NTNU
Det skapende universitet

Kollisjonsdeteksjon og -respons ved animasjon av tekstiler og klær

Janne Beate Lervik Bakeng

Master i datateknikk
Oppgaven levert: Mai 2007
Hovedveileder: Torbjørn Hallgren, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Oppgavetekst

Oppgaven består i å studere metodene som brukes for å detektere kollisjoner ved animasjon av tekstiler og klær og hvordan det gies respons når kollisjon er fastslått.

Oppgaven gitt: 03. oktober 2006
Hovedveileder: Torbjørn Hallgren, IDI

Sammendrag

Denne rapporten omhandler teknikker og metoder som kan benyttes ved kollisjonsdeteksjon og -respons i animering av tekstiler og klær. Det blir lagt vekt på å se trenden i feltet og hvilke teknikker som er reperesentativ i forhold til den.

Feltet preges av at det finnes svært mange områder som krever spesialdesignede løsninger. Dette fører til at de fleste teknikker og metoder som presenteres gir svært spesialiserte løsninger. Innkapslede volum hierarkier(IVH), distansefelt og teknikker for feiloppretting er en del metoder som er mye brukt. Teknikkene utelukker ikke hverandre og en kombinasjon av IVH og feiloppretting virker som en svært effektiv løsning for forskjellige typer systemer.

Forord

Denne rapporten er et resultat av en masterfagsoppgave ved Instituttet for Datateknikk og Informasjonsvitenskap, Norges Tekniske og Naturvitenskaplige Universitet høsten 2006.

Målet med oppgaven har vært å gjennomføre et litteraturstudie for å kartlegge ulike metoder som brukes for å detektere kollisjoner ved animasjon av tekstiler og klær og hvordan den gies respons når kollisjonen er fastslått.

Trondheim, 4. mai 2007

Janne Beate Lervik Bakeng

Innhold

Figurer

Tabeller

Kapittel 1

Introduksjon

Dette kapittelet beskriver motivasjonen for dette arbeidet og fastsetter hvilke mål som skal nåes med rapporten. Rapportens struktur forklares og innholdet i hvert etterfølgende kapittel forklares kort. Til slutt blir dette kapittelets innhold kort oppsummert.

1.1 Motivasjon

Modellering av klær og tekstiler har lenge vært, og er, et interessant et tema innenfor grafikk. Kollisjonsdeteksjon og -respons er en flaskehals i modellering, dette på grunn av mengden mulige kollisjoner tekstiler og klær kan ha med andre objekter i scenen, og ikke minst med seg selv[7]. I følge Brazel et al [5] kan ikke mennesker skille mellom fysisk-korrekt og fysisk-mulig oppførsel. Dette åpner for muligheter til å animere klær og tekstiler på mange andre måter enn kun fysisk korrekte metoder.

Fordi animasjon av tekstiler og klær brukes i mange forskjellige typer applikasjoner vil metodene som finnes gjenspeile dette mangfoldet. I lys av dette vil det ikke være riktig å måle de forskjellige metodene opp mot hverandre. En god metode for en interaktiv applikasjon vil kanskje ikke være bra nok for modellering av klær på et menneske i en animasjonsfilm. Dette er fordi den interaktive applikasjonen verdsetter rask rendring, mens for klær i en animasjonsfilm er det visuelle resultatet mye viktigere.

1.2 Mål

Ettersom tekstiler og klær har vært og fortsatt er et viktig tema innenfor grafikk er det naturlig at det produseres er stort antall artikler. Når informasjonsmengden blir i denne størrelsesorden kan det være vanskelig å skaffe seg oversikt over materialet som finnes.

Målet med dette litteraturstudiet er å gi en oversikt over metoder kan brukes ved kollisjonsdeteksjon og respons i animering av tekstiler og klær innen datagrafikk. Det er ikke hensikten å forsøke å skape en fullstendig oversikt over alle eksisterende metoder. Hensikten er å presentere og vurdere en rekke metoder som representerer trendene slik de er i dag og har vært en tid tilbake.

1.3 Struktur

Rapporten er strukturert som følger. Kapittel 2 gir en kort innføring i bakgrunnskunnskap som er relevant for emnet.

Hovedtyngden i rapporten er i kapittel 3. Her presenteres de forskjellige metodene som er blitt vurdert som relevant for kollisjonsdeteksjon og respons i animasjon av tekstiler og klær i dette studiet.

I kapittel 4 konkluderes og oppsummeres det gjennomgåtte stoffet fra kapittel 3.

1.4 Oppsummering

Dette kapittelet forklarer motivasjonen bak det valgte temaet og hva som er målet med arbeidet med dette litteraturstudiet. Rapportens struktur presenteres deretter for å gi et innblikk i innholdet av rapporten.

Kapittel 2

Bakgrunn

Dette kapittelet gir en introduksjon til temaet rundt kollisjonsdeteksjon og respons i tekstiler og klær. Først skal vi se litt på fakta om tekstiler og klær generelt. Deretter vil viktige faktorer ved modellering av deformerende objekter bli gjennomgått og til slutt presenteres noen mulige bruksområder og de krav som stilles i sammenheng med disse.

Kapitlet er delt inn i underkapitler, kapittel 2.1 tar for seg hvorfor tekstiler er unike, hvordan dets egenskaper kan overføres til en modell, hvordan modellen kan konstrueres og hva som er viktig når det gjelder kollisjonsdeteksjon. Kapittel 2.2 oppsummerer stoffet som er gjennomgått i kapittelet.

2.1 Tekstiler og klær

Vi omgir oss med tekstiler og klær hver dag og i mange forskjellige situasjoner. Sengetøyet vi sover i, duken på bordet, klærne på kroppen, gardinene i vinduet og mye mer er alle eksempler på tekstiler vi omgir oss med i det daglige. For å modellere noe vi kjenner så godt stilles det høye krav til modellen. Grunnen til at det stilles strenge krav er at de minste avvik i det modellerte tekstilets oppførsel raskt blir fanget opp av øyet og avvist som unaturlig.

2.1.1 Tekstiler er unike

Tekstiler har en rekke egenskaper som for eksempel, bøyelighet, strekk i både bredde- og lengderetning, evne til kompresjon, vekt, tykkelse og

Figur 2.1: Et stykke skrukkete tekstil [9]

.

varmeledningsevne er noen av dem. Dette er egenskaper som andre materialer også har.

Det som gjør tekstiler så unike er evnen deres til å motstå strekk samtidig som de lett lar seg bøye. En annen karakteristikk er måten de bøyer seg eller skrukker. Hvis disse egenskapene ikke blir tatt hensyn til vil ikke animasjon av tekstiler se naturlig ut [9].

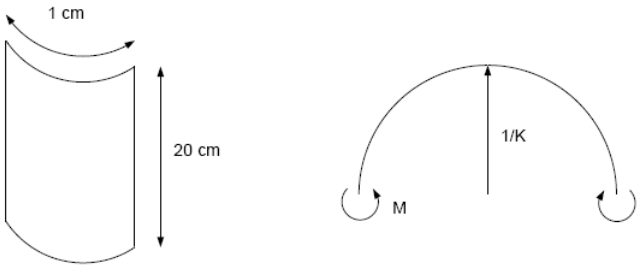
For at, for eksempel, tekstilindustrien skal kunne utvikle grafiske systemer hvor de kan teste ut ulike typer tekstiler, er det viktig at de kan rekonstruere et tekstil så realistisk som mulig. Ved å avgjøre hvilke egenskaper som gjør tekstiler unike kan disse implementeres som parametere i den grafiske applikasjonen.

2.1.2 Kawabata systemet

Kawabata Evaluerings System (KES) er et standardisert sett med tester utviklet av "The Textile Machinery Society of Japan"[1]. Systemet måler viktige egenskaper som bøyning, strekking, forskyvning, kompresjon og overflatestruktur [6].

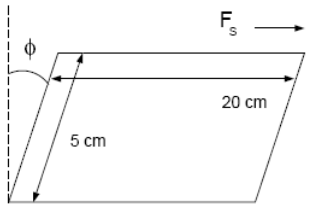
Bøyning, strekking og forskyvning måles gjennom å måle hvilke krefter og momentum som skal til for å deformere tekstilet, mens kompresjon blir målt etter hvor fort tekstilet finner tilbake sin opprinnelige form etter at den ytre påvirkningen forsvinner. Overflatefriksjon, tykkelse og vekt er andre egenskaper systemet finner verdier for.

[1] http://www.soc.nii.ac.jp/tmsj/japan/index-e.html



Figur 2.2: Diagram som viser bøyningstesten [6]

.



Figur 2.3: Diagram som viser forsyvningstesten [6]

.

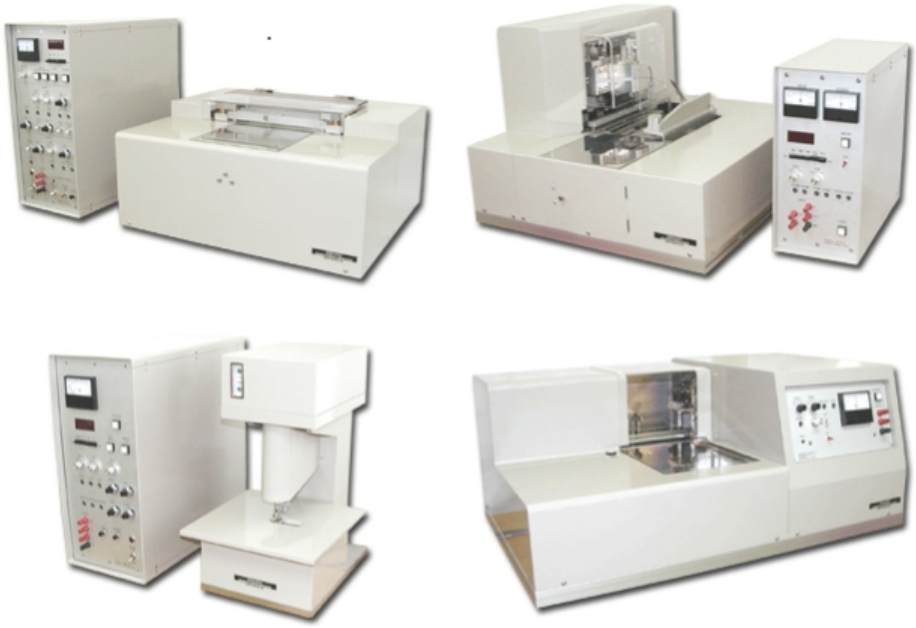
Bøyningstesten foregår ved at et 20 cm $\times$ 1 cm stort stykke tekstil festes langs to motstående kanter, tekstilet bøyes og kreftene som brukes måles. Forsyningstesten forgår på lignende måte, et stykke tekstil, 20 cm $\times$ 5 cm, påføres en kraft langs den ene kanten mens den motstående kanten holdes på plass, kreftene måles underveis. Figur 2.2 og 2.3 illustrerer disse testene.

Gjennom tre steg kan data fra Kawabata systemet gjøres om til funksjoner som kan brukes i en simulering. Steg én er å finne tilnærmede funksjoner som lager kurver som følger verdiene fra Kawabata testene. Steg to er å kombinere disse funksjonene med modellen av tekstilet. Det finnes en rekke ulike modeller som kan brukes for å modellere tekstiler som underkapittel 2.1.3 vil vise. Til slutt skaleres resultatene slik at systemet opererer med standard fysiske verdier.

KAWABATA laboratory i Tsjekkia er et eksempel på noen som i praksis bruker disse testene, de benytter seg av fire forskjellige maskiner, se figur 2.4, for å måle egenskaper hos tekstiler [20].

2.1.3 Modellering av tekstiler og klær

Innen området som omhandler simulering og modellering av deformerende objekter, som for eksempel tekstiler og klær, går det igjen en del begreper som det er nødvendig å ha en forståelse av.



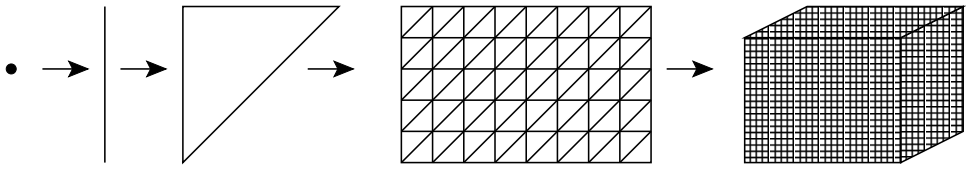
Figur 2.4: De fire maskinene som brukes av KAWABATA laboratory. (øverst til venstre) Automatic Tensile & Shear Tester-KES-FB1 (øverst til høyre) Automatic Surface Tester - KES-FB4 (nederst til venstre) Automatic Compression Tester - KES-FB3 (nederst til høyre) Automatic Pure Bending Tester - KES-FB2 [20]

Et polygon er en plan figur spesifisert av tre eller flere koordinatposisjoner, kalt noder, som er koblet sammen av rette linjesegmenter, kalt kantene til polygonet [18]. Dersom flere polygoner kobles sammen side om side danner de et nettverk, også kalt nett (eng. mesh) [12].

Det å modellere er definert som dannelsen og manipuleringen av en systemrepresentasjon, en modell er derfor en representasjon av et virkelig system [18]. Dersom kun overflaten modelleres kalles det en overflatemodell. Slike modeller brukes ofte for å modellere tekstiler hvor tykkelsen er neglisjerbar. I andre tilfeller vil en volummodell være mer korrekt å bruke fordi indre forhold i objektet kan påvirke og bli påvirket av krefter som påvirker modellen.[1]

Figur 2.5 illusterer sammenhengen mellom de grunnleggende begrepene node, kant, polygon, nettverk og modell.

Videre kan en modell være diskret, kontinuerlig eller en kombinasjon. Forskjellen er over hvilket domene modellen er definert. En diskret modell er definert for et endelig sett av punkter, mens den kontinuerlige modellen



Figur 2.5: (A) En node og (B) en kant. (C) Et polygon. (D) Et nettverk av polygoner. (E) En modell.

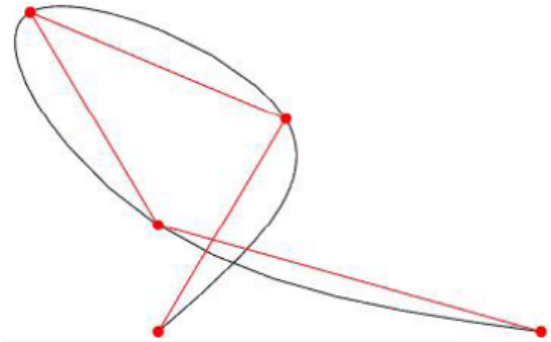
er definert for alle verdier. En kombinasjon kan være at modellene er diskret definert visse steder, mens kontinuerlig definert andre steder.

Geometriske modeller representerer kun formen til det objektet de skal modellere. Det vil si den beskriver formen til en fysisk eller matematisk modell med geometriske konsepter, som for eksempel punkter, linjer, kvadrater eller sirkler. Fysiske modeller derimot beskriver i tillegg de fysiske lovene som gjelder [30]. Eksempler på fysiske lover som kan gjelde for en fysisk modell er eksterne og interne krefter, som gravitasjon, luftmotstand, demping, friksjon, kollisjonsresponskrefter og lignende.

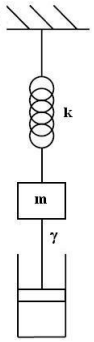
Dersom beregningseffektivitet er det viktigste kriteriet kan geometriske modeller være løsningen. Designeren har mye kontroll over modellen, men simuleringen kan by på problemer dersom krefter skal virke på modellen [14]. Et eksempel på en modelleringsteknikk som lager geometriske modeller er splines, se figur 2.6 for et eksempel på en kubisk spline.

Det finnes flere forskjellige typer splines, men alle har til felles at de spesifiserer et sett koordinatpunkter, kalt kontrollpunkter. Polynomer tilpasses disse kontrollpunktene ved approksimasjon eller interpolasjon. De forskjellige typene splines oppstår ved å bruke forskjellige typer polynomer og kontinuitetsbetingelser for kontrollpunktene. De typene som er mest brukt i grafisk programmering er Bézier og B-splines, fordi de er enkle å implementere og kan tilpasses hvilket som helst antall kontrollpunkter [18]. På grunn av mangelen av fysiske lover i geometriske modeller brukes de svært sjelden.

Masse-fjær modellen er en metode som modellerer tekstiler ved hjelp av fysiske lover. Her modelleres det myke vevet ved hjelp av et diskret antall noder som er koblet sammen med fjær i et nettverk. Hver av nodene får tildelt en masse, (m) og fjærene får en stivhetsgrad (k). I noen modeller er det også aktuelt å innføre en dempningskraft (γ) for å simulere at tekstilets bevegelser dempes av forskjellige årsaker. Fjærene er ofte lineære (Hookean), men ikke-lineære fjær kan også benyttes for å modellere uelastisk oppførsel [14]. Figur 2.7 illusterer hvordan fjærene mellom massepunktene fungerer.



Figur 2.6: Et eksempel på en kubisk spline og hvordan den er bygd opp.



Figur 2.7: Illustrasjon av parametrene som påvirker kreftene som simulerer fysiske lover i en masse-fjær modell sammen.

Bruk av partikkel modeller utviklet seg fordi masse-fjær modellen ikke kunne gi den nøyaktigheten simulering av tekstiler og klær krever. Partikkel modellen gir mer nøyaktig animering av den "klebrige" og samtidig elastiske egenskapen tekstilets overflate har [38]. Deformerende objekter modelleres som et 2D nett av noder som representerer partiklene, i partikkel modeller virker kreftene kun på partiklene [11]. Slike systemer kan brukes til for eksempel å modellere tekstiler, vann og flammer.

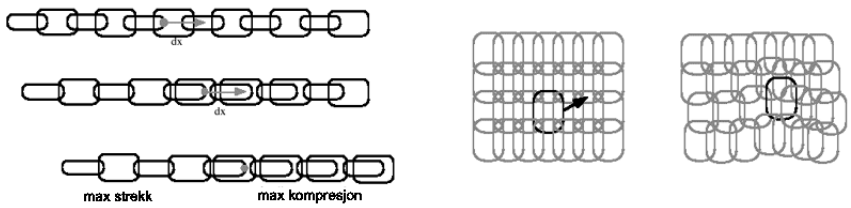
Masse-fjær teknikken gir en diskret modell med rimelige beregningskostnader. Dersom større presisjon i de fysiske lovene er ønskelig kan Endelige Elementers Metode (eng. Finit Element Method(FEM)) være et alternativ. Metoden definerer en kontinuerlig modell, dette fører til at deformasjoner vil kreve mye større prosesseringskraft ved beregninger i forhold til diskrete modeller. En av fordelene med metoden er at den overholder loven om konservering av all energi. I praksis går metoden ut på å beregne likevekt i energi, dette gjøres ved at modellen deles inn i elementer og de kontinuerlige likevektsligningene beregnes for hvert element. Flere elementer gir nøyaktigere modell og mer kompliserte beregninger [14].

Både masse-fjær modeller og FEM modeller vil i de fleste tilfeller være for kostbare for å simulere tekstiler med kollisjonshåndtering i sanntid. En teknikk som gir mulighet til å simulere deformerende objekter i sanntid, både overflate- og volummodeller, ble foreslått av Gibson i 1997. Metoden går ut på å behandle modellen som en kobling av kjeder slik figur 2.8 illusterer. Hvert ledd i kjeden har en max strekk og kompresjonsverdi i forhold til sine naboledd, det er disse verdiene som i forhold til tekstiler vil kontrollere strekk, forskyvning, bøyning og kompresjon [15].

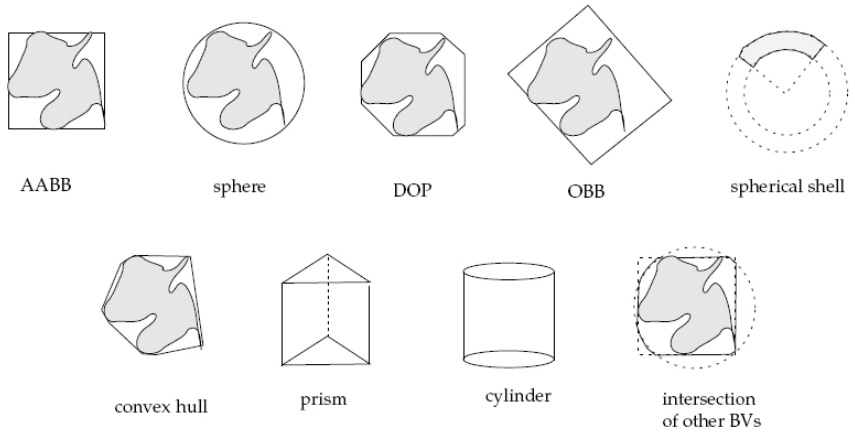
Ofte gjennomføreres det et preprosesseringssteg etter at modelleringen er ferdig og før simuleringen begynner. Forskjellige datastrukturer kan konstrueres for å effektivisere blant annet kollisjonshåndtering. Et eksempel på en slik struktur er innkapslende volum hierarki (IVH) (eng. bounding volum hierarki). IVH defineres som følger: I IVH skal hver node i treet være assosiert med et subset av objektets primitiver, hvert subset er innkapslet i en spesifikk form. Et design valg er hvilken form som skal brukes. Figur 2.9 viser et utvalg muligheter av former som kan brukes. Det bemerkes at OOBs of k-DOPs er av spesiell interesse for deformerende objekter på grunn av de kan konstrueres og oppdateres effektivt [33].

En annen datastruktur som kan beregnes i et preprosesseringsteg er distansefelt. Et distansefelt $D : \mathbb{R}^3 \longrightarrow \mathbb{R}$ defineres som null nivå settet, det vil si $S - \{p|D(p) - 0\}$. D er distansefunksjonern, p er et punkt, $\mathbb{R}$ er domenet og S er settet. I praksis fungerer feltet som en "innbruddsalarm" som varsler dersom punkter beveger seg for nærme et objekt, i tilfelle alarmen går vil kollisjonsrespons bli iverksatt [33].

Tekstiler og klær modelleres ofte som deformerende objekter [7][29][35] og simulering av tekstiler uten begrensninger anses som et løst problem [3]. Det finnes få systemer som ikke har begrensninger i simuleringen av tekstiler i dag, kollisjonsrespons, eksterne og interne krefter, som gravitasjon, luftmotstand, demping og lignenede, og interaktive brukere er noen eksempler på parametere som begrenser tekstilet.



Figur 2.8: "Chainmail" modellen består av kjeder koblet sammen i et nettverk.[15]



Figur 2.9: Eksempler på forskjellige typer innkapslede volum.[33]

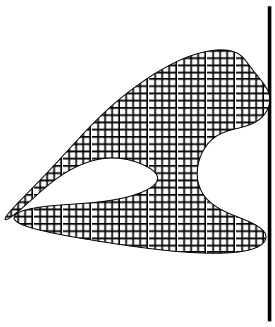
Tekstiler kan kollidere med både seg selv og andre objekter i scenen, samtidig som de stadig er under påvirkning av både interne og eksterne krefter, som for eksempel gravitasjon, luftmotstand og lignende. Dette er faktorer som må beregnes kontinuerlig under simuleringen.

2.1.4 Kollisjonsdeteksjon

Kollisjoner anses som en flaskehals i simulering av tekstiler og klær. I en scene har, i teorien, alle overflatepunkter muligheten til å kollidere med hverandre. Dersom det finnes n slike punkter i en scene, må n^2 kollisjonsdeteksjoner utføres for hvert tidssteg. Med tanke på at en scene kan inneholde titusenvis av slike punkter vil denne fremgangsmåten fort bli alt for beregningskostbar [7].

Håndtering av kollisjoner ble først studert for harde objekter, men fordi deformerende objekter har andre egenskaper enn harde objekter måtte det utvikles egne algoritmer til for kollisjonshåndtering av tekstiler og klær [29].

I en scene med både harde og deformerende objekter kan forskjellig typer kollisjoner oppstå. Harde objekter kan kollidere med enten harde eller deformerende objekter og motsatt. Det som er spesielt for deformerende objekter er at de i tillegg til disse kollisjonene kan kollidere med seg selv, selvkollisjon, se figur 2.10. Kollisjonsdeteksjon mellom harde objekter kan la være å søke etter selvkollisjoner, og ofte holder det å finne kun ett kontaktpunkt mellom to kolliderende objekter. Deformerende objekter opplever ofte både selvkollisjon og at kollisjoner har mer enn ett kontaktpunkt [33].



Figur 2.10: Illustrasjon av et deformerende objekt som kolliderer med seg selv og et annet objekt hvor det oppstår to kontaktpunkter.

Som tidligere nevnt er det vanlig å gjennomføre et preprosesseringsteg for å redusere antallet mulige kollisjoner en algoritme trenger å søke etter. Ved å benytte romlige datastrukturer, som for eksempel innkapslende volum hierarkier, distanse felt eller lignende, kan søking etter kollisjoner aksellereres effektivt for harde objekter. Deformerende objekter kan forandre form mellom hvert tidssteg, på grunn av dette vil preprosesseringssteget måtte gjentas ofte og effektiviteten vil bli redusert [33].

En annen stor forskjell mellom harde og deformerende objekter er at deformerende objekter trenger mer informasjon i forbindelse med en kollisjon. For eksempel må et deformerende objekt ha informasjon om hvor langt det kolliderende objektet penetrerte for å kunne fastslå korrekt respons i form av deformasjon og frastøtende kraft [33].

For detaljert gjennomgang av generell kollisjonsdeteksjon anbefales [4] og [10].

2.1.5 Bruksområder for modellerte tekstiler og klær

Simulering av tekstiler og klær brukes hovedsaklig innen to kategorier, animasjon og CAD(Computer Aided Design). Animasjon har ofte som mål at resultatet skal være visuelt overbevisende, dette kan for eksempel ses i animasjonfilmer av Pixar©[3].

Fysisk korrekt oppførsel er derimot viktigst for brukere av CAD verktøy. Denne kategorien er ofte opptatt av å måle tradisjonelle mekaniske verdier og en fysisk korrekt modell er derfor svært essensielt. Verdier generert av Kawabata systemet, beskrevet i 2.1.2 på side 4, kan brukes i simulering av slike modeller.

Under disse to kategoriene finnes sanntids- og ikke-sanntidssystemer, og systemer med og uten interaktivitet. Dette arbeidet skiller ikke mellom

teknikker fra disse kategoriene, men ser istedet generelt på metoder for kollisjonshåndtering for tekstiler og klær.

2.2 Oppsummering

Dette kapitlet gir noe bakgrunnskunnskap om tekstiler og klær, hva som gjør at tekstiler er ulikt andre materialer, og hvordan dets egenskaper kan måles ved hjelp av et system kalt Kawabata systemet. Videre ser kapittelet på hvorfor modellering av deformerende objekter krever spesiell oppmerksomhet på kollisjonshåndtering i forhold til harde objekter. Til slutt gies noen eksempler på i hvilke sammenhenger slike systemer kan brukes.

Kapittel 3

Kollisjonshåndtering ved simulering av tekstiler og klær

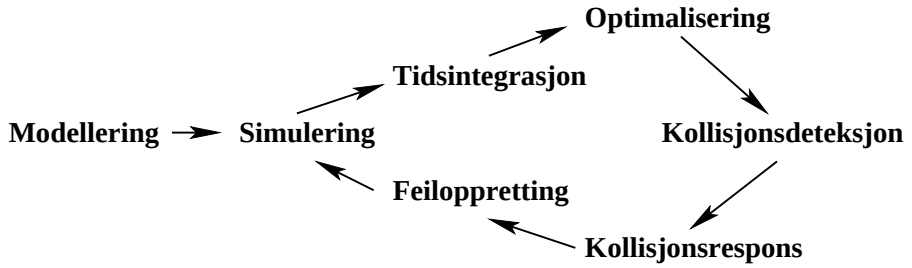
Kollisjonshåndtering ved simulering av tekstiler og klær referer til prosessen hvor kollisjoner oppdages og en respons gies, altså kollisjonsdeteksjon og -respons. Kollisjonsdeteksjon er å finne kollisjoner mellom elementer, for eksempel mellom punkter og polygoner. Kollisjonsrespons vil si de handlinger som utføres når en kollisjon er oppdaget, det vil i de fleste tilfeller være krefter som påvirker de involverete elementene, for eksempel friksjon.

Kapittelet er delt opp i underkapitler som representerer teknikker som benyttes i arbeider med kollisjonshåndtering. Merk at flere enn èn teknikk kan benyttes per løsning.

3.1 Hovedproblemer ved kollisjonshåndtering

I følge Provot[29] er det fire hovedproblemer ved kollisjonshåndtering. Først må kollisjonen oppdages, så må det benyttes teknikker for å effektivisere søking etter kollisjoner, videre må det gies en respons til kollisjonen og til slutt må det sjekkes at ingen feil oppsto underveis. Det vil også være et problem å samle inn informasjon som berører kollisjonen. Dette fordi beregning av, for eksempel penetreringsdybde, kan være dyrt og tregt for deformerende objekter samtidig som det er svært viktig for å gi mest mulig realistisk respons [33].

Selv om metodene og teknikkene for simulering av tekstiler og klær kan være svært forskjellig har de alle en felles struktur og oppbygning, hvor de gjennomgår en rekke trinn. Disse trinnene kan oppsummeres som følger:



Figur 3.1: Først skapes en modell, deretter startes simuleringen som foretar tidsintegrasjon, optimalisering, kollisjonsdeteksjon, kollisjonsrespons og feiloppretting per tidssteg.

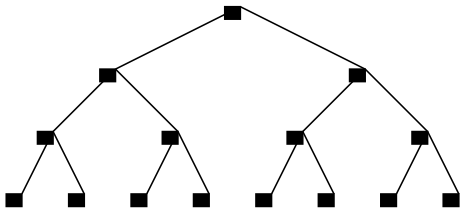
- Modellering
- Simulering og Tidsintegrasjon
- Optimalisering
- Kollisjonsdeteksjon
- Kollisjonsrespons
- Feiloppretting

Før simuleringen kan begynne må scenen bygges opp gjennom modellering, den underliggende strukturen og lover for hvilke krefter som skal gjelde innføres. Simuleringen består av flere trinn som itereres for hvert tidssteg.

Rekkefølgen på disse trinnene er ikke helt fastlåst og flere trinn kan være sammenslått. Først beregnes nye verdier for parametere som for eksempel hastighet og posisjoner for å skap et inntrykk av at objekter beveger seg og blir deformert. Deretter oppdateres eller rekonstrueres datastrukturer som skal benyttes til å optimalisere effektiviteten av simuleringen. Disse strukturene kan så benyttes til å søke etter kollisjoner i scenen. Når kollisjoner oppdages må det gies respons og til slutt må eventuelle feil som har oppstått rettes opp før tidsintegrasjon for neste tidssteg kan gjenomføres. Figur 3.1 illusterer hvordan disse stegene henger sammen.

3.2 Innkapslede volumhierarkier

Innkapslede volumhierarkier (eng. Bounding volume hierarchies) er trær hvor nodene er deler av scenen. Nodene inneholder en del av scenen, mer spesifikt hva nodene inneholder avhenger av typen volum hierarki, for eksempel kan det være et polygon.



Figur 3.2: Illustrasjon av en binær trestruktur.

Treet er organisert slik at noder på et nivå er et subsett av sin foreldrenode. Rotnoden er den øverste representasjonen av scenen og har derfor ikke noen foreldrenode. Løvnoder er de nederste nodene i treet, og de inneholder de minste elementene i hierarkiet, det vil si de har ingen barnenoder. Antall barnenoder en forelder har avhenger av typen innkapslede volum hierarki, for eksempel i binære trær vil hver node ha to barn og oktaltrær har åtte barn.

Figur 3.2 viser et binærtre, nodene har to barn hver. Ved å sjekke foreldrenoder mot diverse kriterier kan man eliminere eller velge å søke videre i dens barnenoder [33].

Slike hierarkier kan bygges ved å dele inn scenen[1] rekursivt i soner som oppfyller bestemte kriterier. Kriteriene for oppdelingen kan være posisjon i 2D teksturrom [29] eller noder som allerede finnes i den underliggende modellen[26].

Innkapslede volum hierarkier effektiviserer søk etter kollisjoner og kan finne kollisjoner mellom forskjellige objekter eller selvkollisjon [13][29][7]. For animasjon av klær og tekstiler er dette viktig da det ikke er uvanlig at tekstiler kolliderer med seg selv. Et viktig design valg er hvilken type volum som skal brukes. Dette valget vil avgjøre blant annet hvor effektivt treet kan søkes gjennom og hvor enkelt det er å oppdatere. Figur 2.9 på side 10 illustrerer volum som inngår i forskjellige typer IVH [33].

3.2.1 Collision detection and response for computer animation [25]

Moore og Wilhelms foreslo allerede i 1988 å bruke innkapslede volum hierarkier for å minimalisere beregningskostnadene ved kollisjonsdeteksjon [25]. De presenterer to algoritmer for kollisjonsdeteksjon hvor den ene er beregnet på deformerende objekter, slik som tekstiler og klær, og den andre er beregnet på rigide objekter.

[1] Scenen kan være et eller flere objekter avhengig av behov.

Modellering
Den underliggende modellen består av triangler formet av noder og kanter. De erkjenner at en korrekt test må teste både kanter og noder, ettersom polyhedraer kan kollidere kant mot kant uten at noen noder er innvolvert, men de velger å presentere en algoritme som kun tester noder mot triangler. Algoritmen de presenterer er delt opp i to tilfeller, et enkelt og et hardt tilfelle, ett tilfelle for triangler som ikke beveger seg og ett for trianler som har en hastighet.

Simulering og Tidsintegrasjon
Systemet simuleres ved at posisjoner, hastigheter og tilsvarende variabler i modellen oppdateres med nye verdier for hvert tidssteg, slik etterfølgende avsnitt forklarer.

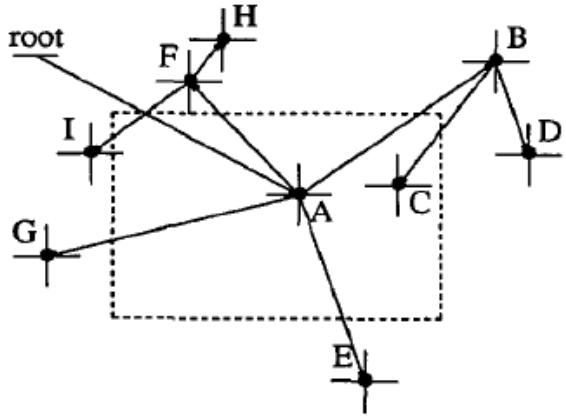
Moore og Wilhelms sier ikke noe om hvordan tidsintegrasjon utføres i situasjoner hvor kollisjoner ikke inntreffer, de antar at systemet har metoder for å håndtere dette.

Optimalisering
Ettersom det kan bli svært mange tester som må gjennomføres innføres et optimaliseringssteg. En enkel test utføres som et preprosesseringssteg for å minimere kostnadene ved å utføre kollisjonsstestene. Normalen fra punktet til planet utspent av trianglet kalkuleres for $t = 0$ og $t = 1$. Kun dersom fortegnet til denne normalen har forandret seg vil de overstående testene gjennomføres.

Moore og Wilhelm erkjenner at nok et optimaliseringssteg kan redusere antall kollisjonstester som må gjennomføres, men sier samtidig at de velger å ikke gjennomføre det.

Optimaliseringsmetoden går ut på at et volum spennes ut over trianglets start og sluttposisjon, den resulterende boksen utvides tilsvarende distansen punktet som skal testes mot trianglet beveger seg i tiden $t = [0, 1]$. Dette innebærer et firedimenjsonalt volum som dekker området trianglet beveger seg i løpet av tidssteget, og at volumet er utvidet med distansen til punktet er nødvendig for å forsikre at punktet ikke passerer uoppdaget gjennom trianglet.

Et oktaltre konstrueres ut ifra alle punktene i modellen og dette treet bygges på nytt for hvert animasjonssteg. Dette treets noder sjekkes mot det firedimensjonale volumet. Hvis et punkt i oktaltreet viser seg å være inne i volumet må alle dets barn søkes gjennom. I motsatt tilfelle kan halvparten av barnenodene elimineres. Når man finner noder som ikke kan elimineres må testene beskrevet av ligning 3.2, eller ligning 3.1 dersom trianglet ikke beveger seg, utføres. Dette steget er $O(m\log m)$ for å bygge treet og $O(n\log m)$ for å søke i det, mot $O(nm)$ uten det, der n er antall triangler og m er antall punkter. Figur 3.3 viser en 2D versjon av dette søket.



Figur 3.3: 2D illustrasjon av oktaltre søket brukt i [25]. Bokstavene representerer nivåer i hierarkiet til et objekt det skal søkes etter kollisjoner mot. Den stiplede boksen representerer et annet objekt som kolliderer, de bokstavene som er innenfor boksen må sjekkes for kollisjon.

Den stiplede boksen i figur 3.3 er en 2D versjon av volumet som spennes ut over trianglets start og sluttposisjon og bokstavene A-I representerer nodene som inneholder punktene i modellen. Denne versjonen tilsvarer et kvadtre, men illustrerer hvordan et oktaltre søk vil gjennomføres.

Et slikt optimaliseringssteg vil også kunne eliminere faren for at punkter kommer seg uoppdaget gjennom et triangel fordi det har for stor fart.

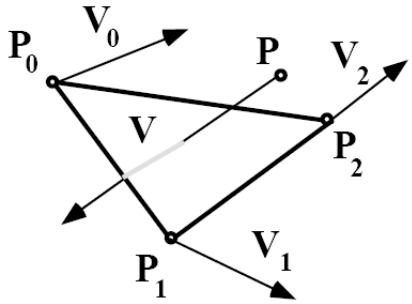
Begrunnelsen for å utelukke dette steget er at det er sjelden at punkter passerer uoppdaget. Et argument som i mange tilfeller ikke holder for dagens applikasjoner som jobber med tekstiler og klær der presisjon er avgjørende.

Kollisjonsdeteksjon
For å teste hvilke punkter som kolliderer med triangler uføres kollisjonstester. Når trianglet ikke er i bevegelse representeres det ved vektorligningen:

$$P + (P' - P)t = P_0 + (P_1 -_0)u + (P_2 - P_0)v \tag{3.1}$$

hvor P og P' er start og sluttposisjon for det punktet som er i bevegelse og P_i definerer trianglet. t er 0 ved starten av simulasjonssteget og 1 ved slutten. Ved å løse ligningen for parameterene u og v kan kollisjoner oppdages, dersom $0 \leq t \leq 1$ samtidig som $u \geq 0$ og $v \geq 0$ konkluderes det med at punktet har kollidert med trianglet i gjeldene tidssteg.

I det harde tilfellet må farten til trianglet inkluderes i ligningen og den blir



Figur 3.4: Trianglet består av tre noder i hvert hjørne, kanter kobler nodene sammen. Hver node har en hastighet og posisjon. Punktet blir testet om det beveger seg fra en side av trianglet til den andre. Illustrasjon av sammenhengen mellom variablene i ligning 3.2 [24].

som følger:

$$\begin{aligned} P + Vt &= P_0 + V_0 t \\ +((P_1 - P_0) + (V_1 - V_0)t)u &+ ((P_2 - P_0) + (V_2 - V_0)t)v \end{aligned} \tag{3.2}$$

der V_i er farten til punkt P_i i trianglet. Figur 3.4 illustrerer hva variablene refererer, trianglet er et element i nettverket som danner overflaten.

Ved å rearrangere ligningen og løse det lineære systemet for t kommer de frem til en 5. grads ligning. Denne løses ved hjelp av et binært søk fordi det er garantert å konvergere.

Kollisjonsrespons
Når kollisjonen er oppdaget vil en dynamisk animasjon ha behov for at en respons beregnes av systemet. Moore og Wilhelms foreslår to algoritmer for å løse dette. Ved beregning av respons er det viktig at lineært og angulært momentum bevares for at responsen skal være overbevisende for brukeren.

Responsalgoritmen de presenterer går ut på det følgende. I kollisjonspunktet plasseres en fjær som virker på begge de kolliderenede objektene med like stor kraft, men i motsatt retning. Ved å bruke lover for fjær, som for eksempel K/d, der d er avstanden mellom objektene og K er fjærkonstanten som avgjør hvor stiv fjæren er, kan kraften som virker på objektene avta med at avstanden mellom objektene øker.

Kollisjonen kan gjøres elastisk eller uelastisk ved å variere konstanten ϵ, der ϵ=1 er en perfekt uelastisk kollisjon og ϵ=0 er en perfekt elastisk kollisjon der ingen kinetisk energi går tapt. Denne konstanten kombineres med fjærkonstanten K på følgende måte $K_{forlatende} = \epsilon K_{motende}$.

For å kunne benytte denne metoden må den underliggende modellen kunne håndtere eksterne krefter. Problemet med metoden er at den er svært beregningskostbar. Harde kollisjoner innebærer bruk av stive fjær, som fører til stive ligninger som igjen krever små tidssteg for å løses numerisk korrekt. Dette betyr at en animasjon som går glatt kan begynne å bremse opp dersom det oppstår en voldsom kollisjon. Dette er ikke akseptabelt for sanntidssystemer og metoden brukes sjelden på grunn av sine begrensninger.

Responsbehandling ved bruk av fjær kan brukes både på rigide og deformerende objekter, den neste metode de presenterer er analytisk og konserverer momentum for de kolliderende objektene. Denne metoden derimot har de ikke tilpasset deformerende objekter.

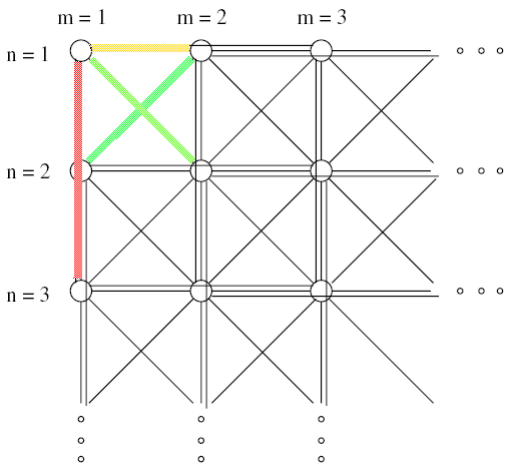
Diskusjon Det er svakheter med responsmetodene presentert av Moore og Wilhelms i forhold til tekstiler og klær. Antagelsen om at det er bare ett kollisjonspunkt er ikke sannsynlig fordi tekstiler ofte kolliderer både med seg selv og med objekter i omgivelsene. Videre sier ikke metodene noe om hvordan objektene skal deformeres, for tekstiler er dette et viktig punkt ettersom måten tekstiler oppfører seg kjennetegnes ved hvordan de deformeres.

Fordelene med metoden som presenteres her er at animatøren ikke behøver å visuelt inspisere hver eneste kollisjon for deretter å bestemme hvilken respons som skal gies, det vil si at denne metoden gjør at simuleringen kan opptre dynamisk. Historisk sett var dette en stor fremgang for grafisk animering.

3.2.2 Collision and self-collision handling in cloth model dedicated to design garments[29]

Tidligere har Provot[28] foreslått en algoritme for å modellere tekstiler og klær ved hjelp av masse-fjær modellering. Med dette arbeidet utvikler han et systemet som skal fungere på den tidligere modellen for å håndtere kollisjoner med andre objekter og tekstilet selv. Provot foreslår å gjennomføre denne prosessen i fire trinn, kollisjonsdeteksjon, optimalisering, respons og feiloppretting.

Modellering
Den underliggende modellen antas å være et masse-fjær nettverk av triangler. Der hver node har en masse og er koblet sammen med sine nabonoder ved hjelp av masseløse fjær[28]. Massene er koblet sammen på tre forskjellige måter gjennom struktur fjær, bøyningsfjær og fjær som kontrollerer forskyvningen i tekstilet. Figur 3.5 illustrer hvordan disse fjærene virker på massene i modellen.



Figur 3.5: Figuren illustrerer masse-fjær-modellen Provot bruker for å modellere tekstiler, den har strukturfjær (gul), bøyningsfjær (rød) og forskyvningsfjær (grønn).

Simulering Tidsintegrasjon
Simuleringen foregår ved at nye verdier for krefter, hastigheter og andre variabler blir beregnet på nytt for hvert tidssteg.

Integrasjon av tidssteget skjer eksplisitt ved bruk av en enkel Euler metode[28].

$$\begin{cases} a_{i,j}(t+\Delta t) = \frac{1}{\mu} F_{i,j}(t) \\ v_{i,j}(t+\Delta t) = v_{i,j}(t) + \Delta t a_{i,j}(t+\Delta t) \\ P_{i,j}(t+\Delta t) = P_{i,j}(t) + \Delta t v_{i,j}(t+\Delta t) \end{cases} \tag{3.3}$$

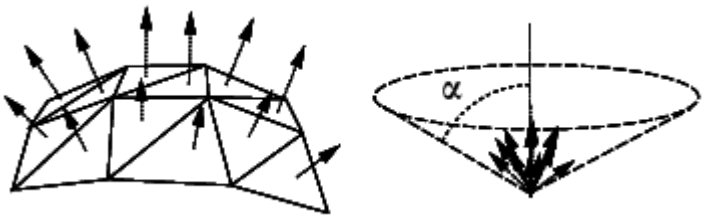
Der $F_{i,j}$ representerer både interne og eksternekrefter i systemtet. $a_{i,j}$ er aksellereasjonen til node i, j, $v_{i,j}$ og $P_{i,j}$ er posisjonen. t er tiden ved starten av tidssteget og Δt er størrelsen på tidssteget.

All bevegelse som oppstår uten fysisk kontakt mellom objekter simuleres ved hjelp av denne teknikken.

Optimalisering
For å redusere antall kollisjonstester som må gjennomføres konstrueres et hierarki av innkapslede volum. Først bygges hierarkiet på grunnlag av 2D-teksturkartet[2] til tekstilet.

Dette hierarkiet konstrueres kun en gang. Posisjonen til massene i modellen i forhold til 2D-teksturkartet brukes som kriteriefor å bestemme innholdet

[2]2D-teksturkartet er en separat 2D datastruktur som inneholder teksturen og mønsteret til objektet.



Figur 3.6: (venstre) For hvert overflatepolygon beregnes en overflatenormal. (høyre) Kjegler konstrueres ut ifra normalene [29].

i hver av nodene i hierarkiet. For hvert tidssteg lages et IVH med dette hierarkiet som grunnlag, hvert volum inneholder posisjonene til nodene i hierarkiet ved tidssteg t_0 og $t_0 + \Delta t$. Ved å søke gjennom dette treet kan kollisjonstester elimineres og algoritmen effektiviseres. Provot sier ikke noe om hvilken type IVH som bør benyttes.

For selvkollisjoner innføres en optimalisering til. Overflatenormalen blir kalkulert for hvert av trianglene involvert, dette gjøres for konvekse områder av modellen da dette er områder hvor selvkollisjon kan oppstå. Et felles kontakt punkt for disse normalene beregnes og ut ifra dette punktet konstrueres en kjegle. Se figur 3.6 for en illustrasjon.

Dersom $\alpha < \pi$, hvor α er halve vinkelen til kjeglens vidde, se figur 3.6, og π er konstanten Pi, kan ikke området kollidere med seg selv og det er derfor ikke nødvendig å gjennomføre en kollisjonstest for selvkollisjon.

Kjeglene konstrueres på grunnlag av tekstur hierarkiet, forklart i forrige avsnitt, og dette skjer derfor ikke ved hvert tidssteg. I beste tilfelle kan denne teknikken totalt eliminere behov for testing av selv-kollisjoner, ved for eksempel et helt flatt tekstil.

Kollisjonsdeteksjon
Kollisjonsdeteksjon gjennomføres for hvert tidssteg og det antas at alle noder beveger seg med konstant fart i løpet av hvert steg. Dette trinnet håndterer deteksjon av to typer kollisjoner, punkt som kolliderer med triangler og kanter som kolliderer med andre kanter.

Første test sjekker om et punkt har kollidert med et triangel. Dersom en kollisjon har oppstått vil punktet ha følgende relasjon til trianglet:

$$\begin{aligned} &\exists t \in [t_0, t_0 + \Delta t] \; slik \; at \\ \exists u, v \in [0,1], u + v \leq 1, &\overrightarrow{P_0P}\,(t) = u\,\overrightarrow{P_0P_1}\,(t) + v\,\overrightarrow{P_0P_2}\,(t) \end{aligned} \tag{3.4}$$

der P er punkter og P_i er trianglets noder. Ved å kombinere denne ligningen med normalen til trianglet kan det løses som et lineært system og det kan

fastslås om en kollisjon har oppstått.

Ettersom det er mulig at objekter kan kollidere kant mot kant uten at noen punkter er involvert tester Provot også for dette.

$$\begin{gathered} \exists t \in [t_0, t_0 + \Delta t] \; slik \; at \\ \exists u, v \in [0,1], u \; \overrightarrow{AB}\,(t) = v \; \overrightarrow{CD}\,(t) \end{gathered} \tag{3.5}$$

Likning 3.5 gir et ikke-lineært system som løses ved å inkludere informasjonen om at i det en kollisjon skjer vil de to kantene ligge i samme plan.

Kollisjonsrespons
Når kollisjon er et faktum, utføres det tredje trinnet i prosessen som er å skape en respons. Når objekter kommer i kontakt med hverandre vil det oppstå friksjon. Lovene for friksjon fungerer på et mikroskopisk nivå og i et uendelig lite tidsintervall, disse egenskapene gjør at lovene ikke kan implementeres direkte. En grunn til dette er at algoritmen jobber med tidsintervaller med endelige størrelser.

Farten til objektene etter kollisjonen er en av parameterene som skal beregnes. Ved å anta at kreftene som er involvert er proporsjonale med farten kan utgangsfarten approksimeres og benyttes for å beregne friksjonskreftene. På grunn av denne antagelsen og antagelsen om at objekter har konstant fart i $[t_0, t_0 + \Delta t]$, kan farten i t_0 benyttes som akselerasjon etter kollisjonen og friksjonen kan beregnes. Fordi kreftene antas å være proposjonale med hastigheten som er konstant i løpet av tidssteget, er dette det samme som å si at kollisjonen skjedde på tidspunktet t_0.

Ved å se på kreftene som er involvert i selve støtet mellom objektet kan ligningen, se figur 3.7 for en illustrasjon, kan følgende ligning settes opp:

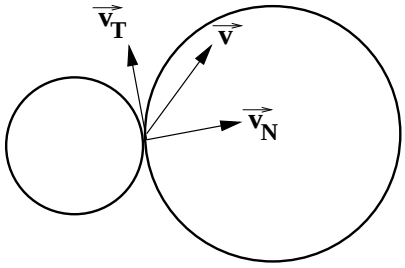
$$\vec{v}' = \vec{v}_T - k_d \; \vec{v}_N \tag{3.6}$$

for deretter å brukes til å beregne farten etter kollisjonen, der v_N er farten normalt på kollisjonspunktet og v_T er farten som tangerer det samme punktet. k_d er en material avhengig koeffisient.

Provot kombinerer uttrykkene for friksjon og de kreftene som er involvert i støtet for å skape et uttrykk for total respons.

$$\begin{cases} Hvis \| \vec{v_T} \| \geq k_f \| \vec{v_N} \|, & \vec{v'} = \vec{v_T} - k_f \| \vec{v_N} \| \frac{\vec{v_T}}{\|\vec{v_T}\|} - k_d \; \vec{v_N} \\ Hvis \| \vec{v_T} \| < k_f \| \vec{v_N} \|, & \vec{v'} = k_d \; \vec{v_N} \end{cases} \tag{3.7}$$

der k_f er friksjonskoeffisienten som er avhengig av materialet.



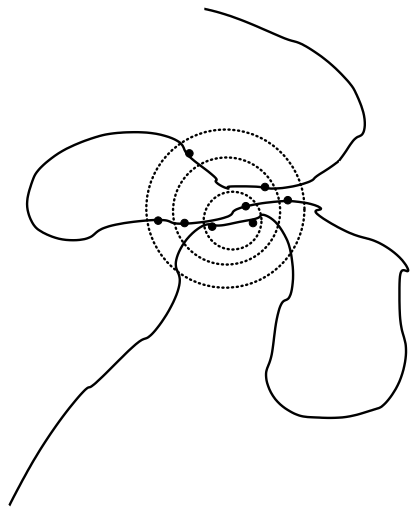
Figur 3.7: To objekter som kolliderer, $\vec{v}$ er farten til det kolliderende punktet, $\vec{v}_T$ er den tangerte farten og $\vec{v}_N$ er normalen til farten.

Dersom trianglet er i bevelgelse beregnes et referansekoordinatsystem i massesentrum av objektene involvert og farten i dette systemet benyttes i beregningene. I tilfeller hvor et objekt kolliderer med seg selv kalkuleres koordinatsystemet ut ifra massene til partiklene som er involvert, enten det er et punkt og et triangel, eller to kanter.

Feiloppretting
Provot erkjenner at metodene han har presentert så langt ikke er tilstrekkelig for å forsikre at feil ikke oppstår. En av grunnene til dette er at han ikke behandler avrundingsfeil tilstrekkelig [7]. For å kompensere for dette innfører han et siste trinn som skal rette opp eventuelle feil og håndtere situasjoner hvor det oppstår flere kollisjoner i samme tidssteg[29].

Metoden som er beskrevet så langt takler alle kollisjoner hvor to elementer er involvert, men ofte er flere elementer involvert. Ved at elementene involvert i en kollisjon får nye hastigheter og posisjoner kan nye kollisjoner oppstå som resultat av dette. For å håndtere dette foreslår Provot å innføre kollisjonssoner. Disse sonene lages ved at et nytt søk etter kollisjoner foretas etter at responskreftene er påført systemet. Alle punkter og kanter som påvirkes av en gitt kollisjon legges til i en kollisjonssone til denne kollisjonen og dersom det oppdages at det finnes felles elementer i forskjellige kollisjonssoner kombineres disse sonene. Dette steget itereres til sonene slutter å vokse og forandre seg. Figur 3.8 viser et snitt av et stykke tekstil hvor flere kollisjoner oppstår, de stiplede sirklene representerer kollisjonssonene.

Elementene i en sone antas å bevege seg som en enhet. Dette er det samme som å anta at kollisjonene som oppstår inne i sonene er perfekt ikke-elastiske, og at det ikke foregår noen gliding mellom elementene. Sonene oppfører seg som rigide objekter i tidssteget og har derfor en gruppehastighet, $\vec{V_G}$, og en gruppevinkelhastighet, $\vec{\Omega_G}$.



Figur 3.8: Snitt av et stykke tekstil hvor flere kollisjoner oppstår. Hver kollisjon danner en kollisjonssone (stiplede sirkler) som kan vokse og kombineres med andre soner. Prikkene er eksempler på noder som blir innlemnet i kollisjonssonene.

$$\vec{V_G} = \frac{1}{n} \sum_{M \in Z_c} \vec{V_M} \tag{3.8}$$

der V_G er gjennomsnittshastigheten til de n punktene, M, i sonen, Z_c.

$\vec{\Omega_G}$ beregnes ut ifra det geometriske sentrumet, G, av sonen:

$$\vec{OG} = \frac{1}{n} \sum_{M \in Z_c} \vec{OM} \tag{3.9}$$

$$\vec{\Omega_G} = \frac{1}{n} \sum_{M \in Z_c} \frac{\vec{GM} \wedge (\vec{V_M} - \vec{V_G})}{\| \vec{GM} \|^2} \tag{3.10}$$

der O er origokoordinatsystemet. Kollisjonsresponsen kan derfor beregnes ut ifra den nye hastigheten:

$$\forall M \in Z_c, \quad \vec{V'_M} = \vec{V_G} + \vec{\Omega_G} \wedge \vec{GM} \tag{3.11}$$

Ingen informasjon fra tidligere kollisjonssoner blir lagret. Derved utføres beregningene for hvert tidssteg hvor det dannes kollisjonssoner. Dette skal forsikre at tekstilet ikke penetrerer seg selv, selv ved tilfeller hvor svært mange kollisjoner oppstår i samme område.

Algoritmen som er presentert er ikke beregnet på sanntidsimulering og den har flere svakheter i at den blant annet håndterer kollisjoner i serie. Dersom

mange kollisjoner oppstår i et tidssteg kan dette tidssteget bli en flaskehals fordi kollisjonene behandles en etter en, i serie.

3.2.3 Robust Treatment of Collisions, Contact and Friction for Cloth Animation[7]

Bridson et al [7] sin algoritme jobber videre med ideen fra Moore og Wilhelms om å benytte fjær ved lette kollisjoner og en analytisk respons ved kraftigere kollisjoner. Metoden presentert av Moore og Wilhelm [25] måtte løse beregningstunge femte grads ligninger og håndtere stive fjær ved kraftige kollisjoner. Metoden ble derfor ikke interessant før Provot [29] presenterte sin algoritme som forenklet flere av problemene.

Modellering
Tekstilet modelleres ved hjelp av et masse-fjærnettverk bestående av triangler, men teknikkene presentert i [7] for kollisjonshåndtering kan og har blitt benyttet i systemer med mer avanserte underliggende modeller.

Et problem med en slik underliggende modell er at trianglene kan bli deformert slik at det påvirker effektiviteten til algoritmen, for eksempel kan triangler deformeres så mye at de blir til punkter eller kanter.

Gjennom å iterativt oppdatere hastighetene til nodene i modellen forhindrer Bridson et al at trianglene blir deformert mer enn 10%. Ved hver iterasjon kalkuleres fjærenes nye lengde på grunnlag av hastighetene, dersom det viser seg at fjærene blir mer enn 10% deformert i forhold til sin hvile lengde benyttes en momentumkonserverende kraft for å forhindre dette. Fordi en slik korreksjon kan ha ringvirkninger til nabonoder fortsetter iterasjonen.

Simulering og Tidsintegrasjon
Algoritmen benytter tidssteg som tilpasser seg aktiviten i simuleringen. Ved kollisjoner har algoritmen mulighet til å halvere tidsstegene og på den måten behandle hendelsen mer detaljert. Når kollisjonen er ferdig behandlet gjenopprettes lengden på tidsstegene.

Ved hvert tidssteg kalkuleres nye hastigheter for nodene i modellen. Disse nye verdiene brukes deretter for å beregne den nye lengen på fjærene som holder nodene sammen. For å passe på at trianglene ikke blir deformert mer enn 10% påføres en momentumbevarende kraft på kantene, som er representert av fjær. Ettersom denne handlingen kan gjøre at nærtliggende triangler blir deformert mer enn 10%, må steget utføres iterativt på alle triangler.

Parallell Jacobi iterasjon benyttes for å beregne steget, problemet med valget av den metoden er at den ikke kan garantere å konvergere, men i de fleste tilfeller vil bare et par iterasjoner være nok.

Optimalisering
I starten av simuleringen konstrueres et hierarki på grunnlag av topologien til det underliggende materialet. Nodene i dette hierarkiet består av volum, hvert volum inneholder en subgrupper av modellens triangler, disse volumene er så utvidet med tekstilets tykkelse. Dette er gjort fordi punkter ikke skal kunne passere uoppdaget.

Volumene er plassert slik at de ligger parallelt med aksene i koordinatsystemet. Konstruksjonen starter nederst i hierarkiet og nivåer blir tilført til rotnoden nåes. For at hierarkiet alltid skal gjenspeile den korrekte topologien til tekstilet blir det rekonstruert ved hvert tidssteg [7]. Alternativt kunne hierarkiet blitt oppdatert, men det er ikke nødvendigvis noen fordeler med dette [33].

Kollisjonsdeteksjon
Gjennom å traversere hierarkiet avgjøres det hvilke triangler og punkter faktiske kollisjonstester må utføres på [7]. Testene utføres på punkter mot triangler og kanter mot kanter på lik måte som i Moore og Wilhelms [25] og Provots [29] metoder. Dersom et punkt er nærmere enn en forhåndsdefinert avstand fra planet trianglet ligger i, eller dersom to kanter er nærmere enn en konstant, vil kantene bli testet for kollisjon og tidspunket for kollisjonen blir beregnet. For eksempel, i dette tilfellet søkes det etter punkter og kanter som er nærmere enn 1mm, som er tekstilets tykkelse.

Kollisjonsrespons
Når kollisjonen er oppdaget beregnes kreftene som skal gi respons. Ettersom punkter hvor som helst på tekstilet kan inngå i en kollisjon, benyttes lineær interpolasjon av hastigheten til hjørnene i i gjeldende triangel for å finne hastigheten. Når krefter skal påvirke stoffet veies kollisjonspunktet i forhold til hjørnene i trianglet og kreftene påføres deretter i riktig mengde til riktig hjørne.

Når hastigheter er bestemt kan respons beregnes. To typer frastøtende krefter benyttes. Den første er basert på en ikke-elastisk kollisjon og den andre er en fjær. Ved en kollisjon vil tekstilet deformeres og energi lagres i stoffet. Denne energien slippes løs når tekstilet finner tilbake til sin opprinnelige form og tekstilet vil da sprette ut ifra kollisjonspunktet. Sett at to punkter er nær hverandre og har relativ fart v_N slik at de er på kollisjonskurs, da vil kraften som hindrer kollisjonen være $I_c = mv_N/2$. Denne kraften fordeles ved hjelp av interpolasjon slik forklart i forrige avsnitt.

For å modellere at tekstilet presses sammen i en kollisjon benyttes en fjær. Kraften til fjæren er proporsjonal avstanden punktet beveger seg inn i tekstilet. Dersom fjæren kan bli uendlig lang oppstår stive ligninger, men fordi bruk av kollisjonssoner, lik de i [29], forhindres fjæren å bli lengre enn

tekstilets tykkelse ettersom tekstilet ikke kan penetrere seg selv.

Responskreftene kan enten påføres systemet serielt eller parallelt. Ulempen ved den parallelle løsningen er at krefter kan påvirke hverandre slik at de blir forsterket. For eksempel, triangel A påvirkes av krefter fra triangel B og C i tidssteg 1, i tidssteg 2 har triangel B blitt påvirket av krefter fra triangel C som igjen gjør at triangel B kolliderer med triangel A. Dette kan føre at triangel A blir påvirket av de samme kreftene fra triangel B mer enn en gang, og på denne måten blir kreftene forsterket.

Newtons tredje lov sier at når det virker en kraft på et legeme, virker det en like stor og motsatt rettet kraft fra legemet. Responskreftene som ble beskrevet i de to foregående avsnittene er den motsatte av kreftene som deformerer tekstilet.

Den delen av hastighetsvektoren som tangerer kollisjonspunktet gir hastigheten før friksjonen virker, $\vec{v}_T^{pre}$, friksjonskraften virker i motsatt retning av denne. Integrasjon av farten gir kraften som virker, ved å teste denne kraften mot en friksjonskoeffisient kan det avgjøre hvor mye farten skal bremses eller om den skal stoppe helt. Ligningen for den nye farten i kollisjonspunktet etter at friksjonen er:

$$\vec{v}_T = max(1 - \mu \frac{\Delta v_N}{|\vec{v}_T^{pre}|}, 0)\ \vec{v}_T^{pre} \tag{3.12}$$

der μ er friksjonskoeffisienten.

For å håndtere flere kollisjoner som oppstår i samme tidssteg benyttes samme teknikk som i Provot [29]. Det bemerkes imidlertid at likningen for den angulære farten prestentert der, inneholder feil og en korrigert versjon fremstilles derfor her. Først beregnes massesentrum og dens gjennomsnittlige hastighet:

$$\vec{x}_{CM} = \frac{\sum_i m\ \vec{x}_i^n}{\sum_i m}, \quad \vec{v}_{CM} = \frac{\sum_i m\ \vec{v}_i^{n|1/2}}{\sum_i m} \tag{3.13}$$

Angulær hastighet vil være er:

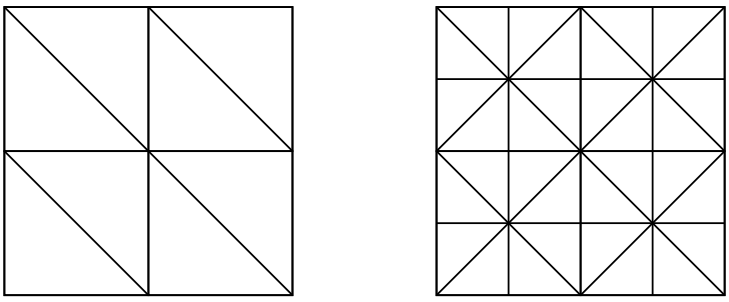
$$\vec{\omega} = I^{-1}\ \vec{L} \tag{3.14}$$

$$\vec{L} = \sum_i m(|\ \vec{x}_i^n - \vec{x}_{CM}) \times (\vec{v}_i^{n+1/2} - \vec{v}_{CM}) \tag{3.15}$$

$$I = \sum_i m(|\ \vec{x}_i^n - \vec{x}_{CM}\ |^2 \delta - (\vec{x}_i^n - \vec{x}_{CM}) \otimes (\vec{x}_i^n - \vec{x}_{CM})) \tag{3.16}$$

Ut ifra disse likningene blir den nye hastigheten til noden i:

$$\vec{v}_i = \vec{v}_{CM} + \vec{\omega} \times (\vec{x}_i - \vec{x}_{CM}) \tag{3.17}$$



Figur 3.9: Figur som illusterer hvordan det underliggende nettverket kan forfines ved å dele hvert triangel langs kantene.

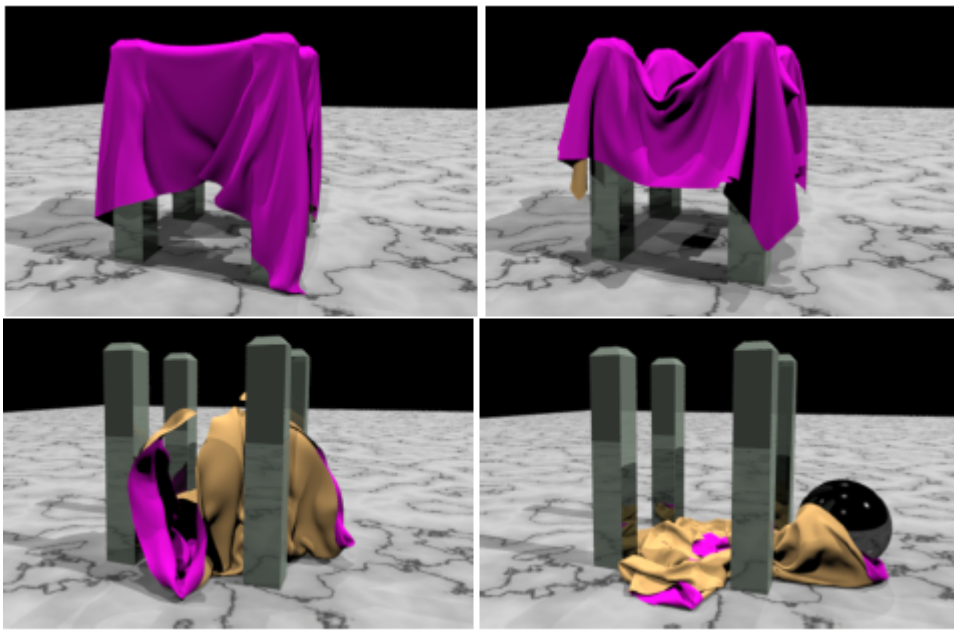
Sammen med responskreftene passer denne teknikken på at kollisjonssonene holder seg små, isolerte og sjeldne. Når de først oppstår lever de kort tid fordi responskreftene dytter de kolliderende elementene fra hverandre. Likevel kan det oppstå synlige "feil" under animeringen, spesielt gjelder dette i tilfeller hvor det er kraftig bøying av tekstilet.

Feiloppretting
For å motvirke visuelle feil forfines det underliggende nettverket ved at noder blir tilført ved midtpunktet av hver kant. Figur 3.9 illusterer hvordan denne forfiningen kan foregå.

Bridson et al påpeker at dette arbeidet ikke behandler hvodan tekstiler oppfører seg i kollisjoner med skarpe gjenstander eller når det blir klemt mellom objekter. Kombinasjonen av den robuste behandlingen av geometriske kollisjoner og den effektive responsbehandlingen gir visuelt gode resultater slik figur 3.10 illusterer.

3.2.4 Oppsummering

Innkapslede volum hierarkier (IVH) er blant de mest effektive datastrukturene brukt ved kollisjonshåndtering. For deformerende objekter er det viktig å bruke en datastruktur som lar seg oppdatere eller rekonstruere effektivt. Moore og Wilhelms [25], Provot [29] og Bridson et al [7] har alle brukt denne datastrukturen som basis for sine algoritmer og har i tillegg benyttet teknikker som gir realistisk kollisjonsrespons samtidig som tekstilet forhindres fra å trenge gjennom seg selv eller andre objekter uten at dette er hensikten.



Figur 3.10: Figur som viser animasjon av tekstiler ved hjelp av metoden presentert av Bridson et al[7].

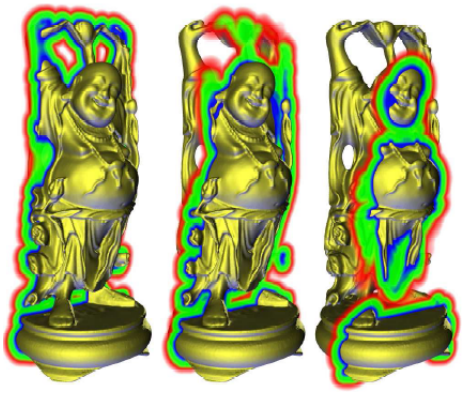
3.3 Distansefelt

Distansefelt gir informasjon om minimum distanse til nærmeste overflate for alle punkter i et felt. Uniforme 3D-nettverk, oktaltre og BSP-trær er alle datastrukturer som har blitt brukt til å konstruere slike felt.

Ved uniforme 3D nettverk beregnes verdier for hver node i nettverket, og interpolasjon benyttes for å gi mellomliggende verdier. Oktaltre har blitt brukt til å konstruere samplingstilpassede distansefelt (SDF)[3] slik som for eksemple i Bridson et al [8]. Ved bruk av BSP-trær kan minnebruken reduseres, men bygging av slike datastrukturer er nokså dyrt. Distansefelt kalles også distansevolum, distansefunksjoner og kraftfelt [33].

Fargene rundt statuen i figur 3.11 illusterer avstand til overflaten, hver farge inneholder alle punkter som ligger en viss distanse, δ fra overflaten. Fargene illustrerer det som utgjør et distansefelt rundt statuen.

[3]eng. Adaptive sampled distance field (ADF)



Figur 3.11: Fargene illusterer et distansefeltet som omgir figuren. Figur en, to og tre viser tre forskjellige snitt av feltet langs en akse [33].

3.3.1 Cloth Animation with Self-Collision Detection[24]

Lafleur et al jobber videre med ideene fra Moore og Wilhelms [25]. De ser spesielt på problemet med deformerende objekter som kolliderer med rigide objekter, for eksempel tekstiler som kolliderer med kroppsdeler. Et problem med de foreslåtte metodene fra [25] er at den kan ikke garantere at alle kollisjoner blir oppdaget, på den måten kan det oppstå situasjoner hvor stoffet hekter seg sammen på unaturlig vis.

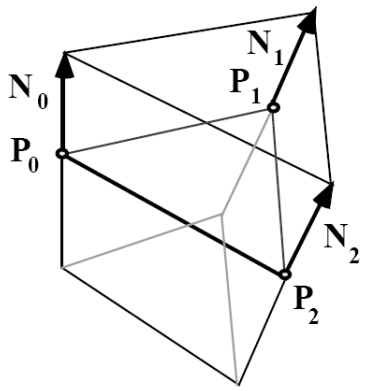
Modellering
Det underliggende nettverket er modellert som i [25], noder med masse, koblet sammen av fjær i form av triangler som danner overflaten til tekstilet.

Simulering og Tidsintegrasjon
Simuleringen forgår i tidssteg, denne teknikken krever at tidsstegene er små nok til at punkter ikke kan gå ubemerket gjennom en overflate.

Problemer kan oppstå når triangler deformeres for mye eller når vinkelen mellom nabotriangler er større enn 180°. Disse problemene løses ved å angi funksjoner som begrenser vinkelen mellom triangler og som angir et minimalt og maksimalt areal et triangel kan ha.

Tidsintegrasjon foregår ved at det oppnåes en likevekt mellom alle de interne og eksterne kreftene som påvirker modellen. Lafleur et al sier ikke direkte hvilke krefter de bruker i sin modell.

Optimalisering
Et kraftfelt skal beskytte overflaten slik at penetrering skal unngås. Feltet konstrueres ved at modellen deles inn i ikke-overlappende soner, eller celler, som dekker hvert sitt triangel. Hver sone omgis av et volum som



Figur 3.12: Trianglene blir omgitt av et kraftfelt som skal forhindre at punkter penetrerer overflaten, feltet defieneres av normalene i hjørnene til trianglet og en konstant w som gir dybden til feltet på hver side av modellen [24].

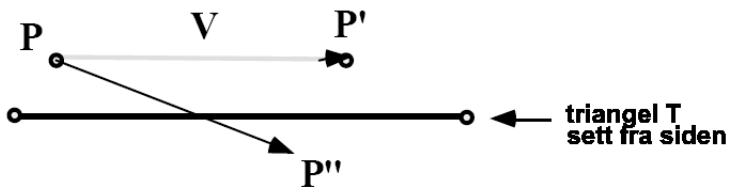
defineres av normalene i hvert av hjørnene i trianglet og en konstant w som sier noe om hvor tykt feltet skal være på hver side av modellen. Disse volumene organiseres ved hjelp av et hierarki og på den måten er denne algoritmen en kombinasjon av IVH og distansefelt. Figur 3.12 viser hvordan et triangel blir omgitt av et slik distansefelt, eller kraftfelt som det blir kalt.

Så snart et punkt beveger seg inn i kraftfeltet skal det påvirkes av responskrefter slik at punktet ikke får mulighet til å havne på feil side av trianglet.

Kollisjonsdeteksjon
Kollisjonsdeteksjon løses slik som foreslått av Moore og Wilhelm i [25] med en modifikasjon slik at det fungerer med distansefeltet. Punkter blir testet mot triangler for tilfeller hvor trianglet er stillestående, se ligning 3.1 på side 17, og for tilfeller hvor trianglet er i bevegelse, se ligning 3.2 på side 18.

Ligningene 3.1 og 3.2 antar at punktet følger retningsvektoren gitt ved starten av tidssteg, da dette ikke nødvendigvis er sant kan det oppstå uforutsette kollisjoner. Tekstiler er ofte involvert i dynamiske systemer der krefter kan påvirke punktene i løpet av et tidssteg, på den måten kan det oppstå kollisjoner som ikke oppdages av denne ligningen. Figur 3.13 viser hvordan krefter som påvirker punktet i løpet av tidssteget fører til en kollisjon som ikke kan fortsees av ligningene.

Dette kan løses ved å beregne ligningene på nytt med de nye posisjonene og hastighetene for neste tidssteg og deretter repetere dette steget til penetrering er ungått. Dette ville vært svært beregningstungt og dersom



Figur 3.13: På grunn av krefter som påvirker P i løpet av tidssteget vil sluttpunktet være P″ og ikke P′[24].

det oppsto mange kollisjoner i løpet av en kort tidsperiode kunne algoritmen gått inn i en evig løkke.

Distansefeltet som ble forklart under optimaliseringen har som formål å løse disse vanskelighetene. Kollisjon med distansefeltet beregnes ved en modifisert utgave av ligning 3.2.

$$\begin{aligned} P = P_0 + N_0 w \\ +((P_1 - P_0) + (N_1 - N_0)w)u + ((P_2 - P_0) + (N_2 - N_0)w)v \end{aligned} \tag{3.18}$$

for $0 \leq u < 1$, $0 \leq v < 1$, $u + v = 1$ og $-0.5w_{min} \leq w \leq 0.5w_{min}$. Parameterne u,v og w definerer kraftfeltets volum og w_{min} er lengden av den korteste kanten i det gjeldende trianglet.

Kollisjonsrespons
Ved hjelp av en fjær som midlertidig plasseres i "kollisjonspunktet" vil punktet føres bort fra kraftfeltet. Kraften består av en del fjærkraft og en del dempningskraft, dempningskraften skal redusere effekten av eventuelle osillasjoner som kan oppstå. Dersom fjæren kan bli uendelig lang, oppstår stive ligninger som er svært vanskelig å løse, i dette systemet er det ingen garanti mot at dette kan skje ettersom punktet kan være så raskt at det passerer gjennom kraftfeltet uten å oppdages.

3.3.2 Simulation of Clothing with Folds and Wrinkles [8]

Realisme blir stadig viktigere innenfor områder av grafikk. Spesielt innen filmproduksjon er det essensielt at simulerte tekstiler opptrer så nært opp mot virkeligheten som mulig og det er dette som er motivasjonen til Bridson et al [8].

Modellering
Teknikkene er ikke avhengig av en spesifikk underliggende modell for å fungere. Kravet til modellen vil være at den kan gi den informasjonen teknikkene krever i form av hastigheter og posisjoner.

Simulering og Tidsintegrasjon
Simuleringen foregår ved å beregne nye verdier for de variablene som påvirker systemet, dette skjer ved å benytte tidssteg som tilpasser seg hendelsene i løpet av simuleringen.

En svært stor del av animasjonprosessen er å løse de differensial ligningene som oppstår. Eksplisitte metoder gir fleksibilitet og enkelhet, mens implisitte metoder er bedre for effektivitet.

Bridson et al eksperimenterer med å kombinere eksplisitte og implisitte metoder for å dra nytte av fordelene med teknikkene. Eksplisitt integrasjon benyttes på elastiske krefter, for eksempel krefter som er uavhengig av hastigheter, mens implisitte teknikker benyttes i forbindelse med dempingskrefter.

Posisjoner blir oppdatert ved bruk av den eksplisitte "leapfrog" teknikken, og hastigheter behandles av trapesregelen som er implisitt. Algoritmen for simulering av et tiddssteg ved bruk av disse teknikkene er gitt av:

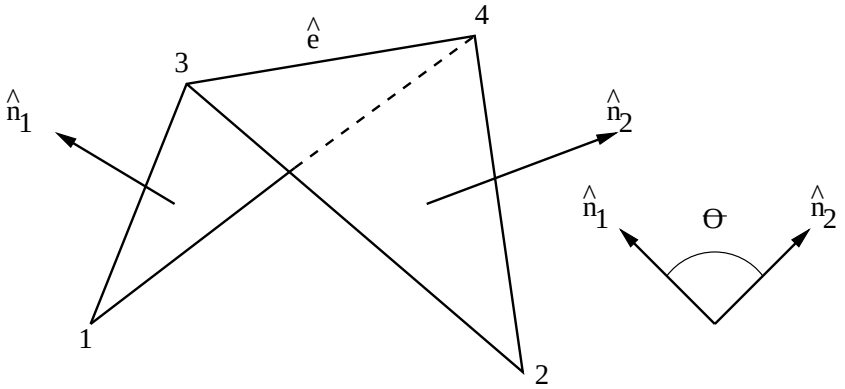
- $v^{n+1/2} = v^n + \frac{\Delta t}{2} a(t^n, x^n, v^n)$ (eksplisitt)
- Modifiser $v^{n+1/2}$ for å få $\tilde{v}^{n+1/2}$ slik at begrensninger etc. blir oppfylt
- $x^{n+1} = x^n + \Delta t \tilde{v}^{n+1/2}$ (eksplisitt)
- $v^{n+1} = v^{n+1/2} + \frac{\Delta t}{2} a(t^{n+1}, x^{n+1}, v^{n+1})$ (implisitt)
- Modifiser v^{n+1} slik at begrensninger etc. blir oppfylt

Ved å benytte trapesregelen i stedet for en eksplisitt metode, som for eksempel Runge-Kutta, kan algoritmen ta langt færre tidssteg. Forskjellen mellom disse to metodene betyr èn orden færre steg. Dette betyr at der Runge-Kutta ville brukt 10^2 tidssteg vil trapeseregelen bruke kun 10^1 steg. Implisitte metoder med større tidssteg ville fungere, men dette ville ført til unaturlig demping av krefter.

Måten tekstiler bøyer seg på er karakteristisk, men fysikken bak de underliggende mekanismene er dårlig forstått. Likevel kan en del egenskaper identifiseres slik at fysisk sannsynlige simuleringer er mulig å gjennomføre. Ved å se på triangler som deler en kant kan en del variabler og deres samspill utredes, figur 3.14 illustrerer dette forholdet. $1, 2, 3$ og 4 er hjørnene i trianglene, $\hat{n}_1$ og $\hat{n}_2$ er overflatenormalene til trekantene, $\hat{e}$ er en akse brukt i beregningen og Θ er vinkelen mellom normalene som benyttes for å avgjøre hvor stor bøyingen er.

Bøyingskraften består av en elastisk, F_i^e, og en dempende kraft, F_i^d. Disse defineres som følger:

$$F_i^e = k^e \frac{|E|^2}{|N_1| + |N_2|} (sin(\Theta/2) - sin(\Theta_0/2)) u_i \tag{3.19}$$



Figur 3.14: Illustrasjon av sammenhengen mellom variabler involvert i bøying. $\hat{n}_1$ og $\hat{n}_2$ er overflatenormalene til trianglene og Θ er vinkelen mellom disse normalene, altså den sier hvor stor bøyning det er mellom trianglene.

$$F_i^d = k^d |E| (d\Theta/dt) u_i \tag{3.20}$$

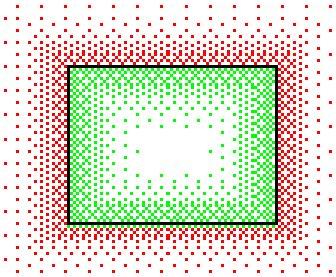
der k^e og k^d er koeffisienter for elastisitet og demping, $N_1 = (x_1 - x_3) \times (x_1 - x_4)$, $N_2 = (x_2 - x_4) \times (x_2 - x_3)$, $E = x_4 - x_3$ og Θ_0 er hvilevinkelen. Disse kreftene bevarer lineært og angulært momentum og bidrar derfor til å lage en fysisk mulig modellering.

Optimalisering
Selv-kollisjoner løses slik som i den tidligere artikkelen av Bridson et al [7], mens kollisjoner mellom tekstiler og andre objekter behandles ved hjelp av distansefelt.

Modellen initialiseres ved små positive og negative verdier for utsiden og innsiden av modellen. Deretter benyttes en såkalt fort marsjerende metode (eng. fast marching method) for å generere distanse funksjonenes som vil gi distansefeltet. Disse funksjonene tilpasser seg detaljer i topologien ved å sample oftere der det er mange detaljer og lagres i oktaltre for å minimere lagringskostnader. De deformerende objektene kan forandre form og derfor må hele distansefeltet oppdateres i starten av hvert tidssteg.

Kollisjonsdeteksjon
Kollisjoner sjekkes ved å søke etter punkter som kommer innenfor dette feltet. Punktet, p, har en posisjon i forhold til feltet og ϕ er verdien som sier hvor i feltet punktet befinner seg. Kollisjoner oppstår for når p er i feltet slik at $\phi < 0$, hvordan ϕ avtar med avstanden fra objektet er illustrert i figur 3.15.

For punkter som har $\phi < 0$ beregnes avstanden p må bevege seg langs normalen, N, for å komme til overflaten, denne teknikken refereres til som overflatevarsel (eng. interface forecasting). Dersom punkter bare ble



Figur 3.15: På utsiden av polygonet er ϕ positiv (rød) og på innsiden er ϕ negativ (grønn). ϕ gir avstanden til kanten av polygonet.

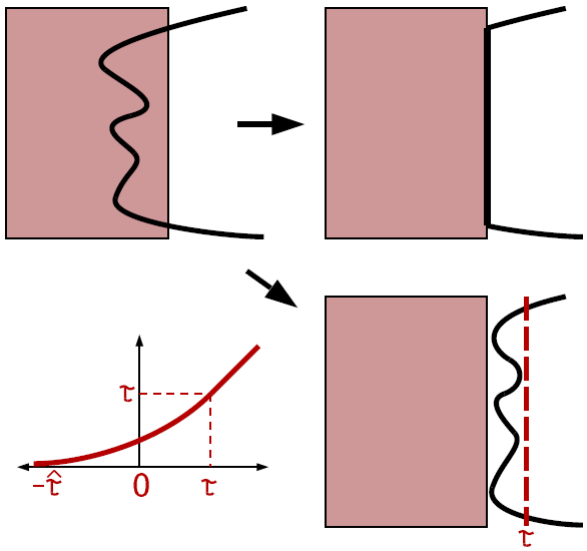
skjøvet til overflaten av objektet de penetrerte ville folder og rynker bli glattet ut og animasjonen ville se unaturlig ut. Problemstillingen er illustrert i figur 3.16. τ er en gitt toleransegrense som definerer et intervall hvor tekstilet kan skyves tilbake inn i, det vil si τ definerer en sone på utsiden av modellen hvor tekstilets folder og rynker skal være innenfor etter at de er flyttet ut av modellen. Punkter med $\phi > \tau$ modifiserers ikke, ettersom folder og rynker i denne situasjonen vil bli bevart uansett, for alle andre punkter benyttes en monoton funksjon som mapper $[-\infty, \tau]$ til $[0, \tau]$, denne funksjonen bevarer folder og rynker, for å flytte tekstilet fra innsiden til utsiden av objektet.

Kollisjonsrespons
Respons på kollisjonene beregnes gjennom friksjon som kalkuleres på samme måte som i Bridson et al [7] (se blant annet likning 3.12 på side 27).

Til slutt foreslår Bridson et al å innføre klebrige soner i situasjoner hvor animatøren har behov for å kontrollere tekstilets bevegelse og oppførsel ut over de lovene tøyet er programmert til å følge. Disse sonene forandrer seg med tiden og følger den underliggende geometrien. Sonene består av punkter som festes relativt til et annet element i scenen, når elementet beveger seg følger punktet etter under påvirkning av en fjær. Fjæren har ingen lende når den ikke er under påvirkning av noen krefter. Dersom avstanden mellom punktet og elementet blir for stor, det vil si fjæren påføres for mye kraft, vil fjæren ryke og punktet beveger seg ikke lengre relativt til elementet. Ved hjelp av disse klebrige sonene kan det visuelle resultatet kontrollere til en viss grad.

3.3.3 Oppsummering

Distansefelt kan konstrueres ved å gjøre utsiden av modellen positiv og innsiden negativ. Dersom et punkt er i et felt hvor verdien er negativ har



Figur 3.16: Tekstiler kan skyves til overflaten slik at rynker glattes ut (øverst) eller de kan skyves ut til en grense på utsiden av objektet gitt av τ (nederst) [8]

.

en kollisjon oppstått, gitt at punktet startet på utsiden av objektet. Bridson et al [8] har i tillegg foreslått en teknikk som bevarer folder og rynker ved kollisjoner som gir økt realisme til simuleringen.

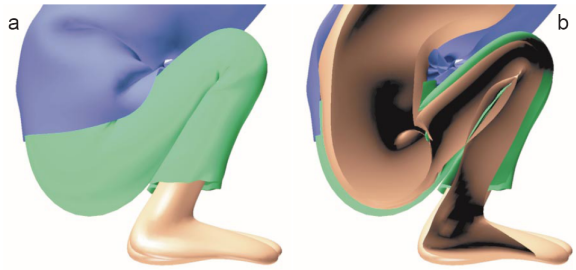
3.4 Teknikker for å korrigere feil

I tilfeller hvor kollisjoner ikke blir oppdaget kan simuleringen av tekstilet føre til unaturlig oppførsel som, for eksempel, at tøy hekter seg sammen. Det er veldig vanskelig å finne teknikker som kan garantere at dette ikke skjer og som samtidig er effektive nok til å benyttes i simuleringer.

De fleste teknikker baserer seg på historien til simuleringen for å løse slike problemer, men dersom det har oppstått feil som ikke har blitt oppdaget kan tekstilet permanent hekte seg i andre objekter eller seg selv uten at det finnes muligheter for å rette opp dette [3].

3.4.1 Untangling Cloth [3]

Ved hjelp av en historiefri global analyse, som sjekker tekstilet for kollisjoner, presenterer Baraff et al[3] en teknikk for å løse opp eventuelle



Figur 3.17: Eksempel på trange områder hvor tekstiler lett kan hekte seg sammen.[3]

kollisjoner som ikke skulle ha oppstått. Simulering av mennesker kommer ofte opp i situasjoner hvor tekstiler blir klemt tett inntil hverandre, for eksempel i armhuler og knehaser, se figur 3.17. Tekstiler kan hekte seg sammen på unaturlig vis og teknikker for å løse opp dette, som for eksempel den presentert i [7], kan føre til visuelle feil i simuleringen.

I følge forfatterne selv er dette den første algoritmen som retter opp slike feil uten bruk av historie. De påpeker at denne algoritmen er nødvendig, selv om Bridson et al [7] garanterer at tekstiler ikke skal kunne hekte seg sammen. Grunnen til dette er at det kan forekomme eksterne krefter som tvinger frem slike feil og Bridson et als algoritme tar ikke høyde for slike krefter.

Modellering
Tekstilet er modellert slik beskrevet tidligere av Baraff et al [2] med noen modifikasjoner. Modellen er et nettverk av triangler som påvirkes av både eksternekrefter og dempningskrefter. Krefter i lengde- og bredderetning formuleres per triangel og bøyingskreftene beregnes for nabotriangler.

Simulering og Tidsintegrasjon
Baraff et al [3] har erfart at tekstiler som er klemt sammen mellom stillestående objekter, opptrer mest realistisk dersom tekstilet da også er helt bevegelsesløst. En opplagt måte å løse dette på er å frata fastklemte partikler hastigheten, for å forsvare denne teknikken sier de at friksjonskreftene i slike situasjoner overgår alle andre krefter som måtte være tilstede. De kaller teknikken fluepapir, fordi partikler klistres fast i bestemte posisjoner.

Beregning av nye verdier for de interne og eksterne kreftene gjøres ved hjelp av en ett-stegs implisitt Euler metode.

Optimalisering
La S være overflaten til et solid objekt og la $W_S(\mathbf{b}, t)$ være en funksjon som transformerer $\mathbf{b}$ fra å være et punkt i et objekts koordinatsystem til å bli

et punkt i verdenskoordinatsystemet i tiden t. Den inverse funksjonen er $W_S^{-1}(\mathbf{w}, t)$, der $\mathbf{w}$ er et punkt i verdenskoordinatsystemet.

For å kunne benytte fluepapirteknikken må partikler kunne festes til flere solide objekter i relativ bevegelse samtidig. Først relateres det fastklemte punktet til hvert av de aktuelle solide objektenes koordinatsystemer. Deretter defineres et "'mål"' for de relative punktene.

$$\mathbf{b}_i = W_S^{-1}(\mathbf{p}(t_0), t_0) \quad og \quad \mathbf{g}_i(t) = W_S(\mathbf{b}_i, t) \tag{3.21}$$

Det fastklemte punktets posisjon i tiden t_1 kan nå begrenses ved hjelp av

$$\mathbf{p}(t_1) = \frac{1}{n} \sum_i \alpha_i \mathbf{g}_i(t_1). \tag{3.22}$$

der α_i er en verdi som veier hvor mye hvert solide objekt skal påvirke punktet. Dersom en person sklir på knærne over gulvet vil buksen sitte fast til knærne og ikke gulvet, her vil $\alpha_i = 1$ for knærne. Et annet eksempel er dersom et menneske gnir seg på magen, da vil skjorten til en viss grad følge hendenes bevegelse alt ettersom hvor hardt personen gnir, her vil α_i ha høyere verdi for magen enn for hånden.

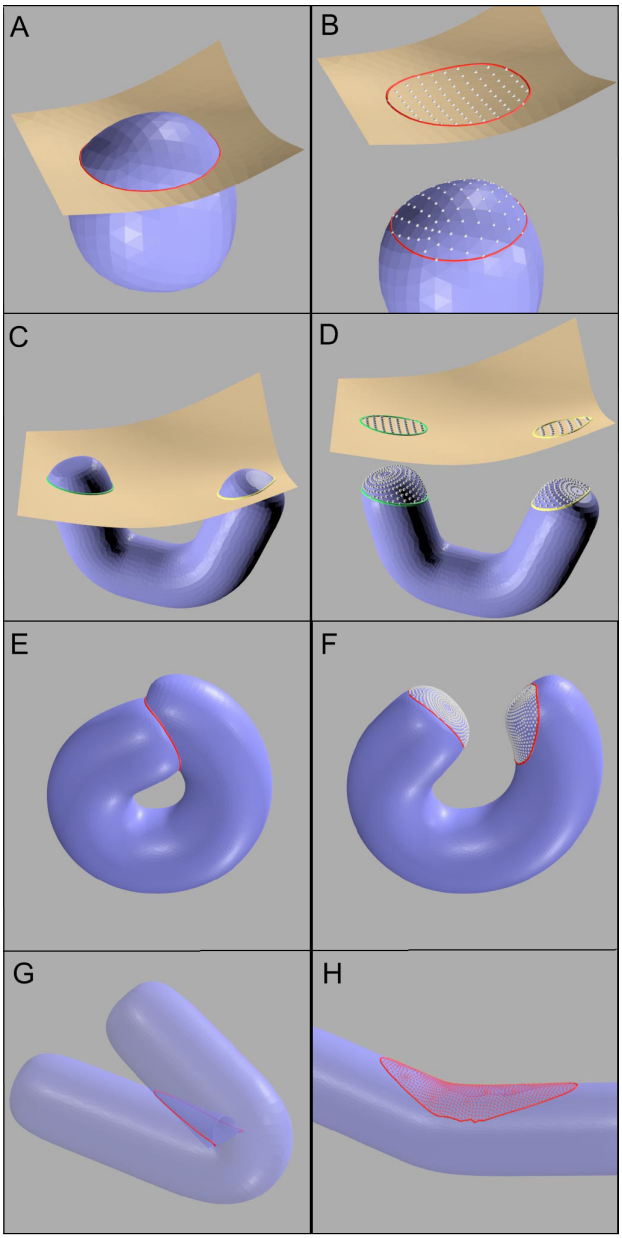
Ved å ignorere den simulerte historien til tekstilet må vektoren $\mathbf{b}_i$ beregnes på nytt for hvert tidssteg. Fordelen er at ved plutselige forandringer i antall solide objekter ikke påvirker det fastklistrede punktet.

Kollisjonsdeteksjon og Feiloppretting
For å bestemme hvilke punkter som skal festes med fluepapir benyttes en teknikk kalt globalt kryssningspunkt analyse (eng. global intersection analysis (GIA)). GKA prosessen er som følger:

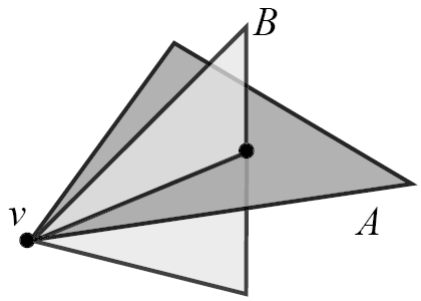
1. Finn kurven som deler de kolliderende objektene.
2. Velg det minste området kurven definerer og farg de berørte trianglene for begge de berørte objektene.

Kollisjonene som danner kurven mellom de kolliderende objektene kan lokaliseres ved å søke igjennom et forhåndsberegnet hierarki slik som i [7] der alle kanter i det første objektet (M_1) sjekkes mot alle triangler i det andre objektet (M_2) og motsatt. Når en kollisjon blir oppdaget går algoritmen langs kantene som kolliderer til den kommer tilbake til utgangspunktet Dersom kanten av et av objektene nåes i løpet av denne prosessen starter algoritmen fra startpunktet og går den andre veien. Når den da fullfører forenes de to stiene til en enkel kurve.

Figur 3.18 viser de forskjellige måtene objekter kan kollidere på. Venstre side viser den orginale tilstanden, mens høyre side viser objektene separert slik at de fargede områdene blir synlig, de fargede områdene representerer det berørte området som skal separeres.



Figur 3.18: (A,B) er to objekter med ett kollisjonsområde og (C,D) er to objekter med to kollisjonsområder. (E,F) viser et objekt som kolliderer med seg selv på samme måte som (A,B). (G,H) er et spesialtilfelle av selvkollisjon.[3]



Figur 3.19: Trianglene som kolliderer deler en node, denne noden kalles en "loop node".

Spesialtilfeller (G,H) innebærer at triangler kolliderer slik figur 3.19 illustrerer, trianglene deler en "loop node". Beregningen av kurven i dette tilfellet foregår ved at kanter følges til en loop node"oppdages, algoritmen hopper da tilbake til startpunktet og går den andre veien til nok en "loop node" oppdages, deretter kombineres de to stiene.

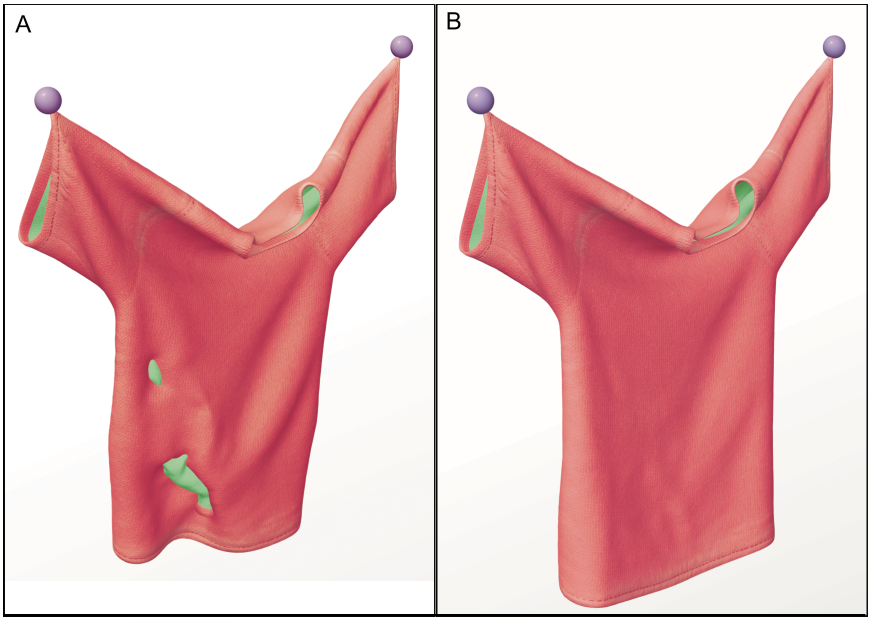
Når kollisjonskurvene er beregnet benyttes en standard "flood-fill" algoritme for å farge områdene de definerer. Områder som kollidere med andre objekter farges svart og hvitt, mens områder som kolliderer med seg selv farges rødt. Området som har minst areal velges ut som området som skal påføres responskrefter. Begrunnelsen for denne utvelgelsen er at områder som blir hektet sammen er relativt små så lenge tidsstegene ikke blir alt for store. Teorien er at ved små tidssteg har ikke tekstilet "tid" til å danne store kollisjonsområder slik det er snakk om her.

Dersom kollisjonskurven ikke er lukket vil dette oppdages ved at samme område blir forsøkt farget to ganger. Fargingen kan effektivisers ved å farge områdene parallelt. Det området som blir ferdigfarget først er minst og velges (så lenge alle triangler er like store). Områder som kolliderer med seg selv farges rødt, og områder der to objekter er involvert farges svart og hvitt.

Et problem er kollisjoner som gir en kollisjonskurve, men ikke et kollisjonsområde. Baraff et al [3] erkjenner at de ikke har noen løsning på dette problemet.

Kollisjonsrespons
Når GKA er fullført returneres fargene med tilhørende noder. Respons blir da beregnet ut ifra det følgende:

1. Kollisjonsområdene dras mot hverandre dersom tilsvarende punkter i to områder har forskjellig farge. Dette gjør at objektene ikke lenger er hektet sammen. Se figur 3.18 på forrige side (A->F).



Figur 3.20: (A) Skjorte med visuelle feil. (B) Samme skjorte etter at fluepapirteknikken og GKA er benyttet [3]

.

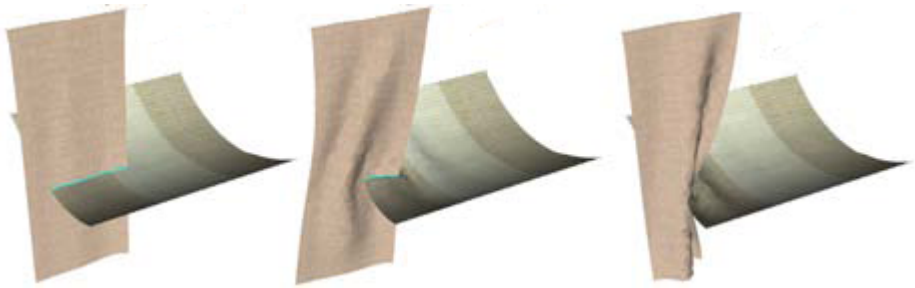
2. Ingen responskrefter påvirker triangelet dersom det er farget rødt, responskrefter fra andre kollisjoner vil løse dette i følge Baraff et al [3]. Se figur 3.18 på side 39 (G,H).
3. Objektene skyves fra hverandre dersom punkt 1 og/eller 2 ikke utføres, og kollisjonen unngås.

Det vil oppstå problemer med punkt 2 dersom dette er den eneste kollisjonen som oppstår, men ettersom denne teknikken er beregnet for bruk i animasjonsfilmer med mennesker som bruker klær vil det neppe hende ofte.

Figur 3.20(A) viser en skjorte som er utsatt for krefter som har ført til at unaturlige kollisjoner har oppstått. I figur 3.20(B) brukes de to teknikkene, fluepapir og GKA, og resultatet er at feilene rettes opp.

3.4.2 Resolving Surface Collisions through Intersection Contour Minimization [40]

Problemet med teknikken presentert i [3] er blant annet at den avhenger av godt definerte kollisjonskurver. I praksis har det vist seg, i følge Volino og Thalmann [40], at slike situasjoner ofte oppstår spesielt når søking



Figur 3.21: Orienteringen til kollisjonsområdet har ingen mening her i forhold til teknikkene presentert av Baraff et al[3], det vil si det er ikke definert en løsning for disse situasjonene.[40]

etter kollisjoner foregår langs grensen til objekter. Et annet problem er situasjoner hvor orienteringen til kollisjonsområdet ikke har noen mening, det vil si det er ikke definert en løsning for orienteringen, slik illustrert i figur 3.21.

Volino og Thalmann vil med [40] overkomme disse problemene ved å fokusere på å minimere konturen av kollisjonen istedet for å bruke tid på å identifisere kollisjonsområder som i [3].

Modellering
Modellen er bygd opp som et nettverk av polygoner som er flate. Dersom nettverket består av triangler er dette alltid tilfelle. Nærhet mellom flater oppdages ved å sjekke noder mot polygoner, mens kollisjoner ofte skjer mellom kanter og polygoner.

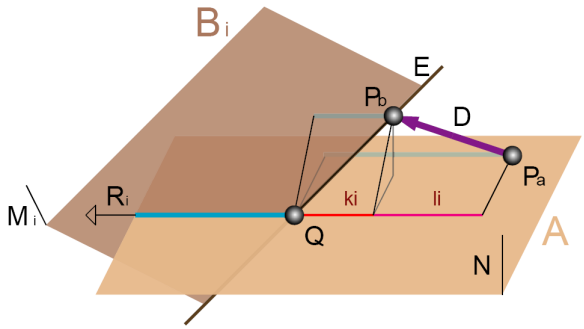
Simulering
Metoden kan tilpasses de fleste teknikker for tidintegrasjon, kollisjonsdeteksjon og -respons, og det presenteres derfor ikke en forslag til hvilke teknikker som bør benyttes.

Kollisjonsdeteksjon og Feiloppretting
En kant, E, grenser til ett eller flere polygoner, B_i. Når denne kanten penetrerer et polygon A oppstår en situasjon lik den i figur 3.22. D er polygon Bs relative forflytning i forhold til A. Gjennom å ta kryssproduktet av normalene til polygon A og B_i, henholdsvis N og M_i, beregnes vektor R_i som representerer retningen til kollisjonen mellom de to polygonene.

Ut ifra geometrien til kollisjonen kan følgende ligninger utredes:

$$l_i + k_i = \bar{P_a Q} \cdot R_i = D \cdot R_i + \bar{P_b Q} \cdot R_i = D \cdot R_i - \frac{N \cdot D}{N \cdot E} E \cdot R_i \tag{3.23}$$

$$k_i = \bar{P_b Q} \cdot R_i = -\frac{N \cdot D}{N \cdot E} E \cdot R_i \tag{3.24}$$



Figur 3.22: Kanten E beveger seg lengden D i forhold til polygon A.[40]

Ved å legge sammen alle endringene polygonene B_i bidrar med beregnes en vektor, se ligning 3.25 under, som uttrykker hvilken retning E må flyttes for å oppnå best mulig reduksjon av konturlengden til kollisjonen.

$$\sum(l_i + 2k_i)G \cdot D \quad der \quad G = \sum(R_i - 2\frac{E \cdot R_i}{E \cdot N}N) \tag{3.25}$$

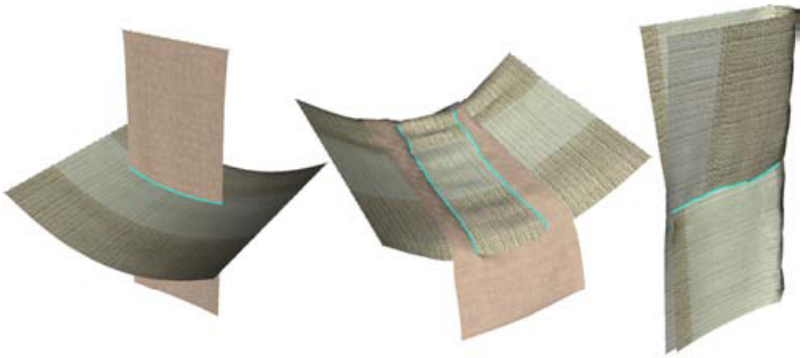
Produktet $G \cdot D$ gir lengden av kollisjonskonturen og G angir retningen kanten E må bevege seg i forhold til A for å redusere konturlengen mest mulig.

Ettersom det ikke er noen antagelser om hvor mange polygoner som er naboer til kanten E, kan denne teknikken benyttes selv i situasjoner hvor kompliserte sømmer er tilstede på tekstilet.

Metoden er generell og enkel, og den kan kombineres med de fleste vanlige teknikker for kollisjonsdeteksjon og -respons. Dette fører til at måling av hvor effektiv den er, avhenger veldig av implementasjonen av simuleringen. Simuleringsteknikkene som er valgt for å teste metoden illustrerer styrker og svakheter.

De ekstra beregningskostnadene ved å implementere denne metoden er nesten ikke merkbare for simulering ved hjelp av "standard" kollisjonshåndteringsteknikker. Videre har det vist seg at det er mest effektivt å løse kun et begrenset antall kollisjoner per tidssteg, istedet for å forsøke å løse alle kollisjoner. Denne teknikken har i forsøket vist at den kan løse opp kollisjoner bare få tidssteg etter at de oppstår og kan på denne måten forsikre at ingen kollisjoner er uløst ved slutten av simuleringen.

Figur 3.23 viser tre tilfeller hvor denne metoden ikke klarer å løse opp kollisjonene. Volino og Thalmann foreslår at bruk av teknikken til Baraff et al [3] kan benyttes for å løse disse situasjonene, men dette vil kreve at kollisjonsområdene er svært godt definert noe som vil være svært dyrt.



Figur 3.23: Tre situasjoner hvor reduksjon av konturlengder ikke fungerer [40].

3.4.3 Oppsummering

Ettersom det er et kjent problem at kollisjonshåndteringsteknikker har problemer med å unngå at kollisjoner oppstår, har en gruppe teknikker som har som mål å korrigere slike feil oppstått. Når det er oppdaget feil i form av kollisjoner som ikke skulle ha oppstått benyttes disse teknikkene underveis i animasjonen for å rette opp feilene. Enten ved hjelp av å definere kollisjonsområder som må separeres eller ved å redusere lengden på kollisjonskonturkurven.

3.5 Annet relevant arbeid

Artiklene som er gjennomgått til nå representerer kun et utvalg av interessant stoff. Kollisjonshåndtering har blitt behandlet på mange forskjellige måter med varierende suksess opp igjennom årene.

Volino og N. Magnenat-Thalmann har gjennom en rekke artikler [35] [36] [37] [38] [39] [40] utviklet løsningen de presenterer i [40]. Arbeidet har for det meste sett på hvordan hierarkier sammen med kollisjonskorreksjon kan effektivisere kollisjonhåndtering.

I [35] foreslår de å løse kollisjon og selv-kollisjon gjennom en algoritme som lager et hierarki av en polynomisk overflatemodell og hovedideen er at de utnytter regulariteten i regioner for å kunne hoppe over områder under selv-kollisjonstestingen. I [36] samarbeider de med M. Courchesne om å utvikle teknikker for å finne orienteringen av kollisjonen for så å korrigere eventuelle feil som kan ha oppstått. Stikkord her er historiebaserte proksimitetssøk og kollisjonskorreksjon. [37] ser på bruk av et hierarki som er bygd fra bunnen opp imotsetning til [29] som bygger hierarkiet fra

toppen og ned. Kollisjonsrespons som korrigerer posisjoner, hastigheter og aksellerasjoner blir presentert i [38]. [39] konsenterer seg om å finne en best mulig kombinasjon av teknikker, blant annet blir bruk av partikkelsystemer diskutert. Elementer fra de fleste foregående artikler blir benyttet for å simulere systemet i [40].

I [28] presenterer Provot en modell som er basert på et masse-fjær system som brukes til å modellere deformerende objekter som for eksempel flagg. Han fokuserer på å tilføre begrensninger til modellen for å unngå det han kaller "super-elastikk". Disse begrensningene blir tilført fjærene ved bruk av en dynamisk invers prosedyre. Kollisjonshåndtering blir først presentert i [29].

I [19] utvides Provots definisjon av et kollisjonsområde fra [29] til å inkludere hele området hvor en kollisjon inntreffer, også kollisjoner med andre objekter og selv-kollisjon. Metoden dere løser kollisjoner som oppstår samtidig uten å innføre nye kollisjoner mens denne prosessen er i gang. Bevaring av momentum er et viktig punkt.

W.R. Wong og G. Baciu [41] benytter en litt annen fremgangsmåte for å finne kollisjoner. De benytter en markeringsteknikk som skal hjelpe til å filtrere kollisjonspar. Et lag med filter finner kolliderende par og sender disse videre til neste steg som tar seg av kollisjonsinformasjonen. Alle par som finnes av filteret blir kodet av *sipCode* som sier noe om forholdet mellom elementene.

N. Pelechano, L. Bull og M. Slater[26] ser på bruken av orienterte innkapslede volum hierarkier i forbindelse med et sanntidssystem som har som mål å kunne simulere mennesker som skal vises på tv. Prosjektet heter PROMETHEUS og har som mål at skuespillere skal styre bevegelsene til de simulerte menneskene. Visuell god simulering av klær på menneskene er et sentralt tema.

Artikkelen av A. Fuhrmann, C. Gross og V. Luckas [13] fokuserer også på teknikker som skal kunne simulere i sanntid. Deres metode går under kategorien approksimerende metoder da beregningstunge teknikker som er fysisk korrekt ikke passer seg for sanntidssytemer. En av forenklingene de gjør er å neglisjere bøyningskreftene og de anisotropiske strekkreftene. Også her benyttes et hierarki for å optimalisere søk etter kollisjoner, og dette hierarkiet blir oppdatert ved hvert tidssteg.

For systemer som benytter en B-rep representasjon for å strukturere dataene, foreslår G. Tan, G. Li, D. Chen og X. Cui [31] å generere normalvektorer for hver vektor tilhørende et polynom for så å finne hvor disse møtes. Ut ifra dette kan de beregne hvorvidt kollisjoner kommer til å oppstå.

I følge Teschner et al [33] finnes det også metoder som faller under andre kategorier, som for eksempel stokastiske metoder, romlig oppdeling og bilde-rom teknikker. Stokastiske metoder motiveres av observasjoner som at polygonnettverk er en approksimasjon av virkelig geometri og at mennesker ikke kan skille mellom fysisk korrekt og fysisk mulig oppførsel.

J. Klein og G. Zachmann presenterer en stokastisk metode i [21] hvor sannsynlighet for å approksimere kollisjoner ut ifra et gitt kvalitetskriterie. $Pr(A_i, B_i)$, som er sannsynligheten for at $_i$ og B_i kolliderer, beregnes for nodene i et hierarki og kvaliteten på deteksjonen av kollisjoner kan kontrolleres gjennom en variabel. De bemerker at teknikken deres er beregnet på tidskritiske systemer og at hvilken som helt IVH kan benyttes.

I [32] benyttes romlig oppdeling på et uniformt nettverk for å oppdage kollisjoner. Istedet for å bruke datastrukturer som oktaltre eller BSP trær, benyttes en hash-funksjon til å mappe geometrien inn i en tabell. Kollisjonsdeteksjonen foregår i tre trinn. Først mappes informasjonen om objektet inn i en hashtabell. Neste steg søker igjennom tabellen og tredje steg sjekker om kollisjoner har oppstått. Algoritmen er utviklet for bruk i sanntidssystemer.

T. Vassilev, B. Spanlang, og Y. Chrysanthou [34] presenterer en teknikk for animere tekstiler og klær som har som mål å kunne implementeres i et system som skal fungere over internett. Brukeren skal kunne laste opp en virtuell versjon av kroppen sin for å kunne prøve og eventuelt kjøpe klær. De videreutvikler Provots [28] masse-fjær modell ved å løse problemet med superelastisitet. Denne metoden benytter seg av bilde-rom teknikken, disse teknikkene bruker GPU[4] for å rendre scenen og kollisjonstesting kan derfor reduseres fra et 3D problem til et 2,5D problem. Dette gjøres ved at dybdekartet sjekkes for å finne kollisjoner. Maskinvarebaserte metoder har den fordel at de ofte er veldig raske.

N.K. Govindaraju, S. Redon, M.C. Lin, og D. Manocha [16] presenterer også en bilde-rom teknikk. Her benyttes GPU for å søke etter kollisjoner og punkter som er på kollisjonskurs. Hovedideen bak teknikken deres er å kalkulere et "mulig kolliderende sett" av objekter som deretter utsettes for tester som sjekker mer nøyaktig for kollisjoner. Disse testene jobber seg nedover i et hierarki. Hierarkiet består av dette settet som inneholder objekter og subgrupper med dere representative elementer. I [17] utvides teknikken av N. K. Govindaraju, D. Knott, N. Jain, I. Kabul, R. Tamstorf, R. Gayle, M. C. Lin og D. Manocha med å tilføre metoder for å oppdage selvkollisjoner.

[4]Graphic Processing Unit

3.6 Oppsummering

I dette kapittelet har metoder for kollisjonsdeteksjon og -respons som kan brukes i forbindelse med animering av tekstiler og klær. Bruk av innkapslede volum hierarkier og distansefelt har vist seg å kunne gi gode resultater. I tillegg er det blitt gjennomgått teknikker for å korrigere eventuelle kollisjoner som fører til at tekstiler fester seg unaturlig til seg selv eller andre objekter. Til slutt er det nevnt en rekke artikler som er aktuelle for dette temaet, men som ikke ble viet spesiell oppmerksomhet i denne studien.

Kapittel 4

Konklusjon

Dette kapittelet diskuterer og evaluerer de teknikkene og metodene som ble gjennomgått i kapittel 3.

Kapittelet er delt opp i tre underkapitler. Første underkapittel nevner noen viktige aktører, og diskuterer og evaluerer metodene som ble presentert i kapittel 3 i forhold til bruksområder og relevans. Andre underkapittel konkluderer rapportens arbeid og til slutt oppsummeres kapittelets innhold.

4.1 Evaluering og diskusjon

Simulering av tekstiler og klær er et svært interessant område innen grafikk i dag, dette blant annet på grunn av de mange og store markedskreftene som har fått øynene opp for viktigheten av å kunne simulere tekstiler og klær troverdig.

4.1.1 Sentrale aktører

I tillegg til en rekke enkeltpersoner, universitetet og lignende, har noen større aktører engasjert seg innen feltet. Disse representerer markedskreftene og kan antyde utviklingen i feltet.

Innenfor utvikling av grafiske applikasjoner som involverer tekstiler og klær er MiraLab [23] en stor aktør og bidragsyter. Deres teknikker har blitt brukt i et stort spekter av applikasjoner, blant annet innen tekstildesign applikasjoner [33]. MiraLab ble grunnlagt i 1989 av professor Nadia Magnenat-Thalmann som har gitt ut en lang rekke aktuelle artikler innen

simulering av tekstiler og klær [24] [33] [35] [36] [37] [38] [39] [40], hvorav mange av disse i samarbeid med Pascal Volino.

En annen stor aktør som er med på å presse frem utviklingen er Pixar. De har vært ledende innen nytenking og utvikling av metoder for å simulere grafikk for filmer i en lengre periode [27]. Blant annet så er [3] gitt ut gjennom Pixar.

4.1.2 Diskusjon

Utvikling av nye metoder og teknikker har ikke skjedd over natten. En grundig forståelse av feltet oppnåes først ved å se på gamle så vel som nye teknikker. Utviklingen fra de gamle til de nye teknikkene viser en tankerekke som gjenspeiler markedets behov, en slik forståelse er nødvendig for å skjønne hvilke problemer som bør løses videre.

De resultatene som har blitt oppnådd fungerer ofte svært godt for den nisjen de er designet for, men flere av områdene deler ikke kriterier og felles løsninger blir derfor vanskelig å finne. For eksempel vil produsentene av filmer vektlegge nøyaktighet og visuell realisme fremfor muligheter for sanntidsimulering, mens produsenten av et designer verktøy kan tillate seg å ofre akkurat disse egenskapene i bytte mot teknikker som fungerer i sanntid. På grunn av disse ulikhetene i designmål vil det ikke finnes en løsning som fungerer optimalt i alle situasjoner [33].

I stedet for å måle teknikker opp mot hverandre kan deres styrker og svakheter sees i sammenheng med forskjellige situasjoner de kan bli brukt i.

Innkapslede volum hierarkie (IVH)
IVH har vist seg å være svært effektive i mange sammenhenger. I slike situasjoner er det viktig å velge riktig type volum og teknikk for oppdatering eller rekonstruksjon da disse faktorene er av stor betydning for beregningskostnadene. AABB hierarkier har vist seg å være et godt valg for deformerende objekter ettersom de effektivt lar seg oppdatere eller rekonstruere, i tillegg har det vist seg at trær med 4 eller 8 noder per forelder har vært mer effektive enn binære trær [33]. Slike hierarkier kan benyttes i både i systemer som krever sanntidsimulering og ikke.

Bridson et al [7] bruker IVH til å simulere et system som rendrer, mens N. Pelechano, L. Bull og M. Slater [26] og A. Fuhrmann, C. Gross og V. Luckas [13] har utviklet sine teknikker til å fungere i sanntidssystemer, hovedsaklig er forskjellen at de kutter ned på visuell nøyaktighet for å oppnå dette. Responsbehandling er også enklere i sanntidssystemen. For eksempel [13]

behandler respons kun ved å flytte det kolliderende punktet, mens [7] benytter flere teknikker som skal forsikre at momentum blir bevart.

Distansefelt
Distansefelt muliggjør veldig rask evalurering av normaler og distanse som brukes til kollisjonsdeteksjon og -respons [33]. På grunn av informasjonen som er lagret om penetrasjonsdybde og normaler i et distansefelt egner de seg spesielt godt til å behandle kollisjoner mellom rigide og deformerende objekter.

Dessverre er det slik at konstruksjonen av distansefelt er relativt tregt og egner seg derfor ikke til sanntidssytemer hvor interaktivitet er viktig. Dette fordi beregningskostnadene ved å rekonstruere feltet hver gang det deformerende objektet forandrer form er for krevende. Distansefelt egener seg best til applikasjoner hvor det er behov for å balansere nøyaktighet og effektivitet. Enkel tilgang til penetrasjonsdybde og normaler gjør kollisjonrespons mer effektivt, dette faktum benyttes både av B. Lafleur, N. Magnenat-Thalmann and D. Thalmann [24] og R. Bridson, S. Marino and R. Fedkiw [8].

Stokastiske metoder
En av fordelene med stokastiske metoder er at har mulighet til å kontrollere nøyaktig kollisjonsdeteksjon mot beregningskostnader. Det har blitt vist at metoder, som for eksempel vist av J. Klein og G. Zachmann [21], kan benyttes til å simulere tekstiler og klær i sanntidssystemer. De bruker IVH i tillegg til metoder for å beregne sannsynligheten for at kollisjonerspar oppstår. Metoden egner seg ikke spesielt godt til applikasjoner som krever høy fysisk korrekthet, men ved å benytte robuste kollisjonsdeteksjon og -respons teknikker kan det tenkes å bruke den til filmer.

Romlig oppdeling
Ettersom teknikker for romlig oppdeling ikke begrenser hvilken underliggende modell som kan benyttes, kan disse metodene tilpasses de fleste behov [33]. Fysisk korrekte metoder vil kanskje benytte FEM, mens sanntidssystemer vil bruke masse-fjær nettverk. M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets og M. Gross [32] har designet et sanntidssystem som bruker denne teknikken på en modell av tetraeder, men sier samtidig at ved å skifte ut kollisjonstesten så vil andre primitiver også fungere.

Bilde-rom
Så langt har alle teknikker presentert krevd preprosesseringssteg, det er ikke nødvendig med bilde-rom metoder [33]. Ettersom deformerende objekter krever at datastrukturen i preprosesseringssteget oppdateres eller rekonstrueres for hvert tidssteg vil det være effektivt å kunne kutte ut dette steget. Ulempen med bilde-rom metoder er at de jobber med diskrete modeller, som innebærer at nøyaktigheten begrenses av hvor detaljert

modellen er. En annen ulempe med bilde-rom teknikker er at de gir lite informasjon som kan brukes i kollisjonshåndtering applikasjoner som krever fysisk korrekt oppførsel. T. Vassilev, B. Spanlang og Y.Chrysanthou [34] foretar kollisjonsdeteksjon i forbindelse med denne teknikken gjennom å danne dybdekart, beregne normaler og hastigheter for så å sjekke om kollisjoner er tilstede. De løser responsproblemet ved kun å forandre hastighetene til objektene involvert.

4.1.3 Evaluering

Gjennom dette underkapittelet skal jeg forsøke å skjematisere mine funn.

Som tidligere nevnt er det vanskelig å sammenligne metoder og teknikker ettersom mange av dem ikke er konstruert for samme formål. Likevel er det mulig å evaluere de opp mot bruksområder.

Tabell 4.1 forsøker å oppsummere i hvilke områder de forskjellige teknikkene fra kapittel 3 har sine styrker og svakheter.

Metodene blir vurdert på en skala fra 1-5, hvor 1 er "lite egnet" og 5 er "meget godt egnet", på grunnlag av den foregående diskusjonen. Sanntid referer til bruk i systemer hvor simulering i sanntid er viktig, film kategorien vektlegger visuell nøyaktighet og design(CAD)[1] trenger algoritmer som er så nært fysisk korrekt som mulig.[2]

Under følger en liste som forklarer forkortelsene og vurderingssystemet brukt i tabell 4.1.

Innkapslede Volum Hierarkier(IVH), kapittel 3.2:

- Moore og Wilhelms: Collision detection and response for computer animation [25]
- Provot: Collision and self-collision handling in cloth model dedicated to design garments [29]
- Bridson et al 2002: Robust Treatment of Collisions, Contact and Friction for Cloth Animation [7]

Distansefelt, kapittel 3.3:

- Lafleur et al: Cloth Animation with Self-Collision Detection [24]

[1] Computer Aided Design

[2] Det er ikke hensikten å forsøke å bestemme hva forskjellige applikasjoner har av behov da dette kan variere. Hensikten med kategorinavnene er å gi konkrete eksempler på situasjoner hvor teknikkene er nyttige.

- Bridson et al 2003: Simulation of Clothing with Folds and Wrinkles [8]

Teknikker for å korrigere feil, kapittel 3.4:

- Baraff et al: Untangling Cloth [3]
- Volino og Magnenat-Thalmann: Resolving Surface Collisions through Intersection Contour Minimization [40]

	Sanntid	Film	Design (CAD)
Moore og Wilhelms	1	2	1
Provot	1	3	3
Bridson et al 2002	1	4	3
Lafleur et al	1	3	2
Bridson et al 2003	1	5	4
Baraff et al	3	5	4
Volino og Magnenat-Thalmann	1	4	5

Tabell 4.1: Teknikkene vurderes opp mot mulige bruksområder.

Tabellen viser at, av de metodene som ble grundig gjennomgått i kapittel 3, vil D. Baraff, A. Witkin, og M. Kass [3] metode egner seg best til sanntidssystemer. Baraff et al sammen med R. Bridson, S. Marino og R. Fedkiw [8] har løsninger som vil fungere godt i filmproduksjon. P. Volino og N. Magnenat-Thalmann [40] har gode forslag til CAD verktøy.

Hvor godt disse observasjonene stemmer med virkeligheten avhenger av systemene de blir satt inn i. En rask teknikk satt inn i et tregt system vil ikke nødvendigvis gi mulighet for sanntidssimulering for eksempel.

4.2 Konklusjon

Arbeidet med denne rapporten har ført til kunnskap om en lang rekke metoder og teknikker som kan brukes i forbindelse med animasjon og simulering av tekstiler og klær. Metodene og teknikkene er ikke spesifisert til å kunne fungere på en bestemt type systemer. Dersom et system skal designes bør de styrker og svakheter som ble nevnt i kapittel 4.1 vurderes opp mot kriteriene som blir satt.

Det er ikke sannsynlig at det finnes en generell løsning for simulering av tekstiler[33] og det er heller ikke sannsynlig at det er ønskelig med tanke på de store forskjellene i kriterier ulike systemer finnes. De spesialiserte løsningene har gitt mange gode teknikker og området er fortsatt i utvikling.

Metoder basert på IVH er populære, men andre teknikker utvikles rask og skaper interesse i feltet.

4.3 Oppsummering

Dette kapittelet har oppsummert og diskutert de funnene som ble gjort i kapittel 3. Metodene sett på i forhold til mulige bruksområder og styrker og svakheter ble påpekt. En tabell illustrerer nyttigheten av teknikkene i forhold til mulige bruksområder i tråd med diskusjonen. Til slutt konkluderes arbeidet i rapporten.

Bibliografi

[1] A Al-khalifah og D Roberts (2004) Survey of modelling approaches for medical simulators. The International Conference Series on Disability Virtual Reality and Associated Technologies, 2004

[2] D. Baraff og A. Witkin (1998) Large steps in cloth simulation. Computer Graphics (Proc. SIGGRAPH), 43-54.

[3] D. Baraff, A. Witkin, og M. Kass (2003) Untangling cloth. ACM Trans. Graph. (SIGGRAPH Proc.), 22

[4] Gino Van Den Bergen (2004) Collision Detection in Interactive 3D Environments. Morgan Kaufmann publications.

[5] Ronen Barzel, John F. Hughes, og Daniel N. Wood. (1996) Plausible Motion Simulation for Computer Graphics Animation. In Computer Animation and Simulation '96, Proceedings of the Eurographics Workshop, pages 184–197, Poitiers, France.

[6] D.E. Breen, D.H. House, M.J. Wozny (1994) Predicting the Drape of Woven Cloth Using Interacting Particles, Proc. SIGGRAPH'94, Addison-Wesley, pp 365-372

[7] R. Bridson, R. Fedkiw og J. Anderson (2002) Robust treatment of collisions, contact and friction for cloth animation, Proc. of SIGGRAPH'02, San Antonio, Texas, pp. 594603.

[8] R. Bridson, S. Marino og R. Fedkiw (2003) Simulation of Clothing with Folds and Wrinkles, ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA), edited by D. Breen og M. Lin, pp. 28-36.

[9] Kwang-Jin Choi og Hyeong-Seok Ko (2002) Stable but responsive cloth. In Proceedings of ACM SIGGRAPH ´02, pages 604-611. ACM Press.

[10] Christer Erison (2005) Real-Time Collision Detection Morgan Kaufmann publications.

[11] Olaf Etzmuß, Bernhard Eberhardt, Michael Hauth og Wolfgang Straßer (2000) Collision Adaptive Particle Systems, hentet fra citeseer.ist.psu.edu/656971.html 24. april 2007

[12] L.De Floriani, L. Kobbelt og E.Puppo (2004) A Survey on Data Structures for Level-Of-Detail Models. Advances in Multiresolution for Geometric modelling series in Mathematics and Visualisation.

[13] A. Fuhrmann, C. Gross og V. Luckas (2003) Interactive Animation of Cloth Including Self Collision Detection, Proc. of WSCG'03, University of West Bohemia, Czech Republic, pp. 141-148.

[14] Sarah F. F. Gibson og Brian Mirtich (1997) A Survey of Deformable Modeling in Computer Graphics. Mitsubishi Electric Research Laboratories. Technical reports 1997-19.

[15] Sarah F. F. Gibson (1997), 3D ChainMail: a Fast Algorithm for Deforming Volumetric Objects, Proc. 1997 Symposium on Interactive 3D Graphics

[16] N.K. Govindaraju, S. Redon, M.C. Lin, og D. Manocha (2003) CULLIDE: Interactive Collision Detection between Complex Models in Large Environments Using Graphics Hardware, Proc. Graphics Hardware Conf., pp. 25-32

[17] Naga K. Govindaraju, David Knott, Nitin Jain, Ilknur Kabul, Rasmus Tamstorf, Russell Gayle, Ming C. Lin og Dinesh Manocha (2005) Interactive Collision Detection between Deformable Models using Chromatic Decomposition. ACM Transactions on Graphics (TOG), v.24 n.3, July 2005

[18] Donald Hearn og M. Pauline Baker (2004) Computer Graphics with OpenGL, third edition.

[19] Suejung Huh, Dimitris N. Metaxas and Norman I. Badler (2001) Collision Resolutions in Cloth Simulation. In Proceedings of Computer Animation, pages 122-127

[20] KAWABATA Laboratory. Hentet 25. mars 2007, fra http://www.kod.vslib.cz/laboratore/Kes/index_eng.html

[21] J. Klein og G. Zachmann (2003) Adb-trees: Controlling the error of time-critical collision detection. In 8th International Fall Workshop Vision, Modeling, and Visualization (VMV) (University Mnchen, Germany, Nov.19–21 2003). Hentet fra http://citeseer.ist.psu.edu/klein03adbtrees.html 25. april 2007

[22] William D. MacMillan (1936) Dynamics of Rigid Bodies. Dover Publications, Inc, New York.

[23] MiraLab. Hentet 25. april 2007 fra http://www.miralab.unige.ch/

[24] Benoit Lafleur, Nadia Magnenat Thalmann og Daniel Thalmann (1991) Cloth Animation with Self-Collision Detection. IFIP conference on modeling in computer graphics proceedings, p. 179-97

[25] Matthew Moore og Jane Wilhelms (1988) Collision Detection and Response for Computer Animation (Proceedings of SIGGRAPH 88) Annual Conference Series. ACM SIGGRAPH, 289-298.

[26] N. Pelechano, L. Bull og M. Slater (2003) Fast Collision Detection Between Cloth and a Deformable Human Body. Technical report, University College London, hentet fra: http://www.cs.ucl.ac.uk/research/vr/Projects/Prometheus/ Report/paperCollDet02.pdf, 27. mars 2007

[27] Pixar Animation Studio. Hentet 25. april 2007 fra http://www.pixar.com/companyinfo/about_us/overview.htm

[28] X. Provot. (1995) Deformation constrains in a mass-spring model to describe rigid cloth behavior. In Graphics Interface '95, Québec Canada, 17-19 mai 1995

[29] X. Provot. (1997) Collision and self collision handling in cloth model dedicated to design garments. In Computer Animation and Simulation '97, pages 177–189.

[30] K. Sundaraj (2004) Real-Time Dynamic Simulation And 3d Interaction Of Biological Tissue: Application To Medical Simulators, Hentet 23. april 2007 fra citeseer.ist.psu.edu/sundaraj04realtime.html

[31] G. Tan, G. Li, D. Chen og X. Cui (1999) Detection of Collisiion and Interference between 3D Objects Using Normal Vector. Vehicle Electronics Conference, 1999. (IVEC '99) Proceedings of the IEEE International, p. 286-289 vol.1

[32] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets og M. Gross (2003) Optimized spatial hashing for collision detection of deformable objects pp. 47–54. Hentet fra http://citeseer.ist.psu.edu/teschner03optimized.html 25. april 2007

[33] M. Teschner, S. Kimmerle, G.Zachmann, B. Heidelberger, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnetat-Thalmann og W. Strasser (2005) Collision detection for deformable objects. Computer Graphics Forum 24(1)

[34] T. Vassilev, B. Spanlang og Y.Chrysanthou (2001) Fast cloth animation on walking avatars. Computer Graphics Forum (Proceedings of Eurographics'01). v20 i3. 260-267

[35] P. Volino og N. Magnenat-Thalmann (1994) Efficient self-collision detection on smooth dizcretized surface animations using geometrical shape regularity. Computer Graphics Forum (Proceedings of EuroGraphics 1994), 13(3), pp 155-166

[36] P. Volino, M. Courchesne og N. Magnenat-Thalmann (1995) Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects. SIGGRAPH '95 Conference Proceedings, pp. 137–144.

[37] P. Volin og N. Magnenat-Thalmann (1995) Collision ans Self-Collision Detection: Effecient and Robust Solutions for Highly Deformable Surfaces, In Comp, Animation and Simulation, Springer Verlag, pp 55-65

[38] P. Volin og N. Magnenat-Thalmann (2000) Accurate Collision respons on polygonal Meshes. Computer Animation 2000, IEEE Computer Society, pp 154-163

[39] P. Volin og N. Magnenat-Thalmann (2005) Accurate Garment Prototyping and Simulation, Computer-Aided Design and Applications, CAD Solutions, 2(5), pp 645-654

[40] P. Volino og N. Magnenat-Thalmann (2006) Resolving Surface Collisions through Intersection Contour Minimization. ACM Transactions on Graphics (SIGGRAPH 2006 proceedings), ACM Press, 25(3), pp. 1154-1159.

[41] W.R. Wong og G. Baciu (2006) A randomized marking scheme for continuous collision detection in simulation of deformable surfaces. In Proceedings of the 2006 ACM international Conference on Virtual Reality Continuum and Its Applications