UPPSALA
UNIVERSITET

UPTEC F12 010
Examensarbete 30 hp
Februari 2012

Stabilization of handheld firearms using image analysis

Alexander Lindstedt

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Stabilisering av handeldvapen med bildanalys

Stabilization of handheld firearms using image analysis

Alexander Lindstedt

When firing a handheld weapon, the shooter tries to aim at the point where he wants the bullet to hit. However, due to imperfections in the human body, this can be quite hard. The weapon moves relative to the target and the shooter has to use precise timing to fire the shot exactly when the weapon points to the intended target position. This can be very hard, especially when shooting at long range using a magnifying rifle scope.
In this thesis, a solution to this problem using image analysis is described and tested. Using a digital video camera and software, the system helps the shooter to fire at the appropriate time. The system is designed to operate in real-time conditions on a PC. The tests carried out have shown that the solution is promising and helps to achieve better accuracy. However it needs to be optimized to run smoothly on a smaller scale embedded system.

Handledare: Göran Backlund
Ämnesgranskare: Cris Luengo
Examinator: Tomas Nyberg
ISSN: 1401-5757, UPTEC F12 010
Sponsor: Combitech AB

1 Populärvetenskaplig sammanfattning

Då en skytt avfyrar ett handhållet vapen försöker skytten sikta mot den punkt där han vill att kulan ska träffa. Eftersom den mänskliga kroppen inte är helt stabil kommer vapnet att röra sig runt denna punkt och skytten måste försöka avfyra skottet precis vid den tidpunkt då vapnet pekar mot rätt punkt. Detta är särskilt svårt vid stora avstånd, då små vinkelskillnader i vapnets pipa ger större utslag med ökande avstånd till målet. I denna uppsats beskrivs och utvärderas ett system konstruerat för att minimera inverkan av de ofrivilliga rörelserna. Systemet använder sig av en videokamera monterad i siktet och en dator med mjukvara som utför analys och behandling av videoströmmen för att avgöra när vapnet bör avfyras. Tanken är att i ett färdigt system implementera algoritmen i ett portabelt inbyggt system som kan monteras i kikarsiktet tillsammans med kameran. Mjukvaran kan sedan styra avfyrningen elektroniskt efter att skytten gett sitt godkännande genom att lägga tryck på avtryckaren. Testen som genomförts visar att angreppssättet är lovande. Systemet fick i samtliga fall bättre resultat än då skyttarna avfyrade skott manuellt.

Contents

1 Populärvetenskaplig sammanfattning **3**

2 Introduction **5**
2.1 Background 5
2.2 Specifications 5
2.3 Delimitations 6

3 Pre-study and system overview **7**
3.1 Finding the target or tracking the motion 8
3.1.1 Pattern recognition 8
3.1.2 Image stabilization 8
3.2 Initial system layout 8
3.2.1 Hardware 9

4 System description **10**
4.1 Hardware platform 10
4.2 Software platform 11
4.3 Physical model 12
4.3.1 Pinhole camera model 12
4.3.2 2-D motion model 13
4.4 Motion estimation 13
4.5 Motion pattern analysis 16
4.6 Point of aim estimation 18
4.7 Motion prediction 19
4.8 Firing decision 19
4.8.1 Criteria 19
4.9 Error sources 20
4.10 User interface 21

5 System evaluation **22**
5.1 Test method 22
5.2 Results 23

6 Discussion and conclutions **24**
6.1 System performance 24
6.2 Physical limitations 24
6.3 Real-Time properties 25
6.4 Robustness 25
6.5 Safety 25
6.6 Recommendations for future work 25

2 Introduction

2.1 Background

Since the mid 1800's, telescopic sights have been used to extend range and improve accuracy when firing handheld weapons. The use of these sights is widespread in military, law enforcement and hunting applications. These sights help the shooter by magnifying the field of view, typically by a factor 3 to 20.

Shooting at a long range introduces problems that need to be solved in order to achieve good accuracy. Since the bullet travels a long distance before reaching the target, many outside forces such as wind and gravity have to be taken into account. Also, the high level of magnification and the long distance to the target make even very small angular movements of the weapon have a large effect on the point of aim.

Since their introduction, several features have been added to telescopic sights to help the shooter achieve better results. Bullet drop compensation helps the shooter by compensating for the force of gravity, which acts on the bullet when traveling the distance to the target.

With the development of large scale integrated circuits, the possibility to integrate electronics into scopes without adding too much bulk has become reality. Several companies now offer features such as integrated rangefinders for estimation of the distance to the target. There are also complete ballistic systems, which calculate the ballistics of the bullet, taking into consideration ammunition type, air density and other factors relevant to the bullet path.

The rapid development of processing speed allows for more complex calculations to be carried out in integrated systems. This makes new features feasible for implementation.

One problem that still needs to be solved is how to compensate for the unintentional movements the shooter makes when trying to aim at a target. These movements become especially noticeable when the target is at long range. One of the most important factors that plays a role in where the bullet hits is how well the shooter is able to time his shot. Perfect timing is very hard, since the finger movement to pull the trigger cannot be very fast without introducing unwanted movement in the rifle. The system described in this thesis tries to compensate for this human imperfection by using image analysis and signal processing to help the shooter fire at the time when the weapon points at the intended target. The demonstrator system is based on a pending patent application by Göran Backlund and Gert Johansson at Combitech AB.

2.2 Specifications

The system developed is a technology demonstrator built to test the possibilities of using this approach to increase accuracy. The system consists of a digital camera mounted on a rifle scope, and connected to a computer. The computer does all the calculations and simulates shooting, so that the precision of the system can be evaluated. The system should be as fast and computationally effective as possible taking into consideration the possible future implementation in a smaller embedded system.

2.3 Delimitations

The proposed system is a demonstrator for a possible future product. It is not finished for implementation in an embedded enviroment, nor is it adequatly tested to be safe or reliable enough for use in a commercial product. It is a proof of concept and a test to see whether the proposed method is good enough to continue development toward a commercial product. The system is limited to handling stationary (non-moving) targets.

3 Pre-study and system overview

The proposed solution uses a digital video camera as its only sensor and the output of the system is an electrical signal telling the weapon to fire. The system is initiated by the shooter pressing the trigger and can be aborted at any time by releasing pressure from the trigger.

To be able to determine when to fire the weapon, the information from the camera needs to be analyzed. The information obtained from the camera is the same as the field of view of the shooter trying to aim at the target as seen in figure 2.

Figure 1: System functionality

Figure 2: The field of view as seen by the digital camera

For the purpose of finding the optimal time at which to fire the weapon, several parameters need to be estimated. The target position in the image needs to be known, and we need to know where the shooter is currently aiming. In addition to this, due to the sampling interval not being infinitesimally small and due to delays in the system, we also need an estimate of the velocity with which the current aiming point is moving. This is necessary in order to predict the aiming position until the next sample is received from the sensor.

3.1 Finding the target or tracking the motion

For the target position estimation problem, two main approaches were evaluated. The first approach would be to try to find the target in each image using a template matching technique to find the target in the image as has been done in [1]. The second approach is to follow the motion of the weapon only and not make any assumptions about the target itself or try to recognize it in any way. To determine which approach is the best in this case, the strengths and weaknesses of each approach were analyzed.

3.1.1 Pattern recognition

This approach relies on the assumption that we have information about the target that makes it recognizable. Features or a template image that describe the characteristics of the target are required to be able to match the target.

Although very information-demanding, there are clear advantages with this approach. It is fast, since in the ideal case we would only need to process one frame to be able to obtain a good estimate of the target position in relation to the current aiming position.

If the system is designed to be applied for very specific cases, e.g. to shoot down aircraft, this could be realizable. However in our case the system needs to be more general, since the target could be almost any type of object in a real life situation. It would also be very hard to distinguish a camouflaged target against a background that is extremely similar to the target itself. Problems would also occur if the target is almost completely occluded, which would not be an unusual situation especially in hunting and military applications.

3.1.2 Image stabilization

The image stabilization approach instead relies on the assumption that we should be able to obtain a good estimate of the position of the target using knowledge of the motion of the weapon during the aiming period. In this case we can view the problem of finding the target position as a signal filtering problem, where the true signal (target position) is estimated from a series of noisy measurements. The problem of finding the global motion in a video stream has been solved for other applications such as digital video stabilization [2].

The main drawback of this approach is that we may need quite a few samples of the position of the weapon to be able to obtain a good estimate of the target position. This could make the system slow and unresponsive.

The biggest advantage of using image stabilization is that the system would be completely independent on the enviroment in which it is used as long as the motion can be tracked. Since the only information needed is the motion of the shooter, this would be a better approach in the sense that it could be used in a much wider range of applications.

3.2 Initial system layout

To get a good overview of what needed to be implemented, an initial flowchart of the demonstrator system was created. This chart describes the general functionality of each step that needs to be taken in order to reach the final goal of firing the weapon at the best time possible.

The first step in the process is to capture the current frame from the video camera. This frame contains all the information needed to perform the analysis.

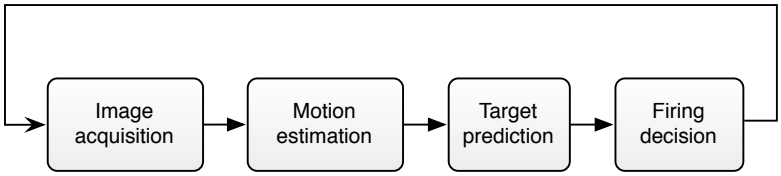
The second step is to estimate the motion of the aiming point. This motion is needed to be able to estimate where the target is and where the shooter is currently aiming.

From this motion, we need to estimate the point where the shooter intends to aim. This can be interpreted as a problem of finding the original signal (the motion of the target point) that is hidden in noise. Here, we define the unintentional motion as noise. This way the problem can be viewed as a pure signal processing problem, and solved with existing signal processing techniques.

When we have an estimate of the target point, we need to know when the weapon will pass this point the next time, so that the firing process can be initiated at the correct time. This means that we have to implement a short term prediction of the motion and an algorithm that decides when the time is right to fire the weapon.

This initial analysis of the system will be the basis for the development. A flowchart of this process is presented in figure 3.

Figure 3: Initial system flowchart



3.2.1 Hardware

To obtain the data needed to perform the image analysis, a video stream needs to be captured from the telescopic sight. A mount needs to be designed that makes it possible to capture video while the shooter is still able to aim through the sight.

4 System description

4.1 Hardware platform

To be able to obtain video from the telescopic sight, a camera mount had to be designed. A rough design of this mount was already available at Combitech AB, but some modifications had to be made to make the system more reliable and easy to use.

The mount uses a semi-transparent mirror placed at a 45 degree angle to the optical axis. This mirror acts as a beamsplitter. Approximately 50 percent of the light is transmitted through the mirror and into the eye of the shooter and the other 50 percent is reflected to the camera, which is placed orthogonal to the optical axis. This way the shooter can aim normally through the sight, while the camera can provide a digital video stream of the same view. The downside of this approach is that the scene will be a bit darker. A schematic illustration of the beamsplitter (figure 4) and the entire mount (figure 5) is shown below. The angle between the incident ray and the surface normal equals the angle between the reflected ray and the surface normal due to the law of reflection.

Figure 4: Beamsplitter, I = incident ray, R = reflected ray, T = transmitted ray

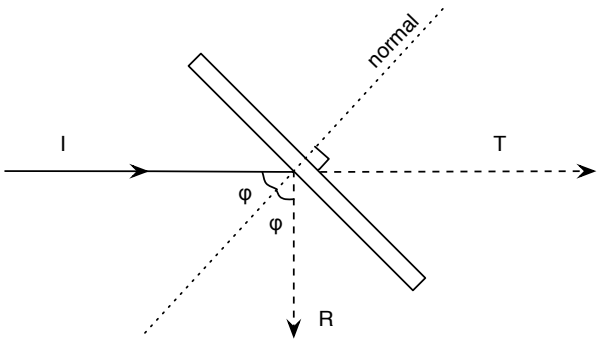
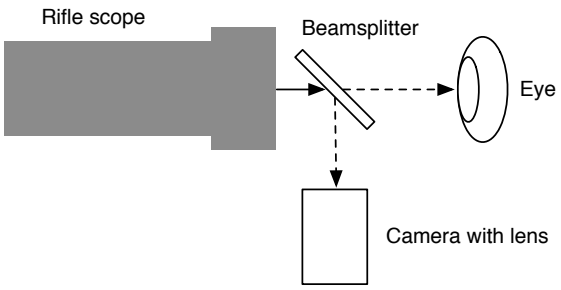


Figure 5: Mount schematic



The image sensor used in the system is a modified Sony Playstation 3 eye camera. The original optics for the Playstation 3 eye only supports two zoom levels corresponding to 56 or 75 degrees field of view. To be able to control the zoom level more exactly, the sensor is mounted on the back of the optical system of an old video camera. This makes it possible to adjust the focus and zoom of the optics manually using two small screws. These adjustments are necessary to be able to obtain a good image.

The reasons this camera was chosen were because of its low cost compared to professional camera options and high configurability and performance compared to most other consumer-grade miniature USB-cameras. There is a strong point in showing that the system can be built without the use of expensive equipment.

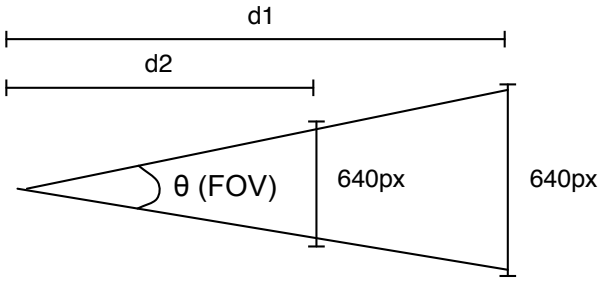
There is also a third-party API available for the camera from code laboratories. This makes it possible to adjust parameters such as exposure, frame rate and resolution directly in the program code by sending commands to the camera. Most other cheap consumer cameras don't have this kind of configurability but rely on general APIs such as Microsoft DirectShow, which offer a lot less flexibility when programming.

Initially, the system was tested using a cheap Logitech webcam. This proved to be a bad solution, since it was not possible to fix the frame rate. The frame rate could drop from 30fps to 10fps if lighting conditions changed slightly.

The Playstation 3 camera is able to output an uncompressed video stream of 640x480 pixels at a maximum 60 frames per second or a 320x200 pixel stream at 120 frames per second. The frame rate can be fixed.

The real-world resolution measured in meters per pixel depends on the zoom level (field of view) of the rifle scope and the distance to the target. In the tests carried out in this project, one pixel corresponds to approximately 3 mm at the target distance.

Figure 6: Absolute resolution field of view and distance dependence



4.2 Software platform

The choice of software platform was made considering that the system should operate on a PC but be easily ported to an embedded platform in the future. One other important factor to consider is that the software should be as fast as possible, as higher computational speed and efficiency can be used to increase the precision of the system. Thus a low-level programming language was used to utilize the hardware as efficiently as possible.

To be able to satisfy both the speed and portability demands of the system, the system is developed in the C++ programming language.

Many of the image analysis and image processing algorithms that will be tested and used in this project are widely used and implementations in C++ are already available. This makes the design process a lot quicker and more time can be spent looking at this specific implementation, as a smaller amount of time has to be spent implementing algorithms and functions.

One of the most widely used libraries for these implementations today is OpenCV [3], an open source library developed by Intel Corp. It contains a lot of advanced image analysis functions and is generally regarded as efficient and stable. It is also cross-platform and is free for both commercial and academic use. This library is used for most of the calculations in this project.

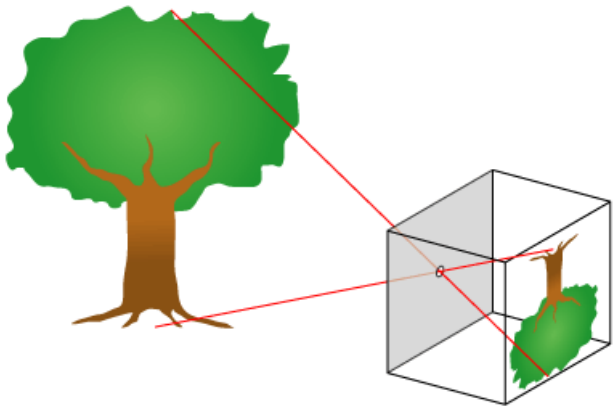
The developing enviroment chosen was Visual C++ 2010 Express on a windows XP laptop. The laptop is equipped with a single-core Intel Pentium M processor and 1GB of RAM.

4.3 Physical model

4.3.1 Pinhole camera model

The simplest way to describe the projection of the three-dimensional world into the image plane of a digital camera is the pinhole camera model. This model describes the aperture of the camera as an infinitely small point and neglects all lens effects. The drawbacks of using this approximation is that the effect of lens distortion is ignored. This effect should however be small enough not to cause any big problems in this case. It can also be reversed by using a transformation before doing the analysis. However this functionality is not implemented here. By using the pinhole camera approximation, calculations are simplified compared to using a more advanced camera model. This is a big advantage in this case, since the developed algorithm has to be as fast as possible to be able to run at high enough frame rates on the target hardware.

Figure 7: Pinhole camera model



The projection of a point in the real world on the image plane will be dependent on the angle to the aperture. Two objects at the same angle will be

projected on the same pixel in the image sensor.

4.3.2 2-D motion model

When aiming at the target, the shooter is assumed to be stationary. The main motion that is to be estimated is the pan and tilt of the gun barrel. Pan is rotation around a vertical axis and tilt is rotation around a horizontal axis perpendicular to the optical axis. The translational motion of the gun and the roll (rotation around the optical axis) are assumed to be small.

For small angles, the pan and tilt motion of the camera can be approximated as a translational motion of the field of view. In this case that assumption should be valid, since the angular motion when trying to aim steadily at a target should be small enough. Each pixel in the image then corresponds to a certain angle in the pan and tilt directions. The pan and tilt angular motion can thus be interpreted as moving the image plane left and right. This is a pure translational motion in the image plane. The motion can be described as $x_{t+1} = x_t + d_x$, where x_i is the x- and y-coordinate of a pixel at time t and d_x is the global motion of the frame. This is an approximation and won't be true for all pixels in the image. However the effects of this can be minimized by using techniques described in the next section.

4.4 Motion estimation

To be able to estimate the position where the shooter intends to aim, a good estimation of the motion of the point of aim has to be obtained. The sensor information available in this system is the video stream captured by the camera, so the motion estimation should be based on image analysis. Lots of work has been done in the field of estimating motion between subsequent frames in a video stream. The idea is to use information in two subsequent frames to be able to determine how much the scene has moved.

Two of the most common applications of inter-frame motion estimation in video streams are digital image stabilization and video compression. In our case we need to find an algorithm that is fast enough to run in real time on limited hardware and accurate enough to provide reliable estimates.

In digital image stabilization, the inter-frame camera motion is estimated in much the same way as will be the case in our system. So this was a natural starting point when trying to find good motion estimation algorithms.

The task of finding the motion vector field in a stream of images is referred to as optical flow estimation. The optical flow of an image is a vector field describing the motion of each pixel from one frame to another. In [4] and [5] several optical flow algorithms are thoroughly evaluated and compared considering processing speed and accuracy. In both papers, the Lucas-Kanade optical flow algorithm is concluded to have the best overall performance. This tracker was chosen for this project due to its low computational cost and high reliability. The version implemented is the one described in [6], which is a version of the Lucas-Kanade algorithm applied to image pyramids. This provides better performance when large inter frame motion is involved, and leads to higher robustness in the system.

The first approach to estimating the optical flow field of the field of view would be to simply try to estimate the optical flow for each pixel in the original

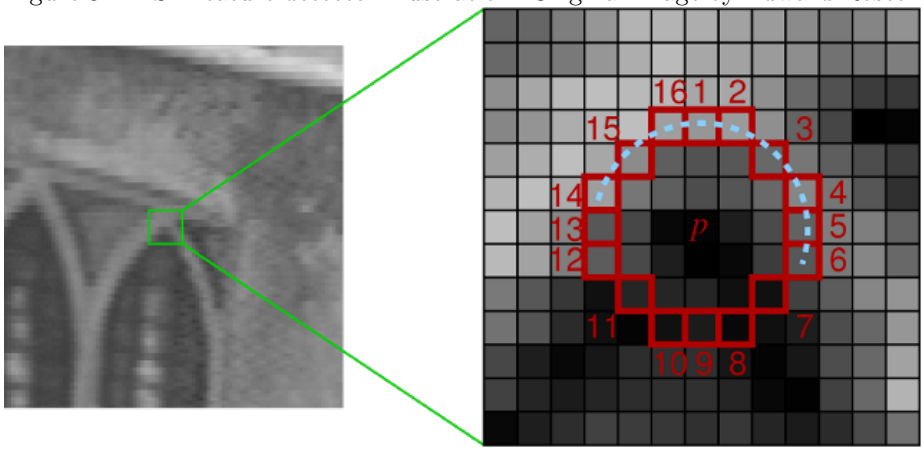
image. However, this approach would be highly computationally intensive, since one 640x480 image contains more than 300 000 pixels. Many pixels would also be impossible to track reliably. If the neighborhood around the pixel contains little or no intensity variation, there is no information that can be used to distinguish one pixel from another. This would lead to a lot of erroneous motion vectors that would make the estimated motion unreliable.

Instead, an approach that is widely used is to find the points in the image that are most suitable for tracking and only applying the optical flow algorithm in these points. This, however, can also be computationally intensive if the feature finding algorithm itself is slow. Two feature detectors were evaluated for use in this application. The good features to track algorithm by Shi and Tommasi as described in [7] was tried first, due to the fact that it finds the optimal features for use together with the Lucas-Kanade optical flow algorithm.

However, the Shi-Tommasi feature detector proved not to be fast enough to run smoothly in real time, especially on a low power embedded microprocessor. Instead, another feature detector called FAST, developed by Edward Rosten and Tom Drummond and described in [8], [9] was implemented. This lead to a huge performance increase. In the timing tests presented by Rosten/Drummond comparing the speed of FAST to several other feature datectors, including Shi-Tommasi, the increase in performance was drastic. FAST was able to process 179MPix/s while Shi-Tommasi only processed 6.50MPix/s. The percentage of successfully tracked features was comparable to the other, much slower detectors.

FAST is an abbreviation for features from accelerated segment test. It works by evaluating pixel values on a Bresenham circle of radius r around the point of interest. The point of interest is considered a feature point if n pixels on the circle differ by a threshold t from the intensity value of the point of interest. The version of the detector used uses a value of $r = 3$ and $n = 9$. The algorithm has been optimized using machine learning to minimize the number of logical statements that have to be evaluated for each pixel.

Figure 8: FAST feature detector illustration. Original image by Edward Rosten



The main drawback of this feature detector is that it is hard to control the

amount of features detected, due to the requirement of a threshold for deciding what is a feature and what is not. Since a stable execution time is desired, this can pose a problem if too many features are found and tracked.

The issue of limiting the amount of tracked features could be resolved by choosing a random subset of the detected features and pass them on to the tracking algorithm. However, it is desirable to use the most prominent features to make the tracking as reliable as possible. The solution to this problem is to make the threshold adaptive in the way that if too many features are found, the threshold is set to a higher value telling the algorithm to be more selective and vice versa. This way, the number of features do not vary too much and the highest scoring features are always used. The threshold is set using a proportional controller [10]. The threshold is increased or decreased by a value proportional to the error, $e(t)$, which is the difference between the number of detected features and the desired amount of detected features.

$$u(t) = K_p e(t)$$

Where K_p is a constant. $u(t)$ is then added to the current threshold, T

$$T(t+1) = T(t) + u(t)$$

Since T needs to be an integer, the value is also rounded off to the closest integer value. The new threshold is then used in the next iteration.

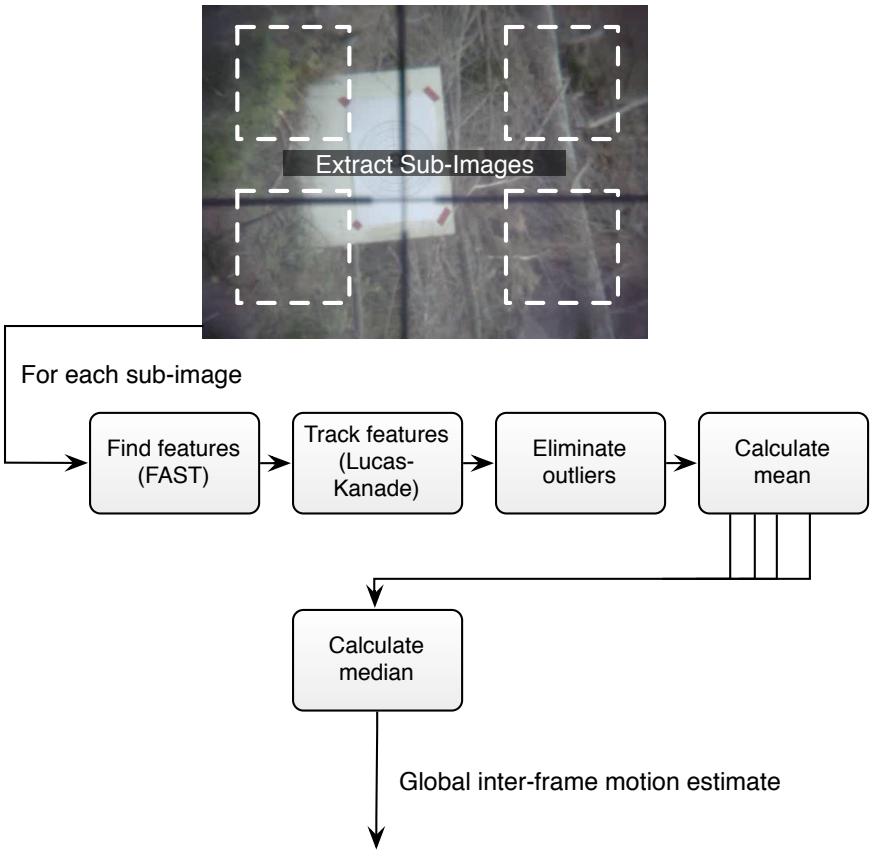
The constant K_p determines how fast the threshold adapts to changes in the image. If K_p is large, the system will compensate more and if it is small the system will be slower. If K_p is large, the risk of overcompensation will also be greater, which means that the system changes the threshold too much resulting in too many or too few feature points being found. The choice of the constant K_p is therefore a tradeoff between speed and overcompensation. The value for K_p was set using manual tuning to a value of 0.1. For this value, the overshoot was not too big and the system responded fast enough to changes in the scene. In reality, these changes are small due to the system being used only for a few seconds at a time in a set environment that does not change much during this limited period. The same algorithm is used for initial calibration of the threshold before the algorithm starts.

These two algorithms form the base of the inter-frame global motion estimation technique applied in this project. The simplest way to use them would be to apply the feature detector to a full frame of video and use the tracking algorithm to find the motion for these features from that frame to the next. Then the mean of all the motion vectors could be used as the global motion estimate. However, if an object in the frame is moving, the motion estimation would be affected in a negative way if features are assigned to pixels on the moving object. This way, leaves blowing in the wind or a bird flying past could affect the accuracy of the tracking.

The way this problem is addressed is to divide the frame into four sub-frames, where each sub-frame has its own threshold value. The global motion for each of these four sub-frames is then estimated in the way described above. The global motion for the entire frame is then calculated as the median of the four sub-frame motion vectors. This is similar to the method applied in [11]. This way if there is an object in one part of the image that is moving differently compared to the stationary background, this does not affect the inter-frame

motion estimate. The sub-frame division also makes sure that the detected features are spread out in all parts of the image. The general stucture of the inter-frame global motion estimation is depicted in figure 9.

Figure 9: Motion Estimation Algorithm



4.5 Motion pattern analysis

To be able to make correct assumptions when designing the system, the motion patterns that were obtained when performing tests on the system were analyzed to see if the assumptions were correct and to draw some conclusions regarding the speed of convergence of the estimates. Figures 10, 11 and 12 show some of the interesting characteristics of three different motion patterns produced by different users. The data from the algorithm was imported and analyzed in MATLAB to produce the plots.

Figure 10: Motion pattern for user 2 shot 3

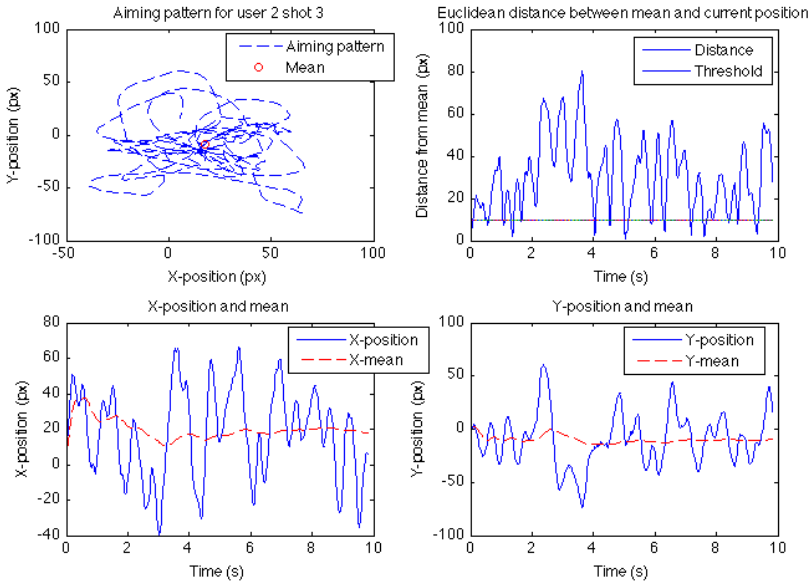


Figure 11: Motion pattern for user 3 shot 1

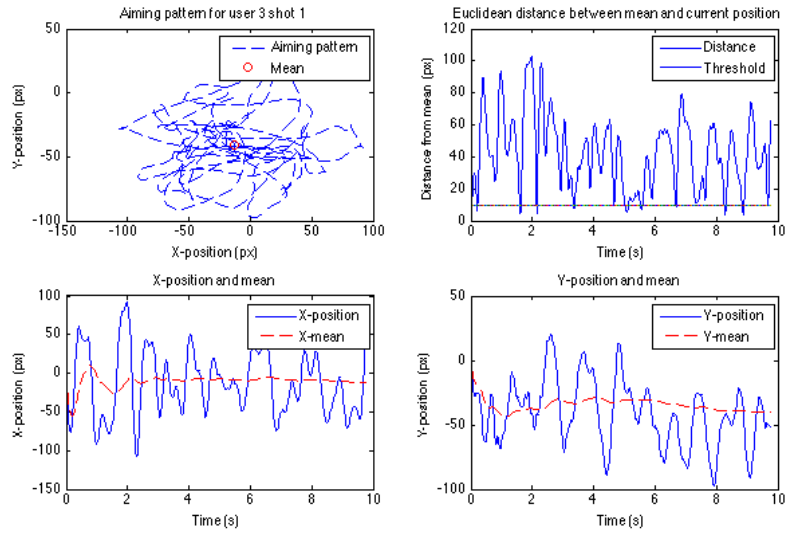
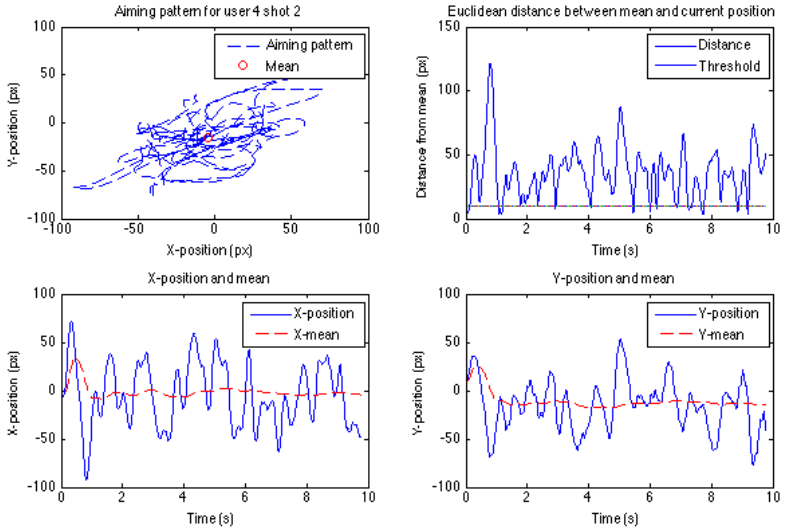


Figure 12: Motion pattern for user 4 shot 2



In the top left plots of each figure the motion pattern is shown in the x-y image plane. It looks like a distribution of points around the mean. The distributions look like they could be quite well approximated by a two dimensional gaussian distribution, since the points are denser around the mean and sparser further away from the mean.

In the bottom two plots, the x- and y-position and the x- and y-mean respectively are plotted as a function of time. From these plots it can be concluded that the mean seems to stabilize after around two to four seconds. Since we want to have a high reliability in the system, the system can be fired some time after the mean has stabilized. This time will be a limit for how quick the system can be from the time the user starts pressing the trigger to the time the system initiates the firing procedure.

In the top right plot, the distance from the mean to the current position is shown as a function of time. It can be seen that the current position passes close to the mean approximately one to two times per second when the mean has stabilized. These are the points in time when the weapon could be fired, since the goal is to fire the weapon when passing as close to the mean as possible.

4.6 Point of aim estimation

When trying to aim at a target, stationary or moving, there is always more or less unintentional motion added to the intentional motion of the shooter following the target. This motion is caused by forces originating from the body of the shooter and external forces such as wind. When studying these motions around the aiming point, one can see that they are approximately randomly distributed around the point at which the shooter tries to aim. A plot of these motions is depicted in figure 10. A small error is also introduced in the motion estimation algorithm. This error should also be normal distributed around the

true target point. We can therefore add these two distributions together to a single zero mean gaussian distribution, since they are independent.

The measured point of aim at a certain time, $z(t)$, can therefore be approximated by the position of the target, $x(t)$ with additive gaussian noise, $v(t)$ with mean zero and a certain covariance that depends on the skill and stability of the shooter and external influences.

$$y(t) = x(t) + v(t)$$

All of the variables in this equation are in this case two-dimensional vectors containing the x- and y-coordinates in the image plane.

The goal of our algorithm is to filter the noise $v(t)$ and estimate $x(t)$ from the noisy measurements $z(t)$.

In our case, where the target is assumed to be stationary, one can see that a good estimator would be the mean of the measurements, since the noise is randomly distributed with zero mean. As the number of measurements reach infinity, the mean of the samples would converge to the target position according to the law of large numbers.

4.7 Motion prediction

Since we don't have access to continuous measurements of the motion, only discrete samples taken with a small time difference, we need to be able to predict the motion between the samples. This is necessary to be able to fire the weapon at points in time that are between two samples.

To do this, a model has to be used to describe the motion that occurs between two subsequent samples. The model used in this system is a constant velocity model. This means that the velocity is assumed to be constant for the short period of time between two frames. So the aiming position one sampling interval ahead in time is predicted as

$$x_{t+1} = x_t + x'_t * dt$$

where x_t is the 2-D vector describing the aiming position at time t, x'_t is its derivative and dt is the sampling interval.

4.8 Firing decision

This part of the system is responsible for making the critical decision whether to fire the weapon or not. The two main concerns to be taken into account are safety and precision. It is crucial that the system does not fire without active action taken by the operator, and that the system has a good enough estimate not to fire in the wrong position. In this section criteria that are not possible to implement in the demonstrator system, but could be implemented in a final version of the system, are also discussed.

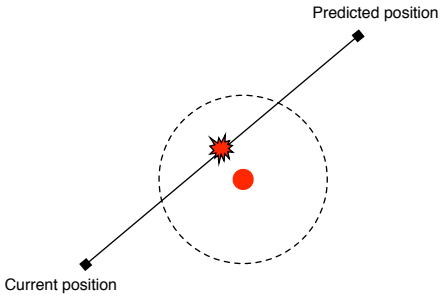
4.8.1 Criteria

The first and foremost criterium that has to be fulfilled in order for the system to initiate firing procedure is that the shooter has given the signal to the system to fire. In this case this is represented by the action of applying pressure on the trigger, slowly moving it inwards. As long as the trigger is moving inwards, the

system is in a state where it is cleared to fire. If the shooter at any point stops pulling the trigger or releases it quickly, the system is reset and can not fire. This is a very important safety feature and makes sure the shooter is always in control of the system.

The second criterium is also related to safety, but even more to accuracy. Before firing the weapon, the system needs to know that the estimated target position is reliable enough. This is implemented by looking at the derivative of the estimated target position. If the derivative is small enough, the target position is considered stationary and the shot can be fired.

Figure 13: Firing decision proximity criterium



The third criterium concerns the position in which the shooter is currently aiming, and the speed at which the weapon is moving. A line is drawn from the current aiming position to the position where it is predicted to be when the next frame will be received from the camera. If this line passes within a threshold distance to the estimated target position, firing will be initiated. The firing will be initiated with a delay, so that the weapon fires at the point on the line where it is closest to the estimated target position. This is illustrated in fig. 13.

4.9 Error sources

When estimating the motion of the weapon using the optical flow algorithm, there are always a percentage of the motion vectors that are not correctly computed. Using the outlier elimination algorithm most of these erroneous vectors can be filtered out. However, if the difference between the erroneous vector and the correct vectors is small enough, the global motion estimate will be influenced by vectors that deviate from the true motion in the system. If the motion between two frames is too big, the motion estimation algorithm will fail, since the window where the algorithm looks for features is limited. This window can be set to be bigger, but this influences the computation speed in a negative way. Currently there is no way for the system to tell if it looses the tracking because of too big motions. This has to be implemented in a finished product to ensure that it is safe to operate.

The firing decision is based on the assumption that the weapon moves with approximately constant velocity between two frames. If the shooter for some reason abruptly changes speed in the time between the frame when firing is initiated and the time when the weapon fires, the estimated position where the weapon fires will be wrong. However, if the sampling interval is short enough,

this error will be small.

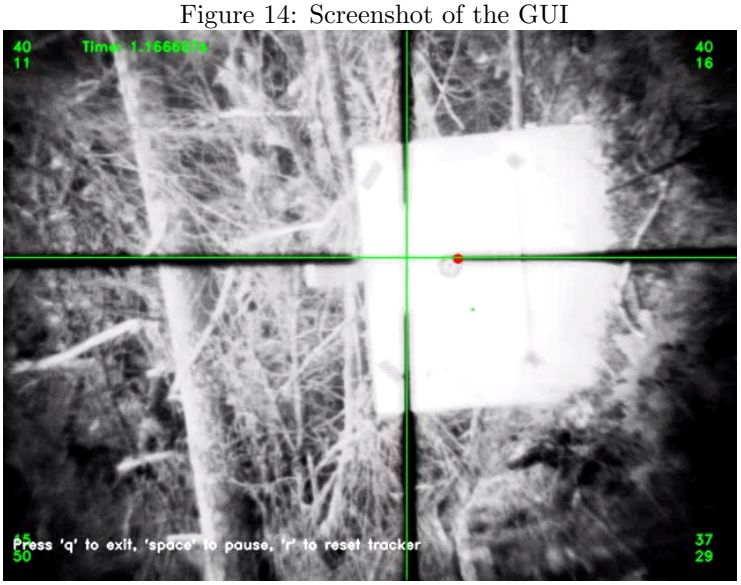
The assumptions made in the physical model of the system introduces errors.

If the global motion between frames is wrongly estimated, the coordinate system will drift away from its original position, causing the reference point of the entire system to be shifted. This shift is generally small, in the order of a few pixels. This corresponds to a few millimeters up to a few centimeters. It would be possible to compensate for this effect by doing some kind of image matching between frames with larger intervals to see how the frame has moved in a longer time perspective. This has not been implemented, but is possible to do in a future version of the system.

4.10 User interface

The user interface of the software was designed so that the tracking could be followed in real time and be easy to evaluate. The output window (fig. 14) shows the view through the scope and the information needed to follow the calculations of the algorithm. For each of the four tracking windows, the number of tracked features and the threshold level is displayed in each corner. The red dot represents the current estimated target position and the green dot represents the starting point of the tracking. The current time is also displayed for reference.

Figure 14: Screenshot of the GUI



When the shot is fired, the window stops and one more frame is grabbed. A line is drawn between the position of the last two frames and the position where the shot was fired is marked.

The input of the software can be either a recorded movie file or real time video from a camera. The output data is saved to a file in a format that can be easily imported into other software such as MATLAB for further analysis.

5 System evaluation

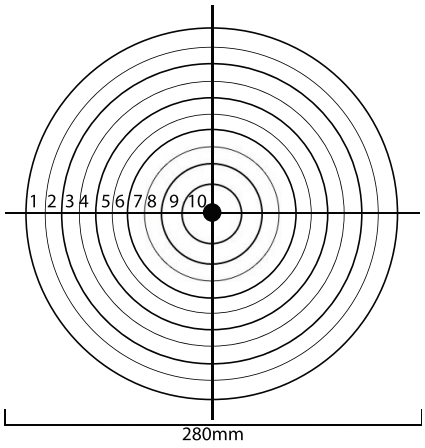
In this chapter, the methods for testing the system are described. Results from the tests are also presented.

5.1 Test method

The system was evaluated by letting four different persons with varying experience of firing hunting rifles try to shoot as well as possible at a target in a shooting range. Each of the four persons shot three series of five shots using no assistance from the system and one series each that was filmed for later testing with the system. Due to lack of time, user 4 shot only two series without assistance. The filmed series were run through the algorithm and the shots fired were simulated by the software so that the performance of the system could be compared to the performance of the same persons firing a weapon without use of assistance.

The performance of both methods is measured by looking at the scores the shooters were able to achieve when shooting at the target (fig. 15). The maximal score for one shot is 10 points. This test should give a good indication of whether the system is able to improve the performance.

Figure 15: Target



The point where the virtual shot hits is simulated by using the image before the shot was fired by the algorithm and the image after the shot would have been fired. The motion between these frames is then measured using the same algorithm the system uses for stabilization. The velocity of the vector field between the two frames is assumed to be constant, and the point where the shot hit is calculated by interpolating the distance which the weapon would move between the first frame and the delay set by the algorithm.

The test area was set up to be as close to a normal operating enviroment as possible. The test was carried out in a forest area, and the shots were fired from a hunting rifle. The distance to the target was 50 meters and the shots were fired from a standing position.

5.2 Results

The total scores of the shooting tests are presented in table 1.

	Series 1	Series 2	Series 3	With Assistance
User 1	28	14	24	39
User 2	36	32	18	43
User 3	18	5	0	39
User 4	1	8	-	31

Table 1: Total score for all users with and without assistance

The individual scores of users 1, 2, 3 and 4 are presented in tables 2, 3, 4, and 5. Observe that the shots are not in chronologic order for the series without assistance, since the shots were fired on the same target and there was no way of telling which shot was fired first.

	Series 1	Series 2	Series 3	With Assistance
Shot 1	8	6	5	9
Shot 2	7	5	6	8
Shot 3	8	2	7	7
Shot 4	5	1	6	7
Shot 5	0	0	0	8
Sum	28	14	24	39

Table 2: User 1 scores

	Series 1	Series 2	Series 3	With Assistance
Shot 1	10	9	6	8
Shot 2	9	9	6	8
Shot 3	9	6	6	8
Shot 4	4	5	0	10
Shot 5	4	3	0	9
Sum	36	32	18	43

Table 3: User 2 scores

	Series 1	Series 2	Series 3	With Assistance
Shot 1	7	5	0	6
Shot 2	5	0	0	7
Shot 3	6	0	0	8
Shot 4	0	0	0	7
Shot 5	0	0	0	9
Sum	18	5	0	37

Table 4: User 3 scores

	Series 1	Series 2	Series 3	With Assistance
Shot 1	1	7	-	6
Shot 2	0	1	-	5
Shot 3	0	0	-	5
Shot 4	0	0	-	10
Shot 5	0	0	-	5
Sum	1	8	-	31

Table 5: User 4 scores

6 Discussion and conclutions

6.1 System performance

In the tests carried out in the scope of this project, the system performed considerably better than what the users could achieve without using the aid of the system. The results were pretty clear and the conclusion that can be drawn from this is that the concept is good and should be a worthy candidate for further development and testing. One really interesting part of the results is the fact that the shots fired by the system were never worse than a score of 5, while the manually fired shots often missed the target completely. This implies that the system is very good at eliminating the shots that would otherwise have been bad hits or even misses. It is a very good property, since this could mean a big difference in real world situations.

However, due to lack of time, only a few tests were performed. To be able to achieve higher certainty regarding the performance of the system it should be more thoroughly tested with a wider range of scenarios and users.

One aspect of the tests that is hard to know the impact of is the psychological effect. Since the weapon was not fired when doing the tests with the system, the shooters might have been more calm and relaxed when performing that part of the test compared to the test using live ammunition. This could have affected the results in favor of the assisted shots.

6.2 Physical limitations

As was shown in the motion pattern analysis, the motion pattern has an irregular pattern with some periodic characteristics in the x- and y- direction when viewed over time. The periodic nature of the motion can probably be attributed to the way the human body and brain compensates for its own instability. When the point of aim moves away from the target a force is applied to move the point of aim back to the target. The frequency of the periodic component of the motion is around 3-4Hz. To be able to obtain a good and stable estimate of the target position, the system must follow the motion for around 1-4 seconds depending on the stability of the user. This time is a lower limit for how fast the system can be ready to fire in a safe way. When the mean is stable, in most cases there will be less than 0.5 seconds delay until the weapon is fired, due to the frequency of the oscillating motion with which the weapon moves.

6.3 Real-Time properties

The system was able to run in real time with a frame rate of 60fps on the hardware platform used during the tests. However, this hardware differs considerably from the target platform. To be able to run the system in real time on embedded processors further optimization must be performed to increase speed while maintaining good accuracy and robustness.

This should be possible to achieve, especially if the system is implemented on dedicated hardware. The algorithms used are easy to parallelize and it has been shown that performance can be increased a lot by doing so. In [12] a speed increase by a factor 100 was achieved when implementing the Lukas-Kanade optical flow algorithm with highly parallel computations on an NVIDIA GPU. This implies that a high degree of optimization can be made to further increase the speed of the computations.

6.4 Robustness

As the system is designed, there is always the possibility of the system losing the tracking of the motion. This will result in a displacement of the reference coordinate system. When this happens, the estimate will be faulty and the weapon could fire in a position that was not intended by the user. This problem needs to be addressed either by aborting execution when this happens or by finding a solution that eliminates this problem.

The optical system of the demonstrator system is not optimized for handling low light conditions and the system does not work very well in these conditions. If a more light sensitive camera sensor is used and an optical system that is more well suited for the application is designed, the system could be made more robust when it comes to lighting conditions. It would also be possible to use an infrared sensor to capture heat radiation instead of using the visible light spectrum without having to modify the system much.

6.5 Safety

Safety is always an important aspect to take into consideration when working with weapon systems. It is crucial that the system works as expected at all times and that the user is never surprised by the actions taken by the system. The user should always control the system and not the opposite. Since the system only operates and is able to fire when the user explicitly tells it to do so by applying pressure to the trigger, this should not be a problem. To further increase the perceived control, the estimated target position could be presented to the user while aiming by using a projected dot in the scope. Some kind of feedback could also be used to tell the user when the system has a stable estimate and is ready to fire.

6.6 Recommendations for future work

The most important step to take in order to evaluate the system further is to build a more robust mount for the camera that is safe to have attached to the gun while firing live ammunition. This way the uncertainty regarding the psychological effects of knowing that no shot will be fired will be eliminated.

Another important aspect to investigate is the real time behavior when running the system on an embedded platform. Implementation on either a DSP, FPGA or small processor would be a good step to take in order to make the system work without the need of using a PC to do the computations.

References

[1] F. Jurie and M. Dhome, "Real time robust template matching," in *in British Machine Vision Conference 2002*, pp. 123–131, 2002.

[2] Y. Matsushita, E. Ofek, X. Tang, and H.-Y. Shum, "Full-frame video stabilization," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 1, pp. 50–57, 2005.

[3] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly Media, Inc., 1 ed., September 2008.

[4] B. Galvin, B. Mccane, K. Novins, D. Mason, and S. Mills, "Recovering motion fields: An evaluation of eight optical flow algorithms," in *British Machine Vision Conference*, pp. 195–204, 1998.

[5] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *International journal of computer vision*, vol. 12, pp. 43–77, 1994.

[6] J.-Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm," 2000.

[7] J. Shi and C. Tomasi, "Good features to track," 1994.

[8] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking.," in *IEEE International Conference on Computer Vision*, vol. 2, pp. 1508–1511, October 2005.

[9] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, vol. 1, pp. 430–443, May 2006.

[10] T. Glad and L. Ljung, *Reglerteknik - Grundläggande teori*. Studentlitteratur, 4:3 ed., 2007.

[11] A. Yeni and S. Erturk, "Sast digital image stabilization using one bit transform based sub-image motion estimation," *Consumer Electronics, IEEE Transactions on*, vol. 51, pp. 917 – 921, aug. 2005.

[12] J. Marzat, Y. Dumortier, and A. Ducrot, "Real-time dense and accurate parallel optical flow using cuda," in *17th International Conference in Central European Computer Graphics Visualization and Computer Vision*, pp. 105–111, 2009.