NTNU
Norwegian University of
Science and Technology

An Easier Predictive Display Based on Image Transformation for Low Cost Teleoperation of Vehicles With Time Delay

Martin Løland

Master of Science in Mechanical Engineering
Submission date: June 2018
Supervisor: Martin Steinert, MTP

Norwegian University of Science and Technology
Department of Mechanical and Industrial Engineering

Abstract

Teleoperation of remotely operated vehicles has become an increasingly viable solution in many fields as technology has improved and the requirements for risk and cost reduction has increased. When operating vehicles, especially at long distances, unwanted latency is introduced to the system. As a result, cognitive workload increase and performance is degraded. Predictive technology has proven to be an effective method to reduce these effects. But many of the current implementations rely on expensive equipment or extensive knowledge of the robotic system.

A new type of predictive display based on image transformation has been developed as part of this thesis. It does not require any additional hardware and can be implemented on a wide range of vehicles without much configuration. This thesis aimed to investigate H1: a simple predictor display based on image transformation can increase the operator performance. And H2: a simple predictor display based on image transformation will decrease the operator's subjective workload.

An experiment was performed where the 58 participants were given a modified "peg-in-hole" task. During a test time of 90 seconds the subjects had to move the vehicle and score as many hits as possible. This was performed using three different conditions. Condition one using a 750ms delay, condition two having a 750ms delay with predictor screen and condition three with a 250ms long delay but no predictive screen.

The results showed that participants performed on average 20.6% better on condition two with the predictive display versus condition one with no predictive display. The results also showed that particpants who play games weekly or more, got almost twice the benefit from the predictive display. Gamers had a 30.13% increase while non-gamers only gained a 16.91% performance increase. The participants reported no statistical difference in their mental, physical and temporal demand. The predictive display did therefore not reduce the subjective workload.

Sammendrag

Fjernstyring av roboter har blitt et stadig mer populært alternativ til tradisjonelle operasjoner etter hvert som teknologien har blitt tilgjengelig og kravene til sikkerhet og økonomistyring har økt. Når roboter fjernstyres, spesielt fra lange distanser, oppstår det uønsket tidsforsinkelse i systemet. Som et resultat øker den kognitive påkjennelsen og operasjonseffektiviteten synker. Prediktiv teknologi har vist seg å være et bra alternativ for å minske de negative effektene. Mange av de nåværende løsningene har dog krav til avansert utstyr eller omfattende informasjon om roboten.

En ny type prediktivt grensesnitt basert på forskyvning og skalering av video har blitt utviklet. Dette grensesnittet krever ikke ekstra utstyr og kan anvendes på en rekke forskjellige robotkonfigurasjoner. Denne masteroppgaven ønsket å undersøke følgende påstander. H1: et prediktivt grensesnitt basert på forskyvning og skalering av video kan øke operasjonseffektiviteten og H2: et prediktivt grensesnitt basert på forskyvning og skalering av video vil senke den subjektive kognitive påkjennelsen.

Et eksperiment ble utført hvor 58 deltakere ble gitt en oppgave hvor de måtte styre en robot inn i en rekke hull i løpet av 90 sekunder. Denne testen ble gjennomført under tre forskjellige betingelser. Første inneholdt en tidsforsinkelse på 750ms, den andre inneholdt den samme tidsforsinkelsen, men med det prediktive grensesnittet. Den siste betingelsen hadde en forsinkelse på 250ms og ingen ekstra hjelp.

Resultatene viste at deltakerne utførte oppgaven 20.6% bedre under betingelsen som inneholdt det prediktive grensesnittet kontra den samme tidsforsinkelsen uten prediktivt grensesnitt. Resultatene viste også at personer som spiller videospill på en ukentlig basis gjorde det bedre enn resten. De hadde en positiv økning på 30.13%, mens resten av deltakerne oppnådde 16.91%. Deltakerne rapporterte ingen statistisk forskjell når det kom til fysisk, mental eller stressende påkjenning. Det prediktive grensesnittet hadde derfor ingen reduksjon på den subjektive kognitive påkjennelsen.

Preface

This document represents the final dissertation of Martin Løland in connection with the master thesis written in the spring of 2018 at the Norwegian University of Science and Technology, Department of Mechanical and Industrial Engineering.

eduROV is a project started in Trondheim which aims to create an affordable and open-source remotely operated vehicle for use by students and hobbyists. During the spring of 2018 work was done on this project as part of this dissertation. In addition, it became evident that there was a lack of solutions for predictive displays that would suit the open source project of eduROV.

During the semester a new python package for robot control and video feed was developed. In addition, a simple predictive display that can be applied to many remotely operated vehicles was created and tested. This was the first encounter with an experiment including people. It provided many interesting challenges concerning experiment design and statistical analysis.

I would like to thank PhD Candidate Kristoffer B Slåttsveen for his feedback on the development of the python package and thesis. I also want to thank my supervisor Professor Martin Steinert who has been very helpful with pointing me in the right direction for interesting research topics. Lastly, PhD Candidate Achim Gerstenberg has provided helpful information about statistical analysis and robot experiments.

Martin Løland

Trondheim 10.06.2018

Table of Contents

List of Figures

List of Tables

1 Introduction and Theory

The eduROV project started as an idea at Trondheim Maker Faire in 2014. From there on it has been developed into a "functioning Open-Source ROV project at a new level of affordability."[1]. The project is now managed by Norwegian University of Science and Technology (NTNU) and in specific the *engage* project by Centre for Engaged Education through Entrepreneurship.

I, the author of this thesis, joined the project in December 2017 to improve the software. My objective was to decrease the video latency which had a negative effect on the user experience. In addition, more objectives such as increased functionality surfaced during the development period.

It also became interesting to look at how video latency effects user performance and what previous research in the field has done to compensate for it. Predictive displays arose as the most popular method. As many of the predictive algorithms rely on advanced hardware, none were found to be applicable to the open source eduROV project.

A new type of predictive display based on image transformation was therefore created. In addition to the eduROV software, this thesis will present the developed predictive display, the experiment and its results.

1.1 Thesis Structure

Chapter 1: The current chapter will present the applications and challenges related to teleoperation. In addition, it included an introduction to the most popular methods in predictive technology.

Chapter 2: Dedicated to the development of the *eduROV* python package. For those only concerned with predictive technology and relevant research, this chapter can be omitted.

[1] `https://www.edurov.no/`

Chapter 3: Presents the developed predictor display based on image transformation.

Chapter 4: Describes experimental design and procedure of the experiment used to test the developed predictive display.

Chapter 5: Results and discussion, will present the findings of the performed experiment and its discussion.

Chapter 6: Conclusion and summary with respect to the hypotheses.

1.2 Teleoperation

Teleoperation, the operation of controlling vehicles from a remote location, has gained popularity since it first became possible. This includes underwater, ground, aerial and space vehicles. The controlled vehcicle is referred to as a *remotely operated vehicle* (ROV). The word *robot* will also be used interchangeably with ROV throughout this thesis. There are many locations and tasks where ROVs are useful. These includes places that are to risky for people, like post disaster areas, underwater operations, space, conflict zones etc. Other times, using humans is to costly or just impossible. Offshore maintenance and heavy duty mining are some of the tasks.

The focus of this thesis has been on *human-in-the-loop* teleoperation between ROV (slave) and remote human operator (master), by the means of video feedback. The operator views a video feed of the ROV in a remote environment and controls it by control input. This form of teleoperation can be effective because it is easy for the operator to understand and simple to implement. Other forms of teleoperation can be achieved by increasing the level of autonomy (LOA). In such situations, the human operator can be excluded from the control loop. By using other sensory input than camera feed, such as radars, a different kind of teleoperation is also attained. None of these will any focus in this thesis.

Although an unmanned ground vehicle (UGV) has been used in the experiment, the findings can be applied to teleoperation of all types of vehicles, that be aerial, ground, underwater and space. It does however not apply to situations where the camera is overlooking the environment from a fixed position while the robot is free to move. This configuration is often used in telemedicine (Kumcu et al., 2017) or robot arm manipulation (Bejczy et al., 1990).

1.2.1 Telepresence

Draper et al. (1998) defined telepresence as "the perception of presence within a physically remote or simulated site". He also stated that "telepresence is generally hypothesized to improve efficiency or reduce user workload" and that telepresence is beneficial to mission performance.

Chen et al. (2007) went through 150 papers and checked different teleoperation factors and how they influence user performance. They found eight main factors; Field of view (FOV), orientation, camera viewpoint, depth perception, video quality and frame rate (FR), time delay, and motion. FOV describes the amount of environment that is visible in the video. Orientation is the rotation of the robot in the environment, and can be difficult to perceive if there is a lack of known reference points. Camera viewpoint is often *egocentric* (robot view) or *exocentric* (birds view), which can lead to tunneling or loss of true ground view respectively. Lack of depth perception can cause wrong estimation of distances and video quality can reduce target identification. Time delay effects are very task dependent but often cause reduced driving performance. Motion describes the situation where the operator itself is moving and can cause motion sickness.

1.2.2 Time delay

Among the factors mentioned above, time delay or *latency* has been found to impose large impacts one teleoperation performance (Chen et al., 2007). Chen noted that latencies as low as $10 - 20$ ms can be detected by people. Arthur et al. (1993) found that latencies (ranging from 50 to 550 ms) to be a more important factor than frame rate (30, 15, or 10 fps) on human performance.

Time delay introduces a situation where the commands of the operator does not correspond to the visual feedback he or she is getting. Because of this, human drivers tend to over steer and oscillate with their *correcting* steering commands (Appelqvist et al., 2007). This increases the cognitive workload as the operator has to remember the input already given when giving new control commands (Matheson et al., 2013). Ricks et al. (2004) found that the mental load required to keep track of the robot pose adversely affects the operator's ability to effectively control the robot. A principle of reducing the workload is therefore to maintain correlation between commands issued by operator and changes in the interface (Nielsen et al., 2007).

Some of the research that has investigated the effect of video latency on human performance can be seen in Table 1.1. It shows the increase factor for different tasks and delay times. An increase factor of 1.40 is equal to a 40% increase in task completion time. The actual detrimental effect of latency is very task dependent. In the table, an increase factor of 1.5 can be found at 100ms for a needle-driving task, while for the robot car movement task the same factor was found at 2000ms. Some argue that task completion time increase linearly with delay time (Ando et al., 1999), (Lane et al., 2002). While others experience an exponential increase (Xu et al., 2014).

Table 1.1: Task completion time increase factor for different delays. N = number of participants.

Author	Task	N	Time delay [ms] and increase factor		
			100-300	400-700	800-1500
Fabrizio et al., 2000	Pin transfer	6	1.04-1.21*	1.17-1.41*	1.11-1.58*
Xu et al., 2014	Energy dissection	16	1.4-1.8	2.7-4.3	
Xu et al., 2014	Needle-driving	16	1.5-2.1	2.5-6.2	
Perez et al., 2016	Surgical simulator	37	0.75	1.5	
Lum et al., 2009	Block transfer	14	1.45	2.04	
MacKenzie et al., 1993	Target acquisition	8	1.64		

* Estimated from graph

The reasons for time delay in a teleoperation system can be many and is not the focus of this thesis. In general, the total latency is a result of software and hardware design as well as physical limitations and distance. Processing and transfer of commands from the master control to the slave ROV will contribute to the total time. As will the time it takes for the robot to capture and compress video frames, and sending it back to the operator for viewing. In this thesis the *total perceived delay* is of most interest. This is the total elapsed time from when the operator issues a command, until the robot can be seen moving on the screen.

1.2.3 Delay compensation

There are three main ways to combat the detrimental effects of time delay. First, an increased level of automation (LOA), the operator workload is reduced. The results of Luck et al. (2006) showed that the higher LOA, the better performance in terms of both time and number of errors made. In some cases, such as a communication blackout, autonomy is essential (Dorais et al., 1999). This option is not always available and may not even be possible, as it could require very advanced hardware and software, depending on the task. Goodrich et al. (2001) argued that adjustable autonomy could be used to increase the robot effectiveness. He also mentioned that a more autonomous robot is required when longer time delays are present. On the other hand, he also stated that "as autonomy level increases, the breadth of tasks that can be handled by a robot decreases".

Secondly, instead of increasing LOA, providing more information to the operator may increase situational awareness and therefore performance. Miller et al. (2005) performed an experiment where the operator was reminded of what commands had been given by providing them with a streaming command indicator. The preliminary results showed that the operator reported lower fatigue levels. But there are limitations to how much information an operator can digest in a finite amount of time. Chen et al. (2007) explained that overlaying information on video feed can potentially lead to cognitive tunneling.

Table 1.2: Predictive technology experiments with task time reduction. Ordered by date.

Author	Robot system Task	Predictor method Camera	Participants Delay	Task time reduction
Lu et al., 2018	Car simulator Driving	Model-free framework Simulated human	12 Not reported	8%
Hu et al., 2016	2-6 DOF manipulator Camera aligment	Simulated 3D scene Virtual	15 300, 500, 1000	33%, 58%, 65%*
Zheng et al., 2016	Car simulator Driving	Model-free framework Simulated human	5 900	35%
Lovi et al., 2010	Robot arm on Segway Object alignment	Vision-based monocular modeling At end effector	5 300	33%*
Matheson et al., 2013	Rover Driving	Projected field of view estimation Fixed to car	12 3000	48%-64%*
Rachmielowski et al., 2010	Virtual with Phantom OMNI Alignment	Reconstructed 3D environment At end effector	12 300	29%-30%*
Mathan et al., 1996	Lunar vehicle Manovuering	Superimposed directional information Fixed to car	8 5000	24%-30%
Bejczy et al., 1990	6DOF PUMA robot Tapping	Superimposed phantom robot Fixed	2 1000, 4000	13%-34%, 40%-56%

* Estimated from graph

Lastly, as a third option, there is the use of *predictive technology*. These are displays, control algorithms and graphical models that try to predict the future state of the ROV. They are based on the vehicle's current state and commands given by the operator. Predictive displays has proven to be the most promising solution, as Chen et al. (2007) concluded:

> If these delays cannot be engineered out of the system, it is suggested that predictive displays or other decision support be provided to the operator.

1.3 Predictive Technology

Table 1.2 shows a summary of some experiments that has been done in the field of predictive technology. The experiments span a wide variety of robot configurations, experiment tasks and predictive methods and can not necessary be compared directly.

The robot system can be roughly divided into two main groups, either the exact robot configuration is known or it is not. The former includes robot arm manipulators fixed to a defined reference frame where its configuration is a result of user input only. In the latter, the robot configuration is subjected to external forces or freely floating. ROVs typically belong in this group since they are able to move around in the environment. This makes the prediction more complicated as unknown and changing external factors has to be considered.

As previously mentioned, there is a great variety in the tasks used in Table 1.2. They do however have one thing in common; they all include some sort of lateral movement. Typically the operator is required to perform an alignment or aiming task. These kinds of tasks are particular exposed to the detrimental effects from communication delay. It can cause the operator to *overshoot* the target and transition to an inefficient *move and wait* strategy which can be measured by task completion time. Lane et al. (2002) noted that this behaviour started to appear at around one second of time delay.

In all kinds of predictive technology a future predicted state of the robot has to be calculated. Variables used and method of calculation varies. Some methods rely on the dynamic equations of the system. Zhang et al. (2017) implemented a version where he used the state equations of a spacecraft and its dynamic properties to calculate its predicted state. The operator was then presented with a future predicted image of the spacecraft and gave commands correspondingly. This can be a good approach in space since all external forces can be accurately modeled.

In situations where the external forces can not be calculated exactly and the ROV is free to move around, a *model free* approach (no dynamics) are often used instead. The method of conveying this information can be divided into three groups: superimposed predictive information, 3D graphic models and video manipulation.

1.3.1 Superimposed predictive information

In this category, predictive information is overlayed or *superimposed* on the delayed video feed. In that way the operator is able to see estimates on where the ROV is going to end up. The prediction is often visualized as vector graphics in the form of lines or points along a path. Mathan et al. (1996) used this approach when he superimposed directional velocity information related to a lunar rover on a video display.

A similar example can be seen in airplanes and helicopters where a *tunnel in the sky* display shows where the aircraft should be going and a cross indicating the predicted trajectory (Grunwald et al., 1981). In cases with large amounts of lateral movement this approach might not be applicable as the predicted heading can come off screen.

1.3.2 3D graphic models

About half of the experiments in Table 1.2 would adhere to this category. Generally, a 3D world is constructed from sensory input such as laser ranging, stereo cameras, image tracking or others. Images taken by normal cameras are then mapped to the surface of the computer generated world. Lastly, a virtual camera is placed inside the virtual world in the predicted position of the real camera. The operator is then given the virtual video feed as virtual reality (VR), or in a combination with the real one, augmented reality (AR). As Hu et al. (2016) put it:

> In [a] VR-based Predictive display (PD), instead of delayed visual feedback from the remote robot site, an immediately and predicted visual feedback is rendered from a graphics model in response to the operator's motion command.

Some of the technologies used for capturing the 3D world are *Monocular Simultaneous Location and Mapping* (SLAM), stereo imagery, vision-based structure from motion (SFM), light detection and ranging (LiDAR) or radio detection and ranging (radar).

This method is particular popular in conjunction with robot arm manipulators. In these cases the 3D environment can be constructed in advance and the exact location of the robot arm is known (Ricks et al., 2004). A limitation with this approach arises when tasks are performed in unknown and unstructured areas. Then geometry can not be created in advance and real time mapping and rendering can be difficult. In addition, it can require additional hardware such as stereo cameras and the calculations can be computer intensive.

1.3.3 Video manipulation

Video manipulation is a more simple solution as it does not require 3D information about the environment. This approach tries to make alterations to the delayed video such that it looks like the ROV is actually moving in real time. A simple example would be to zoom into the image if the robot is moving forward. Matheson et al. (2013) halved the task completion time at a latency of three seconds in his experiment. He described the method as such:

> [The] display is obtained by estimating the current rover position within the delayed drive camera image, finding the current field of view edges given the rover's location and orientation, and manipulating the delayed image through cropping and projection, to approximate the view from the current rover location.

A similar result is obtained by capturing a wide FOV video, possibly 360 degrees, and then only displaying a section of that image to the operator. The section can then be moved around in the video as a response the operator's commands and thus provide fluid and seemingly real time feedback (Baldwin et al., 1999).

The approach of video manipulation has the advantages of being low cost, easy to implement and it does not rely on a structured environment. In addition, since the displayed video are merely alterations to the last image, no prediction error propagation will happen. It is however not able to recreate *parallax* movement, which can be achieved using the 3D model method above. An example of parallax movement would be when passing a corner or object. New parts of the environment should be visible, but it can not be constructed from a delayed image.

1.4 Problem Statement

Most of the previous research seems to be concerned about 3D environment reconstruction from sensory data. While it shows promising results and great reduction in task completion times, this is a method that requires advanced algorithms and possibly expensive equipment. Many of the mentioned predictive technologies also require extensive information about the environment and the robot in order to function. This is either not possible or a time consuming task. In comparison, the video manipulation method provides and easy and cheap way to increase operator performance.

The *projected display* method described by Matheson et al. (2013) is easiest video manipulation method from Table 1.2, while still providing a good performance increase. This method requires information about ROV ground trajectory to calculate changes in perspective however.

By ignoring the effects of perspective change and instead apply positional and scale transformations, an even simpler approach is obtained. The goal of this thesis is therefore to investigate the following hypotheses:

H1: A simple predictor display based on image transformation can increase the operator performance

H2: A simple predictor display based on image transformation will decrease the operator's subjective workload

2 eduROV Python Package

The eduROV project aims to let hobbyists, enthusiasts and schools create a simple, affordable and open-source underwater ROV. A prototype has been created.

The ROV consisted of a Raspberry Pi 3 Model B+ (RPi) and an Arduino Micro. The Arduino was responsible for reading sensor values and controlling motors. The RPi had multiple tasks, it communicated with the Arduino by reading sensor values and sending motor speed commands. In addition, it captured video from the RPi camera module and displayed this to the user. Lastly, it processed user input from the operator and forwarded these commands to the Arduino.

On the RPi a Python program using the *Pygame*[1] package was running. This is an open source package created for making games, but it can also be used to display video feed and read user input. When initiated, this program would display a window on the RPi. *RealVNC*[2], a software for remote desktop viewing was then used to display this window on the remote computer used for control. Figure 2.1 shows this user interface (UI). The software did not require any installation, instead the correct files had to be copied from a GitHub repository.

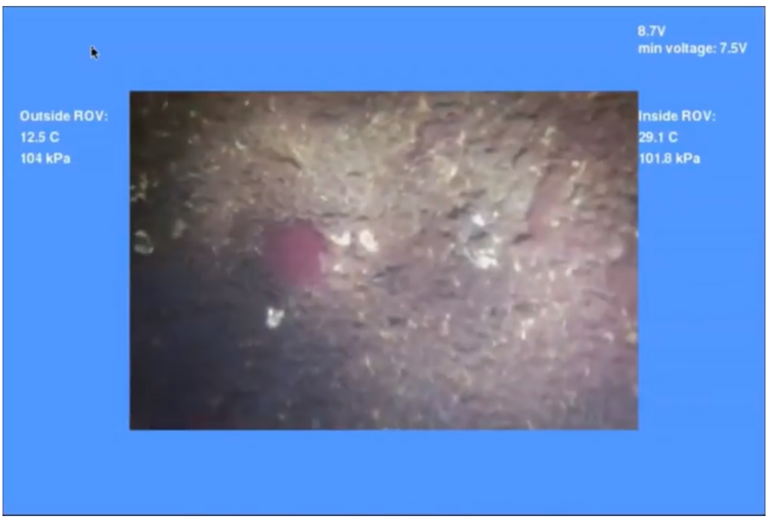


Figure 2.1: User interface for the original eduROV software.

[1] `https://www.pygame.org/`
[2] `https://www.realvnc.com/`

Five main goals were set for my contribution to the project:

- Reduce the video latency as much as possible while still having the possibility for high resolution images.
- Streamline the installation process, i.e. remove the need for visiting any website or manually coping files.
- Remove the need for any third party applications, that would mean removing the RealVNC dependency for video transfer.
- Increase customization while still maintaining a high level application programming interface (API).
- Make the UI more attractive, include more UI features without overwhelming the operator.

2.1 Current Alternatives

There exists a wealth of software created for operating ROVs. This introduction will be limited to those that are open source and created in Python. The most well known and probably most used is the *Robot Operating System* (ROS)[3] which is ported to Python as a client library called *rospy*[4]. Although a powerful framework, it does not suit the needs of this project, as it is originally written in C++. This means that the documentation is mostly for C++ and for anyone who wanted to customize the eduROV software in the future would have learn ROS in addition to Python. It was decided that making ROS fit the needs of the eduROV project would require more time and be limiting to the development, in comparison to creating a tailored software from scratch.

There is also a software called *GoPiGo*[5]. This started as a Kickstarter project and is now a hardware and software project that can be bought online. It provides robot

[3] `http://www.ros.org/`
[4] `http://wiki.ros.org/rospy`
[5] `https://www.dexterindustries.com/gopigo3/`

communication with video feed, but the software seems to be created specifically for the robots they sell and not as a package meant for other users to build on.

In summary, existing alternatives were not found to be good alternatives for the eduROV project. Actually, I was not able to find any Python packages created for ROV communication with video support built in. There are many guides on the internet that will walk you through how to create this, but the whole process can be really intimidating for programmers with limited experience. In addition, many of the guides online require installation of multiple software and other files from additional places, not very user friendly. Also, in the guides online the ROV is controlled by pushing buttons on the screen, not by keyboard input. Lastly, they contain limited to none documentation.

2.2 Development

All popular and well known packages in the Python community is developed in correspondence with the *Python Packaging User Guide*[6]. This guide establishes multiple rules on how packages should be developed and distributed. It is always possible to upload code to a remote repository and ask users to download it from there, but there are many good reasons why serious actors follow the packaging guide.

First reason, by distributing code through the *Python Package Index*, anyone can install the package by running `"pip install edurov"` in a terminal. There is no need to visit websites or copying files. This command will download and install the required files automatically. Second reason is that it greatly simplifies the process of documentation. By creating special files as stated by the guidelines, a separate website with all the documentation is created and uploaded automatically. Thirdly, it also specifies rules for a versioning scheme. This lets the developer create alpha, beta, release candidates and deployment versions of the software. It makes it easy to make sure that everything is tested properly before it gets publicly available.

[6] `https://packaging.python.org/`

Git version control was used throughout the project. All the code was uploaded to the remote repository at `https://github.com/trolllabs/eduROV`. Git branches was used for rapid prototyping of different ideas. This meant that different approaches were developed concurrently in each their branch. They were then removed one after another as it became clear that the approach did not meet the requirements. The finished package code can also be seen in Appendix B on page 89.

In the first phase of the development, two main methods were tested. The first method was based upon the *pygame* package. It required the operator to install Python and the eduROV package on both the ROV and the controlling computer itself. When the software was started, a program window would pop up on the controlling computer and display the video feed. Any customization to the features and UI would require the user to learn pygame as all the graphics are created using the pygame API. The original software also used pygame, but this approach did not rely on RealVNC for transmission. Instead it used socket communication to transfer data.

The second method was based upon a web server approach. This method served a web page from the ROV which could then be viewed on any device connected to the same network as the ROV. This meant that the operator would not have to install any software on his or hers computer. It also meant that the UI would be created using html and css instead of pure pygame. This approach were chosen for the new eduROV package. It would completely remove the need to install anything on the operators computer. It would make it possible to view the video stream at multiple devices at the same time. In addition, web browsers has been around for a very long time and much effort has been spent on making them as efficient and flexible as possible. By using the browser as a medium it is possible to take advantage of this. Some high schools also have web development and html as part of their curriculum.[7] By basing the the eduROV package on a web server framework it becomes possible to let the operator customize the UI with their knowledge of html and css.

[7] `https://www.udir.no/kl06/INF1-01/`

When the main method were chosen, proceeding development were administered through the *GitHub issues workflow*[8]. Figure 2.2 shows a section of the *issues* page on GitHub. On this page, feature requests and bug descriptions were posted by me and one other that tested the software. These were then completed in turn and uploaded as *commits* and *releases*. Issues were labelled in correspondence with application area and severity.

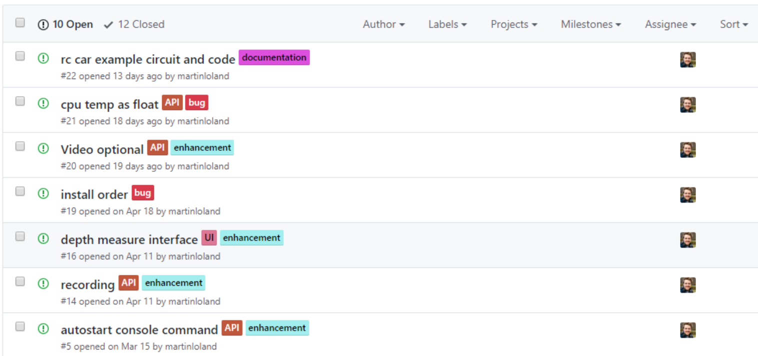


Figure 2.2: GitHub eduROV issues overview.

2.3 Architecture

The eduROV package is based upon a HTTP web server framework. This means that any information sent between the ROV and the user is communicated though HTTP GET requests. For increased performance and robustness many of the different tasks are spread on multiple processes running in parallel. This ensures utilization of multiple CPU cores. The Pyro4[9] Python package uses socket TCP communication and were chosen to facilitate transfer of data between processes. It is fast, well maintained and easy to use.

Figure 2.3 pictures the flow of information between different processes and parts of the system. When the user interact with the keyboard, this is sent as a HTTP request from the web browser to the web server on the ROV. This is a threaded HTTP server, which means that multiple requests can be handled at the same time in different computer threads. The web server will forward this information

[8] `https://github.com/trolllabs/eduROV/issues`
[9] `https://pythonhosted.org/Pyro4/`

to the *synchronize process* which is responsible for holding an updated version of all variables. The *Arduino process* checks the synchronize process many times per second and forwards any new key presses to the Arduino through a serial connection, which then moves the motors correspondingly. Sensor values moves in a similar fashion, only in the opposite direction. The camera captures frames, compresses them to .jpg files and store them in a memory buffer. The webserver will then send the image to the web browser as soon it is ready, directly from the buffer. Lastly, the *system monitor process* regularly checks that the drive space and CPU temperature is ok and notifies the synchronize process.

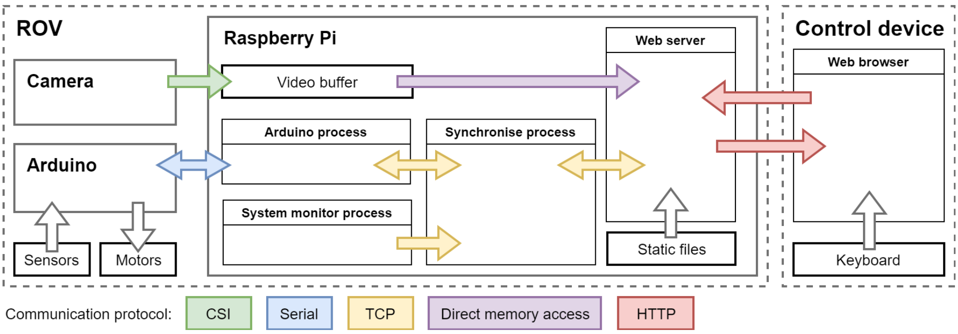


Figure 2.3: System architecture of the eduROV software.

2.4 Graphical User Interface

Figure 2.4 shows the finished UI. The layout is dynamic which means that it will fit any screen size and ratio. The side panels will stay the same size, but the video will shrink and increase in size to what's available. Left panels shows sensor values from the Arduino and RPi. Center section shows the video feed. There is also a roll indicator that shows how the ROV is oriented in the water. This indicator can be toggled on/off from the button menu in the right panel. From this panel the operator has multiple options, such as to arm the robot. If the robot is not armed it will not move. Cinema mode will hide all panels and scale the video feed to its maximum size. Some of the actions can also be triggered with hotkeys. With this layout, the user can chose whether to view all information or nothing except the video feed. An approach with information in side panels was chosen because, as mentioned in the introduction, Chen et al. (2007) argued that ”overlaying information on video feed can potentially lead to cognitive tunneling”.

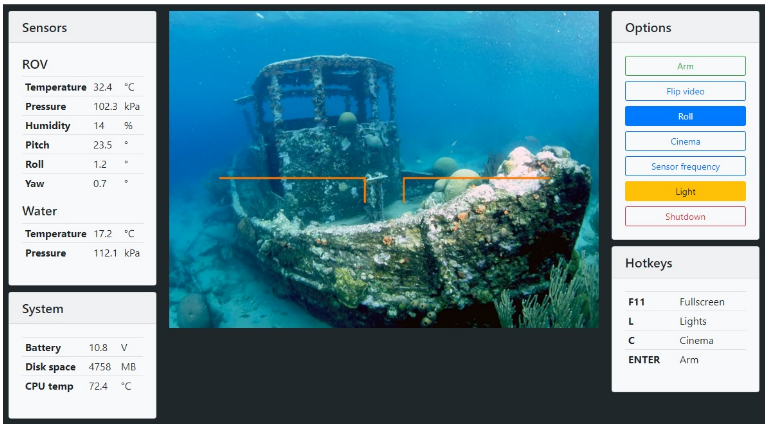


Figure 2.4: The graphical user interface in the eduROV package.

The UI seen above is in the context of the Engage eduROV submersible. But since the UI is created purely in HTML, CSS and JavaScript, any user of the package can customize the look and feel of the webpage in any way he or she would like. In fact, a completely different UI was created for the experiment in chapter 4, but still used the eduROV framework for handling requests.

2.5 Application Programming Interface

The application programming interface (API) is how the user interact with the software package. One of the goals of the project was to create an API that would get the user up an running in a matter of minutes. In addition, provide a flexible API that provides extensive flexibility and customization. There is one single main class called `WebMethod`. By initiating this class with the path to the `index.html` file which decribes the layout, the web server will start running and serving the web page and video feed. In addition to that, the user is able to customize which functions that should be started in their own processes, custom responses to GET methods, resolution and frame rate and much more.

The reader is recommended to take a look at the API[10] and getting started[11] section of the documentation. These pages describes the API and provide a much better user experience than what can be provided in a book.

2.6 Documentation

The documentation is written using the *reStructuredText*[12] markup syntax and compiled using the Sphinx[13]. This enables in-line program documentation. This is very helpful because the documentation for the classes, methods and functions can be written in the same place as the actual code. This creates fewer files which makes it easier to maintain. In addition, sphinx will automatically detect classes, functions and methods and create a corresponding documentation structure.

The GitHub repository has been connected to an account at *readthedocs.io*. By connecting these accounts, a documentation website is automatically created from the sphinx structure when updates are committed to the repository. The documentation can be seen online[14] or in Appendix A on page 57. A sample can be seen in Figure 2.5.

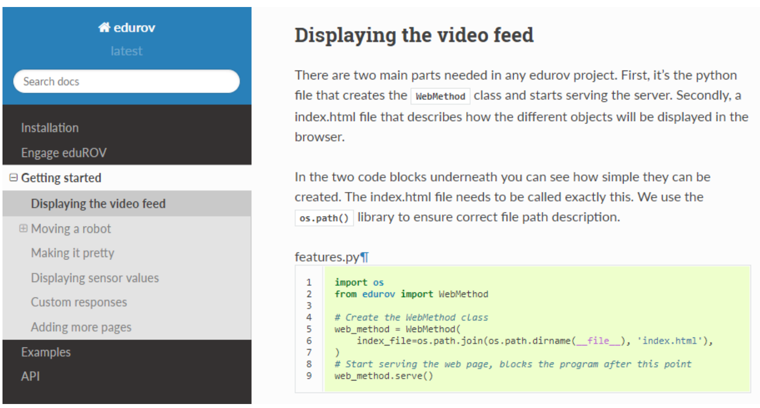
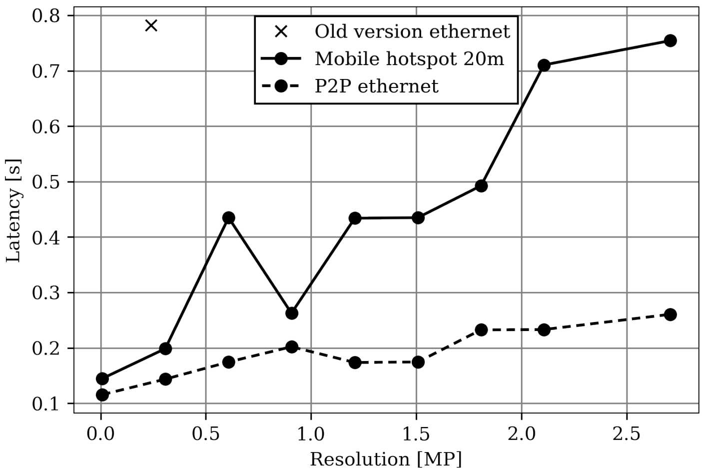


Figure 2.5: eduROV documentation at readthedocs.io.

[10] `http://edurov.readthedocs.io/en/latest/api.html`
[11] `http://edurov.readthedocs.io/en/latest/started.html`
[12] `http://docutils.sourceforge.net/rst.html`
[13] `http://www.sphinx-doc.org/`
[14] `http://edurov.readthedocs.io`

2.7 Performance and Novelty Features



MP	Resolution
0.007	100x75
0.307	640x480
0.608	900x675
0.907	1100x825
1.208	1268x951
1.508	1417x1063
1.808	1552x1164
2.107	1676x1257
2.708	1900x1425

Figure 2.6: Video latency for eduROV 0.0.5 30fps for multiple resolutions.

As part of the development of the eduROV package, a latency test was performed. Figure 2.6 shows this test for version 0.0.5 of the eduROV package. With a resolution of 0.3 mega pixels on a wired ethernet connection, the video latency has been reduced from 782ms to 143ms. This is a 82% reduction. It is even possible to stream full HD video with a latency below 300ms. When using wireless transmission the latency is affected by factors such as distance, interference and hardware.

The test was performed in the same way as Jennehag et al. (2016) did in their test. By manually comparing two timers, one in real time on the monitor and the other as captured by the camera and transmitted back to the same monitor. This was performed two times and the displayed value is the average.

A summary of the most novelty features in comparison with other similar solutions can be listed as follows:

Low video latency. Possibility to stream high definition video with a delay below 200ms.

No setup required. The controlling computer does not need any software installed. The ROV can even be controlled from a mobile phone.

Very easy to use. One command in the terminal window will install all required files. One additional command will start the web server.

Highly customizable. Since the UI is created in html the user can customize the look and feel of the web page in any way.

Easy true parallelism. Custom functions can be spawned on multiple CPU cores while still maintaining the possibility to share variables.

For future work there is one limitation to the current design. The client browser communicates with the web server with GET requests. Each time the UI is updated, the client has to ask for this update, there is no way that the server can send new information to the client on its own. This is unless a *WebSocket* connection is used. Instead of creating a new connection each time a request is done, a websocket is open as long as the client is viewing the web page. This enables the server to push information when it have something new and thus removing a lot of unnecessary communication. This would probably require comprehensive changes to the underlying workings of the eduROV package, but is probably where the next big performance gain can be achieved.

3 Predictive Display Scheme

This chapter describes the developed predictive display. The final results only requires a few lines of code and can be applied to most ROVs. In the coming explanation a very simple and limited ROV is considered, but section 3.4 describes how the principle can be expanded to more complicated configurations.

3.1 Robot Configuration

To explain how the predictive display (PD) works, let us consider the self balancing two wheeled robot depicted in Figure 3.1. The upper part of the figure shows the robot from above with two objects in front of it, a black cube and a gray barrel. The ROV is drawn at time equal to $t = 0$ and $t = \Delta t$. The bottom part of the image depicts the viewport of the onboard camera mounted to the ROV.

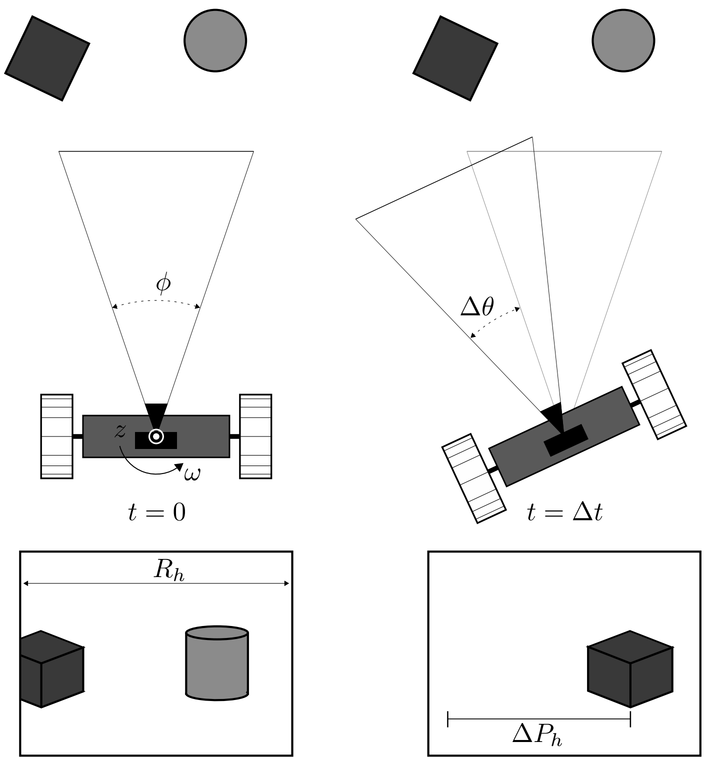


Figure 3.1: Two wheeled robot before and after counter clockwise rotation.

It has a forward facing camera with a FOV of ϕ degrees. The camera captures a video feed with a resolution of R_h pixels horizontally. Its center of rotation is located in the vector z pointing out of the paper. It is able to rotate with an angular velocity of ω deg/sec around its center of rotation z.

Let us first consider a situation without delay and where the ROV can only be given two commands, to turn either left or right. The commands are given by pressing one of two buttons, not by a joystick with variable output. If the operator holds down the *left* button for a period of Δt seconds, the ROV would make an angular rotation of $\omega \cdot \Delta t = \Delta\theta$ degrees. This is depicted in the right side of Figure 3.1.

In the viewport, the cube and barrel would move to the right as the ROV turned left. These objects has moved a finite number of pixels horizontally ΔP_h, which can be calculated by Equation 3.1. It is simply the ratio between the angular rotation and the FOV, times the pixel screen width. By substituting in the expression for angular rotation, Equation 3.2 is obtained. Here η is used to denote the *pixel turn rate*; the pixel rate at which objects in the video moves left or right when the operator turns the ROV.

$$\Delta P_h = \frac{\Delta\theta}{\phi} \cdot R_h \tag{3.1}$$

$$\Delta P_h = \left(R_h \frac{\omega}{\phi} \right) \Delta t = \eta \cdot \Delta t \tag{3.2}$$

η is a constant and depending on the screen resolution, camera FOV and the angular velocity of the ROV. By multiplying this factor by the amount of time the operator holds down the left or right button, the number of pixels the objects in the frame should move is obtained.

3.2 Predictive Visualization

Let us now consider a situation where there is a t_d seconds delay from when the commands are given by the operator, to the changes can be seen in the video feed. This is the *total perceived delay* described in the introduction, section 1.2.2. For simplicity, let us also consider a situation where $\Delta t < t_d$.

Figure 3.2 shows a representation of what the video feed would look like as the above maneuver was performed. It shows the situation in three different scenarios. First no delay, secondly with delay and third with the PD implemented using the delayed video. The outer rectangle shows the limitations of the monitor, while the inner rectangle is the video feed itself.

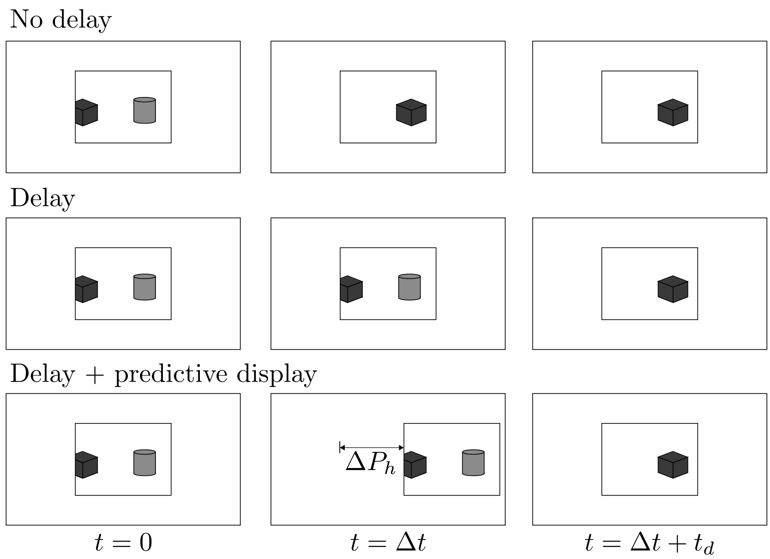


Figure 3.2: Operator view. Outer box total screen size, inner box video feed.

Figure 3.3 plots the visible angular rotation α for the no delay display and the delayed display as a function of time. For the no delay display, visible angular rotation is equal to ROV angular rotation $\alpha = \theta$. In addition, the horizontal image pixel displacement P_h is plotted with the same time axis.

The PD works by moving the video feed on the operator screen the opposite way of what the ROV is moving. The amount of pixels the video P_h is moved is calculated by Equation 3.2. In addition, the video is moved back (the same way as the ROV is moving) after t_d seconds has passed. This makes the objects in the video feed appear in the correct position on the operator screen as if there were no delay. Note that the black box in Figure 3.2 predictor display center column is in the correct position relative to the no delay display. This approach does however assume that the commands will be properly followed by the ROV. But since the prediction is merely an alteration to the last image received, the prediction errors are not cumulative.

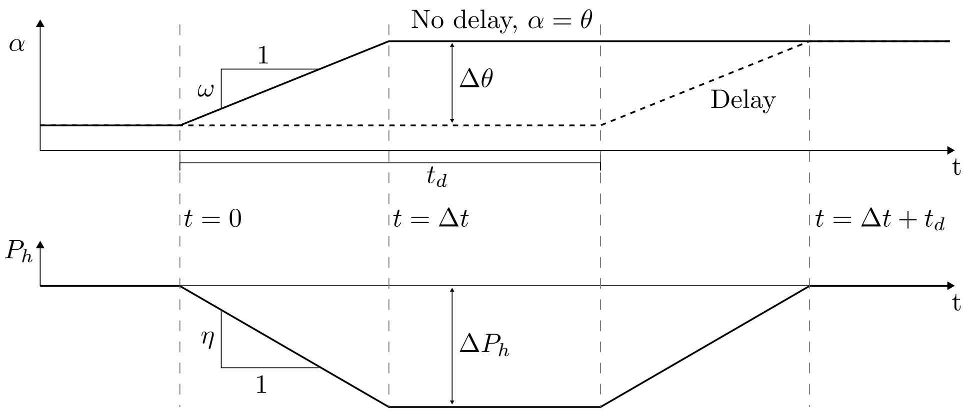


Figure 3.3: Visible angular rotation and horizontal image pixel displacement as a function of time.

3.3 Implementation

Algorithm 1 shows the pseudocode for how this PD is implemented in practice. The horizontal pixel displacement P_h is initialized as zero. Then, the PREDICTOR DISPLAY function is called at a set interval dt. The rate of these calls should happen at least as fast as the frame rate of the video (fps). With a fps of 30, the interval should be $dt <= 1/30 \approx 33ms$. The change in horizontal pixel displacement ΔP_h is then calculated from Equation 3.2 and the interface is updated with the new P_h. In addition, an asynchronous call is done on the MOVE BACK function so that the video is moved back to its original position after t_d seconds has passed. It has to be an asynchronous call so that the main program is not blocked when the MOVE BACK function is waiting.

Algorithm 1 Predictive display

$P_h = 0$ ▷ horizontal pixel displacement
SET INTERVAL(predictor display, dt) ▷ calls function at interval

function PREDICTOR DISPLAY
    if left **then**
        $\Delta P_h = -\eta \cdot dt$ ▷ equation 3.2
    else if right **then**
        $\Delta P_h = +\eta \cdot dt$ ▷ equation 3.2
    else
        $\Delta P_h = 0$
    end if
    P_h += ΔP_h
    UPDATE INTERFACE(P_h)
    MOVE BACK(ΔP_h) ▷ asynchronous call
end function

function MOVE BACK(ΔP_h)
    wait t_d
    P_h -= ΔP_h
end function

Figure 3.4 shows the predictor display as it was implemented in the experiment, which is explained in chapter 4. It contains the video feed from the ROV, in addition to a red arrow to visualize the prediction. The operator has recently turned the ROV to the right, and as a result the video has moved to the left. The red arrow has not moved and works as an indication of where the ROV will be heading when the video feed has caught up with the time delay.

Figure 3.4: Predictor display visualization.

The operator views the predictor screen through a web browser. The predictor algorithm is written in java script and the video feed is moved around by changing css margin properties. The code can viewed in its entirety in Appendix C on page 119, or online.[1]

3.4 Extending and Generalizing

The pixel turn rate η described in section 3.1 was related to the rotation of the ROV. A similar constant can be found for the *pixel scale rate*, which relates how the the video should be scaled when the ROV moves back and fourth. It's a bit more complicated since the apparent scaling of objects in the frame depends on how far away they are, but by using an average distance this can at least be approximated. The same approach as in Algorithm 1 can then be used for backward and forward motion to manipulate the scale of the video feed.

In the case of a varying magnitude of left, right, forward and backward movements, such if the operator is using a joystick with variable output, the PD has to account for this. This can be achieved by applying an adjustment factor to the pixel turn/scale rate proportionate to the magnitude of the command.

The predictor display can then be applied to all moving ROVs. It is just a matter of finding the correct pixel turn/scale rate and adjustment factors corresponding to how the ROV is moving. A submersible ROV would typically have a much lower pixel turn/scale rate because of water friction.

These rates and factors can be found by calculation using ROVs physics and screen resolution. But they can also be found using trial and error. For example, if the visual angular rotation is less than the actual angular rotation, increase the pixel turn rate until they match. In this way, the predictor display can be calibrated without knowing any of the ROVs physics. In this context, *ROV physics* means how the ROV respond to operator input, how fast it moves and turns.

[1] `https://github.com/trolllabs/eduROV/blob/master/examples/experiment/displays/predictive.js`

4 Experiment

The goal of the experiment was to measure the human performance change in a ROV maneuvering task using a predictor display based on image transformation. The participants were given a modified ”peg-in-hole” task, where the peg was mounted on a remotely controlled ground vehicle and the holes were rectangular holes in a wooden box.

4.1 Participants

The participants were voluntary selected from the NTNU Department of Mechanical and Industrial Engineering. A total of 58 participants performed the experiment whereas the first one were excluded from the data foundation. This was due to lack of information that became evident during the first trial. This information were given to the other $N = 57$ participants. None of the subjects had any earlier experience with predictive displays.

33.3% of the participants were female and the total group had an average age of 24.7 years with an standard deviation (SD) of 1.45. This information among others can be seen in Table 4.1.

Table 4.1: Demographic details on participants in experiment.

		Number of people	Percentage	Mean	SD
People tested	Total	58			
	Excluded	1			
Gender	Male	38	66.7		
	Female	19	33.3		
Age				24.7	1.45
Use computer daily		57	100		
Gaming	Daily	2	3.5		
	Weekly	15	26.3		
	Monthly	8	14.0		
	Yearly	17	29.8		
	Never	15	26.3		

4.2 Experimental Design

Figure 4.1 shows an overview of the experimental setup. A 17 inch laptop running with a 2.3GHz Intel Core i7-3610QM CPU and Windows 10 together with the arrow keys were used as the operator's control device. This was connected to the ROV through a direct Ethernet connection.

Figure 4.1: Experimental setup. The computer (left), is used to control the ROV into holes in wooden box (right).

The ROV, Figure 4.2, was a three wheeled robot running a Raspberry Pi 3 Model B+. Two of the wheels where connected to each their DC-motor while the third one was a caster wheel for support. The ROV was equipped with a forward facing Raspberry Pi Camera V2. The camera has a wide angle lens attached with a horizontal FOV of 76.5 degrees. The robot was running the eduROV software outlined in chapter 2. This software was responsible for serving the control interface, handling control commands, logging experiment data and adding the desired latency to the communication.

A wooden box with three holes and LEDs were used to register task performance. The distance between the holes (center to center) was $D = 30cm$ while the holes itself has a width of $W = 10cm$. This translates to a Fitts's *index of difficulty* of $I_d = log_2\,(2D/W) = 2.58\,bits$ (Fitts, 1954).

4.2.1 Task

One by one, in random order, the round LED on the button box would turn on. The operator was then tasked to maneuver the ROV such that the black peg would go inside the corresponding hole. A light sensor inside the hole would register this as a *hit*. This would cause the LED to turn off and one of the other two to turn on. The participants were told to make as many *hits* as possible in the course of 90 seconds.

The participants would repeat this task a total of three times, using three different displays / conditions. The order of these conditions followed a 3x3 Latin Square Design, to eliminate the order effect. Condition one had a total delay of 700 ms which included the inherent system delay of 250ms, plus the added delay of 450 ms. Condition two had the same delay as condition one, but with the predictive display in effect. The third condition had no added delay and only the inherent delay of 250ms. No predictive technology was used in the third condition. The total latency of 700 ms were chosen because it is below the reported threshold for a "move and wait" strategy (Chen et al., 2007), and above what is considered a difficult level in many situations.

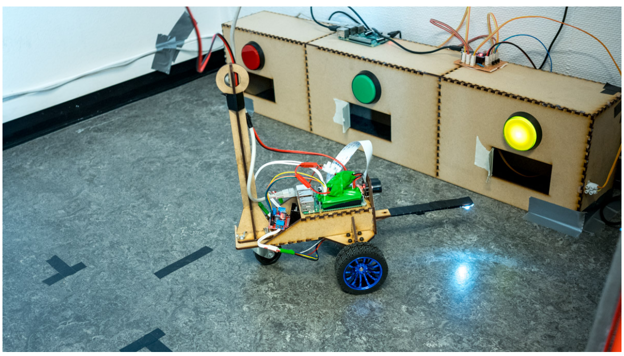


Figure 4.2: Three wheeled robot used in experiment.

Many of the experiments previously mentioned in Table 1.2 used a single task and measured the task completion time in different conditions. This experiment was however designed with a single simple task and measured the achieved score in the course of a fixed time period. There are multiple reasons for this choice.

First, to reduce the learning effect that would accompany a longer maneuvering course. Some of the authors in Table 1.2 reported that the participants performed better for each try when they started to learn the obstacle course. I believe that a longer course would require more time to reduce the learning effect.

Secondly, Chen et al. (2007) reported that the benefit of PD is very task dependent. An easy task was therefore chosen to minimize the effect that task complexity had on the performance results.

Thirdly, some experiments with real or simulated driving have long stretches with forward motion. The PD provides little help in these situations but still contribute to the task completion time in the same way. A task which required the operator to move from side to side as much as possible was therefore chosen. In addition, by not letting the operator accelerate to maximum ROV velocity, ceiling effects from vehicle limitations were reduced.

As a fourth argument, a fixed task time made the experiment length much more predictable. Subjects used on average 10 minutes and 56 seconds with a standard deviation of 1 min and 12 seconds to perform the whole experiment. This again made it easier to recruit new subjects.

As a last argument, the combination of score achievement and time pressure made the subjects fully devoted to the task at hand. This made them performed as close as possible to their potential.

4.3 Procedure

After entering the experiment room, the participants were able to look at the ROV with the button box to get a sense of situational awareness. From that point on, the subject was facing the other way, looking at the computer screen and with the robot outside their FOV. The participants also wore an ear protection headset to remove audible feedback. All the necessary information was given on the screen.

First, the subject S was presented with an initial form collecting demographic data. Then, an information page describing experiment theme, how to steer the robot, the participant's goal and how the experiment would proceed was displayed to the subject. This can be found in Appendix D on page 123. The S was then automatically assigned to a group in correspondence to the 3x3 Latin Square Design. The S then performed a 30 seconds long practice period followed by a 90 seconds long real test. This was done repeatedly for all three conditions. At the end of each test, the S was asked to fill out a test questionnaire. After each practice and test run, the ROV was repositioned to its original position defined by the black markings in Figure 4.2. The S was *not* told that one of the conditions would contain a predictor display or how the predictor display worked.

4.4 Data Recording and Analysis

All the data was recorded with the onboard computer of the ROV using a SQLite database. This included demographics, experiment questionnaire data, hits made by the subjects, number of key presses and more. Time stamp data was also recorded for each hit and the test start and end. A total of 11865 data points were collected during the testing period. All the recorded data with exception of the *hits* table is included in Appendix G on page 155.

The test questionnaire that was completed for each condition included a NASA TLX (task load index) form (Hart et al., 1988). In addition the S had to guess the total delay that they just experienced. This questionnaire can be found in Appendix E on page 125. One modification were done to the NASA TLX form. During the preliminary experiment evaluation, a helper reported that he found it naturally to evaluate a good performance with a high score. In the original questionnaire, a low values translates to a good performance. This metric and the corresponding description was reversed such that a high value would reflect a good performance. After data collection, this value was reversed back such that it can be reported inline with convention.

The number of hits made by the S in the course of 90 seconds was used to quantify performance. This score was normalized in the same way as Rachmielowski et al. (2010) and Lovi et al. (2010) did in their analysis. First, the S's number of hits in a specific condition was divided by the S's average hits achieved in all three conditions. It was then multiplied with the average score for all participants in all conditions. The same normalization has also been done on the reported subjective delay for each condition.

To determine the statistical difference between conditions a two-sided paired sample t-test was used. This was calculated using the `scipy.stats.ttest_rel`[1] function which is a part of the SciPy python library. In the results section, the t-statistics is reported as t, the two-tailed p-value as p and the degrees of freedom $N - 1$ as *df.* A difference is reported as significant if $p \leq 0.05$.

When comparing scores based on demographic groups, the variables are no longer dependent. In those cases, a two-sided Welch's t-test is performed instead. This has been computed using the `scipy.stats.ttest_ind`[2] function. The statistics is reported in the same fashion as in the dependent case, only difference is that the degrees of freedom is calculated using the Welch–Satterthwaite equation (Allwood, 2008).

The *effect size*, which describes the magnitude of difference between conditions was calculated using the Cohen's d formula. This value is reported as d in the results. When testing for linear relationships between dependent and independent variables, linear least-squares regression is used. It has been calculates using the `scipy.stats.linregress`[3] function. The R-squared correlation coefficient is reported as R^2.

The code used for the statistical analysis can be found in Appendix F on page 127.

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html
[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html
[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.linregress.html

5 Results and Discussion

This chapter will present the results and discussion divided into six consecutive themes: general performance, gaming effects, task load index, subjective delay, learning effects and key presses.

5.1 Performance

Figure 5.1 shows the normalized number of hits in 90 seconds (score), for each display type and all N=57 participants. *Delay* refers to the *added* delay, which means that "No delay" translates to the inherent system delay of 250ms. The numerical values are reported in Table 5.1. The statistical significance and effect size between conditions can be seen in Table 5.2.

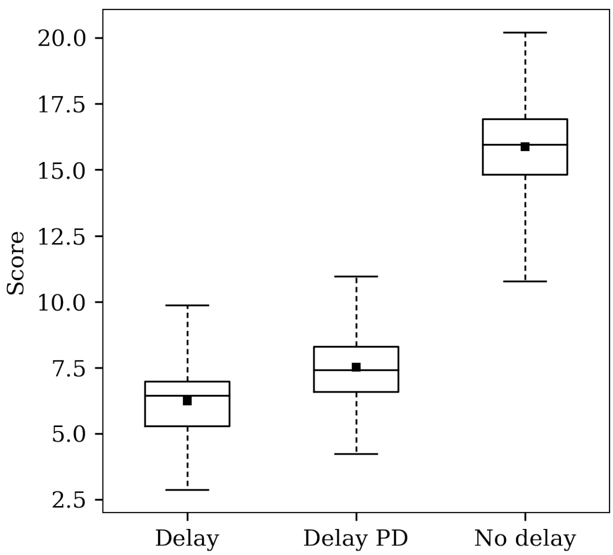


Figure 5.1: Normalized score all participants, N=57.

Figure 5.1 together with the statistical data in Table 5.2 shows that there is a statistical difference in performance when controlling the ROV without and with predictive help. Subjects performed on average 20.6% better with an effect size of $d = 0.904$. This can be categorized as a medium to large effect, especially when considering how easy and cheap this predictive method is to implement.

Table 5.1: Normalized mean scores and standard deviation (SD).

Differentiation	Group	Display	Score	SD
None	All N=57	Delay	6.24	1.39
		Delay PD	7.52	1.43
		No delay	15.87	1.99
Gender	Male n=38	Delay	6.65	1.25
		Delay PD	7.95	1.43
		No delay	17.30	1.71
	Female n=19	Delay	5.39	1.49
		Delay PD	6.61	1.35
		No delay	13.10	2.17
Gaming	Daily n=2	Delay	7.92	0.37
		Delay PD	10.21	1.40
		No delay	18.36	1.77
	Weekly n=15	Delay	6.27	1.22
		Delay PD	8.17	1.51
		No delay	17.62	2.04
	Monthly n=8	Delay	7.05	1.32
		Delay PD	7.77	0.64
		No delay	17.68	0.95
	Yearly n=17	Delay	6.65	1.26
		Delay PD	7.66	1.73
		No delay	15.98	2.25
	Never n=15	Delay	5.06	1.46
		Delay PD	6.21	1.16
		No delay	12.73	1.79

Table 5.2: Mean difference, paired samples t-test and Cohen's d effect size for display pair scores. Gamers = plays weekly or more often.

Group / Pair		Mean difference	t-test for Equality of Means			d
			t	df	p	
All N=57						
Delay	Delay PD	20.62%	4.80	56	<.001	0.904
Delay	No delay	154.37%	23.15	56	<.001	5.569
Delay PD	No delay	110.88%	19.66	56	<.001	4.772
Gamers n=17						
Delay	Delay PD	30.13%	4.34	16	<.001	1.376
Delay	No delay	174.64%	14.93	16	<.001	6.463
Delay PD	No delay	111.05%	10.83	16	<.001	4.965
Non-gamers n=40						
Delay	Delay PD	16.91%	3.20	39	.003	0.731
Delay	No delay	146.46%	18.16	39	<.001	5.237
Delay PD	No delay	110.80%	16.21	39	<.001	4.655

Previous research, Table 1.2, describes a wide range of task time reduction from predictive technology. Everything from 8% to 65%. It is difficult to do a direct comparison to any specific experiment, but a performance increase of 20.6% in this experiment is probably in the lower range of what has been found before. However, the predictive method described in this thesis is the cheapest and easiest to implement, at least when comparing to those in Table 1.2. The *task time reduction* measure is considered to be comparable to the *performance gain* measure in this experiment.

None of the subjects were told that there would be a predictive display or how it worked. Some of the participants immediately identified what the predictive display was trying to tell them. Others however, did not understand that there had been a predictive display until the experiment was over. The ones who tried to use the predictive display the way it was intended typically performed better than those who did not use it. It may be possible that the performance could have been improved if the subjects were informed how the predictive display works. This can however not be verified unless additional experiments are performed.

5.2 Gaming

Those who play games weekly or more were defined as *gamers* (G). They performed on average 30.13% better, while non-gamers (NG) only saw a 16.91% performance increase using the PD. This difference is illustrated in Figure 5.2 and Table 5.2.

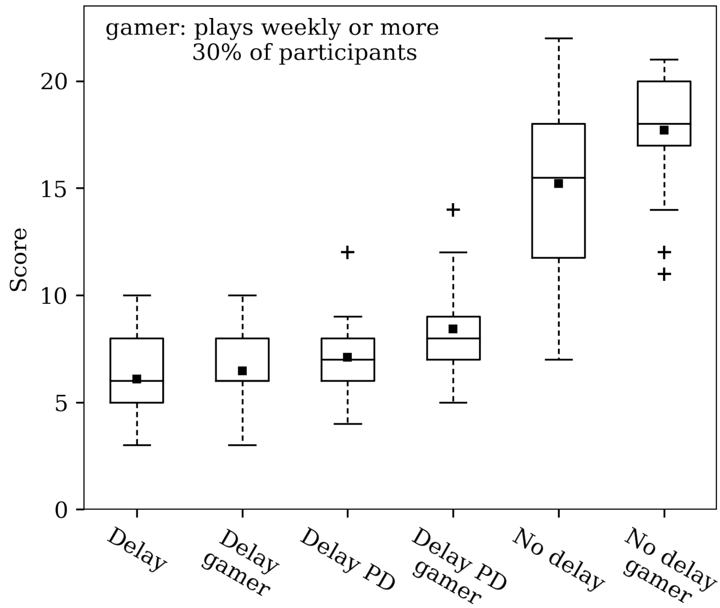


Figure 5.2: Performance of gamers n=17, versus non-gamers n=40. Outliers indicated by plus sign.

It is interesting to see that Gs were able to increase their score almost twice as much as NGs when going from a delayed condition to a delayed condition with PD. Exactly why Gs were able to increase their performance more using the PD is unclear. It could be that the arrow in the PD which acts like an aiming device, is a more familiar concept for gamers. It could also indicate that Gs are generally more adaptable to new an unfamiliar situations in a computer competition setting.

When comparing Gs versus NGs directly, it is also interesting to see that Gs only performed better than NGs in the second and third condition. Delay PD: $t(26.57)=2.23$, $p=0.034$, $d=0.692$ and no delay: $t(40.79)=2.56$, $p=0.014$, $d=0.660$. In the the first condition, there was no significant difference.

5.3 Task Load Index

Figure 5.3 shows the reported NASA TLX scores. The height of the bar describes the mean value while the whiskers shows the SD. Numerical values are reported in Table 5.3.

Subjects reported minimal differences between condition one (delayed video) and two (delayed video with PD). The only significant differences were found in performance and frustration. Subjects felt that they on average performed 14% better using the predictor display, t(56)=3.24, p=0.002, d=0.360. The actual performance increase was 21%. They also reported that they felt 11% less frustrated using the PD, t(56)=2.15, p=0.036, d=0.271. Participants also stated that the no delay condition was better in all metrics, with an exception of *temporal demand* where the difference was not significant.

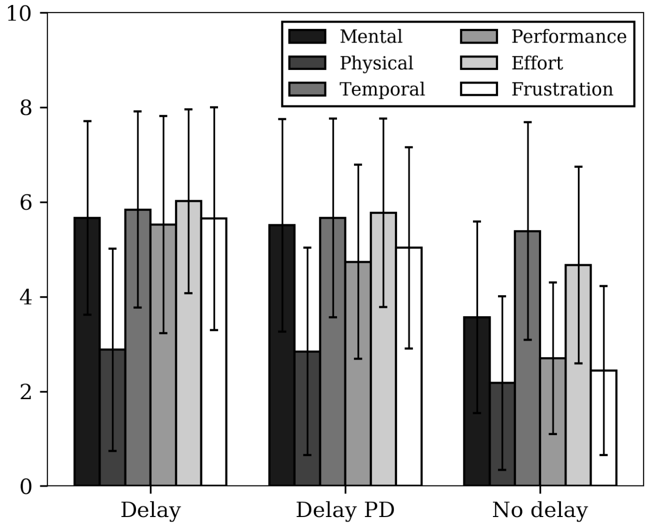


Figure 5.3: NASA TLX (task load index) results for each display type, N=57. Lower is better.

The subjects reported no significant difference in mental, physical and temporal demand between condition one and two. These three metrics is a good description of the total subjective workload. Some subjects, especially those who did not understand what the PD were trying to tell them, even reported the PD as distracting.

Table 5.3: Rated NASA TLX values and standard deviation (SD), N=57. Lower is better.

Metric	Display	Rating	SD	Metric	Display	Rating	SD
Mental	Delay	5.67	2.05	Performance	Delay	5.53	2.29
	Delay PD	5.51	2.25		Delay PD	4.74	2.05
	No delay	3.56	2.03		No delay	2.70	1.60
Physical	Delay	2.88	2.14	Effort	Delay	6.02	1.94
	Delay PD	2.84	2.19		Delay PD	5.77	1.99
	No delay	2.18	1.84		No delay	4.67	2.08
Temporal	Delay	5.84	2.08	Frustration	Delay	5.65	2.35
	Delay PD	5.67	2.10		Delay PD	5.04	2.13
	No delay	5.39	2.30		No delay	2.44	1.79

Because of how the PD works, the video feed is constantly moving around and scaling up and down. This can understandably be distracting. Some participants immediately understood how the PD worked, and they typically reported the PD as helpful. They also seemed to be more relaxed, but there are no recorded data that can prove this relationship.

During the task, a red timer indicating the remaining time was constantly visible for the participant to see in the upper right corner. In addition, the robot had rapid acceleration and was able move fast if the operator managed to do so. Overall, this made for a hectic and exiting experience for the subjects. This may explain why there is no significant change in the temporal demand, even compared to the no delay situation. The fact that the participants reported a better value (smaller) in the other five metrics for the no delay condition, is as expected. This finding supports earlier research that describes how video latency negatively affect the user experience in teleoperation.

5.4 Subjective Delay

Figure 5.4 shows the normalized reported total delay in seconds for the three conditions. The participants reported a 11% decrease in subjective latency using the predictive display versus the normal display with the same latency. This results is however not significant, t(56)=1.40, p=0.167, d=0.356.

About 75% of the participants underestimated the latency in the third condition. Many of them barely reported over 0ms, but the actual latency was 250ms. These findings support previous research, which states that smaller latencies closer to zero is difficult to differentiate from no latency.

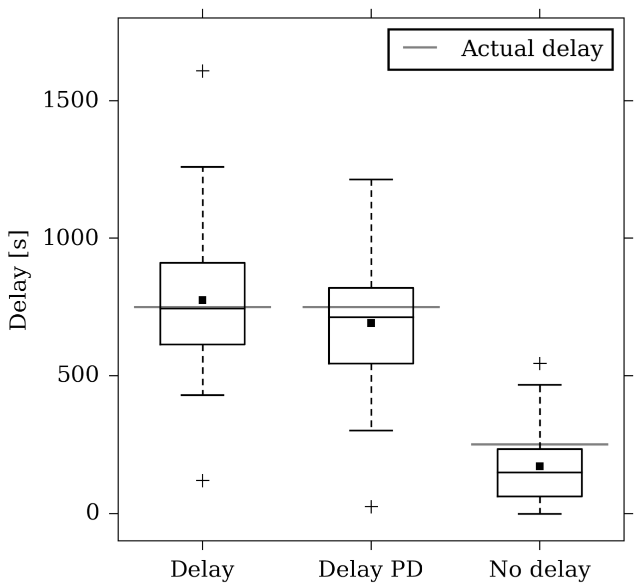


Figure 5.4: Normalized reported subjective latency in seconds.

Since the participants reported less frustration using the PD, it it interesting to look at how frustration and subjective delay time might be related. Figure 5.5 shows a scatter plot of reported frustration and delay time for all conditions collectively. All values has been normalized. The linear relationship is small, but still noticeable. Note that Figure 5.5 presents the subjective latency in all conditions, this means that there are differences between actual latency also. When looking at the different conditions isolated, there are no significant relationships.

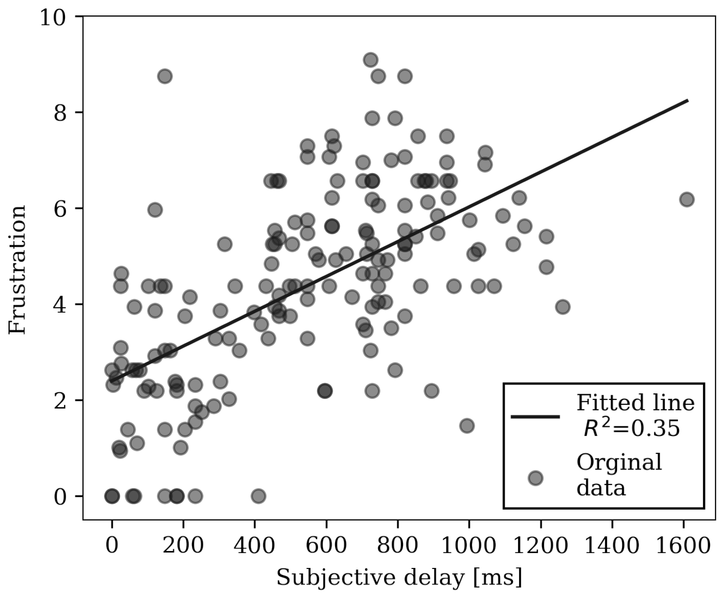


Figure 5.5: Normalized subjective delay versus frustration.

5.5 Learning Effect

It is also interesting the evaluate if participants had any learning effects during the experiment. Figure 5.6 shows the score for each display type and further divided into groups depending if the subject had that display as the first, second or third display. As an example, the first of the nine box plots describes the score achieved in the delay condition for those who had that display as their first display. One visible trend is that the participants who had a particular display as their second display, performed better than those who had that display as their first. This performance increase was significant for all displays. Delay: t(18)=2.19, p=0.042, d=0.671, delay PD: t(17)=2.19, p=0.043, d=0.660, no delay: t(17)=3.26, p=0.005, d=0.902. The performance change from #2 to #3 in all displays were however not found significant.

This indicate that the participants had some learning effect from the first to second display. But after that, the learning effect was eliminated.

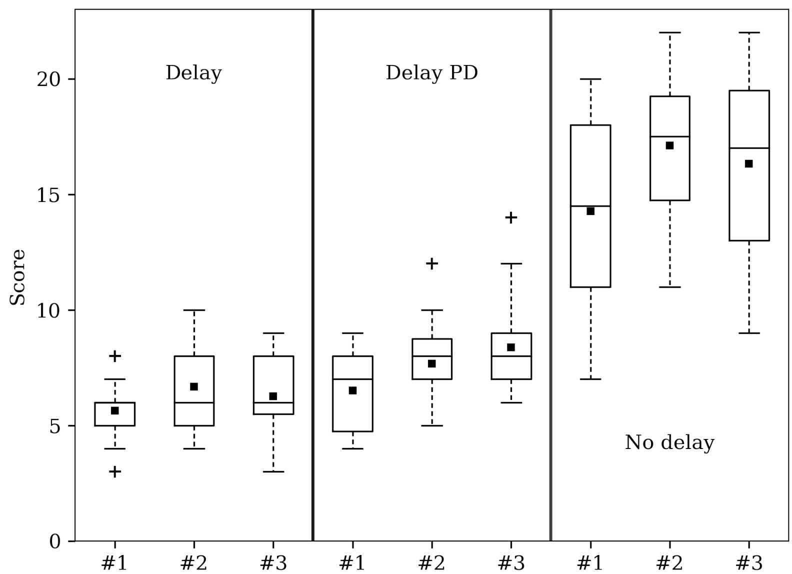


Figure 5.6: Score categorized after display order.

Table 5.4: Mean score and standard deviation (SD) for each group.

Group	n	Display order	Display	Score	SD
Group 1	9	1-2-3	Delay	5.08	1.21
			Delay PD	7.40	1.13
			No delay	15.40	1.83
Group 2	10	1-3-2	Delay	6.24	0.72
			Delay PD	8.45	0.96
			No delay	17.41	1.02
Group 3	10	2-1-3	Delay	6.65	1.15
			Delay PD	6.32	1.03
			No delay	17.04	1.51
Group 4	10	2-3-1	Delay	6.58	1.52
			Delay PD	6.57	0.57
			No delay	16.76	1.59
Group 5	9	3-1-2	Delay	6.78	0.91
			Delay PD	8.42	1.33
			No delay	14.69	1.45
Group 6	9	3-2-1	Delay	6.03	1.84
			Delay PD	8.04	1.45
			No delay	13.59	2.58

Table 5.4 and 5.5 presents the achieved score in each of the experiments groups. These groups are defined by the 3x3 Latin Square design to minimize order effects. It is interesting to see that the two groups who had condition two, the predictor display first, did not show any statistical difference in performance between condition one and two. But all the other groups who had the PD as their second or third display, showed a performance increase from 24.34% to 45.58%. This would indicate that the PD was more helpful, if the subject had tried one of the others first.

It seems that the learning effect from the first to the second display helped the performance of the PD, but *not* the ordinary display with latency, indicated by group three and four. At least when comparing the performance difference between condition one and two.

Table 5.5: Mean difference, paired samples t-test and Cohen's d effect size for display pair scores. Seperated by experiment display order.

Group / Display order Pair		Mean difference	t-test for Equality of Means			d
			t	df	p	
Group 1, n=9, 1-2-3						
Delay	Delay PD	45.58%	4.49	8	0.002	1.867
Delay	No delay	203.00%	10.11	8	<.001	6.274
Delay PD	No delay	108.13%	8.11	8	<.001	4.960
Group 2, n=10, 1-3-2						
Delay	Delay PD	35.42%	4.89	9	<.001	2.474
Delay	No delay	179.12%	22.73	9	<.001	12.050
Delay PD	No delay	106.11%	14.59	9	<.001	8.602
Group 3, n=10, 2-1-3						
Delay	Delay PD	-4.98%	-0.63	9	0.544	-0.288
Delay	No delay	156.29%	12.57	9	<.001	7.347
Delay PD	No delay	169.73%	13.93	9	<.001	7.884
Group 4, n=10, 2-3-1						
Delay	Delay PD	-0.13%	-0.01	9	0.988	-0.007
Delay	No delay	154.87%	9.99	9	<.001	6.211
Delay PD	No delay	155.19%	16.53	9	<.001	8.081
Group 5, n=9, 3-1-2						
Delay	Delay PD	24.34%	2.66	8	0.029	1.366
Delay	No delay	116.81%	11.05	8	<.001	6.163
Delay PD	No delay	74.38%	6.74	8	<.001	4.248
Group 6, n=9, 3-2-1						
Delay	Delay PD	33.42%	2.74	8	0.026	1.147
Delay	No delay	125.50%	5.06	8	<.001	3.190
Delay PD	No delay	69.02	4.18	8	0.003	2.503

5.6 Key Presses

Figure 5.7 shows the number of key presses performed during the 90 seconds of task time for each display type. In condition three, the no added delay condition, participants make a lot more key presses. With a low latency, participants are in a greater degree trying to continuously maneuver the ROV, instead of partially adapting a move and wait strategy.

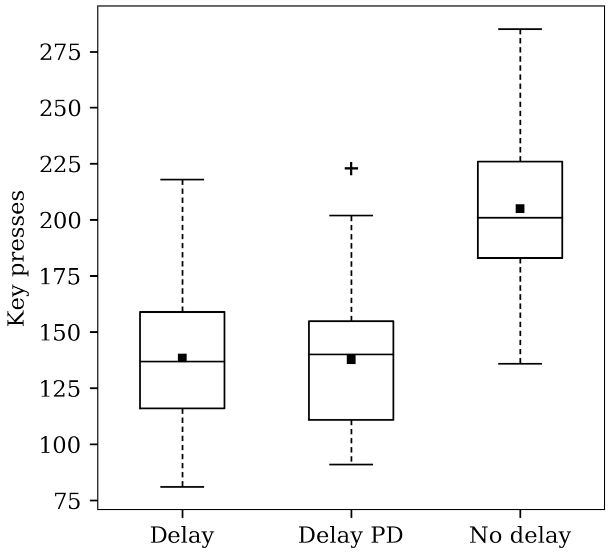


Figure 5.7: The number of key presses performed during 90 seconds.

5.7 Limitations

The choice of not informing the participants about the predictive display before they started the experiment was a conscious one. Because of this, the experiment was limited to testing the performance increase by an *intuitive* understanding of the PD. The results may had been different if the participants were informed in advance. The experiment was also conducted indoors on a flat confined area using a ROV with zero turn radius. More experiments needs to be performed to evaluate if the results in this thesis are applicable to a real ROV in an unstructured environment.

6 Conclusion and Summary

The developed predictive display is very easy to implement and does not require any additional hardware, nor is it very computational intensive. It can be implemented on all ROVs with a fixed onboard camera, that are free to move in an environment. Only a few constants describing the behaviour of the ROV is needed, and those can be found by trial and error.

H1: Participants performed on average 20.62% better using the predictive display versus no predictive display, $t(56)=4.80$, $p<.001$, $d=0.904$. H1, that a simple predictor display based on image transformation can increase the operator performance, is therefore verified.

H2: The participants did not reported any significant difference in the mental, physical or temporal demand using the predictive display. H2, that a simple predictor display based on image transformation will decrease the operators subjective workload, has to be rejected.

The participants who play video games weekly or more were found to have a larger performance increase from the predictive display than those who do not. The predictive display use a red arrow to visualize future position. This can resemble an aiming device which is a more familiar concept for gamers and can explain some of the difference.

The experiment showed that all groups performed relatively worse on the first display than they did on their second and third display. As a result, those who had the predictive display as their first display, did not show any performance difference using the predictive display versus the normal display with same delay. Those who had the predictive display as their second or third condition however, showed a performance increase of 24% to 46%.

The predictive display offers a valuable performance increase, especially considering how easy and cheap it is to implement.

6.1 Future Work

Although the predictive display (PD) increased performance, many participants experienced minor improvements. In addition, some of the subjects even reported that the PD was distracting and confusing. I believe that there are two main reasons for this.

Firstly, the fact that the image itself is moving around and scaling up and down constantly while the operator are using the ROV is distracting in itself. It is very easy that the operators attention is distracted because of all the activity happening on the screen. A good approach could be to incorporate something similar to Baldwin et al. (1999) who used cropped video from a panoramic camera. By only displaying parts of the FOV and changing this selection in response to operator controls, the video would not have to move around on the screen. The PD algorithm in this thesis can easily be altered to this kind of behavior. The disadvantages of such a method is that it would require a wider FOV camera which is typically more expensive. In addition, by only displaying parts of the video the displayed resolution will drop. This can be accommodated by sending a higher resolution image, but this would require more processing power and possibly increase the video latency.

Secondly, many of the operators used the physical black peg as visual guidance even though the red arrow was included. This meant that the operator frequently overshot the target and in practice did not use the predictor. In a case where the peg would not be needed, like an ordinary obstacle course, the subjects only visual aid would have been the red arrow. This would presumably make them use it much more, and reduce the amount of overshoot.

Although the predictive method is model free and can be used on all maneuvering ROVs, the pixel turn/scale rate mentioned in section 3.4 has to be found to use the predictive screen. There is however a way to make the predictive screen work without the need for *any* additional information. By tracking objects in the video a comparison can be made between the predicted movement of objects versus the actual movements. By constantly doing this comparison, the pixel turn/scale rate can be automatically adjusted to minimize the discrepancies. This method can also be used to automatically detect the communication latency. By comparing the time when commands are given and when objects starts to move the delay can be found. Object tracking can be performed using the OpenCV software. This approach would would require a more advanced algorithm and also use more processing power.

References

Kumcu, Asli et al. (June 2017). “Effect of video lag on laparoscopic surgery: correlation between performance and usability at low latencies”. In: *International Journal of Medical Robotics and Computer Assisted Surgery* 13.2, e1758. ISSN: 1478596X. DOI: 10.1002/rcs.1758. arXiv: 1504.07874. URL: http://doi.wiley.com/10.1002/rcs.1758.

Bejczy, A.K. et al. (1990). *The phantom robot: predictive displays for teleoperation with time delay.* DOI: 10.1109/ROBOT.1990.126037.

Draper, John V et al. (1998). *Telepresence.* Vol. 40. 3, pp. 354–375.

Chen, J.Y.C. et al. (2007). “Human performance issues and user interface design for teleoperated robots”. In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 37.6, pp. 1231–1245. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2007.905819. URL: http://ieeexplore.ieee.org/xpls/abs%7B%5C_%7Dall.jsp?arnumber=4343985.

Arthur, Kevin W. et al. (1993). “Evaluating 3D task performance for fish tank virtual worlds”. In: *ACM Transactions on Information Systems* 11.3, pp. 239–265. ISSN: 10468188. DOI: 10.1145/159161.155359. URL: http://portal.acm.org/citation.cfm?doid=159161.155359.

Appelqvist, Pekka et al. (2007). “Development of an unmanned ground vehicle for task-oriented operation - Considerations on teleoperation and delay”. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM.* DOI: 10.1109/AIM.2007.4412567.

Matheson, Adrian et al. (2013). “The effects of predictive displays on performance in driving tasks with multi-second latency: Aiding tele-operation of lunar rovers”. In: *Proceedings of the Human Factors and Ergonomics Society*, pp. 21–25. ISSN: 10711813. DOI: 10.1177/1541931213571007.

Ricks, Bob et al. (2004). “Ecological displays for robot interaction: a new perspective”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* 3, pp. 2855–2860. DOI: 10.1109/IROS.2004.1389842. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1389842.

Nielsen, Curtis W. et al. (2007). “Ecological interfaces for improving mobile robot teleoperation”. In: *IEEE Transactions on Robotics* 23.5, pp. 927–941. ISSN: 15523098. DOI: 10.1109/TRO.2007.907479.

Ando, Noriaki et al. (1999). “A Study on Influence of Time Delay in Teleoperation”. In: *Proceedings of the 1999 IEEUASME*.

Lane, J Corde et al. (2002). “Effects of Time Delay on Telerobotic Contorl of Neutral Buoyancy Vehicles”. In: *Proceedings of the 2002 IEEE International Conference on Robotics & Automation* May.

Xu, Song et al. (Sept. 2014). “Determination of the latency effects on surgical performance and the acceptable latency levels in telesurgery using the dV-Trainer® simulator”. In: *Surgical Endoscopy* 28.9, pp. 2569–2576. ISSN: 0930-2794. DOI: 10.1007/s00464-014-3504-z. URL: http://link.springer.com/10.1007/s00464-014-3504-z.

Fabrizio, M D et al. (2000). “Effect of time delay on surgical performance during telesurgical manipulation.” In: *Journal of Endourology* 14.2, pp. 133–8. ISSN: 0892-7790. DOI: 10.1089/end.2000.14.133. URL: http://www.liebertonline.com/doi/abs/10.1089/end.2000.14.133%7B%5C%%7D0Ahttp://www.ncbi.nlm.nih.gov/pubmed/10772504.

Perez, Manuela et al. (Apr. 2016). “Impact of delay on telesurgical performance: study on the robotic simulator dV-Trainer”. In: *International Journal of Computer Assisted Radiology and Surgery* 11.4, pp. 581–587. ISSN: 1861-6410. DOI: 10.1007/s11548-015-1306-y. URL: http://link.springer.com/10.1007/s11548-015-1306-y.

Lum, Mitchell J.H. et al. (Sept. 2009). “Teleoperation in surgical robotics - Network latency effects on surgical performance”. In: *Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society: Engineering the Future of Biomedicine, EMBC 2009*. IEEE, pp. 6860–6863. ISBN: 9781424432967. DOI: 10.1109/IEMBS.2009.5333120. URL: http://ieeexplore.ieee.org/document/5333120/.

MacKenzie, Scott et al. (1993). “Lag as a Determinant of Human Performance in Interactive Systems”. In: *Proceedings of the ACM Conference on Human*

Factors in Computing Systems. URL: http://www.yorku.ca/mack/CHI93b.html.

Luck, Jason P et al. (2006). "An investigation of real world control of robotic assets under communication latency". In: *HRI '06: Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pp. 202–209. DOI: 10.1145/1121241.1121277. URL: http://doi.acm.org/10.1145/1121241.1121277.

Dorais, Gregory A. et al. (1999). "Adjustable Autonomy for Human-Centered Autonomous Systems". In: *Sixteenth International Joint Conference on Artificial Intelligence Workshop on Adjustable Autonomy Systems* May 2003, pp. 16–35.

Goodrich, Michael A et al. (2001). "Experiments in Adjustable Autonomy Robot Effectiveness Neglect". In: *IJCAI workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, pp. 1624–1629. URL: http://jcrandall.faculty.masdar.ac.ae/jcrandall/IJCAI01.pdf.

Miller, David P. et al. (2005). "Visual aids for lunar rover tele-operation". In: *European Space Agency, (Special Publication) ESA SP* 603, pp. 557–562. ISSN: 03796566.

Lu, Shihan et al. (2018). "Effects of a Delay Compensation Aid on Teleoperation of Unmanned Ground Vehicles". In: pp. 179–180. ISSN: 21672148. DOI: 10.1145/3173386.3177064.

Hu, Huan et al. (2016). "Performance of Predictive Display Teleoperation under Different Delays with Different Degree of Freedoms". In: *2016 International Conference on Information System and Artificial Intelligence Performance*, pp. 2–6. DOI: 10.1109/ISAI.2016.108.

Zheng, Yingshi et al. (2016). "An Experimental Evaluation of a Model-Free Predictor Framework in Teleoperated Vehicles". In: *IFAC-PapersOnLine* 49.10, pp. 157–164. ISSN: 24058963. DOI: 10.1016/j.ifacol.2016.07.513.

Lovi, David et al. (2010). "Predictive display for mobile manipulators in unknown environments using online vision-based monocular modeling and localization". In: *IEEE/RSJ 2010 International Conference on Intelligent Robots and Sys-*

tems, IROS 2010 - Conference Proceedings April 2014, pp. 5792–5798. ISSN: 2153-0858. DOI: 10.1109/IROS.2010.5649522.

Rachmielowski, Adam et al. (2010). "Performance evaluation of monocular predictive display". In: *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5309–5314. ISSN: 10504729. DOI: 10.1109/ROBOT.2010.5509652.

Mathan, Santosh et al. (1996). "Efficacy of a predictive display, steering device, and vehicle body representation in the operation of a lunar vehicle". In: *Conference companion on Human factors in computing systems common ground - CHI '96*, pp. 71–72. DOI: 10.1145/257089.257147. URL: http://portal.acm.org/citation.cfm?doid=257089.257147.

Zhang, Yakun et al. (2017). "Handling qualities evaluation of predictive display model for rendezvous and docking in lunar orbit with large time delay". In: *CGNCC 2016 - 2016 IEEE Chinese Guidance, Navigation and Control Conference*, pp. 742–747. ISBN: 9781467383189. DOI: 10.1109/CGNCC.2016.7828878.

Grunwald, a. J. et al. (1981). "Experimental evaluation of a perspective tunnel display for three-dimensional helicopter approaches". In: 4.6, pp. 623–631. ISSN: 0731-5090. DOI: 10.2514/3.56119.

Baldwin, J. et al. (1999). "Panoramic video with predictive windows for telepresence applications". In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)* 3.May, pp. 1922–1927. ISSN: 10504729. DOI: 10.1109/ROBOT.1999.770389. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=770389.

Jennehag, Ulf et al. (2016). "Low Delay Video Streaming on the Internet of Things Using Raspberry Pi". In: *Electronics* 60.3. DOI: 10.3390/electronics5030060. URL: %7Bhttp://www.mdpi.com/2079-9292/5/3/60%7D.

Fitts, Paul M (1954). "The Information Capacity of the Human Motor system in controlling the amplitude of movement". In: *Journal of Experimental Biology* 47.6, pp. 381–391. ISSN: 0022-1015. DOI: 10.1037/h0055392. URL: http://www.ncbi.nlm.nih.gov/pubmed/13174710.

Hart, Sandra G et al. (1988). "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research". In: *Advances in Psychology* 52, pp. 139–183.

Allwood, Michael (2008). "The Satterthwaite Formula for Degrees of Freedom in the Two-Sample t-Test". In: *AP Statistics*.

Appendices

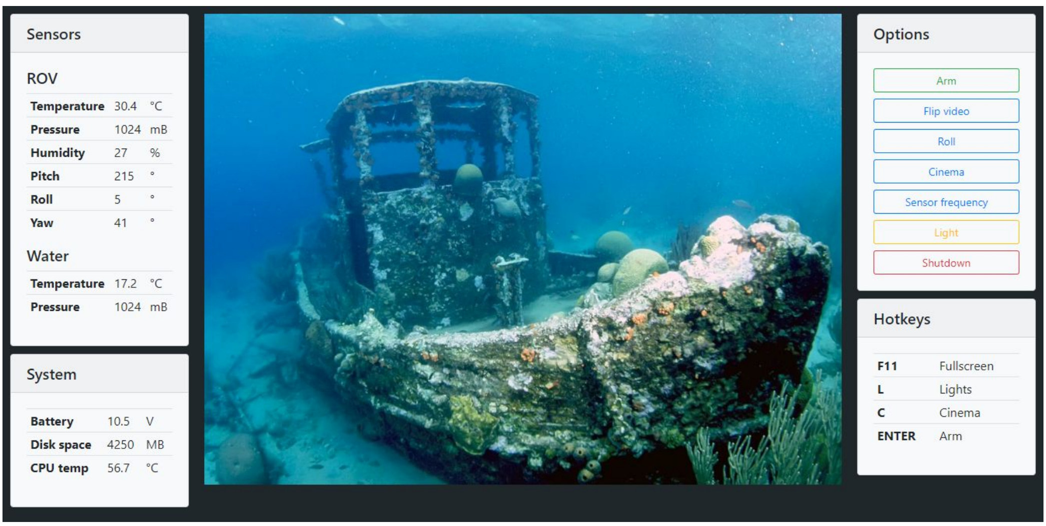
eduROV documentation

Introduction

Stream camera feed from a Raspberry Pi camera to any web browser on the network. Control the robot with your keyboard directly in the browser.

The eduROV project is all about spreading the joy of technology and learning. The eduROV is being developed as a DIY ROV kit meant to be affordable and usable by schools, hobbyists, researchers and others as they see fit. We are committed to be fully open-source, both software and hardware-wise, everything we develop will be available to you. Using other open-source and or open-access tools and platforms.

GitHub:	https://github.com/trolllabs/eduROV
PyPI:	https://pypi.org/project/edurov/
Documentation:	http://edurov.readthedocs.io
Engage eduROV:	https://www.edurov.no/



Main features

1. **Low video latency**

You can stream HD video from the Raspberry Pi camera to any unit on the same network with a video delay below 200ms.

2. **No setup required**

   The package works by displaying the video feed and other content in a web browser. This means that you can use any device to display your interface.

3. **Very easy to use**

   With the exception of Pyro4 (which is installed automatically), edurov doesn't require any other packages or software. Everything is written in python and html. 4 lines of code is everything needed to get started!

4. **Highly customizable**

   Since you define the html page yourself, you can make it look and work exactly the way you want. Use css and javascript as much as you want.

5. **True parallelism**

   Need to control motors, read sensor values and display video feed at the same time? edurov can spawn your functions on multiple CPU cores while still maintaining the possibility to share variables.

Prerequisites

- eduROV requires python 3, if you don't have python installed, you can download it here: https://www.python.org/downloads/
- the camera on the raspberry pi has to be enabled, see https://www.raspberrypi.org/documentation/configuration/camera.md

Installation

Run the following commands in a terminal on the Raspberry Pi.:

```
sudo pip3 install edurov
```

For a more in depth description visit the official documentation.

Usage

Engage eduROV submersible

On the Raspberry Pi, run the following command:

```
edurov-web
```

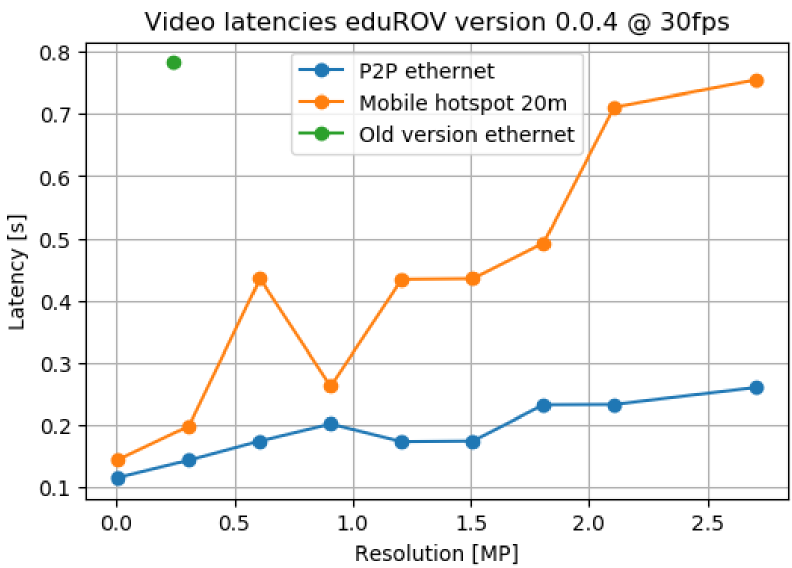
This will start the web server and print the ip where the web page can be viewed, e.g. `Visit the webpage at 192.168.0.197:8000`.

Create your own

The eduROV package includes multiple classes and functions to facilitate easy robot communication with video feed. It will get you up and running in a matter of minutes. Visit the official documentation for a *getting started*, examples and API.

Performance

The eduROV package were created with a strong focus on keeping the latency at a minimum. When deploying on a wireless network the actual performance will vary depending on factors such as distance, interference and hardware.



MP	Resolution
0.007	100x75
0.307	640x480
0.608	900x675
0.907	1100x825
1.208	1268x951
1.508	1417x1063
1.808	1552x1164
2.107	1676x1257
2.708	1900x1425

Author

The package is created by Martin Løland as part of the master thesis at Norwegian University of Science and Technology 2018

Installation

Raspbian

First, you will need a raspberry pi with an operating system running on it. Visit the official software guide for a step by step guide on how to do that..

Remote control

In most cases it is more practical to control the Raspberry Pi using another computer. The two most popular methods are with either SSH or VNC.

Update system

Make sure that your Raspberry Pi is up to date:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

Python version

The edurov package requires python 3. If python 3 si not your default python version (check by running `python --version`), you can either (1) change the default python version, or (2) use pip3 and python3 instead.

1. **Change default python version**

   Take a look at this page.
2. **Use pip3 and python3**

   If you don't want to make any changes, you can call `pip3` instead of `pip` and `python3` instead of `python`. This will use version 3 when installing and running python scripts instead.

Install using pip

Install edurov, sudo rights are needed to enable console scripts:

```
sudo pip install edurov
```

Static IP

If you are remotely connected to the Pi it can be very useful with a static ip so that you can find the Pi on the network. How you should configure this depends how your network is setup. A guide can be found here.

Start at system startup

If you want the edurov-web command to run automatically when the raspberry pi has started. Run the following command:

```
sudo nano /etc/rc.local
```

Then add the following line to the bottom of the screen, but *before* the line that says `exit 0`:

```
edurov-web &
```

Exit and save by pressing CTRL+C, y, ENTER. The system then needs to be rebooted:

```
sudo shutdown -r now
```

Engage eduROV

Terminal command

By calling `edurov-web` in the terminal the edurov-web example will be launched. This command also supports multiple flags that can be displayed by running `edurov-web -h`

`-r`	resolution, use format WIDTHxHEIGHT (default 1024x768)
`-fps`	framerate for the camera (default 30)
`-port`	which port the server should serve it's content (default 8000)
`-d`	set to print debug information

Example

```
edurov-web -r 640x480 -fps 10
```

Will then set the the video to 640x480 @ 10 fps

Getting started

Tip

If you came here to find out how to to use the Engage ROV submersible, the Engage eduROV page is probably for you. If you instead plan to create your own ROV or make some kind of modifications, you are in the right place.

Note

Not all details at explained on this page. You should check the API page for more information on the classes, methods and parameters when you need.

On this page we will walk through the features example, one feature at a time. This example was created with the intention of describing all the features of the edurov package. Let's get started!

Displaying the video feed

There are two main parts needed in any edurov project. First, it's the python file that creates the `WebMethod` class and starts serving the server. Secondly, a index.html file that describes how the different objects will be displayed in the browser.

In the two code blocks underneath you can see how simple they can be created. The index.html file needs to be called exactly this. We use the `os.path()` library to ensure correct file path description.

features.py¶

```python
import os
from edurov import WebMethod

# Create the WebMethod class
web_method = WebMethod(
    index_file=os.path.join(os.path.dirname(__file__), 'index.html'),
)
# Start serving the web page, blocks the program after this point
web_method.serve()
```

The index.html file must have an img element with `src="stream.mjpg"`. The server will then populate this image with the one coming from the camera.

index.html¶

```
<!DOCTYPE html>
<html>
<head>
    <title>Features</title>
</head>
<body>
        <img src="stream.mjpg">
</body>
</html>
```

Our file structure now looks like this:

```
project
├── features.py
└── index.html
```

If you wanted to have a security camera system this is all you had to do. If you instead want to control you robot through the browser or display other information, keep reading.

Moving a robot

This section will let us control the ROV from within the web browser. In computer technology there is something called *parallelism*. It basically means that the CPU does multiple things at the same time in different processes. This is an important feature of the edurov package as it let's us do many things without interrupting the video feed. (It wouldn't be very practical if the video stopped each time we moved the robot).

Reading keystrokes

First, we have to ask the browser to send us information when keys are pressed. We do this by including `keys.js` inside the `index.html` file. We have put it inside a folder called *static* as this is the convention for these kind of files.

index.html¶

```html
<!DOCTYPE html>
<html>
<head>
    <title>Features</title>
    <script src="./static/keys.js"></script>
</head>
<body>
        <img src="stream.mjpg">
</body>
</html>
```

/static/keys.js¶

```javascript
var last_key;

document.onkeydown = function(evt) {
    evt = evt || window.event;
    if (evt.keyCode != last_key){
        last_key = evt.keyCode;
        send_keydown(evt.keyCode);
    }
}

document.onkeyup = function(evt) {
    last_key = 0;
    send_keyup(evt.keyCode);
}

function send_keydown(keycode){
    var xhttp = new XMLHttpRequest();
    xhttp.open("GET", "/keydown="+keycode, true);
    xhttp.setRequestHeader("Content-Type", "text/html");
    xhttp.send(null);
}

function send_keyup(keycode){
    var xhttp = new XMLHttpRequest();
    xhttp.open("GET", "/keyup="+keycode, true);
    xhttp.setRequestHeader("Content-Type", "text/html");
    xhttp.send(null);
}
```

Controlling motors (or anything)

In this example we will not show how to move the motors, instead the program will print out which arrow key you are pressing. You can then change the code to do whatever you want!

features.py¶

```python
1  import os
2  import Pyro4
3  from edurov import WebMethod
4  
5  def control_motors():
6      """Will be started in parallel by the WebMethod class"""
7      with Pyro4.Proxy("PYRONAME:KeyManager") as keys:
8          with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
9              while rov.run:
10                 if keys.state('K_UP'):
11                     print('Forward')
12                 elif keys.state('K_DOWN'):
13                     print('Backward')
14                 elif keys.state('K_RIGHT'):
15                     print('Right')
16                 elif keys.state('K_LEFT'):
17                     print('left')
18 
19 # Create the WebMethod class
20 web_method = WebMethod(
21     index_file=os.path.join(os.path.dirname(__file__), 'index.html'),
22     runtime_functions=control_motors,
23 )
24 # Start serving the web page, blocks the program after this point
25 web_method.serve()
```

On line 22 we are telling the `WebMethod` that `control_motors` should be a `runtime_function`. This starts the function in another process and shuts it down when we stop the ROV. For more information visit the API page. Since this function is running in another process it needs to communicate with the server. It does this by the help of `Pyro4` (line 2). We then connect to the `KeyManager` and `ROVSyncer` on line 7-8. This let's us access the variables we need.

The resulting file structure:

```
project
├── features.py
├── index.html
└── static
    └── keys.js
```

Making it pretty

At this point our web page is very boring. It is white with one image. Since it's a html file we can add whatever we want to it! This time we are adding a header, a button to stop the server and some information. In addition we are adding some styling that will center the content and make it look nicer.

index.html¶

```html
1  <!DOCTYPE html>
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Features</title>
    <link rel="stylesheet" type="text/css" href="./static/style.css">
    <script src="./static/keys.js"></script>
</head>
<body>
    <main>
        <h2>Welcome to the features example</h2>
        <img src="stream.mjpg">
        <p>
            <a href="stop">Stop server</a>
        </p>
        <p>
            Use arrow keys to print statements in the terminal window.
        </p>
    </main>
</body>
</html>
```

/static/style.css¶

```css
body {
    margin: 0;
    padding: 0;
    font-family: Verdana;
}
a {
    text-decoration: none;
}
img {
    width: 100%;
    height: auto;
}
main{
    width: 700px;
    margin-top: 20px;
    margin-left: auto;
    margin-right: auto;
}
```

```
project
├── features.py
├── index.html
└── static
    ├── keys.js
    └── style.css
```

Displaying sensor values

Coming soon

Custom responses

In some cases you want to display information in the browser that you want to create yourself in a python function. The `WebMethod` has a parameter exactly for this purpose.

features.py¶

```python
import os
import subprocess

import Pyro4

from edurov import WebMethod


def my_response(not_used, path):
    """Will be called by the web server if it not able to process by itself"""
    if path.startswith('/cpu_temp'):
        cmds = ['/opt/vc/bin/vcgencmd', 'measure_temp']
        return subprocess.check_output(cmds).decode()
    else:
        return None


def control_motors():
    """Will be started in parallel by the WebMethod class"""
    with Pyro4.Proxy("PYRONAME:KeyManager") as keys:
        with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
            while rov.run:
                if keys.state('K_UP'):
                    print('Forward')
                elif keys.state('K_DOWN'):
                    print('Backward')
                elif keys.state('K_RIGHT'):
                    print('Right')
                elif keys.state('K_LEFT'):
                    print('left')


# Create the WebMethod class
web_method = WebMethod(
    index_file=os.path.join(os.path.dirname(__file__), 'index.html'),
    runtime_functions=control_motors,
    custom_response=my_response
)
# Start serving the web page, blocks the program after this point
web_method.serve()
```

index.html¶

```html
<!DOCTYPE html>
```

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <title>Features</title>
5       <link rel="stylesheet" type="text/css" href="./static/style.css">
6       <script src="./static/keys.js"></script>
7       <script src="./static/extra.js"></script>
8   </head>
9   <body>
10      <main>
11          <h2>Welcome to the features example</h2>
12          <img src="stream.mjpg">
13          <p>
14              <a href="stop">Stop server</a>
15              <button onclick="cpuTemp()">Display CPU temp</button>
16          </p>
17          <p>
18              Use arrow keys to print statements in the terminal window.
19          </p>
20      </main>
21  </body>
22  </html>
```

/static/extra.js¶

```
1   function cpuTemp(){
2       var xhttp = new XMLHttpRequest();
3       xhttp.onreadystatechange = function() {
4           if (this.readyState == 4 && this.stat == 200) {
5               alert('The CPU temperature is '+this.responseText);
6       };
7       xhttp.open("GET", "cpu_temp", true);
8       xhttp.send();
9   }
```

As an example we have created a button in `index.html` (line 15) which calls a function in `extra.js` that asks the server what the CPU temperature is. The new .js file is included as usual (`index.html` (line 7)). On line 7 in `extra.js` we send a GET request with a value of *cpu_temp*. The server does not know how it should answer this request, but since we have defined a `custom_response` (line 37) in `features.py` the request is forwarded to this function and we can create the response our self!

Note that this function needs to accept *two* parameters whereas the last one is path that is requested. If the path starts with `/cpu_temp` we can return the value, else return `None`.

```
project
├── features.py
├── index.html
└── static
    ├── keys.js
    ├── style.css
    └── extra.js
```

Examples

Tip

The following examples can be downloaded from the eduROV examples folder.

Minimal working code

This is a bare minimum example so that the image stream and nothing more can be seen in the browser. A great starting point if you want to expand the functionality yourself.

minimal.py¶

```
from os import path

from edurov import WebMethod

web_method = WebMethod(
    index_file=path.join(path.dirname(__file__), 'index.html')
)
web_method.serve()
```

index.html¶

```
<!DOCTYPE html>
<html>
<head>
    <title>Minimal</title>
</head>
<body>
    <img src="stream.mjpg" style="transform:rotate(180deg)">
    <a href="stop">Stop Server</a>
</body>
</html>
```

```
project
├── minimal.py
└── index.html
```

Features

An example created to explain most of the features in the edurov package. See the *Getting started* page in the official documentation for a full walkthrough.

features.py¶

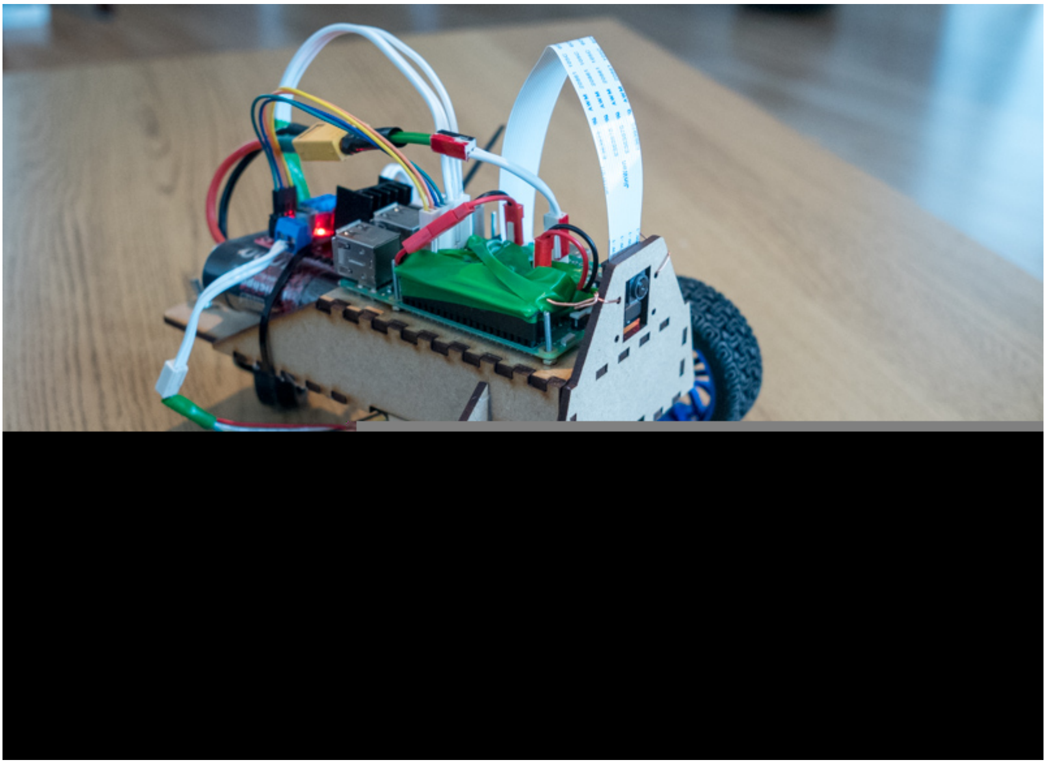
```
import os
import subprocess

import Pyro4

from edurov import WebMethod


def my_response(not_used, path):
    """Will be called by the web server if it not able to process by itself"""
    if path.startswith('/cpu_temp'):
        cmds = ['/opt/vc/bin/vcgencmd', 'measure_temp']
        return subprocess.check_output(cmds).decode()
    else:
        return None


def control_motors():
    """Will be started in parallel by the WebMethod class"""
    with Pyro4.Proxy("PYRONAME:KeyManager") as keys:
        with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
            while rov.run:
                if keys.state('K_UP'):
                    print('Forward')
                elif keys.state('K_DOWN'):
                    print('Backward')
                elif keys.state('K_RIGHT'):
                    print('Right')
                elif keys.state('K_LEFT'):
                    print('left')


# Create the WebMethod class
web_method = WebMethod(
    index_file=os.path.join(os.path.dirname(__file__), 'index.html'),
    runtime_functions=control_motors,
    custom_response=my_response
)
# Start serving the web page, blocks the program after this point
web_method.serve()
```

index.html¶

```html
<!DOCTYPE html>
<html>
<head>
    <title>Features</title>
    <link rel="stylesheet" type="text/css" href="./static/style.css">
    <script src="./static/keys.js"></script>
    <script src="./static/extra.js"></script>
</head>
<body>
    <main>
        <h2>Welcome to the features example</h2>
        <img src="stream.mjpg">
        <p>
            <a href="stop">Stop server</a>
            <button onclick="cpuTemp()">Display CPU temp</button>
        </p>
        <p>
            Use arrow keys to print statements in the terminal window.
        </p>
    </main>
</body>
</html>
```

```
project
├── features.py
├── index.html
└── static
    ├── keys.js
    ├── extra.js
    └── style.css
```

Wireless RC car with camera feed



Create your very own wireless RC car with camera! The streaming video can be viewed in a browser on any device on the same network, it is controlled by using the arrow keys on the keyboard.

Bill of materials

Name	Price USD	Comment
Raspberry Pi Zero WH	18	A full size board can also be used
Raspberry Pi Camera Module V2	33	
DC 6V 210RPM Geard Motor Wheel Kit	23	found on eBay
L298N Dual H Bridge Motor Controller Board	1.8	found on eBay
DC-DC 5V 12V Step Down Module Converter 3A	1.6	found on eBay
Total	**76**	

In addition you will need a swivel wheel, M3/M2.5 bolts and nuts, cables and connectors, 12V battery and a car frame. The car frame used in the picture above was cut from 3mm MDF with a laser cutter and can be found here.

CAD files

Visit https://grabcad.com/library/772279

```
project
├── rc_car.py
├── index.html
├── electronics.py
└── static
    └── keys.js
```

Engage eduROV

This example is used to control the ROV used in the eduROV project, see www.edurov.no.

start.py¶

```
import os
import time

import Pyro4

from edurov import WebMethod
from edurov.utils import detect_pi, serial_connection, send_arduino, \
    receive_arduino, free_drive_space, cpu_temperature

if detect_pi():
    from sense_hat import SenseHat


def valid_arduino_string(arduino_string):
    if arduino_string:
        if arduino_string.count(':') == 2:
            try:
                [float(v) for v in arduino_string.split(':')]
                return True
            except:
                return False
    return False


def arduino():
    lastState = '0000'
    ser = serial_connection()
    # 'letter': [position, value]
    config = {'w': [0, 1],
              's': [0, 2],
              'a': [1, 1],
              'q': [1, 2],
              'd': [2, 1],
              'e': [2, 2]}
    with Pyro4.Proxy("PYRONAME:KeyManager") as keys:
        with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
            keys.set_mode(key='l', mode='toggle')
            while rov.run:
                dic = keys.qweasd_dict
                states = [0, 0, 0, 0]
                for key in config:
                    if dic[key]:
                        states[config[key][0]] = config[key][1]
                states[3] = int(keys.state('l'))
                state = ''.join([str(n) for n in states])
                if state != lastState:
                    lastState = state
                    if ser:
                        send_arduino(msg=state, serial_connection=ser)
                    else:
                        print(state)
                if ser:
                    arduino_string = receive_arduino(serial_connection=ser)
                    if valid_arduino_string(arduino_string):
                        v1, v2, v3 = arduino_string.split(':')
                        rov.sensor = {
                            'tempWater': float(v1),
                            'pressureWater': float(v2),
                            'batteryVoltage': float(v3)
                        }


def senser():
    sense = SenseHat()
    with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
        while rov.run:
            orientation = sense.get_orientation()
            rov.sensor = {'temp': sense.get_temperature(),
                          'pressure': sense.get_pressure() / 10,
```

```
                          'humidity': sense.get_humidity(),
                          'pitch': orientation['pitch'],
                          'roll': orientation['roll'] + 180,
                          'yaw': orientation['yaw']}


def system_monitor():
    with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
        while rov.run:
            rov.sensor = {'free_space': free_drive_space(),
                          'cpu_temp': cpu_temperature()}
            time.sleep(10)


def main(video_resolution='1024x768', fps=30, server_port=8000, debug=False):
    web_method = WebMethod(
        index_file=os.path.join(os.path.dirname(__file__), 'index.html'),
        video_resolution=video_resolution,
        fps=fps,
        server_port=server_port,
        debug=debug,
        runtime_functions=[arduino, senser, system_monitor]
    )
    web_method.serve()


if __name__ == '__main__':
    main()
```

index.html¶

```
<html>
<head>
    <title>eduROV</title>
    <script src="./static/dynamic.js"></script>
    <script src="./static/general.js"></script>
    <script src="./static/keys.js"></script>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
    <link rel="icon" href="favicon.ico" type="image/x-icon">
    <link rel="stylesheet" type="text/css" href="./static/style.css">
    <link rel="stylesheet" type="text/css" href="./static/bootstrap.css">
</head>
<body onload="set_size()">

<div class="grid-container">
    <div class="d-none d-md-block side-panel " style="display:none;">
        <div class="card bg-light cinema">
            <h5 class="card-header">Sensors</h5>
            <div class="card-body">
                <h5>ROV</h5>
                <table class="table table-hover table-sm">
                    <tbody>
                    <tr>
                        <th scope="row">Temperature</th>
                        <td id="temp"></td>
                        <td>&#8451</td>
                    </tr>
                    <tr>
                        <th scope="row">Pressure</th>
                        <td id="pressure"></td>
                        <td>kPa</td>
                    </tr>
                    <tr>
                        <th scope="row">Humidity</th>
                        <td id="humidity"></td>
                        <td>%</td>
                    </tr>
                    <tr>
                        <th scope="row">Pitch</th>
                        <td id="pitch"></td>
                        <td>&#176</td>
                    </tr>
                    <tr>
                        <th scope="row">Roll</th>
                        <td id="roll"></td>
                        <td>&#176</td>
                    </tr>
                    <tr>
                        <th scope="row">Yaw</th>
                        <td id="yaw"></td>
                        <td>&#176</td>
                    </tr>
                    </tbody>
                </table>
                <h5>Water</h5>
                <table class="table table-sm">
                    <tbody>
                    <tr>
                        <th scope="row">Temperature</th>
                        <td id="tempWater"></td>
                        <td>&#8451</td>
                    </tr>
                    <tr>
                        <th scope="row">Pressure</th>
                        <td id="pressureWater"></td>
                        <td>kPa</td>
                    </tr>
                    </tbody>
                </table>
            </div>
```

```
        </div>
        <div class="card bg-light cinema">
            <h5 class="card-header">System</h5>
            <div class="card-body">
                <table class="table table-sm">
                    <tbody class="table-borderless">
                    <tr id="voltageTr">
                        <th scope="row">Battery</th>
                        <td id="batteryVoltage"></td>
                        <td>V</td>
                    </tr>
                    <tr id="diskTr">
                        <th scope="row">Disk space</th>
                        <td id="free_space"></td>
                        <td>MB</td>
                    </tr>
                    <tr id="cpuTr">
                        <th scope="row">CPU temp</th>
                        <td id="cpu_temp"></td>
                        <td>&#8451</td>
                    </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    <div class="center-panel">
        <img id="image" src="stream.mjpg">
        <img class="rollOverlay" id="rollOverlay" src="./static/roll.png">
    </div>
    <div class="d-none d-md-block side-panel">
        <div class="card bg-light cinema">
            <h5 class="card-header">Options</h5>
            <div class="card-body">
                <button type="button" onclick="toggle_armed()" id="armBtn"
                        class="btn btn-outline-success btn-sm btn-block"
                        title="Use this to arm the robot">
                    Arm
                </button>
                <button type="button" onclick="rotate_image()"
                        class="btn btn-outline-primary btn-sm btn-block"
                        title="Will rotate the video 180 degrees">
                    Flip video
                </button>
                <button type="button" onclick="toggle_roll()" id="rollBtn"
                        class="btn btn-outline-primary btn-sm btn-block active"
                        title="Toggle the roll indicator on/off">
                    Roll
                </button>
                <button type="button" onclick="toggle_cinema()"
                        class="btn btn-outline-primary btn-sm btn-block"
                        title="Toggle cinema mode which hides everything except video">
                    Cinema
                </button>
                <button type="button" onclick="set_update_frequency()"
                        class="btn btn-outline-primary btn-sm btn-block"
                        title="Changes the sensor update frequency to desired value">
                    Sensor frequency
                </button>
                <button type="button" onclick="toggle_light()" id="lightBtn"
                        class="btn btn-outline-warning btn-sm btn-block"
                        title="Toggle the light on the ROV on/off">Light
                </button>
                <button type="button" onclick="stop_rov()"
                        class="btn btn-outline-danger btn-sm btn-block"
                        title="Stops the ROV, this page will stop working">
                    Shutdown
                </button>
            </div>
        </div>
```

```
        <div class="card bg-light cinema">
            <h5 class="card-header">Hotkeys</h5>
            <div class="card-body">
                <table class="table table-sm">
                    <tbody>
                    <tr>
                        <td><b>F11</b></td>
                        <td>Fullscreen</td>
                    </tr>
                    <tr>
                        <td><b>L</b></td>
                        <td>Lights</td>
                    </tr>
                    <tr>
                        <td><b>C</b></td>
                        <td>Cinema</td>
                    </tr>
                    <tr>
                        <td><b>ENTER</b></td>
                        <td>Arm</td>
                    </tr>
                    </tbody>
                </table>
            </div>
        </div>

    </div>
</div>

</body>
</html>
```

```
project
├── entry.py
├── start.py
├── index.html
└── static
    ├── keys.js
    ├── general.js
    ├── dynamic.js
    ├── roll.png
    ├── bootstrap.css
    └── style.css
```

API

Tip

If you are having a hard time, you can always have a look at the examples page where the classes, methods and parameters are used in practice.

WebMethod

class `edurov.core.WebMethod`(*index_file*, *video_resolution='1024x768'*, *fps=30*, *server_port=8000*, *debug=False*, *runtime_functions=None*, *custom_response=None*) [source]

Starts a video streaming from the rasparry pi and a webserver that can handle user input and other requests.

Parameters:

- **index_file** (*str*) – Absolute path to the frontpage of the webpage, must be called `index.html`. For more information, see Displaying the video feed.
- **video_resolution** (*str*, *optional*) – A string representation of the wanted video resolution in the format WIDTHxHEIGHT.
- **fps** (*int*, *optional*) – Wanted framerate, may not be achieved depending on available resources and network.
- **server_port** (*int*, *optional*) – The web page will be served at this port
- **debug** (*bool*, *optional*) – If set True, additional information will be printed for debug purposes.
- **runtime_functions** (*callable or list*, *optional*) – Should be a callable function or a list of callable functions, will be started as independent processes automatically. For more information, see Controlling motors (or anything).
- **custom_response** (*callable*, *optional*) – If set, this function will be called if default web server is not able to handle a GET request, should return a str or None. If returned value starts with `redirect=` followed by a path, the server will redirect the browser to this path. The callable must accept two parameters whereas the second one is the requested path. For more information, see Custom responses.

Examples

```
>>> import os
>>> from edurov import WebMethod
>>>
>>> file = os.path.join(os.path.dirname(__file__), 'index.html', )
>>> web_method = WebMethod(index_file=file)
>>> web_method.serve()
```

serve(*timeout=None*) [source]

Will start serving the web page defined by the index_file parameter

Parameters: **timeout** (*int*, *optional*) – if set, the web page will only be served for that many seconds before it automatically shuts down

Notes

This method will block the rest of the script.

ROVSyncer

class **edurov.sync.ROVSyncer** [source]

Holds all variables for ROV related to control and sensors

Examples

```
>>> import Pyro4
>>>
>>> with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
>>>   while rov.run:
>>>       print('The ROV is still running')
```

actuator

Dictionary holding actuator values

Getter: Returns actuator values as dict

Setter: Update actuator values with dict

Type: dict

run

Bool describing if the ROV is still running

Getter: Returns bool describing if the ROV is running

Setter: Set to False if the ROV should stop

Type: bool

`sensor`

Dictionary holding sensor values

Getter: Returns sensor values as dict

Setter: Update sensor values with dict

Type: dict

KeyManager

class `edurov.sync.KeyManager` [source]

Keeps control of all user input from keyboard.

Examples

```
>>> import Pyro4
>>>
>>> with Pyro4.Proxy("PYRONAME:KeyManager") as keys:
>>> with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
>>>     keys.set_mode(key='l', mode='toggle')
>>>     while rov.run:
>>>         if keys.state('up arrow'):
>>>             print('You are pressing the up arrow')
>>>         if keys.state('l'):
>>>             print('light on')
>>>         else:
>>>             print('light off')
```

Note

When using the methods below a **key identifier** must be used. Either the keycode (int) or the KeyASCII or Common Name (str) from the table further down on this page can be used. Using keycode is faster.

`arrow_dict`

Dictionary with the state of the keys *up arrow*, *down arrow*, *left arrow* and *right arrow*

keydown(*key*, *make_exception=False*) [source]

Call to simulate a keydown event

Parameters:
- **key** (*int or str*) – key identifier as described above
- **make_exception** (*bool, optional*) – As default an exception is raised if the key is not found, this behavior can be changed be setting it to *False*

keyup(*key, make_exception=False*) [source]

Call to simulate a keyup event

Parameters:
- **key** (*int or str*) – key identifier as described above
- **make_exception** (*bool, optional*) – As default an exception is raised if the key is not found, this behavior can be changed be setting it to *False*

qweasd_dict

Dictionary with the state of the letters q, w, e, a, s and d

set(*key, state*) [source]

Set the state of the key to True or False

Parameters:
- **key** (*int or str*) – key identifier as described above
- **state** (*bool*) – *True* or *False*

set_mode(*key, mode*) [source]

Set the press mode for the key to *hold* or *toggle*

Parameters:
- **key** (*int or str*) – key identifier as described above
- **mode** (*str*) – *hold* or *toggle*

state(*key*) [source]

Returns the state of *key*

Parameters: **key** (*int or str*) – key identifier as described above

Returns: **state** – *True* or *False*

Return type: bool

Keys table

```
KeyASCII      ASCII   Common Name           Keycode
K_BACKSPACE   \b      backspace             8
K_TAB         \t      tab                   9
K_CLEAR               clear
K_RETURN      \r      return                13
K_PAUSE               pause
K_ESCAPE      ^[      escape                27
K_SPACE               space                 32
K_EXCLAIM     !       exclaim
K_QUOTEDBL    "       quotedbl
K_HASH        #       hash
K_DOLLAR      $       dollar
K_AMPERSAND   &       ampersand
K_QUOTE               quote
K_LEFTPAREN   (       left parenthesis
K_RIGHTPAREN  )       right parenthesis
K_ASTERISK    *       asterisk
K_PLUS        +       plus sign
K_COMMA       ,       comma
K_MINUS       -       minus sign
K_PERIOD      .       period
K_SLASH       /       forward slash
K_0           0       0                     48
K_1           1       1                     49
K_2           2       2                     50
K_3           3       3                     51
K_4           4       4                     52
K_5           5       5                     53
K_6           6       6                     54
K_7           7       7                     55
K_8           8       8                     56
K_9           9       9                     57
K_COLON       :       colon
K_SEMICOLON   ;       semicolon
K_LESS        <       less-than sign
K_EQUALS      =       equals sign
K_GREATER     >       greater-than sign
K_QUESTION    ?       question mark
K_AT          @       at
K_LEFTBRACKET [       left bracket
K_BACKSLASH   \       backslash
K_RIGHTBRACKET ]      right bracket
K_CARET       ^       caret
K_UNDERSCORE  _       underscore
K_BACKQUOTE   `       grave
K_a           a       a                     65
K_b           b       b                     66
K_c           c       c                     67
K_d           d       d                     68
K_e           e       e                     69
K_f           f       f                     70
K_g           g       g                     71
K_h           h       h                     72
K_i           i       i                     73
K_j           j       j                     74
K_k           k       k                     75
K_l           l       l                     76
K_m           m       m                     77
K_n           n       n                     78
K_o           o       o                     79
K_p           p       p                     80
K_q           q       q                     81
K_r           r       r                     82
K_s           s       s                     83
K_t           t       t                     84
K_u           u       u                     85
K_v           v       v                     86
K_w           w       w                     87
K_x           x       x                     88
```

```
K_y           y       y                     89
K_z           z       z                     90
K_DELETE              delete
K_KP0                 keypad 0
K_KP1                 keypad 1
K_KP2                 keypad 2
K_KP3                 keypad 3
K_KP4                 keypad 4
K_KP5                 keypad 5
K_KP6                 keypad 6
K_KP7                 keypad 7
K_KP8                 keypad 8
K_KP9                 keypad 9
K_KP_PERIOD   .       keypad period
K_KP_DIVIDE   /       keypad divide
K_KP_MULTIPLY *       keypad multiply
K_KP_MINUS    -       keypad minus
K_KP_PLUS     +       keypad plus
K_KP_ENTER    \r      keypad enter
K_KP_EQUALS   =       keypad equals
K_UP                  up arrow              38
K_DOWN                down arrow            40
K_RIGHT               right arrow           39
K_LEFT                left arrow            37
K_INSERT              insert                45
K_HOME                home                  36
K_END                 end                   35
K_PAGEUP              page up               33
K_PAGEDOWN            page down             34
K_F1                  F1
K_F2                  F2
K_F3                  F3
K_F4                  F4
K_F5                  F5
K_F6                  F6
K_F7                  F7
K_F8                  F8
K_F9                  F9
K_F10                 F10
K_F11                 F11
K_F12                 F12
K_F13                 F13
K_F14                 F14
K_F15                 F15
K_NUMLOCK             numlock
K_CAPSLOCK            capslock
K_SCROLLOCK           scrollock
K_RSHIFT              right shift
K_LSHIFT              left shift
K_RCTRL               right control
K_LCTRL               left control
K_RALT                right alt
K_LALT                left alt
K_RMETA               right meta
K_LMETA               left meta
K_LSUPER              left Windows key
K_RSUPER              right Windows key
K_MODE                mode shift
K_HELP                help
K_PRINT               print screen
K_SYSREQ              sysrq
K_BREAK               break
K_MENU                menu
K_POWER               power
K_EURO                Euro
```

Utilities

Different utility functions practical for ROV control

edurov.utils.cpu_temperature() [source]

Checks and returns the on board CPU temperature

Returns: **temperature** – the temperature

Return type: float

edurov.utils.free_drive_space(*as_string=False*) [source]

Checks and returns the remaining free drive space

Parameters: **as_string** (*bool*, *optional*) – set to True if you want the function to return a formatted string. 4278 -> 4.28 GB

Returns: **space** – the remaining MB in float or as string if *as_string=True*

Return type: float or str

edurov.utils.receive_arduino(*serial_connection*) [source]

Returns a message received over *serial_connection*

Expects that the message received starts with a 6 bytes long number describing the size of the remaining data. "0x000bhello there" -> "hello there".

Parameters: **serial_connection** (*object*) – the `serial.Serial` object you want to use for receiving

Returns: **msg** – the message received or None

Return type: str or None

edurov.utils.receive_arduino_simple(*serial_connection*, *min_length=1*) [source]

Returns a message received over *serial_connection*

Same as `receive_arduino` but doesn't expect that the message starts with a hex number.

Parameters:

- **serial_connection** (*object*) – the `serial.Serial` object you want to use for receiving
- **min_length** (*int*, *optional*) – if you only want that the function to only return the string if it is at least this long.

Returns: **msg** – the message received or None

Return type: str or None

`edurov.utils.send_arduino`(*msg*, *serial_connection*) [source]

Send the *msg* over the *serial_connection*

Adds a hexadecimal number of 6 bytes to the start of the message before sending it. “hello there” -> “0x000bhello there”

Parameters:
- **msg** (*str or bytes*) – the message you want to send
- **serial_connection** (*object*) – the `serial.Serial` object you want to use for sending

`edurov.utils.send_arduino_simple`(*msg*, *serial_connection*) [source]

Send the *msg* over the *serial_connection*

Same as `send_arduino`, but doesn’t add anything to the message before sending it.

Parameters:
- **msg** (*str or bytes*) – the message you want to send
- **serial_connection** (*object*) – the `serial.Serial` object you want to use for sending

`edurov.utils.serial_connection`(*port='/dev/ttyACM0'*, *baudrate=115200*, *timeout=0.05*) [source]

Establishes a serial connection

Parameters:
- **port** (*str*, *optional*) – the serial port you want to use
- **baudrate** (*int*, *optional*) – the baudrate of the serial connection
- **timeout** (*float*, *optional*) – read timeout value

Returns: **connection** – a `serial.Serial` object if successful or None if not

Return type: class or None

eduROV Package Code

```python
"""
Sever classes used in the web method
"""

import io
import json
import logging
import os
import socketserver
import time
from http import server
from threading import Condition

import Pyro4

from edurov.utils import server_ip, detect_pi, warning

if detect_pi():
    import picamera


class StreamingOutput(object):
    """Defines output for the picamera, used by request server """

    def __init__(self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()
        self.count = 0

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            # New frame, copy the existing buffer's content and notify all
            # clients it's available
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
            self.count += 1
        return self.buffer.write(buf)


class RequestHandler(server.BaseHTTPRequestHandler):
```

```
    """Request server, handles request from the browser"""
    output = None
    keys = None
    rov = None
    base_folder = None
    index_file = None
    custom_response = None

    def do_GET(self):
        if self.path == '/':
            self.redirect('/index.html', redir_type=301)
        elif self.path == '/stream.mjpg':
            self.serve_stream()
        elif self.path.startswith('/http') or self.path.startswith('/www'):
            self.redirect(self.path[1:])
        elif self.path.startswith('/keyup'):
            self.send_response(200)
            self.end_headers()
            self.keys.keyup(key=int(self.path.split('=')[1]))
        elif self.path.startswith('/keydown'):
            self.send_response(200)
            self.end_headers()
            self.keys.keydown(key=int(self.path.split('=')[1]))
        elif self.path.startswith('/sensor.json'):
            self.serve_rov_data('sensor')
        elif self.path.startswith('/actuator.json'):
            self.serve_rov_data('actuator')
        elif self.path.startswith('/stop'):
            self.send_response(200)
            self.end_headers()
            self.rov.run = False
        else:
            path = os.path.join(self.base_folder, self.path[1:])
            if os.path.isfile(path):
                self.serve_path(path)
            elif self.custom_response:
                response = self.custom_response(self.path)
                if response:
                    if response.startswith('redirect='):
                        new_path = response[response.find('=') + 1:]
                        self.redirect(new_path)
                    else:
                        self.serve_content(response.encode('utf
```

```
-8'))
                else:
                    warning(message='Bad response. {}. custom '
                                    'response function returned nothing'
                            .format(self.requestline), filter='default')
                    self.send_404()
            else:
                warning(message='Bad response. {}. Could not find {}'
                        .format(self.requestline, path), filter='default')
                self.send_404()

    def do_POST(self):
        self.send_404()

    def serve_content(self, content, content_type='text/html'):
        self.send_response(200)
        self.send_header('Content-Type', content_type)
        self.send_header('Content-Length', len(content))
        self.end_headers()
        self.wfile.write(content)

    def serve_path(self, path):
        if '.css' in path:
            content_type = 'text/css'
        elif '.js' in path:
            content_type = 'text/javascript'
        else:
            content_type = 'text/html'
        with open(path, 'rb') as f:
            content = f.read()
        self.serve_content(content, content_type)

    def redirect(self, path, redir_type=302):
        self.send_response(redir_type)
        self.send_header('Location', path)
        self.end_headers()

    def send_404(self):
        self.send_error(404)
        self.end_headers()

```

```
    def serve_rov_data(self, data_type):
        values = ''
        if data_type == 'sensor':
            values = json.dumps(self.rov.sensor)
        elif data_type == 'actuator':
            values = json.dumps(self.rov.actuator)
        else:
            warning('Unable to process data_type {}'.format(data_type))
        content = values.encode('utf-8')
        self.serve_content(content, 'application/json')

    def serve_stream(self):
        self.send_response(200)
        self.send_header('Age', 0)
        self.send_header('Cache-Control', 'no-cache, private')
        self.send_header('Pragma', 'no-cache')
        self.send_header('Content-Type',
                         'multipart/x-mixed-replace; boundary=FRAME')
        self.end_headers()
        try:
            while True:
                with self.output.condition:
                    self.output.condition.wait()
                    frame = self.output.frame
                self.wfile.write(b'--FRAME\r\n')
                self.send_header('Content-Type', 'image/jpeg')
                self.send_header('Content-Length', len(frame))
                self.end_headers()
                self.wfile.write(frame)
                self.wfile.write(b'\r\n')
        except Exception as e:
            logging.warning(
                'Removed streaming client %s: %s',
                self.client_address, str(e))

    def log_message(self, format, *args):
        return


class WebpageServer(socketserver.ThreadingMixIn, server.HTTPServer):
    """Threaded HTTP server, forwards request to the RequestHandlerClass"""
    allow_reuse_address = True
```

```python
    daemon_threads = True

    def __init__(self, server_address, RequestHandlerClass, stream_output,
                 rov_proxy, keys_proxy, index_file=None, debug=False,
                 custom_response=None):
        self.start = time.time()
        self.debug = debug
        RequestHandlerClass.output = stream_output
        RequestHandlerClass.rov = rov_proxy
        RequestHandlerClass.keys = keys_proxy
        RequestHandlerClass.base_folder = os.path.abspath(
            os.path.dirname(index_file))
        RequestHandlerClass.index_file = index_file
        RequestHandlerClass.custom_response = custom_response
        super(WebpageServer, self).__init__(server_address,
                                            RequestHandlerClass)

    def __enter__(self):
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        print('Shutting down http server')
        if self.debug:
            finish = time.time()
            frame_count = self.RequestHandlerClass.output.count
            print('Sent {} images in {:.1f} seconds at {:.2f} fps'
                  .format(frame_count,
                          finish - self.start,
                          frame_count / (finish - self.start)))


def start_http_server(video_resolution, fps, server_port, index_file,
                      debug=False, custom_response=None):
    if debug:
        print('Using {} @ {} fps'.format(video_resolution, fps))

    with picamera.PiCamera(resolution=video_resolution,
                           framerate=fps) as camera, \
            Pyro4.Proxy("PYRONAME:ROVSyncer") as rov, \
```

```
            Pyro4.Proxy("PYRONAME:KeyManager") as keys:
        stream_output = StreamingOutput()
        camera.start_recording(stream_output, format='mjpeg')
        try:
            with WebpageServer(server_address=('', server_port),
                               RequestHandlerClass=RequestHandler,
                               stream_output=stream_output,
                               debug=debug,
                               rov_proxy=rov,
                               keys_proxy=keys,
                               index_file=index_file,
                               custom_response=custom_response) as s:
                print('Visit the webpage at {}'.format(server_ip(server_port)))
                s.serve_forever()
        finally:
            print('closing web server')
            camera.stop_recording()

```

```python
import os
import subprocess
import time
from multiprocessing import Process

from edurov.sync import start_sync_classes
from edurov.utils import warning, preexec_function, detect_pi
from edurov.web import start_http_server

if detect_pi():
    import Pyro4

class WebMethod(object):
    """
    Starts a video streaming from the rasparry pi and a webserver that can
    handle user input and other requests.

    Parameters
    ----------
    index_file : str
        absolute path to the frontpage of the webpage, must be called
        ``index.html``
    video_resolution : str, optional
        a string representation of the wanted video resolution in the format
        WIDTHxHEIGHT
    fps : int, optional
        wanted framerate, may not be achieved depending on available resources
        and network
    server_port : int, optional
        the web page will be served at this port
    debug : bool, optional
        if set True, additional information will be printed for debug
        purposes
    runtime_functions : callable or list, optional
        should be a callable function or a list of callable functions, will be
        started as independent processes automatically
    custom_response : callable, optional
        if set, this function will be called if default web server is not able
        to handle a GET request, should return a str or None. If
```

```
 returned value
        starts with ``redirect=`` followed by a path, the browser wil redirect
        the user to this path. The callable must accept two parameters whereas
        the second one is the requested path

    Examples
    --------
    >>> import os
    >>> from edurov import WebMethod
    >>>
    >>> file = os.path.join(os.path.dirname(__file__), 'index.html', )
    >>> web_method = WebMethod(index_file=file)
    >>> web_method.serve()
    """
    def __init__(self, index_file, video_resolution='1024x768', fps=30,
                 server_port=8000, debug=False, runtime_functions=None,
                 custom_response=None):

        self.res = video_resolution
        self.fps = fps
        self.server_port = server_port
        self.debug = debug
        self.run_funcs = self._valid_runtime_functions(runtime_functions)
        self.cust_resp = self._valid_custom_response(custom_response)
        self.index_file = self._valid_index_file(index_file)

    def _valid_custom_response(self, custom_response):
        if custom_response:
            if not callable(custom_response):
                warning('custom_response parameter has to be a callable '
                        'function, not type {}'.format(type(custom_response)))
                return None
        return custom_response

    def _valid_runtime_functions(self, runtime_functions):
        if runtime_functions:
            if callable(runtime_functions):
```

```
                runtime_functions = [runtime_functions]
            elif isinstance(runtime_functions, list):
                for f in runtime_functions:
                    if not callable(f):
                        warning(
                            'Parameter runtime_functions has to be a function '
                            'or a list of functions, not {}'.format(type(f)))
            else:
                warning('Parameter runtime_functions has to be a function '
                        'or a list of functions, not {}'
                        .format(type(runtime_functions)))
        return runtime_functions

    def _valid_index_file(self, file_path):
        if not 'index.html' in file_path:
            warning('The index files must be called "index.html')
        if os.path.isfile(file_path):
            return os.path.abspath(file_path)
        else:
            warning('could not find "{}", needs absolute path'
                    .format(file_path))
        return None

    def serve(self, timeout=None):
        """
        Will start serving the web page defined by the index_file parameter

        Parameters
        ----------
        timeout : int, optional
            if set, the web page will only be served for that many seconds
            before it automatically shuts down

        Notes
        -----
        This method will block the rest of the script.
        """
        start = time.time()
        name_server = subprocess.Popen('pyro4-ns', shell=False,
                                       preexec_fn=
```

```
preexec_function)
        time.sleep(2)
        pyro_classes = Process(target=start_sync_classes)
        pyro_classes.start()
        time.sleep(4)
        web_server = Process(
            target=start_http_server,
            args=(self.res, self.fps, self.server_port, self.index_file,
                   self.debug, self.cust_resp))
        web_server.daemon = True
        web_server.start()
        processes = []
        if self.run_funcs:
            for f in self.run_funcs:
                p = Process(target=f)
                p.daemon = True
                p.start()
                processes.append(p)

        with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
            try:
                while rov.run:
                    if timeout:
                        if time.time() - start >= timeout:
                            break
            except KeyboardInterrupt:
                pass
            finally:
                print('Shutting down')
                web_server.terminate()
                rov.run = False
                if self.run_funcs:
                    for p in processes:
                        p.join(3)
                pyro_classes.terminate()
                name_server.terminate()
```

```python
"""
Synchronizing the state of ROV and controller
"""

import os
import time

import Pyro4


class Key(object):
    """Manages the state of a specific key on the keyboard"""

    def __init__(self, KeyASCII, ASCII, common, keycode, mode='hold'):
        self.state = False
        self.KeyASCII = KeyASCII
        self.ASCII = ASCII
        self.common = common
        self.mode = mode
        if keycode:
            self.keycode = int(keycode)
        else:
            self.keycode = None

    def keydown(self):
        if self.mode == 'toggle':
            self.state = not self.state
        else:
            self.state = True

    def keyup(self):
        if self.mode != 'toggle':
            self.state = False

    def __str__(self):
        return str(vars(self))


@Pyro4.expose
class KeyManager(object):
    """
    Keeps control of all user input from keyboard.

    Examples
    --------
```

```
    >>> import Pyro4
    >>>
    >>> with Pyro4.Proxy("PYRONAME:KeyManager") as keys:
    >>> with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
    >>>     keys.set_mode(key='l', mode='toggle')
    >>>     while rov.run:
    >>>         if keys.state('up arrow'):
    >>>             print('You are pressing the up arrow')
    >>>         if keys.state('l'):
    >>>             print('light on')
    >>>         else:
    >>>             print('light off')

    Note
    ----
    When using the methods below a **key identifier** must be used. Either the
    keycode (int) or the KeyASCII or Common Name (str) from the table further
    down on this page can be used. Using keycode is faster.
    """

    def __init__(self):
        self.keys = {}
        cwd = os.path.dirname(os.path.abspath(__file__))
        with open(os.path.join(cwd, 'keys.txt'), 'r') as f:
            for line in f.readlines()[1:]:
                KeyASCII = line[0:14].rstrip()
                ASCII = line[14:22].rstrip()
                common = line[22:44].rstrip()
                keycode = line[44:].rstrip()
                if keycode:
                    dict_key = int(keycode)
                else:
                    dict_key = KeyASCII
                self.keys.update({
                    dict_key: Key(KeyASCII, ASCII, common, keycode)})

    def set_mode(self, key, mode):
        """
        Set the press mode for the key to *hold* or *toggle*

        Parameters
        ----------
        key : int or str
```

```
            key identifier as described above
        mode : str
            *hold* or *toggle*
        """
        self._get(key).mode = mode

    def set(self, key, state):
        """
        Set the state of the key to True or False

        Parameters
        ----------
        key : int or str
            key identifier as described above
        state : bool
            *True* or *False*
        """
        self._get(key).state = bool(state)

    def _get(self, key_idx, make_exception=True):
        """
        Returns the Key object identified by *key_idx*

        Parameters
        ----------
        key_idx : int or str
            key identifier as described above
        make_exception : bool, optional
            As default an exception is raised if the key is not found, this
            behavior can be changed be setting it to *False*

        Returns
        -------
        key : Key object
            list items is *namedtuple* of type *LiItem*
        """
        if key_idx in self.keys:
            return self.keys[key_idx]
        elif isinstance(key_idx, str):
            for dict_key in self.keys:
                if key_idx in [self.keys[dict_key].common,
                               self.keys[dict_key].KeyASCII]:
                    return self.keys[dict_key]
        if make_exception:
            raise ValueError('Could not find key {}'.format(
```

```python
key_idx))
        else:
            return None

    def state(self, key):
        """
        Returns the state of *key*

        Parameters
        ----------
        key : int or str
            key identifier as described above

        Returns
        -------
        state : bool
            *True* or *False*
        """
        return self._get(key).state

    def keydown(self, key, make_exception=False):
        """
        Call to simulate a keydown event

        Parameters
        ----------
        key : int or str
            key identifier as described above
        make_exception : bool, optional
            As default an exception is raised if the key is not found, this
            behavior can be changed be setting it to *False*
        """
        btn = self._get(key, make_exception=make_exception)
        if btn:
            btn.keydown()

    def keyup(self, key, make_exception=False):
        """
        Call to simulate a keyup event

        Parameters
        ----------
        key : int or str
            key identifier as described above
        make_exception : bool, optional
```

```
            As default an exception is raised if the key is not found, this
            behavior can be changed be setting it to *False*
        """
        btn = self._get(key, make_exception=make_exception)
        if btn:
            btn.keyup()

    @property
    def qweasd_dict(self):
        """
        Dictionary with the state of the letters q, w, e, a, s and d
        """
        state = {
            'q': self._get(81).state,
            'w': self._get(87).state,
            'e': self._get(69).state,
            'a': self._get(65).state,
            's': self._get(83).state,
            'd': self._get(68).state,
        }
        return state

    @property
    def arrow_dict(self):
        """
        Dictionary with the state of the keys *up arrow*, *down arrow*,
        *left arrow* and *right arrow*
        """
        state = {
            'up arrow': self._get(38).state,
            'down arrow': self._get(40).state,
            'left arrow': self._get(37).state,
            'right arrow': self._get(39).state,
        }
        return state


@Pyro4.expose
class ROVSyncer(object):
    """
    Holds all variables for ROV related to control and sensors

    Examples
```

```
    --------
    >>> import Pyro4
    >>>
    >>> with Pyro4.Proxy("PYRONAME:ROVSyncer") as rov:
    >>>   while rov.run:
    >>>       print('The ROV is still running')
    """

    def __init__(self):
        self._sensor = {'time': time.time()}
        self._actuator = {}
        self._run = True

    @property
    def sensor(self):
        """
        Dictionary holding sensor values

        :getter: Returns sensor values as dict
        :setter: Update sensor values with dict
        :type: dict
        """
        return self._sensor

    @sensor.setter
    def sensor(self, values):
        self._sensor.update(values)
        self._sensor['time'] = time.time()

    @property
    def actuator(self):
        """
        Dictionary holding actuator values

        :getter: Returns actuator values as dict
        :setter: Update actuator values with dict
        :type: dict
        """
        return self._actuator

    @actuator.setter
    def actuator(self, values):
        self._actuator.update(values)
        self._actuator['time'] = time.time()

    @property
```

```
    def run(self):
        """
        Bool describing if the ROV is still running

        :getter: Returns bool describing if the ROV is running
        :setter: Set to False if the ROV should stop
        :type: bool
        """
        return self._run

    @run.setter
    def run(self, bool_):
        self._run = bool_


def start_sync_classes():
    """Registers pyro classes in name server and starts request loop"""
    rov = ROVSyncer()
    keys = KeyManager()
    with Pyro4.Daemon() as daemon:
        rov_uri = daemon.register(rov)
        keys_uri = daemon.register(keys)
        with Pyro4.locateNS() as ns:
            ns.register("ROVSyncer", rov_uri)
            ns.register("KeyManager", keys_uri)
        daemon.requestLoop()


if __name__ == "__main__":
    start_sync_classes()

```

KeyASCII	ASCII	Common Name	Keycode
K_BACKSPACE	\b	backspace	8
K_TAB	\t	tab	9
K_CLEAR		clear	
K_RETURN	\r	return	13
K_PAUSE		pause	
K_ESCAPE	^[	escape	27
K_SPACE		space	32
K_EXCLAIM	!	exclaim	
K_QUOTEDBL	"	quotedbl	
K_HASH	#	hash	
K_DOLLAR	$	dollar	
K_AMPERSAND	&	ampersand	
K_QUOTE		quote	
K_LEFTPAREN	(	left parenthesis	
K_RIGHTPAREN	)	right parenthesis	
K_ASTERISK	*	asterisk	
K_PLUS	+	plus sign	
K_COMMA	,	comma	
K_MINUS	-	minus sign	
K_PERIOD	.	period	
K_SLASH	/	forward slash	
K_0	0	0	48
K_1	1	1	49
K_2	2	2	50
K_3	3	3	51
K_4	4	4	52
K_5	5	5	53
K_6	6	6	54
K_7	7	7	55
K_8	8	8	56
K_9	9	9	57
K_COLON	:	colon	
K_SEMICOLON	;	semicolon	
K_LESS	<	less-than sign	
K_EQUALS	=	equals sign	
K_GREATER	>	greater-than sign	
K_QUESTION	?	question mark	
K_AT	@	at	
K_LEFTBRACKET	[	left bracket	
K_BACKSLASH	\	backslash	
K_RIGHTBRACKET	]	right bracket	
K_CARET	^	caret	
K_UNDERSCORE	_	underscore	
K_BACKQUOTE	`	grave	
K_a	a	a	65

```
47 K_b            b       b                    66
48 K_c            c       c                    67
49 K_d            d       d                    68
50 K_e            e       e                    69
51 K_f            f       f                    70
52 K_g            g       g                    71
53 K_h            h       h                    72
54 K_i            i       i                    73
55 K_j            j       j                    74
56 K_k            k       k                    75
57 K_l            l       l                    76
58 K_m            m       m                    77
59 K_n            n       n                    78
60 K_o            o       o                    79
61 K_p            p       p                    80
62 K_q            q       q                    81
63 K_r            r       r                    82
64 K_s            s       s                    83
65 K_t            t       t                    84
66 K_u            u       u                    85
67 K_v            v       v                    86
68 K_w            w       w                    87
69 K_x            x       x                    88
70 K_y            y       y                    89
71 K_z            z       z                    90
72 K_DELETE               delete
73 K_KP0                  keypad 0
74 K_KP1                  keypad 1
75 K_KP2                  keypad 2
76 K_KP3                  keypad 3
77 K_KP4                  keypad 4
78 K_KP5                  keypad 5
79 K_KP6                  keypad 6
80 K_KP7                  keypad 7
81 K_KP8                  keypad 8
82 K_KP9                  keypad 9
83 K_KP_PERIOD    .       keypad period
84 K_KP_DIVIDE    /       keypad divide
85 K_KP_MULTIPLY *        keypad multiply
86 K_KP_MINUS     -       keypad minus
87 K_KP_PLUS      +       keypad plus
88 K_KP_ENTER     \r      keypad enter
89 K_KP_EQUALS    =       keypad equals
90 K_UP                   up arrow             38
91 K_DOWN                 down arrow           40
92 K_RIGHT                right arrow          39
```

K_LEFT	left arrow	37
K_INSERT	insert	45
K_HOME	home	36
K_END	end	35
K_PAGEUP	page up	33
K_PAGEDOWN	page down	34
K_F1	F1	
K_F2	F2	
K_F3	F3	
K_F4	F4	
K_F5	F5	
K_F6	F6	
K_F7	F7	
K_F8	F8	
K_F9	F9	
K_F10	F10	
K_F11	F11	
K_F12	F12	
K_F13	F13	
K_F14	F14	
K_F15	F15	
K_NUMLOCK	numlock	
K_CAPSLOCK	capslock	
K_SCROLLOCK	scrollock	
K_RSHIFT	right shift	
K_LSHIFT	left shift	
K_RCTRL	right control	
K_LCTRL	left control	
K_RALT	right alt	
K_LALT	left alt	
K_RMETA	right meta	
K_LMETA	left meta	
K_LSUPER	left Windows key	
K_RSUPER	right Windows key	
K_MODE	mode shift	
K_HELP	help	
K_PRINT	print screen	
K_SYSREQ	sysrq	
K_BREAK	break	
K_MENU	menu	
K_POWER	power	
K_EURO	Euro	

```python
"""
Different utility functions practical for ROV control
"""

import ctypes
import os
import platform
import signal
import socket
import struct
import subprocess
import warnings


def detect_pi():
    return platform.linux_distribution()[0].lower() == 'debian'


if detect_pi():
    import serial
    import fcntl


def is_int(number):
    if isinstance(number, int):
        return True
    else:
        try:
            if isinstance(int(number), int):
                return True
        except ValueError:
            pass
    return False


def resolution_to_tuple(resolution):
    if 'x' not in resolution:
        raise ValueError('Resolution must be in format WIDTHxHEIGHT')
    screen_size = tuple([int(val) for val in resolution.split('x')])
    if len(screen_size) is not 2:
        raise ValueError('Error in parsing resolution, len is not 2')
    return screen_size

```

```python

def preexec_function():
    signal.signal(signal.SIGINT, signal.SIG_IGN)


def valid_resolution(resolution):
    if 'x' in resolution:
        w, h = resolution.split('x')
        if is_int(w) and is_int(h):
            return resolution
    warning('Resolution must be WIDTHxHEIGHT')


def server_ip(port):
    online_ips = []
    for interface in [b'eth0', b'wlan0']:
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        try:
            ip = socket.inet_ntoa(fcntl.ioctl(
                sock.fileno(),
                0x8915,
                struct.pack('256s', interface[:15])
            )[20:24])
            online_ips.append(ip)
        except OSError:
            pass
        sock.close()
    return ' or '.join(['{}:{}'.format(ip, port) for ip in online_ips])


def check_requirements():
    if detect_pi():
        camera = subprocess.check_output(['vcgencmd',
                                          'get_camera']).decode().rstrip()
        if '0' in camera:
            warning('Camera not enabled or connected properly')
            return False
        else:
            return True
    else:
        warning('eduROV only works on a raspberry pi')
        return False
```

```python


def send_arduino(msg, serial_connection):
    """
    Send the *msg* over the *serial_connection*

    Adds a hexadecimal number of 6 bytes to the start of the message before
    sending it. "hello there" -> "0x000bhello there"

    Parameters
    ----------
    msg : str or bytes
        the message you want to send
    serial_connection : object
        the :code:`serial.Serial` object you want to use for sending
    """
    if not isinstance(msg, bytes):
        msg = str(msg).encode()
    length = "{0:#0{1}x}".format(len(msg), 6).encode()
    data = length + msg
    serial_connection.write(data)


def receive_arduino(serial_connection):
    """
    Returns a message received over *serial_connection*

    Expects that the message received starts with a 6 bytes long number
    describing the size of the remaining data. "0x000bhello there" -> "hello
    there".

    Parameters
    ----------
    serial_connection : object
        the :code:`serial.Serial` object you want to use for receiving

    Returns
    -------
    msg : str or None
        the message received or None
    """
```

```
    if serial_connection.inWaiting():
        msg = serial_connection.readline().decode().rstrip()
        if len(msg) >= 6:
            try:
                length = int(msg[:6], 0)
                data = msg[6:]
                if length == len(data):
                    return data
                else:
                    warning('Received incomplete serial string: {}'
                            .format(data), 'default')
            except ValueError:
                pass
    return None


def send_arduino_simple(msg, serial_connection):
    """
    Send the *msg* over the *serial_connection*

    Same as :code:`send_arduino`, but doesn't add anything to the message
    before sending it.

    Parameters
    ----------
    msg : str or bytes
        the message you want to send
    serial_connection : object
        the :code:`serial.Serial` object you want to use for sending
    """
    if not isinstance(msg, bytes):
        msg = str(msg).encode()
    serial_connection.write(msg)


def receive_arduino_simple(serial_connection, min_length=1):
    """
    Returns a message received over *serial_connection*

    Same as :code:`receive_arduino` but doesn't expect that the message starts
    with a hex number.

```

```
    Parameters
    ----------
    serial_connection : object
        the :code:`serial.Serial` object you want to use for receiving
    min_length : int, optional
        if you only want that the function to only return the string if it is
        at least this long.

    Returns
    -------
    msg : str or None
        the message received or None
    """
    if serial_connection.inWaiting():
        msg = serial_connection.readline().decode().rstrip()
        if len(msg) >= min_length:
            return msg
        else:
            return None


def serial_connection(port='/dev/ttyACM0', baudrate=115200, timeout=0.05):
    """
    Establishes a serial connection

    Parameters
    ----------
    port : str, optional
        the serial port you want to use
    baudrate : int, optional
        the baudrate of the serial connection
    timeout : float, optional
        read timeout value

    Returns
    -------
    connection : class or None
        a :code:`serial.Serial` object if successful or None if not
    """
    try:
        ser = serial.Serial(port, baudrate, timeout=timeout)
        ser.close()
```

```
        ser.open()
        return ser
    except FileNotFoundError:
        pass
    except serial.serialutil.SerialException:
        pass
    except ValueError:
        pass
    warning(message="""Could not establish serial connection at {}\n
    Try running 'ls /dev/*tty*' to find correct port"""
            .format(port), filter='default')
    return None


def warning(message, filter='error', category=UserWarning):
    warnings.simplefilter(filter, category)
    warnings.formatwarning = warning_format
    warnings.warn(message)


def warning_format(message, category, filename, lineno,
                   file=None, line=None):
    return 'WARNING:\n  {}: {}\n  File: {}:{}\n'.format(
        category.__name__, message, filename, lineno)


def free_drive_space(as_string=False):
    """
    Checks and returns the remaining free drive space

    Parameters
    ----------
    as_string : bool, optional
        set to True if you want the function to return a formatted string.
        4278 -> 4.28 GB

    Returns
    -------
    space : float or str
        the remaining MB in float or as string if *as_string=True*
    """
    if platform.system() == 'Windows':
        free_bytes = ctypes.c_ulonglong(0)
```

```
        ctypes.windll.kernel32.GetDiskFreeSpaceExW(ctypes.c_wchar_p('/'),
                                                   None, None,
                                                   ctypes.pointer(free_bytes))
        mb = free_bytes.value / 1024 / 1024
    else:
        st = os.statvfs('/')
        mb = st.f_bavail * st.f_frsize / 1024 / 1024

    if as_string:
        if mb >= 1000:
            return '{:.2f} GB'.format(mb / 1000)
        else:
            return '{:.0f} MB'.format(mb)
    else:
        return mb


def cpu_temperature():
    """
    Checks and returns the on board CPU temperature

    Returns
    -------
    temperature : float
        the temperature
    """
    cmds = ['/opt/vc/bin/vcgencmd', 'measure_temp']
    response = subprocess.check_output(cmds).decode()
    return float(response.split('=')[1].split("'")[0].rstrip())

```

__init__.py

```
from .core import WebMethod

```

Predictive Display Code

predictive.js

```
var up = 38;
var down = 40;
var right = 39;
var left = 37;
var key_status = {up: 0, down: 0, right: 0, left:0};
var base_margin = 0;
var base_image_width = 1024;
var pixel_turn_rate = 30;
var pixel_scale_rate = 5;

var horizontal_move = 0;
var scale_move = 0;
var horizontal_px_move = 0;
var scale_px_move = 0;

var update_interval = 25;
var perceived_delay = 710;


var bodW = 0;

function sleep(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}

async function update_hor_with_delay(amount, delay){
    await sleep(delay);
    horizontal_move += amount;
}

async function update_scale_with_delay(amount, delay){
    await sleep(delay);
    scale_move += amount;
}

var x = setInterval(function() {
    if (key_status[up]){
        scale_move += 1;
        update_scale_with_delay(-1, perceived_delay);

        var factor = 0.8;
        if (key_status[left]){
            horizontal_move += factor;
            update_hor_with_delay(-factor, perceived_delay);
```

predictive.js

```
        } else if (key_status[right]){
            horizontal_move -= factor;
            update_hor_with_delay(+factor, perceived_delay);
        }
    } else if (key_status[down]){
        scale_move -= 1;
        update_scale_with_delay(1, perceived_delay);

        var factor = -0.8;
        if (key_status[left]){
            horizontal_move += factor;
            update_hor_with_delay(-factor, perceived_delay);
        } else if (key_status[right]){
            horizontal_move -= factor;
            update_hor_with_delay(+factor, perceived_delay);
        }
    } else if (key_status[left]){
        horizontal_move += 1;
        update_hor_with_delay(-1, perceived_delay);
    } else if (key_status[right]){
        horizontal_move -= 1;
        update_hor_with_delay(+1, perceived_delay);
    }

    var new_width = base_image_width + scale_move*pixel_scale_rate;
    var new_margin_left = (bodW-new_width)/2 + horizontal_move*pixel_turn_rate;

    horizontal_px_move = base_margin+horizontal_move*pixel_turn_rate;
    scale_px_move = base_image_width+scale_move*pixel_scale_rate;
    margin_left = (bodW-scale_px_move)/2+scale_move*pixel_scale_rate;
    document.getElementById("stream").style.width = `${new_width}px`;
    document.getElementById("stream").style.marginLeft = `${new_margin_left}px`;
}, update_interval);

function set_base_margin(){
    var myImage = new Image();
    var img = document.getElementById("stream");
```

predictive.js

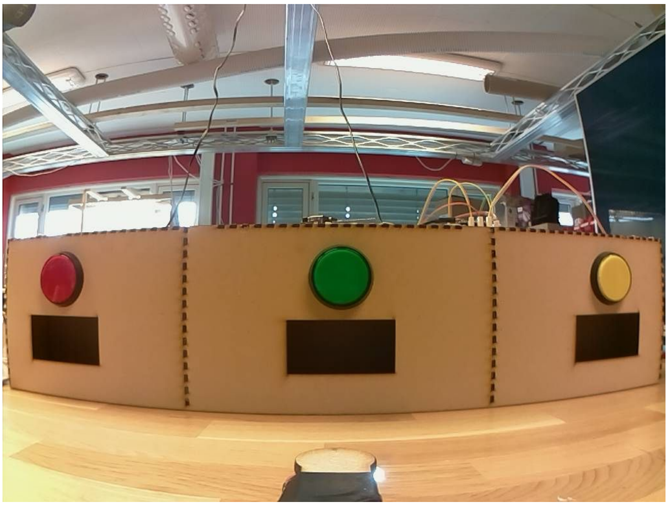
```
    myImage.src = img.src;
    var imgW = myImage.width;
    bodW = document.body.clientWidth;
    base_margin = (bodW-imgW)/2;
    document.getElementById("stream").style.marginLeft = `${base_margin}px`;
    document.getElementById("overlay").style.left = `${base_margin}px`;
}
```

Experiment Info Page

Welcome to the experiment!

This experiment aims to investigate how different ways of presenting a video with time delay effects user performance.

You will soon be presented with a camera feed from a RC car similar to the one below. You control the car with the arrow keys on the keyboard.



One by one, in random order the round LED will light up. You shall then steer the robot such that center pin goes inside the corresponding hole. The LED will turn off when you have made the hit, and a new one will light up.

Your goal

In 90 seconds, your goal is to make as many "hits" as possible.

Last notes

Three different display types will be shown to you. You will be given a 30 seconds long "training period" to get used to steering the car before hits starts to count.

Pages may load a bit slowly, you only need to press buttons one time.

OK, continue Click only once

Experiment Questionnaire

Experiment survey

Mental demand

How mentally demanding was the task?

Very low										Very high
0	1	2	3	4	5	6	7	8	9	10

Physical demand

How physically demanding was the task?

Very low										Very high
0	1	2	3	4	5	6	7	8	9	10

Temporal demand

How much time pressure did you feel because of the task?

Very low										Very high
0	1	2	3	4	5	6	7	8	9	10

Performance

How successful were you in accomplishing what you were asked to do?

Failure										Perfect
0	1	2	3	4	5	6	7	8	9	10

Effort

How hard did you have to work (mentally and physically) to accomplish your level of performance?

Very low										Very high
0	1	2	3	4	5	6	7	8	9	10

Frustration

How insecure, discouraged, irritated, stressed, and annoyed were you ?

Very low										Very high
0	1	2	3	4	5	6	7	8	9	10

Delay time

How many ms do you think the communication delay was? The time it took from you pressed a button to the reaction could be seen in the video? (1s=1000ms)

delay in ms

Submit Click only once

Data Analysis Code

Import and connect to database

In [44]:
```
import pandas as pd
from scipy import stats
import numpy as np
import sqlite3
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from math import pi

conn = sqlite3.connect("data.db")
act = pd.read_sql_query("select rowid, * from actors where valid=1;", conn)
```

Constants

In [78]:
```
def exp_format(x, pos=None):
    names = {1: 'Delay',
             2: 'Delay PD',
             3: 'No delay'}
    return names[x]

pair_dict2 = [{'aName':'Delay', 'bName':'Delay PD', 'a':0, 'b':1},
              {'aName':'Delay', 'bName':'No delay', 'a':0, 'b':2},
              {'aName':'Delay PD', 'bName':'No delay', 'a':1, 'b':2}]
```

Significance, Paired sample t-test and Cohen's D

In [41]:
```
from numpy import std, mean, sqrt

def welch_dof(x,y):
    dof = (x.var()/x.size + y.var()/y.size)**2 / ((x.var()/x.size)**2 / (x.s
ize-1) + (y.var()/y.size)**2 / (y.size-1))
    return dof

def dependent_dof(x,y):
    return (len(x)+len(y))/2-1

def cohen_d(x,y):
    x = x.tolist()
    y = y.tolist()
    nx = len(x)
    ny = len(y)
    dof = nx + ny - 2
    return (mean(x) - mean(y)) / sqrt(((nx-1)*std(x, ddof=1) ** 2 + (ny-1)*s
td(y, ddof=1) ** 2) / dof)

def print_sig(a, b, equal_var=False, dependent=True):
    if len(a) == len(b):
        t_stat, p_value = stats.ttest_rel(b, a)
        dof = dependent_dof(a, b)
    else:
        dependent = False
        t_stat, p_value = stats.ttest_ind(b, a, equal_var=equal_var)
        dof = welch_dof(a,b)
    d_value = cohen_d(b, a)

    if dependent:
        if p_value < 0.001:
            print('t({:.0f})={:.2f}, p$<$.001, d={:.3f}'.format(dof, t_stat,
 d_value))
        else:
            print('t({:.0f})={:.2f}, p={:.3f}, d={:.3f}'.format(dof, t_stat,
 p_value, d_value))
    else:
        if p_value < 0.001:
            print('t({:.2f})={:.2f}, p$<$.001, d={:.3f}'.format(dof, t_stat,
 d_value))
        else:
            print('t({:.2f})={:.2f}, p={:.3f}, d={:.3f}'.format(dof, t_stat,
 p_value, d_value))
```

Recorded data

In [14]:
```
all_act = pd.read_sql_query("select * from actors where valid=1;", conn)
all_hits = pd.read_sql_query("select * from hits where valid=1;", conn)
all_survey = pd.read_sql_query("select * from survey where valid=1;", conn)
print('A total of {} data points were collected'.format(all_act.size+all_hit
s.size+all_survey.size))
```

```
A total of 11865 data points were collected
```

Task times

In [15]:
```
times = pd.read_sql_query("select start, end from actors where valid=1;", co
nn)
length = np.array(times['end']-times['start'])
minutes = length.mean()/60-length.mean()/60%1
seconds = (length.mean()/60)%1*60
minutes_std = length.std()/60-length.std()/60%1
seconds_std = (length.std()/60)%1*60

print('Subjects used on average {:.0f} minutes and {:.0f} seconds with a sta
ndard deviation of {:.0f}min and {:.0f}s'
     .format(minutes, seconds, minutes_std, seconds_std))
```

```
Subjects used on average 10 minutes and 56 seconds with a standard deviatio
n of 1min and 12s
```

Demographics

In [12]:
```
valid_n = len(pd.read_sql_query("select age, gender, education, computer, ey
e from actors where valid=1;", conn))
non_valid = pd.read_sql_query("select age, gender, education, computer, eye
 from actors where valid=0;", conn)
female = pd.read_sql_query("select age, gender, education, computer, eye fro
m actors where gender=1 and valid=1;", conn)
male = pd.read_sql_query("select age, gender, education, computer, eye from
 actors where gender=0 and valid=1;", conn)
ages_df = pd.read_sql_query("select age from actors where valid=1;", conn)
ages = np.array(ages_df)
game = []
frequency = ['Daily', 'Weekly', 'Monthly', 'Yearly', 'Never']
print('Gaming:')
for i in range(5):
    query = "select * from actors where valid=1 and game={};".format(i)
    n_people = len(pd.read_sql_query(query, conn))
    print('{}: {}, {:.1f}'.format(frequency[i], n_people, n_people/valid_n*1
00))


print('{} total participants, {} excluded'.format(valid_n+len(non_valid),len
(non_valid) ))
print('{} males {:.1f}, {} females {:.1f}'.format(len(male), len(male)/valid
_n*100, len(female), len(female)/valid_n*100))
# print('{:.1f}% females'.format(len(female)/valid_n*100))
print('Average age of {:.1f} years with a SD of {:.2f}'.format(float(ages.me
an(axis=0)), float(ages.std(axis=0))))
print('100% said they use computer on a daily basis ')
# print('Gaming: daily {:.0f}%, weekly {:.0f}%, monthly {:.0f}%, yearly {:.0
f}% and never {:.0f}%'.format(*[i/valid_n*100 for i in game]))
```

```
Gaming:
Daily: 2, 3.5
Weekly: 15, 26.3
Monthly: 8, 14.0
Yearly: 17, 29.8
Never: 15, 26.3
58 total participants, 1 excluded
38 males 66.7, 19 females 33.3
Average age of 24.7 years with a SD of 1.45
100% said they use computer on a daily basis
```

Performance normalized

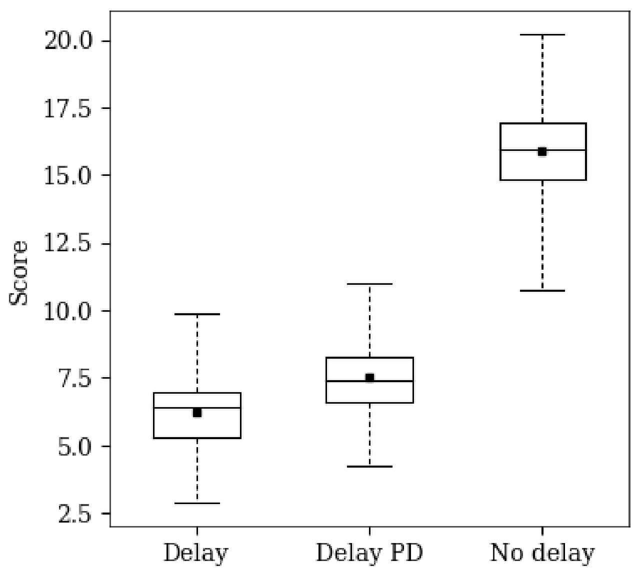
In [148]:
```
def output_statistical_information(normalized_values, group_name='All', subs
et=False):
    """
    Parameters
    ----------
    normalized_values : a Nx3 numpy matrix with normalized values

    group_name : str with the group or subgroup

    subset : if subset, it will print "n" instead of "N"
    """
    if subset:
        n = 'n'
    else:
        n = 'N'

    norm = normalized_values
    display_means = norm.mean(axis=0)
    display_std = norm.std(axis=0)
    display_max = norm.max(axis=0)
    display_min = norm.min(axis=0)
    print('{} {}={}'.format(group_name, n, len(norm)))
    print('\tScore\n')
    for exp_idx in range(3):
        print('\t{:<10} Mean: {:>5.2f}, SD: {:>.2f}'
              .format(exp_format(exp_idx+1), display_means[exp_idx], display
_std[exp_idx]))

    print('\n\tPaired difference\n')
    for di in pair_dict2:
        a = norm[...,di['a']]
        b = norm[...,di['b']]

        print('\t{:<10}- {:<10}'.format(di['aName'], di['bName']), end='')
        print('\t{:>6.2f}\%     '.format((b.mean()/a.mean()-1)*100), end='')
        print_sig(a,b)
    print('')


def boxplot(normalized_values, filename=None):
    """
    Parameters
    ----------
    normalized_values : a Nx3 numpy matrix with normalized values

    filename : str, if defined figure will be saved
    """
    norm = normalized_values
    fig, ax = plt.subplots(figsize=(4,4))
    ax.boxplot(norm, whis=2, widths=0.5)
    ax.xaxis.set_major_formatter(ticker.FuncFormatter(exp_format))
    plt.ylabel('Score')
    plt.show()
    if filename:
        fig.savefig('../img/{}.png'.format(filename), bbox_inches='tight')

def normalize_array(array):
```

```
    """
    Parameters
    ----------
    array : a Nx3 numpy matrix

    Returns
    -------
    array : a Nx3 numpy matrix with normalized values
    """
    hits = array
    total_mean = hits.mean()
    norm = np.zeros((hits.shape[0],3))
    for i, row in enumerate(hits):
        user_mean = np.array([row[0], row[1], row[2]]).mean()
        norm[i,0] = row[0]/user_mean*total_mean
        norm[i,1] = row[1]/user_mean*total_mean
        norm[i,2] = row[2]/user_mean*total_mean
    return norm
```

All

In [149]:
```
hits = np.array(pd.read_sql_query("select tothitsexp0, tothitsexp1, tothitse
xp2 from actors where valid=1;", conn))
norm = normalize_array(hits)
output_statistical_information(norm)
boxplot(norm, 'performance_norm')
```

```
All N=57
        Score

        Delay       Mean:  6.24, SD: 1.39
        Delay PD    Mean:  7.52, SD: 1.43
        No delay    Mean: 15.87, SD: 1.99

        Paired difference

        Delay     - Delay PD     20.62\%     t(56)=4.80, p$<$.001, d=0.904
        Delay     - No delay    154.37\%     t(56)=23.15, p$<$.001, d=5.569
        Delay PD  - No delay    110.88\%     t(56)=19.66, p$<$.001, d=4.772
```



Gender

In [117]:
```
genders = ['Male', 'Female']

for gender_idx, gender in enumerate(genders):
    hits = np.array(pd.read_sql_query("select tothitsexp0, tothitsexp1, toth
itsexp2 from actors where gender={} and valid=1;"
                                      .format(gender_idx), conn))
    norm = normalize_array(hits)
    output_statistical_information(norm, gender, True)
```

```
Male n=38
        Score

        Delay      Mean:  6.65, SD: 1.25
        Delay PD   Mean:  7.95, SD: 1.43
        No delay   Mean: 17.30, SD: 1.71

        Paired difference

        Delay     - Delay PD     19.62\%     t(37)=3.84, p$<$.001, d=0.960
        Delay     - No delay    160.31\%     t(37)=24.66, p$<$.001, d=7.031
        Delay PD  - No delay    117.61\%     t(37)=19.67, p$<$.001, d=5.861

Female n=19
        Score

        Delay      Mean:  5.39, SD: 1.49
        Delay PD   Mean:  6.61, SD: 1.35
        No delay   Mean: 13.10, SD: 2.17

        Paired difference

        Delay     - Delay PD     22.57\%     t(18)=2.82, p=0.011, d=0.835
        Delay     - No delay    142.85\%     t(18)=9.43, p$<$.001, d=4.033
        Delay PD  - No delay     98.14\%     t(18)=8.35, p$<$.001, d=3.495
```

Gaming

In [118]:

```
gamers = ['Daily', 'Weekly', 'Montly', 'Yearly', 'Never']

for gamer_idx, gamer in enumerate(gamers):
    hits = np.array(pd.read_sql_query("select tothitsexp0, tothitsexp1, toth
itsexp2 from actors where game={} and valid=1;"
                                      .format(gamer_idx), conn))
    norm = normalize_array(hits)
    output_statistical_information(norm, gamer, True)
```

```
Daily n=2
        Score

        Delay      Mean:  7.92, SD: 0.37
        Delay PD   Mean: 10.21, SD: 1.40
        No delay   Mean: 18.36, SD: 1.77

        Paired difference

        Delay     - Delay PD     28.88\%     t(1)=2.22, p=0.269, d=1.578
        Delay     - No delay    131.77\%     t(1)=4.87, p=0.129, d=5.762
        Delay PD  - No delay     79.83\%     t(1)=2.57, p=0.236, d=3.606

Weekly n=15
        Score

        Delay      Mean:  6.27, SD: 1.22
        Delay PD   Mean:  8.17, SD: 1.51
        No delay   Mean: 17.62, SD: 2.04

        Paired difference

        Delay     - Delay PD     30.32\%     t(14)=3.85, p=0.002, d=1.336
        Delay     - No delay    180.98\%     t(14)=14.18, p$<$.001, d=6.534
        Delay PD  - No delay    115.62\%     t(14)=10.48, p$<$.001, d=5.090

Montly n=8
        Score

        Delay      Mean:  7.05, SD: 1.32
        Delay PD   Mean:  7.77, SD: 0.64
        No delay   Mean: 17.68, SD: 0.95

        Paired difference

        Delay     - Delay PD     10.26\%     t(7)=1.04, p=0.334, d=0.652
        Delay     - No delay    150.84\%     t(7)=12.76, p$<$.001, d=8.660
        Delay PD  - No delay    127.51\%     t(7)=27.89, p$<$.001, d=11.448

Yearly n=17
        Score

        Delay      Mean:  6.65, SD: 1.26
        Delay PD   Mean:  7.66, SD: 1.73
        No delay   Mean: 15.98, SD: 2.25

        Paired difference

        Delay     - Delay PD     15.24\%     t(16)=2.00, p=0.063, d=0.650
        Delay     - No delay    140.31\%     t(16)=11.65, p$<$.001, d=4.971
        Delay PD  - No delay    108.53\%     t(16)=8.74, p$<$.001, d=4.025

Never n=15
        Score

        Delay      Mean:  5.06, SD: 1.46
        Delay PD   Mean:  6.21, SD: 1.16
```

```
No delay   Mean: 12.73, SD: 1.79

Paired difference

Delay     - Delay PD     22.54\%     t(14)=2.21, p=0.044, d=0.836
Delay     - No delay    151.31\%     t(14)=9.38, p$<$.001, d=4.529
Delay PD  - No delay    105.09\%     t(14)=9.22, p$<$.001, d=4.169
```

Gamers vs non gamers

In [121]:

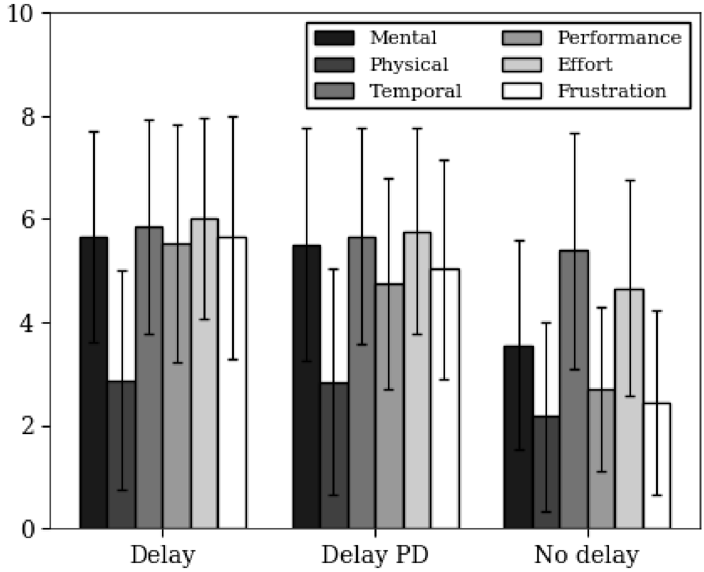
```
gamers = np.array(pd.read_sql_query("select tothitsexp0, tothitsexp1, tothit
sexp2 from actors where valid=1 and game<=1;", conn))
non_gamers = np.array(pd.read_sql_query("select tothitsexp0, tothitsexp1, to
thitsexp2 from actors where valid=1 and game>1;", conn))

output_statistical_information(normalize_array(gamers), 'Gamers', True)
output_statistical_information(normalize_array(non_gamers), 'Non gamers', Tr
ue)
```

```
Gamers n=17
        Score

        Delay      Mean:  6.46, SD: 1.19
        Delay PD   Mean:  8.40, SD: 1.53
        No delay   Mean: 17.73, SD: 2.08

        Paired difference

        Delay     - Delay PD     30.13\%     t(16)=4.34, p$<$.001, d=1.376
        Delay     - No delay    174.64\%     t(16)=14.93, p$<$.001, d=6.463
        Delay PD  - No delay    111.05\%     t(16)=10.83, p$<$.001, d=4.965

Non gamers n=40
        Score

        Delay      Mean:  6.12, SD: 1.41
        Delay PD   Mean:  7.16, SD: 1.39
        No delay   Mean: 15.09, SD: 1.93

        Paired difference

        Delay     - Delay PD     16.91\%     t(39)=3.20, p=0.003, d=0.731
        Delay     - No delay    146.46\%     t(39)=18.16, p$<$.001, d=5.237
        Delay PD  - No delay    110.80\%     t(39)=16.21, p$<$.001, d=4.655
```

Load index

Absolute

In [130]:
```
tlx_metrics = ['Mental', 'Physical', 'Temporal', 'Performance', 'Effort', 'F
rustration']
filename = 'nasa_tlx_bar'
plt.style.use('default')
plt.style.use('thesis.mplstyle')

n_partic = pd.read_sql_query("select rowid from actors where valid=1 ;", con
n).size
fig1, ax1 = plt.subplots(figsize=(5,4))
tlx_answers = []

bar_width= 0.13

for idx, metric in enumerate(tlx_metrics):
    data = np.zeros([n_partic,3])
    for exp in range(3):
        load = pd.read_sql_query("select {} from survey where valid=1 and ex
periment={};"
                                 .format(metric, exp), conn)
        data[...,exp] = np.reshape(np.array(load),(57,))
    if metric == 'Performance':
        data = np.ones_like(data)*10-data
    mean_ = data.mean(axis=0)
    std_ = data.std(axis=0)
    x_pos = np.arange(3)+1 - bar_width*3 +idx*bar_width+bar_width/2
    tlx_answers.append(data)

    ax1.bar(x_pos, mean_, bar_width, yerr=std_, label=metric,
            edgecolor='k',
            linewidth=1,
           capsize=2,
           error_kw={'linewidth':0.8})


ax1.xaxis.set_major_formatter(ticker.FuncFormatter(exp_format))
ax1.set_xticks(np.arange(3)+1)
plt.ylim(0,10)
plt.legend(ncol=2, fontsize='small')
plt.show()
# fig1.savefig('../img/{}.png'.format(filename), bbox_inches='tight')
for idx, metric in enumerate(tlx_answers):
    output_statistical_information(metric, tlx_metrics[idx])
```


10
8
6
4
2
0
Mental
Physical
Temporal
Performance
Effort
Frustration
Delay
Delay PD
No delay

```
Mental N=57
        Score

        Delay      Mean:  5.67, SD: 2.05
        Delay PD   Mean:  5.51, SD: 2.25
        No delay   Mean:  3.56, SD: 2.03

        Paired difference

        Delay     - Delay PD     -2.79\%     t(56)=-0.67, p=0.504, d=-0.073
        Delay     - No delay    -37.15\%     t(56)=-9.31, p$<$.001, d=-1.02
5
        Delay PD  - No delay    -35.35\%     t(56)=-6.36, p$<$.001, d=-0.90
2

Physical N=57
        Score

        Delay      Mean:  2.88, SD: 2.14
        Delay PD   Mean:  2.84, SD: 2.19
        No delay   Mean:  2.18, SD: 1.84

        Paired difference

        Delay     - Delay PD     -1.22\%     t(56)=-0.16, p=0.874, d=-0.016
        Delay     - No delay    -24.39\%     t(56)=-3.10, p=0.003, d=-0.349
        Delay PD  - No delay    -23.46\%     t(56)=-3.15, p=0.003, d=-0.327

Temporal N=57
        Score

        Delay      Mean:  5.84, SD: 2.08
        Delay PD   Mean:  5.67, SD: 2.10
        No delay   Mean:  5.39, SD: 2.30

        Paired difference

        Delay     - Delay PD     -3.00\%     t(56)=-0.79, p=0.431, d=-0.083
        Delay     - No delay     -7.81\%     t(56)=-1.93, p=0.059, d=-0.206
        Delay PD  - No delay     -4.95\%     t(56)=-0.97, p=0.335, d=-0.126

Performance N=57
        Score

        Delay      Mean:  5.53, SD: 2.29
        Delay PD   Mean:  4.74, SD: 2.05
        No delay   Mean:  2.70, SD: 1.60

        Paired difference

        Delay     - Delay PD    -14.29\%     t(56)=-3.24, p=0.002, d=-0.360
        Delay     - No delay    -51.11\%     t(56)=-11.76, p$<$.001, d=-1.4
15
        Delay PD  - No delay    -42.96\%     t(56)=-9.58, p$<$.001, d=-1.09
8

Effort N=57
```

```
        Score

        Delay      Mean:  6.02, SD: 1.94
        Delay PD   Mean:  5.77, SD: 1.99
        No delay   Mean:  4.67, SD: 2.08

        Paired difference

        Delay     - Delay PD     -4.08\%     t(56)=-1.05, p=0.298, d=-0.124
        Delay     - No delay    -22.45\%     t(56)=-6.34, p$<$.001, d=-0.66
5
        Delay PD  - No delay    -19.15\%     t(56)=-4.59, p$<$.001, d=-0.53
8

Frustration N=57
        Score

        Delay      Mean:  5.65, SD: 2.35
        Delay PD   Mean:  5.04, SD: 2.13
        No delay   Mean:  2.44, SD: 1.79

        Paired difference

        Delay     - Delay PD    -10.87\%     t(56)=-2.15, p=0.036, d=-0.271
        Delay     - No delay    -56.83\%     t(56)=-10.70, p$<$.001, d=-1.5
24
        Delay PD  - No delay    -51.57\%     t(56)=-8.23, p$<$.001, d=-1.31
0
```

Significance

In [316]:

```
metric = 0
g0 = tlx_answers[metric][...,1]
g1 = tlx_answers[metric][...,2]
print_sig(g1, g0)
answers_means = np.copy(tlx_answers[metric]).mean(axis=0)
print(answers_means)
print('{:.0f}% decrease in subjective latency using predictor screen'.format
((1-answers_means[1]/answers_means[0])*100))
```

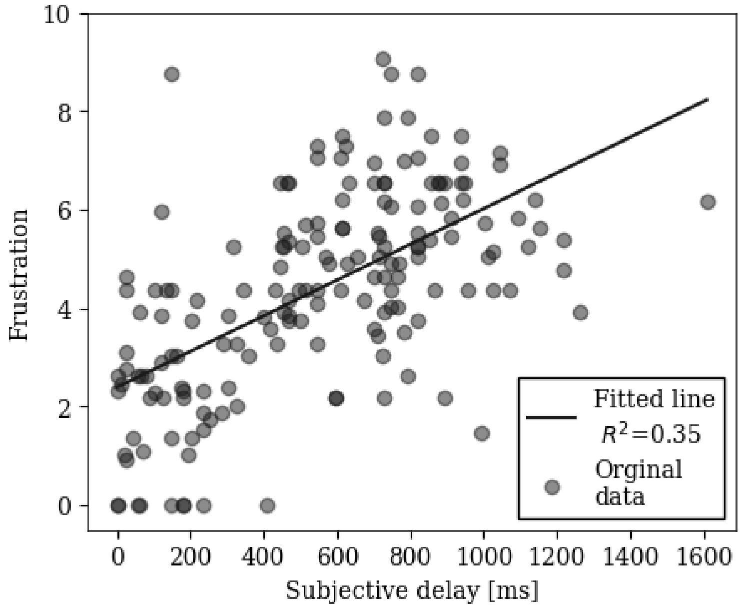
```
t(56)=6.36, p$<$.001, d=0.902
[5.66666667 5.50877193 3.56140351]
3% decrease in subjective latency using predictor screen
```

Subjective delay vs frustration

In [182]:
```
def avg_actor_delay(id):
    avg_delay = np.array(pd.read_sql_query("select delay from survey where v
alid=1 and actor={}".format(id), conn)).mean()
    return avg_delay

def avg_actor_frustration(id):
    avg_frus = np.array(pd.read_sql_query("select frustration from survey wh
ere valid=1 and actor={}".format(id), conn)).mean()
    return avg_frus

total_avg_frustration = np.array(pd.read_sql_query("select frustration from
 survey where valid=1", conn)).mean()
total_avg_delay = np.array(pd.read_sql_query("select delay from survey where
 valid=1", conn)).mean()

act = pd.read_sql_query("select rowid, * from actors where valid=1;", conn)
actor_ids = act.rowid.values
answ = [[],[]]

normalize = True
sel_exp = None
filename = 'delay_vs_frustration'

for actor_id in actor_ids:
    avg_delay = avg_actor_delay(actor_id)
    avg_frustration = avg_actor_frustration(actor_id)
    for exp in range(3):
        sur = pd.read_sql_query("select frustration, delay from survey where
 valid=1 and actor={} and experiment={}"
                                .format(actor_id, exp), conn)
        if normalize:
            frustration = float(sur['frustration'])/avg_frustration*total_av
g_frustration
            delay = float(sur['delay'])/avg_delay*total_avg_delay
        else:
            frustration = float(sur['frustration'])
            delay = float(sur['delay'])

        if sel_exp is None or exp in sel_exp:
            answ[0].append(frustration)
            answ[1].append(delay)

x = answ[1]
y = answ[0]

linreg = stats.linregress(x,y)
# print(linreg)
x_min = min(x)
x_max = max(x)
print('$R^2={:.2f}$, p={:.5f}, err={:.5f}'.format(linreg.rvalue**2, linreg.p
value, linreg.stderr))

plt.style.use('default')
plt.style.use('thesis.mplstyle')
fig, ax = plt.subplots(figsize=(5,4))
ax.scatter(x,y, marker='o', alpha=0.5, label='Orginal\ndata')
ax.plot(np.arange(x_min, x_max), np.arange(x_min, x_max)*linreg.slope+linreg
```

```
.intercept, label='Fitted line\n $R^2$={:.2f}'
        .format(linreg.rvalue**2))
ax.legend()
plt.ylabel('Frustration')
plt.xlabel('Subjective delay [ms]')
plt.ylim([-0.5,10])
plt.show()
# fig.savefig('../img/{}.png'.format(filename), bbox_inches='tight')
```

```
$R^2=0.35$, p=0.00000, err=0.00038
```

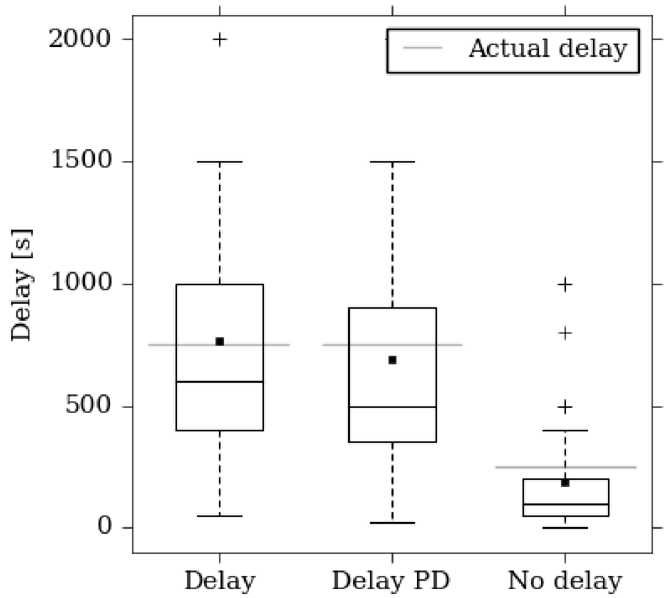


Delay times

In [100]:
```
data = pd.DataFrame()
for exp in range(3):
    data[exp] = pd.read_sql_query("select delay from survey where valid=1 an
d experiment={} order by actor asc;".format(exp), conn)
times = np.array(data)
```

Absolute

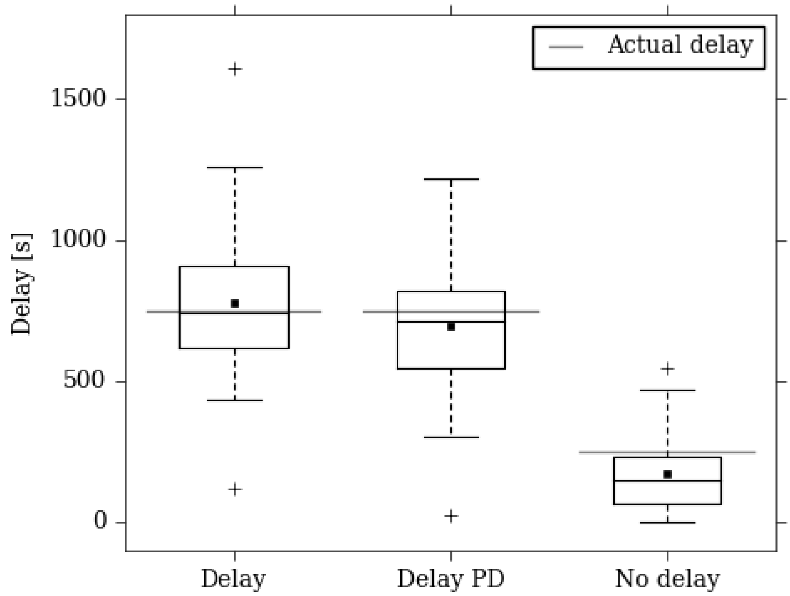
In [101]:
```
filename = 'subjective_delay_abs'
matplotlib.rcParams.update({'font.size': 11})
fig, ax = plt.subplots(figsize=(4,4))
ax.boxplot(times, widths=0.5)
ax.xaxis.set_major_formatter(ticker.FuncFormatter(exp_format))
ax.plot([0.6,1.4], [750, 750], 'k', alpha=0.3, label='Actual delay')
ax.plot([1.6,2.4], [750, 750], 'k', alpha=0.3)
ax.plot([2.6,3.4], [250, 250], 'k', alpha=0.3)
ax.legend()
plt.ylabel('Delay [s]')
plt.ylim([-100,2100])
plt.show()
# fig.savefig('../img/{}.png'.format(filename), bbox_inches='tight')
```



Normalized

In [103]:
```
sums = times.sum(axis=1)
averages = np.copy(times).mean(axis=0)
total_delay_average = np.copy(times).mean()
normalized = np.copy(times)
for idx, row in enumerate(normalized):
    user_avg = np.array([row[0], row[1], row[2]]).mean()
    row[0] = row[0]/user_avg*total_delay_average
    row[1] = row[1]/user_avg*total_delay_average
    row[2] = row[2]/user_avg*total_delay_average
```

In [104]:
```
plt.style.use('classic')
plt.style.use('thesis.mplstyle')
filename = 'subjective_delay_norm'
fig, ax = plt.subplots(figsize=(5,4))
ax.boxplot(normalized, widths=0.5)
ax.plot([0.6,1.4], [750, 750], 'k', alpha=0.5, label='Actual delay')
ax.plot([1.6,2.4], [750, 750], 'k', alpha=0.5)
ax.plot([2.6,3.4], [250, 250], 'k', alpha=0.5)
ax.legend()
ax.xaxis.set_major_formatter(ticker.FuncFormatter(exp_format))
plt.ylabel('Delay [s]')
plt.ylim([-100,1800])
plt.show()
# fig.savefig('../img/{}.png'.format(filename), bbox_inches='tight')
```



In [244]:
```
norm_avg = np.copy(normalized).mean(axis=0)
print('{:.0f}% decrease in subjective latency using predictor screen'.format
((1-norm_avg[1]/norm_avg[0])*100))
print_sig(normalized[...,1], normalized[...,0])
```

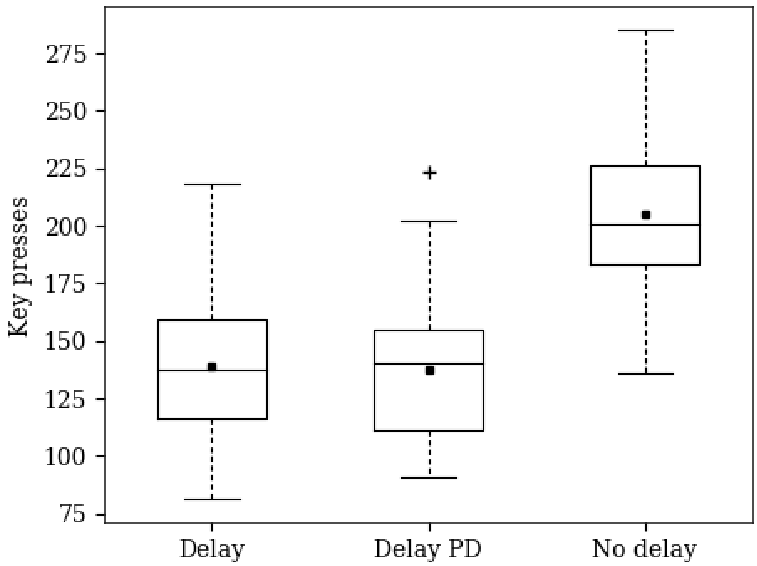
```
11% decrease in subjective latency using predictor screen
t(56)=1.40, p=0.167, d=0.356
```

Key presses

In [95]:
```
data = pd.read_sql_query("select keydowns0, keydowns1, keydowns2 from actors
 where valid=1;", conn)
keys = np.array(data)
```

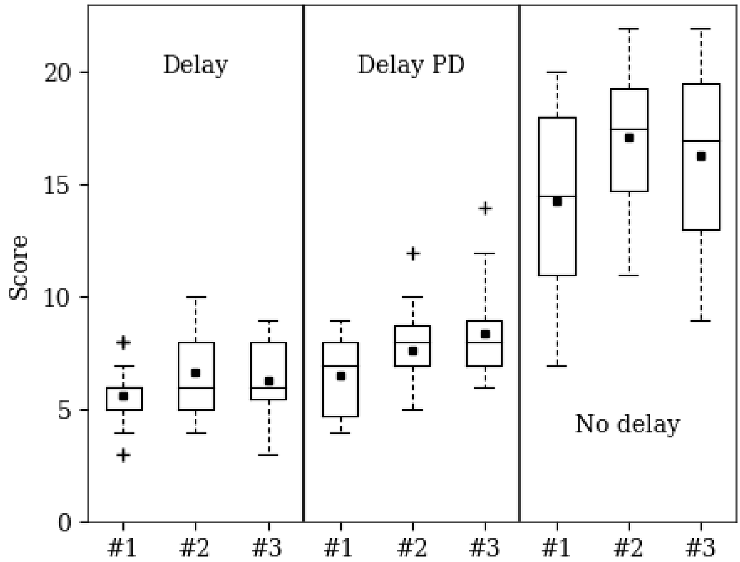
Absolute

In [96]:
```
filename = 'keypresses'
matplotlib.rcParams.update({'font.size': 10})
fig, ax = plt.subplots(figsize=(5,4))
ax.boxplot(keys, widths=0.5)
ax.xaxis.set_major_formatter(ticker.FuncFormatter(exp_format))
plt.ylabel('Key presses')
plt.show()
# fig.savefig('../img/{}.png'.format(filename), bbox_inches='tight')
```



Learning effect

In [93]:
```
def condition_format(x, pos=None):
    names = ['#1', '#2', '#3']*3
    return names[x-1]

filename = 'learning_effect'

pos = [[0,1,2,4,3,5],
      [2,3,0,5,1,4],
      [4,5,1,3,0,2]]

all_li = []

for exp in range(3):
    first = pd.read_sql_query("select tothitsexp{} from actors where valid=1
 and crowd={} or crowd={};"
                              .format(exp, pos[exp][0], pos[exp][1]), conn)
    middle = pd.read_sql_query("select tothitsexp{} from actors where valid=
1 and crowd={} or crowd={};"
                              .format(exp, pos[exp][2], pos[exp][3]), conn)
    last = pd.read_sql_query("select tothitsexp{} from actors where valid=1
 and crowd={} or crowd={};"
                              .format(exp, pos[exp][4], pos[exp][5]), conn)

    li = [first['tothitsexp'+str(exp)], middle['tothitsexp'+str(exp)],last[
'tothitsexp'+str(exp)]]
    all_li.extend(li)

fig, ax = plt.subplots(figsize=(5,4))
ax.boxplot([list(i) for i in all_li])
ax.plot([3.5, 3.5],[0,23])
ax.plot([6.5, 6.5],[0,23])
plt.ylabel('Score')
plt.text(2, 20, 'Delay', fontsize=10, ha='center')
plt.text(5, 20, 'Delay PD', fontsize=10, ha='center')
plt.text(8, 4, 'No delay', fontsize=10, ha='center')
ax.xaxis.set_major_formatter(ticker.FuncFormatter(condition_format))
plt.ylim([0,23])
plt.show()
# fig.savefig('../img/{}.png'.format(filename), bbox_inches='tight')
```



In [142]:

```
num1 = 6
num2 = num1+1
ns = min(len(all_li[num1]), len(all_li[num2]))
print(ns)

print_sig(all_li[num1][:ns], all_li[num2][:ns])
```

```
18
t(17)=3.26, p=0.005, d=0.902
```

In [181]:
```
for group in range(6):
    hits = pd.read_sql_query("select tothitsexp0, tothitsexp1, tothitsexp2 f
rom actors where valid=1 and crowd={};".format(group), conn)
    norm = normalize_array(np.array(hits))
    output_statistical_information(norm, 'Group {}'.format(group), True)
```

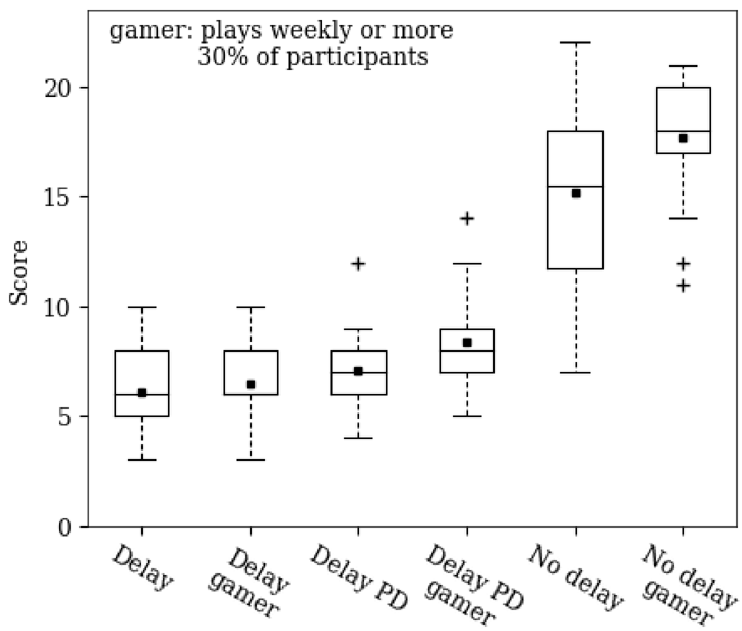
```
Group 0 n=9
        Score

        Delay      Mean:  5.08, SD: 1.21
        Delay PD   Mean:  7.40, SD: 1.13
        No delay   Mean: 15.40, SD: 1.83

        Paired difference

        Delay     - Delay PD     45.58\%     t(8)=4.49, p=0.002, d=1.867
        Delay     - No delay    203.00\%     t(8)=10.11, p$<$.001, d=6.274
        Delay PD  - No delay    108.13\%     t(8)=8.11, p$<$.001, d=4.960

Group 1 n=10
        Score

        Delay      Mean:  6.24, SD: 0.72
        Delay PD   Mean:  8.45, SD: 0.96
        No delay   Mean: 17.41, SD: 1.02

        Paired difference

        Delay     - Delay PD     35.42\%     t(9)=4.89, p$<$.001, d=2.474
        Delay     - No delay    179.12\%     t(9)=22.73, p$<$.001, d=12.050
        Delay PD  - No delay    106.11\%     t(9)=14.59, p$<$.001, d=8.602

Group 2 n=10
        Score

        Delay      Mean:  6.65, SD: 1.15
        Delay PD   Mean:  6.32, SD: 1.03
        No delay   Mean: 17.04, SD: 1.51

        Paired difference

        Delay     - Delay PD     -4.98\%     t(9)=-0.63, p=0.544, d=-0.288
        Delay     - No delay    156.29\%     t(9)=12.57, p$<$.001, d=7.347
        Delay PD  - No delay    169.73\%     t(9)=13.93, p$<$.001, d=7.884

Group 3 n=10
        Score

        Delay      Mean:  6.58, SD: 1.52
        Delay PD   Mean:  6.57, SD: 0.57
        No delay   Mean: 16.76, SD: 1.59

        Paired difference

        Delay     - Delay PD     -0.13\%     t(9)=-0.01, p=0.988, d=-0.007
        Delay     - No delay    154.87\%     t(9)=9.99, p$<$.001, d=6.211
        Delay PD  - No delay    155.19\%     t(9)=16.53, p$<$.001, d=8.081

Group 4 n=9
        Score

        Delay      Mean:  6.78, SD: 0.91
        Delay PD   Mean:  8.42, SD: 1.33
```

```
        No delay   Mean: 14.69, SD: 1.45

        Paired difference

        Delay     - Delay PD     24.34\%     t(8)=2.66, p=0.029, d=1.366
        Delay     - No delay    116.81\%     t(8)=11.05, p$<$.001, d=6.163
        Delay PD  - No delay     74.38\%     t(8)=6.74, p$<$.001, d=4.248

Group 5 n=9
        Score

        Delay      Mean:  6.03, SD: 1.84
        Delay PD   Mean:  8.04, SD: 1.45
        No delay   Mean: 13.59, SD: 2.58

        Paired difference

        Delay     - Delay PD     33.42\%     t(8)=2.74, p=0.026, d=1.147
        Delay     - No delay    125.50\%     t(8)=5.06, p$<$.001, d=3.190
        Delay PD  - No delay     69.02\%     t(8)=4.18, p=0.003, d=2.503
```

Gamers

In [122]:

```
def game_format(x, pos=None):
    names = {1: 'Delay',
             2: 'Delay\ngamer',
            3: 'Delay PD',
            4: 'Delay PD\ngamer',
            5: 'No delay',
            6: 'No delay\ngamer'}
    return names[x]

gamers = pd.read_sql_query("select rowid, * from actors where valid=1 and ga
me<=1;", conn)
ga = np.array(gamers[['tothitsexp0', 'tothitsexp1', 'tothitsexp2']])
non_gamers = pd.read_sql_query("select rowid, * from actors where valid=1 an
d game >1;", conn)
no = np.array(non_gamers[['tothitsexp0', 'tothitsexp1', 'tothitsexp2']])

game_per = len(gamers)/(len(gamers)+len(non_gamers))

filename = 'gamer_performance'
fig1, ax1 = plt.subplots(figsize=(5,4))
# ax1.set_title('Performance gamers vs non gamers')
ax1.boxplot([no[...,0], ga[...,0], no[...,1], ga[...,1], no[...,2], ga[...,2
]])
ax1.xaxis.set_major_formatter(ticker.FuncFormatter(game_format))
plt.ylabel('Score')
plt.text(0.7, 21, 'gamer: plays weekly or more\n            {:.0f}% of parti
cipants'.format(game_per*100), fontsize=10)
plt.xticks(rotation=-30)
plt.ylim([0,23.5])
plt.show()
# fig1.savefig('../img/{}.png'.format(filename), bbox_inches='tight')
```



In [157]:
```
exp = 2
print_sig(no[...,0], no[...,1])
print_sig(ga[...,0], ga[...,1])
```

```
t(39)=3.27, p=0.002, d=0.577
t(16)=4.17, p<.001, d=1.018
```

Collected Experiment Data

Participants

rowid	age	gender	education	game	computer	eye	start	end	crowd	startexp0	startexp1	startexp2	endexp0	endexp1	endexp2	tothitsexp0	tothitsexp1	tothitsexp2	keydowns0	keydowns1	keydowns2	valid
1	24	1	6	3	0	0	1525722398	1525723159	0	1525722495	1525721324	1525721397	1525722585	1525722905	1525723111	7	6	7	97	85	117	0
2	30	0	6	1	0	0	1525723610	1525724257	1	1525723712	1525724110	1525723916	1525723802	1525724204	1525724006	6	9	21	81	99	226	1
3	24	0	4	2	0	3	1525725920	1525726646	2	1525726260	1525726019	1525726495	1525726350	1525726109	1525726588	10	9	20	169	146	218	1
4	25	0	4	2	0	3	1525726901	1525727548	3	1525727395	1525727003	1525727204	1525727485	1525727093	1525727294	6	9	22	170	151	241	1
5	24	0	4	0	0	3	1525724792	1525725370	4	1525725082	1525725242	1525724894	1525725174	1525725332	1525724984	10	14	20	146	184	208	1
6	25	0	4	3	0	2	1525725701	1525726559	5	1525726396	1525726113	1525725832	1525726486	1525726203	1525725922	8	12	18	116	126	177	1
7	25	0	4	2	0	2	1525727058	1525727792	0	1525727203	1525727434	1525727637	1525727293	1525727524	1525727727	6	9	20	97	132	168	1
8	26	0	4	1	0	0	1525729628	1525730349	1	1525729781	1525730196	1525729992	1525729871	1525730286	1525730083	6	9	21	138	144	229	1
9	27	0	5	3	0	0	1525733014	1525733742	2	1525733368	1525733137	1525733573	1525733458	1525733228	1525733664	8	9	19	115	102	214	1
10	23	1	4	2	0	0	1525734238	1525734955	3	1525734789	1525734325	1525734566	1525734879	1525734415	1525734658	6	5	11	87	98	189	1
11	24	1	4	2	0	0	1525735285	1525735932	4	1525735619	1525735796	1525735386	1525735709	1525735886	1525735476	8	8	18	125	120	225	1
12	24	1	4	4	0	3	1525736428	1525737139	5	1525736991	1525736795	1525736536	1525737081	1525736889	1525736626	6	5	7	96	91	136	1
13	26	0	4	3	0	0	1525737764	1525738407	0	1525737846	1525738091	1525738267	1525737936	1525738182	1525738357	4	8	14	163	141	192	1
14	25	0	4	2	0	0	1525738677	1525739295	1	1525738776	1525739154	1525738979	1525738866	1525739246	1525739071	5	7	14	196	148	230	1
15	26	0	6	1	0	1	1525739590	1525740160	2	1525739864	1525739675	1525740041	1525739955	1525739770	1525740131	8	7	20	130	126	221	1
16	24	0	4	2	0	0	1525741547	1525742194	3	1525742058	1525741654	1525741866	1525742148	1525741744	1525741959	9	7	19	162	142	211	1
17	24	0	4	2	0	0	1525742405	1525743029	4	1525742700	1525742874	1525742513	1525742790	1525742964	1525742605	6	8	18	174	147	235	1
19	25	0	5	1	0	0	1525743551	1525744116	0	1525743606	1525743815	1525743986	1525743696	1525743909	1525744076	6	8	12	97	94	196	1
20	25	0	4	1	0	0	1525745109	1525745787	1	1525745212	1525745648	1525745460	1525745302	1525745738	1525745550	6	12	19	102	170	217	1
21	23	0	4	3	0	0	1525746609	1525747249	2	1525746934	1525746703	1525747117	1525747024	1525746795	1525747208	8	8	17	153	176	213	1
22	23	0	4	0	0	1	1525747730	1525748301	3	1525748168	1525747815	1525748003	1525748258	1525747905	1525748093	6	7	16	119	97	186	1
23	24	1	5	4	0	1	1525748751	1525749413	4	1525749058	1525749284	1525748844	1525749148	1525749376	1525748934	5	7	13	130	112	189	1
24	24	0	5	1	0	0	1525749932	1525750587	5	1525750438	1525750249	1525750041	1525750528	1525750339	1525750131	8	9	17	178	153	285	1
25	25	0	5	1	0	2	1525750987	1525751621	0	1525751097	1525751315	1525751489	1525751187	1525751405	1525751579	5	5	17	119	94	198	1
26	26	0	5	1	0	2	1525751987	1525752617	1	1525752077	1525752475	1525752282	1525752167	1525752565	1525752372	6	8	17	102	144	205	1
32	24	0	4	3	0	3	1525756008	1525756606	1	1525756094	1525756472	1525756292	1525756184	1525756562	1525756383	8	9	22	202	185	269	1
33	26	0	4	3	0	3	1525756885	1525757469	2	1525757165	1525756974	1525757337	1525757256	1525757064	1525757427	5	4	19	142	202	173	1
34	27	0	4	1	0	0	1525758112	1525758794	3	1525758657	1525758228	1525758462	1525758747	1525758318	1525758555	8	8	20	147	156	234	1
35	30	0	6	1	0	0	1525759276	1525759843	4	1525759535	1525759711	1525759353	1525759627	1525759802	1525759444	8	9	19	130	158	254	1
36	23	0	3	1	0	3	1525760850	1525761436	5	1525761298	1525761113	1525760923	1525761388	1525761203	1525761013	6	10	11	134	137	222	1
37	25	0	6	1	0	0	1525761985	1525762705	0	1525762092	1525762388	1525762570	1525762182	1525762478	1525762660	3	7	21	130	165	204	1
38	25	0	6	1	0	2	1525762953	1525763607	1	1525763069	1525763464	1525763282	1525763159	1525763554	1525763372	5	7	14	149	117	169	1
39	25	0	4	1	0	0	1525764032	1525764634	2	1525764330	1525764118	1525764504	1525764420	1525764208	1525764594	5	7	18	110	97	200	1
40	25	0	6	3	0	0	1525765094	1525765745	3	1525765589	1525765192	1525765398	1525765679	1525765283	1525765488	8	8	18	149	147	231	1
41	25	0	5	4	0	0	1525765936	1525766606	4	1525766270	1525766447	1525766039	1525766360	1525766537	1525766129	6	8	12	137	130	136	1
42	24	1	4	3	0	2	1525767883	1525768574	5	1525768444	1525768236	1525768000	1525768534	1525768326	1525768090	6	8	8	147	154	216	1
44	23	0	4	3	0	0	1525769061	1525769644	1	1525769154	1525769511	1525769347	1525769246	1525769601	1525769437	8	9	18	125	153	211	1
45	24	1	4	4	0	1	1525770141	1525770779	2	1525770441	1525770238	1525770656	1525770531	1525770328	1525770746	5	4	13	142	110	183	1
46	23	0	5	4	0	0	1525771448	1525772335	3	1525772166	1525771563	1525771961	1525772257	1525771653	1525772051	8	6	16	139	145	261	1
47	24	1	4	3	0	0	1525773336	1525773949	4	1525773626	1525773806	1525773420	1525773716	1525773896	1525773510	6	8	9	127	140	177	1
48	25	1	4	4	0	0	1525774364	1525774983	5	1525774853	1525774678	1525774452	1525774944	1525774768	1525774545	5	7	11	91	100	143	1
49	25	1	4	4	0	2	1525775720	1525776585	0	1525775827	1525776122	1525776391	1525775917	1525776212	1525776482	4	7	11	188	185	183	1
50	24	0	5	3	0	3	1525777084	1525777673	1	1525777142	1525777532	1525777351	1525777235	1525777622	1525777442	7	7	18	160	188	252	1
51	25	0	4	4	0	2	1525780656	1525781271	2	1525780955	1525780757	1525781139	1525781045	1525780847	1525781229	4	4	9	212	183	232	1
52	24	1	4	3	0	2	1525781518	1525782226	3	1525782134	1525781612	1525781844	1525782227	1525781702	1525781935	4	6	16	120	111	195	1
54	23	1	4	4	0	3	1525784018	1525784657	5	1525784535	1525784355	1525784121	1525784625	1525784447	1525784212	3	9	19	218	179	278	1
55	23	1	4	4	0	0	1525785166	1525785933	0	1525785296	1525785584	1525785837	1525785386	1525785677	1525785928	6	6	11	96	96	147	1
57	24	0	4	3	0	0	1525788760	1525789481	2	1525789096	1525788856	1525789335	1525789186	1525788946	1525789425	7	4	13	163	223	149	1
58	25	1	6	4	0	0	1525791218	1525791833	4	1525791525	1525791694	1525791317	1525791618	1525791786	1525791407	4	7	10	153	116	197	1
59	23	1	4	4	0	3	1525792390	1525793100	5	1525792968	1525792785	1525792517	1525793058	1525792875	1525792608	3	5	15	175	167	201	1
60	24	0	5	4	0	0	1525793583	1525794163	0	1525793674	1525793868	1525794034	1525793764	1525793961	1525794124	5	8	16	122	125	162	1

61	24	1	4	4	0	2	1525794624	1525795317	3	1525795150	1525794747	1525794960	1525795240	1525794837	1525795051	3	4	15	116	96	194	1
62	26	0	6	3	0	0	1525796381	1525796948	4	1525796655	1525796824	1525796468	1525796745	1525796914	1525796561	8	6	14	115	117	184	1
63	25	1	4	1	0	0	1525797305	1525798018	5	1525797873	1525797672	1525797456	1525797963	1525797763	1525797546	8	7	18	126	121	198	1
64	26	0	4	3	0	3	1525798819	1525799413	0	1525798921	1525799112	1525799283	1525799011	1525799204	1525799373	6	8	18	145	109	181	1
65	25	1	4	3	0	0	1525799958	1525800543	1	1525800051	1525800413	1525800237	1525800141	1525800503	1525800329	5	7	11	146	109	170	1
66	23	1	4	3	0	0	1525801053	1525801662	2	1525801348	1525801154	1525801525	1525801438	1525801245	1525801615	6	8	22	159	155	276	1
67	23	1	4	4	0	0	1525802165	1525802787	3	1525802638	1525802260	1525802468	1525802728	1525802350	1525802558	8	6	14	110	129	188	1

Test questionnaire

actor	experiment	mental	physical	temporal	effort	performance	frustration	delay	time	valid
1	0	3	3	5	5	3	6	1000	1525722715	0
1	1	5	3	5	3	3	7	1000	1525722969	0
1	2	3	3	4	3	4	5	600	1525723155	0
2	0	4	1	5	5	6	4	600	1525723867	1
2	2	3	2	5	2	8	2	0	1525724068	1
2	1	4	1	4	4	6	4	600	1525724254	1
3	1	6	2	8	10	7	4	800	1525726212	1
3	0	6	3	8	9	7	5	400	1525726445	1
3	2	3	1	9	4	9	0	0	1525726642	1
4	1	6	1	6	4	7	8	1000	1525727158	1
4	2	6	2	7	7	8	2	100	1525727343	1
4	0	8	5	5	8	2	9	2500	1525727544	1
5	2	1	1	5	3	9	0	1000	1525725036	1
5	0	5	2	5	6	7	3	1500	1525725199	1
5	1	6	4	5	6	6	4	1500	1525725366	1
6	2	5	3	7	7	7	2	200	1525726063	1
6	1	7	4	6	6	3	4	350	1525726349	1
6	0	9	6	6	8	0	8	600	1525726555	1
7	0	8	3	6	6	8	1	800	1525727380	1
7	1	6	2	6	6	8	4	900	1525727590	1
7	2	3	1	2	2	9	1	100	1525727788	1
8	0	3	2	5	5	5	7	300	1525729939	1
8	2	4	2	6	5	10	1	10	1525730138	1
8	1	3	2	3	5	6	5	500	1525730347	1
9	1	7	4	6	8	7	5	500	1525733320	1
9	0	7	6	7	7	5	6	600	1525733524	1
9	2	5	4	7	6	9	1	50	1525733739	1
10	1	6	0	3	4	3	5	300	1525734517	1
10	2	3	0	4	3	5	2	100	1525734743	1
10	0	5	0	4	4	4	5	500	1525734952	1
11	2	6	2	6	7	5	4	300	1525735578	1
11	0	8	2	6	8	3	6	600	1525735756	1
11	1	7	2	6	7	4	6	800	1525735929	1
12	2	4	1	5	5	4	2	500	1525736712	1
12	1	3	3	5	5	4	4	2000	1525736945	1
12	0	6	5	5	4	5	5	200	1525737136	1
13	0	6	0	9	6	3	9	400	1525738047	1
13	1	6	1	7	5	6	5	200	1525738229	1
13	2	6	2	6	6	8	3	100	1525738404	1
14	0	4	1	3	3	6	3	300	1525738935	1
14	2	1	1	1	3	8	1	80	1525739108	1
14	1	9	7	6	9	3	9	300	1525739292	1
15	1	7	7	7	7	8	6	250	1525739824	1
15	0	7	5	6	6	5	4	600	1525740002	1
15	2	7	6	4	7	8	2	70	1525740158	1
16	1	7	4	6	6	9	2	700	1525741824	1
16	2	2	2	5	5	8	2	50	1525742017	1
16	0	7	6	6	7	5	6	700	1525742191	1
17	2	3	1	8	7	8	4	20	1525742659	1
17	0	6	3	9	8	3	9	700	1525742834	1
17	1	7	2	7	6	6	6	500	1525743026	1
19	0	8	4	6	7	5	3	200	1525743770	1
19	1	8	6	7	7	6	4	50	1525743942	1
19	2	7	5	8	7	8	3	10	1525744113	1
20	0	4	5	7	8	2	9	600	1525745416	1
20	2	2	2	8	3	8	2	200	1525745603	1
20	1	4	5	8	5	4	6	600	1525745783	1
21	1	5	2	3	5	6	5	250	1525746888	1
21	0	4	2	4	4	6	5	200	1525747076	1
21	2	2	2	2	2	7	3	50	1525747246	1
22	1	3	4	4	4	6	3	200	1525747960	1
22	2	2	1	2	3	8	0	50	1525748128	1
22	0	7	3	2	6	5	3	200	1525748298	1
23	2	3	1	7	4	5	6	400	1525749013	1
23	0	6	5	8	8	2	8	1300	1525749238	1
23	1	4	3	6	5	3	5	1300	1525749411	1

24	2	3	3	7	4	7	3	10	1525750206	1
24	1	5	2	7	6	3	6	500	1525750397	1
24	0	5	3	6	6	3	7	800	1525750585	1
25	0	5	0	9	8	7	4	1000	1525751261	1
25	1	8	4	8	8	7	6	1000	1525751438	1
25	2	3	1	5	7	7	3	200	1525751619	1
26	0	3	0	7	8	4	7	1000	1525752236	1
26	2	1	0	6	3	9	0	100	1525752435	1
26	1	8	1	7	9	2	7	1500	1525752615	1
32	0	5	4	7	6	5	6	200	1525756243	1
32	2	5	5	4	6	7	2	100	1525756419	1
32	1	6	5	3	6	4	7	350	1525756604	1
33	1	7	0	8	5	2	10	350	1525757126	1
33	0	6	1	7	5	5	7	150	1525757296	1
33	2	2	1	5	4	8	2	50	1525757465	1
34	1	3	0	4	4	5	4	400	1525758411	1
34	2	1	0	5	4	6	2	60	1525758616	1
34	0	5	1	5	6	2	4	1000	1525758791	1
35	2	5	3	8	7	8	4	30	1525759492	1
35	0	8	4	7	6	6	7	1000	1525759659	1
35	1	9	5	8	8	6	6	900	1525759840	1
36	2	3	2	7	7	7	3	4	1525761071	1
36	1	8	3	8	8	5	6	25	1525761249	1
36	0	8	3	8	8	3	8	1500	1525761434	1
37	0	5	0	3	6	3	4	800	1525762345	1
37	1	6	0	7	5	5	2	400	1525762531	1
37	2	2	0	4	4	6	1	200	1525762703	1
38	0	6	2	0	7	4	7	600	1525763239	1
38	2	3	1	0	5	6	1	30	1525763423	1
38	1	7	4	2	7	4	6	1500	1525763604	1
39	1	6	1	8	6	7	3	500	1525764278	1
39	0	6	2	7	4	6	4	750	1525764461	1
39	2	4	2	7	4	8	2	100	1525764631	1
40	1	6	2	7	9	6	1	800	1525765358	1
40	2	9	2	8	9	8	3	20	1525765551	1
40	0	10	2	7	9	5	5	500	1525765743	1
41	2	5	5	7	7	5	4	1000	1525766230	1
41	0	8	6	8	7	2	7	1000	1525766407	1
41	1	6	5	6	7	4	5	1000	1525766603	1
42	2	3	6	8	3	10	7	1000	1525768189	1
42	1	2	4	7	3	10	6	1500	1525768400	1
42	0	2	4	8	4	9	9	1000	1525768571	1
44	0	6	3	7	7	6	8	400	1525769306	1
44	2	4	6	6	4	8	3	100	1525769472	1
44	1	8	6	7	7	5	6	400	1525769641	1
45	1	0	0	7	2	10	3	100	1525770400	1
45	0	2	0	3	2	10	3	200	1525770617	1
45	2	0	0	1	1	10	0	50	1525770777	1
46	1	5	0	6	3	7	6	1500	1525771815	1
46	2	1	0	2	0	9	0	0	1525772123	1
46	0	7	2	6	3	5	7	1500	1525772332	1
47	2	5	4	9	6	5	6	500	1525773587	1
47	0	8	7	6	10	3	8	500	1525773767	1
47	1	9	9	7	6	5	6	800	1525773947	1
48	2	3	0	6	5	7	0	100	1525774632	1
48	1	3	0	5	5	4	4	1500	1525774814	1
48	0	2	0	6	3	4	4	1200	1525774980	1
49	0	5	0	9	5	10	5	100	1525776074	1
49	1	4	0	7	2	10	5	80	1525776348	1
49	2	2	0	4	3	10	3	50	1525776582	1
50	0	4	1	6	6	4	1	400	1525777304	1
50	2	6	5	5	6	5	2	100	1525777488	1
50	1	3	4	2	3	2	3	600	1525777671	1
51	1	0	0	0	5	7	1	500	1525780911	1
51	0	1	0	3	1	7	1	1000	1525781098	1
51	2	1	0	2	1	9	1	100	1525781268	1
52	1	4	1	5	6	5	3	400	1525781803	1
52	2	5	2	5	6	7	2	170	1525782020	1
52	0	6	1	5	5	1	6	1000	1525782319	1

54	2	6	5	7	8	7	4	10	1525784306	1
54	1	8	7	9	9	3	9	100	1525784485	1
54	0	8	7	9	8	2	10	50	1525784654	1
55	0	8	4	8	9	10	8	1000	1525785532	1
55	1	10	4	8	9	8	9	1500	1525785712	1
55	2	8	4	8	8	10	7	800	1525786001	1
57	1	5	1	8	6	2	7	50	1525789052	1
57	0	6	1	5	5	5	4	70	1525789297	1
57	2	4	1	3	3	7	2	30	1525789477	1
58	2	2	1	5	4	6	4	300	1525791480	1
58	0	6	5	7	6	2	8	800	1525791653	1
58	1	5	4	5	5	6	4	400	1525791830	1
59	2	6	1	8	7	6	4	500	1525792739	1
59	1	7	1	4	7	3	1	2000	1525792926	1
59	0	7	1	6	7	3	1	3000	1525793097	1
60	0	3	3	0	3	2	6	500	1525793823	1
60	1	2	2	0	1	5	3	500	1525793995	1
60	2	0	0	0	0	8	0	100	1525794160	1
61	1	5	3	8	6	3	5	500	1525794914	1
61	2	6	4	7	5	5	5	200	1525795105	1
61	0	7	5	7	8	0	7	2000	1525795314	1
62	2	1	1	3	5	7	0	200	1525796611	1
62	0	3	2	3	6	6	4	800	1525796780	1
62	1	6	2	4	6	5	6	800	1525796945	1
63	2	3	5	8	5	3	6	500	1525797625	1
63	1	7	6	4	7	3	9	1500	1525797829	1
63	0	7	7	4	5	3	6	2000	1525798015	1
64	0	3	3	5	4	4	8	500	1525799063	1
64	1	2	2	3	3	6	4	500	1525799240	1
64	2	3	3	5	3	8	3	50	1525799409	1
65	0	1	0	6	3	5	1	100	1525800195	1
65	2	1	0	4	2	6	1	80	1525800369	1
65	1	1	0	4	2	6	1	200	1525800539	1
66	1	6	3	3	6	3	4	400	1525801306	1
66	0	6	5	3	6	2	4	700	1525801483	1
66	2	4	5	6	6	6	4	100	1525801659	1
67	1	6	5	8	8	4	9	600	1525802422	1
67	2	5	4	8	6	6	2	200	1525802596	1
67	0	7	6	8	8	3	8	800	1525802784	1