius)^{\gamma}$, der maks radius er den største radiusen i datasettet. Vi endte opp med å bruke denne formelen ny $radius = radius^{\gamma}/radius$. Da alpha og maks radius er konstanter kan vi med litt algebra ende opp med $(\alpha \cdot Rmaks^{\gamma}) \cdot r^{\gamma}$, som forenkles til $K \cdot R^{\gamma}$. Hvilke punkter som blir ytterpunkter er uavhengig K, så den kan derfor strykes. Videre har vi valgt å dele på radiusen til det aktuelle punktet. Dette sørger for at alle radiusene blir mindre enn en, slik at vi ikke for unødvendige

store verdier å jobbe med.

Vi valgte å legge funksjonalitet tilhørende modified convex hull i en egen initialiseringsmetode, da konstruktoren allerede var satt opp slik at den bruker en initialiseringsmetode for å initialisere datene sine. Vi kan da enkelt tilkalle den korrekte metode ut ifra hva brukeren ønsker å bruke.

5.3.1 Resultat

I Listing 5.2 vises kontruktoren til gamut klassen. For å opprette et gamut objekt må alltid fargerom velges samt et sett med fargedatapunkter inneholdt i et colour.data objekt sendes med. Dersom modified convex hull metoden skal benyttes settes gamma til et tall ulikt null, 0.2 gir har vist seg å gi gode resultater [10, s.146]. Gamutens sentrum kan velges, men er dette ikke nødvendig. I initialize_convex_hull() Listing 5.3 generes et konveks hull objekt [20] fargedatapunktene konvertert til ønsket fargerom som parameter. Dette objektet lagres i *hull* medlemsvariabelen, og inneholder en liste med koordinatene til alle fargedatapunktene i hull.points. Indekser til gamutens hjørner legges i *vertices*. Indeksene til hvilke hjørner som tilhører samme fasett lagres i *simplices*. Ved bruk av initialize_modified_convex_hull() benyttes nesten samme fremgangsmåte, men punktene må først modifiseres før de sendes til konveks hull kontruktoren. Vi henter så ut indeksene før vi endrer hull.points tilbake til de originale punktene. På denne måten blir yter punkter satt i henhold til modified convex hull metoden beskrevet over. Til slutt fikses orientasjonen til alle fasette slik at deres normal vektor peker vekk fra gamutens senter med fix_orientation(). Dette var en endring som måtte tilføyes under utvikling av is_inside() da algoritmen baserer seg på propert orienterte polederoverfalter.

Listing 5.2: Utdrag av gamutenklassens konstruktor

```
1  def __init__(self, sp, points, gamma=1, center=None):
2
3          self.data = points       # The data points are stored in the original
               format. Use hull.
4          self.space = sp
5          self.hull = None         # Initialized by initialize_(modified)
               convex_hull
6          self.vertices = None     # Initialized by initialize_(modified)
               convex_hull
7          self.simplices = None    # Initialized by initialize_(modified)
               convex_hull
8          self.neighbors = None    # Initialized by initialize_(modified)
               convex_hull
9          self.center = None       # Initialized by initialize_(modified)
               convex_hull
10
11         if gamma == 1:
12             self.initialize_convex_hull()
13         else:
14             self.initialize_modified_convex_hull(gamma, center)
15         self.fix_orientation()
```

Listing 5.3: Utdrag av initialize_convex_hull()

```
1      def initialize_convex_hull(self):
2          # Calculate the convex hull
3          self.hull = spatial.ConvexHull(self.data.get_linear(self.space),
               qhull_options='QJ')
4          self.vertices = self.hull.vertices
5          self.simplices = self.hull.simplices
6          self.neighbors = self.hull.neighbors
```

```
7           if center is None:
8               self.center = self.center_of_mass(self.get_coordinates(self.vertices
                    ))   # If a center was provided, use it.
9           else:
10              self.center = center
```

Listing 5.4: Utdrag av initialize_modified_convex_hull()

```
1       def initialize_modified_convex_hull(self, gamma, center):
2           # Move all points so that 'center' is origin
3           n_data = self.data.get_linear(self.space)
4           n_data_backup = n_data                          # Save a copy of the
                points, unmodified.
5
6           if center is None:
7               self.center = self.center_of_mass(n_data)   # If a center was
                    provided, use it.
8           else:
9               self.center = center                        # If not, use the
                    geometric center as a default.
10
11          shifted = n_data - center                               # Make center
                the local origin
12          r = np.linalg.norm(shifted, axis=1, keepdims=True)      # Get the radius
                 of all points
13          n_data = shifted * (r ** gamma / r)                     # Modify the
                radius.
14
15          # Calculate the convex hull, with the modified radius's
16          self.hull = spatial.ConvexHull(n_data)      # Set indexes from modified
                points.
17          self.vertices = self.hull.vertices          # Set indexes from modified
                points.
18          self.simplices = self.hull.simplices        # Set indexes from modified
                points.
19          self.neighbors = self.hull.neighbors        # Set indexes from modified
                points.
20          self.center = center
21          self.hull.points = n_data_backup
```

5.4 Testing

Testing av konstruktorene er gjort ved å undersøke om parametere blir satt riktig ut ifra det som blir sendt med. Eksempelvis har vi generert disse testdata Listing 5.5 som blir benyttet til å teste om gamut klassens *vertices* variabel etter opprettelse inneholder de korekte hjørner til gamuten. Konveks hull [20] er en importert metode fra Scipy [21] pakken og det er antatt at denne er testet grundig.

Listing 5.5: Testdata for å teste om vertices variabelen i gamut klassen vil inneholde punkter som er ekte hjørner. Vertices vil inneholde kun indekser til punktene i points variabelen.

```
1
2   cube = np.array([[0., 0., 0.],      # 0  vertices
3                    [10., 0., 0.],     # 1  vertices.
4                    [10., 10., 0.],    # 2  vertices.
5                    [0., 10., 0.],     # 3  vertices.
6                    [5., 5., 5.],      # 4  non vertices.
7                    [4., 6., 2.],      # 5  non vertices.
8                    [10., 10., 10.],   # 6  vertices.
9                    [1., 2., 3.],      # 7  non vertices.
10                   [10., 0., 10.],    # 8  vertices.
11                   [0., 0., 10.],     # 9  vertices.
12                   [0., 10., 10.]])   # 10 vertices.
13  cube_vertices = np.array([0, 1, 2, 3, 6, 8, 9, 10])  # Vertices for the cube.
```

6 PGL-36 FeitoTorres

Metode som avgjør om fargepunkter er innenfor eller utenfor en gamut. Is_inside()(implementasjonen av Feito-Torres algoritmen) er en metode som benyttes i alle GMAer aktuelle for dette prosjektet.

6.1 Teori

Først og fremst ønsker vi å gjøre noen viktige definisjoner for dette kapittelet. Disse lokale definisjonene er kortfattede og delt opp i følgende to kategorier: generell matematisk, og kode og algoritme.

Matematiske definisjoner

- **Original:** Original vil bli benyttet når punktet origo legges til et polytop fra gamuten. Eksempelvis vil en *original trekant* være 2-polytopen definert av hjørnene til en kant i gamuten og origo. Et *originalt tetraeder* vil være 3-polytopen definert av hjørnene til en av gamutens fasetter og origo.
- **Indre punkt:** Et punkt i en polytop, som ikke er del av polytopens ytterpunkter, rand eller overflate. Dersom vi ikke eksplisitt bruker benevningen "indre punkt", anta at ytterpunkter, rand og overflate er inkludert.
- **Format av punkter og vektorer:** Store bokstaver bruker om punkter, men små bokstaver med pil over brukes for vektorer. A er et punkt, $\vec{a}$ er vektoren fra origo til A.

Kode og algoritme

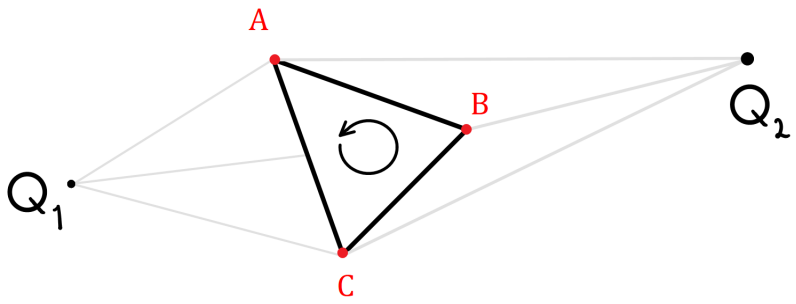
- **Q:** Punktet som det sjekkes om er innenfor eller utenfor den aktuelle gamuten.
- **Inclusion:** Et heltall som telles opp og ned underveis i algoritmen. Verdien til inclusion når algoritmen er ferdig avgjør om Q er innenfor, unntatt når spesialtilfelle 3 inntreffer (Se forklaring av algoritmen).
- **Sign():** Denne funksjonen beregner fortegnet til det signerte volumet til et tetraeeder. Metoden benyttes får å avgjøre hvilken side et punkt er i forhold til et plan (Planet blir definert av fasettenes tre hjørner). Om P er på samme side som planets normalvektor peker returnerer sign() positiv én. I Figur 14 vil Sign(Q_1 ABC) returnere negativ én, sign(Q_2 ABC) returnerer positiv én. Dersom Q er på planet definert av ABC returnerer sign() null.

6.2 Kravspesifikasjon

I dette avsnittet viser vi først oppgaven som den var gitt av oppdragsgiver og så hvordan vi tolket oppgaven.

6.2.1 Kravspesifikasjon fra oppdragsgiver

Metode i Gamut-klassen som avgjør om punktene i et gitt datasett er innenfor eller utenfor gamuten i form av en boolsk ndarray. Pseudokode for syntaks Listing 6.1.



Figur 14: A B og C er hjørner i en fasett. Sign(Q_1 ABC) returnerer negativ én og sign(Q_2 ABC) returnerer positiv én.

Listing 6.1: Der d er colour.data.Data punkter i et gitt fargerom og g vil være gamut objekt.

```
1  d = colour.data.Data(Nx...xMx3, colour.space.XXX)
2  g = colour.gamut.Gamut(...)
3  g.is\_inside(d, colour.space.XXX) returnerer Nx...xM boolsk ndarray
```

6.2.2 Tolkning

Gitt et colour.data.Data objekt med dimensjoner Nx..xMx3, der siste dimensjon inneholder fargedatapunktenes tre koordinater, skal vi skrive en metode som returner en boolsk ndarray med dimensjoner Nx...xM. Resultatet på om fargedatapunktet er innenfor gamuten lagres med samme indeks i den boolske arrayen som den hadde originalt. Retur arrayen skal altså inneholde *true* hvis dette punktet var innenfor gamuten, eller *false* hvis punktet ikke var innenfor.

Metoden må håndtere at arrayen punktene ligger i skal kunne ha et variabelt antall dimensjoner, der siste dimensjon alltid er tre lang og inneholder x, y og z koordinatene til punktet. Retur arrayen skal ikke ha en koordinat i siste dimensjon men kun en verdi(*true/false*).

Bruker sender også med fargerom testen skal utføres i. Det vil si at vi må sørge for å hente ut punkene fra colour.data.Data objektet i riktig fargerom.

6.3 Arbeidsprosess

I dette delkapittelet presenteres arbeidsprosessen som førte til resultatet for denne oppgaven. Dette var en av de første og den største oppgaven i prosjektet, dette er derfor en viktig arbeidsprosess for å vise utviklerens læringsprosess.

For å være sikre på at vi fant frem til en effektiv og god løsning var paret som jobbet med oppgaven veldig tålmodig med å ikke hoppe rett inn i å skrive kode. Vi gjorde grundig litteratursøk rundt hvilke algoritmer som fantes, og om det var tilgjengelig kode for disse. Vi presenter metodene i rekkefølgen de ble vurdert.

Tidlig i prosessen støtte vi på disse to metodene: *Normalvektor-metoden*[22] og *Remake convexHull*[23]. Normalvektor-metoden går i korte trekk ut på å se om skalarproduktet mellom fasettenes normalvektor og en vektor fra Q til et av fasettens hjørner er positiv for alle gamutens fasetter. Remake-convexHull går ut på å generere en ny gamut fra den originale gamutens punkter og Q. Deretter sammenligne denne nye gamuten med den eksisterende. Dersom gamutene har de samme hjørnene er ikke Q blitt et hjør-

ne i den nye gamuten, som vil si at Q var innenfor den originale gamuten. Etter nærmere undersøkelse rundt disse to alternativene kom vi frem til at de ikke kunne benyttes da de ikke tar hensyn til konkave gamuter.

Kalay's metode [24] reduserer den tre-dimensjonale gamuten til en to dimensjonal gamut, for så å bestemme om Q ligger innenfor eller utenfor mens Feito og Torres[25] to spanske forskere har utarbeidet en metode som primært benytter seg av fasetters orientering til å bestemme om et punkt er innenfor eller utenfor en gitt gamut. Begge disse metodene tok hensyn til konkave gamuter og kvalifiserte seg derfor til videre undersøkelse.

Videre litteratursøk presenterte en artikkel som sammenlignet hastigheten til flere *inclusion testing*-algoritmer (Blandt dem var Kalay's metode og Feito-Torres metoden) [26], og konkluderte med at Feito-Torres var den raskeste av de gitte metodene. I Feito-Torres artikkelen var det også nevnt at Kalay's metode er ytterligere mer kompleks å implementere da den må ta hensyn til flere spesialtilfeller [25]. Dermed bestemte vi oss for å bruke Feito-Torres algoritmen.

Vi brukte mye tid på å sette oss inn i Feito-Torres artikkelen. Den var skrevet på et høyt matematisk nivå og krevde derfor at vi ofte måtte lage visuelle representasjoner av konsepter og teknikker som ble omtalt. En viktig avgjørelse vi ønsker å fremheve er at vi valgte å lage vår egen pseudokode basert på pseudokoden presentert i artikkelen, se Figur 15. Vår pseudokode ligger i vedlegg O. Artikkelen var for oss noe tunglest og komplisert å forstå, mens pseudokoden var enklere å relatere til. Dette gjorde at vi bearbeidet koden og dens prinsipper på et detaljert nivå og fikk et godt overblikk over hvilke hovedoperasjoner som algoritmen baserte seg på.

Vi utviklet metoder som gjorde hovedoperasjonene i algoritmen. Dette var å avgjøre om et punkt ligger i/på:

- et linjestykke
- en trekant
- et tetraeder

For å lage en metode som tester om Q er på et gitt linjestykke undersøkte vi løsninger på stackoverflow. Vi fant en løsning vi likte [27] og implementerte denne i metoden_line().

For trekanter måtte vi igjen gjøre en del literatursøk for å finne en effektiv løsning. Det stod til slutt mellom to ulike metoder, *Same Side Technique*[28] og *Barycentric Technique*[29]. Metodene hadde hver sine styrker. *Same side* metoden er meget enkel å forstå og samtidig enkel å implementere, mens Barycentric metoden var noe mer kompleks å forstå men er raskere [30]. Da denne metoden skal brukes på veldig mange punkter er det viktig at den samlede tiden som blir brukt for alle punkter ikke blir for høy. Derfor valgte vi å bruke Barycentric metoden. Det tok litt tid å forstå matematikken som inngikk, men når vi hadde forstått den var metoden grei å implementere. [31].

For tetraedere bestemte vi oss for at vi kunne bruke *Remake conveksHull*-metoden, da alle tetraeder er konvekse polytoper.

Vi møtte på noen vegger underveis i kodingen. Determinant utregning av koordinater var en av de det tok lengst tid å finne ut av. Numpy sin metode for utregningen av determinanter gav ikke riktige resultater. Den endelige grunnen til de ukorrekte utregningene

fant vi ikke ut av, men løsningen ble å bytte ut numpy sin metode med scipy sin linalg determinant metode.

Vi hadde en større diskusjon rundt en løsning på hvordan vi skulle håndtere at punkter kom i et ukjent format. Med ukjent format menes at punktene kunne komme i et format med variabelt antall dimensjoner. Punktene vil altså komme på et slikt format: Nx...xM x3. Der det er et ukjent antall dimensjoner mellom N og M, og x3 betyr at her ligger punktets koordinater(eks X,Y,Z). Eksempelvis vil et enkelt punkt har formatet 1x3 og et bilde med 640x400 piksler vil ha formatet 640x400x3, andre formater som AxBxCxDx3 er også en mulighet(A, B, C og D er vilkårlige konstanter.

Her viste det seg at utviklerparets manglende kunnskap innen programmering kostet mye tid. Vi tenkte først på å bruke en lineariseringsmetode som enkelt forklart tar en ndarray med ukjent antall dimensjoner og flater den ut til Mx3. Det var en enkel sak å linearisere dataene, men vi trodde vi da ikke ville ha mulighet for å endre tilbake til originalt format. Det å endre tilbake til originalt format lærte vi i neste review møte at var en enkel sak, men da var vi allerede ferdig med å implementere en annen løsning. Vi kom opp med ideen om å skrive en rekursiv metode, som vi har mye erfaring med fra et tidligere algoritmisk programmerings fag. Metoden vår ble ikke større enn 14 linjer kode, men er nevneverdig mer tungvinn å implementere en lineariseringmetoden. Men vi kan med stolthet kan si at vår rekursive metode er noe kjappere.

Som Figur 16 viser er det en forskjell i tiden de to metodene bruker på å gi et resultat. Testresultatet presentert i grafen er gjort på to-dimensjonale data. Det vil si eksempelvis punkter fra et bilde. Vi har gjennomført tester på tre-dimensjonale data med 1000 punkter og resultatene er de samme. Traverse metoden er noe kjappere.

Resultat #1 (6.4.2) presenterer den første fungerende løsningen for oppgaven som vi presenterte for oppdragsgiver. Under scrum retrospective møtet ble det av oppdragsgiver gjort tydelig at metoden ikke var godkjent når det gjaldt eksekveringstid. Med dette menes at metoden ikke genererer et resultat innen en godkjent tidsperiode. For 100 punkter brukte metoden 60 sekunder. Det vil si for et reelt bilde med f.eks. en megapiksel (1 million punkter) ville metoden bruke ca 16 tusen timer, og det ville i en forskningssammenheng ikke være brukbart.

På den påfølgende sprinten(sprint 4) ble det bestemt av oppdragsgiver at det skulle undersøkes muligheter for å forbedre metoden. Av oppdragsgiver ble det foreslått å benytte et profiling [32] [33] verktøy. Dette er et verktøy som analyserer tids -/og ressursbruk i kode, og presenterer resultatet i oversiktlige tabeller og grafer. Videre ble det foreslått å undersøke om bruken av Cython [34] kunne gjøre en forskjell. Cython gjør det mulig å eksekvere kode skrevet i programmeringsspråket C, innad i Python koden. Da C kode kompileres før den kjører er den merkbart raskere når det kommer til matematisk tunge deler av kode. En annen mulig løsning kunne være å benytte numba, en såkalt *just in time compiler*. Denne vil i likhet med Cython kompilere koden før den kjører, som gjør koden raskere [35]. Det siste forslaget var å gå over den produserte koden og se etter andre løsninger på de kodebitene som krevde mye tid.

Bruken av et profiling verktøy gav følgende bilde (Vedlegg A). Vi stusset på tidsbruken til is_inside metoden, da Feito-Torres artikkelen [25] selv mente vi at metoden skulle være kjapp. Prosjektgruppen bestemte seg derfor på å bruke noe tid til å sette seg inn i Feito-Torres artikkelen på nytt, og se om det hadde oppstått noen misforståelser under-

veis i tolkningsprosessen. Det viste seg at akkurat dette hadde skjedd. Vi hadde oversett at det i artikkelen gis en felles løsningsstrategi for alle testene metoden behøver for å bevise om et punkt ligger utenfor eller innenfor. Det artikkelen foreslo var å benytte sign, altså orientering av fasetter til å avgjøre punktets plassering i forhold til origo og fasetten. Vi refaktorerte hele Feito-Torres metoden ved å bytte ut testene for linje, trekant og tetraeder med foreslått bruk av sign metoden. Deretter kjørte vi en del tester og brukte profiling verktøyet til å analysere hvorvidt forbedringene gjorde en forskjell. Den nye metoden og sluttresultatet beskrives i detalj under punktet resultat #2 (6.4.3).

6.4 Resultat

6.4.1 Forklaring av algoritmen

I dette avsnittet vil vi forsøke å gi en enkel konseptuell forståelse av algoritmen. Vi tenker oss at vi starter i origo og beveger oss mot Q, langs linjen L Figur 17. Dersom vi starter innenfor polyeederet og krysser et partall antall fasetter før vi ender i Q er Q innenfor polyederet, ellers ved oddetalls kryssninger er Q utenfor. Dersom vi starter utenfor polyederet og vi krysser et oddetall antall fasetter er Q utenfor, ellers ved partalls kryssninger er Q utenfor. For å finne starttilstand og antall kryssninger undersøker vi alle originale tetraedere tilhørende gamutens fasetter som Q er en del av.

Videre følger en mer detaljert forklaring av algoritmen. Alle fasetter(F) i gamuten har et tilhørende originalt tetraeder(T). Dersom Q er et indre punkt i T oppdaterer vi inclusion med sign(T).

Dersom Q er et punkt på overflaten til T dreier det seg som et av tre spesialtilfeller.

Det første spesialtilfelle er når Q er punkt på F, om dette spesialtilfelle inntreffer kan vi umiddelbart konkludere med at Q er innenfor gamuten, da F utgjør en del av dens overflate.

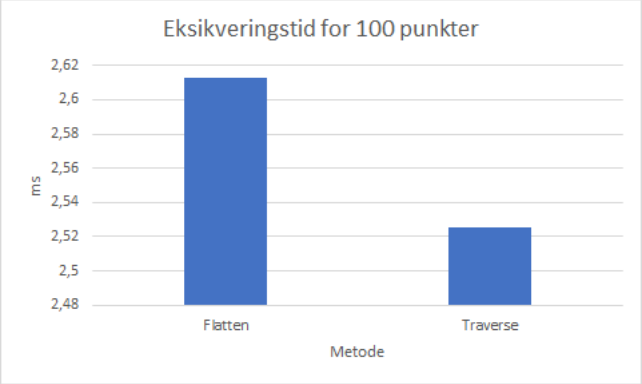
Det andre spesialtilfellet er om Q er et indre punkt i en av T sine originale trekanter. I så fall må vi oppdatere inclusion med 1/2 sign(T), da den nabo fastten til F som deler kanten den originale trekanten trekkes ifra må ha lik sign() verdi om vi faktisk passer gjennom overflaten. Dette er illustrert i figur Figur 18.

Det siste spesialtilfelle er når Q ligger langs en av F sine originale kanter. Dette ligner på spesialtilfelle nummer to, da vi må ta hensyn til sign() verdien til flere fasetter før vi kan konkludere med om vi faktisk passerer igjennom overflaten til gamuten. Settet av alle fasetter som deler hjørnet i punktet den originale kanten trekkes ifra betegnes videre som S. Dersom alle fasetter i S har lik sign() verdi vet vi at vi passerer igjennom overflaten og deres delte sign() verdi bestemmer om vi passerer inn eller ut av gamuten. I algoritmen løses dette ved å kun addere én til inclusion én gang, og kun dersom det finnes minst én fasett i S med positiv sign() verdi. Tilsvarende subtraheres én fra inclusion kun én gang, og kun om det finnes minst én fasett med negativ sign() verdi i S.

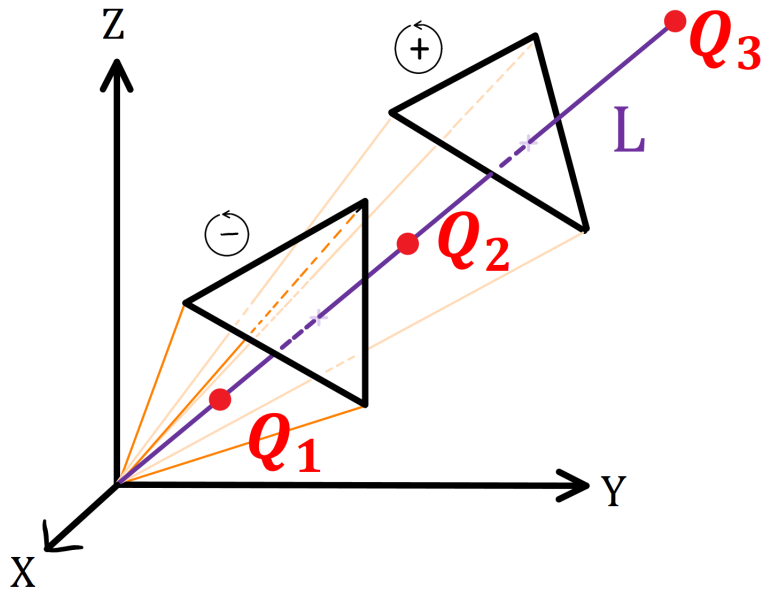
```
INCLUSION=0
V+ (Q)=empty
V- (Q)=empty
for i=1 to n {
        if (Q ∈ Fi) { INSIDE=true; EXIT }
        if ((Q∈ Int(OV_i,1) and ((sign(OFi)>0 and not V_i,1 ∈ V+ (Q) or
          (sign(OFi)<0 and not V_i,1 ∈ V- (Q)) {
            INCLUSION=INCLUSION+sign(OFi)
            if (sign(OFi)<0)
                    V- (Q)= V- (Q) + V_i,1
            else    V+ (Q)= V+ (Q)+ V_i,1 }

        else if ((Q ∈ Int(OV_i,ei) and ((sign(OFi) >0 and not V_i,ei ∈ V+ (Q) or
           (sign(OFi)<0 and not V_i,ei ∈ V- (Q) ))) {
            INCLUSIONS=INCLUSION+sign(OFi)
            if (sign(OFi) <0)
                    V-(Q)= V-(Q)  + V_i,ei
            else    V+(Q)= V+(Q) + V_i,ei }
        else for j=2 to ei-1 {
            if ((Q ∈ Int (O,V_i,1,V_i,j) or (Q ∈ Int (O,V_i,j,V_i,j+1)
              or (Q ∈ Int (O,V_i,j+1,V_i,1)) {
                INCLUSION=INCLUSION + 1/2*sign([O,V_i,1,V_i,j,V_j+1])
            else if ((Q ∈ Int (OV_i,j) and ( ( sign([O,V_i,1,V_i,j,V_j+1])>0
                and not V_i,j∈ V+ (Q) ) or (sign([O,V_i,1,V_i,j,V_i,j+1])<0
                and not V_i,j∈ V- (Q) ) {
                INCLUSION=INCLUSION+sign([O,V_i,1,V_i,j,V_i,j+1]
                if sign([O,V_i,1,V_i,j,V_j+1])<0
                    V- (Q)= V- (Q) + V_i,j
                else V+ (Q)= V+ (Q)+ V_i,j }
            else if (Q ∈ Int (O,V_i,1,V_i,j,V_i,j+1 )
                    INCLUSION=INCLUSION+sign([O,V_i,1,V_i,j,V_i,j+1])
        }
}
```

Let $P=F_1,F_2,..,F_n$ be a polyhedron, where F_i (i=1..n) are the faces, and the vertex of face i are: $F_i=V_{i,1},V_{i,2},...,V_{i,ei}$. Sign($OF_i$) is the sign of the pyramid OF_i. The algorithm decides if the point Q is inside the polyhedron, the result is obtained in the variable INCLUSION, which will be 1 only if Q∈P.

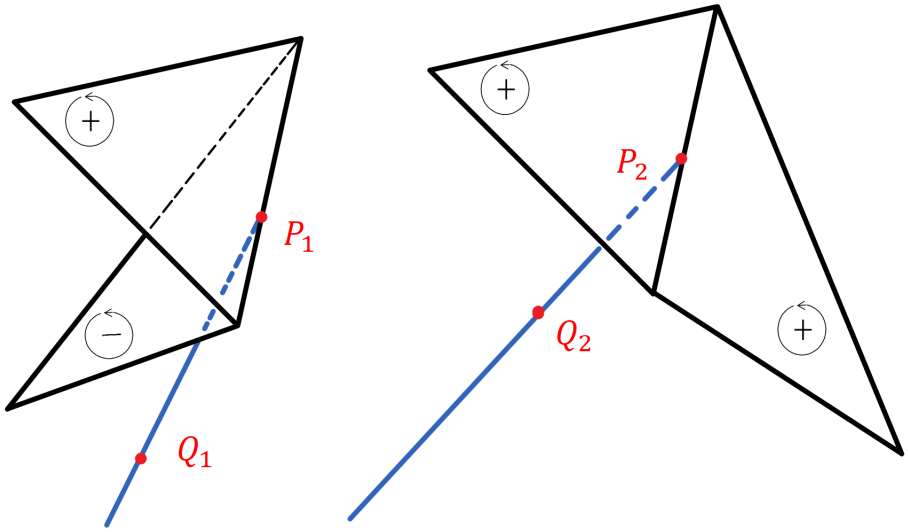
Figur 15: Pseudokode fra Feito-Torres artikkelen [25].



Figur 16: Eksikveringstid på hundre punkter. Testen er gjennomført på en Intel i5-6200U. Lavere er bedre.



Figur 17: Q_1 finnes kun i et originalt tetraeder med negativ sign(), punktet er utenfor gamuten. Q_2 finnes kun i et orgiginalt tetraeder med postiv sign(), punktet er utenfor gamuten. Q_3 finnes ikke i noen orginalt tetraeder, og er utenfor gamuten.



Figur 18: Illustrsjon av to sett med fasetter. Den orginale linjen(blått) passerer ikke gjennom overflaten i P_1. Den orginale linjen passer gjennom gamuten i P_2.

6.4.2 Resultat #1 før refaktorering

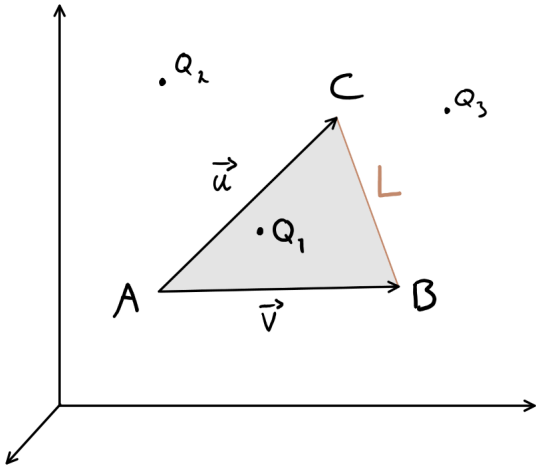
I den første implementasjonen av algoritmen implementerte vi en metode for hver av de nevnte hovedoperasjonene. Vi vil i dette avsnittet forklare kort hvordan disse metodene fungerer.

Første metoden tester om Q er på et linjestykket fra punkt A til punkt B. Først gjøres A til lokal origo for A, B og Q. $\vec{b}$ blir da en vektor fra origo til B sin nye lokasjon, $\vec{q}$ blir en vektor til Q sin nye lokasjon. Kodelinjene i Listing 6.2 refereres til videre i teksten. Vi sjekker da som $\vec{b}$ og $\vec{q}$ ikke er kolineære(linje 1 til 2). Videre sjekker vi om $\vec{b}$ og $\vec{q}$ ikke peker i samme retning (linje 5 og 6). Til slutt sjekker vi om Q ikke er nærmere A enn B er (9 og 10). Om ingen av disse testene gir utslag kan vi konkludere med at Q er på linjestykket mellom A og B.

Listing 6.2: Utdrag av in_line()

```
1     matrix = np.array([[1, 1, 1], b, q])
2     if np.linalg.det(matrix) != 0:
3         return False
4
5     dot_b_q = np.dot(q, b)
6     if dot_b_q < 0:
7         return False
8
9     if np.linalg.norm(p) > np.linalg.norm(b):
10        return False
11
12    return True
```

Andre metoden sjekker om punktet Q er i en trekant i $\mathbb{R}^3$. Figur 19 tilhører dette avsnittet. Først sjekket vi at Q er på planet definert av trekanten. Deretter gjøres punktet A til lokalt origo slik at vi enkelt kan gjøre kryssprodukt- og skalarproduktoperasjoner med vektorene $\vec{u}$ og $\vec{v}$, og vektorer fra A til Q_i. Først sammenligner vi $\vec{u}$ X $\vec{q}$(Kryssproduktet mellom $\vec{v}$ og $\vec{q}$) med $\vec{q}$ X $\vec{u}$. Dersom de resulterende vektorene peker i forskjellig retting, altså at tommelen ifølge høyrehåndsregelen peker i forskjellig retning er $\vec{q}$ utenfor vinkelen mellom $\vec{v}$ og $\vec{u}$. Det samme gjøres for $\vec{u}$ X $\vec{q}$ og $\vec{q}$ X $\vec{v}$. Etter disse to sammenligningene vet vi at $\vec{q}$ peker ut fra A mellom $\vec{u}$ og $\vec{v}$, så da gjenstår bare å undersøke hvilken side av L Q ligger på. Dette gjøres ved å utnytte egenskaper ved barycentriske koordinater[31].



Figur 19: Illustrasjon av en fasett med hjørner A, B og C. Q er punkter det skal sjekkes om er innenfor trekanten.

6.4.3 Resultat #2 Forbedringene gjort i Feito-Forres

Som nevnt under arbeidsprosess (6.3) gjorde vi en større refaktorisering av Feito-Torres algoritmen. Istedenfor å se sjekke om Q er på fasettenes originale linjer, trekanter og tetrahedron som separate operasjoner, løser vi dette ved en sammensatt bruk av sign().

Listing 6.3: Pseudokode for bruk av sign()

```
1   S = sign(OABC)
2   S1 = sign(QABC)
3   S2 = sign(QACO)
4   S3 = sign(QAOB)
5   S4 = sign(QBOC)
6
7   Om S er ulik S1 ... S4
8     Q er ikke i tetraedert OABC. STOP.
9
10  Om S1 er lik 0
11    Q er på gamutens overflate. STOP.
12
13  Z = S1 + S2 + S3 + S4
14
15  Om Z er lik 0
16    Q er et indrepunkt i OABC. STOP.
17
18  Om Z er lik 1
19    Q er et indre punkt er av ABC's originale trekanter. STOP.
20
21  Om Z er lik 2
22    Q er et punkt på en av ABC's originale linjer. STOP.
```

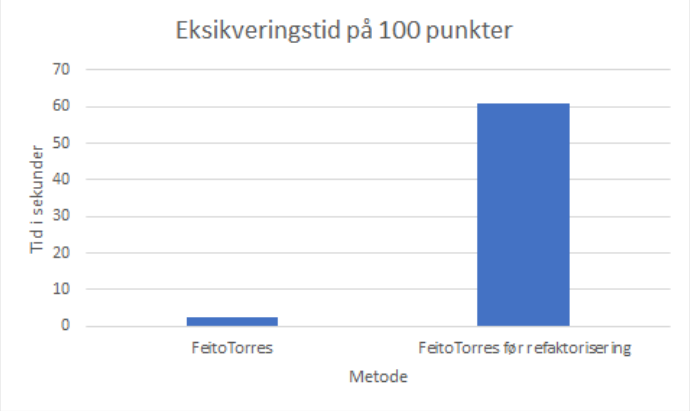
Linje 7 Listing 6.3 stopper algoritmen dersom Q er utenfor det originale tetraederet OABC. Derfor kan vi finne Q sin plassering i OABC ut i fra hvor mange av tetraederts overflater Q befinner seg i. Som en påminnelse nevner vi at dersom sign(...) er lik 0, er parameterpunktene koplane. Her følger de fire mulige tilstandene til Z og hva de representerer:

- **Hvis Sign til tetraederet QABC ble 0:** Siden trekanten ABC er en fasett i gamuten,

betyr dette at punktet Q er et punkt i gamuten.

- **Det ble tilsammen 0, 0er :** Hvis punktet ligger innenfor alle tetraeederets overflater men ikke**på** noen overflatene betyr dette at punktet Q ligger i det originale tetrahederet.
- **Det ble tilsammen 1, 0er:** Hvis kun en av overflatene inneholder Q, ligger punktet på den respektive originale trekanten.
- **Det ble tilsammen 2, 0er:** Hvis to av overflatene inneholder punktet betyr dette at punktet Q ligger på en linje mellom to av det originale tetrahederets fasetter.
- **Det ble flere enn 2, 0er:** Dette vil aldri skje. Om Q befinner seg i et hjørne i det originale tetraederet er Q enten lik A, B eller C og spesialtilfelle 1 tar over, eller så er Q origo. Algoritmen stiller det kravet at origo ikke kan være et hjørne i gamuten.

Etter refaktorisering målte vi en reduksjon i eksekveringstid på 96%. Vi har altså redusert tiden det tar for å generere et resultat for 100 punkter fra 60 sekunder til ca. 2 sekunder se Figur 20



Figur 20: Forbedringen av Feito-Torres metoden gir 96% kjappere resultat. Testen er gjennomført på en Intel i5-6200U. Lavere verdi er bedre.

Dette resultatet er en enorm forbedring fra den originale metoden. Det er fortsatt ikke brukbart i sammenheng med virkelige data fra bilder etc. Et bilde tatt med en moderne smarttelefon vil ha flere millioner piksler og som Figur 21 viser vil et slikt bilde ta mange timer å behandle.

6.5 Testing

Testingen av denne metoden har endret seg en del med tanke på at vi har endret og refaktorisert denne metoden gjennom utviklingsperioden. Vi har skrevet tester for alle delkomponenter og testet is_inside metoden meget nøye. Metoden is_inside er testet ved at vi sender med data i en, to og tre dimensjoner og tester korrekt svar på om punktene var innenfor eller utenfor. Det testes også på at datatypen og dataformen(dimensjonen og struktur) er riktig. Resultatene testets med ulike gamuter, kube og kuleformet gamut fungerer. Metoden er også testet på at både traverse og flatten metodene kan benyttes.

Figur 21: Sammenligning av eksikveringstid målt i sekunder på ulike mengder med fargepunkter. På 1MP ligger Flatten på 7timer, mens Traverse ligger rundt 6timer og 45minutter. Testen er gjennomført på en Intel i5-6200U.

7 PGL-40 Nærmeste punkt i en retning

Metoden skal returnere nærmeste punkter på overflaten langs en rett linje fra hvert enkelt punkt til et gitt referansepunkt. En rett linje mellom datapunkter utenfor gamutobjektet og referansepunktet som er innenfor gamut objektet.

7.1 Teori

- **C:** Et punkt innenfor gamuten. C er ofte gamutens senter.
- **D:** Et punkt utenfor gamuten. Det er for dette punktet vi er interessert i å finne nærmeste punkt på gamutensoverflaten.
- **A:** Et punkt på gamut overflaten. A er det punktet langs L(α) som er nærmest D.
- **L(α):** En parameterisert linje som går igjennom C og D. L(1) = D, L(0) = C.
- **Distanse:** Distanse vil være fra origo til der punktet krysser fasetten/hjørnet(α verdien). Dette finner ved utregning av kryssprodukt mellom to punkter og normalverdien.

7.2 Kravspesifikasjon

I dette avsnittet viser vi først oppgaven som den var gitt av oppdragsgiver og så hvordan vi tolket oppgaven.

7.2.1 Kravspesifikasjon fra oppdragsgiver

Metode i gamut som returnerer nærmeste punkter på overflaten (for farger utenfor gamut) langs linjer fra hvert enkelt punkt til et gitt referansepunkt i et gitt rom. Dette vil typisk være senterpunktet som også er brukt for å generere gamuten hvis det er et modifisert convex hull. Pseudokode for syntaks Listing 7.1

Listing 7.1: Der center er et punkt i fargerommet, og d_clip er et nytt colour.data.Data-objekt.

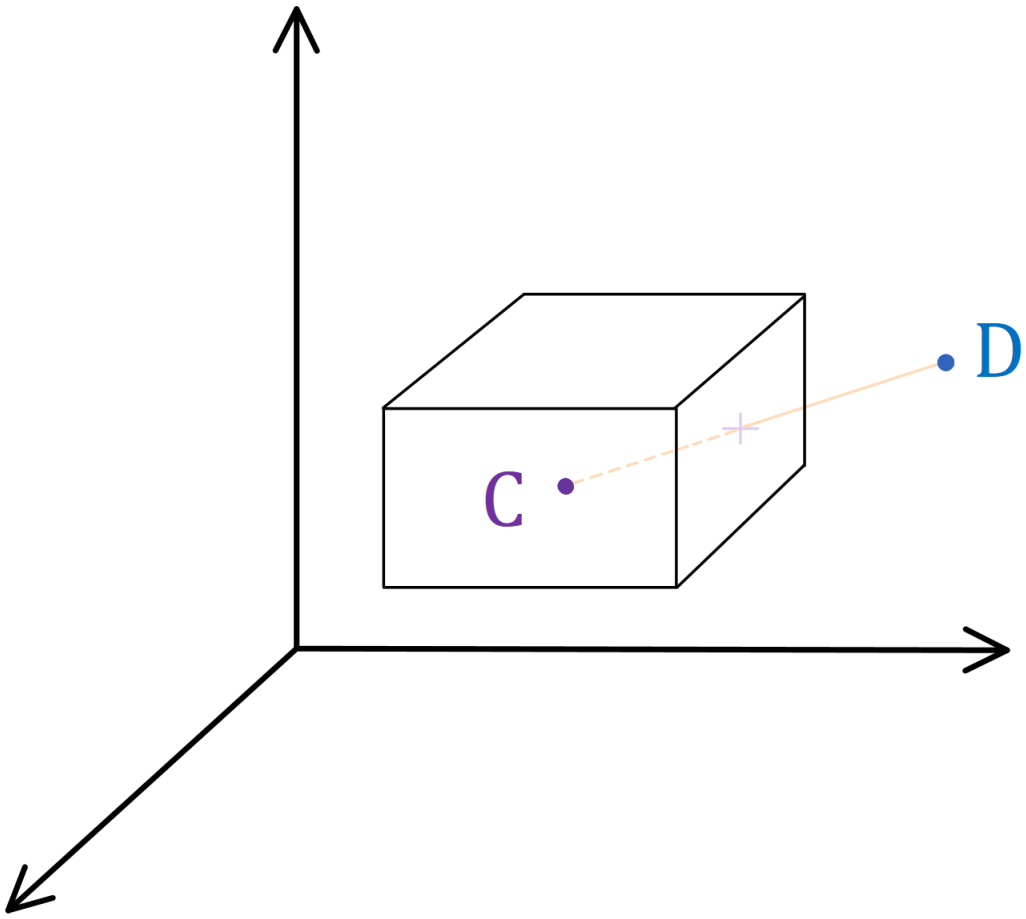
```
1  g = colour.gamut.Gamut(...)
2  d = colour.data.Data(...)
3  d_clip = g.clip_towards(d, colour.space.XXX, center)
```

7.2.2 Tolkning

Det skal skrives en metode som returnerer de nærmeste punktene på overflaten (for punkter utenfor gamuten) langs en rett linje fra hvert enkelt punkt til et gitt referansepunkt. Dette vil typisk være et punkt innenfor gamut objektet eller senterpunktet til gamuten. Metoden skal kunne ta imot tre parametere der det vil være punkter innenfor og utenfor gamut objektet i et gitt fargerom. Parameter *d* vil ta imot ndarray-punkter som er utenfor gamut objektet og *center* vil ta imot ndarray punkter innenfor gamut objektet, men metoden behøver ikke kjenne et *center* punkt. Det må eventuelt lages flere metoder som skal beregne L(α), finne normalvektoren og distansen. Alle metoder blir implementert under gamutklassen. Når metoden har finne nærmeste punkt i en gitt retning skal den returnere punktet der den krysser gamut objektet.

7.3 Arbeidsprosess

Det ble avtalte et møte med oppdragsgiver siden det var mye som skulle gjøres og vi ikke visste hvordan oppgaven skal løses. Etter møte fikk vi bedre innblikk på hvordan gå fram for å løse oppgaven. Vi startet med å skrive pseudokode på hva som trengtes i oppgaven. Første steg er å implementer en hovedmetoden som skal ta imot datapunkter innenfor og utenfor gamut objektet i riktig fargerom. Videre skal variablene bli sendt til neste metode som vil gå gjennom alle fasetter og finne normalverdiene og distansen for så å kunne beregne L(α). Ved å finne L(α) og hvor punktet krysser må det settes opp en matematisk ligning. Etter diskusjon i gruppen kom vi frem til en ligning som kan hjelpe oss med å finne L(α) verdien og deretter kunne regne ut punktet der alphaverdien krysser. Det ble laget en skisseFigur 22 ut ifra hvordan vi skal klare å finne nærmeste punkt i en gitt retning på gamut objekt.



Figur 22: Nærmeste punkt i en gitt retning med en alpha verdi (3D). C er senter av gamuten. D er et punkt utenfor gamuten.

Vi startet med å implementere hovedmetoden som skal ta imot tre parametere. Første parameter vil være colour.data punkter som er utenfor gamuten, andre parameter vil enten være et punkt innenfor gamut eller satt til å være senter punktet til gamuten. Vi diskuterte med oppdragsgiver angående C-punktet og kom fram til at metoden ikke skul-

le kreve å få et senter punkt. Om det ikke skulle sendes med en C-punkt vil metoden da bruke geometrisk senter til gamuten. Tredje parameter er i hvilket fargerom kalkulasjonen skal utføres. Videre i metoden sender vi C-punktet, punktene som er utenfor gamut enkeltvis og i hvilket fargerom til neste metode.

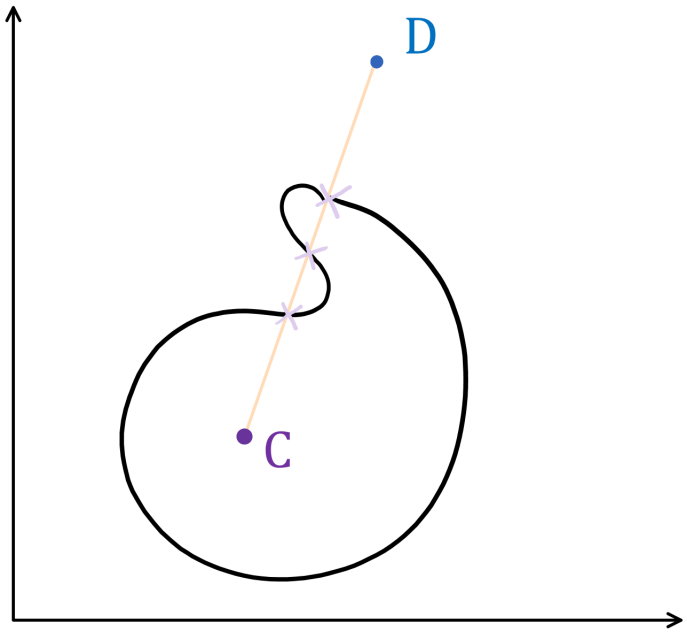
Neste metode vil gå gjennom alle fasetter og finne normalvektor verdiene og distansen. Ved litt kryssprodukt regning og bruk av numpy sin funksjon linalg.norm kan vi finne normalverdiene og deretter finne distansen. Etter å ha funnet normalverdiene og distansen kan ligningen som ble satt opp for å løse L(α) brukes.

L(α) ligningen:

$$\alpha = \frac{(distance - C\vec{[0]} \cdot n\vec{[0]} - C\vec{[1]} \cdot n\vec{[1]} - C\vec{[2]} \cdot n\vec{[2]})}{D\vec{[0]} \cdot n\vec{[0]} - C\vec{[0]} \cdot n\vec{[0]} + D\vec{[1]} \cdot n\vec{[1]} - C\vec{[1]} \cdot n\vec{[1]} + D\vec{[2]} \cdot n\vec{[2]} - C\vec{[2]} \cdot n\vec{[2]}}$$

Der *distance* er avstanden mellom origo og den linjen krysser gamuten. *C* er punktet innenfor gamut objektet, *D* er punktet utenfor gamut objektet og *n* som er normalverdiene.

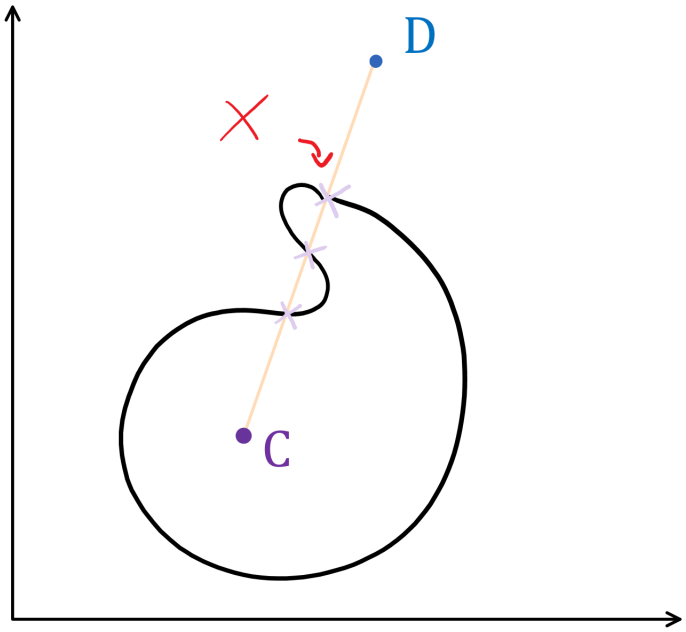
Vi er bare interessert i alpha verdier fra null til en. Som du kan se i Figur 22 så vil rette linjen fortsette ut av punktene C og D. I dette tilfellet er vi ute etter alpha verdier som krysser gamut objektet innenfor disse to punktene.0Ddet vil si null og en. Det ble diskutert i gruppen at man kan få flere alpha verdier innen område null til en hvis gamut objektet er formet på en slik måte som i Figur 23. Videre i metoden sjekkes vi om alpha verdiene er innenfor en fasett ved hjelp av en metode kalt in_triangle. Hvis alpha verdien er innenfor en fasett legges den i en sortert array liste. Den høyeste alphaverdien vil angi det nærmeste punktet hvor linjen krysser gamutobjektet. Med alpha verdien, D og C punktet blir det mulig å regne ut nærmeste punktet[x,y,z] på overflaten i en gitt retting.



Figur 23: Nærmeste punkt i en gitt retning med flere alpha verdier (3D). C er senter av gamuten, D er et punkt utenfor gamuten.

7.4 Resultat

Vi har ligningen for å kunne finne nærmeste krysningspunkt i en gitt retning. Ligningen er satt opp slik: $F(x) = \alpha \cdot D + C - \alpha \cdot C$, der det benyttes den største alpha verdien som ble funnet og D som er punktet utenfor gamut objektet og C som er punktet innenfor gamut objektet. Denne ligningen skal den kunne finne hvor den akkurat krysser punktet X som du kan se på Figur 24 og få resultattet i [x,y,z] form.



Figur 24: Linjen mellom punktet D og C, krysser gamuten på flere fasetter og vi får dermed flere krysningspunkter. Vi ønsker å finne det nærmeste punktet X i rødt.

7.5 Testing

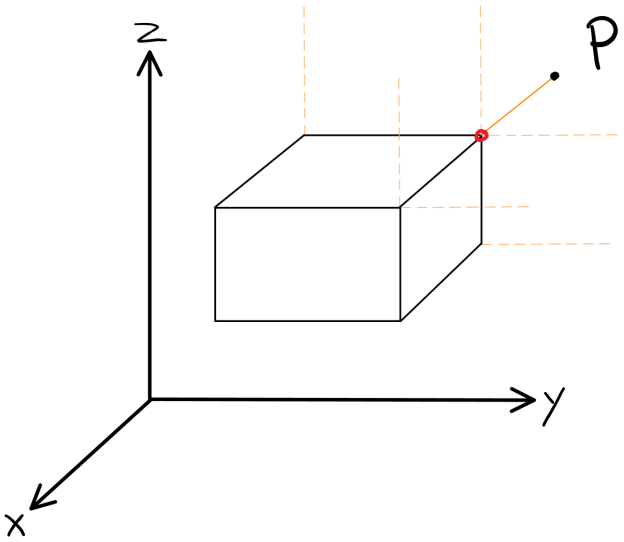
Vi testet ved å lage en kube-gamut, der C vil å være [5,5,5] og D vil å være [5,5,15] og med dette skal resultatet være [5,5,10]. Resultat med hovedmetoden er å returnere nærmeste punkt i en gitt rett linje. Vi fikk dette til å stemme, men oppdragsgiver ønsket at metoden skal ta imot ndarray. Da ble det endringer i hovedmetoden til å ta imot ndarray og sende punktene enkeltvis til neste metode for kunne beregne og finne normalvekorverdiene, distansen og alpha verdien. Vi testet igjen med å sette D til å være ([[15, 5, 5], [5, 15, 5], [5, 5, 15]]) og denne gangen blir ikke C-punktet sendt med. Resultatet som forventes er ([[10, 5, 5], [5, 10, 5], [5, 5, 10]]). Med dette endte vi med riktige punktverdier.

8 PGL-37 Clip Nearest

Metoden skal ta imot datapunkter som er utenfor en enhetsgamut. Ved beregning skal metoden returnere nærmeste punkt på overflaten i en hvilken som helst retning mot en gamut i 3D.

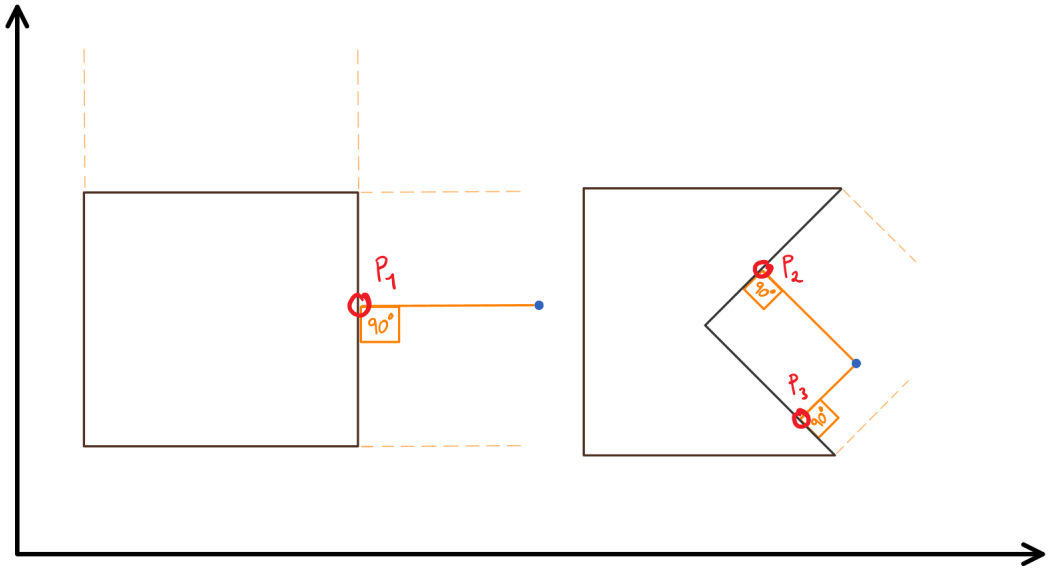
8.1 Teori

For å regne ut distansen vi trenger for å finne nærmeste hjørne Figur 25, finner vi normalen(norm) mellom punktet vi har fått i metoden og alle hjørner som vi har i gamuten. Vi sorterer distansene og bruker den korteste. Når vi har identifisert nærmeste hjørne, går vi gjennom alle fasetter og finner alle som har den på en av sine kanter. Vi finner i såfall alpha verdi og planen for hver fasett, for å finne den alpha verdien som er nærmest null. Punkter med lav alpha verdi vil være nærmest senter.



Figur 25: Nærmeste hjørne(Rød punkt) for gitt punkt P.

For å finne nærmeste punkt brukes formelen, $(P + \alpha \cdot n) \cdot n$. Der P er medsendt punkt verdi, alpha er verdien vi finner for følgende fasett og n er normalen for en fasett. Alpha verdi er der linjen treffer gamut objektet Figur 26. Jo lengre unna senter punktet er jo høyere alpha verdi vil den ha. Vi ser i dette tilfellet etter minste verdi, null er selve sentrum. Vi regner ut punktet og sjekker om den er innenfor fasetten. Om punktet er innenfor blir punktet returnert, hvis ikke blir nærmeste hjørnet returnert i istedet.



Figur 26: Et punkt(blått) kan projekseres på flere fasetter, den nærmeste P velges.

8.2 Kravspesifikasjon

I dette avsnittet viser vi først oppgaven som den var gitt av oppdragsgiver og så hvordan vi tolket oppgaven.

8.2.1 Kravspesifikasjon fra oppdragsgiver

En metode som returnerer nærmeste punkt på overflaten for de punktene som er utenfor gamut. Dette er altså en GMA av absolutt enkleste type. Pseudokode for syntaks Listing 8.1

Listing 8.1: Der d_clip er et nytt colour.data.Data-objekt.

```
1    g = colour.gamut.Gamut(...)
2    d = colour.data.Data(...)
3    d_clip = g.clip_nearest(d, colour.space.XXX)
```

8.2.2 Tolkning

Metoden skal kunne ta imot et datapunkt som er utenfor gamuten og gjennom algoritmen i clip_nearset skal metoden kunne returnere nærmeste punkt på overflaten for gamuten. Metoden skal kun ta imot et datapunkt og skal klare å finne distansen ved hjelp av det.

8.3 Arbeidsprosses

Første steget for oss var å sette oss inn i matematikken og prøve å forstå den og klare å se for oss løsningen. Vi gikk gjennom matematikken og lagde pseudokode for hvordan vi skal gå frem steg for steg, stegene var som følger:

- **Steg 1:** Løkke som går gjennom alle hjørner og henter kortest avstanden mellom hjørne og punktet utenfor gamut. Det er for å finne nærmeste hjørne til punktet.
- **Steg 2:** Løkke som går gjennom alle fasetter som er nabo med hjørner vi sender

med for å kunne finne ut hvilken fasett er den nærmeste i forhold til punkt utenfor.
- **Steg 3:** Se om den nærmeste fasetten er innenfor planet. Hvis ja kjører algoritmen for å finne nærmeste punkt på gamut. Hvis nei returnerer vi kortest avstand fra første løkke. Det vil si nærmeste hjørne.

Steg 2 var å forstå hvordan naboer (neighbors) fungerer i gamut fargerom, det var en litt utfordrende oppgave med tanke på at vi ikke hadde mye forhåndsinformasjon om det og dermed var det tidskrevende å lese seg opp på det.

Metoden clip_nearest går gjennom alle hjørner for først å finne den nærmeste. Vi finner distansen til hvert hjørne og bruker den med kortest distanse. Deretter går vi gjennom alle fasetter som har dette hjørnet som nabo og henter alle koordinatene til dem. Neste steg er å finne plan og alpha verdi for alle faseter som vi har som naboer. Her utnytter vi også eksisterende metoden, find_plane til å finne plan for fasettene. For alpha verdien bruker vi formelen $D + \mathtt{numpy.dot}(P, \mathtt{norm})$. Der P er medsendt punkt verdi, D er distansen og norm er normalen for fasetten. numpy.dot utfører matrise produkt. Siste steg er å sortere alle alphaverdier etter minste verdi først for så å sende alt av verdier til line_alpha som regner frem nærmeste punkt i gamuten og returner det.

Mye av tiden på starten av PGL-en ble brukt for å finne best mulig måte å kalkulere distansen mellom punktene. Som man kan se i Figur 26 kan et punkt ha flere nærme punkter men det er viktig å finne den nærmeste av dem for å kunne gi så nøyaktig resultat som mulig. Dermed hadde vi i utangangspunkt to teori om hvordan vi skal regne oss frem til distansen.

Det første teorien gikk ut på å bruke sentrum punktet for å finne avstand mellom senter og punktet utenfor gamut. Den prøvde vi å implementere og bruke i en liten stund, men fant fort ut at den ikke er riktig i alle situasjoner og ikke kan bruks. Ettersom den krever senter så kan den gi feil svar for gamut uten senter. Samtidig klarte vi å komme oss frem til en annen metode som vi nå har implementert og bruker i biblioteket. Nå finner vi nærmeste punkt ved bruk av $(P + \alpha \cdot n) \cdot n$ som er forklart i teoridelen for denne PGL-en.

Annet problem som dukket opp er at vi fikk positive verdier fra regnestykket vi lagde og dermed fikk vi distanser som er feil og endte med å få plan i feil retning i forhold til punktet. Dermed måtte vi snu fortegn for å kunne få negative verdi og dermed positiv plan som peker i riktig retning mot punktet utenfor gamut.

8.4 Resultat

Gjennom bruk av vår metode klarer vi å finne nærmeste punkt for alle punkter utenfor gamut og kan presentere det tilbake til bruker. Som vist i Figur 26 ser vi at det er flere problemer som kan dukke opp under utregningsprosessen. Nedenfor er dette beskrevet.

Vi kan se at det kan være kun et punkt i noen tilfeller der det er enkelt å skille ut hvilket som er nærmest, mens andre ganger kan det være vanskelig. Det er vanskelig å skille ut hvilket som er nærmest når man har to eller flere punkter som er gode kandidater til å være nærmeste punkt i gamuten. Her kommer algoritmen inn og alle sjekkene som er lagt inn for å sikre oss om at vi finner riktig punkt. Spesielt å merke seg er alpha verdien som er implementert i algoritmen. Alpha verdien gir et svært nøyaktig distanseverdi som brukes for å skille ut nærmeste punkt fra andre punkter som også kan være nærme, men

vi kan skille distansen mellom de med mange desimaler margin og dermed ha 100% nøyaktighet iforhold til verdiene vi bruker i enhetstestene.

8.5 Testing

For å teste metoden opprettet vi en gamut med datapunkter for en kube. Følgende data punkter [[5, 5, 15], [5, 5, 15], [5, 5, 15]] ble sendt til metoden og vi regnet oss frem til en forventet verdi tilbake med følgende punkter [[5, 5, 10], [5, 5, 10], [5, 5, 10]]. Gjennom å regne oss frem til en forventet verdi utifra medsendt punkter kan testen bevise om metoden fungerer.

9 PGL-39 Komprimerer til gamut i én dimensjon

Metoden skal kunne komprimere alle fargepunkter lineært gitt en akse og returnere et data objekt med de komprimerte punktene i samme format som punktene ble sendt med.

9.1 Teori

- **Likningen til en linje:** Én linje kan beskrives ved likningen $l(x) = bx + a$. Her er b stigningskoeffisienten, som betyr at fra $l(t)$ til $l(t + 1)$ endres funksjonverdien tilsvarende b (t er en vilkårlig konstant). Linjen *plasseres* utifra a verdien, som definerer hva som er funksjonsveriden til $l(0)$, altså hva som er linjens funksjons verdi rett over origo langs y-aksen.
- **Komprimering:** I dette kapittelet kan komprimering forklares på følgende måte. En koordinatverdi skal reduseres basert på funksjonen til en linje. Dersom det skal kompremiteres langs x-aksen og et punkt har x-koordinat har verdien 10, endrer vi punktets x-koordinat til $l(10)$.

9.2 Kravspesifikasjon

I dette avsnittet viser vi først oppgaven som den var gitt av oppdragsgiver og så hvordan vi tolket oppgaven.

9.2.1 Kravspesifikasjon fra oppdragsgiver

Metode i Gamut som komprimerer lineært til gamut i en utvalgt dimensjon i et utvalgt fargerom. Pseudokode for syntaks Listing 9.1

Listing 9.1: Der axis er aksen det skal komprimeres til – 0, 1, eller 2 – i det gitte fargerom, og d_compress er et nytt colour.data.Data-objekt.

```
1    g = colour.gamut.Gamut(...)
2    d = colour.data.Data(...)
3    d_compress = g.compress_axis(d, colour.space.XXX, axis=...)
```

9.2.2 Tolkning

Vi skal skrive en metode som tar tre parametere:

- Fargerom i form av colour.space.
- Punkter i form av et c_data objekt.
- Tall fra en til og med to som angir hvilken dimensjon som skal komprimeres.

Metoden skal komprimere alle fargepunkter i en av tre dimensjonene og returnere et c_data objekt som inneholder de komprimerte punktene i samme format som de originale punktene som ble sendt med til metoden.

9.3 Arbeidsprosess

Under planleggingsmøte med oppdragsgiver fikk vi en gode forklaringer rundt oppgaven og vi fikk tidlig en ide om hvordan vi kunne løse oppgaven. Rent matematisk hadde vi

vært borti liknende før og vi visste derfor at vi ikke behøvde å lete etter noen større eksterne algoritmer eller metoder. Vi begynte å tegne diverse tegninger på en tavle for å visualisere og diskuterte rundt oppgaven for å sikkre oss at vi ikke hadde missforstått noen detaljer. Deretter ble det utarbeidet pseudokode for å gjøre kode skrivingen kjappere og avklare eventuelle uklarheter som kunne oppstå rundt kodingen av oppgaven.

Den kodemessige siden av oppgaven var av den enklere sort. Det vi startet med var å kode en metode som finner de høyeste og laveste koordinatverdier for både gamut og de medsendte punktene tilhørende aksen som brukeren ønsker å komprimere langs. Med ønsket verdi mener vi her den aksen bruker ønsker å komprimere i. Hvis bruker ønsker å komprimere i y-retning sender han med akse=1, og vi finner størst og minst verdi i y-retning.

Etter at vi hadde skrevet pseudokoden begynte vi å lete etter metoder for å finne minste og høyeste verdier langs gitt akse. Vi fant numpy sine amin og amax metoder, men disse leter kun i en dimensjon og siden punktene vi sender med som oftest kommer i Nx3 format fungerer ikke disse metodene. Et koordinat i rommet består av tre tall som representerer X, Y og Z. Vi fant ingen metoder som gir høyest og lavest verdi kun blant X, Y eller Z koordinater i et datasett. Derfor skrev vi en egen liten kodesnutt som vist Listing 9.2. Koden leter kun i tuppelens ønskede data verdi og finner ut om verdien er høyere eller lavere en den til nå høyeste/laveste verdien.

Listing 9.2: Finner høyeste og laveste verdi langs gitt akse(ax) for punktene som skal bli komprimert.

```
1 for p in points:
2     if p[ax] > p_max:
3         p_max = p[ax]
4     elif p[ax] < p_min:
5         p_min = p[ax]
```

Vi har likningen for en linje $f(x) = bx + a$. Vi behøver derfor å regne ut b og a for å kunne finne de nye posisjonene for alle punkter. Vi regner derfor ut avstanden mellom laveste og høyeste verdi for gamut og de medsendte punktene fra nå av kjent som Δg og Δp. Vi kan deretter finne stigningstallet b for linjen, $b = \Delta g/\Delta p$. Den siste ukjente er nå a og finner vi ved å løse likningen $f(g_{min}) = b \cdot p_{min} + a$, $a = g_{min} - b \cdot p_{min}$

Nå som vi har alle likningens variable går vi gjennom alle punkter og endrer koordinatenes ønskede verdi til den komprimerte verdien. Dette høres ut som en større og kanskje komplisert operasjon men er i programmerings verdenen gjøres dette på en effektiv måte på kun to linjer kode som vist i kodesnutten Listing 9.3.

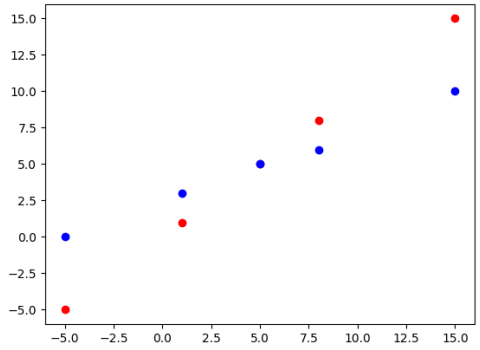
Listing 9.3: Endring av alle koordinater på en akse.

```
1 for i in range(0, points.shape[0]):
2     points[(i, ax)] = b*points[(i, ax)] + a axis
```

9.4 Resultat

Metoden compress_axis tar inn et fargerom, fargedata og en ønsket akse. Den vil så komprimere alle punkter lineært gitt høyeste og laveste verdi i gamuten langs den gitte akse. Figur 27 viser hvordan komprimeringen ser ut i praksis langs Y-aksen. De røde punktene er originalpunktene mens resultatet etter komprimeringen er de blå punktene. Figuren viser komprimering i et to dimensjonalt rom for å øke leseligheten da en lignende

figur i 3D ble meget uoversiktlig. Figuren illustrerer prinsipper veldig oversiktlig og under den vanligste bruken av metoden i tre dimensjoner hender dette altså for alle punkter langs en akse.



Figur 27: Originale punkter i rødt, og de komprimerte punktene i blått. Jo lenger det originale punktet er unna den lineære linjen desto mer blir punktet beveget i retning linjen.

9.5 Testing

Vi har testet hastigheten til metoden og ble overrasket da vi støtte på problemet med at minne på testmaskinen ikke holdt for 10M punkter. Det høyeste antall punkter vi har testet er derfor 7M og resultatet ble gitt etter gjennomsnittlig 38s. Testmaskinen har en Intel i5-6200U og 8GB minne.

Vi er fornøyde med resultatet da bilder på 7MP er reelt og metoden derfor kunne benyttes i en virkelig setting. Vi ser imidlertid et forbedringspotensiale rundt bruken av for-løkkene. Vi vil mot slutten av oppgaven gjøre re-faktorisering og da vurdere sammen med oppdragsgiver om det skal gjøres en forsøk på å effektivisere dette.

10 PGL-45 Nærmeste punkt på plan PGL-48 HPmindDE

Under PGL-45 skal metoden mappe alle fargepunkter til gamutens overflate gitt en vinkel definert av en akse. Aksen vil sammen med punktet lage et plan som benyttes til å finne nærmeste punktet.

Under PGL-48 tar metoden inn fargepunkter og mapper alle farger som ligger utenfor en enhetsgamut til nærmeste farge med samme fargetone i fargerommet CIELAB.

Vi har valgt å ikke skrive et eget kapittel for PGL-48, da dette var en enkel sak siden alt som måtte gjøres var å lage en metode og tar imot punktene og sender dem videre til clip_constant_angle() med CIELAB som setter fargeromsparameter. Dette kalles gjerne en wrappermetode.

10.1 Teori

- **Skjæring:** Der overflaten til to geometriske figurer overlapper.
- **Projeksjon:** Et punkts projeksjon på en linje er det nærmeste punktet på linjen til punktet. Vinkelen mellom linjen og projeksjonsvektoren er alltid 90 grader.

10.2 Kravspesifikasjon

I dette avsnittet viser vi først oppgaven som den var gitt av oppdragsgiver og så hvordan vi tolket oppgaven.

10.2.1 Kravspesifikasjon fra oppdragsgiver

Metoden Gamut.clip_constant_angle(...) skal i et gitt fargerom mappe alle fargene som ligger utenfor en gamut i et datasett til nærmeste farge på gamutoverflaten med samme vinkel til en gitt akse, altså i planet som består av fargen selv og den gitte aksen (som typisk vil være gråaksen). Siden det er litt forskjellig i forskjellige fargerom hvilken av koordinataksene som er gråaksen, må dette være en parameter til metoden. Det enkleste her er å benytte en tallparameter (0, 1, 2) som henviser til aksen. Metoden skal kunne benyttes slik. Pseudokode for syntaks Listing 10.1

Listing 10.1: Der axis sette til 0 og sender med fargerom, colour.data.Data punkter og axis til metoden clipped_im

```
1    axis = 0 # aksen vi regner vinkelen med
2    im = colour.data.Data(...)
3    gamut = colour.gamut.Gamut(...)
4    clipped_im = gamut.clip_constant_angle(sp, im, axis)
```

10.2.2 Tolkning

Oppgaven går ut på å finne nærmeste punkt på gamutens overfalte dersom et punkt Q er utenfor gamuten. Vi skal begrense søket til den delen av gamuten som skjæres av et plan definert av Q og en gitt koordinatakse.

En heltallparameter sendes med til metoden for å definere hvilken koordinatakse som skal benyttes for klipping. Det sendes også med punkter i form av en colour.data.Data objekt, der hvert punkt kan være innenfor eller utenfor gamuten. En parameter som definerer hvilket fargerom det skal beregnes i, i form av et colour.space objekt. Metoden skal returnere punktene i et colour.data.Data objekt der indeksene til punktene er bevart, mens punkter som er utenfor gamuten har blitt endret til nærmeste i gamuten og planet.

10.3 Arbeidsprosess

For å sikre at oppgaven var korrekt forstått, drøftet programeringsparet rundt oppgaven og benyttet en whiteboardtavle for å skissere opp og lage figurer rundt konseptene vi måtte løse. Et av hovedproblemene i denne oppgaven er å finne en løsning for å beregne den linjen som skjærer mellom et plan og et polyeder. Deretter må vi beregne hvilket punkt på den aktuelle skjæringslinjen som er nærmest Q.

Litteratursøk presenterte ingen komplette løsninger på oppgaven, men vi fant en effektiv algoritme for å finne skjæringen mellom et plan og en trekant [36] . Da gamutens overflater er konstruert av trekanter kan dette brukes til å bryte ned oppgaven til å iterere over fasettene og for hver av dem finne linjestykket for skjæring med planet, og til slutt finne nærmeste punkt på linjestykket.

For å effektivisere løsningen, idemyldret vi oss frem til kriterier for hvilke fasetter vi umiddelbart kunne utelukke fra søket. Dette resulterte i tre idéen. Den første idéen var å utelukke fasetter som er på motsatt halvdel av gamuten i forhold til Q. Den andre idéen var å utelukke fasetter som ikke skjærer planet. Den siste idéen var å utelukke fasetter hvor alle hjørnene er lengre unna en det foreløpig nærmeste punktet.

Før implementasjon av løsningen begynte hadde programmeringsparet et møte med Ivar for å diskutere løsningsstrategien. Møtet bekreftet at løsningen ville fungere, men vi hadde en lengre diskusjon rundt den siste effektiviseringsidéen. Det ble belyst tilfeller hvor denne testen kunne feile, men på bakgrunn av gamuters generelle form ble antagelsen om at denne testen ville fungere gjort.

Et førsteutkast av implementasjon ble utviklet, og en god løsning for næreste punkt på linjestykket fant vi på stackoverflow.com [37]. Under testing av denne implementasjonen så vi at den feilet når Q var kolinjær med aksen. Dette ble løst vi å sjekke for kolinjæritet, og tilkalle clip_nearest i disse tilfellene.

10.4 Resultat

Resultatet av denne oppgaven er metodene clip_constant_angle() og _clip_constant_angle(). Den første metoden tar imot et sett med punkter med format Mx...xNx3 og konverterer disse til ønsket fargerom. Disse punktene omtales videre som c_data. Deretter brukes is_inside() til å beregne hvilke punkter som er utenfor gamuten. Is_inside returnerer en boolsk array med format Mx...xN, der resultatet på om et punkt et utenfor eller innenfor ligger lagret med samme indeks som i punktets indeks i c_data. Dermed kan vi iterere over den boolske arrayen, og når en celles verdi er False endrer vi det korresponderende punktet i c_data til resultatet fra _clip_constant_angle() med det gitte punktet som Q parameter.

Listing 10.2: Pseudokode for PGL-45/48

```
1   Hvis Q ligger på aksen:
```

```
2       RETURNER clip_nearest(Q).
3
4     P = Beregn plan(Akse, Q)
5
6     Nærmeste = NULL
7     For alle fasetter i gamuten:
8
9        Om Dot(Q-Center, Vertex,0-Center) er negativ:
10          Hopp over fasett.
11
12       Above = Bergen punkter over P.
13       Below = Beregn punkter under P.
14
15       Om Above eller Below er tom:
16          Hopp over fasett.
17
18       V, W = Endtepunkter for Skjæringslinje mellom fasett og P
19
20       Om V er lik W:
21     Kandidat = V
22
23       Ellers:
24    Beregn t for Q prosjektert på linjen V + t*(W-V)
25
26     Om t er mindre enn 0:
27        Kandidat = V
28      Om t er større enn 1:
29        Kandidat = W
30     Om t er mellom 0 og 1:
31        Kandidat = V + t*(W-V)
32
33    Om Kandidat er nærmere Q enn, Nærmeste
34        Nærmeste = Kandidat
35
36    RETURNER Nærmeste.
```

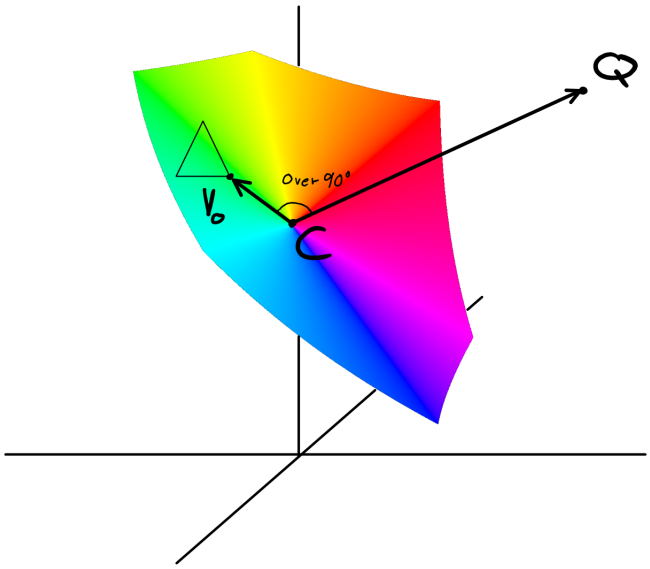
I Listing 10.2 vises pseudokoden for metoden _clip_constant_angle(). Linje 1 og 2 håndterer tilfeller der Q er kolinjær med aksen. Dette må gjøres fordi metoden ellers benytter seg av planet definert av aksen og Q, og tre punkter på en linje definerer ikke et plan. For å beregne planet benytter vi oss av find_plane() som tidligere ble utviklet i PGL-37.

I linje 6 oppretter vi *Nærmeste* som vil inneholde det nærmeste punktet til Q langs skjæringen mellom planet og gamuten. Vi starter så iterasjonen over alle gamuten fasetter hvor vi først har to avskjæringstester for å øke metodens effektivitet. Den første på linje 8 hopper over alle fasetter som er på feil halvdel av gamuten, illustrert i Figur 28.

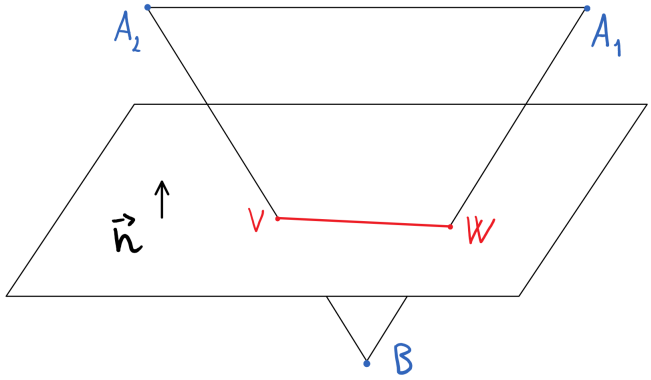
Den andre avskjæringstesten hopper over fasetter som ikke skjærer planet. For hver av fasettens hjørner tar vi dotproduktet mellom planets normalvektor og vektoren til det aktuelle hjørnet. Dersom dotproduktet et positivt (eller null) legges hjørnet til "Above"-listen, dersom dotproduktet er negativt legges hjørnet til Below"-listen. For at fasetten skal skjære planet må den ha punkter både over og under planet.

Neste steg er å finne endepunktene for skjæringslinjen mellom fasetten og planet, V og W. Se Figur 29. Her er det to punkter, A_1 og A_2 i above-listen,og et punkt , B, i below-listen. Vi vet da at V er på den parameteriserte linjen $\vec{B} + s \cdot (\vec{A_1} - \vec{B})$. Verdien av s som viser til V langs linjen beregnes ved $\mathrm{dot}(n, (\vec{P} - \vec{B}))/(\vec{n} \cdot (\vec{A_1} - \vec{B}))$. Tilsvarende beregning gjøres for W langs linjen $\vec{B} + t \cdot (\vec{A_2} - \vec{B})$. Metoden beskrevet her fungerer for plan som går gjennom origo.

Når vi har linjestykket mellom V og W, finner vi det nærmeste punktet til Q på linje-



Figur 28: Fasetten med første hjørne Vo kan utelukkes fra søket etter nærmeste punkt til Q.



Figur 29: Fasetten med hjørner A1, A2 og skjærer planet mellom V og W.

stykket og lagrer denne i *Kandidat*. Et spesialtilfelle hvor fasetten kun skjærer planet i et punkt håndteres på linje 14 og 15. Ellers finner vi *Kandidat* ved Q sin projeksjon på den parameteriserte linjer $\vec{V} + t \cdot (\vec{W} - \vec{V})$. Vi finner t med $\text{dot}(\vec{Q} - \vec{V}, \vec{W} - \vec{V})/|\vec{W} - \vec{V}|^2$. Dersom t er mindre enn 0 er projeksjonen utenfor linjestykket og V er den beste kandidaten. Dersom t er større enn 1 er projeksjonen utenfor linjestykket og W er den beste kandidaten. Dersom t er mellom 0 og 1 er den beste kandidaten $\vec{V} + t \cdot (\vec{W} - \vec{V})$.

Distansen fra Q til *Nærmeste* sammenlignes med distansen fra Q til *Kandidat*, og *Nærmeste* settes lik *Kandidat* om den er nærere Q. Når vi har itterert gjennom alle gamutens fasetter inneholder *Nærmeste* det nærmeste punktet til Q på gamuten og planet.

10.5 Testing

For å testing satte vi opp en punktliste med 4 punkter utenfor gamuten, der et av de var på aksen, samt et punkt innenfor gamuten. Vi beregnet en fasitliste for hånd, og brukte

en assert-funksjon for å kontrollere at returverdien fra HPminDE() med punktlisten som parameter stemte overens med fasitlisten.

11 PGL-61 minDE

Metoden tar inn fargepunkter og mapper alle farger som ligger utenfor en enhetsgamut til nærmeste farge i fargerommet CIELAB.

11.1 Teori

Teorien for denne er basert på PGL-37 og PGL-36. se Avsnitt 6 og Avsnitt 8

11.2 Kravspesifikasjon

I dette avsnittet viser vi først oppgaven som den var gitt av oppdragsgiver og så hvordan vi tolket oppgaven.

11.2.1 Kravspesifikasjon fra oppdragsgiver

MinDE skal ta inn et bilde (som colour.data.Data objekt) og returnere bildet gamut-klippet til nærmeste farge i CIELAB. Det vil si at alle fargene som er inne i gamut (i CIELAB) skal forbli uendret, og alle som er utenfor skal erstattes med nærmeste farge på gamut-overflaten. Pseudokode for syntaks Listing 11.1

Listing 11.1: Der im er colour.data.Data punkter og g er gamut objektet.

```
1    im = colour.data.Data(...)
2    g = colour.gamut.Gamut(...)
3    mapped_im = g.minDE(im)
```

11.2.2 Tolkning

Metoden går ut på å kombinere metoder fra tidligere PGL i en mindre metoden for å utføre en spesifikk oppgave. Metoden må kombinere clip_nearest fra PGL 37 og is_innside fra PGL 36. Metoden skal ta imot en liste med datapunkter i fargerommet cialab og sjekke hvilken som er innenfor og hvilken som er utenfor gamuten og finne nærmeste punkt på gamut for de som er utenfor og bytte koordinatene med de nye som er funnet.

11.3 Arbeidsprosess

Ettersom vi har mye av hovedmetodene til denne metoden allerede laget fra før i tidligere PGLer som nevnt i Avsnitt 11.1 krevde ikke den mye forarbeid og litteratursøk, men vi trengte og gå tilbake til disse metodene som vi har laget før og lese gjennom de og dobbelt sjekke at de faktisk passer inn i denne metoden.

minDE metoden skal kunne ta imot en liste(array) med punkter som er både innenfor og utenfor en gamut og de skal være i fargerommet CIELAB. Først må metoden finne ut hvilken punkter som er innenfor gamut og hvilken som er utenfor gamut. Det er et veldig viktig steg for å kunne skille ut hvilken som er utenfor og kun bruke de punktene til å finne nærmeste punkt på gamut.

Vi utnytter en motode som heter is_inside() som vi har laget tidligere for Feito-Torres metoden under Avsnitt 6.

Metoden går gjennom alle punkter som vi får tilsendt og sjekker hvilken som er innenfor og returnerer resultat som en liste også, men listen inneholder sant og usant. Det vil si at for punker som er utenfor vil det være usant og sant for de som er innenfor. Dermed kan vi sammenligne de to listene vi har og vite akkurat hvilke som er utenfor og hvilken som er innenfor.

Ettersom vi tidligere i Avsnitt 8 laget en metoden for å finne nærmeste verdi for et punkt kan vi utnytte den her også.

Vi bruker alle punker som er usant og sender med videre til clip_nearest metoden for å sjekke nærmeste punkt. Etter at vi har gått gjennom alle punker settes de inn i liste igjen som vi fikk fra bruker. Det vil si at vi bytter ut verdien for punktene som er utenfor med nærmeste punkt på gamuten og sender verdiene tilbake til bruker.

11.4 Resultat

Metoden utnytter to allerede eksisterende og ferdig testet metoder for å finne nærmeste punkter. Dermed kan vi finne nærmeste punkt på gamut for hver punkt utenfor med 100% nøyaktig i forhold til verdiene vi har i testen og bytte ut verdien med nærmeste punkt på gamut. Se Vedlegg Q for testverdier.

11.5 Testing

For å teste metoden genererte vi en kule som har datapunkt mindre enn 10 for så å sende den til metoden. Deretter sjekker vi verdiene som blir returnert fra metoden også sjekker vi om punktene som er returnert er mindre enn 10 og stemmer med forventet resultat.

12 Kvalitetssikring

12.1 Dokumentasjon

I løpet av utviklingsprosessen skal alle gruppemedlemmer skrive dokumentasjon under utviklingen av biblioteket for kode, teorier osv. Vi dokumentere hver eneste PGL som vi jobber med underveis, og etter hver PGL er ferdig skal paret som jobbet med denne gjøre ferdig dokumentasjonen og sørge for å tilfredsstille malen som vi har utarbeidet underveis i utviklingen. For dokumentasjon av PGL har vi en guide som vi følger for dokumentasjonen. Alt dokumentasjon om en PGL skal være ferdig før paret starter med nytt PGL, det gjøres for å forsikre oss om at informasjon ikke blir glemt underveis og at jobben det blir gjort så godt som mulig.

Det er tre viktige programmer som vi dokumenterte i. JIRA bruker vi for registrering av timer og for å holde oversikt over alle PGL-er og hvem som har ansvar for hvilken PGL. I tillegg gir JIRA oss gode statistikker og diagrammer for hvor mye tid som har blitt brukt og hvor mye har blitt utført. I Bitbucket lagres alt av kode som blir skrevet, Bitbucket gir oss backup muligheter og versjonskontroll er nødvendig om feil oppstår og/eller vi ønsker å gjenopprette en eldre versjon av koden. I tillegg har vi kodedokumentasjon i form av Wiki der alt av bruker manualer og kodebeskrivelse lagres. For rapportskriving bruker vi Sharelatex ettersom alle da kan skrive sammen i sanntid og være mye mer effektive sammen. Alt av rapportskriving er lagret i Sharelatex og den hjelper oss med å generere PDF for sluttrapporten.

Ved enden av utviklings tiden har vi en dokumentasjon periode som er dedikert til kun rapport skriving og ferdig stilling. I denne perioden kombinerer vi PGL dokumentasjon med hovedrapporten og syr det sammen. Hele perioden skal kun brukes til å skrive rapporten og skrive om alle tema, lage illustrasjoner, skisser og alt annet som skal inn i rapporten.

Samtidig som vi jobber med utvikling og dokumenterer PGL-er skal vi også skrive rapport. Det vil si at vi skriver om utviklingsprosessen, teori, framgangs metode og andre relevante tema underveis for å utnytte tiden mest mulig fram til avslutningen av rapportskriving perioden. Gruppen fordeler tema for hver person i gruppen, her jobber vi individuelt og ikke i par som i utviklingen og PGL rapportering. Vi velger et tema eller får tildelt en fra gruppeleder og jobber med den under utviklings perioden.

For at vi i gruppen skal sørge for å levere god og høy kvalitets rapport så må alt som blir skrevet, under hver tema bli lest over og gjennomgått av en annen gruppemedlem. Det er for å utelukke skrivefeil, slurv og andre feil i teksten som kan påvirke kvaliteten av rapporten.

12.2 Scrum

I dette kapittelet ønsker vi å belyse ytterligere hvordan vi har gjennomført hver scrum syklus og hvorvidt vi mener det har ført til et bedre miljø innad i prosjektgruppen samt om vi mener det har vært en del av kvalitetssikringen vi i forprosjektet mente at det ville

være. Vi vil også reflektere rundt eventuelle forbedringer vi mener kunne vært aktuelle for en bachelorgruppe, og hvorvidt utviklingsmodellen er fungerende for et bachelorprosjekt.

12.2.1 Planning poker

Før hver sprintstart holdt vi et planleggingsmøte, på dette møte var alle utviklerne, oppdragsgiver og veileder tilstede. Det forekom at veileder ikke var tilstede på disse møtene, men som oftest var alle tilstede. Møtet ble ledet av scrum-master og målet med dette møte var å planlegge påfølgende sprint ved å estimere tidsbruk for oppgaver oppdragsgiver har prioritert. Som nevnt i forrapporten var den ukesmessige arbeidsmengde for bachelorprosjektet for hver utvikler 30 timer. Av disse 30 timene var 10 timer satt av til rapportskriving og de resterende 20 timer til utvikling og dokumentasjon. Dette betyr at vi hadde 160 timer med arbeid å fordele på den forestående sprint. Oppdragsgiver lagde en liste med prioriterte oppgaver, disse var altså hans subjektive ønsker for funksjonalitet i biblioteket. Det neste steget var da å estimere tidsbruken for hver av disse oppgavene.

Metoden vi benyttet for estimering var planning poker. Hver av utviklerne samt oppdragsgiver og veileder fikk en liten bunke med kort med tallverdier på. Disse verdiene representerte antall arbeidstimer den enkelte estimerte en oppgaven ville ta. Oppdragsgiver presenterte en oppgave, gruppen diskuterte rundt oppgaven og deretter la alle sin subjektive estimering av oppgaven med tallverdien(ansikt ned) på bordet. Når alle hadde lagt sitt kort ned ble alle kortene vist samtidig. Så diskuterte personen med den laveste verdien med personen som la den høyeste verdien for å belyse meningene rundt forskjellen i estimeringen. Deretter fikk alle muligheten til å endre sin estimering, vi tok så gjennomsnittet av alle estimeringene og satte dette som estimert tidsbruk for oppgaven i Jira (Vedlegg L). Hvis det var oppgaver som ikke ble ferdigstilt på en foregående sprint, eller oppgaver som var estimert men ikke valgt ut til en tidligere sprint ble disse estimert på nytt ved behov på samme vis som nye oppgaver.

Etter at alle estimeringene var gjort fikk oppdragsgiver muligheten til å velge ut de oppgavene han ønsket som sprintens mål. Det ble valgt ut oppgaver med en samlet estimert timesverdi på ca. 160 arbeidstimer. Det ble tatt hensyn utviklernes ønske når det kom til utvalg av arbeidsoppgaver, spesielt når det gjaldt ferdigstilling av oppgaver fra en foregående sprint.

12.2.2 Review meeting

Når så en sprint nærmet seg slutten forberedte utviklerne seg på å presentere alle påbegynte og ferdigstilte oppgaver for oppdragsgiver og de resterende gruppemedlemmene på et review-møte (Vedlegg K). Deltagerne var de samme som på planleggingsmøtet da review-møtet blir holdt som en avsluttende del av en sprint og rett før planning-møte til neste sprint. Hensikten med review-møte var at alle i prosjektgruppen skulle skape seg et overblikk over prosjektet. Særlig viktig var det at oppdragsgiver skulle få en oppdatering på hva som var blitt gjort og hvordan prosjektet som helhet lå an, samt om utviklerne hadde kommet i havn med det satte sprint-målet. På dette møtet ble det også avgjort om kvalitet og dokumentasjon til den enkelte oppgaven var god nok til å kunne gis videre til brukertesting og integrering med de andre ferdigstilte oppgavene.

Presentasjonene ble holdt på en uformell måte og tavle eller prosjektor ble kun brukt ved behov. Under dette møte ville oppdragsgiver påpeke både små og større forbedrings-

forslag på metoder. Den fremførende utvikler ville først forklare hva oppgaven var og hvordan vi hadde tolket den, så gjennomgå metodene som var utviklet og svare på spørsmål sammen med programmerings-partneren som deltok i utviklingen. Til slutt ble tester og eventuelle grafer gjennomgått. Møtet ble hevet når alle oppgaver var gjennomgått og spørsmål var besvart.

12.2.3 Retrospective meeting

Etter at både review- og planning-møtene var avsluttet, kunne utviklerne starte det såkalte retrospective-møte. Dette møtet skulle kun holdes mellom scrum-master og utviklerne, og hadde som oppgave å se tilbake på sprint forløpet og samspill mellom utviklerne. Hvis det hadde oppstått noen konflikter i løpet av sprinten skulle disse reflekteres over og diskuteres av de involverte og resten av utviklerne. Det var viktig å reflektere over hvordan sprinten hadde gått, hva hadde vi gjort riktig, hva som ikke hadde gått som forventet og hvorfor. Se referater fra disse møtene i Vedlegg J

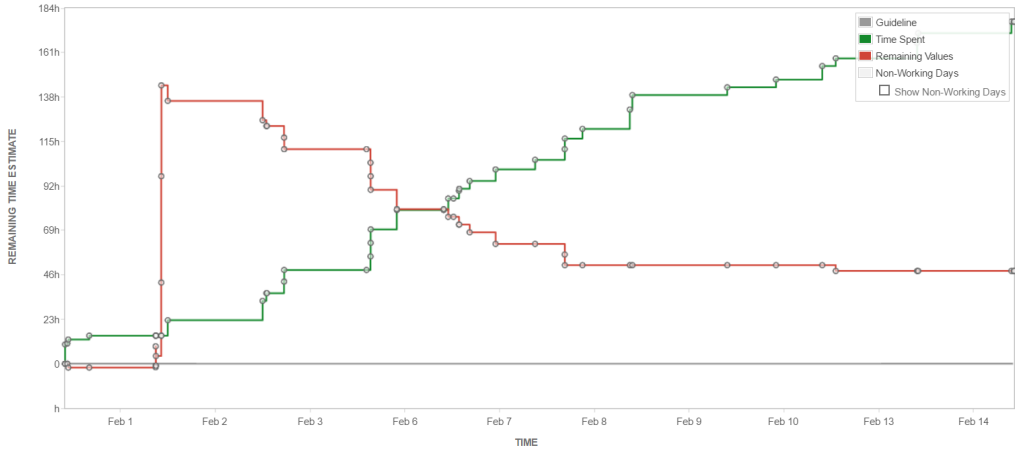
12.2.4 Gir scrum økt kvalitetssikring i et bachelorprosjekt?

Ingen i prosjektgruppen hadde noen tidligere erfaring med den praktiske siden av scrum. Oppdragsgiver var den eneste i prosjektgruppen som hadde bred erfaring med utvikling i Python. Oppdragsgiver og veileder hadde begge meget god kompetanse om den faglige siden av prosjektet, og dette hjalp med den erfaringsbaserte estimeringen. Uerfarenheten til utviklerne viste seg godt under de første sprintene i form av estimeringene. Det må tas i betraktning at både programmeringsspråket og fagfeltet var meget nytt, selv om utviklerne har hatt en innføring i programmeringsspråket, viste det seg at det å finne metoder i eksisterende bibliotek og optimalisering tok mye tid. Utviklingen i estimeringen var imponerende. Vi kan tydelig se at det var en stor forbedring fra starten av utviklingen der vi estimerte tidsbruken til å være mye lavere enn det vi endte opp med å benytte, se Figur 30.

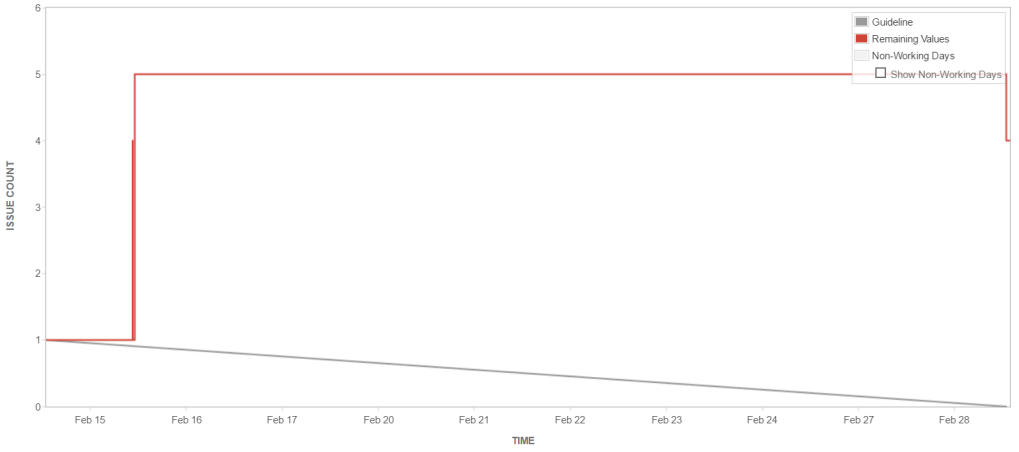
I sprint to startet vi med fem oppgaver der noen var grunnleggende modulstrukturering, mens andre var de største og mest komplekse kjernemetodene som PGL-36 kapittel 6. Som Figur 31 viser, kom vi ikke i nærheten av å nå målet med antall ferdigstilte oppgaver, men vi kom likevel langt på de individuelle oppgavene, og tilegnet oss mye kunnskap rundt bruken av biblioteket og fargevitenskapen bak oppgavene.

I løpet av utviklingssprint tre fokuserte utviklerne først på å ferdigstille de oppgaver som ikke ble gjort ferdig i sprinten før. Det viste seg at erfaringene fra sprint en og to gjorde arbeidet merkbart lettere for utviklerne. Vi hadde satt oss godt inn i bibliotekets andre klasser og metoder, vi visste hva og hvordan vi skulle ferdigskille oppgavene vi hadde valgt ut til sprinten og dette gjorde at gruppen kom i mål mye fortere denne gangen, som Figur 32 viser.

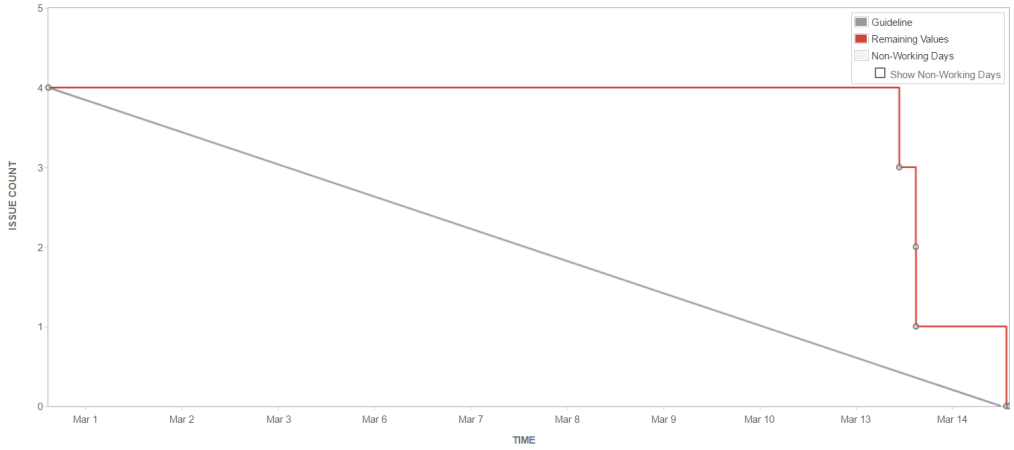
Vi har benyttet erfaringsbasert estimering og gradvis sett at estimeringene har blitt mer nøyaktige med tiden og gitt oppdragsgiver og utviklerne mer sikkerhet rundt leveranser og arbeidsmetodikk. Vi har sett at estimeringen har vært svært påvirket av erfaringene vi har samlet underveis i prosjektet. Dette kan være en grunn til å ikke velge scrum i et bachelorprosjekt da utviklerne mangler den kompetansen som behøves for å gi et nøyaktigere estimat. Samtidig har vi gjennom å benytte scrum-modellen blitt godt kjent med en utviklingsmodell som benyttes i mange store selskaper i dag [38] og har samlet mange erfaringer rundt møte strukturene og estimerings prosessen gjennom å



Figur 30: Tidsestimeringen for PGLer i sprint en. I rødt er estimert tid som gjenstår på oppgaver, i grønt er den reelle tidsbruken vi har loggført. Loggført tid på sprinten er 180 timer, dermed har utviklerne brukt 20 timer mer på sprinten. På grunn av en teknisk feil ble estimater ikke automatisk overført fra planleggingsfasen, de ble lagt inn manuelt en dag etter sprint start. Derfor starter grafen med estimert tidsbruk lik null.



Figur 31: Figuren viser antall oppgaver fra sprint to. Det oppsto en teknisk feil i starten av sprinten, derfor startet sprinten kun med en oppgave og fire ble lagt til kort etter start. Gruppen feilestimerte grovt og endte sprinten med kun en ferdigstilt oppgave.



Figur 32: Figuren viser antall oppgaver fra sprint tre. Den bratte avslutningen på grafen skyldes at oppgavene ikke ble satt til ferdig men forble på ut-sjekk i Jira frem til mot slutten av sprinten. Prosjektgruppen nådde sprintmålet og ferdigstilte dermed alle oppgavene i sprinten.

utføre planning poker som det gjøres i arbeidslivet.

Bruken av en scrum-master har for gruppen vært praktisk fordi gruppen kunne legge mye av det administrative arbeidet på en pålitelig person som da fikk det overordnede ansvaret for fordeling av arbeid, arrangering av møter og ledelse under møter. Det har oppstått situasjoner der scrum-master har avgjort det videre forløpet som har gjort at gruppen kommer seg videre i stedet for å bruke for mye tid på en problemstilling. Bruken av retrospektive-møtene har også vært viktige for gruppen. Spesielt i starten av utviklingen har det vært en god metode for å samle alle gruppens meninger og opplevelser av situasjoner spesielt rundt utviklingsprosess, kommunikasjon og samarbeid.

Daglige korte møter (Vedlegg I) enten i et møterom eller over Skype gjør at scrum-master og alle utviklere vet hva som er gjort, status på hver oppgave og hva som er ferdig. På disse møtene kan også generelle beskjeder tas opp og gruppen kan diskutere med hverandre rundt oppgaver hvis det er uklarheter, og det kan fordeles nye arbeidsoppgaver hvis det behøves. Retrospective-møtene var en god måte for oppdragsgiver, veileder og utviklerne til å holde seg oppdatert på hva som var gjort under en sprint, om kvaliteten var god nok og om sprintmålet var møtt.

Review-møte var en mulighet for utviklerne å ytre meninger om arbeidsprosess og interne konflikter, og det kan besluttes tiltak rundt endringer av regler eller arbeidsmetode.

For å oppsummere kan vi si at, scrum har vært behjelpelig med kvalitetssikring fordi scrum tilrettelegger for en god planleggingsfase der sprintene har en fast lengde og utviklerne kan konsentrere seg på å arbeide med det som er satt av arbeidsoppgaver ved sprintstart. På grunnlag av hvordan prosjektgruppen har anvendt scrum, med de modifiseringene vi har gjort, mener vi at modellen gir økt kvalitetssikring og vi kan anbefale denne modellen til bruk under bachelorprosjekter.

12.2.5 Fordeling av arbeid

Fordeling av arbeid har i grunn vært en demokratisk prosess i gruppen både for rapportarbeid og utviklings oppgaver. Det var prosjektleders ansvar å ha arbeidsoppgaver klare for gruppemedlemmer til enhver tid og legge disse inn i et Trello-brett. Prosjektleder kom med en anbefaling for hvert medlem utfra hva som måtte gjøres og medlemmets sterke sider og hver av gruppemedlemmene kunne så velge seg ut den oppgaven de ønsket. Programmeringsparene som jobbet med en PGL hadde ansvaret for å dokumentere denne i rapporten. Etter planleggingen av en sprint ble alle PGLer lagt inn i Jira og etter retrospective-møtet ble gruppens medlemmer enige om hvilke oppgaver hvilket par skulle arbeide med. Som beskrevet under kvalitetssikring og scrum, ble de daglige-møter benyttet til å fordele arbeid i løpet av sprintene.

12.3 Enhetstesting

I vårt prosjekt benyttet vi enhetstesting[39]. Enhetstesting er en del av Python 3 og vi benyttet siste versjon 3.6 til videreutviklingen av biblioteket. Enhetstesting ga oss muligheten til å kunne lage separate tester for hver enkel modul og metode som er i modulen, noe som ga oss mye fleksibilitet til å skape test metoder og teste ut på mange forskjellige måter.

Den minste testbare delen av et program kan man definere som en enhet. I prosessorprogrammering kan en enhet være en hel modul, men det er oftest en individuell metode eller prosedyre. I objektorientert programmering er en enhet ofte et helt grensesnitt, for eksempel en klasse, men det kan også være en individuell metode. Enhetstester er kortkodefragmenter opprettet av programmerere eller av og til av testere under utviklingsprosessen. Det danner grunnlaget for komponenttesting.

Hvit-boks testing[40] er når man tester fra innsiden og har tilgang til internkoden og kan teste den, mens svart-boks testing[41] er når man tester programmet fra utsiden og ikke har tilgang til kildekoden.

Ideelt sett er hver test del uavhengig av de andre. Enhetstester skrives vanligvis og drives av programvareutviklere for å sikre at koden oppfyller sin design og oppfører seg som beregnet. Fordi en metode til tider kan ha referanser til flere klasser og metoder, kan testing ofte gå utover flere klasser og metoder.

I koden Listing 12.1 under har vi et Python kode som utfører et enkelt regnestykke.

Listing 12.1: Python eksempel

```
1 class AdderImpl:
2
3     def multiply(self, a,  b):
4         result = int(a) * int(b)
5         return result
```

Eksemplet Listing 12.1 under viser hvordan vi kan teste ut ved hjelp av Unittesting. I testen ser vi at vi lager en test for hver regnestykke som vi ønsker å prøve. Det er for å gjøre jobben så oversiktlig som mulig og for å kunne lett identifisere hvor feilen er. Det er en av fordelene med slik testing at man lager mange små individuelle tester som er lett å oppdage feil i, istedenfor store komplekse tester som gjør det vanskelig for utviklere å finne hvor feilen er.

Listing 12.2: Unittest eksempel

```
1
2 class TestUM(unittest.TestCase):
3     def test_numbers_3_4(self):
4         self.assertEqual( multiply(3,4), 12)
5
6     def test_strings_a_3(self):
7         self.assertEqual( multiply('a',3), 'aaa')
```

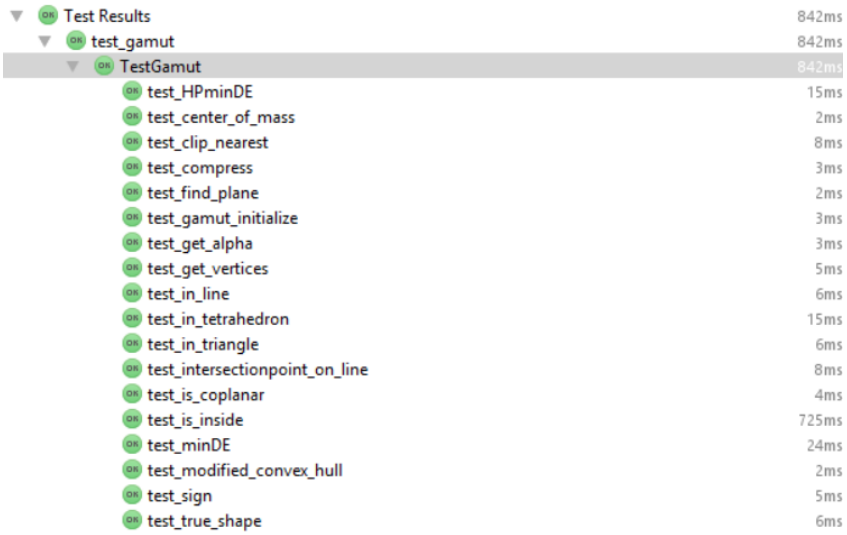
12.4 Testmodul

For å skape best mulig testmiljø har vi opprettet en separat test pakke som inneholder en test for hver modul i Colour pakken. Vi har ansvaret for å lage tester i gamut modulen som tester alle metoder som vi har ansvar for i dette bachelor prosjektet. Alle andre moduler som er laget for å kunne teste resten av Colour pakken har fargelaben og Ivar Farup ansvar for.

Test modulen for gamut er bygget opp for å kunne teste hver metode som er implementert i gamut modulen og sørge for at metoder fungerer optimalt og i henhold til kravspesifikasjonen.

Hver test metode vil opprette en gamut og kjøre de metodene som er knyttet til den spesifikke testen. Det er et mål at testen skal utløse alle mulige utfall som kan forekomme i metodene. Enhetstest pakken har egne metoder for å sammenligne en returverdi fra metoder med en forventet verdi.

Gjennom nøye testing sørger vi for at hver metode kjører som forventet og gir korrekte verdier. Det er viktig for at vi skal kunne utelukke mest mulig feil i vårt bibliotek. Figur 33 viser alle tester, samt tiden de bruker på å verifisere at koden de tester fungerer som den skal.



Figur 33: Alle tester som har blitt skrevet for å teste gamut modulen. Grønn farge på tester betyr at testen har passert. Tallet til høyre er tiden målt i millisekunder det tar for testen å teste koden.

12.5 Parprogramering

Vi har benyttet parprogrammering, der arbeidsmetodikken legger til rette for at to utviklere skal kunne sitte sammen og utvikle kode. Det skal være fokus på én PC-skjerm hele tiden når man parprogrammer. Framgangsmåte vi har benyttet når vi parprogrammerte:

Starte med å skrive/skissere hva vi har tenkt å gjøre før vi begynner å kode. Lag pseudokode og diskuter om fremgangsmåten på løse oppgaven. Være sikre på at vi forstår hverandre og tenker likt, for å kunne delta aktivt med koding. En av utviklere skriver kode, mens den andre observerer. Observatør sin jobb er å lese og vurdere: feil, uleselighet, forbedringspotensialet. Alltid viktig å snakke med observatøren for å sjekke om han henger med. Vi tar oss tid til å feire etter ha fullført en test. Bytt rolle ofte, minst hver time.

12.6 Versjonkontroll

Under begrepet versjonskontroll legger vi følgende: Et system for å loggføre endringer gjort i en eller flere filer, derunder muligheten til å se endringene, hvem som har gjort dem og når. Det skal også være mulighet til å tilbakestille filer til en tidligere loggført tilstand. [42]

12.6.1 Kode

Som beskrevet i forrapporten Vedlegg B, kapittel(5.3) har vi valgt å benytte git sammen med bitbucket som versjonskontroll verktøy for alt kode og tilhørende dokumentasjon. Alle utviklere har under utviklingsprosessen benyttet dette verktøyet for å holde orden på kode og dokumentasjon som ble skrevet.

På starten av utviklingen jobbet alle utviklere direkte på originalfilen. Da vi benyttet git gikk dette i utgangspunktet greit, men vi la merke til at det ble veldig mye merging og det oppsto ofte duplikater av metoder og data i filene. Vi bestemte oss derfor for å finne en løsning på dette rotet som oppstod. Løsningen ble at vi innførte bruken av grener(branch)[43] ved begynnelsen av den andre sprinten. Grener gir mulighet til å ta en kopi av en fil og arbeide med denne kopien i stedet for originalen. Når man er ferdig med arbeidet kan man da merge kopien med originalen. Dette gjøres i realiteten på hele prosjektet om gangen, man jobber derfor ikke bare på en kopi av en fil, men en kopi av hele prosjektet. Da utviklerne som oftest jobber med flere filer av prosjektet om gangen og behøver alle filer til å kunne teste den skrevne koden, er dette en veldig nyttig funksjonalitet å ha.

På denne måten kan et programmeringspar jobbe med en kopi av prosjektet mens de andre jobber på sin kopi, og så når oppgavene er ferdige kan slås alle endringene gjort i kopiene slås sammen med den originale(master) -grenen. Dette sikrer også integriteten til mester grenen, fordi det alltid kontrolleres nøye at det som slås sammen med denne grenen er fungerende og har passert kvalitetskontrollen. Vi opprettet en gren for hver PGL i Jira, og prosjektleder hadde ansvaret for å slå disse sammen med mester -grenen kun hvis koden og tester, samt dokumentasjonen, var godkjent. Etter opprettelsen av grener møtte vi på veldig få problemer med duplisering av kode, og liknende.

12.6.2 Rapportskriving

Rapportskrivingen ble gjort i ShareLaTeX. Den har også funksjonalitet for versjonskontroll i denne teksteditoren[44]. Det gjør det mulig å spore endringer på likt nivå som git og, man kan tilbakestille dokumentet til en ønsket loggført tilstand. I tillegg til å se av hvem og når endringer er gjort. Vi støtte kun på et fåtall av tilfeller der dette var nødvendig, og disse endringene gjaldt tekst som var blitt slettet, men ikke skulle ha blitt det. Bruken av tilbakestillingen sparte oss en del tid tilsammen og vi er veldig glade for at vi hadde denne muligheten.

Etter prosjektstart kom det en oppdatering til ShareLaTex som integrerte et kommenteringsverktøy i teksteditoren [45]. Denne nye funksjonaliteten var helt nydelig for oss som en gruppe. Den gjorde det mulig å foreslå avsnitt, foreslå sletting av tekst, kommentere på tekst, svare på kommentering og mye mer. Dette er meget nyttig for en gruppe som skriver på et dokument. Når en av medlemmene leser et avsnitt og mener noe mangler, eller har forslag til omformulering, kan han i stedet for å gjøre endringen foreslå denne. Så kan hele gruppen avgjøre i felleskap om det er en god endring eller ikke. Alle gruppens medlemmer har som oftest en mening, og på denne måten kan man diskutere disse uten å måtte møtes fysisk.

12.7 Kildekode

Vi har under hele utviklingsfasen benyttet kodestandarden pep8 [46]. Pep8 er den mest brukte kodestandarden for Python og anbefales av de aller fleste Python utviklere[47]. Noen utviklere følger denne standarden til punkt og prikke, mens andre argumenter for at kodestandarden er for streng og medfører for mye administrativt arbeid som kan ødelegge flyten i kodeskrivingen.

I prosjektgruppen utnevnte vi et medlem som skulle sette seg godt inn i standarden og så holde en innføringspresentasjon for de andre medlemmene. Det utnevnte medlemmet hadde også et ansvar under prosjektet om å gjennomgå all kildekoding og med jevne mellomrom gjøre en kvalitetssjekk av hvorvidt pep8 standarden ble opprettholdt. Denne fremgangsmåten har fungert meget bra og vi har vært nøye på å følge standarden under hele prosjektet.

Vi har valgt å gjøre noen unntak fra pep8 for å holde leseligheten i koden så høy som mulig. Eksempler på steder der vi valgte å ikke overholde standarden er når vi ønsket å være mer forklarende rundt et kodesegment og koden sammen med kommentaren ville ha blitt for lang. Vi valgte da å sette kommentaren på oversiden av kodesegmentet Listing 12.3.

Listing 12.3: Ikke pep8 kommentar men nødvendig som forklaring av koden under

```
1 # If surface is to be excluded, return False if p is on surface.
2
3 if true_interior and (self.in_triangle(np.delete(t, 0, 0), p) or
4                       self.in_triangle(np.delete(t, 1, 0), p) or
5                       self.in_triangle(np.delete(t, 2, 0), p) or
6     return False
```

En annen viktig dokumentering av kilde kode er doc-strenger. Disse er ikke definert av pep8, men av pep258[48]. Gruppen avgjorde i fellesskap at denne dokumenterings strukturen ikke var god nok fordi den virket rotete og ikke beskrev parametere i metoden på en oversiktlig måte. Gruppen valgte å benytte standarden reStructuredText[49] som

også var tilgjengelig å gjennom PyCharm IDEen som vi benytter. Listing 12.4 viser et eksempel metoden in_tetrahedron og dens doc-streng. Først er det en kort forklaring på hva metoden gjør og, så kommer alle parametere med navn og type samt en kort forklaring. Til slutt listes retur verdien med type og en forklaring.

Listing 12.4: reStructuredText doc-string

```
1  def in_tetrahedron(self, t, p, true_interior=False):
2          """Checks if the point P, pointed to by vector p, is inside(including
               the surface) the tetrahedron
3              If 'p' is not guaranteed a true tetrahedron, use interior().
4
5          :param t: ndarray
6              The four points of a tetrahedron
7          :param p: ndarray
8              The point to be tested for inclusion in the tetrahedron.
9          :param true_interior: bool
10             Activate to exclude the surface of the tetrahedron from the search.
11         :return: Bool
12             True if q is inside, or on the surface of the tetrahedron.
13         """
```

Pep8 tar for seg alt innen formatering og syntaks av kode også. Vi velger og ikke vise eksempler på dette i rapporten, men for de som er spesielt interessert viser vi til kilden om pep8[46] og til kildekoden i Vedlegg P. Målet var å levere alt av kildekode med dokumentasjon i pep8 og dette målet har vi nådd med noen få unntak av kommentering og doc-stringer.

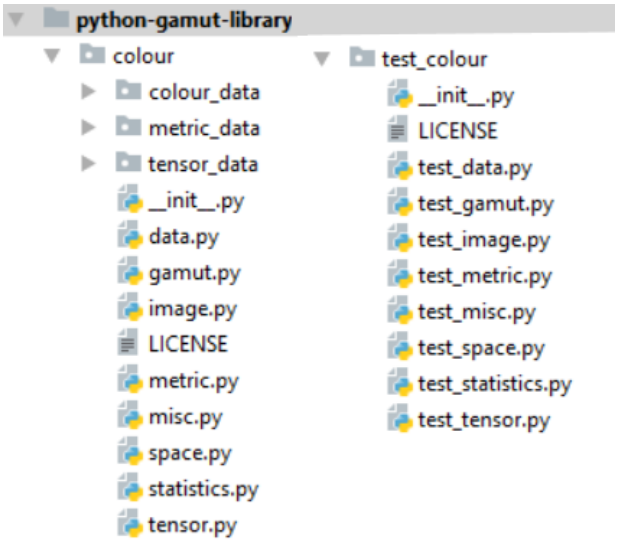
13 Design og struktur

I teori under Avsnitt 2.3 beskrev vi hvordan biblioteket var før vi begynte med videreutvikling. I dette kapitelet viser vi den oppdaterte strukturen i biblioteket, hvordan vi har strukturert gamutklassen, samt hvordan brukere installerer biblioteket.

13.1 Bibliotekets struktur

Filstrukturen i biblioteket Figur 34 var i utgangspunktet gitt av hvordan det var strukturert da prosjektgruppen startet videreutviklingen. Prosjektgruppen har fjernet alle tester som var i colour pakken og opprettet egne moduler for hver av dem i en ny pakke med navn test_colour.

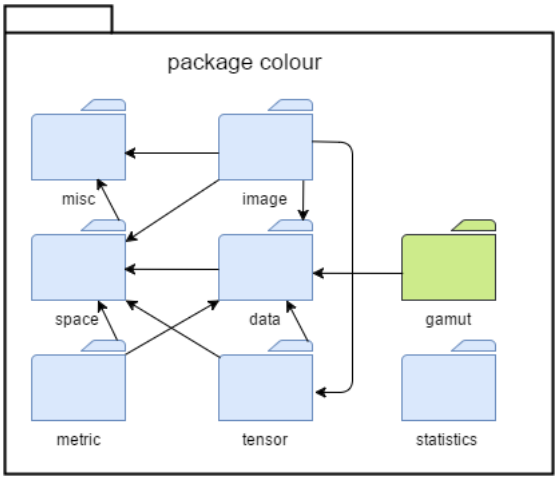
Biblioteket består av en overordnet mappe som inneholder pakkene colour og test_colour. Alle moduler er i egne filer, og den indre strukturen i test_colour ble laget lik strukturen i colour. Pakkene med navnending _data colour pakken er data/ressursfiler som behøves av de respektive modulene.



Figur 34: Figuren viser filstrukturen for biblioteket. For hver modul i colour er det en testmodul i test_colour pakken.

Avhengighet mellom bibliotekets moduler er en indikator på hvor nøstet bibliotek er internt. En avhengighet betyr at metoder i en modul behøver tilgang til en annen modul sine klasser eller metoder og at uten inkluderingen vil ikke modulen kunne utøve alle sine funksjonaliteter. Figur 35 viser den interne avhengigheten i colour pakken. Eksempelvis er gamut modulen kun avhengig av data modulen fordi det ikke opprettes noen klasser som er inneholdt i noen av de andre modulene. Gamut inneholder metoder som returnere data i form av et colour.data.Data objekt og er derfor avhengig av å benytte data modulen. Alle modulene i biblioteket er avhengig av flere pakker og moduler som

ikke er del av biblioteket, disse eksterne pakkene benyttes i stor grad til å utføre matematiske beregninger og er veldig stabile og anerkjente pakker. Se i Avsnitt 13.3 for en liste over disse eksterne pakkene.



Figur 35: Figuren viser de indre avhengigheter i colour pakken. Gamut modulen er uthevet i grønt og er kun avhengig av colour.data modulen.

13.2 Gamut klassen

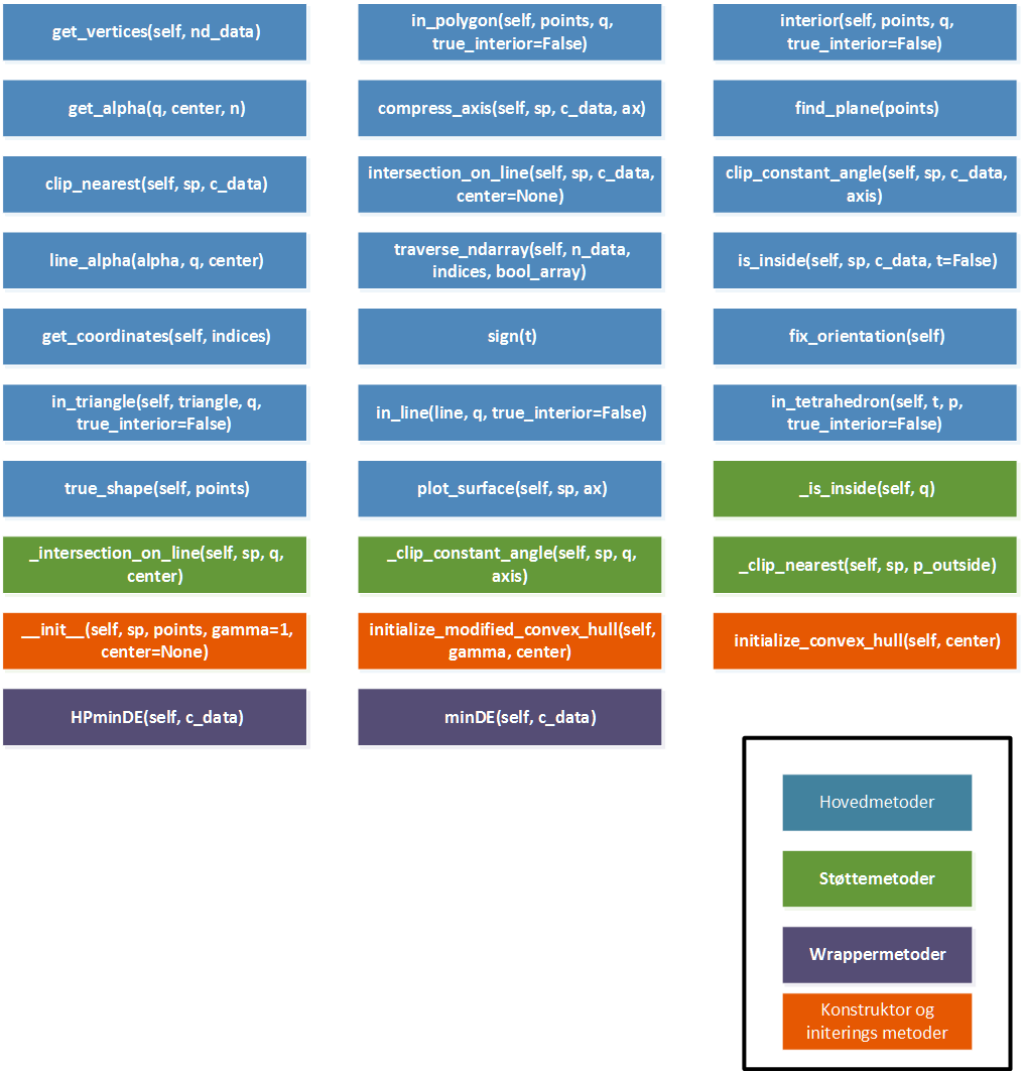
Figur 36 viser en oversikt over alle metoders navn og hvilken rolle de har i koden. Hovedmetoder, wrappermetoder, konstuktormetoder og initieringsmetoder er tilgjengelig for brukere og kan dermed bli tilkalt der brukeren ønsket det.

Konstruktor og initialiseringsmetoder: Gamut klassen består av en konstruktor, to initierings metoder samt resten av metodene som er laget i PGLene. Hovedkonstuktor __init__() setter diverse variabler og starter den meget viktige metoden som orienterer alle fasetter riktig. Den vil også starte en av to initialiserings metoder ut ifra en parameter som blir sendt med om gamuten er konveks eller om bruker ønsker å beregne gamuten med hensyn til konkavheter. Klassens første initierings metode initialize_convex_hull() brukes for å initialisere en konveks gamut. Klassens andre initierings metode er initialize_modified_convex_hull() som brukes for å initialisere konkave gamuter.

Wrappermetoder: Wrappermetoder er de metoder som er ferdige algoritmer og kan benyttes i en ferdig kodet løsning. Eksempelvis er HPminDE og minDE slike metoder. Wrappermetoder benytter som oftest is_inside() til å avgjøre om punkter er utenfor eller innenfor en gamut og vil så utføre en mapping algoritme på de punkter som ligger utenfor.

Støttemetoder: Støttemetoder som er alle metoder som gjør selve mapping av punkter. Disse er det ikke ment at brukeren skal tilkalle direkte og derfor merket med _ i starten av metode navn.

Hovedmetoder: Hovedmetoder er de metodene brukeren tilkaller. De kombineres med støttemetoder for å lage mapping algoritmer og andre større metoder. Mange grunnleggende beregninger gjøres her.



Figur 36: Oversikt over alle metoder. Blå er hovedmetoder, grønn er støttemetoder, lilla er wrappermetoder og oransje er konstruktor og initialiseringsmetoder.

13.3 Installering og bruk av biblioteket

For å kunne benytte biblioteket fullt og helt forutsetter vi at følgende er installert og konfigurert:

- Python 3.6
- Følgende pakker:
  - Numpy 1.12.0+mkl
  - scipy 0.18.1
  - matplotlib 2.0.0

For andre versjoner av Python eller pakkene listet over kan vi ikke garantere at alle bibliotekets metoder vil fungere, eller at alle tester vil passere.

13.3.1 Installering

Installeringen av biblioteket gjøres ved å klone eller laste ned biblioteket fra github reposetoriet[1] til Ivar Farup. Derretter må enten biblioteket ligge i mappen der Python er installert eller du legge stien til biblioteket til i path variabelen [50].

13.3.2 Bruk av biblioteket

For å benytte biblioteket må du importere colour pakken i din IDE eller Python kommandovindu på følgende måte:

Listing 13.1: Importering av modulene

```
1
2 import colour
3
4 from colour import data, space, gamut
```

Det er skrevet en manual til biblioteket se Vedlegg F. Denne er oppdatert med endringer vi har utført på eksisterende moduler samt at det er lagt til en manual for gamut modulen og test-pakken. Manualen er på engelsk, og forutsetter at leseren har grunnleggende programmeringsferdigheter. Seksjonen vi har skrevet for gamut modulen og test-pakken følger en ny standard derunder pep8 og retningslinjene[51] gitt av scipy.org. Vi har skalert ned hvordan vi har dokumentert metodene, da det ville ha blitt for mye arbeid og ikke nødvendig for biblioteket å dokumentere så godt som scipy ønsker dette.

13.3.3 Testing av biblioteket

Pakken test_colour inneholder alle tester for alle moduler i biblioteket i separate filer. Prosjektgruppen har kun utviklet unit tester for Gamut modulen og dens metoder, disse testene ligger i test_colour/test_gamut.py. Alle testene kan kjøres individuelt eller hele testfilen kan kjøres om det er ønsket. Det er også mulighet til å kjøre alle testfiler modulen inneholder ved å eksekvere test_colour/init.py filen.

14 Avslutning

14.1 Drøfting og diskusjoner

Vi ønsker i dette avsnittet å skrive om viktige diskusjoner og valg som er blitt tatt underveis i prosessen.

I første sprint valgte vi å arbeide direkte på master-grenen, som vi så på som enkleste løsning. Under utvikling i løpet av sprinten førte dette til problemer da endringer fra et annet programmeringspar kunne føre til at koden ikke delvis ble borte uten at det første paret hadde kjennskap til hvorfor. Dette ble diskutert i retrosepctive-møtet 14. Februar Vedlegg J.1, og alle var enig om at vi burde starte å benytte grener for hver PGL. På denne måten ville master-grenen alltid inneholde siste fungerende versjon, og utvikling av løsning for hver PGL kunne foregå isolert fra hverandre.

Vi har underveis måttet ta en del vurderinger på om vi skulle refaktorisere kode når en bedre løsning har presenter seg. Dette gikk ut på å estimere hvor mye bedre den nye løsningen var i forhold til den eksisterende, og om den eksisterende løsningen var effektiv nok allerede eller ikke. For mindre tilfeller har programmeringsparene tatt denne vurderingen selv underveis, mens for større og mer tidkrevende tilfeller har dette vært en diskusjon under review/planning møter. Det største tilfelle av refaktorering, er refaktoreringen av metoden is_inside(), PGL-36 Kapittel 6. Resultatene fra testing av den første løsningen ble presentert i review-møtet for andre sprint. Her konkluderte utviklerne med at metoden ikke var rask nok, og oppdragsgiver var enig i dette. Det ble derfor besluttet å lete etter forbedringspotensiale i neste sprint. En bedre løsning ble funnet og koden ble refaktorisert med gode resultater, se Avsnitt 6.4.3 for mer utfyllende informasjon. Det må sies at det var litt bittert å bytte ut store deler av arbeidet vi hadde gjort til da, men refaktorering er en viktig del av utvikling og det har vi sett verdien av under arbeidet med denne metoden.

14.2 Kritikk og refleksjon rundt oppgavebeskrivelsen

I dette avsnittet reflekterer vi rundt gruppens subjektive oppfattelse og meninger rundt oppgavebeskrivelsen som gitt av oppdragsgiver og ble satt som grunnlag for bachelorprosjektet . Det var kun denne oppgaven som appellerte til alle gruppens medlemmer, som tidligere nevnt i forord, og vi var alle enige i å ta kontakt med oppdragsgiver får å sikre oss denne oppgaven. Denne oppgaven ble kun tilbudt dataingeniørstudenter, noe som vi med tilbakeblikk kan forstå. Vi håper at det i løpet av hoveddelen i denne oppgaven har kommet frem hvor matematisk oppgaven har vært og at gruppen har måttet sette seg inn i et mangfold av komplekse konsepter innen matematikk og fargevitenskap. I tillegg har gruppen måttet koble all denne nye kunnskapen opp mot programmering, og benyttet seg av programmeringsbiblioteker som inneholder komplekse metoder brukt av forskere, ingeniører og matematikere verden rundt.

Prosjektgruppen har delte meninger og følelser rundt oppgaven. Det alle gruppens medlemmer kan stille seg bak er at vi mener oppgavebeskrivelsen ikke tydeliggjør hvor

matematisk tung oppgaven er. Vi mener også at det ikke kommer godt frem i oppgavebeskrivelsen i hvilken grad det skal utvikles metoder og algoritmer selv.

Vi holdt et oppstarts møte i starten av januar med oppdragsgiver der vi fikk forklart mer rundt oppgaven og hva som skulle gjøres samt hva som var forventet. I løpet av dette møtet ble det klart for gruppen hvor mye vi måtte utvikle selv og det matematiske nivået dette lå på. Gruppen trodde at de fleste av algoritmene og de grunnleggende utregningene var utviklet av fargelaben og at det meste av arbeidet lå i å samle disse under en felles standard og oversette det som var skrevet i andre programmeringsspråk som Java eller C, til Python.

Vi mener at det var helt riktig at oppgaven kun ble tilbudt dataingeniører på grunnlag av kombinasjonen av matematikk- og programmeringsferdighetene som måtte til for å kunne gi et godt resultat på oppgaven.

Ingeniørstudenter blir generelt utdannet høyest innen matematikk på bachelornivå innen informasjonteknologilinjene og av ingeniørlinjene er det dataingeniørene som har størst fordypning innen algoritmer. For å runde av utdanningen tar de fleste ingeniørstudentene fordypning i programmering.

Gruppen ville hatt et bedre grunnlag for å løse oppgaven hvis alle medlemmene hadde fordypet seg innen matematikk. Matematikken som alle burde hatt mer kompetanse innen var romgeometri og mer generell kunnskap innen fargevitenskap. Noe mer erfaring innen Python og forskningsorienterte bibliotek ville også vært til stor hjelp for å spare tid på starten av prosjektet.

Colour biblioteket benyttes aktivt av oppdragsgiver og andre forskere ved fargelaben og vi ønsket å levere et produkt som kan benyttes i forskningssammenhenger med virkelige bilde- eller sensordata. Dette har betydd at det for oss ikke holdt å se etter en hvilken som helst løsning på en oppgave. Vi fokuserte på å lage algoritmer og metoder så effektive at de kan benyttes med virkelige fargedata med millioner av fargepunkter.

14.3 Evaluering av gruppens arbeid

14.3.1 Innledning

Vi har skrevet en refleksjon og kritikk rundt oppgavebeskrivelsen som vi påpeker at burde leses før dette kapittelet.

Gruppen har som tidligere nevnt delte meninger om prosjektet. Deler av gruppen satt stort sett med en oppgave de mente var utenfor deres matematiske kunnskapsområde og besto av sammensettingen og koding av komplekse algoritmer og metoder. Dette har gruppen måtte ta hånd om underveis i utviklingsprosessen og rapportskrivingen og har hatt påvirkning på begge disse. Under organisering ønsker vi å omtale hele prosjektgruppen med alle fire utviklerne, prosjektleder, veileder og oppdragsgiver, fordi vi mener alle har vært en viktig del av prosjektet.

14.3.2 Organisering

Vi viser til organisasjonskartet Figur 1 som vist i innledningen for en oppfrisking av hvordan prosjektgruppen er organisert.

Gruppen mener prosjektleder har gjort et meget godt arbeid med å videreformidle informasjon mellom oppdragsgiver, veileder og utviklerne. Møter ble satt opp på gode

tidspunkter som passet flest mulig.

Hvis møter måtte flyttes på grunn av veileder eller oppdragsgiver sin arbeidsdag og de nye tidspunktene ikke passet for alle i gruppen gjorde han en god jobb med å videreformidle det som ble sagt i møtet. Prosjektleder var også stort sett alltid tilgjengelig, noe som førte til at gruppemedlemmene raskt kunne få svar på spørsmål til arbeidsoppgaver og annet prosjektrelatert.

Vi er veldig fornøyd med oppdragsgiver. Han har vært lett tilgjengelig på epost og på sitt kontor. Han var til tider bortreist på arbeidsrelaterte møter, men var da tilgjengelig på epost. Samarbeidet har vært meget verdifullt for gruppen som helhet. Under planlegging av sprinter var han meget aktiv og tok hensyn til utviklernes mangel på erfaring. Han var tydelig på prioriteringene sine, men imøtekom utviklernes ønsker om å lære seg nye verktøy, samt å bruke mer tid på testing av metoder.

Veileder har også vært meget tilgjengelig hele perioden, og har vært en verdifull ressurs for gruppen. Han har hatt en del på reiser, men har da vært lett tilgjengelig på epost og alltid sagt ifra til prosjektleder i god tid når veiledningsmøter måtte flyttes eller han ikke kunne møte på planleggingsmøtene. Vi har fått gode råd når det kommer til hvordan vi har strukturert rapporten, da vi har strukturert denne litt annerledes enn hvordan det klassisk gjøres i bacheloroppgaver. Også tilbakemeldinger på avsnitt og rapporten som helhet har vært meget gode og oppfordret til endring der det behøvdes mest, samtidig har han utpekt små detaljer som burde rettes opp i.

14.3.3 Fordeling av arbeid

Gruppen har fordelt arbeid som beskrevet i kapittel 6.2.5 og holdt seg til dette. Vi har hatt stort fokus på å fordele arbeidet slik at programmeringsparene satt med oppgaver de følte at de kunne mestre. Dette har ført til at de oppgavene med høyest kompleks matematikk i grunn har gått til det ene paret mens oppgaver som hadde hovedfokus på ren koding gikk til det andre paret. For å motvirke dette bestemte gruppen seg for å sette av tid på begynnelsen av andre sprint til at medlemmet med den største matematiske kompetansen tok seg tid til å forklare all matematikk rundt oppgaver til resten av gruppen. På denne måten kunne paret tildeles mer matematiske oppgaver og utviklingen kunne fortsette på en mer produktiv og effektiv måte. Fordelen med dette ble oppdaget tidlig i andre utviklingssprint og ble benyttet til slutten av prosjektet.

Bruken av parprogrammering har vist seg spesielt effektivt på begynnelsen og på komplekse deler av oppgaver. Parene har delt seg der de selv har ment det er fordelaktig. For eksempel har det vist seg å være mye mer effektivt å lete etter artikler og løsninger alene, enn i par. Det samme gjelder for rapportskriving. Ingen par har jobbet sammen når de har skrevet på rapport. Men parene har selvfølgelig kvalitetssikret arbeidet den andre har gjort, og diskutert og omskrevet der det var nødvendig.

Det er helt tydelig for prosjektgruppen at parprogrammering har vært en stor fordel under denne bacheloroppgaven og det er alle gruppens medlemmer er enige i. Det har blitt vurdert å bytte par underveis i utviklingen, noe som absolutt kan være en fordel for å få variert erfaring fra å samarbeide så tett med forskjellige programmerere. Gruppen bestemte enstemmig at parene både passet best sammen slik de allerede var med tanke på den matematiske forståelsen og fordi to av medlemmene var pendlere og bodde meget nærme hverandre og arbeidet mye sammen hjemme hos hverandre.

14.3.4 Prosjekt som arbeidsform

Bachelorprosjektet er det største og mest omfangsrikes prosjektet bachelorstudenter arbeider med i løpet av studiet. Det må anvendes prosjektstyring, utvikling og holdes fokus på kunde og prosess hele veien gjennom prosjektet. Et slikt prosjekt er meget relevant for arbeidslivet fra alle aspekter. Det må tas hensyn til kunden fra start til slutt, det skal utarbeides en omfangsrik prosjektplan og skrives en detaljert rapport.

Under et prosjekt med slik størrelse møter gruppens medlemmer på utfordringer rundt hvordan man oppfører seg i møte med oppdragsgiver og kunder. Gruppen lærer seg å sette kundens behov før sine personlige meninger og sedvaner, dette gjør at man som utvikler også finner nye metoder å løse problemer på og samler erfaringer rundt dette.

Det er åpent for gruppene å selv velge gruppestørrelser og dermed tilpasse utviklingsmodeller og metoder de benytter under prosjektet. Dette gir gruppen mulighet til å eksperimentere med teorien de har tilegnet seg under studiet og teste denne i praksis. Under studiet har vi blitt introdusert til mange forskjellige verktøy og metoder innen fag som systemutvikling, objektorientert programmering og algoritmiske metoder. Det er under et prosjekt som bachelorprosjektet at studentene nå har ansvaret selv får å velge de verktøyene de vil benytte under utvikling. En utfordring mange grupper støter på er å finne en balanse mellom utvikling og rapportskriving. Samtidig er det også en utfordring å skrive rapporten slik at tester og konklusjoner kan reproduseres og at det kommer frem hva gruppen har fokusert på og hva man har lært.

Vi har samlet mange nye erfaringer fra dette prosjektet. Gjennomføringen av en god og detaljert prosjektplan hvor tidsbruk, verktøy og metoder er klart definerte har resultert i at det kan fokuseres på utvikling og rapport under prosjektet. Vi har også gjort mange erfaringer med hvordan man forholder seg til oppdragsgiver og kunden generelt. Her må det nevnes at vi har hatt et veldig tett samarbeid med oppdragsgiver, noe som har vært meget behjelpelig for forståelsen av prosjektet og når vi løste utfordringer underveis i utviklingen.

Gruppens medlemmer har alle måttet samkjøre sine vaner rundt programmering, og bruken av en felles kvalitetsstandard har gjort dette enklere. Når vi tenker tilbake på de første sprintene ser vi at gruppen burde ha fokusert mer på å notere valg og diskusjoner underveis. Men vi dokumenterte godt nok underveis til å kunne reprodusere og skrive gode resultater. Miljøet i gruppen har til tider vært varierende, men mye av frustrasjonen kan relateres tilbake til selve oppgaven og ikke til relasjonen mellom gruppens medlemmer.

Det viktigste for gruppen er at vi er stolte over hva vi har utviklet og, at oppdragsgiver og fargelaben sitter igjen med et godt produkt de kan benytte seg av. I tillegg vil alle den læring vi sitter igjen med etter prosjektslutt være viktig for oss i videre studier og arbeidslivet.

14.4 Videre arbeid

Vi vet godt at oppdragsgiver har en omtrent uendelig liste med metoder han ønsker å implementere i både gamut modulen og andre moduler innen i colour biblioteket. Prosjektgruppen ser for seg at det absolutt kan skrives flere bacheloroppgaver rundt videreutvikling av colour biblioteket. Det er mulighet for å utforme oppgavebeskrivelser

rundt flere moduler enn gamut og det kan gis flere alternative oppgaver til gruppene. En interessant utvidelse ville vært å få flere grupper til å programmere på tvers av moduler samtidig. Vi ønsker å komme med noen anbefalinger til oppdragsgiver og prosjektgruppene som kan gjøre det enklere å utforme en god oppgavebeskrivelse og komme i gang med prosjektet.

Vi mener det burde settes høye krav til fremtidige gruppers kunnskap innen matematikk og programmering før de tildeles oppgaven. Parprogrammering har vært verdifullt for oss med tanke på litteratursøk, leting etter metoder og kvalitetssikring og vi anbefaler derfor dette videre til fremtidige grupper som arbeider med oppgaver rundt colour biblioteket. Vi kan også anbefale fremtidige grupper å lese denne oppgaven får en innføring i biblioteket og spesielt gamut modulen. Oppdragsgiver kan også gi en dypere innledning i biblioteket og hvordan eksempelvis colour.data pakken fungerer og hvordan man henter ut dataene derifra. Utviklerne har avtalt et møte med oppdragsgiver der feil, mangler og videre arbeid skal diskuteres i detalj. Noen av utvidelsene utviklerne vil foreslå er vist i Vedlegg G.

15 Konklusjon

Prosjektgruppen har levert i overkant av 1350 kodelinjer. Koden er integrert med eksisterende kode og ligger nå på Ivar Farup sitt colourspace git repository [1], under PGL-3.0 lisensen[52] og dermed tilgjengelig for alle som ønsker å benytte biblioteket.

Vi har levert en modul og en pakke til biblioteket, gamut modulen og test_colour pakke. Begge modulene følger pep8[46] standarden og er skrevet i Python v3.6[53] som er blant de nyeste versjonene tilgjengelig av Python.

Gruppen har også skrevet og levert en brukermanual på engelsk se Vedlegg F som også er tilgjengelig på hovedsiden til biblioteket [1]. Denne manualen gir leseren blant annet innsikt i klasser, metoder og variabler. Manualen er skrevet for hele biblioteket der gruppen kun har skrevet delene om gamut og test modulen.

Når vi ser tilbake på oppgavebeskrivelsen har vi oppfylt alle funksjonaliteter som oppdragsgiver har spurt etter med unntak av et punkt. Funksjonaliteten for å hente fargedata ut av bilder og ICC-profiler er ikke implementert. Dette er fordi oppdragsgiver under sprinter har nedprioritert disse oppgavene. Vi påstår derfor si at vi har oppfylt oppdragsgivers ønske til funksjonalitet, og er meget fornøyde med dette resultatet.

Et personlig mål for utviklerne var å kunne implementere brukbare GMAer. Altså metoder som gjør tar en enhetsgamut og fargedata og utfører en ønsket gamut mapping for å tilpasse fargene i bilde til enhetsgamuten. Dette målet nådde gruppen da vi implementerte GMAene HPminDE og minDE. Disse to GMAer benytter mange metoder som vi har skrevet og representerer gruppens suksess med oppgaven.

Gruppen har satt seg læringsmål som beskrevet i Avsnitt 1.3.3. Disse mener vi å ha oppnådd. Vi har hatt en bratt læringskurve med tanke på Python som programmeringsspråk, og generelle programmeringsferdigheter har økt gjennom utviklingen. Prosjektet har vært matematisk krevende og vi har måtte friske opp i gammel kunnskap, lære oss å anvende disse på nye måter og tilegne oss ny kunnskap innen matematikk. Vi har tidligere i studiet lært teori om utviklingsprosesser, men før nå har vi ikke fått satt denne teorien ut i praksis og det har vært meget lærerikt å anvende og følge en utviklingsmodell. Når det kommer til algoritmiske metoder og komplekse konsepter har vi måtte sette oss inn i alt fra forumposter til avanserte forskningsartikler. Dette har til tider vært en meget krevende prosess, og vi har samlet mye erfaring med å bryte ned problemer i mindre deler og utforme illustrasjoner for raskere og dypere forståelse.

Vi har skrevet mange metoder som er grunningredienser i større metoder og har dermed lagt et solid fundament for forskerne ved fargelaben og rundt i verden å bygge videre på. Vi ønsker å nevne at noen metoder behøver å effektiviseres med tanke på eksekveringstid for å kunne benyttes med virkelige store bildedata. Disse metodene er diskutert med oppdragsgiver under review møter, og oppdragsgiver har godkjent resultatet.

Bibliografi

[1] Farup, I. 2017. Ifarup/colourspace. https://github.com/ifarup/colourspace. Hentet 18-01-2017.

[2] Farup, I. 2016. A computational framework for colour metrics and colour space transforms. *PeerJ Computer Science*, 2, e48.

[3] Farup, I., Hardeberg, J. Y., Bakke, A. M., Kopperud, S., & Rindal, A. 2002. Visualization and interactive manipulation of color gamuts. In *Color and Imaging Conference*, volume 2002, 250–255. Society for Imaging Science and Technology.

[4] Wikipedia. 2017. ICC profile — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=ICC%20profile&oldid=736931974. [Online; accessed 13-May-2017].

[5] Wikipedia. 2017. Comma-separated values — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Comma-separated%20values&oldid=778638268. [Online; accessed 13-May-2017].

[6] Thomas, F. 2014. How to read your textbooks more efficiently - college info geek. https://www.youtube.com/watch?v=tgVjmFSx7rg. Hentet 28-01-2017.

[7] Atlassian. Jira hjemmeside. https://www.atlassian.com/software/jira. Hentet 11-05-17.

[8] 2017. Scrum guides. http://www.scrumguides.org/scrum-guide.html#team. Hentet 25-01-2017.

[9] McCallum, S. 2017. Copcse-ntnu/bachelor-thesis-ntnu. https://github.com/COPCSE-NTNU/bachelor-thesis-NTNU. Hentet 19-01-2017.

[10] Morovič, J. 2008. *Color gamut mapping*. John Wiley Sons, ltd.

[11] info@color hex.com. Fargeverdier. www.color-hex.com/color-names.html. Hentet 09-05-17.

[12] Peerj. Peerj hjemmeside. https://peerj.com/computer-science/. Hentet 12-05-17.

[13] Wikipedia. Polytop. https://no.wikipedia.org/wiki/Polytop. Hentet 13-05-17.

[14] Polyeder wikipedia. https://no.wikipedia.org/wiki/Polyeder. Hentet 31-03-2017.

[15] Hjørne. https://no.wikipedia.org/wiki/Hj%C3%B8rne. Hentet 10-05-17.

[16] Right hand rule. https://en.wikipedia.org/wiki/Right-hand_rule#Cross_products. Hentet 31-03-2017.

[17] community, T. S. The n-dimensional array (ndarray). https://docs.scipy.org/doc/numpy/reference/arrays.ndarray.html. Hentet 12-05-17.

[18] mplot3d tutorial — matplotlib 2.0.0 documentation. https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html. Hentet 26-04-17.

[19] Images scipy spatial convexhull. https://docs.scipy.org/doc/scipy-0.19.0/reference/_images/scipy-spatial-ConvexHull-1.png. Hentet 26-04-17.

[20] community, T. S. Convexhull referance guide. https://scipy.github.io/devdocs/generated/scipy.spatial.ConvexHull.html. Hentet 12-05-17.

[21] Scipy.org. Scipy. https://www.scipy.org. Hentet 14-05-17.

[22] Brummitt, C. Normalvektor metoden. http://stackoverflow.com/questions/16750618/whats-an-efficient-way-to-find-if-a-point-lies-in-the-convex-hull-of-a-point-cl Hentet 14-05-17.

[23] feqwix. Remake hull metoden. http://stackoverflow.com/questions/16750618/whats-an-efficient-way-to-find-if-a-point-lies-in-the-convex-hull-of-a-point-cl Hentet 14-05-17.

[24] 1982. Determining the spatial containment of a point in general polyhedra. *Computer Graphics and Image Processing*, 19(4), 303 – 334.

[25] F. R. Feito, J. C. T. Inclusion test for general polyhedra. http://hera.ugr.es/doi/15016870.pdf. Hentet 03-04-17.

[26] 2005. Point in solid strategies. *Computers Graphics*, 29(4), 616 – 624. doi:https://doi.org/10.1016/j.cag.2005.05.012.

[27] Karmann, C. How can you determine a point is between two other points on a line segment? http://stackoverflow.com/questions/328107/how-can-you-determine-a-point-is-between-two-other-points-on-a-line-segment. Hentet 11-05-17.

[28] blackpawn. Same side technique. http://blackpawn.com/texts/pointinpoly/default.html. Hentet 15-05-17.

[29] Nourai, R. Barycentric coordinates and point in triangle tests. https://blogs.msdn.microsoft.com/rezanour/2011/08/07/barycentric-coordinates-and-point-in-triangle-tests/. Hentet 15-05-17.

[30] Point in triangle test. http://blackpawn.com/texts/pointinpoly/default.html. Hentet 03-04-17.

[31] Nourai, R. Barycentric coordinates and point in triangle tests. https://blogs.msdn.microsoft.com/rezanour/2011/08/07/barycentric-coordinates-and-point-in-triangle-tests/. Hentet 03-04-17.

[32] Profiling , howpublished = https://en.wikipedia.org/wiki/profiling_(computer_programming), note =.

[33] Pycharm profiler. https://www.jetbrains.com/help/pycharm/2016.3/profiler.html. Hentet 03-04-17.

[34] About cython. http://cython.org. Hentet 03-04-17.

[35] pypy. How fast is pypy. http://speed.pypy.org. Hentet 03-04-17.

[36] Sunday, D. Intersection of a triangle with a plane. http://geomalgorithms.com/a06-_intersect-2.html. Hentet 05-05-17.

[37] Grumdrig. Shortest distance between a point and a line segment. http://stackoverflow.com/questions/849211/shortest-distance-between-a-point-and-a-line-segment. Hentet 05-05-17.

[38] for smidig utviklingsmodeller, F. Selskaper benytter scrum. http://www.agileweboperations.com/big-companies-using-scrum. Hentet 13-05-17.

[39] Foundation, P. S. Unittesting Python 3.6. https://docs.python.org/3/library/unittest.html. Hentet 26-04-17.

[40] Various. White-box testing. https://en.wikipedia.org/wiki/White-box_testing. Hentet 26-04-17.

[41] Various. Black-box testing. https://en.wikipedia.org/wiki/Black-box_testing. Hentet 26-04-17.

[42] Om versjonskontroll. https://git-scm.com/book/no-nb/v1/Komme-i-gang-Om-versjonskontroll. Hentet 05-04-17.

[43] Branches in a nutshell. https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell. Hentet 05-04-17.

[44] Allen, J. Sharelatex versionskontroll. https://www.sharelatex.com/blog/2014/03/31/track-changes-in-your-latex-documents.html. Hentet 05-04-17.

[45] Henry. Sharelatex kommentering. https://www.sharelatex.com/track-changes-and-comments-in-latex. Hentet 05-04-17.

[46] Guido van Rossum, Barry Warsaw, N. C. 2001. Style guide for Python code. https://www.python.org/dev/peps/pep-0008/. Hentet 17-01-17.

[47] Reitz, K. Python code style. http://python-guide-pt-br.readthedocs.io/en/latest/writing/style/. Hentet 24-04-17.

[48] Goodge, D. 2001. Docutils design specification. https://www.python.org/dev/peps/pep-0258/. Hentet 17-01-17.

[49] daouzli. Python docstring formats. http://stackoverflow.com/questions/3898572/what-is-the-standard-python-docstring-format. Hentet 26-04-17.

[50] Colin. Legge til sti i path variabelen. https://superuser.com/questions/949560/how-do-i-set-system-environment-variables-in-windows-10. Hentet 14-05-17.

[51] Contributing to scipy. http://scipy.github.io/devdocs/hacking.html. Hentet 20-04-17.

[52] Wang, K. General public license v3.0. https://tldrlegal.com/license/gnu-general-public-license-v3-(gpl-3). Hentet 10-05-17.

[53] Foundation, P. S. Python v3.6. https://www.python.org/downloads/release/python-360/. Hentet 10-05-17.

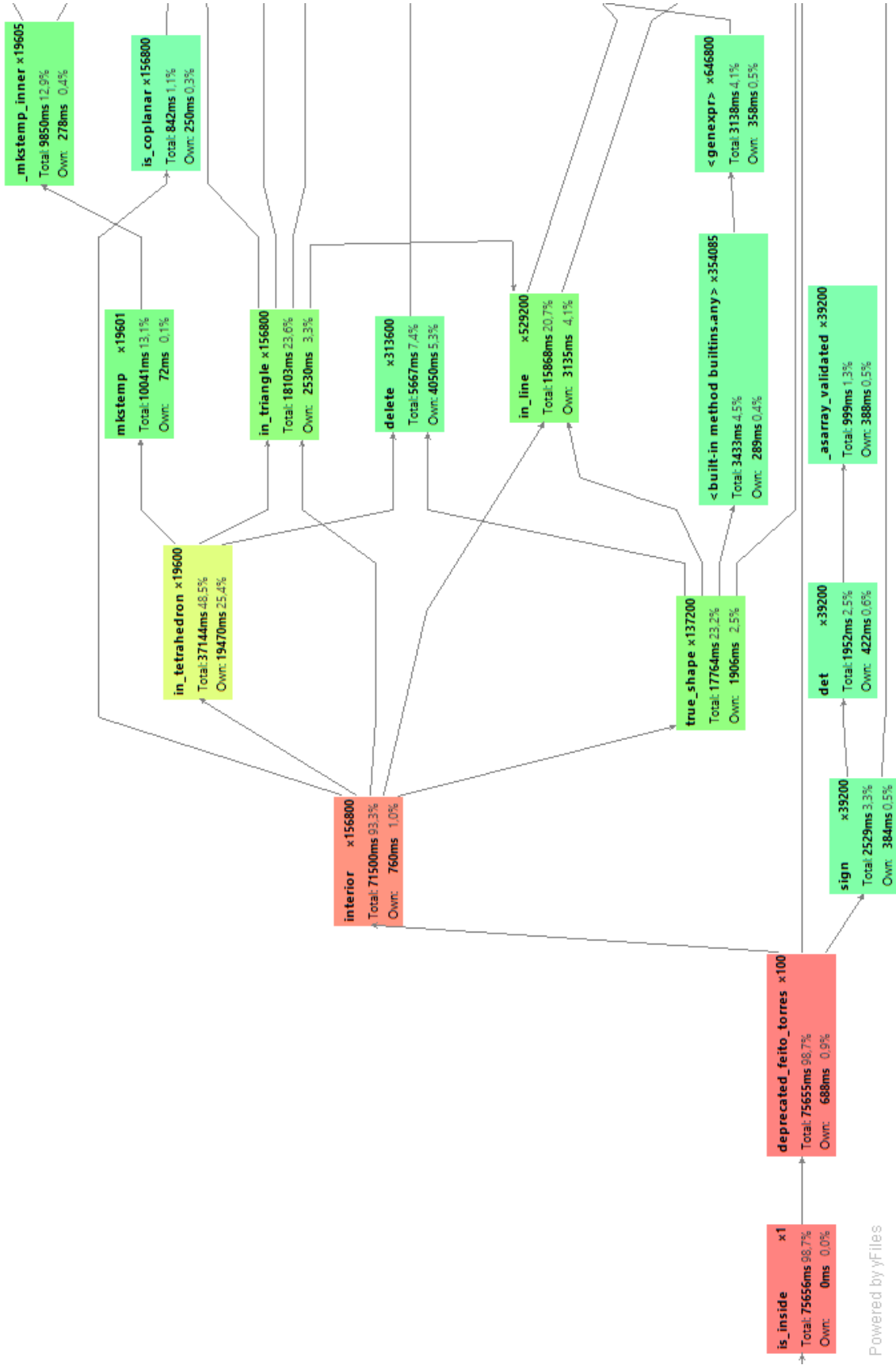
[54] Farup, I. Python-bibliotek for fargegamuter. https://bitbucket.org/LarsNiebuhr/python-gamut-library/src/bac3b32ab3396df616b8ea3be4cdda845b3a5e59/oppgavebeskrivelse%20(Ifarup).pdf. Hentet 25-01-2017.

[55] ICC. https://en.wikipedia.org/wiki/International_Color_Consortium. Hentet 25-01-2017.

[56] ICC homepage. http://www.color.org/index.xalter. Hentet 28-01-2017.

[57] 2017. Ordbok. https://www.tek.no/ordbok?letter=A. Hentet 25-01-2017.

A Profiling av is_inside



Figur 37: Profiling av is_inside metoden. Linjene som går videre kobles til eksterne bibliotek.

B Forprosjekt

NTNU
Norwegian University of
Science and Technology

Forprosjekt - Python Gamut Library

Author(s)

Jakob Michael Voigt
Lars Michael Niebuhr
Nawar Maher Behenam
Sahand Sohil Lahafdoozian

Bachelor of Science in Engineering - Computer Science
20 ECTS
Department of Computer Science
Norwegian University of Science and Technology,

27.01.2017

Supervisor(s) Marius Pedersen

Innhold

Definisjoner

- Lars - Lars Michael Niebhur
- Nawar - Nawar Maher Behenam
- Sahand - Sahand Lahafdoozian
- Jakob - Jakob Michael Voigt
- Utviklerene - Lars, Nawar, Sahand og Jakob
- Oppdragsgiver - Ivar Farup, professor ved NTNU Gjøvik, også product owner.
- Veileder - Marius Pedersen, førsteamenuensis ved NTNU Gjøvik.
- Prosjektgruppen - utviklerene, oppdragsgiver og veileder.
- Feriedag - 6 timer som kan trekkes ifra forventet antall arbeidstimer.
- Gamut - Forskjellige enheter som gjengir fargebilder (skjermer, projektorer, printere etc.) har ulike begrensninger i hvilke farger som kan gjengis. Den totale mengden av farger som kan gjengis på en gitt enhet kalles dens (farge)gamut. [1]
- GMA - Gamut mapping algoritme, en algoritme som gjør endringer i et bilde for å tilpasse bildet en ny gamut.
- Brukergruppen - Ivar, forskere og studenter som potensielt skal benytte seg av coulor biblioteket, og da også modulen prosjektgruppen utvikler.
- CSV-filer - En liste med komma separerte fargedatapunkter gitt av et måleinstrument.
- ICC-profiler - En standard for et sett data som beskriver en farge eller et fargerom [2, 3].
- Kjerneoperasjon - En grunnleggende matematisk operasjon som benyttes i GMAer. Eksempel: Finne ut om et fargedatapunkt er innenfor en gamut.
- Fargedatapunkt - Et punkt i et tredimensjonalt fargerom.
- Cusp - Et punkt på gamut-overflaten som er lengst unna et annet punkt eller linje i gamuten, ofte begrenset til et gitt plan. Oftest brukes det om det punktet på gamut-overflaten som er lengst unna gråaksen i et plan med konstant fargetone (hue
- API - Application Program Interface spesifiserer grensesnittet mot et system. Dette kan være et program, et operativsystem, en driver e.l.

  API-et definerer hvilke funksjoner du har tilgjengelig når du programmerer noe som skal kommunisere med systemet. Typisk vil API-et til et operativsystem tilby funksjoner for filbehandling, kommunikasjon med skjerm osv. [4]

- PBI - Product backlogg item. Et issue i backloggen i JIRA, altså en oppgave oppgave oppdragsgiver har definert.

1 MÅL OG RAMMER

1.1 Bakgrunn

I 2014 startet oppdragsgiver utvikling av biblioteket colour for å forenkle arbeid innenfor fargeviktenskap og fargebildeteknologi. [5]. Biblioteket benytter seg av et objectorientert beregnings rammeverk for å behandle fargedata og metriske tensorer for fargerommet. Ved å objektorientere komponentene i prosessen og utvikle metoder som utfører de vanligste transformasjonene kan arbeid som før kunne ta dager, gjøres ved hjelp av noen få kodelinjer. [6] Biblioteket mangler per dags dato støtte for å behandle med gamuter.

Et tilgjenelig verktøy for arbeid på gamuter, ICC3D(Interactive color correction in 3 dimensions) [7], ble utviklet i 2002 i sammarbeid med "The Norwegian Colour and Visual Computing Laboratory"(fargelaben). ICC3D ble utviklet for å hjelpe forskere og fagfolk til å forstå og møte utfordringene i farge og bildegjengivelse, spesielt for gamut mapping. Med tiden har ICC3D blitt utdatert da den baserer seg på biblioteker som ikke er vedlikeholdt noe som gjør at API'et er unødvendig tungvindt å bruke.

1.2 Prosjektmål

1.2.1 Effektmål

- Redusere antall programmeringsbibliotek brukergruppen må forholde seg til ved arbeid med gamuter.
- Forenkle prosessen med å sammenligne resultater fra forskjellige GMA, ved bruk av kun colour biblioteket.
- Redusere tiden det tar å anvende GMAer.
- Redusere tiden det tar å tilføye nye GMAer i colour biblioteket.

1.2.2 Resultatmål

Alle prosjektets mål realiseres gjennom modulen "colour.gamut". Den må derfor være i følgende tilstand ved prosjektets fullendelse.

- Ha funksjonalitet som lar brukeren kartlegge gamuter utifra bilder, ICC-profiler og CSV-filer.

- Lagre kartlagt informasjon om gamuter.
- La brukeren visualisere gamuten som et 3D objekt, ved bruk av vektorgrafikk
- Tilby et grensesnitt som kan utføre følgende kjerneoperasjoner på et sett med fargedatapunkter.
  - Svare på om et fargedatapunkt er innenfor eller utenfor en gitt gamut.
  - Finne et fargedatapunkts nærmeste punkt i en gitt gamut.
  - Finne avstanden fra et fargedatapunkt til gamutoverflaten, målt i et spesifisert fargerom.
  - Finne fargedatapunktets nærmeste punkt innenfor et gitt plan og gamuten. Disse planene skal kunne være plan parallelle med to koordinat akser i et kartesisk rom, eller et plan med en konstant vinkel ved bruk av sylinderkoordinater
  - Finne fargedatapunktets nærmeste punkt i en gitt gamut og på en gitt linje. Linjen skal kunne være en linje mellom to valgte punkter.
  - Finne sentrum til en gitt gamut.
  - Finne fargedatapunktets nærmeste punkt langs cuspen.
  - Finne lysheten til cuspen
  - Sigmoidal lightness mapping.

1.3 Rammer

1.3.1 Tidsramme

Prosjektgruppen må forholde seg til følgende tidsfrister:

- Innlevering av forprosjekt 28. januar 2017
- Innlevering av bacheloroppgave 16. mai 2017.
- Åpen fremføring av bachelorprosjekt 6. eller 7 juni.

1.3.2 Teknologisk

Vi benytter oss av Python 3.6, som er nyeste versjon ved prosjektstart og endrer ikke til ny versjon underveis i utviklingen. Oppdragsgiver har fastsatt at vi følge colour bibliotekets eksisterende kode- og dokumentasjonsstandard. Colour biblioteket er underlagt GPL lisensen, dermed vil all skreven kode også underlegges denne lisensen.

2 OMFANG

2.1 Fagfelt / Problemområdet

Oppdragsgivers bibliotek mangler idag funksjonalitet for gamut relatert arbeid. Han ønsker å utvide biblioteket med verktøy for å beregne, representere og visualisere gamuter tilhørende forskjellige enheter og bilder. Han ønsker også et grensesnitt for implementering av GMAer.

2.2 Avgrensning

Vi skal ikke videreutvikle direkte på noen av de eksisterende modulene i colour biblioteket. Biblioteket trenger kun være kompatibelt med python. Alt som skal inngå i gamut modulen er allerede forsket frem, og vi skal kun implementere det i python. Gamut expansion skal ikke være en del av denne prosjektoppgaven.

2.3 Oppgavebeskrivelse

Prosjektgruppens oppgave er å videreutvikle python biblioteket colour, ved å legge til modulen colour.gamut. Modulens funksjonalitet kan tredeles på følgende måte for enkel oversikt.

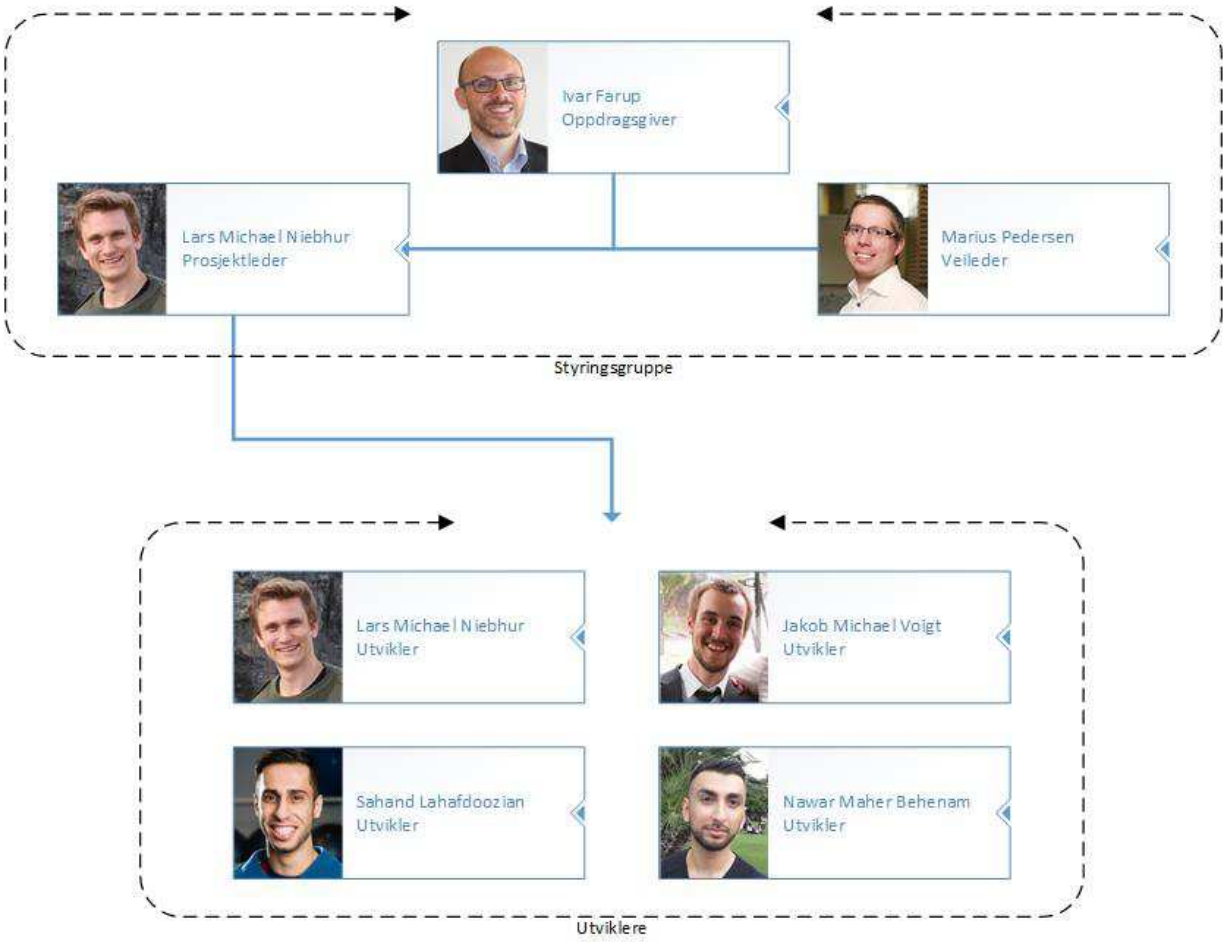
- Gamutoverflater: Modulen skal kunne beregne gamuter utifra bilder, ICC-profiler og CSV-filer. Når gamuten er beregnet skal den lagres og representeres som et objekt. Biblioteket skal kunne visualisere gamutens overflate som et 3D objekt.
- Kjerneoperasjoner: Biblioteket skal tilrettelegge et sett med kjerneoperasjoner, se punkt 1.2.2, på en slik måte at de kan settes sammen for å konsturere GMAer.
- GMA: GMAene HPminde og SGCK skal implementeres gjennom bruk av kjerneoperasjonene. Dersom utviklerene får tid kan flere GMAer implementeres.

3 PROSJEKTORGANISERING

3.1 Ansvarsforhold og roller

Lars, Jakob, Nawar og Sahand er prosjektgruppens utviklere. I tillegg til utviklerrollen er Lars prosjektleder, og Jakob referent. Ivar er prosjektets oppdragsgiver, og har særskilt ansvar for å være en ressurs for teknisk kompetanse innen oppgavens fagfelt. Marius er prosjektgruppens veileder, og har ansvar for generell veileding av bacheloroppgaven. Han har også god kompetanse innen fargebildeteknologi. Se punkt 4.3 for roller definert av utviklingsmodellen.

3.2 Organisasjonskart



3.3 Rutiner og regler i gruppen

3.3.1 Timelister

Alle utviklere i prosjektgruppen skal føre timer i et felles dokument.

3.3.2 Forventet arbeidsinnsats

Får å måle arbeidsinnsatsen til utviklerene, er det bestemet et innputt basert mål og et output basert mål [8].

- Input: Utviklerene skal arbeide minst 30 timer i gjennomsnitt per uke, og skal aldri ha arbeidet mindre enn 15 timer under forventet antall timer.
- Output: Dersom en utvikler en uke ikke ferdigstiller oppgaver tilsvarende mer enn 25 timer estimert tidsbruk skal det skriftlig rapporteres hvorfor. Gruppen kan kreve at personen det gjelder ferdigstiller mer arbeid.

3.4 Sykdom

Ved sykdom trekkes det ifra 3 timer per sykedag på antall forventede timer.

3.5 Fravær

Dersom en utvikler ikke skal møte en dag der det er planlagt at gruppen skal arbeide fra samme lokasjon, skal dette meldes inn minst en dag før til Lars.

3.6 Feriedager

I tillegg til planlagt ferie 8. april til 17. april, disponerer utviklerene enkelt vis 3 feriedager.

4 PLANLEGGING, OPPFØLGING OG RAPPORTERING

4.1 Valg av utviklingsmodell

I valg av utviklingsmodell ble det først konkludert å benytte én smidig utviklingsmodell. Det er flere grunner til at en smidig utviklingsmodell er å foretrekke for prosjektgruppen.

- Oppgaven har ikke en definert slutttilstand der det ikke er mulig å utvikle videre. Derfor passer det godt i dette prosjektet å bruke en iterativ utviklingsprosess, der vi er åpne for å legge til nye arbeidsoppgaver ved behov. [9, p.74]
- I utviklerenes første møte med oppdragsgiver opplyste han om at han var positivt innstilt til bruk av en smidig utviklingsmodell.
- Ved valg av en smidig utviklingsmodell forventer utviklerne å bruke relativt mindre tid på å følge modellens retningslinjer, i forhold til plandrevene modeller som ofte kan være tyngere å følge [9, p.76].
- Utviklerene foretrekker smidig over plandreven utvikling.
- Prosjektet har 4 utviklere. For mindre grupper passer smidige modeller ofte godt. Med scrum som eksempel anbefales maks 7 personer [9, p.85]

Deretter ble følgende utviklingsmodeller evaluert.

4.1.1 Scrum

Funksjonaliteten som etterspørres gjennom dette prosjektet har oppdraggiver bruk for allerede i dag. Fungerende inkrementer som han kan ta ibruk fortløpende er derfor et pluss.

Scrum baserer seg på selvorganiserende utviklere som selv bestemmer hvordan oppgaver løses på best mulig måte. I en liten gruppe på 4 utviklere er det viktig at alle har egendriv og tørr å ta egne beslutninger.

Gjennom rollen som product owner, involveres oppdragsgiver i utviklingsprosessen. Oppdragsgiver har god teknisk forståelse av prosjektet, fordi han har utviklet biblioteket som skal videreutvikles. Derfor er det en styrke for prosjektet å involvere oppdragsgiver mest mulig. Prosjektgruppen kan også være trygge på at product backloggen blir godt prioritert.

En utfordring ved bruk av scrum modellen vil trolig bli estimering av PBI. Utviklerene er ferske og lite erfaring med hvor lang tid utvikling tar. For å styrke estimeringsprosessen kan veilederen og oppdragsgiver inviteres med på planning poker [10].

4.1.2 Open Kanban

Hovedfordelen med denne modellen er at den er veldig enkel å følge. Verdiene til Open Kanban om å gi ansvar til enkelt personer, oppfordre til diskusjon der noen er uenige, og sterkt fokus på kommunikasjon og samarbeid passer er forenelig med en bachelor gruppe på fire personer der alle har sterk tilhørighet til oppgaven.

Et kanband board [11] vil kreve at vi kan hente ut oppgaver fra backloggen kontinuerlig, de burde ligge klare i backloggen i en prioritert rekkefølge.

4.1.3 Extreme Programming

Extreme Programming(XP) baserer seg på tett dialog med kunden, gjennom det som kalles onsite customer [12]. En slik dynamikk kan fungere for prosjektgruppen, da utviklerene og oppdragsgiver har arbeidsplass på samme lokasjon. Det må imidlertid vurderes om denne strukturen delegerer for mye arbeid til oppdragsgiver.

Modellen fokuserer også på trivsel blant utviklerene og en sunn jevn flyt i arbeidet uten overtidsarbeid og skippertak [13]. Dette er absolutt en nøkkel til suksess for enhver bachelorgruppe.

XP introduserer også bruken av parprogrammering. Dette kan føre til effektivt arbeid og god kvalitet i koden. [14]. Utviklerene ser også på det som en fordel å være to om å forstå de matematiske operasjonene og algoritmene vi skal kode. Denne typen programmering er også sunn for arbeidsmiljøet [15]

I XP brukes også prototyping [16]. Essensen i prototyping er at man lager enkle utkast av programvare som kan presenteres for oppdragsgiver, dette er ikke noe som behøves i dette prosjektet da vi ikke skal utvikle noe form for visuelt brukergrensesnitt.

4.2 Forklaring av vår utviklingsmodell

Scrum var den foretrukkene utviklingsmodellen for hele prosjektgruppen. Vi har valgt å innsnevre og modifisere utviklingsmodellen noe utifra våre behov og ønker. Innsnevringer er hovedsakelig gjort for å spare tid i utviklingen. Modefiseringer er gjort for å ta med elementer fra andre utviklingsmodeller som passet spesielt godt.

Det blir kun definert en Definition of done(DoD) [17] for PBIer som krever skrevene kodelinjer. For de øvrige PBIene benyttes kun beskrivelsen gitt i JIRA. Beskrivelsen, og DoDen dersom PBIen krever kodelinjer, brukes som en sjekkeliste for programmeringspar før en PBI flyttes til review. En utvikler fra et annet programmeringspar forholder seg til det samme da han inspiserer og flytter fra review til done.

Sprinter vil ha varighet på 10 arbeidsdager med noen avvik, se punkt 6.1. Vi har valgt en litt kortere sprintlengde for å få hypping innspill fra oppdragsgiver gjennom review- og planning meeting. Dette synes utviklerne passer godt, da de har lite erfaring. Veileder vil også bli brukt aktivt, det vil derfor holdes flere møter enn hva scrum i utgangspunktet tilrettelegger for, se veiledningsmøter punkt 5.1.3.

Det er blitt besluttet at parprogrammering skal benyttes. Med parprogrammering ønsker vi å oppnå at utviklerne kommer raskere i gang med hvert inkrement og skriver mer effektiv og velldokumentert kode. Bruken av parprogrammering evalueres underveis, og kan fjernes dersom utviklerene uttrykker ønske om det.

Det er kun utviklerene som er med på retrospective meeting. Tilbakemeldinger på prosessen fra oppdragsgiver og veileder kan gis i veiledningsmøtene. Review- og planning meeting legges til samme møte annenhver tirsdag.

4.3 Roller i valgt utviklingsmodell

4.3.1 Scrum master

Prosjektleder for rollen som scrum master. Han har ansvaret for å fordele arbeidsoppgaver og godkjenne kvalitetskravene som definert i punkt 5. Scrum master har også hovedansvar for kommunikasjon mellom oppdragsgiver, veileder og utviklere. En viktig del av oppgavene til scrum master er å motivere utviklere og holde retrospektive meeting.

4.3.2 Produkt eier

Produkt eier har ansvaret for å fylle backloggen i JIRA slik han mener er mest hensiktsmessig og prioritere sprint backlogg etter estimering.

4.3.3 Utvikler

Deres oppgave er å utarbeide all kode og dokumentasjon samt utarbeide forrapport, bacheloroppgave og holde presentasjon.

5 KVALITETSSIKRING

5.1 Dokumentasjon

5.1.1 Dokumentasjonkrav under utvikling

Pep8 er brukt som dokumentasjonsstandard i eksisterende deler av colour biblioteket, og skal brukes som standard videre. Hver enkel utvikler har ansvar for å dokumentere skreven kode etter denne standarden kontinuerlig i utviklingen. Prosjektleder skal godkjenne all dokumentasjon av kode før sprint slutt.

5.1.2 Dokumentasjonkrav til levert produkt

Det skal utarbeides en brukermanual for modulen på engelsk slik det er gjort for de eksisterende modulene av colour biblioteket [18].

5.1.3 Scrum og veiledning

Review møtene sikrer kvalitet i utviklingen, mens veiledningsmøtene annenhver tirsdag sikrer kvalitet i rapporten.

5.2 Kodestandard

Pep8 benyttes som kodestandard, som er den eksisterende kodestandarden i colour biblioteket [19].

5.3 Verktøybruk/ Konfigurasjonsstyring

Det vil bli brukt JIRA for oppfølging av scrum relatert systemutvikling. JIRA har integrert, backlogg, scrum board, burndown chart og tid/ressurs overvåking.

Det vil under hele utviklingsfasen bli brukt et git repository for versionskontroll. Bitbucket er valgt fordi utviklerne får tilgang til student-lisenser gjennom NTNU [20] som gir mulighet for private repository og tillater at seks aktører kan knytte seg til repositoryet [21], public lisenser gir kun fem aktører tilgang til et repository samtidig [22]. Bitbucket er også integrerbart med JIRA. Kombinasjonen av JIRA og bitbucket gir oppdragsgiver, veileder, prosjektleder og utviklere et brukervennlig og oversiktlig bilde av hvilket arbeid som er utført, hvem som jobber med en oppgave, og hva som gjenstår av arbeid.

Utviklerene har blitt enige om å bruke PyCharm som IDE. Vi har tidligere erfaring med verktøyet og er komfortable med programmet.

Ukentlig timesregistrering vil bli loggført i et Google spreadsheet som opplyst i punkt 3.2.1. All rapportskriving vil foregå i sharelatex, der malen gitt av NTNU benyttes [23].

Verktøy for utarbeiding av Gantt skjema er Microsoft Project, og diverse grafer og diagrammer vil bli utarbeidet i Microsoft Visio 2016.

5.4 Testing

All kode skal fortløpende testes av utviklerne under sprinter. Utviklere skal gjøre grunnlegende interntesting for å utelukke feil i koden og funksjonalitetens kompatibilitet. Med grunnlegende interntesting menes at det skal rettes opp i skrivefeil, funksjonkall og andre feil som gjør at koden ikke kan kompileres. Der PBI beskrivelsen ber som det skal utviklerene også skrive enhetstester. Eventulle avvik dokumenteres i JIRA.

Mer grundig testing vil bli gjort av oppdragsgiver. Oppdragsgiver vil avgjøre om funksjonelle krav og kvalitetsstandarder er møtt. Oppdragsgiver skal sammenligne utviklernes resultat etter funksjonalitet i forhold til den satte kravspesifikasjoner og skal avgjøre om det leverte produktet er godkjent. Tilgang til kildekode er gitt ved å gi oppdragsgiver tilgang til bitbucket repository.

5.5 Risikoanalyse

5.5.1 Identifikasjon og analyse av prosjektrisikoer

All risiko som gruppen gjør tiltak for har vi markert med * ved nummeringen. I Tabell 1 benytter vi følgende skala:

- Sannsynlighet: svært lav, lav, moderat, høy, svært høy.
- Påvirkningsgrad: tålererbart, moderat, alvorlig, katastrofe.

#	Riskio	Sannsynlighet	Påvirkningsgrad
1	Prosjektgruppen feilestimerer i planning poker	Svært høy	Tålererbart
2*	Underleverer på kode kvalitet	Moderat	Alvorlig
3	Scrum master forlater projektet	Svært lav	Katastrofe
4	Utvikler forlater prosjektet	Svært lav	Alvorlig
5	Oppdragsgiver ønsker endring i kravspesifikasjonen	Moderat	Tålererbart
6*	Dårlig dokumentasjon	Moderat	Alvorlig
7	Ikke ferdig med resultatmål innen 16.mai	Lav	Alvorlig
8*	Sykdom i gruppen	Høy	Moderat

Table 1: Risikoanalyse

5.5.2 Plan for håndtering av de viktigste risikoene

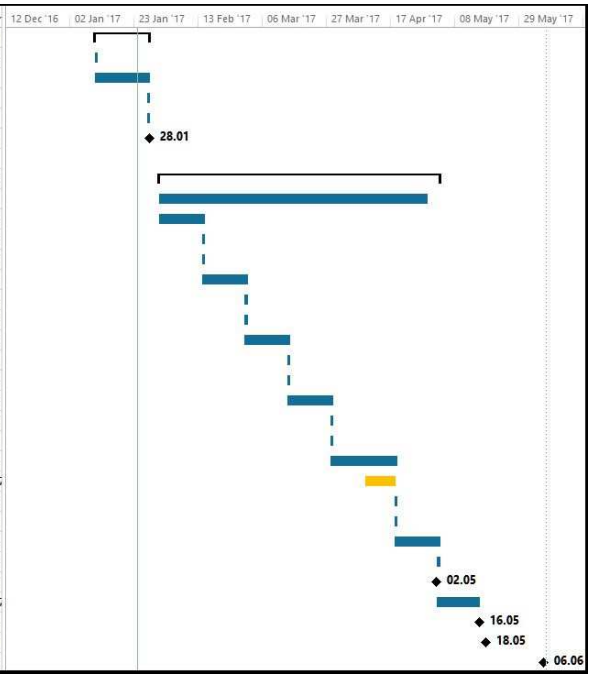
Risk	Strategi
Underleverer på kode kvalitet	Scrum master evaluerer fremgangen underveis. Dersom scrum master ser behovet, kan flere arbeidstimer bli lagt til for å jobbe inn tapt tid og komme i mål.
Dårlig dokumentasjon	I DoDen står det at pep8 skal følges, dermed kan ikke et programmeringspar ferdigstille en oppgave før den er dokumentert etter ønsket standard. Dokumentasjonen kontrolleres igjen av en annen utvikler når oppgaven skal evalueres i review. Scrum master vil i tillegg overse dokumentasjonen før koden leveres til oppdragsgiver.
Sykdom i gruppen	For å sikre at eventuelt sykdom ikke senker utviklingtempoet for mye, er det lagt til et punkt om dette i 3.3 Rutnier og regler i gruppen.

Table 2: Handlingsplan

6 PLAN FOR GJENNOMFØRING

6.1 Gantt-skjema

Task Name	Duration	Start	Finish
Forprosjekt	**14 days**	**Tue 10.01.17**	**Sat 28.01.17**
Planning meeting	1 day	Tue 10.01.17	Tue 10.01.17
Sprint 1	14 days	Tue 10.01.17	Fri 27.01.17
Retrospective	1 day	Fri 27.01.17	Fri 27.01.17
Planning meeting-/ Review meeting	1 day	Fri 27.01.17	Fri 27.01.17
Levere forprosjekt	0 days	Sat 28.01.17	Sat 28.01.17
Utviklingfase	**66 days**	**Tue 31.01.17**	**Tue 02.05.17**
Rapport	64 days	Tue 31.01.17	Fri 28.04.17
Sprint 2	11 days	Tue 31.01.17	Tue 14.02.17
Retrospective	1 day	Tue 14.02.17	Tue 14.02.17
Planning meeting-/ Review meeting	1 day	Tue 14.02.17	Tue 14.02.17
Sprint 3	11 days	Tue 14.02.17	Tue 28.02.17
Retrospective	1 day	Tue 28.02.17	Tue 28.02.17
Planning meeting-/ Review meeting	1 day	Tue 28.02.17	Tue 28.02.17
Sprint 4	11 days	Tue 28.02.17	Tue 14.03.17
Retrospective	1 day	Tue 14.03.17	Tue 14.03.17
Planning meeting-/ Review meeting	1 day	Tue 14.03.17	Tue 14.03.17
Sprint 5	11 days	Tue 14.03.17	Tue 28.03.17
Retrospective	1 day	Tue 28.03.17	Tue 28.03.17
Planning meeting-/ Review meeting	1 day	Tue 28.03.17	Tue 28.03.17
Sprint 6	16 days	Tue 28.03.17	Tue 18.04.17
Påskeferie	7 days	Sun 09.04.17	Mon 17.04.17
Retrospective	1 day	Tue 18.04.17	Tue 18.04.17
Planning meeting-/ Review meeting	1 day	Tue 18.04.17	Tue 18.04.17
Sprint 7	11 days	Tue 18.04.17	Tue 02.05.17
Retrospective-/ Review meeting	1 day	Tue 02.05.17	Tue 02.05.17
Slutten av utviklingen	0 days	Tue 02.05.17	Tue 02.05.17
Ferdigstille rapport	10 days	Tue 02.05.17	Mon 15.05.17
Levering av prosjekt	0 days	Tue 16.05.17	Tue 16.05.17
Presentasjon av prosjektet for ColourLab	0 days	Thu 18.05.17	Thu 18.05.17
Prsentasjon av prosjekt	0 days	Tue 06.06.17	Tue 06.06.17



I forprosjektet holder vi en sprint som er 14 arbeidsdager lang, mens vi jobber med sprinter på 10 arbeidsdager resten av prosjektet. Vi gjør et unntak når det kommer til sprint nummer 6. Denne vil være på 8 arbeidsdager fordi utviklerne har planlagt påskeferie fra 09/4 til 17/04.

Veiledningsmøtene annenhver tirsdag er ikke tatt med i gantt skjema, fordi den ikke er direkte tilknyttet utviklingen.

Utviklingsfasen avsluttes etter sprint 7 den 02/05. Den resterende tiden av prosjektet brukes til å ferdigstille rapporten. Det er kun satt av 10 dager til dette, fordi rapporten jobbes med parallelt med utviklingen.

Før presentasjon av prosjektet for sensorer og medstudenter har vi avtalt med oppdragsgiver at utviklerne skal holde en presentasjon av den ferdige modulen for ansatte og interesserte rundt fargelaben.

Kilder

[1] Ivar Farup. Python-bibliotek for fargegamuter. https://bitbucket.org/LarsNiebuhr/python-gamut-library/src/bac3b32ab3396df616b8ea3be4cdda845b3a5e59/oppgavebeskrivelse%20(Ifarup).pdf. Hentet 25-01-2017.

[2] Icc. https://en.wikipedia.org/wiki/International_Color_Consortium. Hentet 25-01-2017.

[3] Icc homepage. http://www.color.org/index.xalter. Hentet 28-01-2017.

[4] Ordbok. https://www.tek.no/ordbok?letter=A, 2017. Hentet 25-01-2017.

[5] Ivar Farup. Ifarup/colourspace. https://github.com/ifarup/colourspace, 2017. Hentet 18-01-2017.

[6] Ivar Farup. A computational framework for colour metrics and colour space transforms. *PeerJ Computer Science*, 2:e48, 2016.

[7] Ivar Farup, Jon Y Hardeberg, Arne M Bakke, Ståle Kopperud, and Anders Rindal. Visualization and interactive manipulation of color gamuts. In *Color and Imaging Conference*, volume 2002, pages 250–255. Society for Imaging Science and Technology, 2002.

[8] Frank Thomas. How to read your textbooks more efficiently - college info geek. https://www.youtube.com/watch?v=tgVjmFSx7rg, 2014. Hentet 28-01-2017.

[9] Ian Sommerville. *Software engineering*. Pearson Education, Harlow, United Kingdom, 10 edition, 08 2015.

[10] Mike cohn. https://www.mountaingoatsoftware.com/tools/planning-poker, 1998. Hentet 23-01-2017.

[11] Kanban board example. https://dl.dropboxusercontent.com/u/73847422/Kanban-Board/Kanban-Board-Example-AgileLion.png. Hentet 18-01-2017.

[12] Don Wells. On site customer. http://www.extremeprogramming.org/rules/customer.html. Hentet 24-01-2017.

[13] Don Wells. Sustainable pace. http://www.extremeprogramming.org/rules/overtime.html, 2017. Hentet 24-01-17.

[14] Pair programming. http://www.extremeprogramming.org/rules/pair.html, 2017. Hentet 25-01-2017.

[15] Pairprogrammingbenefitsna. http://wiki.c2.com/?PairProgrammingBenefits. Hentet 23-01-2017.

[16] Don Wells. Introducing extreme programming. http://www.extremeprogramming.org/ introduction.html. Hentet 23-01-2017.

[17] Scrum guides. http://www.scrumguides.org/scrum-guide.htmlteam, 2017. Hentet 25-01-2017.

[18] Ivar Farup. Ifarup/colourspace. https://github.com/ifarup/colourspace, 2017. Hentet 23-01-2017.

[19] Nick Coghlan Guido van Rossum, Barry Warsaw. Style guide for python code. https://www.python.org/dev/peps/pep-0008/, 2001. Hentet 17-01-17.

[20] last added by Erik Hjelmås Arne Styve. Use of bitbucket in courses at ntnu. https://www.ntnu.no/wiki/display/copcse/Bitbucket+-+git+repository, Sist oppdatert 22.08.2016. Hentet 17-01-2017.

[21] What are the guidelines for academic licenses. https://confluence.atlassian.com/bitbucket/what-are-the-guidelines-for-academic-licenses-296094528.html, Ukjent. Hentet 17-01-2017.

[22] Atlassian. Bitbucket pricing. https://www.atlassian.com/software/bitbucket/pricing?tab=cloud, Sist oppdatert 22.08.2016. Hentet 17-01-2017.

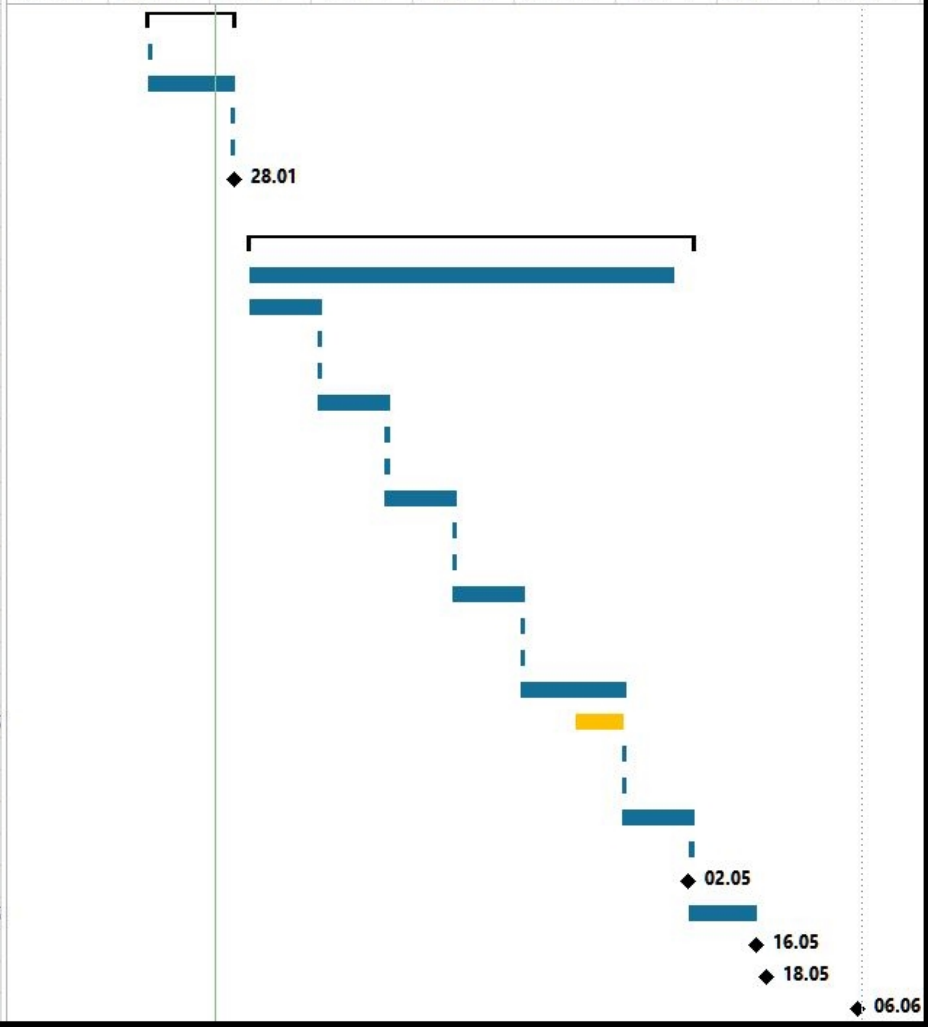
[23] Simon McCallum. Copcse-ntnu/bachelor-thesis-ntnu. https://github.com/COPCSE-NTNU/bachelor-thesis-NTNU, 2017. Hentet 19-01-2017.

C Definisjoner

- Gamut - Forskjellige enheter som gjengir fargebilder (skjermer, projektorer, printere etc.) har ulike begrensninger i hvilke farger som kan gjengis. Den totale mengden av farger som kan gjengis på en gitt enhet kalles dens (farge)gamut. [54]
- GMA - Gamut mapping algoritme, en algoritme som gjør endringer i et bilde for å tilpasse bildet en ny gamut.
- Brukergruppen - Ivar, forskere og studenter som potensielt skal benytte seg av coulor biblioteket, og da også modulen prosjektgruppen utvikler.
- CSV-filer - En liste med komma separerte fargedatapunkter gitt av et måleinstrument.
- ICC-profiler - En standard for et sett data som beskriver en farge eller et fargerom [55, 56].
- Kjerneoperasjon - En grunnleggende matematisk operasjon som benyttes i GMAer. Eksempel: Finne ut om et fargedatapunkt er innenfor en gamut.
- Fargedatapunkt - Et punkt i et tredimensjonalt fargerom.
- Cusp - Et punkt på gamut-overflaten som er lengst unna et annet punkt eller linje i gamuten, ofte begrenset til et gitt plan. Oftest brukes det om det punktet på gamut-overflaten som er lengst unna gråaksen i et plan med konstant fargetone (hue
- API - Application Program Interface spesifiserer grensesnittet mot et system. Dette kan være et program, et operativsystem, en driver e.l.
  API-et definerer hvilke funksjoner du har tilgjengelig når du programmerer noe som skal kommunisere med systemet. Typisk vil API-et til et operativsystem tilby funksjoner for filbehandling, kommunikasjon med skjerm osv.[57]
- PBI - Product backlogg item. Et issue i backloggen i JIRA, altså en oppgave oppdragsgiver har definert.

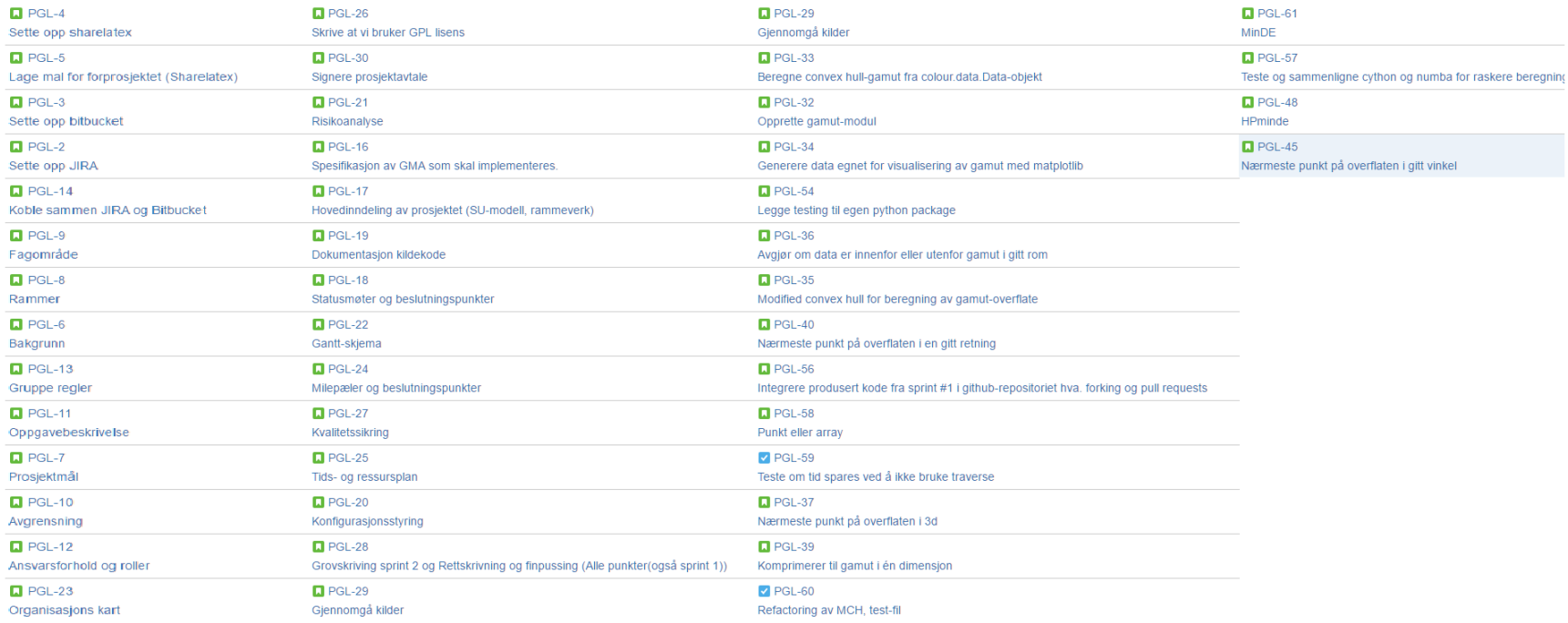
D Ganttdiagram

Task Name	Duration	Start	Finish
Forprosjekt	**14 days**	**Tue 10.01.17**	**Sat 28.01.17**
Planning meeting	1 d ıy	Tue 10.01.17	Tue 10.01.17
Sprint 1	14 lays	Tue 10.01.17	Fri 27.01.17
Retrospective	1 d ıy	Fri 27.01.17	Fri 27.01.17
Planning meeting-/ Review meeting	1 day	Fri 27.01.17	Fri 27.01.17
Levere forprosjekt	0 days	Sat 28.01.17	Sat 28.01.17
Utviklingfase	**66 days**	**Tue 31.01.17**	**Tue 02.05.17**
Rapport	64 days	Tue 31.01.17	Fri 28.04.17
Sprint 2	11 days	Tue 31.01.17	Tue 14.02.17
Retrospective	1 day	Tue 14.02.17	Tue 14.02.17
Planning meeting-/ Review meeting	1 day	Tue 14.02.17	Tue 14.02.17
Sprint 3	11 days	Tue 14.02.17	Tue 28.02.17
Retrospective	1 day	Tue 28.02.17	Tue 28.02.17
Planning meeting-/ Review meeting	1 day	Tue 28.02.17	Tue 28.02.17
Sprint 4	11 days	Tue 28.02.17	Tue 14.03.17
Retrospective	1 day	Tue 14.03.17	Tue 14.03.17
Planning meeting-/ Review meeting	1 day	Tue 14.03.17	Tue 14.03.17
Sprint 5	11 days	Tue 14.03.17	Tue 28.03.17
Retrospective	1 day	Tue 28.03.17	Tue 28.03.17
Planning meeting-/ Review meeting	1 day	Tue 28.03.17	Tue 28.03.17
Sprint 6	16 days	Tue 28.03.17	Tue 18.04.17
Påskeferie	7 days	Sun 09.04.17	Mon 17.04.17
Retrospective	1 day	Tue 18.04.17	Tue 18.04.17
Planning meeting-/ Review meeting	1 day	Tue 18.04.17	Tue 18.04.17
Sprint 7	11 days	Tue 18.04.17	Tue 02.05.17
Retrospective-/ Review meeting	1 day	Tue 02.05.17	Tue 02.05.17
Slutten av utviklingen	0 days	Tue 02.05.17	Tue 02.05.17
Ferdigstille rapport	10 days	Tue 02.05.17	Mon 15.05.17
Levering av prosjekt	0 days	Tue 16.05.17	Tue 16.05.17
Presentasjon av prosjektet for ColourLab	0 days	Thu 18.05.17	Thu 18.05.17
Prsentasjon av prosjekt	0 days	Tue 06.06.17	Tue 06.06.17



Figur 38: Ganttdiagram som viser planlagt tidsbruk for prosjektet

E Produkt backlog



Figur 39: Full oversikt over alle PGL i JIRA

F Brukermanual

colour

colour is a Python package for colour metrics and colour space transforms. Many common colour spaces and colour metrics are available, and more are continuously added. A description of the design of the computational framework is available as Farup I. (2016) A computational framework for colour metrics and colour space transforms. *PeerJ Computer Science* 2:e48 https://doi.org/10.7717/peerj-cs.48

Modules

The package consists of eight modules:

- colour.data
- colour.space
- colour.metric
- colour.tensor
- colour.statistics
- colour.misc
- colour.image
- colour.gamut

All the modules are imported when importing the package. The basic functionality of supposedly general interest is found in the three first modules. Only the really basic functionality is documented here. For more advanced features, please refer to the code (which is documented with standard pydoc docstrings), or contact the author.

Representing and Converting Colour Data

Basic numerical colour data are represented as numpy arrays of dimensions Nx...xMx3. In other words, colour data can be of any dimension, as long as the last dimension is the colour dimension. In particular, single points in the colour space will be ndarrays of shape (3,), lists of colour data will have dimension (N,3), and colour images will have dimension (M,N,3).

Colour data is scaled such that the natural maximum value is unity for most colour spaces, including XYZ having Y=1 for the whichever white point, and the RGB spaces having (1,1,1) as white points. For colour spaces with explicitly defined scaling like CIELAB and CIELAB (where the white point is defined as L*=100), the original scaling is used.

Colour data, i.e., colour points, lists of colours, colour images etc., are most conveniently represented by objects of the colour.data.Data class. In order to construct such an object, the colour data and the colour space has to be specified. Typically, if `col_xyz` is an ndarray with colour data in XYZ, a colour data object can be constructed as

```
col_data = colour.data.Data(colour.space.xyz, col_xyz)
```

Then the corresponding ndarray data in other colour spaces can be acquired by the get method:

```
col_srgb = col_data.get(colour.space.srgb)
col_lab  = col_data.get(colour.space.cielab)
col_xyY  = col_data.get(colour.space.xyY)
```

and so on. The colour conversions are computed only once and buffered within the Data object, so no extra overhead (besides the function call) is caused by sequential calls to the get method with the same colour space as the argument. Currently, the following colour spaces are available:

- **colour.space.xyz**: The CIE XYZ colour space.
- **colour.space.xyY**: The CIE xyY colour space.
- **colour.space.cielab**: The CIELAB colour space with D65 white point.
- **colour.space.cielch**: Polar coordinates in the CIELAB colour space.
- **colour.space.cieluv**: The CIELUV colour space with D65 white point.
- **colour.space.ciecat02**: The colour space of the CIECAT02 colour adaptation transform.
- **colour.space.ciede00lab**: The underlying colourspace of the CIEDE2000 colour metric.
- **colour.space.srgb**: The sRGB colour space.
- **colour.space.rgb_adobe**: The Adobe RGB colour space.
- **colour.space.ipt**: The IPT colour space.
- **colour.space.lgj_osa**: The OSA-UCS colour space.
- **colour.space.lgj_e**: The Euclidised OSA-UCS colour space used in the $\#E_E$ metric.
- **colour.space.din99x**: The various DIN99x colour spaces for the corresponding metrics (x is empty, b, c, or d).

There are also built-in colour data sets available. They are all represented by Data objects that can be constructed upon need by functions in the colour.data module. These functions have names starting with `d_`. Most of these data sets are mainly of interest for colour metrics researcher, but some of them will have broader interest, such as the various CIE colour matching functions, and the data of the Munsell patches.

Computing Colour Metrics

The most common colour metrics are available as functions in the colour.metrics module. All the metric funtions take two colour Data objects as parameters. The two objects must be of the same dimension. If the colour data in the Data objects are of the dimension Nx...xMx3, the return value of the metric functions are ndarrays of dimension Nx...XM. For example, the $\#E_{ab}$ colour difference between the two datasets `dataset1` and `dataset2` is computed as

```
diff = colour.metric.dE_ab(dataset1, dataset2)
```

The following metrics are available:

- **colour.metric.dE_ab**: The CIE76 standard $\#E_{ab}$ – the Euclidean distance in CIELAB.
- **colour.metric.dE_uv**: The Euclidean metric in CIELUV, $\#E_{uv}$.
- **colour.metric.dE_00**: The CIEDE2000 non-Euclidean colour metric.
- **colour.metric.dE_E**: The Euclidean colour metric $\#E_E$.
- **colour.metric.dE_DIN99x**: The DIN99x colour metrics (where x is empty, b, c or d).

Additionally, a general Euclidean colour metric in a given colour space can be computed as

```
my_diff = colour.metric.euclidean(my_colour_space, dataset1, dataset2)
```

Constructing Colour Spaces

New colour space objects are constructed by starting with an exisiting base colour space, and applying colour space transformations. For example, the fictive colour space `my_rgb` can be defined by a linear transformation from XYZ using the matrix `my_M` followed by a gamma correction with `my_gamma` as follows

```
my_rgb_linear = colour.space.TransformLinear(colour.space.xyz, my_M)
my_rgb = colour.space.TransformGamma(my_rgb_linear, my_gamma)
```

Any existing colour space can be used as the base for the transformation. Currently, the following common colour space transformations have been defined:

- **colour.space.TransformLinear**: A linear transformation defined by a 3x3 matrix (represented as a ndarray of shape (3,3)).
- **colour.space.TransformGamma**: A gamma correction applied individually to all channels. If the channel values are negative (should not be the case for RGB type spaces, but occurs, e.g., in IPT), the gamma transform is applied to the absolute value of the channel, and the sign is added in the end.
- **colour.space.TransformPolar**: Transform the two last colour coordinates from cartesian to polar. This can be used, e.g., to transform from CIELAB to CIELCH. Keep in mind, though, that this transformation is singular at the origin of the chromatic plane, and that this can confuse some colour metrics. The angle is represented in radians.
- **colour.space.TransformCartesian**: The oposite of the above.
- **colour.space.TransformxyY**: The projective transform from, e.g., XYZ to xyY, with that order of the coordinates.
- **colour.space.TransformCIELAB**: The non-linear transformation from XYZ to CIELAB, including the linear part for small XYZ values. The white point is a parameter, so this transformation can be used also to create CIELAB D50, etc. The base space does not have to be XYZ, so this transform can also be used to create, e.g., the DIN99 colour spaces.
- **colour.space.TransformCIELUV**: The non-linear transformation from XYZ to CIELUV, including the linear part for small XYZ values.The white point is a parameter, so this transformation can be used also to create CIELUV D50, etc.
- **colour.space.TransformSRGB**: The non-linear gamma-like correction from linear RGB with sRGB primaries to the non-linear sRGB colour space.

In addition, the following more special colour space transforms are available:

- colour.space.TransformLogCompressL
- colour.space.TransformLogCompressC
- colour.space.TransformPoincareDisk
- colour.space.TransformCIEDE00
- colour.space.TransformLGJOSA
- colour.space.TransformLGJE

Adding to this, new colour space transforms can be defined as classes inheriting from the colour.space. Transform base class.

Common white points are available as the following Data objects:

- colour.space.white_A
- colour.space.white_B
- colour.space.white_C
- colour.space.white_D50
- colour.space.white_D55
- colour.space.white_D65
- colour.space.white_D75

- colour.space.white_E
- colour.space.white_F2
- colour.space.white_F7
- colour.space.white_F11

Gamut

(This module is written in python 3.6)

Constructing Gamut

To construct a new Gamut we need to provide a colour space in the format provided by colour.space, and data/colour points in the format given the colour.data.Data class. If we want to construct the new Gamut in the colourspace RGB and the fictive points my_points, we would do it as follows

For convex hull

```
# First generate the Data objekt to use
c_data = data.Data(space.srgb, my_points)
# Pass along the colourspace and c_data
g = gamut.Gamut(space.srgb, c_data)
```

For modified-convex hull

When using the modified constructor, we have to choose an exponent for modifying the gamut radius (gamma), and define a center for expansion.

```
# First generate the Data objekt to use
c_data = data.Data(space.srgb, my_points)
# Pass along the colourspace, c_data, gamma and center
g = gamut.Gamut(space.srgb, c_data, gamma=0.2, center=my_center)
```

Examples

For all examples:

- **space:** a colour.space.Space object
- **c_data:** a colour.data.Data object
- **p_in/p_out:** a point inside/outside the gamut

All examples presupposes that you have created a colour Data object(c_data) and a gamut(g) object.

```
# Generating the colour Data object
c_data = data.Data(space, gamut_points)
```

```
# Creates a new gamut
g = gamut.Gamut(space, c_data)
```

is_inside()

The function receives two parameters, colourspace and a colour data object(c_data). The function checks if points are inn the convex hull and return boolean-array containing true/false in the last dimension.

```
a = g.is_inside(space, c_data)               # Call the method
```

plot_surface()

The function receives two parameters axis and space. The function will visualize a gamut figure in 3D.

```
fig = plt.figure()                           # Creates a figure
axis = fig.add_subplot(111, projection='3d')  # Creates a 3D plot ax
space = g.space                              # Specifies the color space
g.plot_surface(axis, space)                  # Call the method
```

intersection_on_line():

The function receives three parameters. The colour space, the points in the c_data format, and center(if no center is defined, it will use the default gamut center). The function will return nearest point along a line between the point and the given center.

```
# Points outside the gamut object
points = np.array([[15, 5, 5], [5, 15, 5], [5, 5, 15]])
# data.Data object
c_data = data.Data(space.srgb, points)
# Call the method
re_data = g.intersection_on_line(space.srgb, c_data)
```

clip_nearest()

The function receives two parameters. Points outside are colour data object and are represented as numpy arrays of dimensions Nx...xMx3. The function will return nearest point in 3D.

```
# Points outside the gamut object
points = np.array([[5, 5, 15], [5, 5, 15], [5, 5, 15]])
# data.Data object
c_data = data.Data(space.srgb, points)
# Call the method
re_data = g.clip_nearest(space.srgb, c_data)
```

compress_axis()

The function receives three parameters. The color space, pints in the c_data format, and the axis to compress as integer. The axis range is [0,1,2] where 0==X, 1==Y and 2==Z.

```
c = g.compress_axis(space, c_data, axis)    # Call the method
```

HPminDE()

The function receives one parameter. The points in the c_data format, Maps all points that lie outside of the gamut to the nearest point on the plane formed by the point and the L axe in the CIELAB colour space. Returns coordinate for the closest point on plane in the colour.data.Data format.

```
# Points outside the gamut object
points = np.array([[0, 8, 8], [4, 0, 9], [4, 4, 3], [0, 10, 0], [15, 0, 0]])
# data.Data object
c_data = data.Data(space.cielab, points)
# Call the method
re_data = g.HPminDE(c_data)
```

minDE()

The function receives one parameter. The points in the c_data format, and maps all points that lie outside of the gamut to the nearest point on the gamut in CIELAB colour space. Returns the nearest point in the colour.data.Data format.

```
mapped_im = g.minDE(c_data)              # Call the method
```

Attributes

Attribute	Description
data	The data.Data object used when constructed.
space	The original colour space used when constructed.
hull*	The gamuts convex hull in the desired colour space.
vertices	Indices of points forming the vertices of the convex hull.
simplices	Indices of points forming the simplices facets of the convex Hull.
neighbors	Indices of neighbor facets for each facet.
center	The Gamuts geometric center.

*see documentation on convex hull for a list of attributes. https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.ConvexHull.html

Methods

- **is_inside()**
- **plot_surface()**
- **intersection_on_line():**
- **clip_nearest()**
- **compress_axis()**
- **HPminDE()**
- **minDE()**

Method	Description	Return
is_inside(sp, c_data, t=false)	Returns a boolean array containing T/F for all points in the array.	boolean array
plot_surface(ax, sp)	Plot the gamut's simplices.	-
intersection_on_line(sp, c_data, center=None):	Returns the nearest point in a line on a gamut surface from the given point to the given center.	np.array
_clip_nearest(sp, p_out, side)	Returns the nearest point on a gamut in 3D.	np.array
compress_axis(sp, c_data, ax):	Compresses the points linearly in the desired axel and colour space.	colour. data.Data object
HPminDE(c_data):	Get coordinate for the closest point on plane and return mapped points	colour. data.Data object
minDE(c_data):	Get nearest point and return mapped points.	colour. data.Data object

test_colour

This is a test package containing one test module for each module in colour. The test modules does unittesting of each major function in the correlating module. All tests are automatically run when you run test_colour/init.py. Each unit test can be run separately. If you are using pycharm, navigate to the module containing the test you want to run, right-click the test function and select 'run unittest XXX'

Tests currently exists for the following modules:

- Gamut (v3.6)

G Produkt utvidelse

PGL-41	Generer gamut fra måledata-fil	1w 3d
PGL-46	Sigmoidal mapping i gitt koordinat-akse	2d 2h
PGL-42	Generer gamut fra ICC-profil	
PGL-43	Nærmeste punkt på overflaten i gitt koordinat-plan	
PGL-44	Nærmeste punkt på overflaten i gitt sylinder	
PGL-47	SGCK	
PGL-49	Segment maxima	
PGL-50	Alpha shapes	
PGL-51	Tilrettelegge for visualisering av flere gamuter samtidig i matplotlib	
PGL-55	Lage full tetrahederstruktur med naboer, ligninger for alle plan etc.	

H DOD

1. Koden er produsert og all ‘to do’ i koden fullført.
2. Koden ligger i egen gren.
3. Koden er kommentert, kjørt og test med siste versjon.
4. Koden er sett over av begge utviklings par.
5. Sørge for at koden ikke inneholder noen error.
6. Enhetstesting er skrevet og koden passer dem.
7. Alle endringer/konfigurasjon implementert, dokumentert og lastet opp i repo.
8. Alle relevant dokumentasjon og diagrammer ferdig skrevet/oppdatert.
9. Alle timer registrert og arbeid er kommentert i JIRA.

I Daglig møtereferater

I.1 Daily Scrum, 2. Februar

- Jakob Gamut konstruktor og initialiserings funksjon ferdig.
- Lars Gamut konstruktor og initialiserings funksjon ferdig.
- Sahand PGL-32 Ferdig. Sett på matplot funksjoner for generere data til visualisering.
- Nawar PGL-32 Ferdig. Sett på matplot funksjoner for generere data til visualisering.

- JakobFinne egnet algorimer for å avgjøre om et punkt et innenfor et objekt
- LarsFinne egnet algorimer for å avgjøre om et punkt et innenfor et objekt
- Sahand Forsette research av matplotlibs visualiseringsfunkjoner, og hvilket innput de tar.
- Nawar Forsette research av matplotlibs visualiseringsfunkjoner, og hvilket innput de tar.

I.2 Daily Scrum, 3. Februar

- Jakob Research av algoritmer for å avgjøre om en pkt er innenfor gamuten.
- Lars Research av algoritmer for å avgjøre om en pkt er innenfor gamuten.
- Sahand Reaserach av matplotlib funksjoner for PGL-34
- Nawar Reaserach av matplotlib funksjoner for PGL-34

- Jakob Implementere enkleste løsningfor PGL-36.
- Lars Implementere enkleste løsningfor PGL-36.
- Sahand Fortsette research og fikse python på maskinene sine.
- Nawar Fortsette research og fikse python på maskinene sine.

I.3 Daily Scrum, 6. Februar

- Jakob Begynt på en funksjon som rekusivt travaserer en ndarray, mens den tar var på stien dit". Jobbet litt med tester for PGL-36. Bugfixing i PGL-36 noen timer på kvelden.
- Lars Begynt på en funksjon som rekusivt travaserer en ndarray, mens den tar var på stien dit". Jobbet litt med tester for PGL-36
- Sahand PGL-34, sett på hvordan vi kan hente ut koordinatene til verticis
- Nawar PGL-34, sett på hvordan vi kan hente ut koordinatene til verticis

- Jakob Det logiske er på plass, men gjenstår bugfiksing på travaseringen.
- Lars Det logiske er på plass, men gjenstår bugfiksing på travaseringen.
- Sahand Fortsette å finne ut hvordan man finner koordinatene.
- Nawar Fortsette å finne ut hvordan man finner koordinatene.

I.4 Daily Scrum, 7. Februar

- Jakob Fikset på testene til PGL-36, prøvde å finne ut av ndarray fra Data.

- Lars Timetracking i Jira, språk i mal, flytte testinga til egen fil.
- Sahand Laget gamut.vertecis, og laget en test for den.
- Nawar Laget gamut.vertecis, og laget en test for den.

- Jakob Jobbe med traversendarray()
- Lars Jobbe med traversendarray()
- Sahand Jobbe med getsurface()
- Nawar Jobbe med getsurface()

I.5 Daily Scrum, 8. Februar

- Jakob Ferdigstilt testest og rekursiv funksjon. Møte.
- Lars Ferdigstilt testest og rekursiv funksjon. Møte.
- Sahand Getsurface(), testet ut matplotlib , getverticis og testen for det.
- Nawar Getsurface(), testet ut matplotlib , getverticis og testen for det.

- Jakob Feito-Torres all the way!
- Lars Feito-Torres all the way!
- Sahand Rapportering rundt blind vei"i PGL-34.
- Nawar Rapportering rundt blind vei"i PGL-34.

I.6 Daily Scrum, 9. Februar

- Jakob Leste om feito-torres. Renskrevet referater.
- Lars Leste om feito-torres.
- Sahand Testet trisurf og surface funksjone og fant ut at ingen av de fungerte veldig bra.
- Nawar Testet trisurf og surface funksjone og fant ut at ingen av de fungerte veldig bra.

- Jakob Begynne implementasjon av feito-torres.
- Lars Begynne implementasjon av feito-torres.
- Sahand Lage egen plot funksjon
- Nawar Lage egen plot funksjon

I.7 Daily Scrum, 13. Februar

- Jakob Lagd noen av funkjsonene som trengs for Feito-Torres".
- Lars Lagd noen av funkjsonene som trengs for Feito-Torres".
- Sahand Borte på turnering
- Nawar Plot funksjon, som kan sende med akser.

- Jakob Prøve å bli ferdig med feito-torres.
- Lars Prøve å bli ferdig med feito-torres.
- Sahand Gjøre ferdig plot funksjonen
- Nawar Gjøre ferdig plot funksjonen

I.8 Daily Scrum, 14. Februar

- Jakob Skrev ferdig in_tringle, restrukturterte litt på testene og laget ferdig testen til in_line. Startet å sette sammen feito-torres.

- Lars Skrev ferdig in_tringle, restrukturterte litt på testene og laget ferdig testen til in_line. Start å sette sammen feito-torres.
- Sahand Lagd en plot funksjon og test for den.
- Nawar Lagd en plot funksjon og test for den.

- Jakob Fortsette med feito-torres
- Lars Fortsette med feito-torres
- Sahand Restrukturere testfilene med korrekt pakke struktur.
- Nawar Restrukturere testfilene med korrekt pakke struktur.

I.9 Daily Scrum, 15. Februar

- Jakob Møter og singed_, feito torres_
- Lars Møter og singed_, feito torres_
- Sahand Møter, og dokumentering.
- Nawar Møter, og dokumentering.

- Jakob Feito-torres
- Lars Feito-torres
- Sahand Dokumenterer verteces, og plot funksjon.
- Nawar Dokumenterer verteces, og plot funksjon.

I.10 Daily Scrum, 20. Februar

- Jakob Fridag for LAOS kurs.
- Lars Prosjektavtale, skriving, mal i latex
- Sahand PGL-40: lang funksjoner, og gjort research.
- Nawar PGL-40: lang funksjoner, og gjort research.

- Jakob Bortrest.
- Lars Jobbet med rapport i sharelatex
- Sahand PGL-40, fortsette, bli ferdig med plit()
- Nawar PGL-40, fortsette, bli ferdig med plit()

I.11 Daily Scrum, 22. Februar

- Jakob Feilsøking i feito-torres. Fungerer nå når de er utenfor og på overflaten men ikke innenfor.
- Lars Feilsøking i feito-torres. Fungerer nå når de er utenfor og på overflaten men ikke innenfor.
- Sahand Jobber med PGL-40, fant en algoritme i går som kanskje kan fungere.
- Nawar Jobber med PGL-40, fant en algoritme i går som kanskje kan fungere.

- Jakob Jobbe videre med feito-torres
- Lars Jobbe videre med feito-torres. Skrive rapport mens Jakob har student assisten time.
- Sahand Jobbe videre med PGL-40.
- Nawar Jobbe videre med PGL-40.

I.12 Daily Scrum, 24. Februar

- Jakob Skidag.
- Lars Skidag. Skrevet litt rapport.
- Sahand Testing for PGL-40, og kommentering. Skrevet litt rapport om testmodul, og kildekode og kommentering.
- Nawar Testing for PGL-40, og kommentering. Skrevet litt rapport om testmodul, og kildekode og kommentering.

- Jakob Feito-torres.
- Lars Feito-torres.
- Sahand Jobbe på rapporten.
- Nawar Jobbe på rapporten.

I.13 Daily Scrum, 27. Februar

- Jakob Vært sjuk.
- Lars Forord rapport.
- Sahand Kildekode, og komenteringer (pep8, versjonkontroll) 4,5 visualisering. PGL 40.
- Nawar Skrevet rapprort testing, gamut modul og testmodul.

- Jakob Hjelpe til med matten på PGL-40. Feito-torres
- Lars Begynne å se på PGL-54", mens Jakob hjelper på PGL-40. Feito-torres.
- Sahand PGL-40.
- Nawar PGL-40

I.14 Daily Scrum, 28. Februar

- Jakob fix_orientation(), bug fikse feito, skrevet testing for feitotorres.
- Lars fix_orientation(), bug fikse feito, skrevet testing for feitotorres.
- Sahand Lette etter funksjoner for å løse likninger.
- Nawar Lette etter funksjoner for å løse likninger.

- Jakob Feito torres... bug fiksing.
- LarsFeito torres... bug fiksing.
- Sahand Forbrede seg på hva vi skal si til Ivar.
- Nawar Forbrede seg på hva vi skal si til Ivar.

I.15 Daily Scrum, 1. Mars

- Jakob Møter. Jobbet med feito torres tester.
- Lars Møter. Jobbet med feito torres tester.
- Sahand Møter og PGL-40.
- Nawar Møter og PGL-40.

- Jakob Feito torres fikse koordinat problem
- Lars Feito torres fikse koordinat problem
- Sahand Implementere løsningen vi diskuterte med Ivar for PGL-40.
- Nawar Implementere løsningen vi diskuterte med Ivar for PGL-40.

I.16 Daily Scrum, 2. Mars

- Jakob Muligens fikest den siste feilen i feito. skriver en test med sphere koordinaten.
- Lars Mligens fikset den siste feilen i feito. skriver en test med sphere koordinaten.
- Sahand Lagd en funksjon som lager likning for plan til alle simplicis.
- Nawar Lagd en funksjon som lager likning for plan til alle simplicis.

- Jakob Siste finpuss på feito-torres
- Lars Siste finpuss på feito-torres
- Sahand Jobbe videre med PGL-40.
- Nawar Jobbe videre med PGL-40.

I.17 Daily Scrum, 6. Mars

- Jakob Rapport.
- Lars Rapport.
- Sahand Rapport.
- Nawar Rapport.

- Jakob Rapport.
- Lars Rapport.
- Sahand PGL-40.
- Nawar PGL-40.

I.18 Daily Scrum, 2. Mars

- Jakob Helt fungerende løsning av feito-torres. Mangler bare omstrukturering, kommentering og finpuss. Var borte mandag
- Lars Helt fungerende løsning av feito-torres. Mangler bare omstrukturering, kommentering og finpuss. Skrevet rapport og hjalp Nawar og Sahand litt på mandag med PGL-40.
- Sahand Jobbet videre med PGL-40, litt omstrukturering, brukt tid på matematikken og diskusjon med Ivar. Skrevet test.
- Nawar Jobbet videre med PGL-40, litt omstrukturering, brukt tid på matematikken og diskusjon med Ivar. Skrevet test.

- Jakob Omstrukturere feito-torres, og begynne med pgl-54 integrasjon med resten av color bibliotektet.
- Lars Omstrukturere feito-torres, og begynne med pgl-54 integrasjon med resten av color bibliotektet.
- Sahand Videre med PGL-40.
- Nawar Videre med PGL-40.

I.19 Daily Scrum, 8. Mars

- Jakob Startet omstrukturering og siste finpussing av feito-torres. true_shape() Snakket med Hornæs om programmer for matematiske illustrasjoner, maple er et alternativ.
- Lars Startet omstrukturering og siste finpussing av feito-torres. true_shape()
- Sahand PGL-40 refactoring.

- Nawar PGL-40 refactoring.

- Jakob Gjøre ferdig omstrukturering av feito-torres.
- Lars Lars starter på PGL-54 (integrere med colour)
- Sahand Ferdigstille PGL-40 og skrive tester for PGL-40
- Nawar Ferdigstille PGL-40 og skrive tester for PGL-40

Jakob og Lars valgte å splitte paret idag, da de mente oppgavene for dagen mest effektivt gjøres alene.

I.20 Daily Scrum, 9. Mars

- Jakob Omstrukturerte ferdig feito-torres. true_shape(), interior()
- Lars Fikset feil bruk at assert funksjoner og lagde en test for is_inside(), Så på pgl-56.
- Sahand Lage tester for alle funksjonene i PGL-40. Fant en feil, nærmeste punkt blir altid sentrum. Jobebet med fikse det.
- Nawar Lage tester for alle funksjonene i PGL-40. Fant en feil, nærmeste punkt blir altid sentrum. Jobebet med fikse det.

- Jakob Først gjøre ferdig PGL-56. Oppdatere wiki'en i vår git fork.
- Lars Først gjøre ferdig PGL-56. Oppdatere wiki'en i vår git fork.
- Sahand Fortsette bugfiksing.
- Nawar Fortsette bugfiksing.

I.21 Daily Scrum, 13. Mars

- Jakob Startet med PGL-35, nesten ferdig.
- Lars Startet med PGL-35, nesten ferdig.
- Sahand Ferdig med PGL-40, reviewet PGL-36, skrevet manual om PGL-40
- Nawar Ferdig med PGL-40, reviewet PGL-36, skrevet manual om PGL-40

- Jakob Gjøre ferdig PGL-35, rapport.
- Lars Gjøre ferdig PGL-35, rapport.
- Sahand Rapport.
- Nawar Rapport.

I.22 Daily Scrum, 14. Mars

- Jakob PGL-35, review PGL-40, wiki og PGL-54 ferdig.
- Lars ferdig PGL-35, skrev teori og diskusjon om PGL-35
- Sahand Readme filen, fikset automatisk akse i PGL-34, opprydding av PGL-40 etter review.
- Nawar Undersøkte hvilke alternativer som finnes for å lage 3d moddeller til rapporten, opprydding av PGL-40 etter review.

- Alle Fortsette med rapport.

I.23 Daily Scrum, 16. Mars

- Jakob Jobbet med PGL-59, effektivitets testing. Nærmer oss ferdig der. Lynkur for rapport.

- Lars Jobbet med PGL-59, effektivitets testing. Nærmer oss ferdig der. Lynkur for rapport.
- Sahand Rettet opp i et par ting i planes funksjonen og skalar akse. (Ting som ble tatt opp på møtet 14.03) Begynt å jobbe med komprimitering
- Nawar Rettet opp i et par ting i planes funksjonen og skalar akse. (Ting som ble tatt opp på møtet 14.03) Begynt å jobbe med komprimitering

- Jakob Bli ferdig med PGL-59, og starte med PGL-37
- Lars Bli ferdig med PGL-59, og starte med PGL-37
- Sahand Begynne med pgl-37
- Nawar Begynne med pgl-37

I.24 Daily Scrum, 17. Mars

- Jakob Ferdig PGL-59, nesten ferdig PGL-39. Hjulpet med mattematikken i PGL-37 ved å lage psudo-kode.
- Lars Ferdig PGL-59, nesten ferdig PGL-39.
- Sahand Jobbet med PGL-37
- Nawar Jobbet med PGL-37

- Jakob Rapport, srkive om PGL-35, PGL-36. Møte med Ivar. Skrive ferdig testing for PGL-39
- Lars Rapport, srkive om PGL-35, PGL-36. Møte med Ivar. Skrive ferdig testing for PGL-39
- Sahand Skrive om diskusjoner for PGL-34 og PGL-40
- Nawar Jobbe med å få rapporten til å kompilere i sharelatex

I.25 Daily Scrum, 20. Mars

- Jakob Jobbet med generell struktur, møte med Ivar, PGL-39. Leste på feio-torres og skjønte den!
- Lars Jobbet med generell struktur, møte med Ivar, PGL-39
- Sahand Skrevet om diskusjonene på PGL 37.
- Nawar Jobber med rapport, ordnet Latex slik at den nå kompilerer. "æ ø å"i filnavn er ikke lov.

- Jakob PGL-58, fikse points for PGL-35, møte med Ivar, PGL-39, Rapport
- Lars PGL-58, fikse points for PGL-35, møte med Ivar, PGL-39, Rapport
- Sahand PGL-37.
- Nawar PGL-37.

I.26 Daily Scrum, 22. Mars

- Jakob Feito torres rapport, møte med Ivar.
- Lars Feito torres rapport, møte med Ivar.
- Sahand PGL 37, Feilsøking i PGL-37.
- Nawar PGL 37, Feilsøking i PGL-37.

- Jakob Tenge illustasjoner i Onenote for feito torres, skrive teori om feito torres.
- Lars Rapport om feito, teori om orientasjon, vertex og simplex. orginal linje/tre-

kant/tetrahedron
- Sahand Fortsette med PGL-37.
- Nawar Fortsette med PGL-37.

I.27 Daily Scrum, 23. Mars

- Jakob Skrev rapport om PGL-36, Skrev om feito torres algoritmen under metode kapittellet. Skrevet test for pgl-39.
- Lars Fikset sharelatex visual bugs, kilder, chapters. Skrevet test for pgl-39.
- Sahand Skrevet om diskusjon for PGL-40.
- Nawar Fikset sharelatex visual bugs, kilder, chapters.

- Jakob Jobbe med rapport, fokus på å skrive ferdig om feito-torres. Deretter skrive introduksjons kapitell.
- Lars Jobbe med rapport, fokus på å skrive ferdig om feito-torres. Deretter skrive introduksjons kapitell.
- Sahand Reviewe PGL-58 og PGL-39. Fortsette utvikling av PGL-37.
- Nawar Reviewe PGL-58 og PGL-39. Fortsette utvikling av PGL-37.

Generell diskusjon: Vi må begynne å srkive rapport deler utover kun direkte tilknyttet PGLer.

I.28 Daily Scrum, 24. Mars

- Jakob Merge ferdige PGL-er med master. Skrevet rapport om PGL-36. Reviewet PGL-37.
- Lars Merge ferdige PGL-er med master. Skrevet rapport om PGL-36. Reviewet PGL-37.
- Sahand Fikset noen bugs i PGL-37 og testet.
- Nawar Fikset noen bugs i PGL-37 og testet.

- Jakob Rapport.
- Lars Rapport.
- Sahand Rapport.
- Nawar Rapport.

- Nawar ikke møtte til møtet.
- Brukte tid på å gå over reviewen av PGL-37 og forklare mattematikken bak hva vi tror feilen angående avhengighet av center.

I.29 Daily Scrum, 27. Mars

- Jakob Illustrasjoner, rapport om PGL-36
- Lars Rapport om PGL-36.
- Sahand Rapport om PGL-40 og PGL-34.
- Nawar Rapport om PGL-40. Så på PGL-37 etter review, må endres.

- Jakob Effektivisere Feito-torres.
- Lars Effektivisere Feito-torres.
- Sahand PGL-37 rette opp i feil.
- Nawar PGL-37 rette opp i feil.

- Spørsmål: Hvordan skal vi skrive om arbeidsprosess, hvor detaljert?
- Svar: Hold det overordnet og belys større valg i prosessen.

I.30 Daily Scrum, 28. Mars

- Jakob Effektiviisert feito-torres, 30 ganger raskere. Innstallert profiler.
- Lars Effektiviisert feito-torres, 30 ganger raskere. Innstallert profiler.
- Sahand PGL-37, fikset håndtering av Nx..xMx3 for PGL-37.
- Nawar PGL-37, fikset håndtering av Nx..xMx3 for PGL-37.

- Jakob Gå over referater og få de klare for rapport vedlegg.
- Lars Oppdatere wiki med nye funksjoner som er lagd denne sprinten, legge til eksempeler.
- Sahand PGL-37.
- Nawar PGL-37.

I.31 Daily Scrum, 29. Mars

- Jakob Rapport og møter. Møte med Frode.
- Lars Rapport skriving, Scrum møter, sprint start.
- Sahand Jobbet med PGL-37, feilsøkt mottagelse av feil alpha verdi.
- Nawar Jobbet med PGL-37, feilsøkt mottagelse av feil alpha verdi.

- Jakob Opptatt med student assistent jobb idag.
- Lars Rapport om feito torres og videre med rapport layout.
- Sahand Prøve å bli verdig med PGL-37, hvis blir ferdig, jobber med rapport
- Nawar Prøve å bli verdig med PGL-37, hvis blir ferdig, jobber med rapport

- Møte med Frode: Dersom vi bruker engelske ord som er velkjente slik som "contructorkan disse skrives på engelsk, men dersom ordet skal bøyes må det skrives på norsk. Eksempelvis Lagd contructor for gamut", Endret på begge kontruktørene slik at de ..."
- Misnøye i gruppen: Missnøye av Nawar om at Jakob ikke er på møtet.

I.32 Daily Scrum, 31. Mars

- Jakob Skrevet litt rapport og lagd noen illustrasjoner. Hjulpet til med PGL-37
- Lars Skrevet rapport, mye om feito-torres
- Sahand Rapport og jobbet med PGL-37, tegninger på parpir.
- Nawar Rapport og jobbet på PGL-37

- Jakob Rapport.
- Lars Rapport.
- Sahand Rapport.
- Nawar Rapport.

Snakket om spesial tilfelle for pgl-37

I.33 Daily Scrum, 3. April

- Jakob Illurstrasjoner, rapport: pgl-36 definisjoner, definisjoner omformulering
- Lars Rapport: Pgl-36, tester og tilhørende grafer, prossess

- Sahand Kvalitetsikring med par programmering, teori og PGL-40 og PGL-34
- Nawar Rapport: PGL-37 og dokumentasjon

- Jakob Illustrasjoner, renskriving og formatering av referatene.
- Lars Rapport, skrive resultater for feito torres
- Sahand PGL-37 rette spesialtilfelle: to punkter projekserer på in_trianglereturner den nærmeste, ikke første. Sette inn figurer.
- Nawar PGL-37 rette spesialtilfelle: to punkter projekserer på in_trianglereturner den nærmeste, ikke første. Rapport: Dokumentasjon

Diskusjon: Bruke tid på effektivisering av koden vår?

- Nawar Vil heller utvikle mer.
- Lars Bedre og levere mindre med mer kvalitet.
- Jakob Til syvende og sist Ivar som bestemmer uansett.

Snakket om hva som burde med i illustrasjoner.

- Jakob Påpek alle nødvendige spesialtilfeller, uten å oversvømme tegningen med informasjon.

I.34 Daily Scrum, 4. April

- Jakob Illustrasjoner og formatering av referater. Review av PGL-37. Hodepine.
- Lars Rapportskriving, resultat for feito-torres,
- Sahand Jobbet med PGL-37.Skrevet inn eksempler og mer i wikien.
- Nawar Skrevet rapport om dokumentasjon. Jobbet med PGL-37

- Jakob Formatere ferdig referater. Lese igjennom det Lars har skrevet om feito-torres. Gå over "matematikkdelene i rapporten.
- Lars Siste innpurt på rapport om feito-torres.
- Sahand Endre wiki eksemplene. Skrive rapport.
- Nawar Skrive rapport.

I.35 Daily Scrum, 21. April

- Alle Review og retorspective møte
- Jakob Fikset bug i is_inside(), revidering av tekst om feito torres.
- Lars Fikset bug i is_inside(), rapport.
- Sahand Startet med PGL-61. Rapport om PGL-34
- Nawar Startet med PGL-61. Rapport om PGL-37

- Jakob Revidere ferdig. Begynne på PGL
- Lars Fortsette å skrive om impementering og bruk av bibliotek. Skrive om struktur. Begynne på PGL
- Sahand Jobbe med rapport om PGL-34.
- Nawar Jobbe med rapport PGL-37.

Lars Eksempel på hvordan man legger ved kode i latex finnes i PGL-39.

I.36 Daily Scrum, 25. April

- Jakob Reveidering av det som er skrevet of PGL-36. Reasearsh på løsning for PGL-45.
- Lars Rapportskriving om struktur. Reasearsh på løsning for PGL-45.
- Sahand Rapport om PGL-34, ferdig!
- Nawar Rapport om PGL-37, diskusjon, revidering og lagt til bilder.

- Jakob Revidere ferdig feito-torres kapitell. Starte med PGL-45 utvikling.
- Lars Skrive om testing. Starte med PGL-45 utvikling.
- Sahand Jobbe med rapport om PGL-40. Starte med PGL-61 utvikling.
- Nawar Rapport om ferdig om 37. Starte med PGL-61 utvikling.

I.37 Daily Scrum, 26. April

- Jakob
- Lars Var syk, jobbet litt med PGL-45 koding
- Sahand Begynnte på PGL-61, rapport PGL-40
- Nawar Begynnte på PGL-61,

- Jakob
- Lars review av sahand sine rapport skriving, skrive på kildekode(rapport), og koding på PGL-45
- Sahand rapport PGL-40 forbedringer og koding PGL-61,
- Nawar rapport PGL-37, forbedringer og kode PGL-61

I.38 Daily Scrum, 28. April

- Jakob Utvikling av PGL-45, litt rapport og møter.
- Lars Utvikling av PGL-45, litt rapport og møter.
- Sahand —
- Nawar Utvikling PGL-61. Rapport, omskriving vekk fra "vi gjorde", skrevet om testing, rapport om PGL-37.

- Jakob Gjøre ferdig PGL-45.
- Lars Gjøre ferdig PGL-45.
- Sahand —
- Nawar Starte rapport om PGL-61

I.39 Daily Scrum, 1. Mai

- Jakob Feridg med PGL-45 og PGL-48. Rapport oppgave beskrivelse, vurdering av gruppens arbeid.
- Lars Feridg med PGL-45 og PGL-48
- Sahand PGL-61 (minDE) ferdig. Rapport omskriving "vi gjordePGL-35.
- Nawar PGL-61 (minDE) ferdig, begynt på rapport om PGL-61

- Jakob PGL-57, Numba og Cython.
- Lars –
- Sahand Rapport om parprogrammering, videre omskriving av PGL-40. m
- Nawar PGL-57, Numba og Cython.

I.40 Daily Scrum, 3. Mai

- Jakob Møte, Rapport om PGL-45
- Lars Møte, Rapport om Kritikk av oppgaven", Rapport organisering". Merge.
- Sahand Ferdig rapport om PGL-40. Rapport om parprogrammeing.
- Nawar Møte. Reviet PGL-er. Rapport.

- Jakob Rapport om PGL-45. Illustasjon for PGL-40.
- Lars Se over PGL-40, PGL-61 og PGL-37 rapport
- Sahand Rapport PGL-include".
- Nawar Skrive ferdig PGL-ene.

Diskusjon: Slå sammen Arbeidsprosess, forarbeid og hoveddel?

I.41 Daily Scrum, 5. Mai

- Jakob Nesten ferdig med rapport om PGL-45
- Lars –
- Sahand Skrev om PGL includeFormaterte matematikk. Review PGL-39, pep8, kildekoder (Rapport).
- Nawar Ferdig rapport om PGL-37 og PGL-61. Reviewet PGL-36 rapport.

- Jakob Ferdigstille PGL-45. Illustrasjoner. Teori hvis tid. Review av 35 (mate matikk formatering)
- Lars Endre PGL-39, kritikk av opg. bes", pep8 og kildekode"ifølge reviw notater fra Sahand. PGL Avhengihetstre.
- Sahand Review "instalasjon og Bruk"
- Nawar Vedlegg.

J Retrospect møtereferater

J.1 Retrospect, 14. Februar

- Fork - Review: hva vil det si? - bra /dårlig

Sahand: Dårlig Vanskelig å komme igang, kastet bort timer som kunne blitt spart om ting kunne blitt avklart med Ivar.

Bra kommunikasjon, hjelpe hverandre videre.

Jakob: Vi må bli flinkere til å bruke mail for konkrete spørsmål til Ivar.

Fokusere mer på rapport. Bra at vi tar enkle notater underveis, men burde se hva som skal til for å få det på rapport format"

Fortsette og gjøre det mer: Forklare hva som er blitt ferdig og hvordan man bruker det. Avklarende spørsmål.

Synes vi ikke skal ha oppe PC-er under møter. Resten.

Nawar: Kommet lengere enn vi trodde. Ikke flinke nok til å dokumentere kode og rapport. Vi burde skrive manual før vi går videre.

Diskusjon: Hva skal man gjøre som review og hvordan skal kommentrar/endringsforslag logføres

Jakob: Block comments for endringsforslag

Resten: Nei.. men vi kan logfør i jira! Men hva kan reviewer gjøre da? Revier skal ikke endre noe kode, men det er lov å endre på kommentarer og pep-8.>

Lars: Synes vi skal begynne med branches, en branch per PGL, og alt i master branch skal være bugfritt. Lars tar seg av all merging r esten: Ja, gjerne!

D.iskusjon: Fungerer det å ikke møtes hver dag(fysisk)? Hamar: Ja, det er bra å slippe å dra til Gjøvik hver dag Lars: men da tar vi det på skype eller facebook hver dag.

Viktig at vi er punktlige med skype-ds.

Fortsatt greit å møtes(fysisk) 1 eller 2 ganger i uka.

J.2 Retrospect, 28. Februar

Manual underveis!! Mer rapport!

Lars: Spørre om hjelp om man ikke kommer noen vei. Vi tar oss tid til å hjelpe. Gå gjennom trello og rapport! Vi må skjerpe oss med time loggføring, viktig for sensor å se at vi har jobbet jevnt og ført timer aktivt. Lars finner en dag i uken som passer der vi skriver rapport SAMMEN.

Jakob: Jakob hjelper med matte psudo? Når vi møtes på skolen blir Jakob på å drøfte løsning. Burde møtes skolen og jobbe med rapport sammen!

Nawar: Bytte par-programmerings par etterverdt. Repson: Gjerne. Men fra neste sprint da Jakob og Lars brude bli ferdig med Feito-torres først. Bra at vi har vert tilgjengelige for hverandre.

Sahand: Alt er sakt

J.3 Retrospect, 14. Mars

Jakob God sprint gjort mye bra jobb og fornøyd med at vi gjorde ferdig alle PGLer. Synes at vi har fått løst problemet bra og har fått lært litt om konflikt håndtering i gruppen.

Lars Lars lovpriser gruppen for bra instats denne sprinten, men bekymret over intern konflikter som har oppstått underveis når en av gruppens medlemmer ikke møtte opp til møte og ikke ga beskjed iforhold til regeler. Ønsker å ta det i fellesskap og løse problemet.

Sahand Det var en god sprint og vi burde ha flere slik framover og synes vi har estimert bra denne gangen og burde fortsette slik. Synes det er bra vi tok opp problemet og løser det fort.

Nawar Føler det var en av de beste sprintene vi har hatt så langt og er veldig stolt over innsatsen vi har gjort. Enig om å ta det i gruppen og ønsker og snakke om det åpnet blant oss for å løse det så fort som mulig og gå tilbake til arbeid.

J.4 Retrospect, 28. Mars

Jakob på mattematikken> Bra> mer aktiv rolle for å forklare mattematikken. Bra utvilking. Føler det er viktig at vi bruker meg til å skrive om mattematikken, og prioriterer mine arbeidstimer på det.

Nawar: Alt var bra, fornøyd med sprinten

Sahand: Jakob flink til å forklare matten, da slipper vi å stille alle spørsmål til Ivar.

Lars: Viktig at vi jobber effektivt fremover, vitkig at vi jobber godt med rapport.

Diskusjon om hva vi skal skrive om PGLer, arbeidsprosses og metode> Viktig å ikke beskrive hver kode linje, men forklare overodnet løsning.

J.5 Retrospect, 20. April

Jakob Ønsker å skrive de "matematiske avsnittenedirekte, istedenfor å endre skrevet tekst. Tror dette tar mindre tid. Diskusjon: Gruppen er enig. Viktig at det blir sammenheng i hele PGL avsnittene.

Lars Bruk det som er skrevet om feito torres som eksempel. Ønsker mer innhold og resultat avsnitt. Tolkning må inneholde alle parametere og return. jobb hver dag med rapport, evt start med det.

Sahand Ønsker å komme fort igang med spritens oppgaver. Diskusjon: Resten av gruppen ønsker hovedfokus på rapport, det er viktigst. Nawar Alt er sakt.

J.6 Retrospect, 5. Mai

Jakob Negativt: Jeg synes vi burde satt Nawar og Sahand på PGL-57 fra begynnelsen av sprinten, da denne oppgaven ikke inneholdt noe matematikk. Positivt: Vært deilig å få skrive en rapport del fra begynnelsen, istedenfor å bli satt på å revidere.

Sahand: Negativt: Synes vi kom tregt igang med programmering denne sprinten. Positivt: Vært flinke til å kommunisere.

Nawar: Negativt: Synes det har gått for mye tid på daily scrums, noen ganger opp til 30min. Burde hvert 15min maks. Positiv: Synes vi har blidt flinkere underveis til å fullføre spritenes oppgaver (De siste sprintene)

Lars: Negativt: Synes folk har vært uselvstendige. Positivt: Oppgaver har blitt gjort.

Prosjektorganisering:

bra: Scrum: Prioritere oppgaver underveis. Kommunikasjon med oppdragsgiver for klarifikasjon om oppgaver.

K Review- og planleggings møtereferater

K.1 Møtereferat, 31. Januar

Punkter i midten av en gamut er alltid irrelevant? - Ja, vi tror det, men snakk med Ivar

Rapportmal: Vi oversetter bare til norsk selv.

Sentrum av gamutt? - Noen fargerom har definerte sentrum, slik som L*a*b der det er 50,0,0. - Tyngdepunktssentrum.

Fargepunkt nærmest cupsen? - Fikk forklar hva cupsen er.

Tips: Lag overskriftene. - Det har vi gjort.

Tips: Color gamut mapping bok, verdt å se på. Janm

K.2 Møtereferat, 2. Mai

Q: Trenger vi å omtale alt fra forrapport A: Kun det som er hensiktmessig.

Til diskusjon: Glemt å loggføre timer i jira?

Q: Hvordan synes dere SCRUM har fungert Ivar: Liker fleksibiliteten, iforhold til å lage en stor kravspek i begynelsen. Estimeringen har vært lærerik, og den har blitt bedre ettervert.

Storfornøyd! Fungert bra med oppklaringer etterhvert. (Onsite customer ish)

Hadde ønsket å få mer tid til å teste selv underveis.

Marius: Stort sett fornøyd. Etterlyst rapport et par ganger. Lars: Vi burde absolutt vært flinkere til å gi delleveranser av rapporten underveis.

Sensor har god peiling på kode: Holde oss til psudokode?

Vi skal ha presentasjon for fagmiljøet.

K.3 Møtereferat, 7. Februar

Møteplan: - Først rapport, så utvikling. Neste tirsdag Sprint slutt. - Review -> Planning poker -> Mer tid? - Tips til rapport

- NS:

- (Fjerne siste dimensjon) nda.shape, fjerne siste verdi fra tupel. shape(a)[:-1]

- Vise til nå hvis tid

RAPPORT: Hva er det lurt å skrive ned ***underveis*** ?

Det vi tror: - Prosess: Hva vi har gjort, hvilke artikler leste vi, Vurderinger Komplikasjoner

Svar: - (Prinsipielle)valg og begrunnelser, diskusjonen, fordeler/ulemper, alternativer - ... og mindre valg - KISS!

Møte neste tirdag: Start 10:30

- (Fjerne siste dimensjon) nda.shape, fjerne siste verdi fra tupel. shape(a)[:-1]

Verticis -> koordinat ->

K.4 Møtereferat, 14. Februar

Navn på gamut surface/ hull? A: Hull

Alltid rasket å initialisere arrayer når vi vet lengden? A: Ja det er rasket, men vi venter med å gjøre noen endriger til effektiviteten blir er problem.

Review PGL-36 - Allerede vist rekursiv. Tenker å gjøre effektivitessammenligning med Ivar's løsning. Tror kanskje den er raskere.

- in_tetrahedron: Bruker (hull == hull + p),

- in_line: kryss == 0 -> coplanar. dot > 0, samme retning. lengde?

- in_triangle: Barycentric koodrinater.

Feito-trorres:

Referat til møte:

Lars viste fram burndown chart, og annet fra Jira. Vi har ikke blitt ferdig med alt. Noe av det som tok mer tid var å skrive testene. Angående tester har Ivar laget en ny user story, der han ber om en pakke colour_test, men en modul for å teste hver modul.

Se på slicing, slice ut de delene vi trenger.

PGL-34: Automatiske akser utifra minimum og maksimum verdier som skal plottes.

Avik på pep-8. Store bokstaver på variabel navn som repesenterer punkter.

Planning poker SUM: 126 PGL-36: 18t PGL-35: 19 PGL-37: 25 PGL-38: 30 PGL-40: 36 PGL-56: 12 PGL-54: 8

K.5 Møtereferat, 28. Februar

Problem tilfelle med at facet_normal dot vertex_center kan gi feil resultat ved modified, kommer vi ikke borti (obs må ekspandere fra samme center)

Hvordan håndtere integrering:

Må antagel sette en annen utstream, og sende pullrequest (Ivar som Aksepterer)

trinagle..

Denne gangen velger vi å ikke gjennomføre PP, men heller bare gange med 1.5

grahm-smidt ortogonalisering projesere ned på planet gitt basisene (som ikke er normal vektorene)

pgl-40

for alle simplexer finn likning for planet regn ut alpha for skjering, og sjekke aplha mellom 0 og 1 bruk aplha til å finne koordinat for skjæring send koordinatet til in_triangle

om innenfor: large i a[] finn korteste vektor i a[]

dokumenter godt om p1 eller p2 er innenfor gamuten.

la likningene for planet bli en del av gammuten. (normal vektor, og distanse)

lag en dictionary planes [srgb]

Annet: Send mail til Ivar om relevante fag.

K.6 Møtereferat, 7. Mars

Vise matten som vi har selv laget og ikke så mye av det som vi har fått av Ivar. Litt dypt også og vis godt at vi har forstått det og klarer og snakke om det på en riktig måte

Rapporten skal skrives i tiden etter vi er ferdig! Fordi all rapport skal være skrevet etter at jobben er ferdig. Derfor skrives den som om det er i ettertid

Skriv i introduksjon hvordan vi har valgt og skrive oppgaven og hvorfor vi velger å skrive det sånn, og hvorfor rapporten blir sene ut som den gjør.

Skriv en test kapitel etter implementasjon

IMRAD er for vitenskapelige artikkler, passer dette? Vi føler I og M flyter litt sammen. Er kravspekk I eller M?

Passer ITMRD bedre? Introduksjon Teori: Her kan vi skrive om nødvendig teori for å forstå atrikkelen. Denne kan enkelt hoppes over av folk med peilingKan redusere behovet for lange forklaringer ellers i rapporten. Metode og Matriale: Kan vi her skrive om hver oppgave for seg? Resultater: Mer overdornet. Slik fungerer ting som en helhet. Diskusjon

Svar: Fint med Teori kapitell etter innledning. Følg heller SU-rapport skriving, ikke så vanlig med IMRAD.

Introduksjon og konklusjon: Lestbar for alle! Test: Skjønner pappa dette?". Det er som er midten kan være mer teknisk.

1.6 utviklingsmodell: Skal vi skrive: Slik var utviklingsmodellen i praksiseller bare ctrl+c fra forrapport? Svar: Slik tenkte vi i forraport, disse justeringene gjorde vi, og dette er hvorfor vi gjorde endringer.

1.7 Rapportorganisering Kan vi her si ifra til leseren om "hopp over 2. og 3 om du ønsker å lese om dette"altså, lage en liten leserguide som man kan finne i bøker. Om du har disse forkunskapene kan du hoppe over ...

Svar: Passer godt, kan forklare med tekst eller figur. 1.7 / Til slutt passer godt.

Marius: Vær stolt over det vi leverer å la det skinne gjennom i rapport og fremmføring.

Underveis: Diskusjon (Diskusjoner), men også under Metode(Metode), Teori. Resultater(Resultat av testene)

Testkapitell etter implementasjon.

Design: Klasse-diagrammer, pakke for testing. Ide: Fargekoding av "våre klasser". Struktur / sysstem arkitektur.

Svar: Ikke så viktig hva overskriftene er, men at rapporten har god flyt og at ting gir mening. 1.7 Kan skrive om valg av struktur.

Kravspesifikasjon i Scrum. Vi føler det blir litt overflødig, er userstories nok kravspekk?

DOD, User Stories, review meeting, review of code. Retrospective. Krav spekk for hver sprint.

Retrospective "use-case"

Matematisk oppgave: Se gjennom tidligere oppgaver selv.

Bakgrunn(Teori): Tema: farge biten", programmering", "matematikken"

Målgruppe for rapporten: Legg språket på nivå slik at medstudenter skal kunne forstå (etter å lest Teori)

Målgruppe for produkt: Hvem vi mener produktet vårt passer for. Svar:For forskere som jobber med bildeproblematikk. Undervisning på høyt nivå. For "opensource"community / stackoverflow".

Gi mulighet for forskere å teste ut ideer før spesialtilpassede GMA-er utvikles.

Kravspesifikasjon

Kan bruke Use-case for å samle.

SCRUM: Kan bruke use-case, aktivitets diagram. Snapp shot av product backlogg.

Kan ta mail som vedlegg. Kan ha med eksmepler/tilnærmigsform til mail løsning.

Martin Nyfløtt

Viktig å få frem systematikken i hvordan vi jobber videre.

K.7 Møtereferat, 14. Mars

Møteplan - Lars viser burndown charts

- Presentere Lars: Kjører testene, integrering og wiki Jakob: PGL-35 Sahand: Plot Nawar: PGL-40

- Spørsmål

-Planlegge neste sprint Planning poker (Ta oss tid til spørsmål om oppgavene)

Spørsmål: Skal vi tilrettelegge for at datapunkter skal kunne legges til undreveis? NEI

Dele på aktuell radius istedenfor r maks avdekker fler konkavheter? STHAPE SPILLET

Referat:

Lars viste frem burndown charts. Alle user stories ferdig for første gang! Lars viste frem det vi har skrevet i wiki'en. Ivar prøvde å følge test_colour veiledningen. Fungerte ikke å kjøre init.py

Ivar: Angående integrering, pull request er bare en beskjed om at man nå kan du integrere oppover. Vi kan alltids pull fra Ivars repo. Vi burde fortløpende pulle fra Ivars repo, slik at vi får mindre merge konflikt.

JL> lage default center hvis ikke sendt med til konstruktor(bruke None)

JL> legge til at points blir lagret –jakob staaaahp

NS> nawar skal gjøre det samme?! linalg, get_linear(); (done)

NS> skalere akser: ivar, kan bli rar fordi skalaene i fargerommene er så forskjellige endre til en prosent av min/max, gjøre litt smartere ellers veldig bEra NS> et eller annet med numpy array og lister,

PGL> lage en moetode som funner ut om et punkt eller en array med punkt sendes med.

PGL> get linear saa, ta vare på shape, reshape

—Planning poker— Timer målt i totale arbeidstimer logget i jira

Vi legger arbeidsmenged til 3uker estimert arbeidsmengde istedenfor 4 uker for å få litt mer tid til rapport.

PGL-37: Ofte er nærmeste punkt en av verteksene.

—Retrospective meeting—

Nawar: Fornøyd med sprinten, og føler vi har forbredredt det vi snakket om sist retrospective.

Lars: Fungert bra at vi spør hverandre mer om hjelp og at Jakob hjelper til med matematikken. Fungerer bra med branches.

Sahand: GODT! Vi har ikke skrevet manualer om alt. Dette må vi gjøre!

Kommunikasjonssvikt. NS-paret skulle bruke in_triangle() og JL-paret tenkte ikke på

at det var en bugg der som vi hadde fikset men fiksen lå på branch, ikke master.

Jakob: Fungert bra med mail korrospondanse med Ivar. Fungert å ikke parprgrammere enkelte ganger, der vi ser det hensiktsmessig.

IMRAD er for vitenskapelige artikkler, passer dette? Vi føler I og M flyter litt sammen. Er kravspekk I eller M?

Passer ITMRD bedre? Introduksjon Teori: Her kan vi skrive om nødvendig teori for å forstå atrikkelen. Denne kan enkelt hoppes over av folk med peilingKan redusere behovet for lange forklaringer ellers i rapporten. Metode og Matriale: Kan vi her skrive om hver oppgave for seg? Resultater: Mer overdornet. Slik fungerer ting som en helhet. Diskusjon

Svar: Fint med Teori kapitell etter innledning. Følg heller SU-rapport skriving, ikke så vanlig med IMRAD.

Introduksjon og konklusjon: Lestbar for alle! Test: Skjønner pappa dette?". Det er som er midten kan være mer teknisk.

1.6 utviklingsmodell: Skal vi skrive: Slik var utviklingsmodellen i praksiseller bare ctrl+c fra forrapport? Svar: Slik tenkte vi i forraport, disse justeringene gjorde vi, og dette er hvorfor vi gjorde endringer.

1.7 Rapportorganisering Kan vi her si ifra til leseren om "hopp over 2. og 3 om du ønsker å lese om dette"altså, lage en liten leserguide som man kan finne i bøker. Om du har disse forkunskapene kan du hoppe over ...

Svar: Passer godt, kan forklare med tekst eller figur. 1.7 / Til slutt passer godt.

Marius: Vær stolt over det vi leverer å la det skinne gjennom i rapport og fremmføring.

Underveis: Diskusjon (Diskusjoner), men også under Metode(Metode), Teori. Resultater(Resultat av testene)

Testkapitell etter implementasjon.

Design: Klasse-diagrammer, pakke for testing. Ide: Fargekoding av "våre klasser". Struktur / sysstem arkitektur.

Svar: Ikke så viktig hva overskriftene er, men at rapporten har god flyt og at ting gir mening. 1.7 Kan skrive om valg av struktur.

Kravspesifikasjon i Scrum. Vi føler det blir litt overflødig, er userstories nok kravspekk?

DOD, User Stories, review meeting, review of code. Retrospective. Krav spekk for hver sprint.

Retrospective "use-case"

Matematisk oppgave: Se gjennom tidligere oppgaver selv.

Bakgrunn(Teori): Tema: farge biten", programmering", "matematikken"

Målgruppe for rapporten: Legg språket på nivå slik at medstudenter skal kunne forstå (etter å lest Teori)

Målgruppe for produkt: Hvem vi mener produktet vårt passer for. Svar:For forskere som jobber med bildeproblematikk. Undervisning på høyt nivå. For "opensource"community / stackoverflow".

Gi mulighet for forskere å teste ut ideer før spesialtilpassede GMA-er utvikles.

Kravspesifikasjon

Kan bruke Use-case for å samle.

SCRUM: Kan bruke use-case, aktivitets diagram. Snapp shot av product backlogg.

Kan ta mail som vedlegg. Kan ha med eksmepler/tilnærmigsform til mail løsning.

Martin Nyfløtt

Viktig å få frem systematikken i hvordan vi jobber videre.

Lars viste frem rapporten.

Marius: Pass på å ikke få for små avsnitt. Eksempel slå sammen sykdom og fravær.

Q: Noe fra forrapport som ikke skal være med? S: Nei, tror ikke det.

1.10 Rapport og layout Struktur forklart for hver målgruppe.

Q: Dele inn i BakgrunnsgrupperS: Mattematikk,

Eventulelt beskriv overordnet hva som er med i de forskjellige kapittelene og la brukeren velge selv hva han vil hoppe over.

1.5 Prosjektgruppen Bakgrunn vi hadde

Kunnskap vi måtte tilegne oss. Geometrisk mattematikk"ikke matte 3

K.8 Møtereferat, 22. Mars

Marius: Bruk chapter, section og section for å fikse rapportstruktur.

1.9 Utviklingsmoddel Her skriver vi om utviklingsmoddellen slik den startet, endringer vi gjorde og hvordan den ble anvendt. Henvis til diskusjoner om valg senere.

2.1 Kravspekk

Q: Fornuftig med Jira snappshot -> Vår tolkning S: Ja!Tekst fra ivar i kursiv. Overordnet beskrivelse først (Skrive til slutt)

Definisjonsliste i bakgrunn og teori kapittel først så anta at lesere av kravspekk har den kunnskapenen de trenger.

Q: Oppdeleing i PGL? S: I teori burde vi samle etter logisk inndeling, all matematikk forklares et sted.

Teori: Eksempel pgl-35 Bare "hva er modified convex hull. Ikke forlkar vår løsning. Mye av det vi nå har i teori, må flyttes til metode. Teori er BAKGRUNNSKUNNSKAP.

Skriv om grunnleggende opperasjoner. Kan nevne hvilke metoder som finnes.

Bruk illustasjoner!

Metode / Resultater Grupper etter PGL

Skriv ifølge PGL skrive guide, og strukturer siden.

K.9 Møtereferat, 28. Mars

Møteplan: - Lars viser burndowncharts etc - Jakob viser frem ny feito - Jakob viser Komprimiter - - Sahand: PGL-37

Neste srpint: Legge inn refactor(points,.. generellt)?

Referat: Lars snakket generelt om sprinten. Vi ble ferdig med alt bortsett fra PGL-37 somm er nesten ferdig. Lars viste frem effektiviseringsmåling gammel og ny løsning av feito_torres() av med profilerverktøy. Jakob forklarte den nye løsningen av feito_torres() Lars viste frem compress_axis()

Sahand viste frem get_clip_nearest() Ivar snakket om matematikken i PGL-37

Jakob forespurte en task til neste sprint som går på å finpusse"

Vi så på burndown charts igjen for å se hvordan estimatene fra sist var. Siste estimeringsrunde traff godt.

Vi hadde en diskusjon angpende sprint lengde og neste møte som klasjer med app-utvikling

Vi planlegger en sprint med lite arbeid, og mye rapport som avsluttes torsdag etter påsken på grunn av at 3/4 utviklere trenger tid på app utvikling eksamens forbredelser.

K.10 Møtereferat, 4. April

Q: Skal vi bruke - mellom PGL-dokumentasjon eller uten -? S: Si det..? Viktigste er at vi er kosistene. Vi sjekker opp.

Q: Kvalitetssikring parprogrammering hva det skal stå der? S: Reflekter, ikke si PP-arprogrammering er..Få med erfaringer om hva som har fungert ikke fungert og back det opp"med parprogrammerings litteratur"

Q: Generellt figurer, oppsett, plasering, tekst under figuren S: Ser bra ut. Viktig med tekst tilhørende figur som forklarer figuren.

Q: Hvordan liker han oppsettet i feks feito-torres S: Kravspekk: Bra!

Q: Diskusjon integrert med arbeidsprosses-pgl, OK? S: Dele inn i 1:"innsamling av informasjon(Det har andre gjort, sammendrag av hva som finner som er relatert til problemstillingen)", 2: "implimentering/koding (Vår løsning av PGLen)", 3: testingfor økt leslighet. Fungerer med diskusjon underveis, men da helst som egne avsnitt. Større diskusjoner som påvirker flere PGLer kan tas i sluttdiskusjon.

Q: OK figurer? S: Bra nok! Pass på at ting blir leslig (lys oransj) Bra å nangi viktige punkter. (svart hvit print).

Andre innspill fra Marius - Burde ha en del om fellestrekk ved PGL arbeidsprosses og arbeidsprossess for sprinter. - Pass på at tekst i figurer er leselig

-hvordan referere til arbeidet gjort i eks: wiki og kode - Istedenfor vektor grafikk kan vi legge ved høyoppløslig bilder som da heller skaleres ned.

Vi syr sammen main og sender til Marius for gjennomlesning, og kommenterer hvordan tilbakemelding som trengs på de forskjellige stedene.

- Ikke tenkt så mye på side tall, men hva som står på sidene.

Q: Her er mitt spørsmål A: Her refererer Jakob underveis i møtet.

Q: Skal vi bruke - mellom PGL-dokumentasjon eller uten -?

Sahand: Q Kvalitetssikring parprogrammering hva det skal stå der?

-Generellt figurer, oppsett, plasering, tekst under figuren -hvordan liker han oppsettet i feks feito-torres -hvordan referere til arbeidet gjort i eks: wiki og kode

K.11 Møtereferat, 20. April

Info: 4 tester kjører ikke.

Møtestart

Lars viste frem burndowncharts.

Lars og Jakob fortalte om refaktorering.

Planlegge sprint: 128 utvikler timer

Diskusjon rundt hva som skal med i neste sprint. Alternativ 1: Måledata og gamut fra ICC-profil

Estimering Oppgave Utvikler timer 124 PGL-61 MinDE: 16t PGL-45 I vinkel: 64t PGL-48 26t PGL-46 sigmoidal 18t

K.12 Møtereferat, 22. April

Q: Hvordan referere til vedlegg(latex) A: På samme måte som kapittler

Q: Hvordan skal 7.X Evaluering av gruppens arbeid være: vis eks! A: Hent innspirasjon fra tildigere rapporter. Evaluering av prosjekt som arbeidsformhvordan har arbeid godt, bra/dårlighva har vi lærtetc..

Q: EKS:PGL_include, skal denne teksten stå i innledningen eller kan den stå der den står nå? A: Kan stå begge steder, men ikke copy-paste.

Q: Psudo kode?? A: Funger bra

Q: Hvordan refere til "her ligger biblioteket". A: For denne kan vi bruke Tilbake melding på feito-torres (generell struktuk)

Q: Hvordan skrive om PGLer som er like. A: Kan refere til tidligere like steg.

Start med en introduksjon (helt i begynnelsen), trenger bare å være en eller to setninger.

Generell intro kravspekk (kanskje bare i første PGL)

Prøv å skrive mer objektivt/forskningsrettet. Mindr

"Vi gjorde grundig forskning-> Literatursøket viser at disse metodene finnes"

"Metodene presenteres i rekkefølge de ble.."

Forarbeid: Literatursøk og forståelse. All implementasjon burde flyttes til hoveddel.

Kanskje spleise Forarbeid med Hoveddel

Ikke bruke figueren over", altid bruk autoref

Alle figurer skal refereses i teksten.

Flytt "Var bittert å refaktoreretil refleksjonskapittel.

Prioriter å få ferdig alle deler først før vi begynner å prike på detaljer. bruk todo.

Prioriter konsistent kvalitet rapporten.

TODO:Finn en konsistent måte å skrive funksjoner på sign()". Kanskje i kursiv? Se på andre rapporter.

TODO:Begrunn valg av psudo/figur/kode i introduksjon. Vi har brukt det som er mest hennsikstmessig i det enkelte tilfelle.

Rapporten skal ha nok informasjon til at en som leser rapporten skal kunne reprodusere de samme resultatene.

TODO: Hvilken gamut ble brukt for effektiviserings målinger, kan legges i appendix.

TODO: Flytkart over PGL sammenheng.

TODO: Et enkelt kappittel over PGL vi ikke har prioritert å skrive så mye om.

TODO: Under evaluering: kvalitetsikringsmetdo av rapport, kryss lesning.

Kan være lurt å prioritere resterende arbeids oppgaver og sette tidsfrister.

Fordelig av arbeidseidstyper

Jakob: hovedanvar for "matematiske forklaringer"

Sahand: Latex formatering av mattematikk.

Legge innvedlegg

K.13 Møtereferat, 2. Mai

Q: Trenger vi å omtale alt fra forrapport A: Kun det som er hensiktmessig.

Til diskusjon: Glemt å loggføre timer i jira?

Q: Hvordan synes dere SCRUM har fungert Ivar: Liker fleksibiliteten, iforhold til å lage en stor kravspek i begynelsen. Estimeringen har vært lærerik, og den har blitt bedre ettervert.

Storfornøyd! Fungert bra med oppklaringer etterhvert. (Onsite customer ish)

Hadde ønsket å få mer tid til å teste selv underveis.

Marius: Stort sett fornøyd. Etterlyst rapport et par ganger. Lars: Vi burde absolutt vært flinkere til å gi delleveranser av rapporten underveis.

Sensor har god peiling på kode: Holde oss til psudokode?

Vi skal ha presentasjon for fagmiljøet.

L JIRA logg

Time Tracking Report for Python Gamut Library

Only including sub-tasks with the selected version

Progress: 65% **14w 2d 3h 36m** completed from current total estimate of **21w 4d 6h 35m**

Accuracy: -60% Issues in this version are behind the original estimate of **13w 3d 2h** by **8 weeks, 1 day, 4 hours, 35 minutes**.

Key		Summary	Original Estimate	Σ	Est. Time Remaining	Σ	Time Spent	Σ	Accuracy	Σ
PGL-41	TO DO	Generer gamut fra måledata-fil	1w 3d	1w 3d	1w 3d	1w 3d	-	-	on track	on track
PGL-35	DONE	Modified convex hull for beregning av gamut-overflate	1w 1d 6h	1w 1d 6h	1w 5h 29m	1w 5h 29m	1d 31m	1d 31m	on track	on track
PGL-57	DONE	Teste og sammenligne cython og numba for raskere beregninger	1w 3h	1w 3h	4d 5h 30m	4d 5h 30m	5h 30m	5h 30m	on track	on track
PGL-45	DONE	Nærmeste punkt på overflaten i gitt vinkel	1w 3d	1w 3d	4d 4h	4d 4h	3d 4h	3d 4h	on track	on track
PGL-56	DONE	Integrere produsert kode fra sprint #1 i github-repositoriet hva. forking og pull requests	4d 6h	4d 6h	3d 6h	3d 6h	1d	1d	on track	on track
PGL-39	DONE	Komprimerer til gamut i én dimensjon	4d 7h	4d 7h	3d 1h	3d 1h	1d 6h	1d 6h	on track	on track
PGL-46	TO DO	Sigmoidal mapping i gitt koordinat-akse	2d 2h	2d 2h	2d 2h	2d 2h	-	-	on track	on track
PGL-58	DONE	Punkt eller array	2d 2h	2d 2h	1d 7h	1d 7h	3h	3h	on track	on track
PGL-48	DONE	HPminde	1d 5h	1d 5h	1d 4h	1d 4h	1h	1h	on track	on track
PGL-54	DONE	Legge testing til egen python package	2d	2d	1d 2h	1d 2h	6h	6h	on track	on track
PGL-60	DONE	Refactoring av MCH, test-fil	1d 4h	1d 4h	5h	5h	7h	7h	on track	on track
PGL-32	DONE	Opprette gamut-modul	1d 3h	1d 3h	1h	1h	1d 2h 30m	1d 2h 30m	-30m	-30m
PGL-61	DONE	MinDE	1d	1d	0m	0m	1d 4h	1d 4h	-4h	-4h
PGL-59	DONE	Teste om tid spares ved å ikke bruke traverse	2d 2h	2d 2h	0m	0m	4d 3h 30m	4d 3h 30m	-2d 1h 30m	-2d 1h 30m
PGL-40	DONE	Nærmeste punkt på overflaten i en gitt retning	1w	1w	0m	0m	2w 3d 1h 5m	2w 3d 1h 5m	-1w 3d 1h 5m	-1w 3d 1h 5m
PGL-37	DONE	Nærmeste punkt på overflaten i 3d	7h	7h	0m	0m	1w 3d	1w 3d	-1w 2d 1h	-1w 2d 1h
PGL-36	DONE	Avgjør om data er innenfor eller utenfor gamut i gitt rom	1w	1w	0m	0m	4w 4d 1h 30m	4w 4d 1h 30m	-3w 4d 1h 30m	-3w 4d 1h 30m
PGL-34	DONE	Generere data egnet for visualisering av gamut med matplotlib	4d 6h	4d 6h	0m	0m	1w 3d 5h	1w 3d 5h	-3d 7h	-3d 7h
PGL-33	DONE	Beregne convex hull-gamut fra colour.data.Data-objekt	5h	5h	0m	0m	1d 1h	1d 1h	-4h	-4h
		Total		**13w 3d 2h**		**7w 2d 2h 59m**		**14w 2d 3h 36m**		**-8w 1d 4h 35m**

Figur 40: Full oversikt over tidsbruk i JIRA

M Prosjektavtalen

NTNU
Norges teknisk-naturvitenskapelige universitet

PROSJEKTAVTALE

mellom NTNU v/Avd. Informatikk og Medieteknikk (NTNU/AIMT) (utdanningsinstitusjon), og

Ivar Farup, IDI, NTNU

(oppdragsgiver), og

Lars Michael Niebuhr, Nawar Behenam Sahand Lahafdoozian, Jakob Michael Voigt.

(student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra Januar 2017 til Juni 2017

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der AIMT yter veiledning.
   Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra AIMT å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra Gjøvik/AIMT. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. AIMT står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av faglærer/veileder og sensor (intern og ekstern sensor). Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

   Tilgjengeliggjøring i det åpen arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne

prosjektavtalen, og må evt. gi skriftlig melding til studenter og dekan om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av AIMT til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU/AIMT og/eller studenter har interesser.

6. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

7. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

8. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i Fronter. I tillegg leveres et eksemplar til oppdragsgiver.

9. Denne avtalen utferdiges med et eksemplar til hver av partene. På vegne av AIMT er det dekan/prodekan som godkjenner avtalen.

10. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og AIMT som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten AIMT som partner.

11. Når NTNU/AIMT også opptrer som oppdragsgiver, trer NTNU/AIMT inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

12. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene i mellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

13. Deltakende personer ved prosjektgjennomføringen:

NTNU/AIMTs veileder (navn): MARIUS PEDERSEN

Oppdragsgivers kontaktperson (navn): MAK FAREP

Student(er) (signatur): Lars Nicholas dato 19.01.17

[redacted] dato 19.01.17

Samand Lahrafdoozian dato 19.01.17

[redacted] dato 19.01.17

Norges teknisk-naturvitenskapelige universitet

Oppdragsgiver (signatur): _____ dato 19/1-2017

Signert avtale leveres digitalt i Fronter(IMT3912)
Godkjennes digitalt av AIMTs dekan

Om papirversjon med signatur er ønskelig, må papirversjon leveres til AIMT i tillegg.
Plass for evt sign:
AIMT Dekan/prodekan (signatur): _____ dato _____

N Oppgavebeskrivelse

Python-bibliotek for fargegamuter

Bacheloroppgave våren 2017

Oppdragsgiver Fargelaben ved NTNU i Gjøvik

Kontaktperson Ivar Farup, tlf. 61 13 52 27, epost ivar.farup@ntnu.no

Oppgavebeskrivelse

Forskjellige enheter som gjengir fargebilder (skjermer, projektorer, printere etc.) har ulike begrensninger i hvilke farger som kan gjengis. Den totale mengden av farger som kan gjengis på en gitt enhet kalles dens (farge)gamut. Når et bilde skal gjengis på en enhet, må fargeinnholdet i bildet mappes til fargegamuten til enheten. Altså må fargegamuten både kunne beregnes, representeres, visualiseres og mappes til.

Det er gjort mye forskning på dette området på fargelaben gjennom årene, og mange algoritmer er utviklet. Målet med dette prosjektet er å få et enhetlig programvarebibliotek for alle disse algoritmene. Biblioteket skal kunne benyttes fra Python, og må enten være en del av eller i det minste kompatibelt et allerede utviklet bibliotek for konvertering av fargedata og metriske data mellom ulike fargerepresentasjoner.

Biblioteket må kunne

- Lese ut fargedata fra bilder og ICC-profiler og beregne en overflate for disse i henhold til etablerte metoder
- Lage representasjoner av gamutene som er egnet for 3D-visualisering
- Gjøre de grunnleggende beregningene som inngår som ingredienser i etablerte gamut mapping-algoritmer
- Utføre et lite utvalg av standard gamut mapping algoritmer
- Tilby et grensesnitt som er egnet for fremditig implementasjon av nye gamut mapping-algoritmer

O Feito-Torres Pseudokode

Definisjoner

V (Q): Vertexer, der vertex til origo inneholder Q.
+ : V tilhører poisitiv orginal tetrahedra,
eller er første eller siste vertex i positivt fjes
- : V tilhører poisitiv orginal tetrahedra,
eller er første eller siste vertex i positivt fjes

Tøm V+ og V- listen
For alle fjes 1...n
Hvis (0) Q er på fjestet, Q= Inside. Stop.

Hvis (1):
(Q er på linja mellom fjesets første vertex og origo)
OG (Fjesets orginale tetrahedra er positivt orientert OG fjesets første vertex ikke er i V+)
eller (Fjesets orginale tetrahedra er negativt orientert OG fjesets første vertex ikke er i V-)
så gjør(1):
Legg til sign(orginale tetrahedra) i INCLUSION
Registrer fjesets første vertex i V+/-

Hvis (2):
(Q er på linja mellom fjesets siste vertex og origo)
OG ((Fjesets orginale tetrahedra er positivt orientert OG fjesets siste vertex ikke er i V+)
eller (Fjesets orginale tetrahedra er negativt orientert OG fjesets siste vertex ikke er i V-)
så gjør(2):
Legg til sign(orginale tetrahedra) i INCLUSION
Registrer fjesets siste vertex i V+/-
Hvis (3)
For fjesets resterende vertexer ..j
Hvis(3.1)
Q er på orginal trekant mellom første vertex, og vertex nummer j
eller Q er på orginal trekant mellom vertex nummer j og j+1
eller Q er på orginal trekant mellom j+1 og første vertex
Så gjør (3.1)
INCLUSION = 1/2 sign(tetrahedra A: mellom origo, første vertex, vertex j, vertex j+1)
Hvis(3.2)
Er (Q på linja mellom origo og vertex j)
OG 1:(sign(Tetrahedra A) er positivt orientert) OG veterx j til orgio ikke er registrert i V+))
eller 2: Samme bare for negativ orientering
Så gjør(3.2)
Inculsion += sign(tetrahedra A)
Registrer vertex j i V+/-

Hvis(3.3)
Q er inni Tetrahedra A
Så gjør
inclusion += sign(A)

Oppsummert

For alle fjes
Q er på fjeset, den er med, stop
Q er på en orginal kant til første vertex i fjes, men ennå ikke registrert i V+ / V-
INCLUSION += sign(Fjesets orginale tetrahedra)
Registrer vertexen
Q er på en orginal kant til siste vertex i fjes, men ennå ikke registrert i V+ / V-
INCLUSION += sign(Fjesets orginale tetrahedra)
Registrer vertexen

P Gamut kode

Listing P.1: Gamut kode

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  """
5  gamut: Colour metric functions. Part of the colour package.
6
7  Copyright (C) 2013-2016 Ivar Farup, Lars Niebuhr,
8  Sahand Lahafdoozian, Nawar Behenam, Jakob Voigt
9
10 This program is free software: you can redistribute it and/or modify
11 it under the terms of the GNU General Public License as published by
12 the Free Software Foundation, either version 3 of the License, or
13 (at your option) any later version.
14
15 This program is distributed in the hope that it will be useful,
16 but WITHOUT ANY WARRANTY; without even the implied warranty of
17 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
18 GNU General Public License for more details.
19
20 You should have received a copy of the GNU General Public License
21 along with this program.  If not, see <http://www.gnu.org/licenses/>.
22 """
23
24 import numpy as np
25 from scipy import spatial
26 import matplotlib.pyplot as plt
27 from mpl_toolkits.mplot3d import art3d
28 import scipy as sci
29 import colour.data as data
30
31
32 class Gamut:
33     """Class for representing colour gamuts computed in various colour spaces.
34     """
35     def __init__(self, sp, points, gamma=1, center=None):
36         """Construct new gamut instance and compute the gamut. To initialize the
                hull with the convex hull method,
37         set gamma != 1, and provide the center for expansion.
38
39         :param sp : colour.Space
40             The colour space for computing the gamut.
41         :param points : colour.Data
42             The colour points for the gamut.
43         :param gamma : float
44             Decides how much the points are expanded when using modified convex
                hull initializing.
45         :param center : ndarray
46             The gamut center. If one is note provided, the geometric center of
                the points is used.
47         """
48
49         self.data = points       # The data points are stored in the original
               format. Use hull.points for actual points.
50         self.space = sp
51         self.hull = None         # Initialized by initialize_(modified)
               convex_hull
```

```
52          self.vertices = None     # Initialized by initialize_(modified)
                convex_hull
53          self.simplices = None    # Initialized by initialize_(modified)
                convex_hull
54          self.neighbors = None    # Initialized by initialize_(modified)
                convex_hull
55          self.center = None       # Initialized by initialize_(modified)
                convex_hull
56
57          if gamma == 1:
58              self.initialize_convex_hull(center)
59          else:
60              self.initialize_modified_convex_hull(gamma, center)
61          self.fix_orientation()
62
63      def initialize_convex_hull(self, center):
64          """Initializes the gamuts convex hull in the desired colour space
65
66          :param sp : Space
67              The colour space for computing the gamut.
68          :param points : Data
69              The colour points for the gamut.
70          """
71          # Calculate the convex hull
72          self.hull = spatial.ConvexHull(self.data.get_linear(self.space),
                qhull_options='QJ')
73          self.vertices = self.hull.vertices
74          self.simplices = self.hull.simplices
75          self.neighbors = self.hull.neighbors
76          if center is None:
77              self.center = self.center_of_mass(self.get_coordinates(self.vertices
                    ))   # If a center was provided, use it.
78          else:
79              self.center = center
80
81
82
83      def initialize_modified_convex_hull(self, gamma, center):
84          """Initializes the gamut with the modified convex hull method.
85
86          Thanks to Divakar from stackoverflow.
87          http://stackoverflow.com/questions/42763615/proper-python-way-to-work-on
                -
88          original-in-for-loop?noredirect=1#comment72645178_42763615
89
90          :param gamma: float
91              The exponent for modifying the radius.
92          :param center: ndarray
93              Center of expansion.
94          """
95          # Move all points so that 'center' is origin
96          n_data = self.data.get_linear(self.space)
97          n_data_backup = n_data                          # Save a copy of the
                points, unmodified.
98
99          if center is None:
100             self.center = self.center_of_mass(n_data)   # If a center was
                    provided, use it.
101         else:
102             self.center = center                        # If not, use the
                    geometric center as a default.
103
104         shifted = n_data - center                               # Make center
                the local origin
105         r = np.linalg.norm(shifted, axis=1, keepdims=True)      # Get the radius
                 of all points
106         n_data = shifted * (r ** gamma / r)                     # Modify the
```

```
            radius.
107
108         # Calculate the convex hull, with the modified radius's
109         self.hull = spatial.ConvexHull(n_data)      # Set indexes from modified
                points.
110         self.vertices = self.hull.vertices          # Set indexes from modified
                points.
111         self.simplices = self.hull.simplices        # Set indexes from modified
                points.
112         self.neighbors = self.hull.neighbors        # Set indexes from modified
                points.
113         self.center = center
114         self.hull.points = n_data_backup
115
116     def is_inside(self, sp, c_data, t=False):
117         """For the given data points checks if points are inn the convex hull
118
119         :param sp : colour.Space
120             The colour space for computing the gamut.
121         :param c_data : colour.Data
122             Data object with the colour points for the gamut.
123         :param t : boolean
124             True if use the traverse method, false if use the flatten method.
125         :return ndarray
126             A array shape(c_data.get()-1) which contains True for each point
                    included in the convexHull, else False.
127         """
128
129         if t:
130             nd_data = c_data.get(sp)                            # Get the data
                    points as ndarray.
131
132             if nd_data.ndim == 1:                               # If only one
                    point was sent.
133                 return np.array([self._is_inside(nd_data)])   # Returns 1d
                        boolean-array.
134
135             else:
136                 indices = np.ones(nd_data.ndim - 1, int) * -1       #
                        Initialize with negative numbers.
137                 bool_array = np.zeros(np.shape(nd_data)[:-1], bool)  # Create a
                        bool-array with the same shape as the.
138                 self.traverse_ndarray(nd_data, indices, bool_array)  # nd_data(
                        minus the last dimension).
139
140                 return bool_array                               # Returns the
                        boolean array.
141         else:
142             shape = c_data.get(sp).shape[:-1]                   # Nx...xMx3
                    color data needs Nx..xM bool array.
143             bool_array = np.zeros(shape, bool)                  # Create a bool
                    array for storing the results.
144             bool_array = bool_array.flatten()
145
146             n_data = c_data.get_linear(sp)
147
148             for i in range(0, bool_array.shape[0]):  # Call feito
149                 bool_array[i] = self._is_inside(n_data[i])
150
151             bool_array = bool_array.reshape(shape)    # Reshape (without last
                    dimension)
152             return bool_array
153
154     def traverse_ndarray(self, n_data, indices, bool_array):
155         """For the given data points recursively traverse the dimensions to
                check if points are inn the convexhull.
156
```

```
157         :param n_data : ndarray
158             An n-dimensional array containing the remaining dimensions to
                    iterate.
159         :param indices : array
160             The dimensional path to the coordinate.
161             Needs to be as long as the (amount of dimensions)-1 in nda and
                    filled with -1's
162         :param bool_array : ndarray
163             Array containing true/false in last dimension.
164         """
165         if np.ndim(n_data) != 1:                            # Not yet reached a
                leaf node
166             curr_dim = 0
167             for index in np.nditer(indices):                # calculate the
                    dimension number witch we are currently in
168                 if index != -1:                             # If a dimension is
                        previously iterated the cell will
169                     curr_dim += 1                           # have been changed
                            to a non-negative number.
170
171             numb_of_iterations = 0
172             for nda_minus_one in n_data:                       # Iterate over
                    the length of the current dimension
173                 indices[curr_dim] = numb_of_iterations      # Update the path in
                        indices before next recursive call
174                 self.traverse_ndarray(nda_minus_one, indices, bool_array)
175                 numb_of_iterations += 1
176             indices[curr_dim] = -1                          # should reset the
                    indices array when the call dies
177
178         else:                                               # We have reached a
                leaf node
179             bool_array[(tuple(indices))] = self._is_inside(n_data)  # Set the
                    boolean array to returned boolean.
180
181     def _is_inside(self, q):
182         """ Tests if a point q is inside the Gamut(general polyhedra)
183
184             :param q: ndarray
185                 Point to be tested for inclusion.
186             :return: bool
187                 True if q is included in the Gamut.
188             """
189         inclusion = 0
190         v_plus = []     # a list of vertices who's original edge contains P, and
                it's face is POSITIVE oriented
191         v_minus = []    # a list of vertices who's original edge contains P, and
                it's face is NEGATIVE oriented
192         origin = np.array([0., 0., 0.])
193
194         for face in self.simplices:                         # Iterate through
                all the Gamuts facets.
195             facet = self.get_coordinates(face)
196             a = facet[0]
197             b = facet[1]
198             c = facet[2]
199
200             s_t = self.sign(np.array([origin, a, b, c]))    # sign of the face's
                    original tetrahedron
201             s_nt = s_t*-1
202             signs = np.zeros(4)                             # array for indexing
                    the sign values
203             zeros = 0
204
205             # Check if q sees the same side of the tetrahedron's facets as
                    origin does. If this is not true,
206             # point is not inside.
```

```
207             signs[0] = self.sign(np.array([q, a, b, c]))
208             if signs[0] == s_nt:
209                 continue
210             signs[1] = self.sign(np.array([q, a, c, origin]))
211             if signs[1] == s_nt:
212                 continue
213             signs[2] = self.sign(np.array([q, a, origin, b]))
214             if signs[2] == s_nt:
215                 continue
216             signs[3] = self.sign(np.array([q, b, origin, c]))
217             if signs[3] == s_nt:
218                 continue
219 
220             for i in range(0, 3):
221                 if signs[i] == 0:            # If sign[i] is zero, q is on the
                        corresponding face of the facet's
222                     zeros += 1               # original tetrahedron.
223 
224             if signs[0] == 0:                # True if q is on the current
                    facet.
225                 return True
226 
227             elif zeros == 0:                 # Tetrahedra.
228                 inclusion += s_t
229 
230             elif zeros == 1:                 # Triangle.
231                 inclusion += 0.5*s_t
232 
233             elif zeros == 2:                 # Line.
234                 inclusion += 0.5*s_t
235 
236                 if signs[1] == 0 and signs[2] == 0:        # Intersection point
                        is on line is between A and O
237                     if s_t > 0 and np.in1d(face[0], v_plus):
238                         v_plus.append(face[0])
239                         inclusion += s_t
240                     elif s_t < 0 and np.in1d(face[0], v_minus):
241                         v_minus.append(face[0])
242                         inclusion += s_t
243                 elif signs[1] == 0 and signs[3] == 0:      # Intersection point
                        is on line is between B and O
244                     if s_t > 0 and np.in1d(face[1], v_plus):
245                         v_plus.append(face[1])
246                         inclusion += s_t
247                     elif s_t < 0 and np.in1d(face[1], v_minus):
248                         v_minus.append(face[1])
249                         inclusion += s_t
250                 elif signs[2] == 0 and signs[3] == 0:      # Intersection point
                        is on line is between C and O
251                     if s_t > 0 and np.in1d(face[2], v_plus):
252                         v_plus.append(face[2])
253                         inclusion += s_t
254                     elif s_t < 0 and np.in1d(face[2], v_minus):
255                         v_minus.append(face[2])
256                         inclusion += s_t
257 
258         if inclusion > 0:
259             return True
260         else:
261             return False
262 
263     def fix_orientation(self):
264         """Fixes the orientation of the facets in the hull, so their normal
                vector points outwards.
265         """
266 
267         c = self.center_of_mass(self.get_coordinates(self.vertices))
```

```
268
269             for simplex in self.simplices:
270                 facet = self.get_coordinates(simplex)
271                 normal = np.cross((facet[1] - facet[0]), facet[2] - facet[0])  #
                        Calculate the facets normal vector.
272                 if np.dot((facet[0]-c), normal) < 0:            # If the dot product
                        of 'normal' and a vector from the
273                                                                 # center of the
                                                                        gamut to the
                                                                        facet is
                                                                        negative, the
274                                                                 # orientation of the
                                                                        facet needs to
                                                                        be fixed.
275                     a = simplex[2]
276                     simplex[2] = simplex[0]
277                     simplex[0] = a
278
279         @staticmethod
280         def sign(t):
281             """ Calculates the orientation of the tetrahedron.
282
283             :param t: ndarray
284                 shape(4,3) The four coordinates of the tetrahedron who's signed
                        volume is to be calculated
285             :return: int
286                  1 if tetrahedron is POSITIVE orientated(signed volume > 0)
287                  0 if volume is 0
288                 -1 if tetrahedron is NEGATIVE orientated(signed volume < 0)
289             """
290
291             matrix = np.array([  # Creating the matrix for calculating a determinant
                    , representing
292                                [t[0, 0], t[1, 0], t[2, 0], t[3, 0]],  # the signed
                                    volume of the t.
293                                [t[0, 1], t[1, 1], t[2, 1], t[3, 1]],
294                                [t[0, 2], t[1, 2], t[2, 2], t[3, 2]],
295                                [1, 1, 1, 1]])
296             return int(np.sign(sci.linalg.det(matrix)))*-1  # Calculates the signed
                    volume and returns its sign.
297
298         def get_coordinates(self, indices):
299             """Return the coordinates of the points correlating to the the indices
                    provided.
300
301             :param indices: ndarray
302                 shape(N,), list of indices
303             :return: ndarray
304                 shape(N, 3)
305             """
306             return self.hull.points[indices]
307
308         def in_tetrahedron(self, t, p, true_interior=False):
309             """Checks if the point P, pointed to by vector p, is inside(including
                    the surface) the tetrahedron
310                 If 'p' is not guaranteed a true tetrahedron, use interior().
311
312             :param t: ndarray
313                 The four points of a tetrahedron
314             :param p: ndarray
315                 The point to be tested for inclusion in the tetrahedron.
316             :param true_interior: bool
317                 Activate to exclude the surface of the tetrahedron from the search.
318             :return: Bool
319                 True if q is inside, or on the surface of the tetrahedron.
320             """
321
```

```
322         # If the surface is to be excluded, return False if p is on the surface.
323         if true_interior and (self.in_triangle(np.delete(t, 0, 0), p) or
324                               self.in_triangle(np.delete(t, 1, 0), p) or
325                               self.in_triangle(np.delete(t, 2, 0), p) or
326                               self.in_triangle(np.delete(t, 3, 0), p)):
327             return False
328 
329         # Check if 'p' is in the tetrahedron.
330         hull = spatial.Delaunay(t)    # Generate a convexHull representation of
                the points
331         return hull.find_simplex(p) >= 0        # return True if 'p' is a vertex
                .
332 
333     @staticmethod
334     def in_line(line, q, true_interior=False):
335         """Checks if a point P is on the line segment AB.
336 
337         :param line: ndarray
338             line segment from point A to point B
339         :param q: ndarray
340             Vector from A to P
341         :return: Bool
342         :param true_interior: bool
343             Set to True if you want to exclude the end points in the search for
                inclusion.
344         :return: Bool
345             True is P in in the line segment from A to P.
346         """
347         if true_interior and (tuple(q) == tuple(line[0]) or tuple(q) == tuple(
                line[1])):
348             return False
349 
350         b = line[1] - line[0]   # Move the line so that A is (0,0,0). 'b' is the
                vector from A to B.
351         p = q - line[0]         # Make the same adjustments to the points. Copy
                to not change the original q
352 
353         # Check if the cross b x p is 0, if not the vectors are not collinear.
354         matrix = np.array([[1, 1, 1], b, p, ])
355         if np.linalg.det(matrix) != 0:
356             return False
357 
358         # Check if b and p have opposite directions
359         dot_b_p = np.dot(p, b)
360         if dot_b_p < 0:
361             return False
362 
363         # Finally check that p-vector is than shorter b-vector
364         if np.linalg.norm(p) > np.linalg.norm(b):
365             return False
366 
367         return True
368 
369     def in_triangle(self, triangle, q, true_interior=False):
370         """Takes three points of a triangle in 3d, and determines if the point w
                is within that triangle.
371             This function utilizes the baycentric technique explained here
372             https://blogs.msdn.microsoft.com/rezanour/2011/08/07/barycentric-
                    coordinates-and-point-in-triangle-tests/
373 
374         :param triangle: ndarray
375             An ndarray with shape: (3,3), with points A, B and C being triangle
                    [0]..[2]
376         :param q: ndarray
377             An ndarray with shape: (3,), the point to be tested for inclusion in
                    the triangle.
378         :param true_interior: bool
```

```
379             If true_interior is set to True, returns False if 'P' is on one of
                    the triangles edges.
380
381         :return: bool
382             True if 'w' is within the triangle ABC.
383         """
384
385         # If the true interior option is activated, return False if 'q' is on
                one of the triangles edges.
386         if true_interior and (self.in_line(triangle[0:2], q) or
387                               self.in_line(triangle[1:3], q) or
388                               self.in_line(np.array([triangle[0], triangle[2]]),
                                        q)):
389             return False
390
391         # Make 'A' the local origin for the points.
392         b = triangle[1] - triangle[0]  # 'b' is the vector from A to B
393         c = triangle[2] - triangle[0]  # 'c' is the vector from A to C
394         p = q - triangle[0]            # 'p' is now the vector from A to the
                point being tested for inclusion
395
396         # If triangle is actually a line. It is treated as a line.
397         if np.array_equal(triangle[0], triangle[1]):
398             return self.in_line(np.array([triangle[0], triangle[1]]), p)
399         if np.array_equal(triangle[0], triangle[2]):
400             return self.in_line(np.array([triangle[0], triangle[2]]), p)
401         if np.array_equal(triangle[1], triangle[2]):
402             return self.in_line(np.array([triangle[1], triangle[2]]), p)
403
404         b_x_c = np.cross(b, c)         # Calculating the vector of the cross
                product b x c
405         if np.dot(b_x_c, p) != 0:      # If p-vector is not coplanar to b and c-
                vector, it is not in the triangle.
406             return False
407
408         c_x_p = np.asarray(np.cross(c, p))          # Calculating the vector of
                the cross product c x p
409         c_x_b = np.asarray(np.cross(c, b))          # Calculating the vector of
                the cross product c x b
410
411         if np.dot(c_x_p, c_x_b) < 0:    # If the two cross product vectors are
                not pointing in the same direction. exit
412             return False
413
414         b_x_p = np.asarray(np.cross(b, p))          # Calculating the vector of
                the cross product b x p
415
416         if np.dot(b_x_p, b_x_c) < 0:  # If the two cross product vectors are not
                 pointing in the same direction. exit
417             return False
418
419         denom = np.linalg.norm(b_x_c)
420         r = np.linalg.norm(c_x_p) / denom
421         t = np.linalg.norm(b_x_p) / denom
422
423         return r + t <= 1
424
425     @staticmethod
426     def is_coplanar(p):
427         """Checks if the points provided are coplanar. Does not handle more than
                 4 points.
428
429         :param p: ndarray
430             The points to be tested
431         :return: bool
432             True if the points are coplanar
433         """
```

```
434         if p.shape[0] < 4:  # Less than 4 p guarantees coplanar p.
435             return True
436 
437         # Make p[0] the local origin, and d, c, and d vectors from origin to the
                other points.
438         b = p[1] - p[0]
439         c = p[2] - p[0]
440         d = p[3] - p[0]
441 
442         # Coplanar if the cross product vector or two vectors dotted with the
                last vector is 0.
443         return np.dot(d, np.cross(b, c)) == 0
444 
445     @staticmethod
446     def center_of_mass(points):
447         """Finds the center of mass of the points given. To find the "geometric
                center" of a gamut
448         lets points be only the vertices of the gamut.
449 
450         Thanks to: http://stackoverflow.com/questions/8917478/center-of-mass-of-
                a-numpy-array-how-to-make-less-verbose
451 
452         :param points: ndarray
453             Shape(N, 3), a list of points
454         :return: center: ndarray
455             Shape(3,), the coordinate of the center of mass.
456         """
457         cm = points.sum(0) / points.shape[0]
458         for i in range(points.shape[0]):
459             points[i, :] -= cm
460         return cm
461 
462     def get_vertices(self, nd_data):
463         """Get all hull vertices and save them in a array list.
464 
465         :param nd_data : ndarray
466             Shape(N, 3) A list of points to return vertices from. The a copy of
                    gamut.points pre-converted
467             to a desired colour space.
468         :return: ndarray
469             The coordinates of the requested vertices
470         """
471         point_list = []                     # Array list for the vertices.
472 
473         for i in self.hull.vertices:        # For loop that goes through all the
                vertices
474             point_list.append(nd_data[i])   # and for each goes to the points
                    and adds the coordinate to the list.
475 
476         return np.array(point_list)          # Returns ndarray.
477 
478     def plot_surface(self, sp, ax):
479         """Plot all the vertices points on the received axel
480 
481         :param ax: Axel
482             The axel to draw the points on
483         :param sp: Space
484             The colour space for computing the gamut.
485         """
486         nd_data = self.data.get_linear(sp)              # Creates a new ndarray
                with points
487         points = self.get_vertices(nd_data)             # ndarray with all the
                vertices
488         x = points[:, 0]
489         y = points[:, 1]
490         z = points[:, 2]
491 
```

```
492         x.sort()                                        # Sort the value from
                smallest to biggest value.
493         y.sort()
494         z.sort()
495
496         for i in range(self.hull.simplices.shape[0]):   # Iterates and draws all
                 the vertices points
497             tri = art3d.Poly3DCollection([self.hull.points[self.hull.simplices[i
                    ]]])
498             ax.add_collection(tri)                      # Adds created points to
                     the ax
499
500         ax.set_xlim([x[0] - (x[0] * 0.20), x[-1] + x[-1] * 0.20])  # Set the
                limits for the plot by calculating.
501         ax.set_ylim([y[0] - (y[0] * 0.20), y[-1] + y[-1] * 0.20])
502         ax.set_zlim([z[0] - (z[0] * 0.20), z[-1] + z[-1] * 0.20])
503         plt.show()
504
505     def true_shape(self, points):
506         """Removes all points in 'points' the does not belong to it's convex
                polygon.
507             Works with 4 or less coplanar points.
508
509         :param points: ndarray
510             Shape(N, 3) Points in 3d
511         :return: ndarray
512             The vertices of a assuming it is supposed to represent a convex
                    shape
513         """
514
515         # Remove duplicate points.
516         uniques = []  # Use list while removing
517         for arr in points:
518             if not any(np.array_equal(arr, unique_arr) for unique_arr in uniques
                    ):
519                 uniques.append(arr)
520         uniques = np.array(uniques)                     # Convert back to ndarray.
521
522         if uniques.shape[0] < 3:                        # one or two unique points are
                garaunteed a point or line.
523             return uniques
524
525         if uniques.shape[0] == 3:                       # If we have 3 points, they are
                 either a triangle or a line.
526             i = 0
527             while i < 3:
528                 a = np.delete(uniques, i, 0)
529                 if self.in_line(a, uniques[i]):  # If a point is on the line
                        segment between two other points
530                     return a                            # Return that line segment.
531                 i += 1
532             return uniques                              # Guaranteed to be a triangle.
533
534         i = 0
535         while i < 4:
536             b = np.delete(uniques, i, 0)
537             if self.in_triangle(b, uniques[i]):  # See if any of the points lay
                    inside the triangle formed by the
538                 return b                                # other points
539             i += 1
540
541         return uniques                                  # return a convex polygon with
                4 vertices
542
543     def in_polygon(self, points, q, true_interior=False):
544         """Checks if q is in the polygon formed by pts
545
```

```
546         :param points: ndarray
547             shape(4, 3). Points on a polygon. Must be coplanar.
548         :param q: ndarray
549             Point to be tested for inclusion
550         :param true_interior: boolean
551             Activate to exclude the edges from the search
552         :return:
553         """
554         if true_interior:
555             # Divide into two triangles and check their true_interior, and their
                    common edge with is in the true
556             # interior or the polygon
557             return (self.in_triangle(np.array([points[0], points[1], points[2]])
                    , q, true_interior=True) or
558                     self.in_line(np.array([points[1], points[2]]), q,
                            true_interior=True) or
559                     self.in_triangle(np.array([points[1], points[2], points[3]])
                            , q, true_interior=True))
560         else:
561             # Divide in two triangles and see is q is in either.
562             return (self.in_triangle(np.array([points[0], points[1], points[2]])
                    , q) or
563                     self.in_triangle(np.array([points[1], points[2], points[3]])
                            , q))
564
565     def interior(self, points, q, true_interior=False):
566         """ Finds the vertices of pts convex shape, and calls the appropriate
                function to test for inclusion.
567         Is not designed to work with more than 4 points.
568
569         :param points: ndarray
570             Shape(n, 3). 0 < n < 5.
571         :param q: ndarray
572             Point to be tested for inclusion in pts true shape.
573         :param true_interior: boolean
574             Activate to exclude the edges if pts is actually a triangle or
                    polygon with 4 vertices, or the surface
575             if pts is a tetrahedron
576         :return: boolean
577             True if the point was inside.
578         """
579         if self.is_coplanar(points):
580             true_shape = self.true_shape(points)
581             if true_shape.shape[0] == 1:
582                 return np.allclose(true_shape, q)
583             elif true_shape.shape[0] == 2:
584                 return self.in_line(true_shape, q)
585             elif true_shape.shape[0] == 3:
586                 return self.in_triangle(true_shape, q, true_interior=
                            true_interior)
587             elif true_shape.shape[0] == 4:
588                 return self.in_polygon(true_shape, q, true_interior=
                            true_interior)
589             else:
590                 print("Error: interior received to many points, retuning False")
591                 return False
592         else:
593             return self.in_tetrahedron(points, q, true_interior=true_interior)
594
595     @staticmethod
596     def get_alpha(q, center, n):
597         """Get the Alpha value by computing.
598
599         :param q: ndarray
600             The start point.
601         :param center: ndarray
602             The center is a end point in the color space.
```

```
603             :param n: ndarray
604                 The normal and distance value for the simplex
605             :return x: float
606                 Returns alpha value.
607             """
608             x = (n[3] - center[0] * n[0] - center[1] * n[1] - center[2] * n[2]) / \
609                 (q[0] * n[0] - center[0] * n[0] + q[1] * n[1] - center[1] * n[1] + q
610                     [2] * n[2] - center[2] * n[2])
611 
612             return x
613 
614         @staticmethod
615         def find_plane(points):
616             """Find normal point to a plane(simplices) and the distance from p to
617                 the cross point.
618 
619             :param points: ndarray
620                 the start point.
621             :return n: ndarray
622                 Returns ndarray with normal points distance. [x, y, z, distance]
623             """
624 
625             v1 = points[2] - points[0]
626             v2 = points[1] - points[0]
627             n2 = np.cross(v1, v2)                           # Find cross product of 2
628                  points.
629             norm = np.linalg.norm(n2)                       # Find normal point.
630             n3 = n2 / norm                                  # Find the distance.
631 
632             return np.hstack([n3, np.dot(points[1], n3)])  # Add the distance to
633                 numpy array, and return it.
634 
635         def intersection_on_line(self, sp, c_data, center=None):
636             """ Returns an array containing the nearest point on the gamuts surface,
637                  for every point
638                 in the c_data object. Cell number i in the returned array
639                     corresponding to cell number i from the
640                 'c_data' parameter. Handles input on the format Nx...xMx3.
641 
642             :param sp: colour.space
643                 The Colour.space
644             :param c_data: colour.data.Data
645                 Colour.data.Data object containing all points.
646             :param center : ndarray
647                 Center point to use when computing the nearest point.
648             :return: ndarray
649                 Shape(3,) containing the nearest point on the gamuts surface.
650             """
651 
652             if center is None:                              # If no center is defined,
653                 use geometric center.
654                 center = self.center
655 
656             re_data = c_data.get_linear(sp)                 # Get linearised colour data
657 
658             for i in range(0, re_data.shape[0]):            # Do _intersection_on_line
659                 re_data[i] = self._intersection_on_line(sp, re_data[i], center)
660 
661             return data.Data(sp, np.reshape(re_data, c_data.sh))
662 
663         def _intersection_on_line(self, sp, q, center):
664             """Finding the Nearest point along a line.
665 
666             :param sp: Space
667                 The colour space for computing the gamut.
668             :param q: ndarray
669                 The start point.
```

```
663             :param center: ndarray
664                 The center is a end point in the color space.
665             :return: ndarray
666                 Returns the nearest point.
667             """
668
669             new_points = self.data.get_linear(sp)          # Converts gamut to new
                    space
670             alpha = []                                     # a list for all the
                    alpha variables we get
671             for i in self.hull.simplices:                  # Loops for all the
                    simplexes
672                 points = []                                # A list for all the
                        points coordinates
673                 for m in i:                                # Loops through all the
                        index's and find the coordinates
674                     points.append(new_points[m])
675                 point = np.array(points)                   # converts to numpy array
676                 n = self.find_plane(point)                 # Find the normal and
                        distance
677                 x = self.get_alpha(q, center, n)           # Finds the alpha value
678                 if 0 <= x <= 1:                            # If alpha between 0 and
                        1 it gets added to the alpha list
679                     if self.in_triangle(point, self.line_alpha(x, q, center)): #
                            And if its in the triangle to
680                         alpha.append(x)
681             a = np.array(alpha)
682             np.sort(a, axis=0)
683
684             a.sort()
685             nearest_point = self.line_alpha(a[-1], q, center)
686
687             return nearest_point
688
689         @staticmethod
690         def line_alpha(alpha, q, center):
691             """Equation for calculating the nearest point.
692
693             :param alpha: float
694                 The highest given alpha value
695             :param q: ndarray
696                 The start point.
697             :param center: ndarray
698                 The center is a end point in the color space.
699             :return: ndarray
700                 Return the nearest point.
701             """
702             return alpha * np.array(q) + center - alpha * np.array(center)  # finds
                    the coordinates for the nearest point
703
704         def compress_axis(self, sp, c_data, ax):
705             """ Compresses the points linearly in the desired axel and colour space.
706
707             :param sp: colour.space
708                 The colour space to work in.
709             :param c_data: colour.data.Data
710                 The points to be compressed.
711             :param ax: int
712                 Integer representing which axis to do the compressing.
713             :return: colour.data.Data
714                 Returns a colour.data.Data object with the new points.
715             """
716
717             shape = c_data.get(sp).shape   # Save the original shape of the points.
718             points = c_data.get_linear(sp)
719             p_min = 9001
720             p_max = 0
```

```
721
722         for p in points:                # Finding the min and max values along
                given axis of the points to be compressed.
723             if p[ax] > p_max:
724                 p_max = p[ax]
725             elif p[ax] < p_min:
726                 p_min = p[ax]
727
728         g_points = self.get_coordinates(self.vertices)  # Using only vertices to
                find min and max points of the gamut.
729         g_min = 9001
730         g_max = 0
731
732         for p in g_points:              # Finding the min and max values along
                given axis of the points in the gamut.
733             if p[ax] > g_max:
734                 g_max = p[ax]
735             if p[ax] < g_min:
736                 g_min = p[ax]
737
738         delta_p = p_max - p_min         # Calculating the delta values.
739         delta_g = g_max - g_min
740
741         b = delta_g / delta_p           # The slope of the line bx + a
742         a = g_min - b * p_min           # Finding start value for the line
743
744         for i in range(0, points.shape[0]):         # For every point.
745             points[(i, ax)] = b*points[(i, ax)] + a  # Compress the coordinates
                    along the given axis.
746
747         return data.Data(sp, points.reshape(shape))  # Return the points as a
                colour.data.Data object.
748
749     def clip_nearest(self, sp, c_data):
750         """ For each point in c_data return the nearest point on the gamut
                surface.
751
752         :param sp: colour.Space
753             A colour space
754         :param c_data: colour.Data
755             A colour.Data object with the points to use.
756         :return:  colour.Data
757         """
758         # Get linearised colour data
759         re_data = c_data.get_linear(sp)
760
761         # Do _intersection_on_line
762         for i in range(0, re_data.shape[0]):
763             re_data[i] = self._clip_nearest(sp, re_data[i])
764
765         return data.Data(sp, np.reshape(re_data, c_data.sh))
766
767     def _clip_nearest(self, sp, p_outside):
768         """ Finds the nearest point in 3D
769
770         :param sp: colour.Space
771             The colour space for computing the gamut.
772         :param p_outside: ndarray
773             The start point.
774         :return: ndarray
775             Returns the nearest point.
776         """
777         gam = self.data.get_linear(sp)                      # Converts gamut to
                new space
778         new_dis = 9001                                      # High value for use
                 in the if
779         point = None
```

```
780
781         for i in self.vertices:                                 # Goes through all
                the vertices to find the closest
782             distance = np.linalg.norm(p_outside - gam[i])        # Finds the distance
                     for the vertices
783             if distance < new_dis:                              # If distance is
                    shorter than previous distance
784                 new_dis = distance                              # Adds value for new
                        distance
785                 point_index = i                                 # Index for the
                        point
786                 point = gam[i]                                  # Coordinates for
                        the new point
787
788         neighbors = []                                          # List for all the
                neighbors
789         for j in self.simplices:                                # Goes through all
                the simplices
790                                                                 # If the simplex has
                                                                        the vertices
                                                                    as a side
791             if point_index == j[0] or point_index == j[1] or point_index == j
                    [2]:
792                 neighbors.append(self.get_coordinates(j))
793
794         a = -9001
795         for simplex in neighbors:                               # Goes through all
                the neighbors
796             n = self.find_plane(simplex)                        # Finds normal and
                    distance
797             a_new = -n[3] + np.dot(p_outside, n[:3])            # Finds new alpha
                    value
798             if np.absolute(a) > np.absolute(a_new):             # If the alpha value
                     is less than the old value
799                 point_on_plane = (p_outside - a_new * n[:3])    # we find the
                        intersection point
800
801                 if self.in_triangle(simplex, point_on_plane):  # If the point is
                         in triangle we return the point
802                     point = point_on_plane
803
804         return point                                        # If we found no points
                that is in triangle we return the vertex
805
806     def clip_constant_angle(self, sp, c_data, axis):
807         """ For all points in c_data, this method finds the nearest point on the
                 gamut, constrained to the
808         plane defined by axis and each point.
809
810         OBS: Make sure all points in c_data are outside the gamut. This method
                maps all points to
811         the gamuts surface.
812
813         :param sp: colour.space.Space
814             The color space to work in, usually cielab for this method.
815         :param c_data: colour.data.Data
816             A set of colour points.
817         :param axis: int
818             0, 1, 2 indicating with axis to use.
819         :return: colour.data.Data
820             The nearest points.
821         """
822         n_data = c_data.get(sp)
823
824         inside = self.is_inside(sp, c_data)
825
826         for i, value in np.ndenumerate(inside):
```

```
827             if not value:
828                 n_data[i] = self._clip_constant_angle(sp, n_data[i], axis)
829 
830         return data.Data(sp, n_data)
831 
832     def _clip_constant_angle(self, sp, q, axis):
833         """ Find the closes point on the gamuts surface that is also on the
                plane defined by q and axis.
834 
835         Thanks to: Grumdrig
836         http://stackoverflow.com/questions/849211/shortest-distance-between-a-
                point-and-a-line-segment
837 
838         Thanks to: Dan Sunday
839         http://geomalgorithms.com/index.html
840 
841         :param sp: colour.Space
842             The colour space to work in.
843         :param q: ndarray
844             The point for which to fin the closest point on plane.
845         :param axis: int
846             0, 1, 2 indicating with axis to use.
847         :return: ndarray
848             coordinate for the closest point on plane.
849         """
850 
851         distance_nearest = 9001
852         nearest = None
853 
854         # Make a line to define the axis
855         if axis == 0:
856             axis = np.array([[-10, 0, 0], [10, 0, 0]])
857         elif axis == 1:
858             axis = np.array([[0, -10, 0], [0, 10, 0]])
859         else:
860             axis = np.array([[0, 0, -10], [0, 0, 10]])
861 
862         pl = self.find_plane(np.array([q, axis[0], axis[1]]))  # Get the normal
                vector and distance to the plane.
863         point_on_plane = np.array([pl[3] * pl[0], pl[3] * pl[1], pl[3] * pl[2]])
                 # A point on the plane.
864         n = np.array([pl[0], pl[1], pl[2]])
                                                 # Normal vector of the plane.
865 
866         if np.allclose(np.cross(axis[1], q), np.array([0, 0, 0])):
867             print("Error, axis and q does not define a plane. Q:", q, "Clipping
                    to nearest point")
868             return self._clip_nearest(sp, q)
869 
870         for simplex in self.simplices:
871             vertecis = self.get_coordinates(simplex)
872 
873             if np.dot(q-self.center, vertecis[0]-self.center) < 0:  # Make sure
                    the simplex is in roughly the right dir
874                 continue                    # If the angle between q and simplex
                        is over 90, skip this simplex.
875 
876             # # Check that one of the vertecis is cloeser than our current
                    closest point.
877             # #TODO: check if distance_nearest is set to closest vertex, gives
                    more accurate results....
878             #  TODO: but nearest is still the current nearest point.
879             # if np.linalg.norm(vertecis[0] - q) < distance_nearest     \
880             #     or np.linalg.norm(vertecis[1] - q) < distance_nearest \
881             #         or np.linalg.norm(vertecis[2] - q) < distance_nearest:
882             above = []  # List for vertices above the plane. (or on)
883             below = []  # List for vertices below the plane.
```

```
884             for vertex in vertecis:
885                 dot_value = np.dot(vertex, n)
886                 if dot_value >= 0:
887                     above.append(vertex)
888                 else:
889                     below.append(vertex)
890 
891             if not above or not below:  # If the simplex does not have vertices
                    on both side of the plane,
892                 continue  # it does not intersect the plane. Skip this simplex.
893 
894             # We now know the simplex intersects the plane, and is close enough
                    that it might contain the nearest
895             # point. Lets find the line segment for intersection.
896 
897             v = None  # One end point of the line segment of intersection of the
                     simplex and plane.
898             w = None  # The other end point of the line segment of intersection
                    of the simplex and plane.
899 
900             # If there are two point above, a and b are found between the below
                    point and each point above.
901             if len(above) == 2:
902                 t = np.dot(n, (point_on_plane - below[0])) / np.dot(n, above[0]
                        - below[0])
903                 v = below[0] + t * (above[0] - below[0])
904 
905                 t = np.dot(n, (point_on_plane - below[0])) / np.dot(n, above[1]
                        - below[0])
906                 w = below[0] + t * (above[1] - below[0])
907             # If there are two point below, a and b are found between the above
                    point and each point below.
908             else:
909                 t = np.dot(n, (point_on_plane - above[0])) / np.dot(n, below[0]
                        - above[0])
910                 v = above[0] + t * (below[0] - above[0])
911 
912                 t = np.dot(n, (point_on_plane - above[0])) / np.dot(n, below[1]
                        - above[0])
913                 w = above[0] + t * (below[1] - above[0])
914 
915             # Find closest point to q on the line segment from a to b.
916 
917             candidate_nearest = None
918 
919             if np.linalg.norm(w - v) == 0:  # Special case where simplex only
                    intersects in one point.
920                 candidate_nearest = v
921             else:
922                 t = np.dot(q - v, w - v) / np.linalg.norm(w - v) ** 2
923                 if t <= 0:
924                     candidate_nearest = v
925                 elif t >= 1:
926                     candidate_nearest = w
927                 else:
928                     candidate_nearest = v + t * (w - v)  # Find the nearest
                            point on the line.
929 
930             dist_cn = np.linalg.norm(candidate_nearest - q)
931             if dist_cn < distance_nearest:
932                 distance_nearest = dist_cn
933                 nearest = candidate_nearest
934 
935         return nearest
936 
937     def HPminDE(self, c_data):
938         """ A general implementation of the gamut mapping algorithm HPminDE.
```

```
            Maps all points that lie outside of..
939         the gamut to the nearest point on the plane formed by the point and
                the L axe in the CIELAB colour space.
940
941         :param c_data: colour.data.Data
942             The colour points.
943         :return: colour.data.Data
944             The mapped points.
945         """
946         # Call method to do the clipping, perform clipping in CIELAB, and use
                the L[axe 0] axe.
947         return self.clip_constant_angle(data.space.cielab, c_data, 0)
948
949     def minDE(self, c_data):
950         """ A general implementation of the gamut mapping algorithm minDE. Maps
                all points that lie outside of..
951             the gamut to the nearest point on the gamut in CIELAB colour space.
952
953         :param c_data: colour.data.Data
954             Colour.data.Data object containing all points.
955         :return: colour.data.Data
956             Returns the nearest points.
957         """
958         # Colour data in cielab.
959         sp = data.space.cielab
960
961         # Get linearised colour data
962         re_data = c_data.get(sp)
963
964         # Returns true/false for points inside/outside as bool array.
965         check_data = self.is_inside(sp, c_data)
966
967         # Do _intersection_on_line
968         for i, value in np.ndenumerate(check_data):
969             if not check_data[i]:
970                 re_data[i] = self._clip_nearest(sp, re_data[i])
971
972         return data.Data(sp, re_data)
```

Q Gamut testing kode

Listing Q.1: Gamut test kode

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
test_gamut: Unittests for all functions in the gamut module.

Copyright (C) 2017 Lars Niebuhr, Sahand Lahafdoozian,
Nawar Behenam, Jakob Voigt

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""

import unittest
import numpy as np
# import matplotlib.pyplot as plt                 # Used for test_plot, which is
    commented out.
from colour import data, gamut, space

# Global variables.
cube = np.array([[0., 0., 0.],      # 0  vertices
                 [10., 0., 0.],     # 1  vertices
                 [10., 10., 0.],    # 2  vertices
                 [0., 10., 0.],     # 3  vertices
                 [5., 5., 5.],      # 4  non vertices
                 [4., 6., 2.],      # 5  non vertices
                 [10., 10., 10.],   # 6  vertices
                 [1., 2., 3.],      # 7  non vertices
                 [10., 0., 10.],    # 8  vertices
                 [0., 0., 10.],     # 9  vertices
                 [0., 10., 10.]])   # 10 vertices
cube_vertices = np.array([0, 1, 2, 3, 6, 8, 9, 10])  # Vertices for the cube
    above.

line = np.array([[0, 0, 0], [3, 3, 3]])             # Line used in testing.
point_on_line = np.array([1, 1, 1])                 # Point inside the line to
    be tested.
point_not_paralell_to_line = np.array([2, 3, 2])    # Point outside the line to
    be tested.
point_opposite_direction_than_line = np.array([-1, -1, -1])
point_further_away_than_line = np.array([4, 4, 4])

tetrahedron = np.array([[10., 10., 10.], [0., 10., 0.], [0., 0., 0.], [0., 0.,
    10.]])  # Tetrahedron used in testing.
tetra_p_inside = np.array([2., 3., 4.])             # Point inside the
    tetrahedron to be tested.
```

```
51  tetra_p_not_inside = np.array([20., 1., 2.])          # Point outside the
        tetrahedron to be tested.
52  tetra_p_on_surface = np.array([0., 5., 0.])
53
54
55  tetrahedron_three = np.array([[10, 10, 10], [10, 10, 0], [10, 0, 10], [0, 10,
        10]])     # Tetrahedron used in testing.
56
57  # Used in test for is_inside
58  points_1d = np.array([5., 11., 3.])
59  bool_1d = np.array([False])
60  points_2d = np.array([[5., 11., 3.], [3., 2., 1.], [11., 3., 4.], [9., 2., 1.]])
61  bool_2d = np.array([False, True, False, True])
62  points_3d = np.array([[[3., 1., 2.], [3., 2., 4.], [10., 3., 11.], [14., 3.,
        2.]]])
63  bool_3d = np.array([[True, True, False, False]])
64
65  triangle = np.array([[0., 0., 0.], [4., 0., 0.], [0., 0., 4.]])
66  triangle_point_inside = np.array([2., 0., 2.])
67  triangle_point_not_coplanar = np.array([2., 2., 2.])
68  triangle_point_coplanar_but_outside = np.array([5., 0., 3.])
69
70  # Same triangle as above, move by vector (2,2,2)
71  triangle2 = np.array([[2., 2., 2.], [6., 2., 2.], [2., 2., 6.]])
72  triangle2_point_inside = np.array([4., 2., 4.])
73  triangle2_point_not_coplanar = np.array([4., 4., 4.])
74  triangle2_point_coplanar_but_outside = np.array([7., 2., 5.])
75
76  polyhedron = np.array([[38., 28., 30.], [31., 3., 43.],  [50., 12., 38.], [34.,
        45., 18.],
77                         [22., 13., 29.], [22., 2., 31.],  [26., 44., 35.], [31.,
                               43., 22.],
78                         [22., 43., 13.], [13., 43., 11.], [50., 32., 29.], [26.,
                               35., 18.],
79                         [43., 3., 11.],  [26., 3., 44.],  [11., 3., 18.],  [18.,
                               3., 26.],
80                         [11., 45, 13.],  [13., 45., 29.], [18., 45., 11.], [2.,
                               32., 31.],
81                         [29., 2., 22.],  [35., 12., 18.], [18., 12., 34.], [34.,
                               12., 50.],
82                         [34., 50., 45.], [45., 50., 29.], [3., 30., 44.],  [29.,
                               32., 2.],
83                         [30., 28., 44.], [50., 30., 32.], [37., 12., 35.], [44.,
                               28., 35.],
84                         [35., 28., 37.], [32., 30., 31.], [31., 30., 3.],  [38.,
                               30., 50.],
85                         [37., 28., 38.], [38., 12., 37.]])
86
87
88  class TestGamut(unittest.TestCase):
89
90      @staticmethod
91      def generate_sphere(r, n):
92          """Generates a sphere or points. Used in tests to generate gamut, and
                inclusion points.
93
94          :param r: int
95              The radius to the points.
96          :param n: int
97              Number of points to be generated.
98          :return: ndarray
99              Numpy array dim(n,3) with the points of the sphere.
100         """
101         theta = np.random.uniform(0, 2 * np.pi, n)
102         phi = np.random.uniform(0, np.pi, n)
103
104         x = r * (np.sin(phi) * np.cos(theta))
```

```
105         y = r * (np.sin(phi) * np.sin(theta))
106         z = r * (np.cos(phi))
107
108         sphere = np.vstack((x, y, z)).T
109
110         return sphere
111
112     def test_gamut_initialize(self):
113         c_data = data.Data(space.srgb, cube)                # Generating the
                colour Data object
114         g = gamut.Gamut(space.srgb, c_data)
115         self.assertTrue(np.allclose(cube_vertices, g.vertices))  # Check that
                the gamut's vertices are correct.
116
117     def test_is_inside(self):
118         c_data = data.Data(space.srgb, cube)
119         g = gamut.Gamut(space.srgb, c_data)
120
121         c_data = data.Data(space.srgb, points_3d)
122         a = g.is_inside(space.srgb, c_data, t=False)
123         self.assertEqual(a.shape, points_3d.shape[:-1])     # Asserts if shape
                is reduced by 1dim
124         self.assertEqual(a.dtype, bool)                     # Asserts is data
                type in the array is boolean
125         self.assertTrue(np.allclose(a, bool_3d))            # Asserts that the
                returned values are correct
126
127         c_data = data.Data(space.srgb, points_2d)
128         a = g.is_inside(space.srgb, c_data, t=False)
129         self.assertEqual(a.shape, points_2d.shape[:-1])     # Asserts if shape
                is reduced by 1dim
130         self.assertEqual(a.dtype, bool)                     # Asserts is data
                type in the array is boolean
131         self.assertTrue(np.allclose(a, bool_2d))            # Asserts that the
                returned values are correct
132
133         c_data = data.Data(space.srgb, points_1d)
134         a = g.is_inside(space.srgb, c_data, t=False)
135         self.assertEqual(1, a.size)                         # When only one
                point is sent, still returned a array
136         self.assertEqual(a.dtype, bool)                     # Asserts is data
                type in the array is boolean
137         self.assertTrue(np.allclose(a, bool_1d))            # Asserts that the
                returned values are correct
138
139         c_data = data.Data(space.srgb, self.generate_sphere(15, 100))
140         g = gamut.Gamut(space.srgb, c_data)
141
142         c_data = data.Data(space.srgb, self.generate_sphere(10, 15))   # Points
                lie within the sphere(inclusion = true)
143         a = g.is_inside(space.srgb, c_data)
144         self.assertTrue(np.allclose(a, np.ones(a.shape)))              # Assert
                that all points lie within the gamut
145
146         c_data = data.Data(space.srgb, self.generate_sphere(20, 15))   # Points
                lie outside the sphere(inclusion = true)
147         a = g.is_inside(space.srgb, c_data)
148         self.assertTrue(np.allclose(a, np.zeros(a.shape)))             # Assert
                that all points lie without the gamut
149
150     def test_get_vertices(self):
151         c_data = data.Data(space.srgb, cube)  # Generating the colour Data
                object
152         g = gamut.Gamut(space.srgb, c_data)
153         n1_data = np.array([[0, 0, 0],      # 0  vertices     Array with just
                the vertices used for comparison.
154                             [10, 0, 0],     # 1  vertices
```

```
155                             [10, 10, 0],     # 2  vertices
156                             [0, 10, 0],      # 3  vertices
157                             [10, 10, 10],    # 6  vertices
158                             [10, 0, 10],     # 8  vertices
159                             [0, 0, 10],      # 9  vertices
160                             [0, 10, 10]])    # 10 vertices
161
162         vertices = g.get_vertices(cube)
163         self.assertTrue(np.array_equiv(n1_data, vertices))    # Compares return
                array with the known vertices array.
164
165         vertices = g.get_vertices(cube)                       # Calls the
                function and add the vertices to the array.
166         self.assertTrue(np.array_equiv(n1_data, vertices))    # Compares
                returned array with the known vertices array.
167
168     # # Uncomment and run to see that a gamut is plotted.
169     # def test_plot_surface(self):                          # Test for gamut.Gamut.
            plot_surface
170     #     fig = plt.figure()                                # Creates a figure
171     #     ax = fig.add_subplot(111, projection='3d')        # Creates a 3D plot ax
172     #
173     #     c_data = data.Data(space.srgb, polyhedron)        # Generating the colour
            Data object
174     #     g = gamut.Gamut(space.srgb, c_data)               # Creates a new gamut
175     #
176     #     sp = g.space                                      # Specifies the color
            space
177     #     g.plot_surface(sp, ax)                            # Calls the plot
            function
178
179     def test_in_line(self):
180         c_data = data.Data(space.srgb, cube)
181         g = gamut.Gamut(space.srgb, c_data)
182
183         self.assertTrue(g.in_line(np.array([[2, 2, 2], [2, 2, 2]]), np.array([2,
                2, 2])))  # All points equal.
184         self.assertFalse(g.in_line(line, point_not_paralell_to_line))
                        # Point in NOT parallel to line
185         self.assertFalse(g.in_line(line, point_opposite_direction_than_line))
                # Point opposite dir then line
186         self.assertFalse(g.in_line(line, point_further_away_than_line))
                        # Point is is further then line
187         self.assertTrue(g.in_line(line, point_on_line))
                                    # Point is on line
188         self.assertFalse(g.in_line(np.array([[3, 3, 3], [4, 4, 4]]), np.array
                ([5, 5, 5])))  # Point is on line
189
190         self.assertFalse(g.interior(line, point_not_paralell_to_line))
                        # Point in NOT parallel to line
191         self.assertFalse(g.interior(line, point_opposite_direction_than_line))
                # Point opposite dir then line
192         self.assertFalse(g.interior(line, point_further_away_than_line))
                        # Point is is further then line
193         self.assertTrue(g.interior(line, point_on_line))
                                    # Point is on line
194         self.assertFalse(g.interior(np.array([[3, 3, 3], [4, 4, 4]]), np.array
                ([5, 5, 5])))  # Point is on line
195
196     def test_in_tetrahedron(self):
197         c_data = data.Data(space.srgb, tetrahedron)
198         g = gamut.Gamut(space.srgb, c_data)
199
200         self.assertTrue(g.in_tetrahedron(tetrahedron, tetra_p_inside))        #
                Point is on the tetrahedron
201         self.assertFalse(g.in_tetrahedron(tetrahedron, tetra_p_not_inside))   #
                Point is NOT on tetrahedron
```

```
202         self.assertTrue(g.in_tetrahedron(tetrahedron, tetra_p_on_surface))    #
                Point is on a simplex(counts as inside)
203
204         self.assertTrue(g.interior(tetrahedron, tetra_p_inside))              #
                Point is on the tetrahedron
205         self.assertFalse(g.interior(tetrahedron, tetra_p_not_inside))         #
                Point is NOT on tetrahedron
206         self.assertTrue(g.interior(tetrahedron, tetra_p_on_surface))
207
208     def test_in_triangle(self):
209         c_data = data.Data(space.srgb, cube)
210         g = gamut.Gamut(space.srgb, c_data)
211
212         self.assertFalse(g.in_triangle(triangle, triangle_point_not_coplanar))
213         self.assertFalse(g.in_triangle(triangle,
                triangle_point_coplanar_but_outside))
214         self.assertTrue(g.in_triangle(triangle, triangle_point_inside))
215
216         self.assertFalse(g.in_triangle(triangle2, triangle2_point_not_coplanar))
217         self.assertFalse(g.in_triangle(triangle2,
                triangle2_point_coplanar_but_outside))
218         self.assertTrue(g.in_triangle(triangle2, triangle2_point_inside))
219
220         self.assertFalse(g.interior(triangle, triangle_point_not_coplanar))
221         self.assertFalse(g.interior(triangle,
                triangle_point_coplanar_but_outside))
222         self.assertTrue(g.interior(triangle, triangle_point_inside))
223
224         self.assertFalse(g.interior(triangle2, triangle2_point_not_coplanar))
225         self.assertFalse(g.interior(triangle2,
                triangle2_point_coplanar_but_outside))
226         self.assertTrue(g.interior(triangle2, triangle2_point_inside))
227
228     def test_sign(self):
229         c_data = data.Data(space.srgb, cube)
230         g = gamut.Gamut(space.srgb, c_data)
231
232         # The tetrahedron should have a positive signed volume.
233         self.assertEqual(1, g.sign(np.array([[-2, 0, 0], [0, -2, 0], [0, 0, 0],
                [0, 0, 2]])))
234         # The tetrahedron should have no volume.
235         self.assertEqual(0, g.sign(np.array([[0, 0, 0], [2, 0, 0], [0, 2, 0],
                [0.5, 0.5, 0]])))
236         # The tetrahedron should have a negative signed volume.
237         self.assertEqual(-1, g.sign(np.array([[10, 10, 10], [0, 10, 10], [10, 0,
                 10], [10, 10, 0]])))
238
239     def test_is_coplanar(self):
240         c_data = data.Data(space.srgb, cube)
241         g = gamut.Gamut(space.srgb, c_data)
242
243         points = np.array([[0, 0, 0], [2, 2, 0], [3, 3, 0], [1, 1, 0]])  #
                coplanar points
244         self.assertTrue(True, g.is_coplanar(points))
245
246         points = np.array([[0, 0, 1], [2, 2, 0], [3, 3, 0], [1, 1, 0]])  # non-
                coplanar points
247         self.assertFalse(False, g.is_coplanar(points))
248
249     def test_center_of_mass(self):
250         c_data = data.Data(space.srgb, cube)
251         g = gamut.Gamut(space.srgb, c_data)
252
253         cm = g.center_of_mass(g.get_vertices(g.hull.points))   # Get coordinate
                for center of the cube
254         cp = np.array([5., 5., 5.])                            # Point in center
                 of cube.
```

```
255         self.assertEqual(cp.all(), cm.all())                    # Assert true
                that the points are the same.
256
257     def test_true_shape(self):
258
259         c_data = data.Data(space.srgb, cube)
260         g = gamut.Gamut(space.srgb, c_data)
261
262         a = np.array([[0, 0, 0], [2, 2, 2], [2, 2, 2]])
263         g.true_shape(a)
264
265         # Test remove duplicates
266         a = np.array([[0, 0, 0], [2, 2, 2], [0, 0, 0], [2, 2, 2]])
267         self.assertEqual(2, g.true_shape(a).shape[0])
268
269         # Test 3 points on the same line should return outer points
270         a = np.array([[0, 0, 0], [2, 2, 2], [3, 3, 3]])
271         self.assertTrue(np.allclose(g.true_shape(a), np.array([[0, 0, 0], [3, 3,
                3]])))
272
273         # Test 4 points that are actually a triangle
274         a = np.array([[0, 0, 0], [0, 3, 0], [3, 0, 0], [1, 1, 0]])
275         self.assertTrue(np.allclose(g.true_shape(a), np.array([[0, 0, 0], [0, 3,
                0], [3, 0, 0]])))
276
277         # Test 4 points that are all other vertices in a convex polygon
278         a = np.array([[0, 0, 0], [0, 3, 0], [3, 0, 0], [5, 5, 0]])
279         self.assertTrue(np.allclose(g.true_shape(a), np.array([[0, 0, 0], [0, 3,
                0], [3, 0, 0], [5, 5, 0]])))
280
281     def test_modified_convex_hull(self):
282
283         # c_data = data.Data(space.srgb, cube)
284         # g = gamut.Gamut(space.srgb, c_data)
285
286         test_points = np.array([[0, 0, 0],          # 0  vertices  # Array with
                just the vertices used for comparison.
287                                 [10, 0, 0],         # 1  vertices
288                                 [10, 10, 0],        # 2  vertices
289                                 [0, 10, 0],         # 3  vertices
290                                 [10, 10, 10],       # 6  vertices
291                                 [10, 0, 10],        # 8  vertices
292                                 [0, 0, 10],         # 9  vertices
293                                 [0, 10, 10],        # 10 vertices
294                                 [4.999, 4.999, 0]]) # Only a vertex in modified
                                        hull
295
296         c_data = data.Data(space.srgb, test_points)
297         g = gamut.Gamut(space.srgb, c_data, gamma=0.2, center=np.array([5, 5,
                5]))
298
299         self.assertTrue(g.vertices.shape[0] == 9)
300
301     def test_get_alpha(self):
302         c_data = data.Data(space.srgb, cube)    # Generating the colour Data
                object.
303         g = gamut.Gamut(space.srgb, c_data)     # Creates a new gamut.
304         d = [0.001, 0.2, 0.2]
305         center = [10, 11, 14]
306         n = [5, 3, 2, 9]
307         a = g.get_alpha(d, center, n)
308
309     def test_find_plane(self):
310         p_data = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
311         c_data = data.Data(space.srgb, cube)    # Generating the colour Data
                object.
312         g = gamut.Gamut(space.srgb, c_data)     # Creates a new gamut.
```

```
313
314         d = g.find_plane(p_data)
315         r = np.array([-0.57735027, -0.57735027, -0.57735027, -0.57735027])
316         np.alltrue(d == r)
317
318     def test_compress(self):
319         c_data = data.Data(space.srgb, cube)  # Generating the colour Data
                object.
320         g = gamut.Gamut(space.srgb, c_data)  # Creates a new gamut.
321
322         col_data = data.Data(space.srgb, np.array([[15, 15, 15], [8, 8, 8], [5,
                5, 5], [1, 1, 1], [-5, -5, -5]]))
323         re_data = g.compress_axis(space.srgb, col_data, 2).get_linear(space.srgb
                )
324
325         fasit_data = np.array([[15, 15, 10], [8, 8, 6], [5, 5, 5], [1, 1, 3],
                [-5, -5, 0]])
326
327         self.assertTrue(np.allclose(fasit_data, re_data))
328
329     def test_intersectionpoint_on_line(self):
330         c_data = data.Data(space.srgb, cube)
331         g = gamut.Gamut(space.srgb, c_data)
332
333         points = np.array([[15, 5, 5], [5, 15, 5], [5, 5, 15]])
                                # points to map
334         mod_points = np.array([[10, 5, 5], [5, 10, 5], [5, 5, 10]])
                            # wanted result
335
336         c_data = data.Data(space.srgb, points)
                                                # data.Data object
337         re_data = g.intersection_on_line(space.srgb, c_data)
                                  # data.Data object returned
338
339         self.assertTrue(np.allclose(re_data.get_linear(space.srgb), mod_points))
                # assert that the points are changed
340
341     def test_HPminDE(self):
342         c_data = data.Data(space.cielab, cube + np.array([0, -5, -5]))
343         g = gamut.Gamut(space.cielab, c_data)
344
345         points = np.array([[0, 8, 8], [4, 0, 9], [4, 4, 3], [0, 10, 0], [15, 0,
                0]])
346         fasit = np.array([[0, 5, 5], [4, 0, 5], [4, 4, 3], [0, 5, 0], [10, 0,
                0]])
347         c_data = data.Data(space.cielab, points)
348         print("This test should produce an error message for the last point")
349         re_data = g.HPminDE(c_data)
350         re_data = re_data.get_linear(space.cielab)
351
352         self.assertTrue(np.allclose(fasit, re_data))
353
354     def test_minDE(self):
355         sphere = self.generate_sphere(6, 10)
356         sphere = sphere + np.array([5, 5, 5])
357         c_sphere = data.Data(space.cielab, sphere)
358
359         g_cube = data.Data(space.cielab, cube)
360         g = gamut.Gamut(space.cielab, g_cube)
361         mapped_im = g.minDE(c_sphere)
362
363         result = True
364         for index, value in np.ndenumerate(mapped_im.get_linear(space.cielab)):
365             if value > 10:
366                 result = False
367         self.assertTrue(result)
368
```

```
369     def test_clip_nearest(self):
370         c_data = data.Data(space.srgb, cube)
371         g = gamut.Gamut(space.srgb, c_data)
372 
373         points = np.array([[5, 5, 15], [5, 5, 15], [5, 5, 15]])
                                    # points to map
374         mod_points = np.array([[5, 5, 10], [5, 5, 10], [5, 5, 10]])
                                # wanted result
375 
376         c_data = data.Data(space.srgb, points)
                                                    # data.Data object
377         re_data = g.clip_nearest(space.srgb, c_data)
                                           # data.Data object returned
378 
379         self.assertTrue(np.allclose(re_data.get_linear(space.srgb), mod_points))
              # assert that the points are changed
380 
381 
382 
383 if __name__ == '__main__':
384     unittest.main(exit=False)
```

R Timelister

Jakob M. Voigt

Dato	Timer	Beskrivelse	Sum timer:	484.5
9.01	5	Oppstartsdag		
10.01	4	Utviklingsmodeller		
11.01	6	Møter, og start forrapport		
12.01	4.5	Forrapport, grov gjennomgang felles		
16.01	4.5	rapport		
17.01	5.5	rapport		
18.01	7	rapport		
19.01	7.5	Lest anbefaltliteratur, oppgave beskrivelse, utviklingsmodell		
20.01	6.5	Renskrevet mr1701, resultatmål ferdig		
21.01	1	Omskrevet frem til 5.4, -4		
23.01	5.25	rapport		
24.01	8.5	Mail til Ivar/marius, renskriving på rapport.		
25.01	8.25	Skrev om etter tilbakemelding fra Marius		
26.01	4.5	Lest litt om scrum, fikset systemer,		
27.01	8.5	rapport		
30.01	4	Hørt på presentasjon om pep8 og parprog. Visualisering av kjerneoperasjoner, kartlegge hva som fines for "Finne sentrum av gamut"		
31.01	4.5	Convekshull error fiksing, lest scypy.spatial dokumentasjon, gamut constructor og initialize funksjon.		
2.02	6.5	Gamut constructor og initialiseringsfunksjon ferdig, test skrevet. Møte, hjulpet med tolking av PGL-33. Startet reaserach for PGL-36		
3.02	7	Rekursiv funksjon for å håndtere variable antall dim og implementert en enklest mulig metode for å avgjøre om et punkt er innenfor eller utenfor et Chull		
3.02	2	Bugfikses i gamut klassen på den rekursive funksjonen.		
6.02	5	Bugfiksing i travaseringen		
7.02	8	Ferdigstilt traverseringsfunksjonen, PGL-36 magnler nå bare "signle_point_inside"		
8.02	7.5	Lest på Feito-Torres algoritmen.		
9.04	4	Lagd egen psudokode for feito torres		
10.02	7	Starter utvikling av feito torres, lagd in_line og in tetrahedron. Sett etter løsningsmetoder for "punkt i trekant i rommet"		
13.02	8	in_triangle(), små endringer i testene, test for in_triangle()		
14.02	8.5	feito_torres(), sign(), møter		
15.02	3.5	Bugleting i feito_torres komponenter. Ferdigstilt sign()		
16.02	8	Fjernet bug i in_triangle, skrevet vidre på feito_torres		
17.02	6	Tatt ut fridag for LAOS		
21.02	3	Feilsøking i feito_torres		
22.02	3	rapport		
24.02	3	Syk		
26.02	1.5	Litt reaserch for Feito-torrest. "divide-and-conquer" o'rourke 1998 bok.		

Jakob M. Voigt

27.02	**7**	**løst problem med orientering av fasett, og testing av hele feito torres**		
28.02	**7**	**Sprint slutt, og ny sprint start med alle møter som inngår. Jobbet med feito torres tester.**		
1.03	**8**	**bugfixing feito torres. Rewrite in_triangle til å også ha mulighet til "true_interior". muligens fikset feito_torres**		
2.03	**7.5**	**Rydding i feito_torres. Lagde generate_sphere() of test for feito_torres med generate_sphere.**		
7.03	**8.5**	**Feito fungerer, med noen warning. Møte. Startet omstrukturering av feio-torres**		
8.03	**7.5**	**Ferdig med omstrukturering av feito, bruker nå interior() (og true_shape()). Finpussing på komentarer.**		
9.03	**7**	**Wiki om testing, hjalp med mattematikken på pgl-40, startet PGL-35(modified convex hull)**		
10.03	**3.5**	**Modified convex hull**		
13.03	**7.5**	**Ferdig med PGL-35, skrevet diskusjon og teori avsnitt om PGL-35**		
14.03	**7.5**	**Rapport, Møte**		
15.03	**5**	**Lynkurs, PGL-59**		
16.03	**6**	**PGL-59 ferdig, PGL-39 nesten ferdig**		
17.03	**6.5**	**Jobbet med generell struktur, møte med Ivar, PGL-39. Leste på feio-torres og skjønte den!**		
20.03	**5**	**Feito-torres illustasjoner og forklaring, PGL-58, møte med ivar**		
22.03	**8**	**Skrevet rapport, om feito-torres, illustrasjoner på papir**		
23.03	**5.5**	**Rapport arbeid, feito-torres. Hjalp til med PGL-37 bugfiksing.**		
24.03	**6**	**Reviewet PGL-37,**		
	12	**"Cashet inn" fridager.**		
27.03	**6.5**	**Effektivisert feito_torres, er nå 30 ganger raskere. Hjulpet til med mattematikken på PGL-37.**		
28.03	**7**	**Rapport og møter**		
30.03	**3**	**Hjulpet til med PGL-37, fikk løsningen til å fungere.**		
31.03	**6**	**Rapport om feito og illustrasjoner.**		
1.04	**1**	**Illustrasjoner PGL-37, modified convex hull**		
3.04	**8**	**Illustrasjoner, referater og hodepine.**		
4.04	**7.5**	**Formatert ferdig referater, Lest over teori for PGL-34/37/40. Omskrinvinger på PGL-37(Definisjoner, Vi skal gjøre A->Vi gjorde A)**		
19.04	**7**	**PGL-60, modified_conv, kommentering, refaktorering, compress axsis, skrevet om til å bruke list comprehension et par steder**		
20.04	**7**	**Møte, bugs i is_inside, rapport om feito.**		
21.04	**7**	**5 timer rapport, 2 timer PGL-45.**		
24.04	**7**	**Noen timer rapport først på dagen. PGL-45 resten, møte med Ivar om PGL-45**		
25.04	**6**	**Mest utvikling av PGL-45, noe rapport**		
26.04	**3.5**	**Rapport. Ferdig med revidering av feito-torres. Startet revidering av rapport om PGL-39.**		
27.04	**5**	**Utvikling PGL-45**		
28.04	**7**	**Utvikling PGL-45**		
1.05	**5.5**	**Utvikling PGL-57**		
2.05	**5.5**	**Møte, rapport PGL-45**		
3.05	**6**	**Rapport PGL-45**		
4.05	**4**	**Rapport om PGL-45**		
5.05	**7**	**Ferdigstilt rapport om PGL-45 med illustrasjoner, review matematikk formatering av PGL-35. Illustrasjoner.**		
8.05	**7.5**	**Rapport om farger og fargerom**		

Jakob M. Voigt

9.05	**6.5**	**Fikset farger for kodelisting. Teori om gamuter. Siste revidering av kildekode.**		
10.05	**6**	**Rapport om teori.**		
11.05	**8**	**Ferdig med teori. Retter opp etter kommentarer fra Marius.**		
12.05	**7**	**Rapport**		
13.05	**8**	**Rapport**		
14.05	**5.5**	**Rapport**		
15.05	**8**	**Finpusse rapport.**		

Lars M. Niebuhr

Dato	Timer	Beskrivelse	Sum timer:	533.25
man 9/01-17	5	satt opp, bitbcket og jira samt definert oppgaver		
tir 10/01-17	5	jobbet med jira, og sharelatex		
ons 11/01-1	6	møte med ivar og marius, jira og bitbucket.		
tor 12/01-17	5	Skriving, og generellt		
fre 13/01-17	4	Skriving, gjort meg litt mer kjent med jira		
lør 14/01	-			
søn 15/01	-			
man 16/01	6	fikset jira, skriving og litt tulling		
tir 17/01	8	planlegging, skriving av rapport, møte		
ons 18/01	5.5	skriving		
tor 19/01	3	Prosjektavtale, skriving, mal i latex		
fre 20/01	6.5	mal ferdig, skriving, gjennomgang rapport		
lør 21/01	-			
søn 22/01	1	Finskriving 4.1.1, 5.1.1		
man 23/01	7.25	ds, finskriving: 4.1.3, 4.2, 5.4, 6.1.1		
tir 24/01	8.5	messe, rapportskriving, møte		
ons 25/01	8	Skriving rapport		
tor 26/01	4.5	Møte, installering av phyton og div tillegg		
fre 27/01	8	Planning meeting, finskriving rapport		
lør 28/01	3	jobbet med kilder. levert rapport		
søn 29/01	-			
man 30/1	7	Forberedning sprintstart: jira, rapport og testskriving		
tir 31/1	6	timetracking jira, ordning med repo, veiledning, PGL-32 PGL-33		
ons 01/02	2	timetracking i jira!%&(!!!		
tor 02/02	6.5	PGL-36, research etter elgoritme som kan implementeres i python		
fre 03/02	7	Skrevet rekursiv funksjon for å håndtere variable antall dim og implementert en enklest mulig metode for å avgjøre om et punkt er innenfor eller utenfor et Chull.		
lør 04/02	-			
søn 05/02	-			
man 06.02	6.5	PGL-33,34,36 fikset timetracking i jira, fikset norsk språk i latex mal, skrevet litt om fremgangsmåte PGL-36		
tir 07.02	7.5	Travasjering av n-dim array fungerer, og test går igjennom. Rydding av kode. i test-fil og modul-fil		
ons 08.02	7.5	Gjort research på FT vs R22D algoritmen.		
tor 09.02	5	Dekryptert FT algoritmen		
fre 10.02	4	Skrevet funksjoner som regner ut om et punkt er på en linje, og på et tetrahedra		
lør 11.02	-			
søn 12.02	-			
man 13.02	6	PGL-36 feito torres og triangle inside.		
tir 14.02	8.5	PGL-36 feito torres, sign(), review, retrospective og planning meeting		
ons 15.02	5.5	Jira planning, feito torres() og ferdigstillt sign()		

Lars M. Niebuhr

tor 16.02	8	fjernet bug i triangle, skrevet delvis løsning. fortsatt ikke helt fungerende			
fre 17.02	-	LOAS kurs			
lør 18.02	-				
søn 19.02	-				
man 20.02	6	Jobbet med rapport i sharelatex			
tir 21.02	3	Feilsøkt, fjernet en stor bug. gåt over til å skrive god tester			
ons 22.02	5	Jobbbet med rapport og struktur			
tor 23.02	6	Feriedag 1 pga Skidag			
fre 24.02	-				
lør 25.02	-				
søn 26.02	-				
man 27.02	7	løst problem med orientering av fasett, og testing av hele feito torres			
tir 28.02	7	Sprint slutt, og ny sprint start med alle møter som inngår. Jobbet med feito torres tester.			
ons 01.23	9	Research sign funkction. bugfixing feito torres. Rewright in_triangle til å også ha mulighet til "true_interior". muligens fikset feito_torres			
tor 02.03	7.5	Integrert alle delkomponenter av PGL-36 til is_inside. paserer alle tester vi har skrevet så langt. Gjort en del opprydning av kommentarer og parametere. Fjernet single_point_inside, mener den er deprivated og går rett på feito_torres. jobbet med rapport			
fre 03.03	5	Rapportskrivning. retting/gjennomgang av skrevet rapport			
man 06.03	6	PGL-40, hjalp til med matten. rapport skrivning			
tir 07.03	6	PGL-36, gjennomgang rapport			
ons 08.03	7.5	Ferdig med omstrukturering av feito, bruker nå interior() (og true_shape()). Finpussing på komentarer.			
tor 09.03	7	oppdatert readme, hjalp med mattematikken på pgl-40, startet PGL-35(modified convex hull)			
fre 10.03	-	jobbintervju i Oslo			
man 13.03	8.5	reviev av PGL-40, readme, jobbet med PGL-35, merging med master			
tir 14.03	7	merging med ivar sitt repo, retro og review meeting			
ons 15.03	5	lynkurs, rapportstrukturering, PGL-59			
tor 16.03	6	Gjort oss ferdige med PGL-59, started med PGL-39, og skrevet test			
fre 17.03	4	Jobbet med rapport, PGL-39			
man 20.03	5	Feito-torres rapport(teori og metode), PGL-58, møte med oppdragsgiver ang hastighet på feito-torres			
tir 21.03	6	Feriedag 2 pga jobbintervjue i Oslo			
ons 22.03	7	Skrevet rapport, om feito-torres,			
tor 23.03	5.5	Rapport arbeid, feito-torres. Hjalp til med PGL-37 bugfiksing.			
fre 24.03	6	Reviewet PGL-37, rapport om feito			
man 27.03	6.5	Forbedring av feito-torres, ble veldig bra!			
tir 28.03	7	Rapport skriving, Scrum møter, sprint start.			
ons 29.03	7	Rapport, Feito-torres kapittel			
tor 30.03	-	Veiledningstimer SU-læringsassisten hele dagen			
fre 31.03	6	Rapport feito-torres, visualisering av testdata og grafer			
man 03.04	7.5	Rapportskriving under feito-torres			
tir 04.04	8	Rapportskriving under feito-torres, møte med veileder, lesing av andres arbeid i sharelatex og forbedringspotensialer utpekt			

Lars M. Niebuhr

ons 05.04	5	Skrevet om versjonskontroll, merget PGL-36 med master og revidert wiki		
tor 06.04	0			
fre 07.04	3	Jobbet med: Restrukturering av sharelatex, skrevet om PGL-39		
man 10.4	4	Restrukturering av sharelatex, laget TRELLO tasks, gjort meg tanker rundt videre strukturering av rapporten		
tir 11.5	0			
ons 12.4	6	Lest igjennom alt i latex og lagt kommentertarer i trello. Skrevet rundt PGL-39 og Jira/Git		
tor 13.04	2	Rapportskriving PGL-39		
fre 14.04	0			
lør 15.04	3	Rapportskriving kvalitetssikring med scrum(planning, review, retrospective)		
søn 16.04	2	Rapportskriving kvalitetssikring med scrum(planning, review, retrospective)		
man 17.04	2	Rapportskriving installasjon og bruk av biblioteket		
tir 18.04	5	Rapportskriving, forberedning rundt vedlegg. Lagret grafer fra Jira til rapporten		
ons 19.04	7	Refaktorisert gamut.py og test_gamut.py		
tor 20.04	7	Review, Planning og retrospective møte, fikset bugs i tester(merge var skylden)		
fre 21.04	7	Rapportskriving: bibliotekets struktur, og startet med PGL-45		
man 24.4	8.5	Rapportskriving, ordnet opp i struktur, begynnt på omskriving av enhetstesting, møte med ivar ang PGL-45, planlegge møter med marius, review skrevet materiale fra nawar og sahand		
tir 25.4	1	Syk =,(, en time med koding.		
ons 26.4	8	review av skrevet materiale fra nawar og sahand, rapportskriving: kildekode, fordleing av arbeid, omstrukturering av avsluttningskapittel og laget sammenhengende dokument med alle PGLer		
tor 27.4	5	Møte med veileder og utvikling PGL-45		
fre 28.4	6	PGL-45, PGL-48 ferdig, mangler god test		
søn 30.04	6	Kritikk av oppgavebeskrivelsen, evarluering av gruppens arbeid		
man 01.5	6	Feriedag 3 (siste fridag 6t)		
tir 02.05	6	møte med veileder og oppdragsgiver, rapportskriving: gruppensarbeid, og gjort klart repo til merging med oppdragsgiver		
ons 03.05	3	Rapport: valg av utviklingsmodell og rapport layout		
tor 04.05	0	Avspasering overtid fra sønndag, avklart med gruppen		
fre 05.05	6	Rapportskriving: Avsluttning, PGL strukturkart og introduksjon		
man 08.05	7	møte med Marius, rapport: Avsluttning, planlegging videre rapport administratict mtp bachelorlevering og fremmlegg		
tir 09.05	6	Rapport: avsluttning, retting av PGLer, møte med Marius		
ons 10.05	6.5	Rapport: avsluttning, Oppgaver og resultat, gjort klart reo til merging, fant en bug i test!		
tor 11.05	8.5	Rapport: Oppgaver og resultat, gjennomgang av avsluttning, fikset en test i koden(repo), rettet litt i innledning		
fre 12.05	9	Rapport: gruppemøte med marius, småfiks i konklusjon, hoveddel omstrukturert, skrevet og omstrukturert "rapport og layout", lagt arbeidsplan for i morgen		
lør 13.05	8	rettet skrivefeil, gjort små endringer i struktur, skrevet om kapittel om scrum(9.2.1), nye navn på alle PGLer, endret mye i innledningen		
søn 14.05	6.5	Mye variert rapportskriving og retting		
man 15.05	8	Finpuss rapport, alle kapittler, levering av kode i fronter, møte med Marius		

Sahand Lahafdoozian

Dato	Timer	Beskrivelse	Sum timer:	506.5
9.01	5	Oppstartsdag.		
10.01	4	Lesing av jobbing på colourspace/ICC3D.		
11.01	6	Forerdelse av møte med Ivar og Marius.		
12.01	5	Jobbing av forprosjektrapport.		
15.01	5	Jobbet med bakgrunn av colourlab og python biblioteket.		
16.01	6	Jobbing med bakgrunn og avgrensning. Legge til punkter i JIRA og sharelatex.		
17.01	7	Finskriving av bakgrunn og avgrensning, møte og jobbet med dokumentasjon.		
18.01	6	Ferdig jobbet med kvalitetssikring (dokumentasjon del).		
19.01	4	Jobbet med Organisasjonskart og lesing av tidliger rapporter.		
20.01	7	Skriving av extreme programmering og prosjekt gjennomgang.		
23.01	6	Lesing og skriving om pep8 python.		
24.01	7	Lesing vidre pep8, verktøy for pep8, driftsbesøk på skolen, møte.		
25.01	6	Laget powerpoint for pep8, skriving om Opprette gamut-modul.		
26.01	6	Jobbet med installasjon, lesing av scrum, møte med ivar.		
27.01	7	Møte, Skriving om pair programming.		
30.02	7	Holdt presentasjon av Pep8 og parprog, Visualisering av kjerneopperasjoner.		
31.02	6.5	Møte, Ferdig med arbeidsoppgaven Opprette gamut-modul (Parrprog).		
1.02	5	Litt endring i PGL-32 og lest på forskjelig metoder.		
2.02	5	PGL-34 Research av metode av plt.plot 3d surface, feilsøking av import i python.		
3.02	5.5	Videre jobbing med PGL-34 og testing av matplotlib funksjoner.		
6.02	6	PGL-34: laget get_vertecis og test.		
7.02	7.5	Møte, PGL-34: rydding og feilretting i test-filen, Laget get_surface.		
8.02	8	PGL-34: Endring i get_surface og oppretting test-fil til get_surface. Snakket med ivar om endring i get_surface funksjon.		
09-10.02	----------	Bort på grunn av kamp.		
12.02	2	Se over kode som ble gjort av Nawar på plot_surface funksjon.		
13.02	7	PGL-34: ferdig laget eget plot_surface og testing passerer, retting på masse merge feil i git.		
14.02	7	Møte, omskrivning til pep8 kode etter ha sett over kode.		
15.02	6	Opprettet ny pakke for testing og test moduler, laget user manual for get_vertices og plot_surface.		

Sahand Lahafdoozian

16.02	**2**	**Kommentering og begynt å se på PGL-40.**		
17.02	**6**	**Research og jobbet med PGL-40.**		
20.02	**7**	**Retting av feil på funksjonen plot_surface, Vidre researching av algoritme for PGL-40.**		
21.02	**6**	**Jobbet med PGL-40.**		
22.02	**7**	**Vidre jobbing med PGL-40 og gjort ferdig find_plane() funksjon (konvertering fra C# til Python).**		
23.02	**6**	**Lage test funksjon for find_plane og skrive dokumentasjon på Kommentering av kode / kildekode og Versionshåndtering.**		
24.02	**6**	**Rapport skriving.**		
27.02	**6**	**Jobbet med PGL-40, utregning av kryssningen i punktet.**		
28.02	**7**	**Jobbet med PGL-40, møte.**		
1.03	**7**	**PGL-40 og om koding + python research.**		
2.03	**6**	**Fridag pga. kickboxing.**		
3.03	**6**	**Fridag pga. kickboxing.**		
6.03	**6**	**Jobbet med PGL-40, testing av funksjoner og utskrifter.**		
7.03	**7**	**Jobbet med PGL-40, ferdig jobbet funksjonen plane_coordinents.**		
8.03	**6**	**Jobbet vidre jobbet med feil i PGL-40.**		
9.03	**6**	**Jobbet med å retting av feil og test funksjoner på PGL-40.**		
10.03	**7**	**Ferdig med PGL-40 og Review PGL-36, Skriving av manual for PGL-40.**		
13.03	**6**	**Jobbet med Readme filen, gjort ferdig PGL-34 sin akse, gått over kode og rettet.**		
14.03	**7**	**Retting av merging med master, jobbet med rapport skriving (akse). Møte, retro.**		
15.03	**6**	**Retting av feil som ivar sa vi må rette på i PGL-40 og PGL-34.**		
16.03	**6**	**Jobbet med PGL-37.**		
17.03	**3**	**Skriving diskusjoner i sharelatex.**		
20.03	**6**	**Jobbet med PGL-37.**		
21.03	**2**	**Testing i PGL-37.**		
22.03	**6**	**Jobbet med Rapport skriving, møte.**		
23.03	**7**	**Gjort ferdig funksjonen clip_nearest i PGL-37.**		
24.03	**7**	**Rapport skriving.**		
27.03	**6**	**Jobbet med PGL-37 etter Review.**		

Sahand Lahafdoozian

28.03	**6**	**Jobbing med PGL-37 og møte.**		
29.03	**6**	**Feilsøking i PGL-37 og Rapport skriving.**		
30.03	**6**	**Ferdig test PGL-37 og rapport skriving.**		
31.03	**6**	**Rapport skriving.**		
3.04	**7**	**PGL-37 retting, jobbet videre med Wiki(overview) under methods og examples.**		
4.04	**6**	**Jobbet med Wikien og skrev om parprogrammering.**		
5.04	**----------**	**Lesing til eksamen i Applikasjonsutvikling.**		
6.04	**----------**	**Lesing til eksamen i Applikasjonsutvikling.**		
7.04	**----------**	**Lesing til eksamen i Applikasjonsutvikling.**		
10.04	**----------**	**Påskeferie.**		
11.04	**----------**	**Påskeferie.**		
12.04	**----------**	**Påskeferie.**		
13.04	**----------**	**Påskeferie.**		
14.04	**----------**	**Påskeferie.**		
17.04	**----------**	**Påskeferie.**		
18.04	**----------**	**Eksamen.**		
19.04	**----------**	**Eksamen.**		
20.04	**6**	**Review møte, lest litt på minDE og retting av rapport skriving i PGL-34.**		
23.04	**7**	**Rapport skriving i PGL-34.**		
24.04	**6**	**Rapport skriving i PGL-40.**		
25.04	**6**	**Rapport skriving i PGL-40.**		
26.04	**9**	**Retting i PGL-34 og jobbet med PGL-61 minDE. Videre skriving i PGL-40.**		
27.04	**8**	**Møte, snakking rundt PGL-er, jobbing med PGL-61. retting i PGL-er.**		
28.04	**6**	**Fridag 3.**		
1.05	**6**	**Rapport endring i PGL-40.**		
2.05	**6**	**Ferdig med PGL-40 og lest på parprogrammering.**		
3.05	**6**	**Rapport pgl-include og satt opp matematikk.**		
4.05	**6**	**Review av PGL-39, Bibliotekets struktur, pep8 og kildekode.**		
5.05	**7**	**Møte, Review av oppgaver i trello, latex matematikk fix.**		

Sahand Lahafdoozian

6.05	**6**	**Review av PGL-45, Scrum, bruken av git og jira.**		
7.05	**5**	**Review Enhettesting, retting av matematikken og dvs.**		
8.05	**6**	**Rapport skriving 3.4 colour biblioteket.**		
9.05	**6**	**Includ 37 og 61, rapport skriving om PGL arbeidsoppgave.**		
10.05	**4**	**Includ og Review av 45 og test_modul**		
11.05	**9**	**Rapport skriving div.**		
12.05	**7**	**Møte, rapport skriving div.**		
13.05	**8**	**Rapport skriving div.**		
14.05	**6**	**Rapport skriving/retting**		
15.05	**8**	**Finpusser rapport.**		
16.05				

Nawar M. Behenam

Dato	Timer	Beskrivelse	Sum timer:	490.5
16.1	6	Satt meg inn i prosjektet og skrevet om rammer		
17.1	6	Risikoanalyse og beskrivelse og tidsrammer		
18.1	7	Ferdig stillit riskoanalyse og rettskriving		
19.1	8	Nesten ferdig gjort gant skjema og løst posisjon problemer for riksio tabell		
20.1	6	Gant og gjennomgang av rapport		
23.1	5.5	Gant skjema og testing + lesing		
24.01	7	Møte og gant endringer og ICC 3d lesing		
25.01	4	PyUnit testing lesing		
26.01	6	Lesing og finpussing		
27.01	8	Kilder og rettskriving og generelt avsluttende jobb		
28.01	1	Kilder rydding		
30.01	6	Retrospect meeting og research før start av andre sprint		
31.01	6	Jobbet med PGL-32 og ferdig stillit den		
1.02	4	Endring i PGL-32 og lest opp på PGL-34		
2.02	5	Lest dokumentasjon om matplotlib, feilsøkt python error i pychar og gått gjennom ekisternde funskjoner i space klassen		
3.02	5	Jobbet videre med PGL 34 og testing for matplotlib		
6.02	6	Laget funksjoner for pgl 34 og test mot get_verticies		
7.02	8	Gjort endring i get_vertices og get_surface og omstrukturert unittestene og feilsøkt testene.		
8.02	8	Ferdig stillt get_vertices og testen i tillegg til å jobbet videre med testing og ferdig stilling av get_surface.		
9.02	6	lest meg opp på plot funksjon og prøvde å utnytte eksempel fra lvar.		
10.02	6	Prøvd å løse lokale python problemer og testing knyttet til plot_surface		
13.02	6	Ferdig stillit plot_surface og testen som tilhører.		
14.02	7	Retrospect meeting + planning + start av ny sprint		
15.02	6	Opprettet ny pakke for testing og test moduler, laget user manual for get_vertices og plot_surface.		
16.02	3	Kommentering og begynt å se på PGL-40		
17.02	6	Research og jobbet med PGL-40		
2017.02.20	6	Fikset feil med plot funksjon fra PGl-34 og gjort algoritme research for PGL-40		
2017.02.21	7	Videre jobbing med PGL-40		
2017.02.22	7	Videre jobbing med PGL-40 og converting av algoritmer		
2017.02.23	6	Jobbet med rapport skriving og laget test for PGL-40 find_plane funksjon		
2017.02.24	6	Jobbet med rapport skriving		
2017.02.27	6	Jobbet med algoritmer for PGL-40 og research for divere løsninger til den		
2017.02.28	7	Retropect meeting + planning + start av ny sprint		
2017.03.01	6	PGL-40 og om koding + python research		
2017.03.02	6	Gjennomgang av struktur og reseach		

Nawar M. Behenam

2017.03.03	**6**	**Dokumentasjon**		
2017.03.06	**6**	**Jobbet videre med PGL 40 og forståelse av mattematikken**		
2017.03.07	**7**	**Avsluttning fasen for PGL 40 alle funskjoner er lagret, mangeler testing**		
2017.03.08	**6**	**Jobbet videre med PGL-40**		
2017.03.09	**5**	**Omskrevet neareast point regningen og feilsøkt diverse error i koden**		
2017.03.10	**7**	**Ferdig stillit PGL-40 og review PGL-36**		
2017.03.13	**6**	**Etterreview av PGL-40 og research for å bruke maple for 3d modeler**		
2017.03.14	**6**	**Rapportskriving og 3d modul**		
2017.03.15	**6**	**Retting av feil som ivar sa vi må rette på i PGL-40 og PGL-34.**		
2017.03.16	**6**	**Jobbet med PGL-37.**		
2017.03.17	**6**	**Rapportskring**		
2017.03.20	**6**	**Jobbet videre med PGL-37**		
2017.03.21	**2**	**Testing PGL-37**		
2017.03.22	**6**	**Møte og rapportskriving**		
2017.03.23	**6**	**Ferdig stilling av PGL-37**		
2017.03.24	**6**	**Rapport skriving om PGL-37**		
2017.03.27	**6**	**Jobbet med PGL-37 og endret kode etter review**		
2017.03.28	**6**	**Møter + feilsøking av PGL-37**		
2017.03.29	**6**	**PGL-37 feilsøking**		
2017.03.30	**6**	**PGL-37 feilsøking og rapport skriving**		
2017.03.31	**4**	**Rapportskriving**		
2017.04.02	**3**	**Rapportskriving**		
2017.04.03	**6**	**Rapport skriving og retting av PGL-37 etter review**		
2017.04.04	**7**	**Rapport**		
2017.04.05	**6**	**rapport**		
2017.04.06	**6**	**rapport**		
2017.04.20	**6**	**Rapport**		
2017.04.21	**4**	**Rapport**		
2017.04.22	**2**	**Rapport**		
2017.04.24	**6**	**Rapport**		
2017.04.25	**6**	**Rapport**		
2017.04.26	**6**	**PGL-61 og rapport**		
2017.04.27	**6**	**PGL-61 og rapport**		
2017.04.28	**6**	**PGL-61 og rapport**		
2017.04.30	**4**	**Rapport**		
2017.05.01	**6**	**Rapportskriving**		

Nawar M. Behenam

2017.05.02	**6**	**Rapportskriving**		
2017.05.03	**6**	**Rapportskriving**		
2017.05.04	**6**	**Rapportskriving**		
2017.05.05	**6**	**Rapportskriving**		
2017.05.08	**6**	**Rapportskriving**		
2017.05.09	**6**	**Rapportskriving**		
2017.05.10	**6**	**Rapportskriving**		
2017.05.11	**9**	**Rapportskriving**		
2017.05.12	**7**	**Rapportskriving**		
2017.05.13	**8**	**Rapportskriving**		
2017.05.14	**6**	**Rapportskriving**		
2017.05.15	**8**	**Finpussing av rapport**		