UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

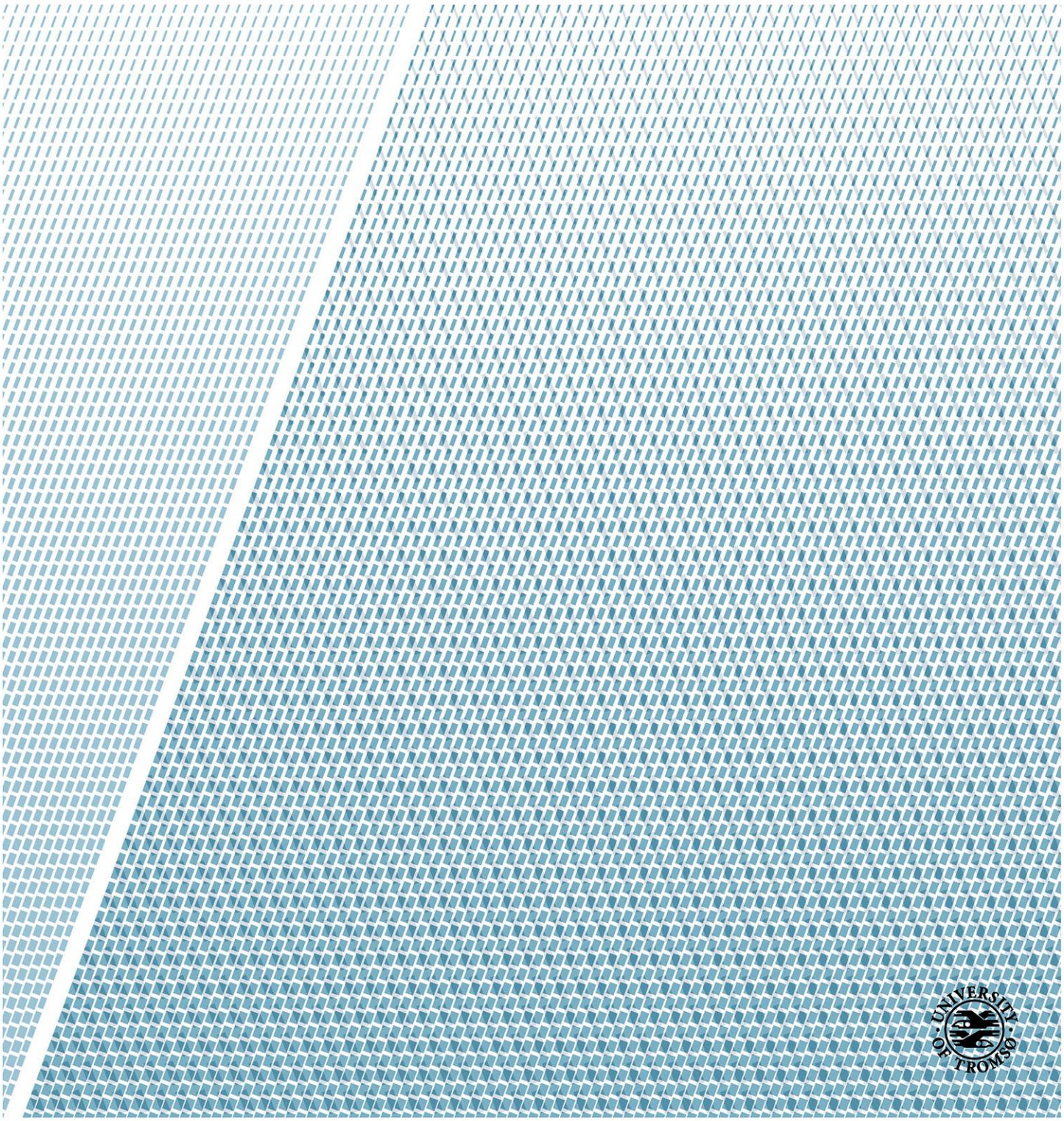
Faculty of Science and Technology
Department of Computer Science

Object tracking for improved telementoring and telestration

—

Dmitry Fatiev

INF-3997 Master's Thesis in Telemedicine and E-health - May 2015



Acknowledgement

I would like to express my gratitude to my two supervisors: Prof. Gunnar Hartvigsen (UiT/NST) and Dr. Etai Bogen (UNN) who provided me with guidance and advice throughout my work on this project.

I would like to thank my friends and relatives for their support and encouragement.

Abstract

Purpose

The purpose of this work is to estimate the performance of a TLD object tracking algorithm and its openly available alternatives in object tracking / camera motion estimation in laparoscopic surgery videos in context of telementoring and stabilized telestrating, and to propose and architecture and implementation of a prototype solution for stabilized telestration which does not require specialized costly equipment.

Motivation

Telementoring in image-guided surgery has long since become a worldwide trend, facilitating faster and better surgical expertise distribution, increased quality of learning for beginner surgeons and better clinical outcomes for patients who no longer have to wait for an expert to travel (or do not have to travel themselves). Since the early days of telementoring, telestration (freehand annotation) has been mentioned as an important and valuable part of surgical telementoring. However, to this day, no commercial solutions tailored for surgical telementoring and telestrating that provide telestration stabilization are present. During virtually any surgical telementoring session with telestration, camera movements occur and once the camera moves, the telestrations lose their value unless this movement is calculated and the telestrations' position is adjusted. Telestration in surgical telementoring needs more attention as an educational and mentoring tool, while at the same time being available without extremely costly equipment and providing all the possible functionalities that a surgeon might need.

Methods

A system that facilitates surgical stabilized telementoring while using the digital image information provided by a laparoscopic camera or a pre-recorded video file has been designed and implemented in modular way. The system was implemented using a combination of C#/Windows and C++/Ubuntu solutions. A literature review was conducted in order to collect the information about similar endeavors and experience of other research groups who work in this field. During the meetings with expert surgeons from UNN (University Hospital of North Norway), requirements for the systems were defined based on literature review findings and the surgeons' feedback. Finally, a system was tested with a web-camera video feed and a pre-recorded video file. Also, extensive testing has been performed using a number of conventional object trackers and a dataset of total 24 laparoscopic surgery videos.

Results

The architecture proposed and the system developed have been proven to be a viable experimental solution for surgical telementoring and telestration in image-guided procedures. After rigorous testing using a data set of 24 videos, the conventional object trackers including TLD and its alternatives have demonstrated that they are capable of tracking camera position in surgical video sequences. However, every object tracking algorithm has demonstrated certain weak and strong sides, according to the test results.

Conclusion

The system that was designed and implemented can be used for future work in improving the telestrating solutions for surgical telementoring, while seamlessly testing various camera position estimation solutions, which is made possible by the system's modular architecture. The architecture proposed makes it possible to perform a cross-platform (including mobile solutions) telementoring with telestrations. Conventional object trackers can be used while solving the task of short-term camera motion estimation in surgical videos.

Table of Contents

List of tables

List of figures

1. Introduction

1.1.Background and Motivation

Telementoring in minimally-invasive image-guided surgery has become increasingly popular and is already a part of many clinical routines in many hospitals. The benefits of telementoring in surgical procedures include significantly shorter patient recovery time due to decreased waiting time, decreased time spent in the operating room and no need to travel for patient and expert surgeon alike. Moreover, telementoring techniques facilitate a significant boost in surgical knowledge disseminations, allowing more inexperienced surgeons to learn given procedure remotely, while observing or being guided by an expert surgeon.

Telestration (free-hand annotation) is one important part of a surgical telementoring procedure that has been reported to be an extremely useful tool. The telestration tool allows a mentoring surgeon to communicate the information to a mentee much faster than it is usually done using verbal communication, which again decreases the time that the patient spends on the operating table.

There are commercial software and hardware systems that offer the telestration functionality, but it still remains vastly unsuited for certain scenarios that often happen while performing a telementoring session.

One of the problems mentioned in context of telementoring and telestration in image-guided surgery is the fact that camera may move while the telestration is still on screen, and the telestration becomes useless as soon as the camera moves.

1.2.Aims

Our main aim is to examine whether generic object trackers can be used for solving the camera movement problem in telestrations in surgical telementoring.

The second aim is to propose an architectural approach to the telestration system and develop a prototype that would prove the possibility of implementation of a non-expensive solution to the problem. A system should not require introduction of excessive costly into a traditional operating room without robotic equipment. The system should be flexible and versatile in terms of operating system on a user-device, and provide a means to test the solution of telestration movement problem using various means of digital image analysis.

1.3.Assumptions and Limitations

The telestration system is designed under assumption that it will be used as an experimental prototype in order to gather user feedback and test the various telestration tool features and image processing algorithms.

1.4.Contributions

A distributed architecture that makes the system extremely flexible and potentially cross-platform has been proposed. An experimental system for testing the telestration features and camera motion estimation has been implemented.

Through a series of tests, it has been shown that traditional "non-surgical" object trackers can be used to solve camera motion estimation problem in some scenarios in surgical video.

Tracker algorithm analysis has been performed based on the test results, providing the information about strong sides and limitations of the algorithms inspected.

1.5.Report Structure

This report is structured as follows:

Theoretical Background

This chapter describes the basic concepts of image processing which are necessary to understand how the image processing algorithms work.

Theory

This chapter explains how the tracking algorithms that are subject to experiments in this project work. The theory behind the image enhancement techniques are also described in this chapter.

Related Work

The Related Work chapter describes the approaches that are commonly used to solve the problem of camera motion estimation or object tracking in the surgical context. Advances in telestration for surgical telementoring are also described in this chapter. Moreover, the chapter gives a highlight on the pioneers in the related fields.

Methods and Materials

This chapter describes the way the insight into the field has been acquired, the way the system is developed, the way the data set has been created and the way the testing of all the subject trackers has been performed.

Requirements Specification

This chapter describes the functional and non-functional requirements to the system and the way they were created.

System Design

This chapter describes the way the system has been designed in order to fit the requirements specified earlier.

Implementation

This chapter describes the system implementation in more details.

Testing

This chapter describes the way the system was tested and the way the trackers in question were tested. The Testing chapter also describes how a data set for testing has been created.

Conclusion and Future Work

This chapter makes the concluding remarks and highlights the possible direction for future work and improvements.

2. Theoretical Background

The theoretical background chapter covers basic concepts in image processing that are the fundamental blocks for a vast majority of computer vision algorithms.

2.1.Notation

Digital **image / frame** retrieved by a camera is a matrix M, having dimensions $H \times W \times C$, where W is the number of columns (otherwise known as width), H is the number of rows (otherwise known as height) and C is the number of channels (e.g. RGB image contains 3 channels – red, green and blue, while a grayscale image only contains one that contains grayscale value intensities). One value in such a matrix at row i and column j is called a pixel and is denoted as $p(i, j)$. Depending on the number of channels, pixel may contain either a single intensity value (e.g. grayscale images) or a tuple of several intensity values for each channel (e.g. RGB images). Depending on the specific image format (e.g. how many bits are used to encode each pixel), intensity values may vary from 0 to $2^n - 1$ where n is the number of bits used to encode one intensity value (e.g. for an image using 8-bit format, intensity values will vary from 0 to 255).

A **video** or **video stream** is a sequence of N digital images $M_1, \dots M_N$ that possess the same $H \times W \times C$ dimensions. Images that are a part of video are commonly delivered at a certain rate of frames per second. Real-time algorithms are supposed to operate at framerates that are close (or higher) than 24 frames per second. In general, the more frames per second a camera can deliver and the more frames can be processed by an algorithm, the better results are acquired.

2.2.Image Features

An image feature is a term used in computer vision and image processing that describes a certain part of a digital image that is valuable for analysis of the given image and has certain informative or required properties (e.g. color, brightness, position).

It is common to describe a feature by its location, size (bounding box), pixel intensity values and some information about its surroundings, depending on the specific feature type.

There are several feature types commonly used in computer vision algorithms and methods.

Edges are points of the image that have dramatic intensity changes around them. Some approaches include a certain amount of neighboring points into an edge feature set based on parameters like edge smoothness, gradient value and shape (see Figure 2.1).

Figure 2.1Edge detection example[1]

Corners are locally 2-dimensionally structured features that are described by high amounts of curvature in the gradient of an image. Corners typically reside in places where edges rapidly change their direction (see Figure 2.2), but are not limited to this case. It became quite common to name such features interest points because of that.

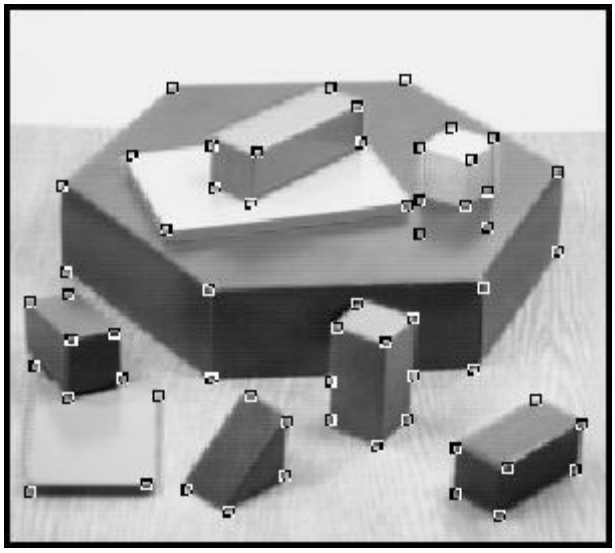


Figure 2.2 Corner detection example[2]

Blobs represent more informative chunks of data about image regions instead of separate small features. Blobs may contain regions that do not contain gradient sharp enough to be recognized by edge or corner detection, or regions with generally high or low intensity values. Blobs may also describe regions encircled by edges (see Figure 2.3).

[1] http://www.comp.leeds.ac.uk/viznet/reports/GPU_report/pictures/PyGPULena.jpg

[2] http://www.ee.surrey.ac.uk/CVSSP/demos/corners/blox-css.jpg

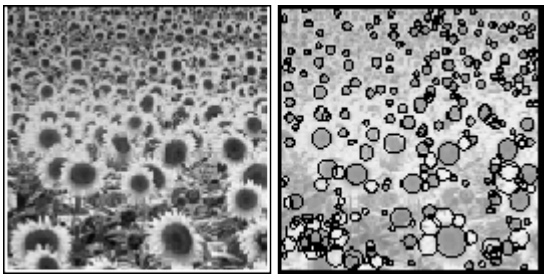


Figure 2.3 Blob detection example[3]

Haar-like features are a special kind of features found in digital images. Haar-like features have first found their application in face-recognition. They were developed by Viola and Jones (1) as a way to tackle computational expensiveness of other feature types, while maintaining sufficient meaningfulness for tracking or detecting applications. Haar-like feature contains rectangular image regions that are adjacent. The intensity differences between pixels in all the regions are calculated and also become a part of feature description. However, it is mentioned that one Haar-like feature is not enough to be a strong object classifier hence a set or a cascade of such features is required to perform robust object detection using a detection window and a grid, each cell of which has Haar-like features calculated for it.

Integral images known as summed area tables are used in Haar-like features' detection and speed up the method to a great extent. An integral image is a data structure designed to facilitate quick computation of value sum in a rectangular subset of a grid. (2)

In an integral image, value at given (x, y) point equals the sum of all values to the left and above of the given point, including the value in the point itself (see 2.1).

$$I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y')$$

2.1

One of the greatest advantage of this method is that it is possible to compute an integral image in one iteration over the image itself either iteratively, or recursively by 2.2.

$$I(x,y) = i(x,y) + I(x-1,y) + I\,(x,y-1) - I(x-1,y-1)$$

2.2

Once the integral image has been computed for the whole image or its subset, a value for any desired rectangle can be retrieved using 4 values in constant time. Given that a rectangle of interest is bound by 4 points A, B, C and D with respective coordinates $(x0, y0)$, $(x1, y0)$, $(x0, y1)$, $(x1, y1)$, the sum will be defined by 2.3:

[3] http://www.nada.kth.se/~tony/cern-review/cern-html/img56.gif

$$\sum_{\substack{x0<x\leq x1\\ y0<y\leq y1}} i(x,y) = I(D) + I(A) - I(B) - I(C)$$

2.3

2.3.Feature Detection

Methods that are used for deciding whether there is an image feature at certain image point or region are called methods of feature detection.

In many cases when computer vision algorithms require a set of certain features, the speed of feature detection is crucial (e.g. in order to facilitate real-time image analysis), hence there are several feature detectors available and researchers still look for faster or more robust methods to perform detection. There are several feature detectors that are most popular in object tracking applications.

FAST

Features from Accelerated Segment Test (3) is a feature detector that is capable of detecting corners. The most outstanding property of this method is its speed (computational efficiency). It is reported to be faster than feature detection methods such as SUSAN, Harris and difference of Gaussians used by SIFT (4). Further speed improvement is possible when machine learning enhancements are applied to FAST, making it a perfect candidate for live video processing.

SIFT

Scale Invariant Feature Transform method (4) is based on an assumption that for every object there is a feature description that can be extracted from an image. SIFT object features can help locate the object under different amounts of scale because of their scale invariant nature. It is also reported that SIFT features can help locate an object under certain degrees affine distortion, illumination and orientation changes. SIFT object descriptor is tolerable to a certain degree of errors in terms of relative (one of requirements for scale invariant nature) distance between points and the more features are selected, the more tolerable it is.

SURF

Speeded Up Robust Features (5) is a feature detector that finds its applications primarily in object recognition reconstruction of 3-dimensional spaces. The method's core idea has similarities with that of SIFT feature detector. Authors claim that their SURF outperforms SIFT to a great degree (several times greater performance). SURF feature detector makes efficient use of **Haar-like features** and summed area tables otherwise known as **integral images**.

2.4. Motion Field

Motion field represents visible motion caused by movements of camera or scene relative to each other. It is based on assumption that every visible 3D point of the scene is projected to a certain 2D point in the image received by a camera and those projections change over time depending on object or camera movements. A motion flow depicts those changes in point positions. (6)

Motion field can be calculated using various methods that fall into two categories:

Differential techniques are direct methods that analyze pixel intensity variations with respect to time and space. (7)

Matching techniques are methods classified as indirect. In this case, a set of features is selected prior to estimating the motion field. The features are then matched between sequential frames and the motion field is retrieved.

Depending on the application and its demands, it is possible to conduct a dense or sparse motion field estimation. Dense motion field relies on a larger amount of features to be calculated and is thus more expensive in terms of processing power and time. (6)

2.4.1. Optical Flow

Optical flow belongs to a differential class of methods for motion field estimation. Optical flow can be calculated between two consecutive frames of a video, indicating observable displacements within the picture. (6)

For a two-dimensional case and two frames retrieved at t and $t + \Delta t$ moments in time, optical flow is calculated under the assumption that a voxel at (x, y) at moment t will be displaced by $(\Delta x, \Delta y)$ at moment $t + \Delta t$ and will keep its brightness, making the statement 2.4 true (8):

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t)$$

2.4

Under the assumption that the movement was small enough, we can apply Taylor series logic to transform 2.4 into:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0$$

2.5

$$\frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0$$

2.6

$$(\nabla I)^T v + I_t = 0$$

2.7

with $v = \begin{matrix} \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial t} \end{matrix}$ and $\nabla I = \begin{matrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{matrix}$ in 2.7. This equation, however, cannot be solved because it has to be solved for two variables, which leads to introducing new equations, assumptions and constraints.

Optical flow estimation techniques are widely used in robotics for object tracking and detection, and for 3-dimensional environment perception.

Lucas-Kanade is a method to calculate optical flow differentially. It works assuming that optical flow is constant in the neighboring pixels of a target pixel (plus the basic

assumption of optical flow stating that displacement is small enough). The fundamental optical flow equation is solved for all neighboring pixels using the criterion of least squares. (9, 10)

The method is reported to be more tolerable to image noise because it is operating on a set of pixels in the neighborhood of a pixel in question. Such a set is commonly a rectangle R as big as $n = N \times N$ pixels, where N is often set to 5. The method works by minimizing the sum of squares error:

$$e(v) = \sum_{p_i \in R} [(\nabla I(p_i))^T]\, v + I_t(p_i)]^2$$

2.8

It is possible to represent 2.7 for a set R of pixels $p_{1\ldots}p_n$ as:

$$Av = b$$

2.9

in which

$$A = \begin{matrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{matrix}, \qquad v = \begin{matrix} v_x \\ v_y \end{matrix}, \qquad b = - \begin{matrix} I_t(p_1) \\ \vdots \\ I_t(p_n) \end{matrix}$$

2.10

And the solution to 2.9 is

$$v = (A^T A)^{-1} A^T b$$

2.11

It is by far not always true that the displacement between two consecutive frames is indeed small because it depends on the speed of objects or camera and the number of frames captured per second. This problem can be solved by applying Lucas-Kanade technique multiple times on an image pyramid in which each consecutive level contains an image with its resolution decreased by a factor of 2 in relation to the previous. LK is performed on the level with the lowest resolution and this result is them passed on to the higher resolution levels.

There are several known problems with this method, however.

Objects can move independently from each other, **overlapping** at will, which will make it harder to analyze target object motion.

The camera itself may introduce **distortions** and **noise** which will affect pixel displacement and grey values which are crucial for the method to work properly.

Grey values may also change their intensity because of **illumination changes**. Light sources that change their position over time may cause alterations in grey values despite the fact that no motion is really happening in the scene.

2.4.2. Feature Matching

A subset of techniques for indirect motion field calculation is called feature matching. The problem is solved by locating a number of features in one frame and then in the next one, then trying to pair the features between the two. Depending on feature descriptors, different methods may be applied. Distance measures (e.g. Euclidean, non-Euclidean) used depend on descriptors and methods chosen. (11)

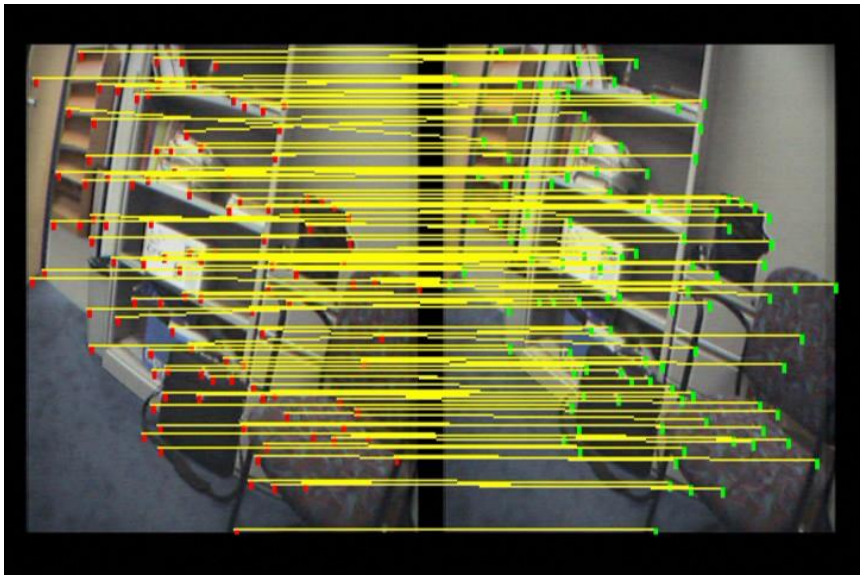


Figure 2.4 Feature matching example[4]

2.4.3. Feature Tracking

As opposed to detection, tracking techniques are more efficient in terms of computations. They also obviously outperform optical flow calculation for all pixels of a given frame. One way to do so is to retrieve a set of features and perform tracking utilizing the Lucas-Kanade method. (9)

2.5. Motion Model

Motion models are utilized to describe the motion that happens within the frame using several degrees of freedom (DOF). Depending on the number of DOF used in given model, pose estimation/object tracking may become more or less computationally expensive.

It is common to represent a motion model using a transformation matrix:

$$H = \begin{matrix} a & c & 0 \\ b & d & 0 \\ dx & dy & 1 \end{matrix}$$

2.12

Different motion models are used depending of the desired or affordable degrees of freedom number.

Translation motion model has 2 degrees of freedom and describes translation along x and y axis.

[4] http://www.consortium.ri.cmu.edu/data/FeatFramework/featureMatching.jpg

$$H = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{matrix}$$

2.13

Similarity model has 4 degrees of freedom including translation, uniform scale and rotation.

$$H = \begin{matrix} a & -b & 0 \\ b & a & 0 \\ dx & dy & 1 \end{matrix}$$

2.14

Affine model adds two more degrees of freedom totaling to 6: translation, rotation, shearing and scaling independently.

Retrieving a feature's position after the transformation has been applied is as simple as multiplying its transposed original position by transformation matrix H.

2.6. Random Sample Consensus Algorithm (RANSAC)

RANSAC is an algorithm developed by Fischler (12) for removing outliers from data sets with relatively large numbers of errors such as feature trackers of all kinds, which makes is especially useful for many applications in image analysis field. It is commonly used in conjunction with feature trackers or matchers in order to get a set of features that fit a certain model or assumption.

The algorithm iterates over a set of data K times. First, a random set of N entries is selected, model parameters are then estimated and the number M of entries that fit the model is calculated, labelling such entries as inliers based on a certain tolerance value ε. As soon as the number of inlier entries becomes greater than a specified threshold value δ, the model is estimated again using the newly retrieved inlier entries.

Figure 2.5 A set of points before RANSAC applied[5]

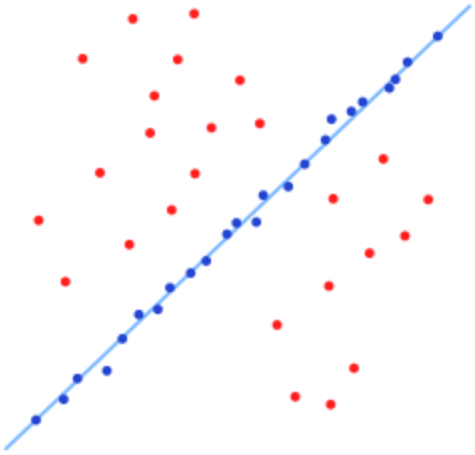


Figure 2.6 A set of points after RANSAC applied[6]

A set of data that better fits the model is likely to be obtained if more iterations of RANSAC are performed.

2.7.Visual Odometry and Egomotion

Determination of position and orientation of a system via analysis of its camera pictures is called visual odometry (13). Visual odometry is heavily used in robotics in addition to GPS, accelerometers and other means to estimate the system position in space (e.g. wheel rotation monitoring). There are several advantages of visual odometry as opposed to its alternatives:

- Does not rely on complex satellite systems or external communications
- Tolerant to wheel slip
- Only relies on visual input

Estimation of camera position and motion relative to a scene (in some cases, an object) is called egomotion.

Common approach to solving the visual odometry or egomotion problem has several steps.

1. Retrieve sequential input images using available camera(s).
    a. Monocular (single camera)
    b. Stereo (two cameras)
    c. Omnidirectional camera
2. Preprocessing
    a. Image enhancement
        i. Distortion correction
        ii. Brightness/contrast enhancement
        iii. Noise removal
        iv. Sharpening

[5] http://upload.wikimedia.org/wikipedia/commons/thumb/b/b9/Line_with_outliers.svg/383px-Line_with_outliers.svg.png

[6] http://upload.wikimedia.org/wikipedia/commons/thumb/d/de/Fitted_line.svg/383px-Fitted_line.svg.png

3. Detection of features
    a. Match features between consecutive frames
    b. Optical flow calculation (e.g. Lucas-Kanade algorithm)
4. Outlier removal (often using RANSAC implementations)
5. Camera self-movement calculation using the resulting optical flow.

Approaches that utilize more than one camera tend to produce far less errors and provide additional information about the surrounding environment.

2.8.Surgical Telementoring and Image-guided Surgery

Telementoring is a form of mentoring that includes providing advice and expertise to a remote mentee. Image-guided surgery is a name for surgical procedures that involve indirect guidance provided by imaging techniques. In the modern world imaging techniques are commonly digital and are an integral part of **minimally-invasive** surgical operations (e.g. laparoscopic, endoscopic prucedures).

Telementoring techniques evolve together with means of communication and nowadays almost always include video and audio translations between the remote mentor and the surgeon on sight. Moreover, often several video streams are transmitted to the mentor's viewport (e.g. overall operating room view, laparoscopic/endoscopic image and a close up view of the patient). (14)

Telestration is a recently emerged method of communication between the mentor and the mentee added on top of a simple video and audio translation layer. In a broader sense telestration involves any kind of drawing or annotation created by either side in order to directly specify regions of interest avoiding prolonged verbal descriptions and thus both saving time and increasing the quality of remote mentoring.

Telestration is commonly used in weather forecasts and sports events broadcasts or discussions. (15)

3. Theory

This chapter explains the theory behind the computer vision tracking algorithms and techniques used to implement the system.

3.1.Consenus-based Matching and Tracking (CMT)

Nebehay and Pflugfelder (16) propose a keypoint-based method for tracking objects in a long-term and in a model-free fashion, which means that no preliminary learning of tracker is required for it to function normally. This feature of the proposed algorithm makes it possible to use the tracker in quite various scenarios.

The proposed tracker is capable to estimate the object center position, scale and the degree of in-plane rotation. The structured pseudocode description of this approach can be seen on (Figure 3.1)**.**

Algorithm 1 CMT

Input: $I_1, \ldots, I_n, b_1$
Output: $b_2, \ldots, b_n$

$O \leftarrow \text{detect}(I_1, b_1)$
$K_1 \leftarrow O$
for $t \leftarrow 2, \ldots, n$ **do**
  $P \leftarrow \text{detect}(I_t)$
  $M \leftarrow \text{match}(P, O)$
  $T \leftarrow \text{track}(K_{t-1}, I_{t-1}, I_t)$
  $K' \leftarrow T \cup M$
  $s \leftarrow \text{estimate\_scale}(K', O)$
  $\alpha \leftarrow \text{estimate\_rotation}(K', O)$
  $V \leftarrow \text{vote}(K', O, s, \alpha)$
  $V^c \leftarrow \text{consensus}(V)$
  $K_t \leftarrow \text{vote}^{-1}(V^c)$
  if $|V^c| \geq \theta \cdot N^O$ **then**
    $\mu \leftarrow \frac{1}{n} \sum_{i=1}^{n} V_i^c$
    $b_t \leftarrow \text{bounding\_box}(b_1, \mu, s, \alpha)$
  else
    $b_t \leftarrow \emptyset.$
  end if
end for

Figure 3.1 CMT explained in pseudocode[7]

At the very first iteration, a set of keypoints is selected from inside the initial bounding box. **BRISK** (17) is used for keypoint selection and description. Each point is assigned its own binary descriptor. The initial object model thus contains from keypoints that are inside the bounding box and their respective descriptors.

[7] 16. Nebehay G, Pflugfelder R. Consensus-based Matching and Tracking of Keypoints for Object Tracking.

During the upcoming iterations, candidate **matches** for previously selected keypoints are found in the picture and again their binary descriptors are calculated. **Hamming distance** (18) is then computed between the old and the new descriptors.

Candidate points from the newly detected point set have to be closer to the nearest neighbor than the second nearest neighbor by a certain ratio. This way, points that belong to background are excluded from the set and a resulting set consists of a subset from the current frame.

After that, **tracking** takes place and Lucas-Kanade pyramidal varioation (9) is used for optical flow calculation. The set of tracked keypoints is retrieved by updating their positions based on optical flow. Keypoints that then turn out to be outside a bounding box are considered to be tracking failures and are removed from the set.

The **tracked** and **matched** sets are then merged. Keypoints that are present in the matched set are removed from the set that is tracked because successful matching is considered more robust than tracking because matching does not rely on previous estimations.

The next important step of the algorithm is keypoint voting. Each keypoint produces a vote for the object center position. In a more simple form keypoints vote for horizontal and vertical position only (translation).

Afterwards, outliers are removed from the keypoint set right before the consensus calculation. The image is clustered into several regions and the cluster with most votes inside becomes the core cluster. Votes from this **consensus** cluster are then used to compute the current object center.

3.2.Median Flow Tracker

Median flow tracker proposed by Kalal et al. (19) addresses the problem of tracking, stressing the fact that the object's appearance often dramatically changes over the course of tracking. The method introduces a possibility of the tracker to evaluate itself. It is based on the assumption that tracking results should be consistent independently of the direction of time flow.

Trajectory validation is performed in 3 steps. First, the tracker produces a trajectory while following the point normally (with time flowing forward). Then another trajectory is calculated by backward tracking the point. And lastly, the two are compared. It is common to describe points that are being tracked with surrounding patches using sum-of-square differences (20, 21), which helps to detect tracking errors related to fast movements or specular occlusions. However, this approach has trouble detecting trajectories that drift slowly.

Lucas-Kanade (9, 22) method is employed to track the selected features. Features displaced less than 2 pixels were considered to be inliers in this case. Lucas-Kanade method calculates the sparse motion flow between two given frames. The flow is only calculated for a bounding box. The resulting trajectory is then evaluated by the forward-backward method.

Kalal et al. (19) solve the tracking initialization problem by first tracking a lot of pixels from the initial frame. Resulting trajectories are then evaluated using forward-backward

error and each pixel is assigned a value which is the error of its respective trajectory. The error values are calculated using sum-of-square differences. An error map consisting of all the pixels tracked shows which pixels can be tracked reliably. If a point received a large error value, it is excluded from the set of points that are being tracked. The ones that remain are later used to estimate the new position for an object bounding box. The bounding box's new position is calculated as a median of all the tracked points' trajectories. The overall iteration description can be depicted by (Figure 3.2)

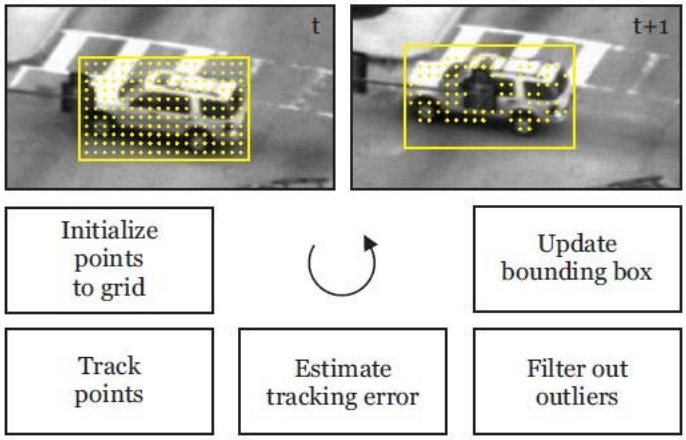


Figure 3.2 Forward-backward error method[8]

The forward-backward error approach can possibly be integrated with other tracking frameworks and significantly improve their quality.

3.3.On-line Boosting

3.3.1. Tracking

Grabner et al. (23) proposed a tracker based on on-line AdaBoost algorithm designed for feature selection. The presented tracker is capable of on-line learning. The model of a tracked object is updated by positive and negative examples from the frame being processed. The algorithm operates on gray values only, which means that it does not require and RGB image.

The tracking component is designed to solve the task of tracking as a binary classification problem. The algorithm robustness is achieved by updating the existing binary classifier as the tracking goes on.

The object that needs to be tracked is assumed to be present in the initial bounding box, which is provided as an input at the very beginning. The region specified by the initial bounding box is thus considered to be a positive example of object appearance for the tracker. The object surroundings, on the other hand, are labeled and remembered as negative examples. The initially obtained samples are crucial to initialize the on-line boosting algorithm by running several iterations in the very beginning using that data only.

[8] 19. Kalal Z, Mikolajczyk K, Matas J, editors. Forward-backward error: Automatic detection of tracking failures. Pattern Recognition (ICPR), 2010 20th International Conference on; 2010: IEEE.

Grabner et al. (23) apply classical approach in tracking steps, typical for template tracking, described by Hager et al. (24). First, current classifier in the region of interest is analyzed. Each position then yields a certain confidence value, which allows to analyze a resulting confidence map and translate the bounding box to a different location, following an object. Using mean shift procedure as described by Comaniciu et al.(25) can be used for improving detection. Furthermore, should a motion model be applied, it would be possible to reduce the search window. After the bounding box has been translated in accordance with confidence map values, the classifier has to be updated in order for it to adapt to object's possible appearance changes. Once again the current selected region becomes a positive example and the surroundings are sampled into negative examples.

3.3.2. On-line AdaBoost

The on-line boost concept that has been introduced into a tracker by Grabner et al. (23) is based on **weak** and **strong classifiers** and **selectors**.

Weak classifiers have weak requirements and have to yield decisions that are slightly more accurate than those of random guessing process. A weak hypothesis h^{weak} corresponds to a feature and is generated by a weak classifier via a learning algorithm.

A **selector** operates on a set of weak classifiers $H^{weak} = \{h_1^{weak}, \dots, h_M^{weak}\}$ with a hypothesis and choses one of them based on an optimization criterion. In this case every classifier has an estimated error value e_i and the selector chooses one classifier with the smallest e_i.

Strong classifiers are computed based on a linear combination of the weak ones. For a sample x the strong classifier binary decision and its confidence value are defined as follows:

$$hStrong(x) = \text{sign}(conf(x))$$

3.1

$$conf(x) = \sum_{n=1}^{N} \alpha_n \cdot h_n^{sel}(x)$$

3.2

The following diagram (Figure 3.3) provided by Grabner et al. (23) illustrates the decision process for one sample.

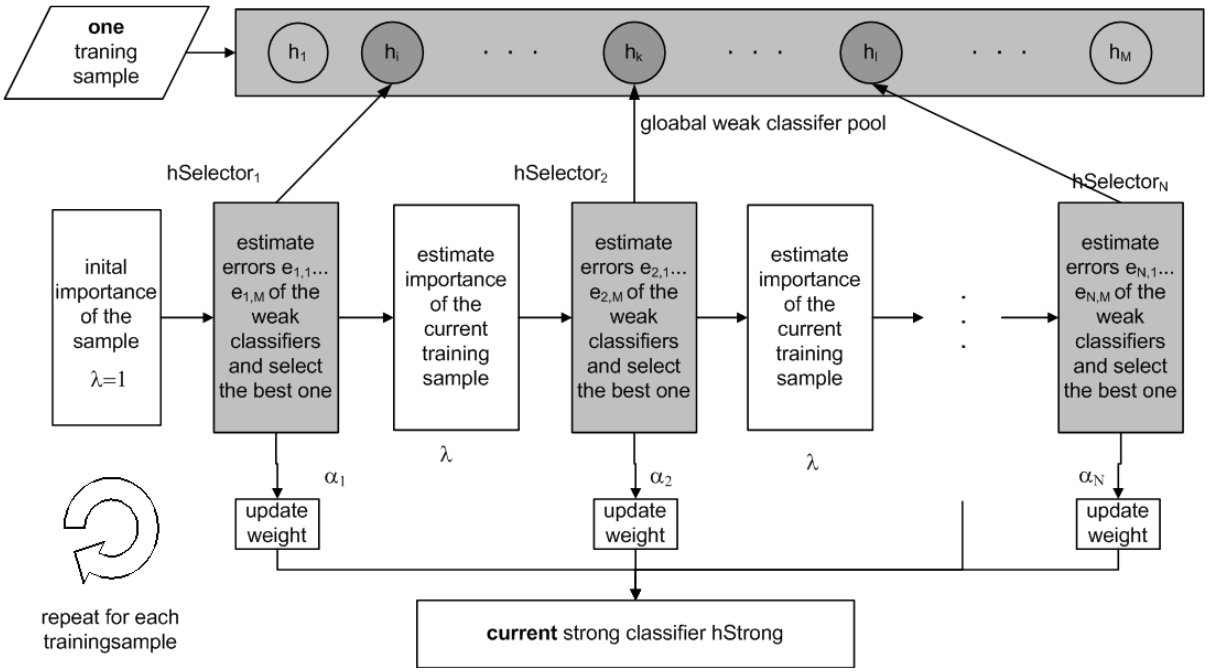


Figure 3.3 AdaBoost strong classifier calculation

Selectors are the core of on-line boosting algorithm concept and are randomly initialized, holding an individual set of weak classifiers. Weak classifiers are updated as soon as the new training sample is available for analysis. As described before, the classifier with the lowest error is chosen by a selector.

Updating the weak classifiers is said to consume the largest amount of processing time, so a global pool of weak classifiers is used instead of a dedicated pool per each selector. This allows to update all the weak classifiers in one go and move on to updating the selectors with their respective weak classifier sets, from which the best one is chosen and the weights are passed on further. Once all the selectors are updated, a strong classifier for given sample may be produced. Moreover, in the end of each iteration the weak classifier with the greatest error is removed from the pool and a new one is randomly chosen, which introduces more diversity into the selection process.

3.3.3. Feature Selection

Three different feature types are utilized in order to produce weak classifiers. Among them are Haar-like features as described by Viola and Jones (1), orientation histograms ((26-28)), simplified local binary patterns(29). It is important to stress that the features can be computed fast enough for the tracking to remain real-time by employing integral images and histograms (27).

3.4.Tracking Learning Detection (30)

In their work work on a TLD tracker, Kalal et al. (30) present a tracking methodology that is tolerant to object disappearances and appearance changes, while not requiring preliminary training in offline mode. The core idea is to combine tracker and detector neither of which are suitable for solving the long term tracking problem separately.

Authors propose the following framework to solve the long-term tracking task (Figure 3.4)

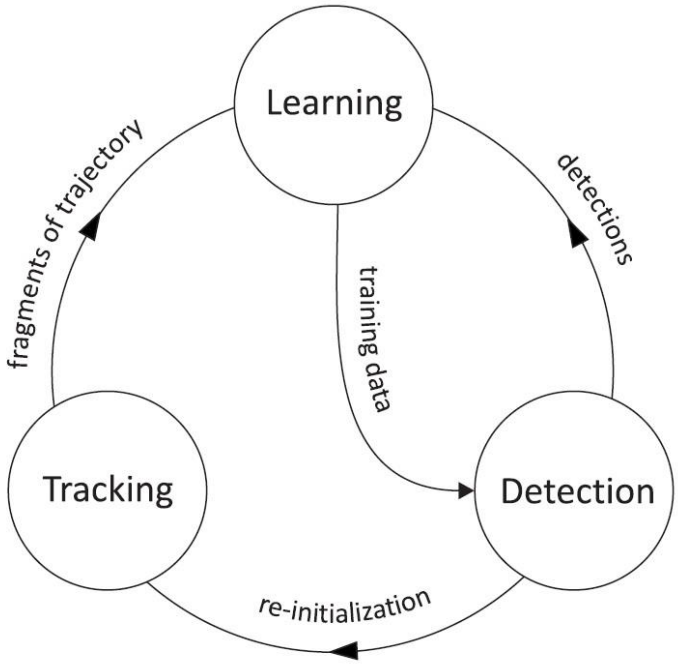


Figure 3.4 TLD information flow diagram

The **tracker** follows the object of interest from frame to frame, while the **detector** attempts to find all the previously seen appearances of the object and corrects the tracker if it deemed necessary. An array of object transforms can be used to estimate the object trajectory. The **tracker** in proposed TLD implementation is based on Median Flow (19) tracker with failure detection introduced. A pyramidal Lucas-Kanade feature tracker is employed (31). Object disappearances (when the average displacement of features is greater than 10 pixels) are handled by returning no bounding box. The tracker is tolerant to fast movements.

Object **detection** component utilizes **patches** of object and its surroundings. A set of grayscale patches that describe the object and its surroundings are generated when the initial bounding box is selected. Possible alterations of the initial object appearance are generated with respect to scale and shift with predefined steps. It is reported that around 50 000 patches are generated for an image of QVGA resolution (320x240 px) and 200 synthetic positive patches are produced as the initial learning set for a detector. Patches are scaled to 15x15 pixels no matter what their original size is. The precise number of patches is said to be proportional to the frame resolution and the object bounding box size.

Patches are an integral part of the object model employed in TLD. Every patch has several similarity measures attached to it:

- Positive nearest neighbor similarity
- Negative nearest neighbor similarity
- Positive nearest neighbor similarity with respect to 50% of earlier positive patches
- Relative similarity (higher values point out that the patch represents the object of interest)

- Conservative similarity (high value leads to a conclusion that the patch resembles object appearances seen in the first half of positive results)

A scanning window grid is generated to perform detection based on patch similarity, which is computed using integral images and takes constant time **(1)**.

If neither the tracker nor detector return a bounding box, the object is considered not present in the view. However, if one of them fails, either the tracked bounding box or a maximally confident detector patch become the new output (confidence derived from similarity measures).

The concept of **positive-negative experts (P-N learning)** is introduced which facilitates the overall self-evaluation and correction of the proposed algorithm. The **detector** is evaluated by these experts during every iteration (every frame). Positive experts retrieve missed detections while the negative one keeps track of false positives. The training set of the detector is obtained live and is constantly updated and improved by feedback from the experts. False negatives retrieved by the P-expert are labeled positive and their addition to the set increases detector's generality (it recognizes more appearances of the object), while false positives are labeled as negative by N-expert and increase detectors ability to discriminate against everything that is not an object of interest.

In Kalal's implementation **P-expert** makes use of temporal structure of input frames and works under assumption that the object follows a certain trajectory, and if a detector labeled current location as negative, while the tracker considers this to be the current object's position, a false negative example is added to detector's set with a positive label.

N-expert, on the other hand, makes use of the fact that there may only be one object appearance in any given frame. It analyzes all detector responses and the only one produced by a tracker. The most confident detector response is chosen as a reference patch. All the other patches that do not overlap with the reference one are labeled as negative.

3.5.Semi-Direct Monocular Visual Odometry (32)

Forster et al. propose a precise and robust algorithm that solves the problem of visual odometry. The method does not employ computationally costly feature extraction and works with pixel intensities. This feature of the algorithm allows it to work at higher frame-rates.

The method is proposed to be used in small aerial vehicles that are controlled remotely or move using predefined patterns (e.g. quadrocopters, drones). A single camera directed downwards is thus employed as the only source of images for the algorithm. This is done in order to decrease the overall vehicle weight, which is crucial for small flying machines.

The proposed algorithm works in two threads, simultaneously estimating the camera position and mapping the exposed area in 3D similar to SLAM algorithms. (33)

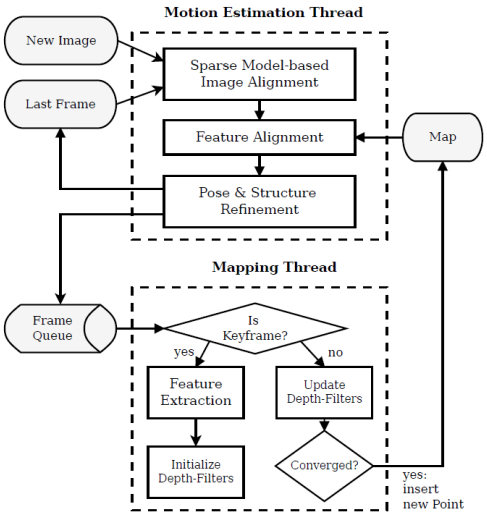


Figure 3.5 SVO parallel tracking and mapping

In a thread that estimates camera relative position, image alignment using a sparse model is performed first. This is achieved by minimizing the photometric error between the current frame and the previous one. Pixels that correspond to projected positions of 3D points (points that belong to a map created by a separate thread) are used to calculate this error. Motion estimation is then performed by minimizing the reprojection error introduced by a feature-alignment step.

The mapping thread uses a probabilistic depth-filter in order to estimate a 3D position for each detected 2D feature. Depth filters use large depth uncertainty values. More depth-filters are initialized when a feature is detected in an image region with a small amount of 3D-2D matches previously detected. One the uncertainty value of a depth filter for given feature becomes small enough, a corresponding three-dimensional point is inserted into the map, which is ready to be used by a thread that calculates camera movement.

Despite the fact that feature correspondence is used in this algorithm, it is only performed when new keyframe with new 3D points is inserted into memory, which gives a significant fps boost to the algorithm.

3.6.Preprocessing

3.6.1. Sharpen

Image sharpening is a process of image modification that results in an image that is less blurry than the original input (see Figure 3.6).

Figure 3.6 Example of image sharpening[9]

In the digital image processing context, a technique called unsharp masking is used to perform image sharpening. This concept is employed in software that is dedicated to working with digital pictures such as GIMP (34). Unsharp masking can also be applied in order to actuate edges and enhance contrast in places that fall under the filter's actions. (35)

Gaussian blur (36) is applied to the original image copy. The copy is then compared with the original and a certain threshold value is used to subtract images from each other if the difference is greater than that threshold.

This technique comes at cost of possible introduction of artifacts such as undesired edges, but this effect can be used to our benefit when the image has to be further analyzed by a computer algorithm (such as a tracker or a detector), but not by a human viewer. There is also a possibility to only apply the effect to select channels of an RGB image or to select regions of the latter.

The following three parameters are commonly used in unsharp masking for digital images (34, 35):

- **Radius**. A smaller radius results in making smaller details more visible.
- **Amount**. This parameter controls the general strength of resulting sharpening effect.
- **Threshold** specifies the minimal amount of difference between pixels when sharpening should take place. This parameter can be used to leave smooth transitions untouched, but increase other subtle details' visibility (both for human viewers and for computer vision algorithms).

[9] http://upload.wikimedia.org/wikipedia/commons/4/43/Unsharped_eye.jpg

3.6.2. Histogram Equalization

Representation of how often a value is encountered in a set is called a histogram. Histograms are traditionally visualized using rectangles, and their width represents the value classes, while their height is proportional to the number of value occurrences in given set. (37)

Histograms may be of special interest when analyzing or modifying/enhancing with digital images. An image histogram contains the number of pixels corresponding to each value of tone (intensity).Figure 3.7 shows an image on the left and its histogram on the right.(38)

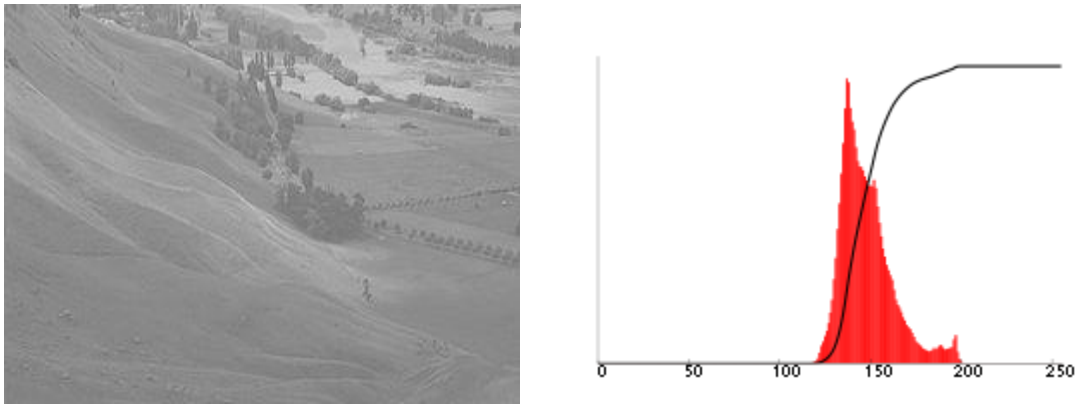


Figure 3.7 Example of image and its histogram[10][11]

In particular, it is possible to use histograms to enhance the picture's brightness and contrast by reassigning pixel intensity values. Stretching the peaks of an image histogram will increase the overall image contrast. Histogram equalization method adjusts the input image in such a way that the histogram of a resulting image becomes uniform, hence removing redundancy in dark or light tones (see Figure 3.8). (39)

10 http://upload.wikimedia.org/wikipedia/commons/thumb/0/08/Unequalized_Hawkes_Bay_NZ.jpg/450px-Unequalized_Hawkes_Bay_NZ.jpg

11 http://upload.wikimedia.org/wikipedia/commons/thumb/4/4e/Unequalized_Histogram.svg/450px-Unequalized_Histogram.svg.png

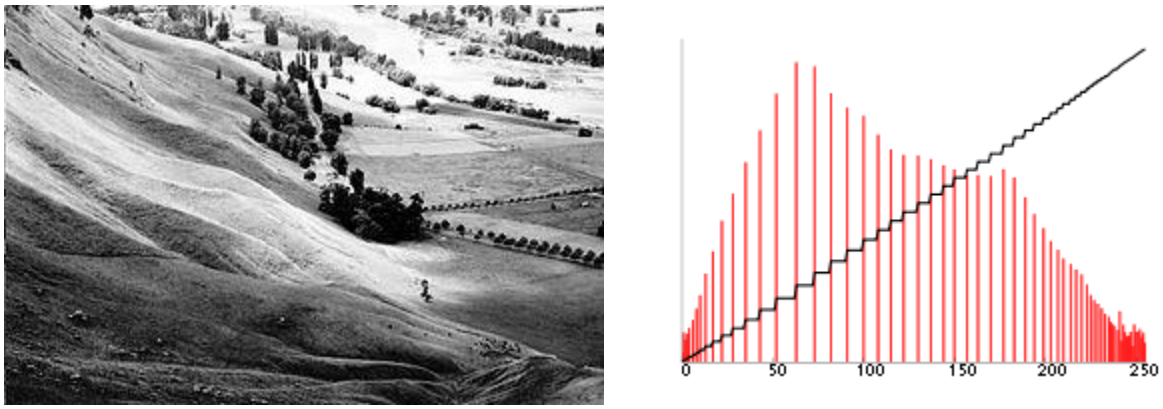


Figure 3.8 Example of an equalized histogram and a corresponding image[12][13]

OpenCV computer vision library offers an implementation of image histogram equalization. As described above, a function provided by OpenCV library alters the image so that its histogram becomes stretched and more uniform. This is achieved by remapping the initial histogram to a new one. This remapping has to be the cumulative redistribution function. Histogram $H(i)$ has a cumulative distribution $H'(i)$ defined as

$$H'(i) = \sum_{0 \le j < i} H(j)$$

$H'(i)$ has to be normalized in a way that the greatest value equals 255, which is the top limit for image intensity value. The image is then transformed using a normalized $H'(i)$ lookup table. (40, 41)

3.6.3. Specular highlight removal

Specular reflections in digital images are a result of light reflection by the observed surfaces. As the word specular suggests, light rays are reflected in a single direction as if the surface was a mirror. Reflections like that are typically seen on glossy surfaces and surfaces of objects with wet nature (see Figure 3.9, Figure 3.9).

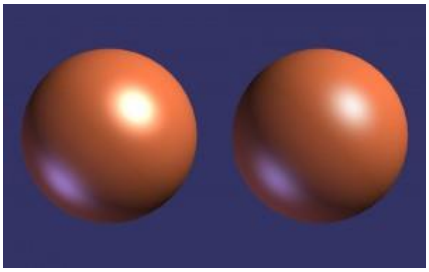


Figure 3.9 Synthetic specular highlight[14]

Figure 3.10 Specular highlights in an endoscopic image[15]

12 http://upload.wikimedia.org/wikipedia/commons/thumb/b/bd/Equalized_Hawkes_Bay_NZ.jpg/450px-Equalized_Hawkes_Bay_NZ.jpg

13 http://upload.wikimedia.org/wikipedia/commons/thumb/3/34/Equalized_Histogram.svg/450px-Equalized_Histogram.svg.png

14 http://www.reallyslick.com/blog/wp-content/uploads/2011/08/quality_specular-300x187.jpg

15 http://www.doc.ic.ac.uk/~benlo/videos/Figure%206.10b%20SFS%20without%20specular%20removal.jpg

Specular highlights often become an obstacle for human eyes while inspecting images or video sequences for prolonged periods of time. Highlights also capable of somehow interrupting object tracking/detection algorithms and other algorithms that depend on the integral object surface visual representation. Most algorithms, however, remove outliers such as irregular specular highlights. (42, 43)

4. Related Work

The following chapter describes current achievements and experiences with telestration for surgical telementoring, camera motion estimation (visual odometry) in minimally invasive surgery (MIS), as well as in object tracking in MIS.

4.1. Search Methods

Google Scholar[16] has been used as a primary source of literature. The Table 1 describes the keywords used and the number of findings. The number of papers "found" denotes the number of papers that contain the necessary keywords and are included in the initial (bigger) set of literature. The number of "relevant" papers is the number of papers that were considered relevant after a more detailed inspection from the first initial set.

Keywords	Found	Relevant
camera+motion+estimation+surgery	26	15
object+tracking+surgery	11	4
telestration+surgery	26	10
visual+odometry+surgery	8	6

Table 1Results of literature search

4.2. Camera and Tissue Motion Estimation and Visual Odometry in MIS

Given that a digital laparoscopic/endoscopic camera is an inevitable necessity in minimally invasive surgical procedures (both in traditional laparoscopic surgeries and those facilitate by robotic systems such as ZEUS and Da Vinci), researchers have long since turned their eyes to benefit from the most abundant source of data provided by a camera – a stream of digital images from inside the patients' body.

Earlier approaches to obtain information necessary for camera motion estimation heavily relied on sensors and techniques other than image analysis of a laparoscopic/endoscopic camera feed (44) (45, 46):

- Artificial visual landmarks for image-processing approaches
    - LED lights
    - Infra-red stickers
- Sensors
    - IR sensors
    - Ultrasonic sensors
    - Mechanical accelerometers

With theoretical framework behind object tracking and camera motion estimation techniques described in 1988 (47), it was not at the time possible to provide robust real-time implementations of such techniques. Early works were focusing 2D beating heart frequency tracking and instrument tracking (44) (46, 48). Heart tracking techniques are still being perfected and are mostly applied in order to counter cardiac shaking in video streams and in instrument positioning. Many works that feature visual tracking (with or

[16] https://scholar.google.no/

without artificial landmarks or external sensors) focus instrument tracking and robotic camera holder as a replacement to human assistant who manually holds the camera. (49)

In time, visual techniques have become more efficient and complex, and have been developed as an alternative (or in addition) to tracking methods such as ultrasonic or infrared scanning, or introducing infra-red or colored markers. The growth of computational power of computing devices also greatly contributed to such turn of events. There are several reasons to avoid marker (artificial landmark) introduction and ultrasonic/infrared techniques (50, 51):

- Possible personal biological intolerance to the markers being introduced, which may result in serious problems for the patient.
- Markers have to be glued or stitched to the organ surface which may take extensive amounts of time and result in poorer clinical outcome.
- It is not always possible or safe to introduce markers because of the space being limited inside the cavity.
- Portions of organs are overlapped by markers and are no longer visible.
- More equipment has to be introduced into the already crowded operating theatre.
- Infrared scene lighting (with or without infrared markers introduced) introduces additional heat to the cavity, which is not always tolerable in the operation's context.

The techniques listed above often facilitate camera or tissue motion estimation with a far greater accuracy and speed than the techniques that rely solely on visual input, but they come at a certain price as just described (52) The following core advantages of purely visual motion estimation have been identified:

- No additional equipment needs to be introduced to the operating theater given that digital video stream is provided.
- No modifications required for visual techniques to work with existing MIS robotic hardware.

Traditional uses of such techniques span across a set of surgical procedures that involve minimally invasive techniques and are image-guided such as bronchoscopy, transnasal skull-base neurosurgery, rhinoscopy and laparoscopic techniques, including those assisted by robotic systems.

Grayscale 8 bit image feed is often used as a source of data. However, sometimes all 3 channels are utilized in order to perform virtual field of view extension. Some approaches such as the one used for 2D instrument tracking and proposed by Bourger et al. (53) make use of alternative color spaces such as HSV.

Many approaches are based on simultaneous localization and mapping (SLAM) method that is widely used in unmanned moving vehicles and use Kalman Filter or its extended version to facilitate the creation of a 3D point map. (54, 55)

Heart tracking

Many research groups use motion estimation in order to perform virtual stabilization of a camera and surgical tools. This kind of stabilization is proven to be of great benefit for beating heart surgeries. If a virtual camera stabilization problem is solved, it will allow to solve the immediately adjacent problem of mechanical robotic instrument stabilization by moving the instruments intact with the patient's heartbeat, making a beating heart surgery not a lot more difficult than a surgery on a heart that is almost still. Otherwise, a surgeon himself is bound to adapt his movements to the rhythm of a heart beating, while that focus and energy could be targeted at other actions.

Such interest and abundance in positive results on virtual camera stabilization in beating heart surgeries are partly justified by the fact that heart is a much more rigid, less deformable and better defined organ as opposed to other parts of human body where endoscopic or laparoscopic surgeries take place (e.g. epigastrum), which makes it easier to track the surface of a beating heart using various methodologies. The fact that robotic systems such as Da Vinci and Zeus do not possess an integrated organ/tissue motion tracking capability to solve the problem, also fuelled the interest to this kind of research. These robotic systems, however, provide precise information on camera and tool motion because these motions are completely controlled by the surgeon via the system's interface or semi-automatically, which leaves the camera motion estimation problem solved in case of robotic systems. (56, 57) Many solutions that are being developed with further integration with robotics in mind are developed for **stereo** endoscopic/laparoscopic cameras. (57)

Bader et al. (57) proposed to describe a beating heart motion estimation as a partial differential equation that has a periodic solution (because the hearts beats repeatedly). Their system is developed using C++ language and a COM interface to MATLAB computer vision and differential equation libraries. Usually, well known portions of heart are tracked, which results in greater robustness of such solutions when that portion is clearly observable.

Sauvée et al. (51) employed a texture tracking approach for solving a beating heart problem as an alternative to tracking artificial landmarks with known geometrical shapes. Texture features and sum of square distances (SSD) from frame to frame were used in order to calculate the organ movement.

Stoyanov et al. (58) propose a solution for soft tissue motion tracking based on two combined feature detectors that help reconstruct the 3D structure. A pre-calibrated stereo laparoscopic camera was used in order to obtain images. The method uses Lucas-Kanade feature tracker in order to follow the features throughout a consequent set of stereo images. (9) (59) The authors mention several less computationally complex alternatives to their method including the use of mechanical or optical accelerometers (60) and **monocular** motion recovery from 2D images (61), the latter two, however, being used exclusively for beating heart motion tracking.

Soft-tissue tracking and visual odometry

As opposed to heart motion tracking, soft tissues are not so easy to track in some scenarios and possess far greater flexibility and in most cases cannot be considered a rigid body. Natural landmarks (features and sets of features) in soft-tissue tracking might

be unexpectedly occluded or change their appearance because of tissue folding and lighting changes. (62, 63)

Applications of soft tissue tracking are also slightly different from those possible in heart surgeries. In this case, tracking may be used to facilitate surgical guidance intraoperatively, impose active restraints on robotic instruments or compensate motion. (58)

In the foldable non-rigid tissue tracking field, a stereo video feed from a camera located on the tip of an endoscope/laparoscope is often used because of the ever increasing popularity of robotic MIS systems. Stereo image gives the researchers an opportunity to benefit from the use of epipolar geometry techniques and in essence obtain more visual data from one frame with less camera movement involved.

A solution based on simultaneous localization and mapping **(SLAM)** has been proposed by Mountney et al. (64). Their method aims to build a persistent three-dimensional map of the cavity while recovering camera position at the same time. Accordingly, Kalman filter (65) is used in order to process a map of 3D points as in many other SLAM approaches. Stereoscope position is estimated based on a 6 degrees of freedom (DOF) motion model. The features used to create a map are Shi-Tomasi features (22), tracked by a Lucas-Kanade feature tracker. The stereo laparoscope has to be calibrated in order for this method to work.

A method for tissue deformation tracking and learning the deformation online has been proposed by the same research group. (66, 67)

Mountney et al. (68) later proposed an extension of the latter approach for stereoscopic cameras that facilitates the dynamic expansion of viewport, while creating a 3D map of the cavity in a SLAM fashion. This method is reported to work in real time. In 2010, a new augmentation to this method was proposed by the same research group, now taking periodic movements such as respiratory or cardiac shaking into account. The system learns such movements and compensates movements within its SLAM procedure. The method is accordingly called MC-SLAM (motion-compensated SLAM). Extended Kalman Filter is being used in this case. (69)

A somehow similar approach to camera motion estimation is proposed by Wang et al. (70) in order to track the position of an endoscopic camera inside the patient's brain. The brain tissue is reported to be abundant in specularities, being poorly texturized. Moreover, the nature of endoscopic lighting and camera movements introduce extreme scene lighting and motion blur. The researchers propose a method that uses **SIFT** feature detector (71). However, an SVD (72) matching algorithm was used instead of the native SIFT matching function, because SVD proved to yield more correct matches. Minimization of reprojection error is used in order to create a 3D point cloud that is then iteratively updated. In this case, RANSAC has been reported to be insufficient to filter outliers since their amount was excessively big. Adaptive Scale Kernel Consensus (ASKC) is proposed instead, which is capable of estimating the accepted amount of outliers dynamically, making it possible to tolerate small inlier amounts and continue tracking.

A technique for retrieving structure from motion in a similar manner was proposed by Hu et al. (73) in 2012, using the LK feature tracker and a special method of outlier removal (instead of traditional RANSAC), called trifocal tensor.

Another approach that employs reprojection error minimization has been described by Noonan et al. (63). Their method allowed to recover 6DOF (velocity and angular velocity) camera movement, while building a 3D cavity map using the SLAM methodology. RANSAC outlier filter was deemed sufficient for the task. However, an extended Kalman Filter (EKF) variation was employed in order to boost SLAM robustness. The researchers stress that small distance between stereo cameras is a limitation for SLAM techniques, as well as the fact that features may be extremely sparse in some cases and be occluded by highlights.

There are several approaches such as the one proposed by Mirota et al. (74) that can be dramatically speeded up (optimized by parallelization) using graphics processing unit (Nvidia CUDA technology is commonly used for such purposes). A method of endoscopic camera motion estimation from a video feed uses preoperative CT scans as a source of data as well. Matlab implementation of SIFT feature detector is used. The researchers claim to achieve a 300 times speed boost by implementing a GPU optimization.

Some researchers state that SIFT/FAST features do not provide enough speed for the structure from motion SLAM algorithms to be robust and real-time at the same moment (75). Instead, a STAR (76) feature detector has been used together with binary robust elementary features descriptors (BRIEF)(77). This combination of feature detector and descriptor yielded better matching percentage than a conjunction of SURF and SIFT respectively. Speed performance has also been reported to be greater, while offering 3D reconstruction of tissue deformations in real time.

A method proposed by Lourenco et al. (78) makes use of structure from motion techniques (3D map from 2D images) for stereoscopic images, while at the same time being tolerant to a certain degree of non-rigid transformations. The scene is segmented under the assumption that it is rigid, but if the non-rigid transformations occur within separate segments, the algorithm still yields correct results. However, this robust structure from motion implementation does cannot be performed in real-time.

Grasa et al. (79) propose their modification of SLAM technique tailored for handheld **monocular** endoscopes. An extended Kalman Filter (EKF) is used together with a joint compatibility branch and bound SLAM (JCCB SLAM). FAST features and simple patch correlation were used. This combination, however, suffers dramatic performance loss if there is even one mismatch, but outlier filtering solves this issue.

Chang et al. (80) propose yet another variation of stereo visual odometry. 6 degree of freedom position transformations are provided by the algorithm. A technique called quadrifocal relationship is used in conjunction with traditional photometric error minimization in order to estimate the features' position in space. The quadrifocal technique makes use of two consequent stereo images in the same way most monocular methods use two consequent "mono" images, in effect spanning the feature position estimation across 4 frames. The fact that the algorithm is optimized for GPU allows it to run in real-time using a dense set of features.

4.3. Telestration in Telementoring

Telestration as a tool has initially been in much use in applications such as sports and weather forecasting (15). Moore et al. (81) were one of the first research groups to have reported their experience in laparoscopic surgical telementoring while empowering the mentor with a telestration functionality. Ever since the early days of telestration in telementoring, its significance has been acknowledged and it has been reported as a feature that greatly enhances communication quality between the mentor and the mentee. (82)

However, the research groups that pioneered in surgical telestration in telementoring reported the fact that it was hard to use telestration in real time because of network bandwidth that was not enough at that time (year 1997) to support a live audio, video and telestration streams at the same time. (82)

Despite the fact that telestration was deemed useful, in the early days of telementoring the telestration tool has not received enough attention, being often omitted in favor of system simplicity or because of bandwidth limitations.

As more research groups performed and evaluated surgical telementoring, its educational and clinical benefits became apparent(83, 84), leaving room to look deeper into its integral part - telestration.

The capability to telestrate, provided by a system used for telementoring has been proven to dramatically increase teaching capabilities of a remote surgeon, making it much easier to reach agreement then while using a combination of audio and video communication (traditional video-conference) (85). Budrionis et al. (86) have conducted an extensive literature review on telestration in surgical telementoring which revealed several advantages of telestration apart from those mentioned above. The fact that communication is enhanced and more informative allows the surgeons conduct an operation in significantly smaller amount of time, which in turn makes the surgeons (both mentor and mentee) more available, and results in improved clinical outcome (86). The fact that telestration is proven to increase the quality of surgical education makes it an invaluable tool to counter the lack of surgical residents that are trained to operate on certain rare conditions that can be taught remotely (15, 85).

However, telestration is often not provided together with commercial telementoring setups. It is known, though, that robotic Da Vinci system offers a possibility to telestrate in two dimensions on its master console. (87)

Budrionis et al. propose the use of WebRTC[17] to be used in order to transmit the video stream and telestrations over the network right into a browser, making the system usable on virtually any device that is capable of running a web-browser and possesses a network connection with sufficient bandwidth(15, 88). This being said, it might be possible to use telestration while performing a telementoring session from a mobile device such as a smartphone or a tablet (89). Using small portable devices that provide natural drawing interfaces such as tablets makes it possible to effortlessly use them for on-site telestrating while being scrubbed in or remotely (15). Using WebRTC and portable devices allows

[17] http://www.webrtc.org/

telementoring to be used as a service (i.e. on demand), making one mentor virtually available anywhere and anytime, providing him with core tools for sharing his knowledge and guiding the less experienced surgeons (88).

Certain limitations are present in use of telestrations. The quality of telestrations vastly depends on network bandwidth and delays. This issue is far less pressing nowadays, but it is not completely obsolete. Santomauro et al. (87) report delays up to 500ms to be tolerable in telementoring, which is possibly also true for telestrations, but has not yet been assessed.

Another problem that is relevant for any telestrating software is related to camera movements that are an integral part of any surgical operation, especially if a mentee is relatively novice in given procedure execution. Any camera movement basically renders telestrations useless because they become displaced and lose their logical meaning, because they are supposed to visually reside over a certain anatomical landmark. Several possible countermeasures were proposed by Budrionis et al. (89), including only allowing telestrations on still images, which eliminates the problem as such at cost of temporary loss of interactivity. One other possible approach is using 3D models retrieved from CT or MRI body scans. It might also be possible to make use of computer vision techniques. The latter two are, however, reported to be problematic to be implemented using mobile platforms.

5. Methods and Materials

5.1.Research Paradigm and Tools

This work has been performed using an engineering approach described by Comer et al. (90) First, the requirements and specifications are defined, then system design is performed, then the system is implemented and is afterwards finally tested.

5.2.Materials

In order to implement the system, a combination of software tools and two different operating systems have been used. Development took place on Windows 8 and Ubuntu 14.04 operating systems.

The following tools were used on a Windows operating system:

- VS Express 2013 for Desktop[18], an IDE provided by Microsoft that includes a .NET and WPF frameworks, used for developing the system's user interface
- C# language has been used to develop the GUI module of the system

The following tools were used on a Ubuntu system:

- Bash[19] scripting command line language has been used to facilitate utility scripting while processing the digital image sets
- Python[20] scripting programming language has been used to facilitate image processing and object tracking for test purposes (using OpenCV for Python)
- Matplotlib[21] Python library has been used for plotting data sequences
- C++ language has been used as a primary interface language for working OpenCV and developing the stabilizer module of the system

The following tools were used on both operating systems:

- OpenCV[22], an open source library that contains a vast collection of image processing tools and algorithms, is released for Windows and Ubuntu and provides interfaces for all the languages used throughout the project
- FFmpeg [23]codec collection has been used to work with video and image sequences

Among the hardware used is a stationary PC (Intel Core i7 4500U CPU, 8Gb RAM, Windows 8) and a laptop (Intel Core i5 3450s, 8Gb RAM, Ubuntu 14.04)

5.3.Data Collection and Experiment Methods

5.3.1. Literature review & Related Work

A literature review has been performed in order to formulate functional and non-functional requirements for the system, as well as to identify the current trends in

[18] https://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx
[19] http://www.gnu.org/software/bash/
[20] https://www.python.org/
[21] http://matplotlib.org/
[22] http://opencv.org/
[23] https://www.ffmpeg.org/

telestration for surgical telementoring and surgical image processing in context of camera motion estimation and object tracking (See Table 1).

5.3.2. Meetings with Expert Surgeons

Several meetings with expert surgeons at UNN (Hiten Patel, Rolv-Ole Lindestmo and Etai Bogen) have been held in order to formulate the requirements for the system and discuss the possible approaches to facilitate the stabilized telestration.

5.3.3. Application Testing

The application has been iteratively tested along the development path, testing the modules separately with surgical residents at UNN (Hiten Patel, Etai Bogen). The application's telestrating capabilities have been tested using input from a web-camera connected via USB, and using pre-recorded video files.

5.3.4. Object Tracker Benchmark

The object tracking algorithms have been collected from an experimental OpenCV module called *contrib*, which provides open source C++ implementations of the Median Flow tracker, the Ada Boost tracker, and the MIL tracker. The open source C++ implementation of TLD (the primary subject tracker of this work) algorithm has been retrieved from a public repository of Georg Nebehay, AIT[24]. The same repository contains the Python implementation of a CMT tracker which has been developed as an alternative or competitor of TLD. Total of 5 object trackers are featured in this work.

A data set of total 6 video sequences has been collected using chunks of laparoscopic videos in public access (www.youtube.com) and the video records provided by Hiten Patel and Etai Bogen. The videos were then enhanced (histogram equalization, sharpening) in different ways and a resulting set of 24 videos has been processed by all the object tracking algorithms featured in this work.

5.4.Evaluation Methods

The application evaluation has been performed by expert surgeons from UNN (Hiten Patel, Etai Bogen). Evaluation of subject trackers has been performed in a semi-automated way by processing the data set that consists of surgical videos with subject trackers and recording the results.

The results were then plotted using Matplotlib and quantitative and qualitative analysis has been performed.

5.5.Critique of Methods Used

The application development should be performed with larger user audience, who could provide feedback on system's usability and desired/undesired features. However, it might not be easy to find large numbers of surgeons.

Getting a larger test group that would include the beginner surgeons, and a systematic questionnaire could provide more thorough information about the kind of telestration tool desired by a broader range of surgeons.

[24] https://github.com/gnebehay

Some SLAM (simultaneous location and mapping) and object tracking algorithms with their source code publicly available could not be tested because of the software constraints or software/hardware portability issues. Some algorithms require pre-calibrated or stereoscopic cameras to be used, while the goal of this study is to try and employ the simpler and less expensive setup for telementoring and stabilized telestration.

Computer vision is a rapidly developing field of study and many computer vision algorithms suitable for the task might appear in the near future or might not have been found by author. This, however, does not stop us from changing the tracking element of the system which can be done easily due to the modular architecture.

5.6.Summary

The following methods were used in this work:

- Data collection
    - Literature review
    - Data set collection
    - Meetings and discussions with expert surgeons
- Engineering approach
    - Requirement specification
    - System design
    - System implementation
- Application testing
    - Peer review
    - In-depth tracker testing
    - Tracker testing results interpretation

The methods used are described in depth in the later chapters.

6. Requirements specification

In this chapter, the definition of requirement specifications if described. A proposed scenario, use cases and the derived requirements are presented.

UML use case diagram was created according to the scenario. Volere Requirements Specification (91) was used as a basis for systematic requirements definition.

6.1.Requirements Source

Requirements for the system are specified in a manner proposed by Robertson and Robertson (91), being divided into functional requirements that specify the actions that the system enables user to perform, while the non-functional ones identify the properties of system behavior.

Requirements are in their mass derived from review of relevant literature and related work that has been previously performed and described by other researches in the same field (object tracking/image analysis in minimally invasive surgeries (MIS), image stabilization and telestration in MIS).

Expert surgeons from University Hospital of Northern Norway (UNN) have also made an impact on the definition of system requirements.

State-of-the-art systems inspired some of the requirements

The way we want to test it later at UNN is also behind the requirements (surgeons are the source) “experts’ suggestions”.

6.2.Requirements

6.2.1. Scenarios and Required System Behavior

Scenario.

A minimally invasive surgical procedure on a certain patient has taken place at some time in the past. A video footage of the whole operation has been recorded. The video itself contains several segments of educational value that an expert mentoring surgeon would like to demonstrate to his less experienced colleagues who are in turn eager to learn.

The mentoring surgeon needs a telestration tool in order to be able to draw lines or shapes with arbitrary properties on top of a moving video footage. However, small respiratory/cardiac and larger camera movements sometimes render the drawings useless because the tissue and the camera are displaced relatively to each other in relation to their position when a drawing has been created.

Behavior.

The system should provide the functionality to process and transmit the video sequence of interest, as well as provide the functionality to draw annotations on top of a video feed and handle camera movements when such movements occur in order to keep the drawings close to their initial logical position within a given video.

6.2.2. Use Cases

The following use case diagram (Figure 6.1) depicts how a mentoring surgeon could use the system and how a system would behave.

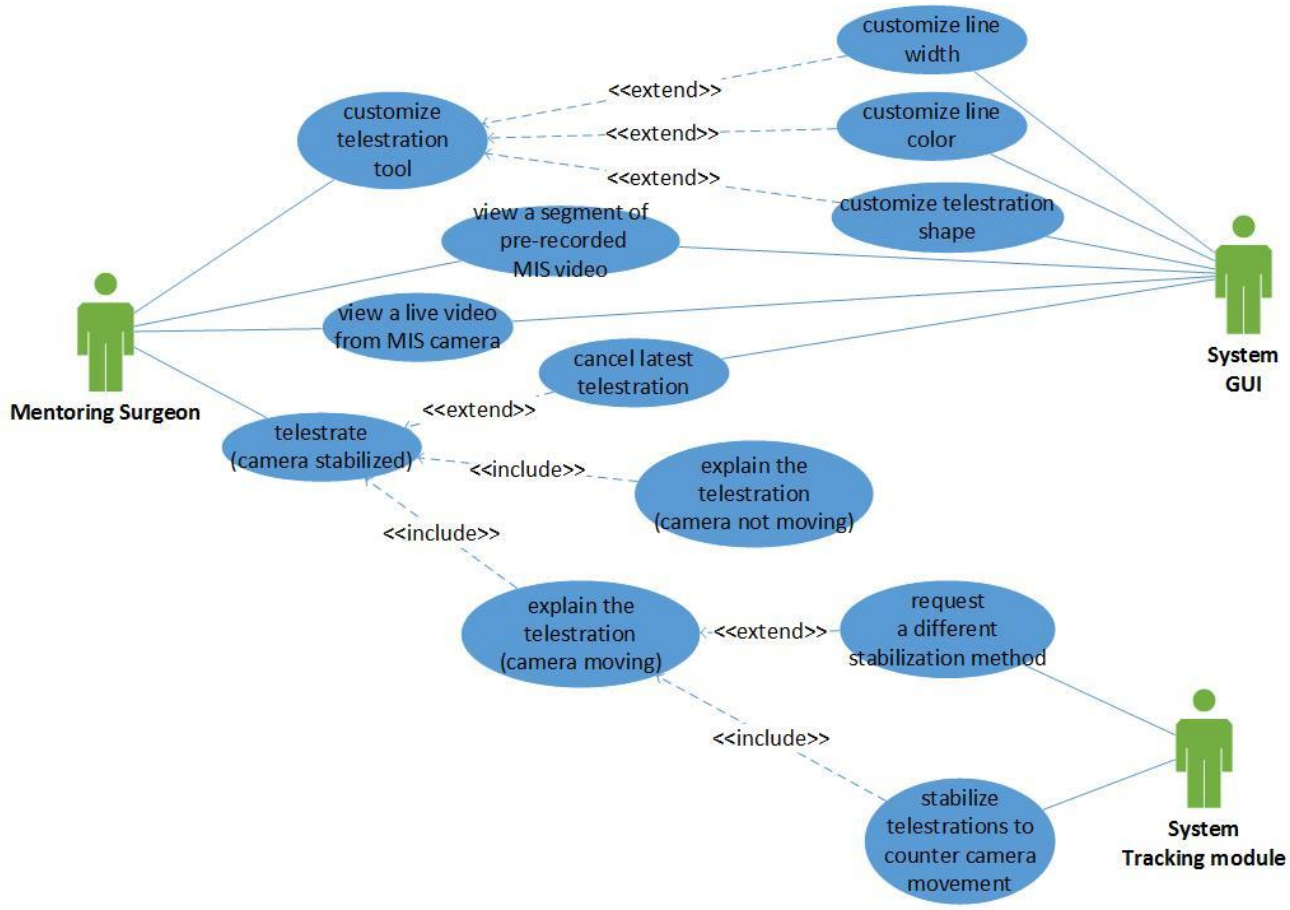


Figure 6.1UML use case diagram

The "extending" nodes follow their parent use cases, but are not obligatory to follow them, while the "included" ones are bound to happen after their parent use cases take place.

A more detailed description of use cases follows.

Use case 1: View a segment of pre-recorded MIS video

A mentor wants to see a surgical video that has been recorded prior to the mentoring session.

Use case 2: View a live video from MIS camera

A mentor wants to see live video from a MIS camera that is inside a cavity (it can be a dry lab, or a simulation, or a training environment)

Use case 3: Telestrate (camera not moving)

A mentor wants to telestrate over a video that is being displayed, while the camera is not moving. Telestrations are not usually made over a moving picture.

Use case 4: Cancel latest telestration

A mentor wants to cancel the latest telestration either because it has been drawn incorrectly or because it is no longer required.

Use case 5: Explain the telestration (camera stabilized)

A mentor wants to take his time and explain the meaning behind a recently drawn telestration. The camera is not moving or is moving insignificantly due to cardiac/respiratory movements.

Use case 6: Explain the telestration (camera moving)

A mentor wants to take his time and explain the meaning behind a recently drawn telestration. The camera is moving significantly either due to cardiac/respiratory movements or because it is otherwise required to move it by the flow of operation.

Use case 7: Stabilize telestrations to counter camera movement

The system stabilizes the telestrations created so far by using the information provided.

Use case 8: Request a different stabilization method

A mentor is not happy with current telestration stabilization method and/or wants to try a different one.

Use case 9: Customize telestration tool

A mentor wants to change the visual properties of telestrations that will be drawn.

Use case 10: Customize line width

A mentor wants to change line width of the telestrations that will be drawn.

Use case 11: Customize line color

A mentor wants to change the color of telestrations that will be drawn.

Use case 12: Customize telestration shape

A mentor wants to change the shapes that will be drawn by a telestration tool.

6.2.3. Functional Requirements

Functional requirements are defined based on the scenario described above, an UML presented, and the feedback from expert surgeons at UNN. Concerns about the system's future development also made an impact on the list of requirements. Functional requirements will be numbered **FREQ#<number>** and will be referred to using such enumeration later on.

FREQ#1

Description:

The system will provide a GUI for viewing videos from laparoscopic/endoscopic cameras and pre-recorder MIS videos.

Use case(s) related:

1, 2

Rationale:

A GUI for viewing videos either live or pre-recorded is a basic functionality that is underlying the possibility to perform mentoring and telestrate.

Source: author

FREQ#2

Description:

The system will provide a way to switch between live and pre-recorded video input modes.

Use case(s) related:

1, 2

Rationale:

There has to be a way to switch the input modes between sessions in order to use it with pre-recorded or live surgical videos. This will also ensure that it is possible to test the system with staged and non-staged videos.

Source: author

FREQ#3

Description:

The system will provide a means of creating annotations on top of a surgical video being displayed.

Use case(s) related:

3

Rationale:

It is necessary to provide a way of drawing annotations for the mentor to be able to actually telestrate while performing a mentoring session.

Source: author, expert sugeons

FREQ#4

Description:

The system will provide a means of removing the last annotation created.

Use case(s) related:

4

Rationale:

This is a direct solution to the use case 4 problem.

Source: author, expert surgeons

FREQ#5

Description:

The system will provide a built-in telestration stabilization method in case of camera movement.

Use case(s) related:

6, 7

Rationale:

It is necessary to cancel out camera movement for telestrations in order for them to remain informative after camera displacement.

Source: author

FREQ#6

Description:

The system will provide a means of switching the method of telestration stabilization while the system is offline.

Use case(s) related:

6, 7, 8

Rationale:

Methods of stabilization may become obsolete or better alternatives can be found, which makes it necessary to be able to switch the stabilization method without re-building the system.

Source: author

FREQ#7

Description:

The system will provide a GUI to customize telestration line width.

Use case(s) related:

9, 10

Rationale:

This is a solution for use cases 9, 10.

Source: author

FREQ#8

Description:

The system will provide a GUI to customize telestration line color.

Use case(s) related:

9, 11

Rationale:

This is a solution for use cases 9, 11.

Source: author

FREQ#9

Description:

The system will provide a GUI to customize telestration shape.

Use case(s) related:

9, 12

Rationale:

This is a solution for use cases 9, 12.

Source: author

6.2.4. Non-functional Requirements

Non-functional requirements are partly derived from basic usability concerns. Future work on the system is also among the factors that influenced non-functional requirements. Non-functional requirements will be numbered using **NFREQ#<number>** notation.

NFREQ#1

Description:

The system's GUI should run natively on Windows OS 7 and higher.

Rationale:

With future testing at UNN in mind, the GUI should be able to run on Windows machines because of the hospital's IT policies.

Source: author, UNN IT stuff

NFREQ#2

Description:

The video should be displayed with delays less than 500 milliseconds.

Rationale:

It has been reported that delays more than 500 milliseconds negatively affect the mentoring session quality.

Source: author, expert surgeons

7. System Design

This chapter describes the proposed system design that would conform to functional and non-functional requirements and be fitting for further testing and development at UiT/UNN hospital.

7.1. High-level System Design

The following diagram (Figure 7.1) depicts a high level system design.

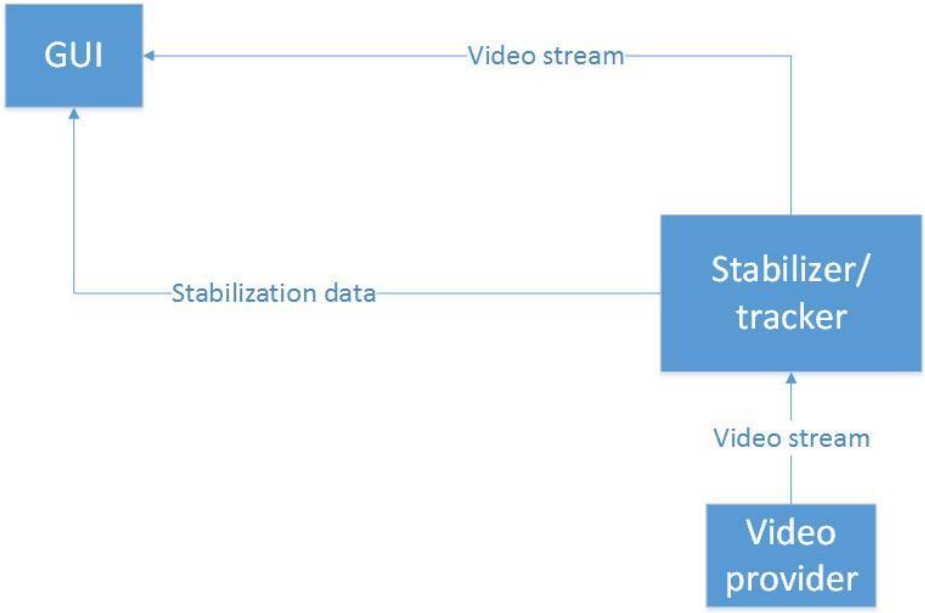


Figure 7.1High level system design diagram

The system is logically split into two modules, which makes it possible to alter one module without affecting the other. This way a stabilizer can be switched without disassembling the whole system and making excessive changes as long as the tracker conforms to the defined message format (both video and stabilization data)(**FREQ#6**).

A video provider is a source of video that can be hooked up to the stabilizer. In the same way, a video provider can be any camera connected to the system or any file that resides in the system's long-term memory. It could also be a video stream from a remote host (**FREQ#2**).

After being retrieved from the video provider and processed by the stabilizer, a video stream is then transmitted to the GUI module and is shown to the user (**FREQ#1**). Stabilization data (i.e. estimated camera motion) is transmitted in parallel with the video, but in a separate data stream (**FREQ#5**). This is necessary to make it simpler to make changes to respective functionality and make the system more flexible in general.

7.2. GUI Module Design

The GUI module will be responsible for providing an interface for the user to see the system's output and to interact with the system by telestrating (Figure 7.2).

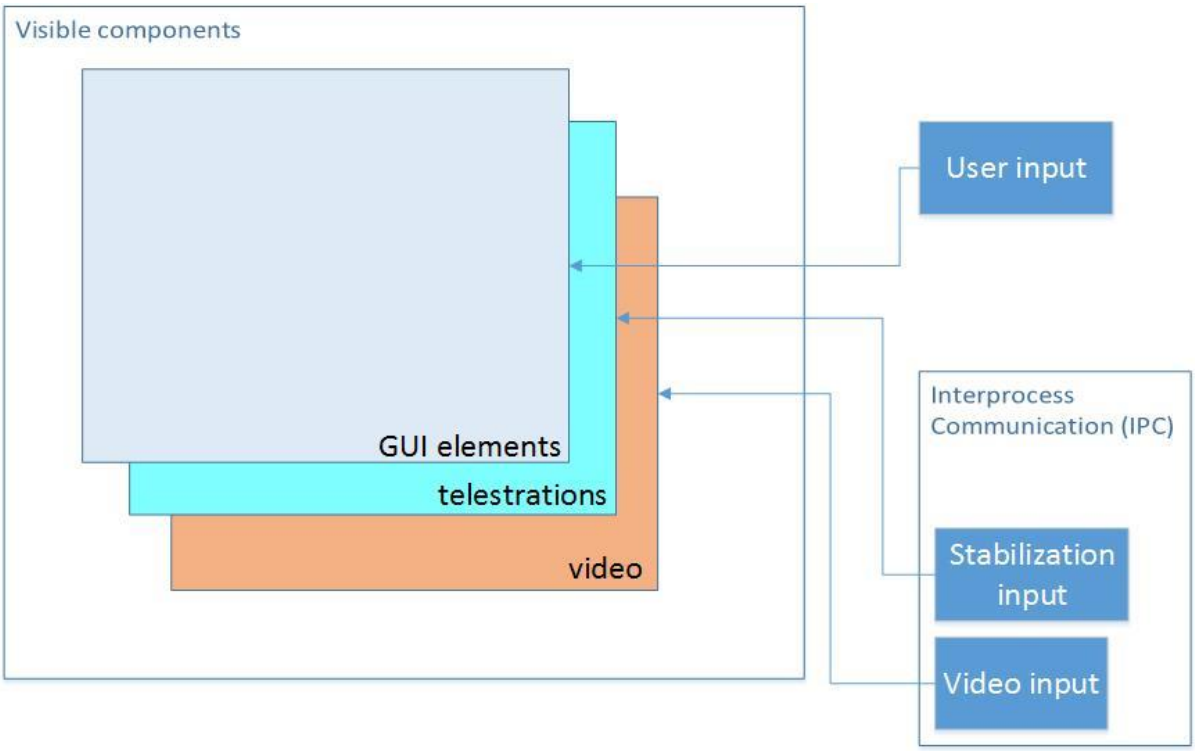


Figure 7.2 GUI module design diagram

The user input is a component provided by operating system and standard input devices (e.g. mouse, keyboard), or any means of input that can be mapped to the standard input methods.

The GUI module will possess two concurrent methods of interprocess communication that will run in parallel. Video input will provide a sequence of images to be displayed on the **video** layer which will be the lowest layer in the visual hierarchy.

Stabilization input will provide information about camera movement, which will be applied to all the telestrations drawn so far, moving them accordingly (**FREQ#5**). The telestrations will be displayed on top of a **video** layer.

The actual GUI elements will be displayed at the very top of visual hierarchy in order to ensure their visibility for the user at all times. User input such as drag & drop with the mouse left button will be interpreted into telestrations that will be placed on **telestrations** layer (**FREQ#3**). The GUI element positioning is displayed on Figure 7.3.

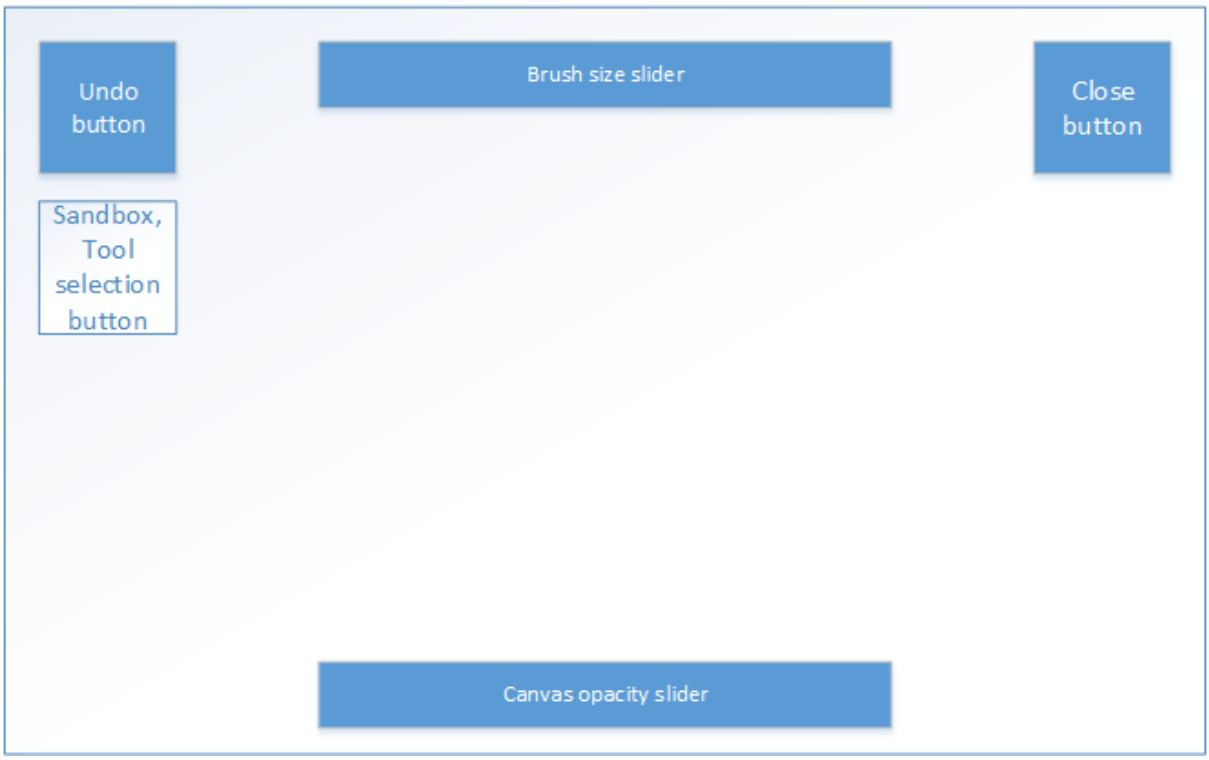


Figure 7.3 UI elements positioning

The undo button is to provide a solution for **FREQ#4**. Once this button is hit, the latest telestration will be removed from the **telestrations** layer.

The sandbox will display the current telestration shape, which is a simple freehand line drawing by default. The same sandbox will be a button (left click) to choose a telestration shape (**FREQ#9**), and a button to change the telestration color (right click) (**FREQ#8**). This way, less screen space will be occupied by the UI elements and a greater portion of underlying surgical video will be seen.

The brush size slider is to provide a functionality to alter line width in a user-friendly way that does not require any numerical keyboard input and can be controlled with simple mouse drag & drop gestures (**FREQ#7**).

The canvas opacity slider is introduced to provide a way to temporarily hide the telestrations or make them transparent to a certain degree (from 0% to 100 %) if the mentor thinks it is necessary to see them through.

A close button is there to give the user the opportunity to stop the system.

7.3. Stabilizer Module Design

The stabilizer module is going to perform most computationally heavy tasks in the system. The overall stabilizer design is depicted in Figure 7.4.

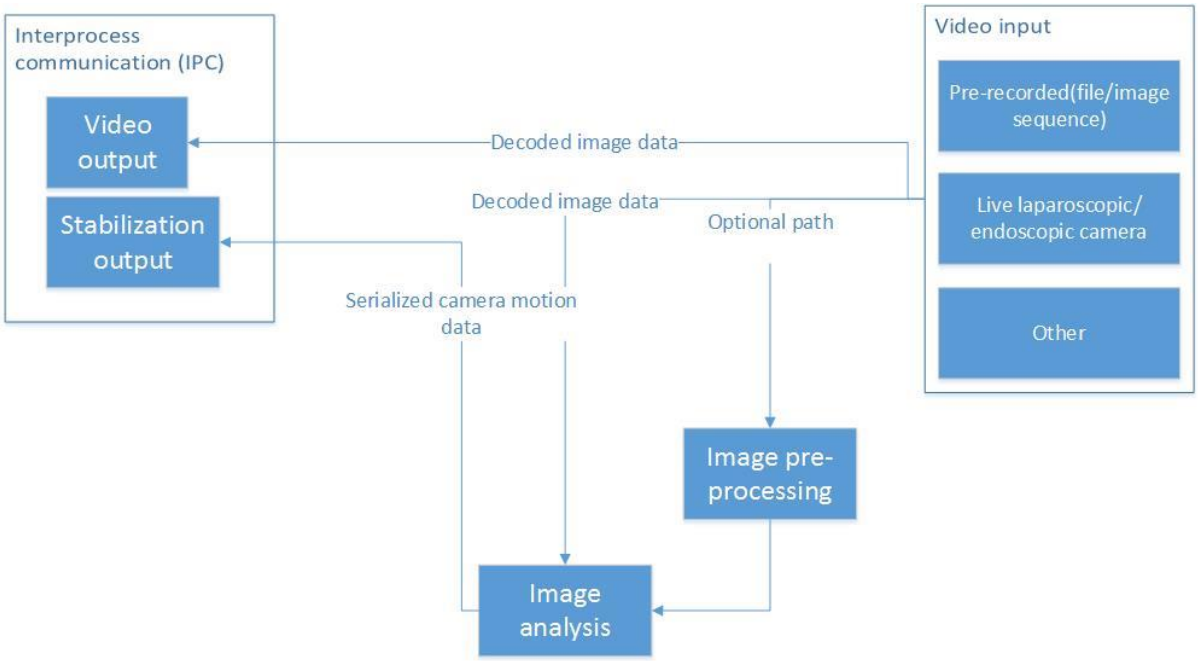


Figure 7.4 Stabilizer module design diagram

Stabilizer's **video input** is a part of stabilizer that is responsible for providing a stream of frames from the desired source. In case a mentoring session is going to feature surgical procedures that were recorded before, a file can be used as a video source. But if a live feed from a laparoscopic/endoscopic camera is needed, video input can be configured to use a camera of choice as a source as long as the camera is connected to a computer, where the module runs (**FREQ#2**).

The decompressed/decoded frames are then passed to a video output thread, which uses an interprocess communication method to transmit the video to the GUI module described previously.

In parallel, the decompressed/decoded frames are optionally passed to an image pre-processing module, where certain adjustments can be made (e.g. brightness/contrast adjustment, sharpening). This step is sometimes necessary in order to enhance the images' quality to make the image analysis more efficient.

Depending on the experiment/mentoring session circumstances, a pre-processed or raw decompressed frames are then passed to an image analysis module, which uses computer vision techniques in order to retrieve camera motion coordinates that are necessary for telestration stabilization.

Note that the image analysis node can be replaced by any other node as long as it yields correct/sufficient stabilization data in the end, including a node using different computer-vision techniques. This also means that if necessary, sensors other than camera can be used for camera motion estimation (**FREQ#6**).

The estimated camera motion data is then serialized into a certain standard and is passed to a thread that performs interprocess communication with a GUI module (**FREQ#5**).

8. Implementation

This chapter describes the implementation of the system's GUI module and the stabilization module in detail. See source code in Appendix A.

8.1.Technologies and Libraries Used

Microsoft .NET framework

A Microsoft .NET framework has been chosen as a basis for the system's GUI module. The framework is designed for and is primarily used on Windows-based systems, which complies with **NFREQ#1**. A framework provides a Framework Class Library that is shared across a certain set of languages, making modules written using .NET interoperable between this language set. A .NET framework-based programs are executed in a common language runtime (CLR) virtual environment, which resembles a JAVA runtime environment technique. The .NET virtual environment handles memory-management and certain levels of security, which allows the developers to not worry about memory leaks and certain security risks.

The latest .NET 4.5 version is used in this project in order to ensure support of most advanced GUI features, as well as the ways to work with asynchronous tasks and network streams (as a part of interprocess communication).

Windows Presentation Foundation

Windows Presentation Foundation (WPF) was chosen as a tool for implementation of the system's GUI. It is shipped together with Microsoft .NET framework, starting with version 3. WPF employs the DirectX collection of APIs to draw the user interface, which makes a great difference in performance if dealing with complex graphical structures.

WPF adopted the philosophy of describing business logic and visual components separately. An XML-based extensible application markup language (XAML) is used to describe the visual components of a WPF application. Each of XAML definitions corresponds to a certain object or attribute from the CLR, making it fully dependent and compatible with Microsoft CLR environment.

The WPF framework provides extensive capabilities for creation of graphical user interfaces.

C Sharp (C#)

C# programming language has been chosen for the system's GUI module implementation. This language is one of the languages that conform to common language infrastructure (CLI). This means that C# can be compiled to an intermediate language and be run on a CLI implementation (e.g. CLR provided by Microsoft). The C# can be used to utilize any functionality provided by the .NET framework.

Microsoft Visual Studio Express 2013

Visual Studio Express 2013 has been used as an integrated development environment for working with .NET, WPF, C# technology stack while developing a GUI module of the system.

OpenCV

Open Source Computer Vision is a library that provides the functionality necessary to operate on digital images. Its primary goal and most common application is processing of real-time image sequences.

Being written in C++, the library provides C++ language as its primary interface. Bindings for other languages are now implemented as well, making it possibly to utilize the library's functions in Python, Java and Matlab. A number of wrappers for other languages exists.

Emgu CV is an OpenCV .NET wrapper that can run on different platforms, which makes it possible to use any CLI language (including C#) to access the OpenCV functionality.

C++

A C++ language which is the primary interface language for OpenCV library, was used to develop an image processing module of the system and make use of some publicly available object tracker C++ Linux implementations.

GCC

The GNU compiler collection GCC has been used to compile the stabilizer module code on a Ubuntu 14.04 system.

Boost

A Boost collection of C++ libraries has been used in order to implement interprocess communication via network (Asio library from Boost collection) on the stabilizer module side, as well as to implement the multi-threaded approach.

Python

Python programming language was used to facilitate utility scripting on Ubuntu while creating test sequences for tracking. It has bindings for OpenCV libraries and allows to operate on video files or image sequences while working with high level abstractions.

Bash

Bash command processor for Linux systems scripting has been used to facilitate batch processing of video files and image sequences by running Python scripts or other programs with necessary command line arguments, which allowed to avoid hard-coding certain values.

8.2.GUI Module Implementation

8.2.1. Visual Hierarchy

The GUI layout has been described using XAML markup language. Most of the visuals are described in a *MainWindow.xaml* file. The hierarchy of visual elements is represented in Figure 8.1.

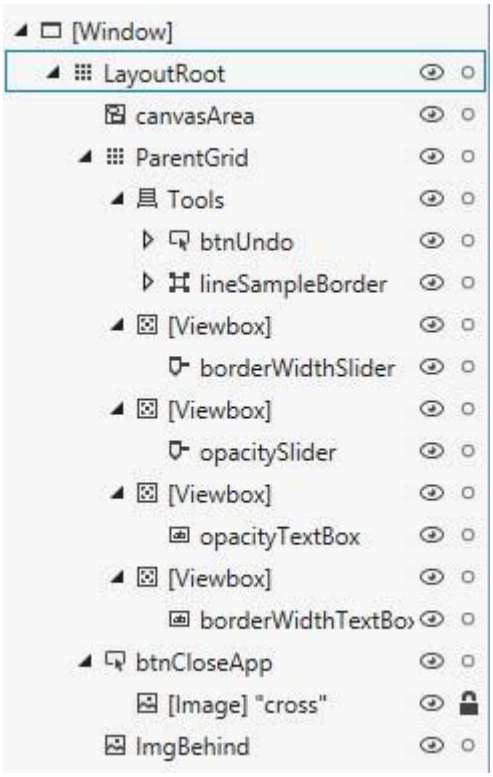


Figure 8.1Hierarchy of visual elements

The *LayoutRoot* component is a *Grid* type container which is used to store GUI elements. As the type name suggests, it has definitions of rows and columns that can contain elements inside of them. Being a root level container, *LayoutRoot* is a logical (and visual, in our case) parent of all the other GUI elements.

The *ImgBehind* is an immediate child of the *LayoutRoot* and is set to stretch all over its parent in order to fill the whole application width and height. This element has a *Source* property which will be used to store and display the incoming video stream. The *Source* property is set to display a blank white picture by default, stretched to fit the *ImgBehind* width and height. This element has the lowest *ZIndex* property (*ZIndex* = -1) which ensures that it will be displayed underneath the telestrations and the GUI control elements.

The *canvasArea* element serves as a visual container for telestrations that will be created by user. It has a *ZIndex* of 0, meaning that it will be displayed right on top of the video feed.

ParentGrid is a container for all the UI elements through which the user interacts with the system (besides the drawing functionality which is provided by a *canvasArea*).

The *Tools* element is a child of the *ParentGrid* and is a *StackPanel* container type. Its children are displayed as a vertical stack, making it a good component to contain the UI buttons.

The *Tools* element is a parent to 2 buttons: *btnUndo* and *lineSampleBorder*. *BtnUndo*, once clicked, will call a *Button_Undo* which cancels the latest telestration and the mechanism behind will be described later. A *lineSampleBorder* is basically a visual wrapper for a *lineSample* element, which acts like a button but is of *Image* element type. This button,

once clicked using a left mouse button, will call a tool selection dialog, and once clicked with a right mouse button, will call a color selection dialog. The dialogs and their logic are defined in separate XAML files. The *lineSample* element is also used to display the current telestration shape, width and color in order to make the user capable of instantly knowing what sort of brush he is going to use.

Several *Viewbox* elements are there to hold the opacity slider and the brush thickness / line width slider and respective numerical inputs.

The *btnCloseApp* element is a button that contains a cross picture and is responsible for shutting the application down.

The dialogs for tool and color picking were defined in separate files: *SelectToolDialog.xaml* and *SelectColorDialog.xaml* respectively. The two were defined in a similar fashion, using a similar visual structure (Figure 8.2,Figure 8.3):

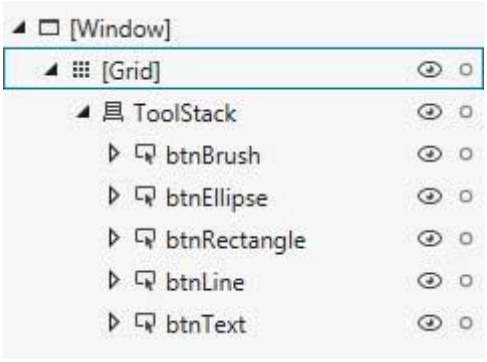


Figure 8.2 Visual structure: tool selection dialog

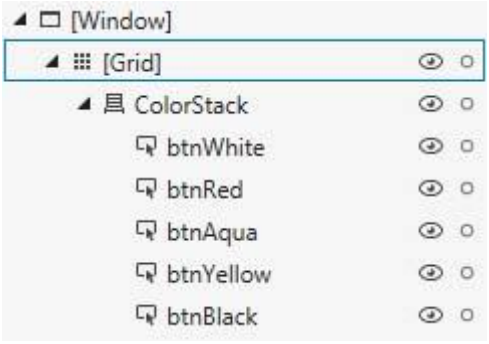


Figure 8.3 Visual structure: color selection dialog

They are both implemented using a grid component as a root-level container with a *StackPanel* inside. The *StackPanel* elements use horizontal orientation, allowing the stack to be filled in in a horizontal manner. A list of buttons in the two StackPanels represents the possible tool and color choices.

It is possible to choose from a simple Brush tool, an Ellipse, a Rectangle, a straight Line, and a Text tool in the tools dialog (Figure 8.4).

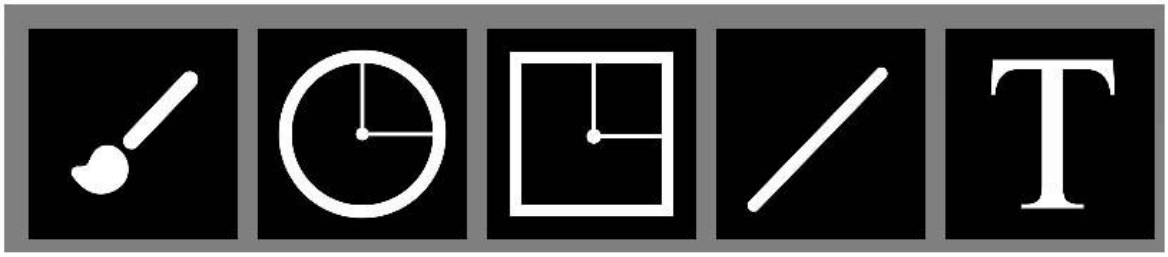


Figure 8.4 Tool picker dialog

The color dialog offers a choice from white, red, bright blue (aqua), yellow and black (Figure 8.5).

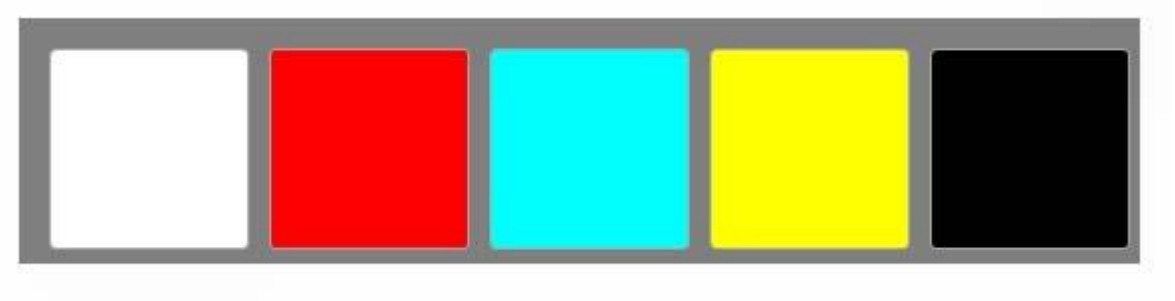


Figure 8.5 Color picker dialog

8.2.2. GUI Logic

Logic is separated from the UI element description but it specifies the way the elements influence the system's behavior and the way they represent data.

The *MainWindow* class contains several private and public variables that are the foundation for displaying a video and for all the telestration functionalities working.

The following variables are responsible for storing an image in OpenCV-compatible blue-green-red format (using an EMGU OpenCV wrapper for .NET), and a .NET Bitmap data structure that can later be displayed by an *ImgBehind* UI element (Code snippet 1).

```
Image<Bgr, Byte> emguImage;
System.Drawing.Bitmap emguImageConverted;
```

Code snippet 1 Emgu Image declaration

A list of telestrations (Code snippet 2) created by the user is also stored as a variable inside a *MainWindow* instance.

```
List<Telestration> listTelestration;
```

Code snippet 2 Telestration list declaration

A *Telestration* class defined in a *Telestracion.cs* file is a class defined to store any given telestration as a single object with certain attributes. A list of *Telestration* instances holds a collection of references to lists of *Shape* elements. *Shape* elements are the way the user-created telestrations are stored inside a *canvasArea* element from the visual hierarchy.

With an instrument and color selected, a user can draw telestrations similar to how it is done in any traditional drawing application – by pressing and releasing left mouse button and moving the cursor while the mouse button is down.

The drawing functionality is controlled by several events attached to a *canvasArea* element. *canvasArea_MouseDown* event tracks the moment when the user wants to start drawing a shape chosen by the tool selection dialog. Once the mouse is down for the first time, the shape, color and width are applied to the telestration that will be drawn.

The actual drawing happens when the mouse starts moving over the *canvas Area*, triggering a *canvasArea_MouseMove* event. Both mouse events provide a cursor position in their *MouseEventArgs* argument which allows to easily retrieve the position for a new telestration being drawn.

If the mouse is down and the mouse is moving, a new telestration is being drawn. If a freehand brush is the current tool, the line will be drawn along with the cursor path. However, if a certain shape-tool has been selected, it will only become persistent after the mouse move occurs without the mouse button being down. Either way, a sequence (in case of freehand brush) or a single instance of *Shape* class will be added as children to the *canvasArea* element. References to the very same recently-created *Shape* element(s) will be incapsulated into a Telestration class and inserted into a list of telestrations which will allow us to later manipulate the sets of telestrations in order to counter the possible camera movements.

Two additional lists are used in order to store a data structure called the "undo stack", which is used to facilitate the latest telestration cancellation (Code snippet 3).

```
List<List<int>> undoList = new List<List<int>>();
List<int> tempUndo = null;
```

Code snippet 3 Undo lists declaration

The *undoList* stores pairs of integer numbers which represent the beginning and the end of a telestration sequence as it is stored inside a *canvasArea*, which allows us to easily remove the latest sequence of *Shapes* created, effectively implementing the "cancel" functionality. Whenever a new telestration is being drawn, the respective *Shape* indices are first stored inside a *tempUndo* list, and after the mouse button has been released, the sequence boundaries are appended to the permanent *undoList*.

8.2.3. *Telestration* Class

The *Telestration* class is a high-level abstraction which is necessary to work with collections of telestrations. It allows the system to memorize the initial telestration's logical position in the "world" 2D coordinates based on the current known camera offset. Every telestration instance also possesses a pair of offset coordinates. A sum of "world" and "offset" coordinates yields the telestration's coordinates on the screen, which makes it possible for the telestrations to follow the camera movements and retain their logical position. The fact that a Telestration instance contains a list of *Shapes* allows to apply the 2D transform to all the *Shapes* in the list at the same time, which will immediately affect the telestrations that logically and visually belong to *canvasArea* element.

The moment a new *Telestration* is instantiated, a new *TranslateTransform* instance is created, which provides the utility functions to work with 2D transformation. The *TranslateTransform* is then incapsulated into a *TransformGroup* object, which can be applied to *Shapes* that will be added to given *Telestration* instance.

The following example illustrates how the telestrations keep their logical location while the camera moves. Whenever a new Telestration is created, it is assigned a current camera transform estimation, which equals (0,0) at the very beginning of the system's workflow. All the incapsulated shapes are assigned a certain $(x1, y1)$ on-screen position. (See Figure 8.6)

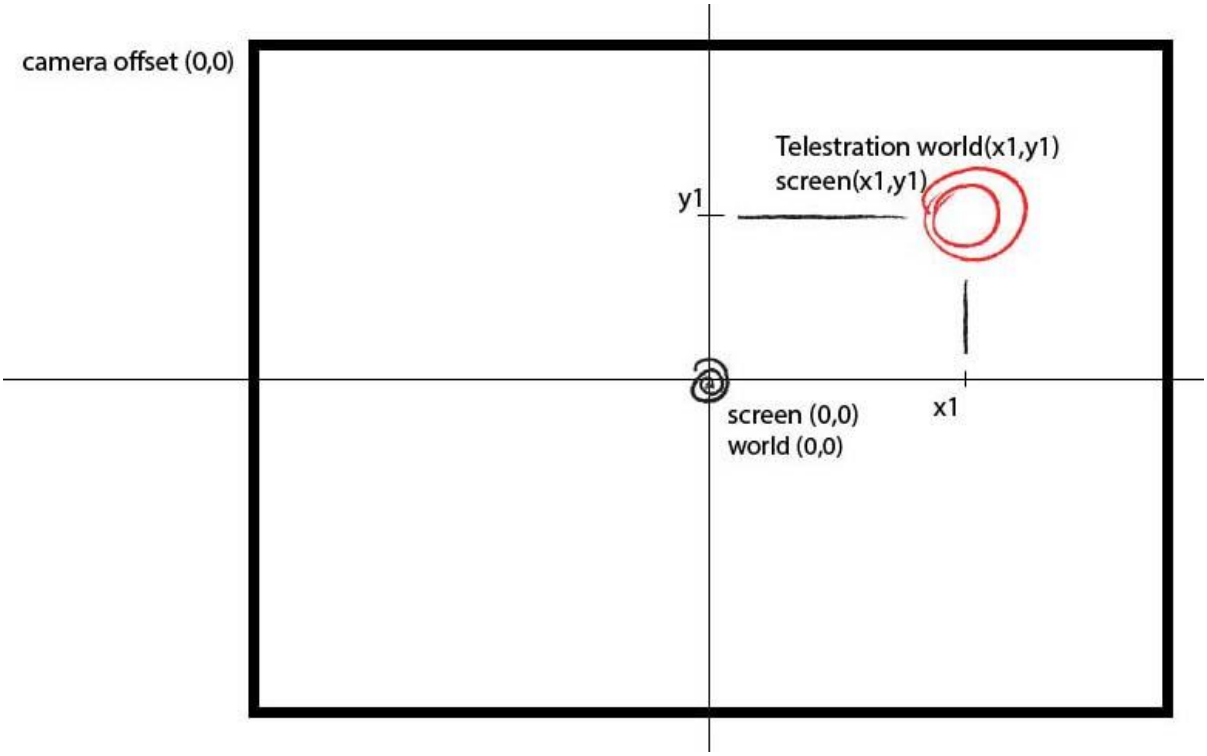


Figure 8.6 Telestration before camera movement

After the camera moves by $(x2, y2)$, this difference in positions is appended to the transform of *Telestration*, which results in positioning it at $(x1 - x2, y1 - y2)$ in screen coordinates while maintaining its world coordinates. (See Figure 8.7)

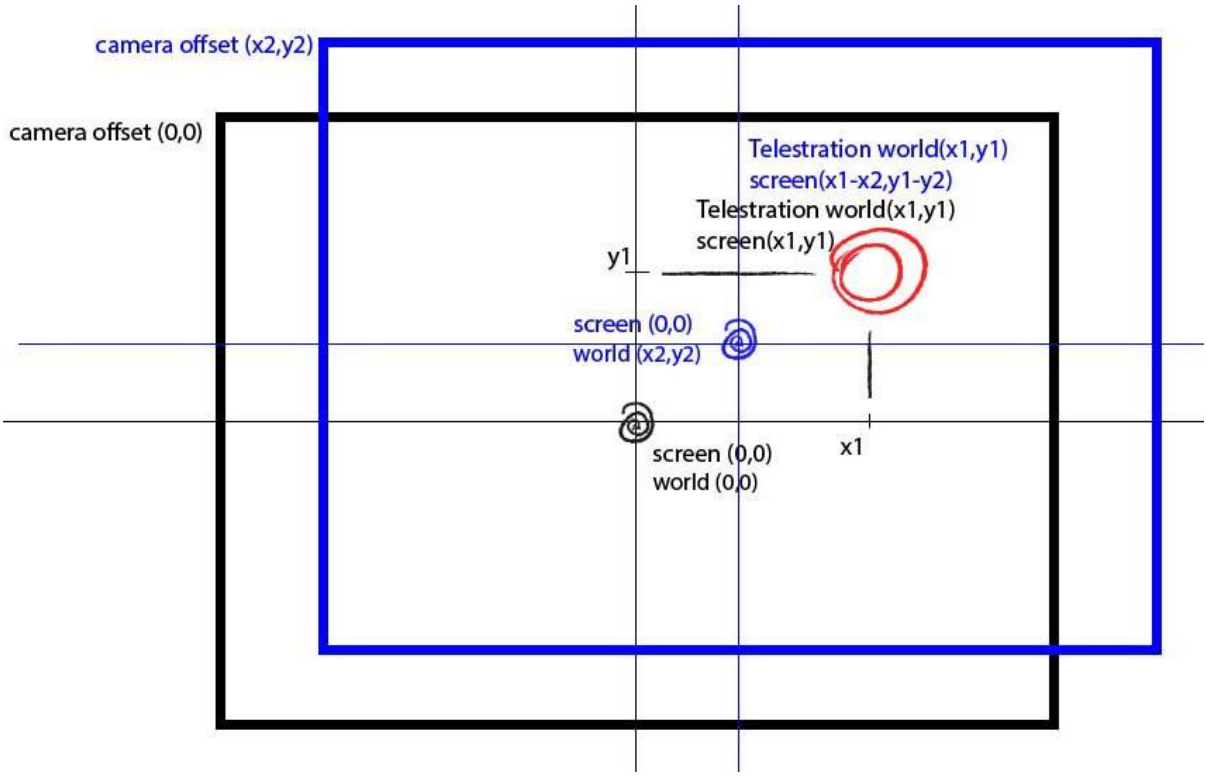


Figure 8.7 Telestration before and after camera movement

The fact that the *TelestrationGroup* is assigned to the *Shapes* by reference makes it possible to adjust the transform of all *Shapes* in one go when the corresponding method is called (Code snippet 4):

```
public void Translate(double x, double y)
        {
            tt.X += x;
            tt.Y += y;
        }
```

Code snippet 4 Appending camera offset to Transform object

8.2.4. Video Reception

TCP sockets have been chosen as a means of interprocess communication for video transmission, which makes it possible to launch the stabilizer module either on the same machine, or on a remote host if it is necessary.

A TCPListener is instantiated in the constructor method of the MainWindow class. It also starts listening to incoming connections with video transmission in the constructor method. The AcceptListener function accepts port number as an argument and is launched in the asynchronous manner, which prevents the UI thread from freezing until the connection is established (Code snippet 5).

```
private async void AcceptListener(int port)
        {
            this.port = port;
            this.ipAddress = new IPAddress(new byte[] { (byte)192, (byte)168,
(byte)230, (byte)1 });

            listener = new TcpListener(this.ipAddress, this.port);
            listener.Start();

            tcpClient = await listener.AcceptTcpClientAsync();
            worker.RunWorkerAsync();
        }
```

Code snippet 5 Asynchronous accept listener

The moment the connection is established, a background worker thread is instantiated in order to asynchronously receive a video stream. The system is designed to work with images of fixed size (640 x 480 px) that have 1 byte per pixel value and 3 color channels (RGB/BGR), meaning that one decompressed frame will be of 921600 bytes size. One a background worker receives a frame, it converts the incoming EMGU-format image into a Bitmap that can be displayed by a WPF *Image* user interface element. The background worker than starts receiving the next frame. The process is repeated indefinitely.

A *NetworkStream* abstraction class provided by .NET has been used to work with the images received. This level of abstraction allows to avoid unnecessary complexity of working with low-level TCP socket interfaces.

8.2.5. Stabilization Data Reception

The stabilization data is received asynchronously in parallel with the video frames that are later displayed to the user. Asynchronous *Task* .NET class is utilized to create an *AsyncService* class which is responsible for receiving the transform deltas from the stabilizer module. The *AsyncService* class is defined in a separate file (*AsyncService.cs*).

An instance of *AsyncService* is created and run together with the *MainWindow* instance. TCP sockets were used as a means of interprocess communication with the stabilizer module. As soon as the AsyncService is instantiated, it starts listening to incoming connections from the stabilizer module. The transforms are sent as a stream of data and are delimited by a newline symbol, which makes it possible to read them using a utility function of a *StreamReader* .NET class (Code snippet 6):

```
string request = await reader.ReadLineAsync();
```

Code snippet 6 Asynchronous line reading from a stream

As soon as the new line is received, it is split into two coordinates using space character as a delimiter. The *Vector* of two doble-precision floating point numbers is then passed

to the MainWindow class as a *TransformReceived* event where it is applied to a *Telestration* list available at the moment and keeps the telestrations intact with estimated camera movement.

8.2.6. GUI Module Summary

The GUI module in essence runs 3 threads and 2 of them are dedicated to asynchronous interprocess communication which is in turn necessary to receive a stream of video and estimated camera position alterations. (See Figure 8.8)

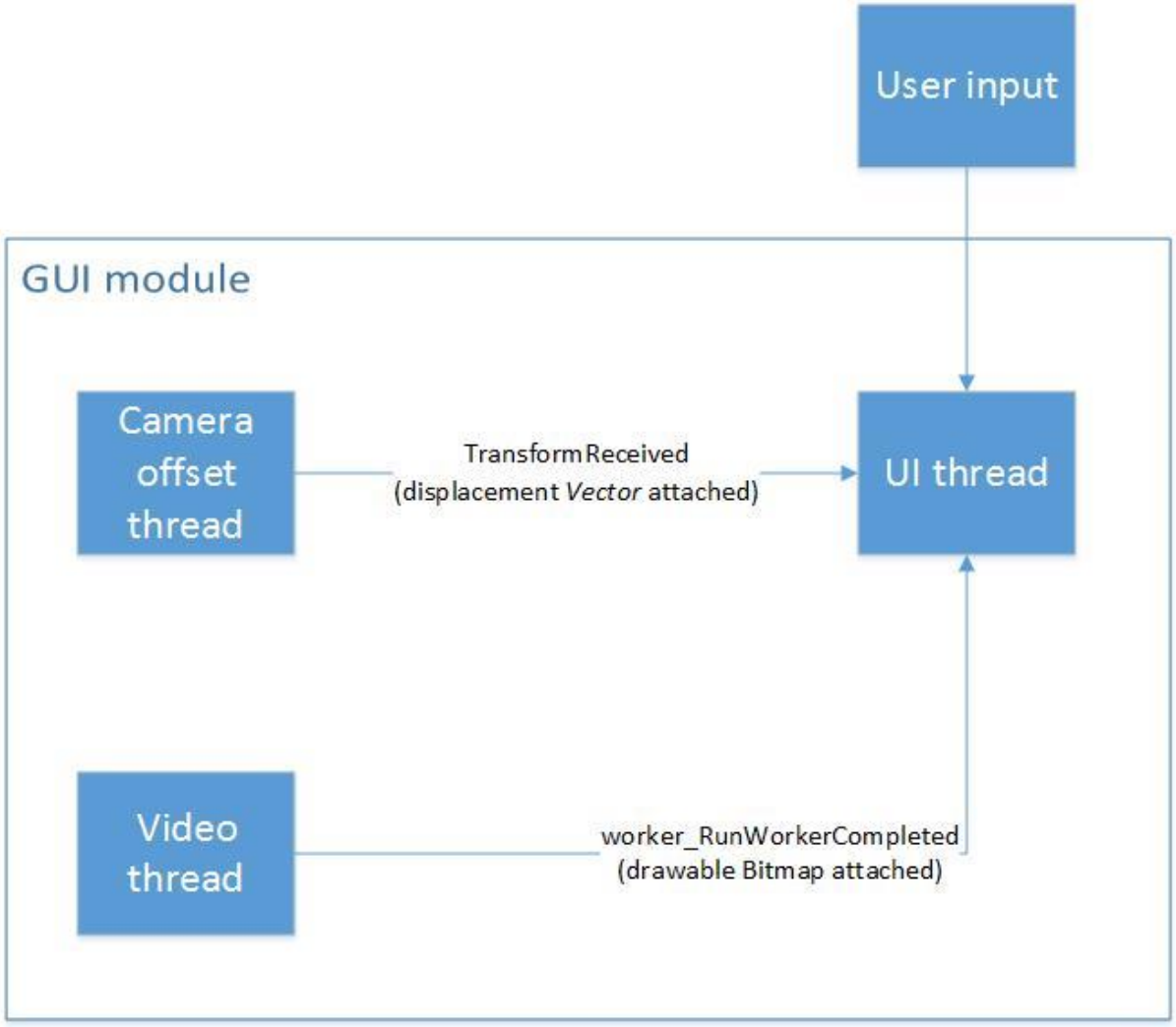


Figure 8.8 GUI module threads

8.3. Stabilizer Module Implementation

The stabilizer module has been implemented on a Ubuntu 14.04 system using the C++ language, a Boost collection of libraries and the OpenCV library for Linux systems, and a GCC compiler for C++. The module is launched with several mandatory and optional command-line parameters that make it possible to choose the parameters for interprocess communication, the source of video, and the object tracker that will be used as a source of stabilization data.

8.3.1. Video Input

The stabilizer module is launched with an optional parameter to specify a video source file. This gives the user an opportunity to telestrate on a pre-recorded surgical footage. If this optional parameter is omitted, the camera connected to a machine that hosts the stabilizer, will be used as a source of frames (see Code snippet 7).

```
VideoCapture cap;    // capture device declared
…
cap = VideoCapture(0); // camera input
…
cap = VideoCapture(argv[5]); // path to a video file (5 th
                             //command-line argument)
```

Code snippet 7 Video capture initialization

If the user chose to telestrate on a video file, it is also possible to specify the offset in seconds so that the telestration session will start from a certain segment of video, possibly omitting the parts that are of smaller interest.

As soon as the parameters are successfully passed, a thread that reads and transmits the video is started.

The thread first waits for the connection with a GUI module to be established and then initiates the reading/sending loop of the data from the chosen capture device (a file or a camera). The frames are read as a decompressed default OpenCV matrix structure that contains a certain number of columns and rows (in our case, 640 x 480) and is in our case 3 channels deep, with each single pixel encoded by 1 byte of information. After the frame is decompressed, it is send asynchronously via a socket connected to the GUI module.

Once one frame is decompressed and fully sent to the GUI module, the thread executes a handler and starts reading and sending the next frame provided by a capture device.

8.3.2. Stabilization Data

Once the command line arguments are successfully parsed, a stabilization thread is started in parallel with the thread for reading a video stream. The stabilization thread first waits for the connection with the GUI module to be established. After that, an instance of tracker that has been chosen is instantiated.

A collection of trackers provided by the tracking module in *opencv_contrib* repository branch[25]. The collection includes the MIL, the Online Boosting, the Median Flow and the TLD trackers. A common interface is provided in order to allow the developers to work with all the trackers in a uniform manner.

[25] https://github.com/Itseez/opencv_contrib/tree/master/modules

First, a bounding box rectangle has to be created in order to define the area that has to be tracked. The very first frame is then passed to the tracker *init* method together with the bounding box, effectively initializing it and making it ready to work (see Code snippet 8). Assuming that the structure of interest is in the center of the screen, a bounding box is positioned to have its center right in the center of the initial image, and is 200 px wide and high.

```
Rect2d boundingBox;
…
boundingBox.x = bbx0;
boundingBox.y = bby0;
boundingBox.width = bbw;
boundingBox.height = bbh;
…
cap >> frame;
…
tracker->init( frame, boundingBox );
```

Code snippet 8 Bounding box and tracker initialization

The stabilizer thread copies frames from the ones read by a video thread and feeds them to the tracker in an infinite loop, calling its *update* method (see Code snippet 9).

```
tracker->update( image, boundingBox )
```

Code snippet 9 Tracker update

The bounding box structure is then upgraded to conform to the camera movement if it happened and it is possible to retrieve the movement offset. After this happens, the new bounding box coordinates are serialized and passed on to the GUI module via the TCP socket as a means of interprocess communication. The serialized bounding box coordinates are a string that contains a pair of double-precision floating point numbers, separated by a space and terminated by a newline character.

The whole image processing loop logically looks as follows (Figure 8.9):

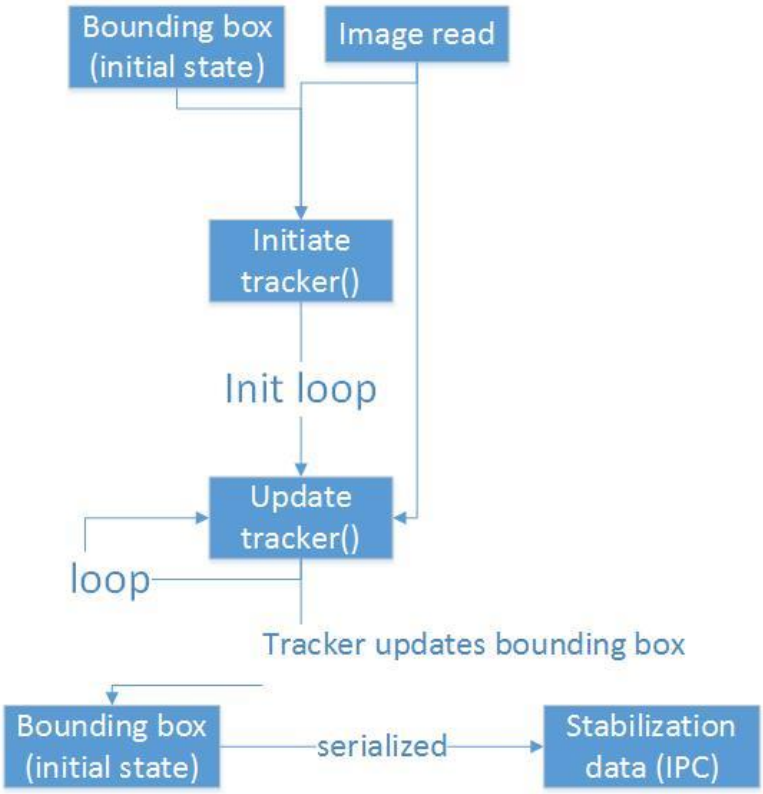


Figure 8.9 Image processing thread

8.3.3. Interprocess Communication

The interprocess communication has been implemented using Unix TCP sockets, wrapped into a Boost library collection in order to work on a higher level of abstraction.

In both threads that are communicating with a GUI module, an individual socket on the same IP address with 2 different ports is used. The ports are specified in the command line parameters of a stabilizer module.

After the connection is established, the data is written asynchronously in chunks, which is handler by an *async_write* function from Boost asio library. After the async_write is completed, a handler function is called to continue sending data as long as something to send is present.

Using separate ports for different kinds of data makes it possible to make changes to the programs' components in a more understandable and scalable way. The video retrieval and compression/decompression method can be altered without touching the image processing code, which in turn can be modified to use a better of a completely different solution to the required image analysis.

8.3.4. Stabilizer Module Summary

In essence, the stabilizer module runs two threads in parallel (Figure 8.10 Stabilizer module parallel threads). Both utilize asynchronous sockets and transmit the data as soon as it is ready to be shipped, providing the necessary information to the GUI module which can safely run either on the same machine or on the remote host, allowing the stabilizer module do the image analysis "heavy lifting". Such an implementation leaves place for introduction of image preprocessing techniques and replacement of the image

processing algorithm in its entirety if a better alternative is found, while not disturbing the implementation of streaming or GUI module.

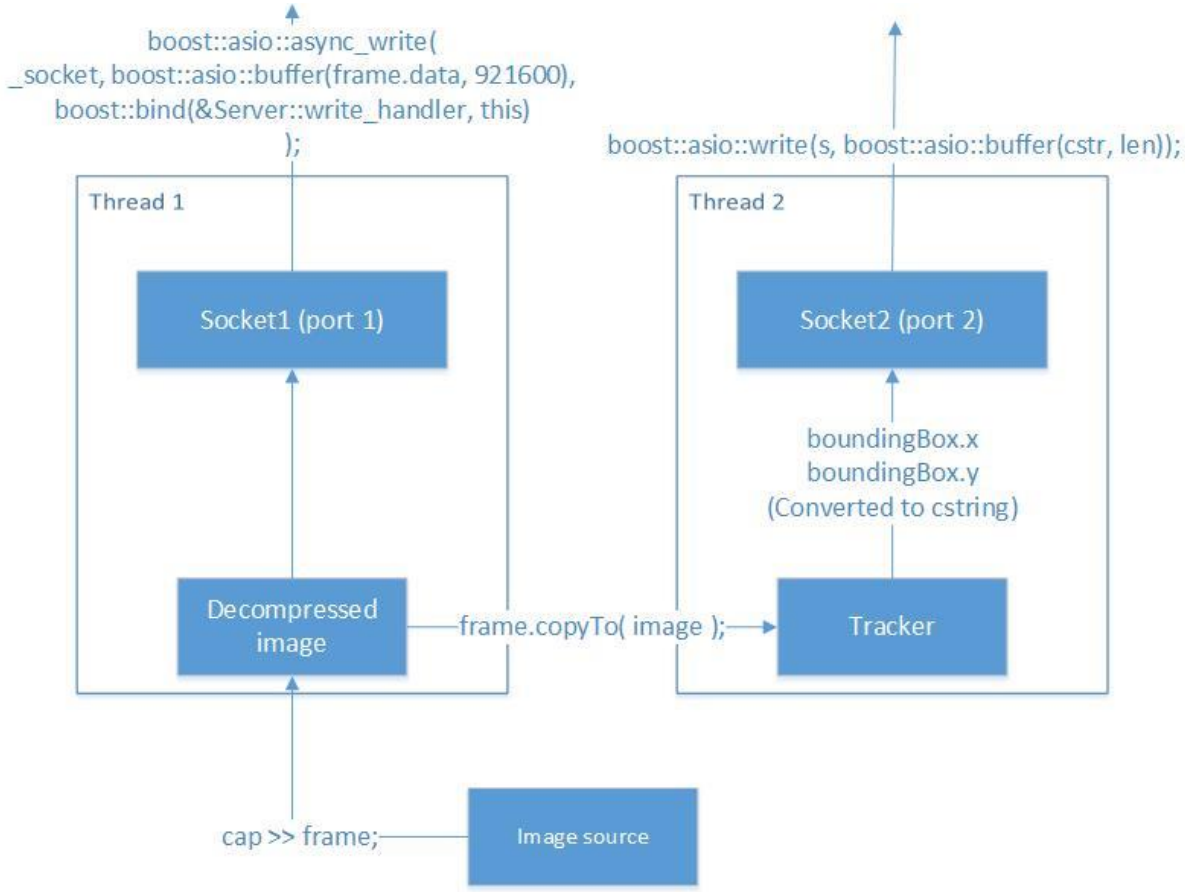


Figure 8.10 Stabilizer module parallel threads

8.4.Image Preprocessing

Image preprocessing can optionally be included into the stabilizer's pipeline. It is possible to visually enhance the image right before it is going to be processed by an image analysis algorithm.

It may sometimes be necessary to enhance the picture's brightness/contrast values or sharpen it, which can help the optical trackers obtain more features to track.

8.4.1. Histogram Equalizer

Histogram equalization has been implemented in Python as a part of offline preprocessing routine. The method uses OpenCV functions which means that it is possible to implement the same approach using a C++ programming language and integrate it into the system if necessary.

The python code highlights that make the equalization happen are displayed in

```
img = cv2.imread(path, 1)
imgHist = cv2.cvtColor(img,cv2.COLOR_BGR2YCR_CB)
channels = cv2.split(imgHist)
channels[0] = cv2.equalizeHist(channels[0])
imgEqualized = cv2.merge(channels)
imgEqualized = cv2.cvtColor(imgEqualized,
    cv2.COLOR_YCrCb2BGR);
```

Code snippet 10 Histogram equalization

The image that has to be equalized is first read into a matrix object as a BGR (blue, green, red) picture with 3 channels. It is then converted to a YCrCb color space, making it possible to equalize a colorful picture using the Y channel.

The image is split into 3 channels and the Y channel is then equalized. After that the 3 channels are merged back in order to represent an image that has 3 channels, and finally the image is converted to BGR format which can then be either passed to an image analysis algorithm or written into persistent memory.

8.4.2. Sharpener (Unsharp Masking)

Image sharpening is implemented using an unsharp masking technique which first calculates a blurred version of the original image and then adds the blurred image to the original using weighted sums (see Code snippet 11).

```
img = cv2.imread(path, 1)
blur = cv2.GaussianBlur(img,(101,101),0)
sharpened = cv2.addWeighted(img, 1.5, blur, -.5,0)
```

Code snippet 11 Unsharp masking

The image is first read as a BGR image. A blurred image is created with 101 and 101 as width and height of the Gaussian kernel. The original image is added to the blurred one, with 1.5 and -0.5 multipliers respectively, efficiently subtracting the blur from the image, sharpening it as a result.

8.5.Summary

The system has been implemented in a distributed/modular fashion, employing a cross-platform method of interprocess communication, which can later be exploited for further development and modifications. Moreover, it is possible to implement a GUI module for any operating system or device (including cross-platform browser-oriented Web-RTC approach), because it is in essence a thin client for a stabilizer module which processes and broadcasts the video.

9. Testing

9.1.Application Testing

The application has been tested with expert surgeons from UNN using a conventional Windows-based PC and an Ubuntu 14.04 laptop.

The laptop (Intel Core i7 4500U CPU) was running the stabilizer module of the application, while a stationary PC (Intel Core i5 3450s) was running a GUI module.

The PC was connected to the laptop via a peer-to-peer WiFi connection with ping ~1ms and the throughput speed at least as big as 50Mb/s, which allows to transmit the decoded OpenCV images of 640x480 size at ~50 FPS, making the transmission real-time with extremely slow delays.

The peer reviewers (Hiten Patel, Etai Bogen) reported the software to work as expected (the telestration moving according to the estimated camera movement). Several suggestions were made on the possible improvements which will be discussed in the Discussion and Future work chapters.

While the systems are in vicinity and the network bandwidth is not a problem, it is safe to assume that one of the potential problems is the image analysis overhead.

9.2.Object Tracker Testing

9.2.1. Object Tracker Selection

A set of 5 conventional object trackers has been selected in order to compare their performance on laparoscopic surgical videos. All the trackers are not proprietary and have their source code in public access.

The 4 trackers available in the OpenCV experimental *contrib*[26] repository were primarily selected. The *tracking* module of *contrib* repository provides C++ implementations of the following trackers described previously:

- MIL
- TLD
- Median Flow
- Online Ada Boost

However, the TLD implementation provided in this module seems to be out of order and performs poorly. As an alternative, an optimized OpenTLD implementation of TLD was found.

In addition, a CMT tracker has been included into the list as it is described as a competitor of TLD and was developed by the same research group that provided an OpenTLD implementation. A Python implementation of CMT is provided[27].

All the trackers listed above provide a universal interface and functionality (Figure 9.1).

[26] https://github.com/Itseez/opencv_contrib
[27] https://github.com/gnebehay/CMT

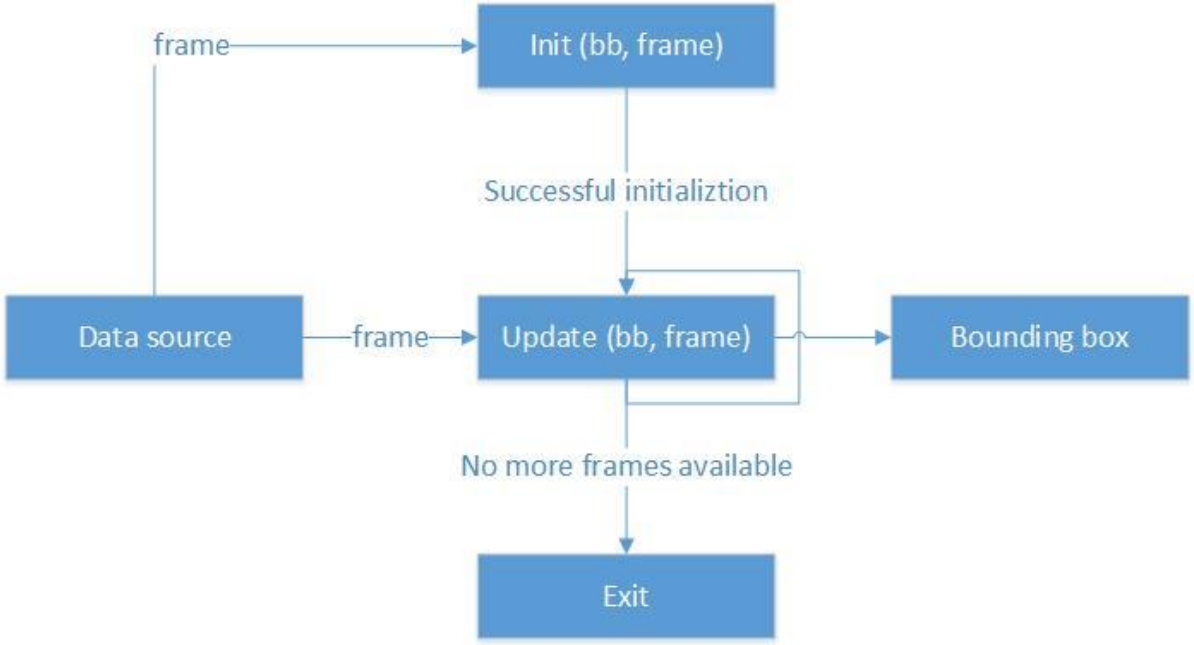


Figure 9.1 Tracker interface diagram

9.2.2. Data Set

A data set for performing the tracker testing has been acquired through several consequent steps.

9.2.2.1. Acquisition of Laparoscopic Surgical Videos

First, a set of laparoscopic surgical videos has been acquired from two sources. Expert surgeons from UNN provided 4 laparoscopic videos. The second portion of videos has been acquired via Youtube[28]

The following keywords were used to perform the search: “laparoscopic surgery”. Only videos that longer than 20 minutes were included into the initial video set. Videos with a large portion of view covered by a black mask (resulting from the camera shape) were removed from the initial set. Due to the fact that the tracking solution presented in this work was tailored to work with frames of 640x480 size, videos with maximum available quality less than 440p were removed from the set.

The videos from a filtered set were examined by author and expert surgeons from UNN, and several parts of videos that may be subject to telestration and object tracking (with different degree of camera shaking and instrument overlapping) were extracted from the original videos. The selected chunks of video can be downloaded via the link presented in Appendix B.

The resulting set contains 6 videos and is presented in Appendix D.

9.2.2.2. Pre-processing

The previously created chunks of videos have been broken down into separate frames in order to perform image enhancement.

[28] www.youtube.com

All the frame sequences have been processed by a script that performed histogram equalization. The original set of pictures has also been processed by an image sharpener script. Finally, the original set of pictures has been processed by both histogram equalizer and image sharpener.

The script for pre-processing of image sequences have been implemented using the Python programming languages and the OpenCV library (See Appendix C).

The sequences have then been stitched back into the videos and given specific prefixes to identify the method of preprocessing that has been applied. *eq_* prefix has been used to identify a video processed by a histogram equalizer. *sh_* prefix has been used to identify a video processed by an image sharpener. *sh_eq_* prefix has been used to identify a video that has been processed by an equalizer first, and then processed by a sharpener. A video that has no prefixes that were mentioned above is the original video.

FFmpeg[29] command line tool has been used to perform video break-down and assembly.

The resulting data-set contains 24 videos, 6 of which are original, and the rest are a result of pre-processing as described above. This way, every original video has 3 enhanced versions, processed by different means.

The videos described above are used to compare performance and precision of the subject trackers. Some videos have more camera shaking than others, either due to respiratory movement or to the instability of manual camera holding. The videos also possess different levels of texturization and blurriness. These characteristics that vary from one video to another will allow us to see which of the subject trackers perform best in challenging conditions and whether the preprocessing affects the tracking result.

9.2.2.3. Feature Detection

A number of frames from the videos described above have been used to perform a feature detection test that would yield evidence whether histogram equalization or sharpening affect the amount of features detected in the same surgical image.

The amount of features detected might significantly boost the tracking quality since all the object tracking approaches as well as the absolute majority of computer vision techniques rely on tracking different kinds of features.

9.2.3. Benchmark

9.2.3.1. Benchmark Structure

Every subject tracker has been given the same bounding box (Figure 9.2) and all the 24 videos were processed this way. As a result, every tracker produced a file having the tracker's name as a prefix followed by a video name: <tracker_name>_<video_name>.txt.

[29] https://www.ffmpeg.org/

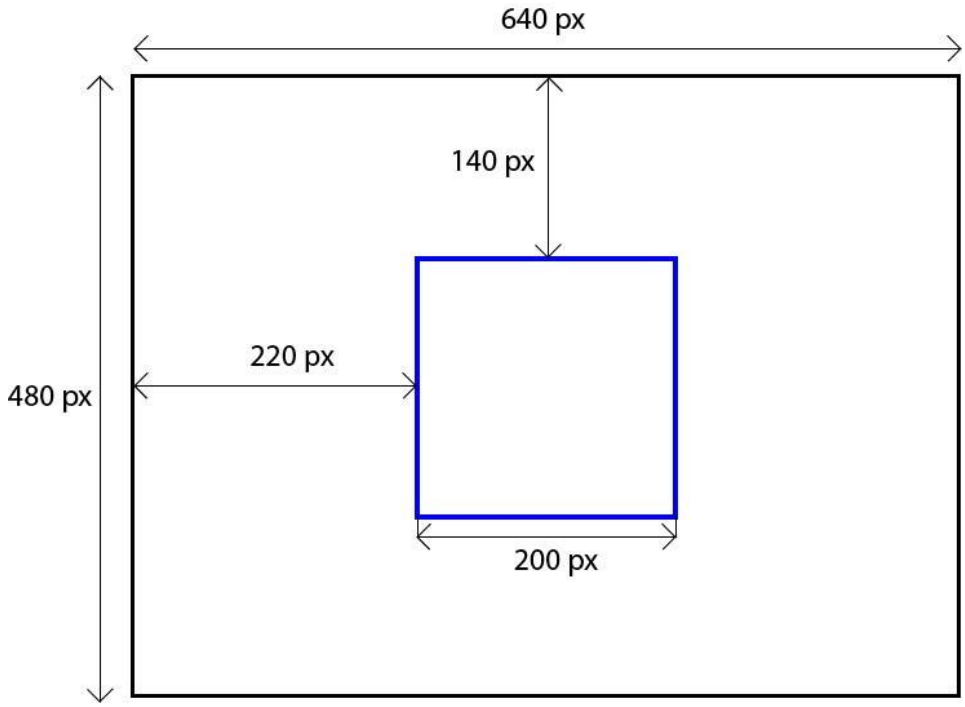


Figure 9.2 Testing: Tracker Bounding Box

Every file contains a number of lines equal to the number of frames in a given video. Every line contains two integer numbers that denote the position of the top left corner of a bounding box at a given frame (e.g. "200 200" line denotes the x=200, y=200 position of the bounding box).

Whenever the current tracker is not able to find an object in the current frame, a NaN (not a number) value is printed out.

Furthermore, while processing the video sequences, the tracker's frames per second rate has been calculated which will allow to see how the algorithm speed varies across different types of video, both enhanced and raw.

9.2.3.2. Benchmark Results

Preprocessing

The methods used for pre-processing in this work demonstrate the FPS rates that can be used in real-time image analysis and streaming (Table 2).

Pre-processing method	FPS
Equalization	96
Sharpening	28
Equalization and Sharpening	22

Table 2 Preprocessing FPS

Feature detection

An experiment on select pictures from the data set has been performed in order to determine whether the amount of features that are extracted from an image alters when histogram equalization or image sharpening are applied.

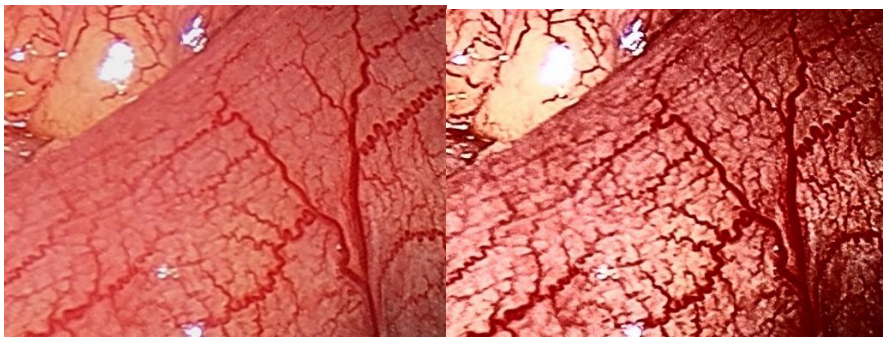


Figure 9.3 Raw image (left), Equalized image (right)

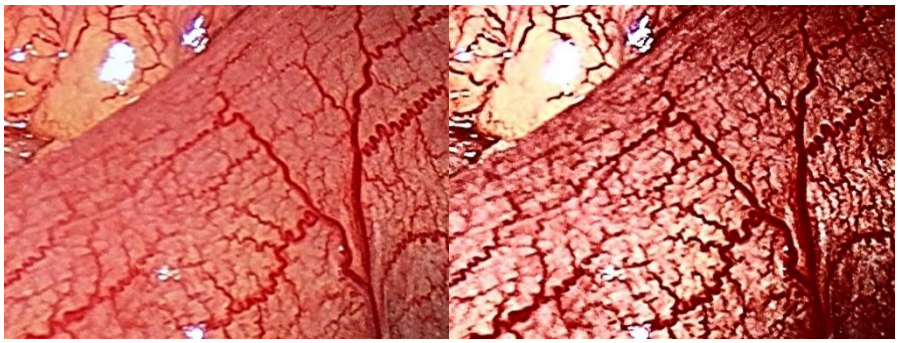


Figure 9.4 Sharpened image (left), Sharpened and Equalized image (right)

BRIEF and FAST feature detectors (the only non-proprietary feature detectors openly available from the library) provided by OpenCV library have been used to retrieve the number of features in the images presented above (see Table 3 for results).

Feature type	raw	equalized	sharpened	eq. and sharp.
FAST	7868	14293	12886	15037
BRIEF	250	1660	697	1945

Table 3 FAST and BRIEF feature detection

Histogram equalization provides a significant boost in the number of features detected in case of both FAST and BRIEF features. Image sharpening provides a considerable increment as well, but in a slightly lesser degree. However, the most prominent feature number boost is observed when both equalizing and sharpening are applied.

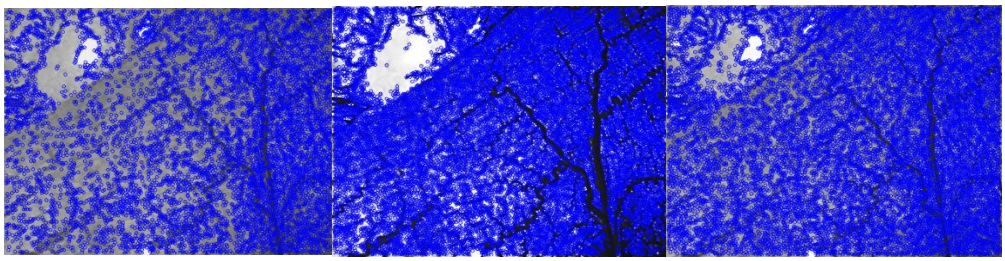


Figure 9.5 FAST features in original, equalized and sharpened images (from left to right)

Frames per second

The frames per second rates of all the trackers on all videos (including pre-processed ones) have been calculated.

Several trackers such as Ada Boost(Figure 9.6), MIL(Figure 9.7) and TLD(Figure 9.8) did not display significant difference between the pre-processed and raw videos.

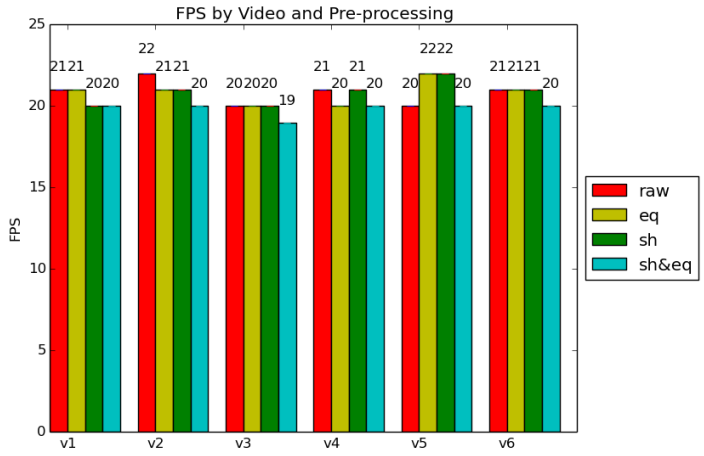


Figure 9.6 Ada Boost FPS

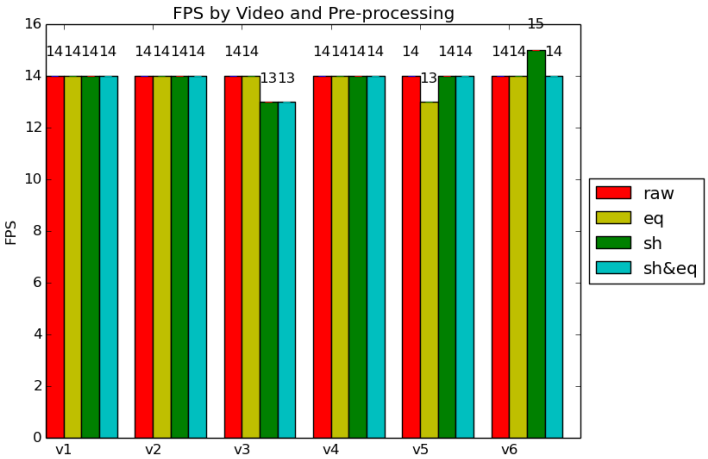


Figure 9.7 MIL FPS

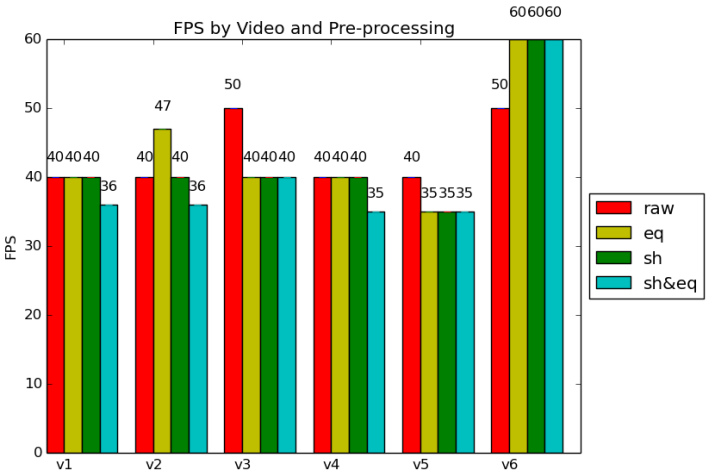


Figure 9.8 TLD FPS

However, TLD displayed higher FPS for raw videos in some cases, which is most likely related to the amount of features found in the picture, which increases with the pre-processing applied.

The CMT tracker displayed severe speed reduction when the video was preprocessed which is most likely due to the high limit of features to track. This, however, makes the tracker more robust and tolerable to instrument overlapping or partial occlusion of region of interest, and with the feature limit set to a lower number, the FPS rates are not going to be as low. This tracker demonstrates the higher FPS on pictures with low texturization (Figure 9.9 CMT FPS).

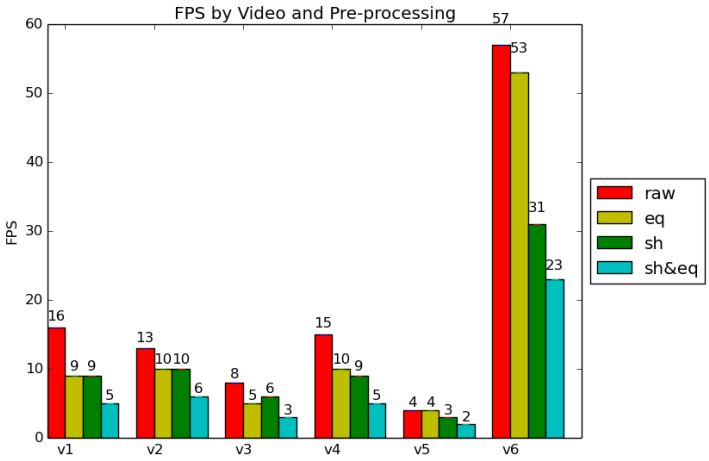


Figure 9.9 CMT FPS

The Median Flow tracker displays framerates that are more than enough for real-time applications, in some cases demonstrating a decline of FPS in response to pre-processing and the increased amount of features found as a consequence. The extremities that exceed 400 FPS are a result of tracking failure and can be disregarded (Figure 9.10 Median Flow FPS).

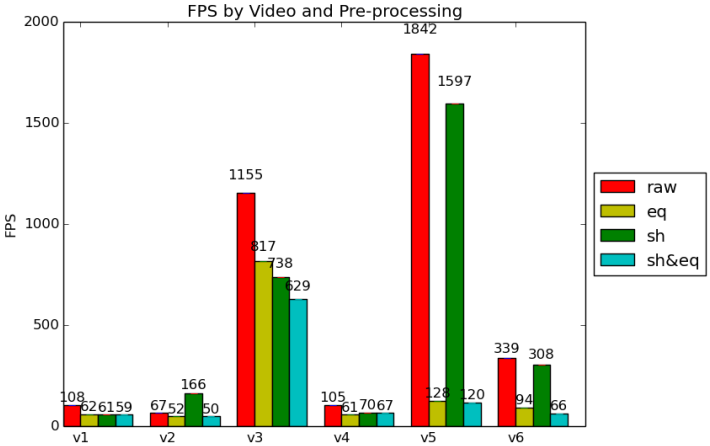


Figure 9.10 Median Flow FPS

Object tracking

Based on the bounding box coordinates retrieved from all the available trackers processing the various versions of available videos, it is possible to observe correlation between the video properties and the tracker robustness and degree of similarity between the results of other trackers. The following video properties and corresponding tracker behavior has been observed. See the videos and the video naming conventions in Appendix D. See the complete set of coordinate plots in Appendix E**.**

v1 video does not contain any large scale movements and is almost static apart from minor camera shaking due to camera holder instability and patient breathing. It can also be characterized as a video with very low texture present. There is, however, a large specular occlusion in the center of the scene.

In this video, the TLD tracker is the only one that correctly follows the camera movements (Figure 9.11).

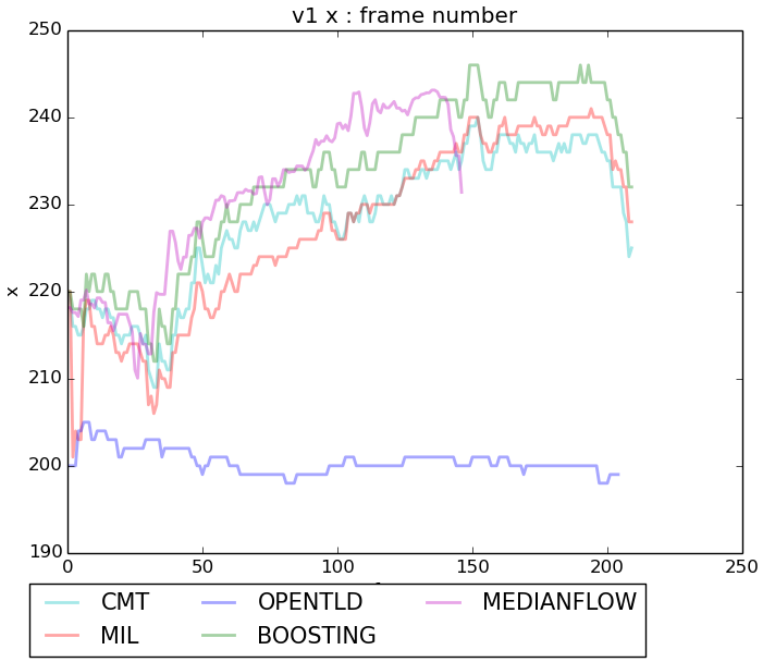


Figure 9.11 v1 raw tracking

The other trackers drift away significantly in the very beginning, which is caused by a specular occlusion in the middle of the scene and its low texturization. However, from then on all the trackers but TLD agree on the trajectory.

Sharpening, equalizing and doing both increases the tracker's trajectory similarity (apart from TLD), and significantly reduces jitter.

The TLD trajectory slightly alters after the pre-processing of the video, but also reduces the jitter.

It is noticeable that the Ada Boost tracker drifts away significantly after the sharpening pre-processing takes place (Figure 9.12).

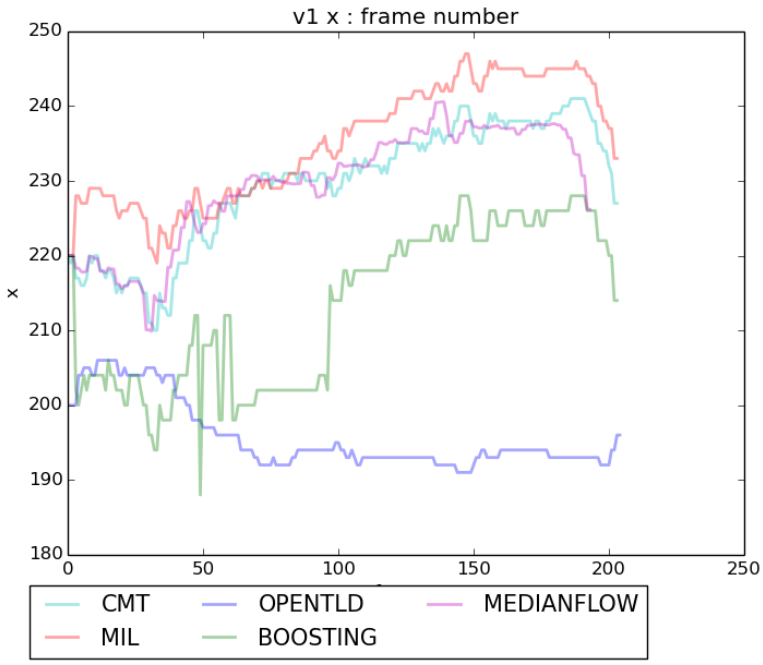


Figure 9.12 v1 sharpened tracking

v2 video demonstrates higher texture presence and a significant amount of movement over time. In this case, all the trackers but TLD recover similar trajectory with Ada Boost tracker deviating slightly more than others. On the contrary with the previous case, TLD does not recognize the high-amplitude movement that occurs in the video. (Figure 9.13)

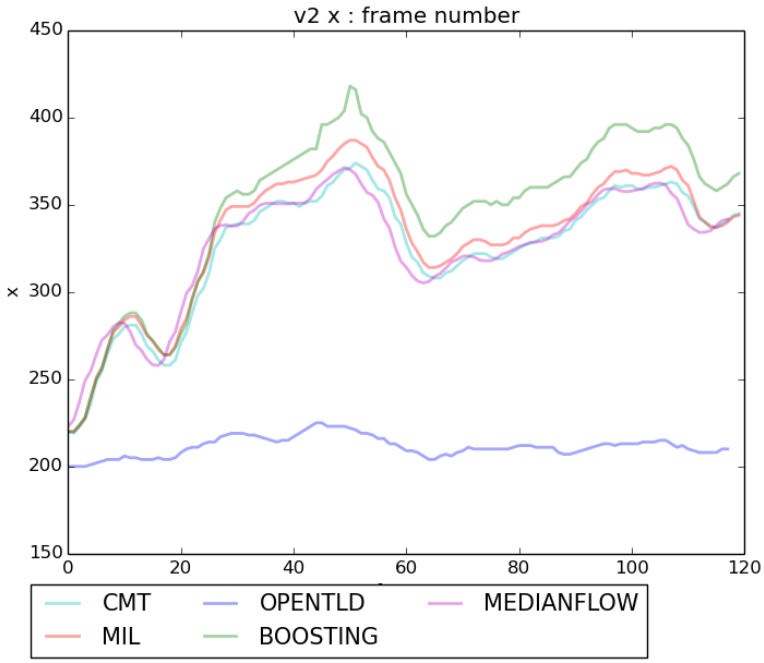


Figure 9.13 v2 raw tracking

Histogram equalization leads to greater similarity between the trackers' results (except TLD). Again, sharpening the image makes the Ada Boost significantly drift from its original trajectory.

The TLD tracker does not display significant changes in trajectory while working with preprocessed videos of any kind.

v3 demonstrates high texture presence, a moderate amount of movement and the introduction of an instrument that overlaps the region of interest.

The TLD tracker fails to recover the position as soon as the instrument is introduced, while other trackers display similar trends (with Ada Boost being offset significantly, while retaining the trend).

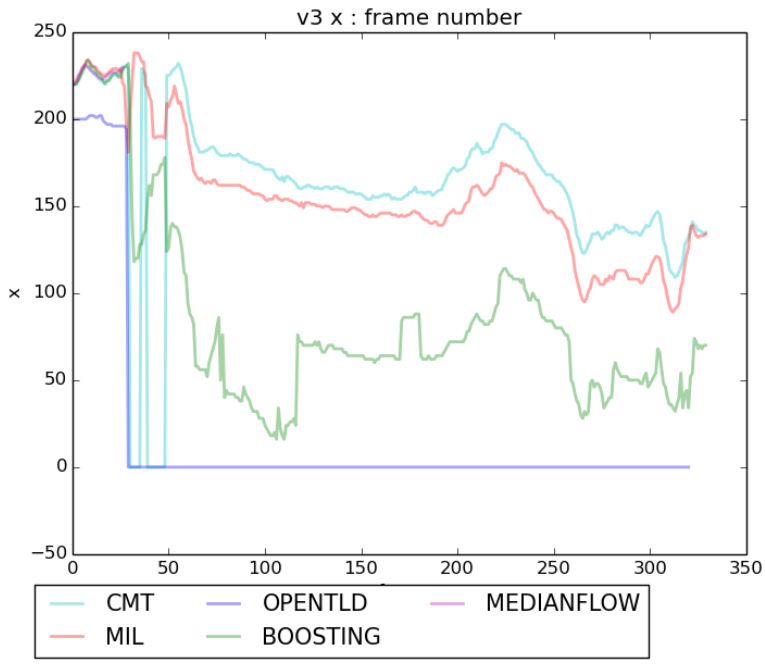


Figure 9.14 v3 raw tracking

Histogram equalization reduces the Ada Boost offset and makes its trajectory more similar to the other trackers (Figure 9.15). Further image enhancements increase the degree of similarity between trackers' results.

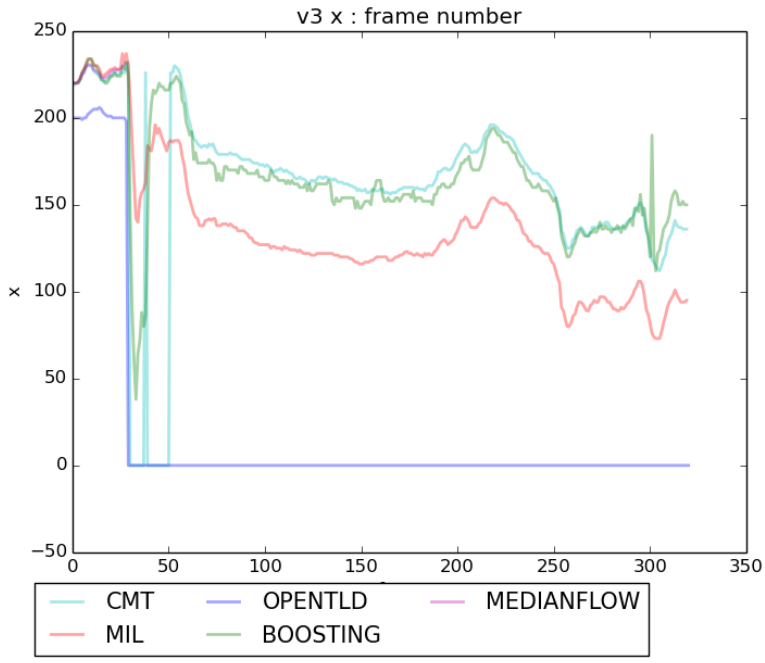


Figure 9.15 v3 equalized tracking

The TLD tracker does not recover after instrument introduction no matter the image enhancement technique.

v4 video is similar to the **v1** sample. A bright specular occlusion and low texture with almost no movement. In this case, the TLD trajectory is the closest to what can be observed in this video. Other trackers drift away from TLD's trajectory due to the specularity. (Figure 9.16)

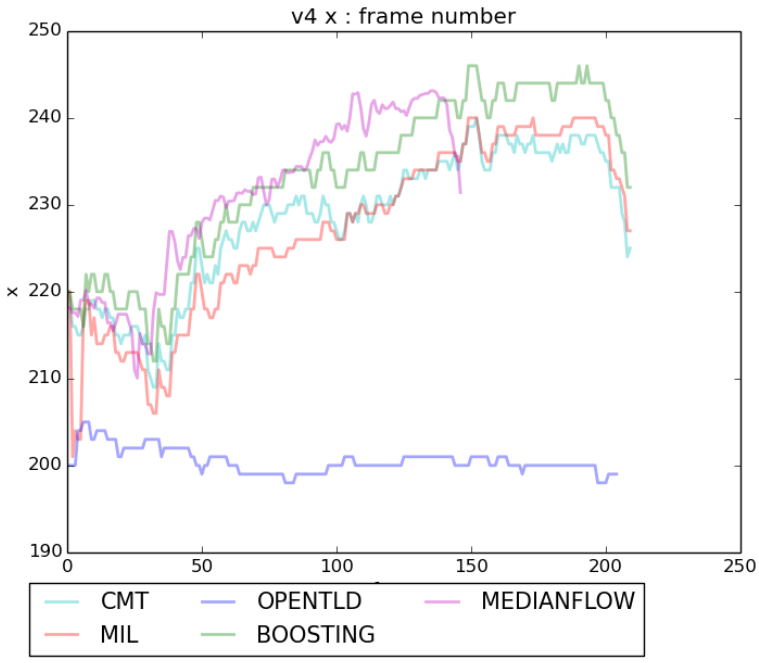


Figure 9.16 v4 raw tracking

Equalization makes the trajectories of all trackers but TLD more similar, while sharpening makes the trajectories of those other than TLD more similar to TLD, but not with a similar trend (Figure 9.17). Equalization combined with sharpening yields results that are similar to those of pure equalization.

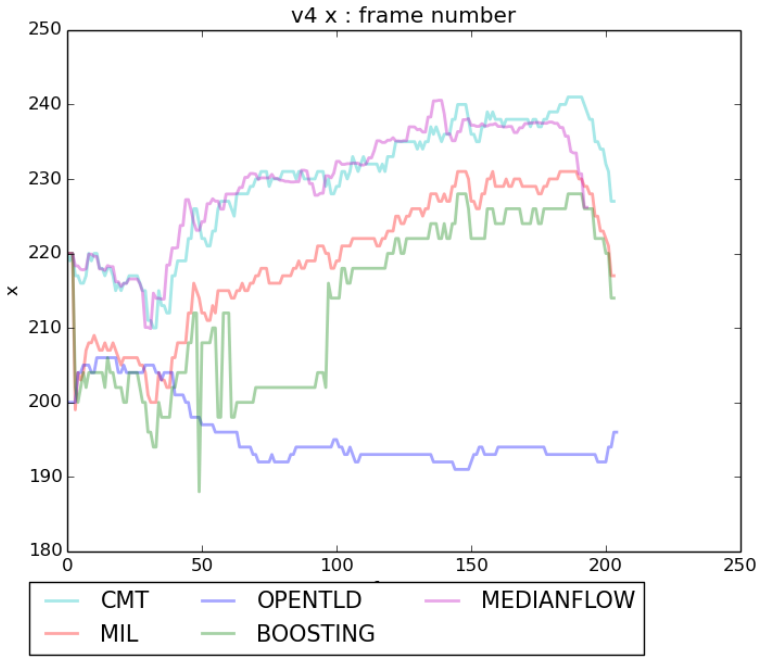


Figure 9.17 v4 sharpened tracking

v5 video demonstrates high texture and a rapid instrument introduction. The TLD tracker immediately fails on a raw video version. The other trackers perform well and with little differences (Figure 9.18).

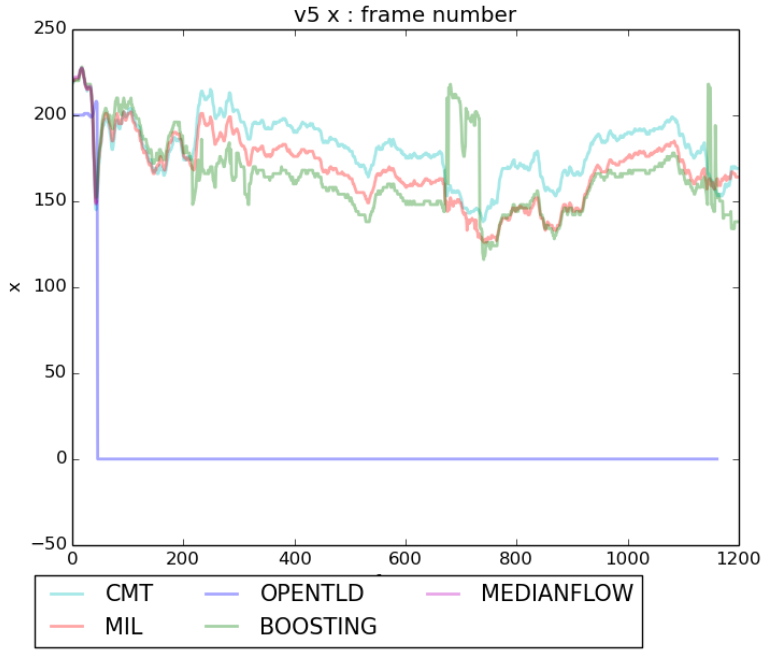


Figure 9.18 v5 raw tracking

Equalization leads to increased similarity between the trackers' results, but the TLD still fails. Sharpening, however, allows the TLD to stop failing and produce a robust trajectory (Figure 9.19). The same effect is achieved by applying equalizer and sharpener at the same time.

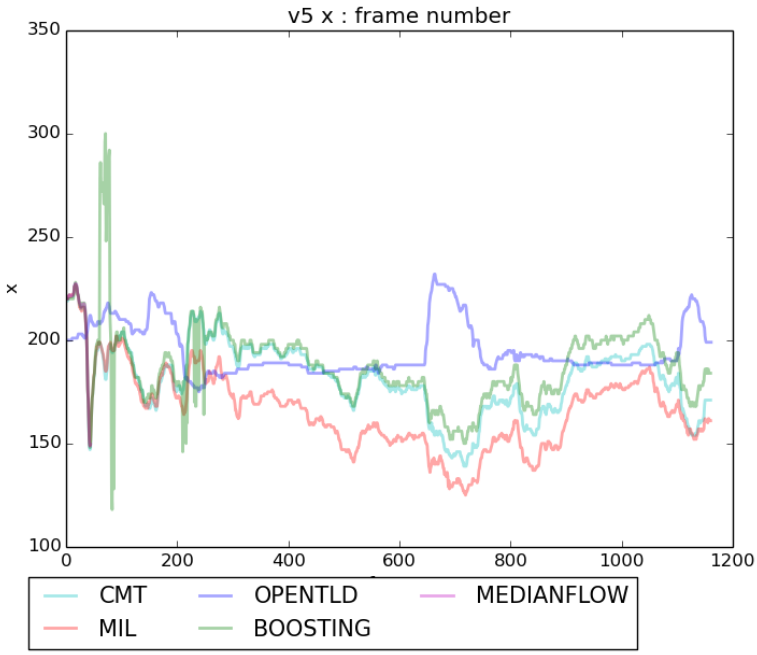


Figure 9.19 v5 sharpened tracking

v6 is a video that does not contain a lot of textures. Calm respiratory movements are present. The TLD tracker trajectory is slightly different from those retrieved by other trackers, but there are no dramatic differences. A TLD tracker briefly fails on a raw tracking sequence.

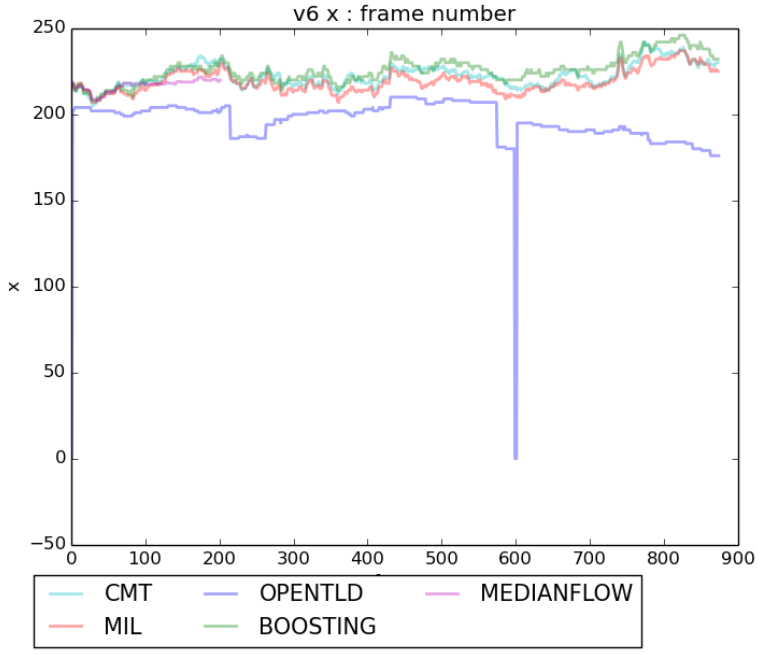


Figure 9.20 v6 raw tracking

Equalization removes the failure point introduced by the TLD tracker, while sharpening introduces one. The other trackers do not display significant differences across the preprocessed videos.

The data obtained after processing the videos was also used to observe dependencies within one tracker and several pre-processing modes.

Ada Boost tracker is seen to change the trajectory dramatically when used on a video with low texture and pre-processed by sharpening (Figure 9.21). In other cases, no significant differences in tracker behavior while different pre-processing modes were used.

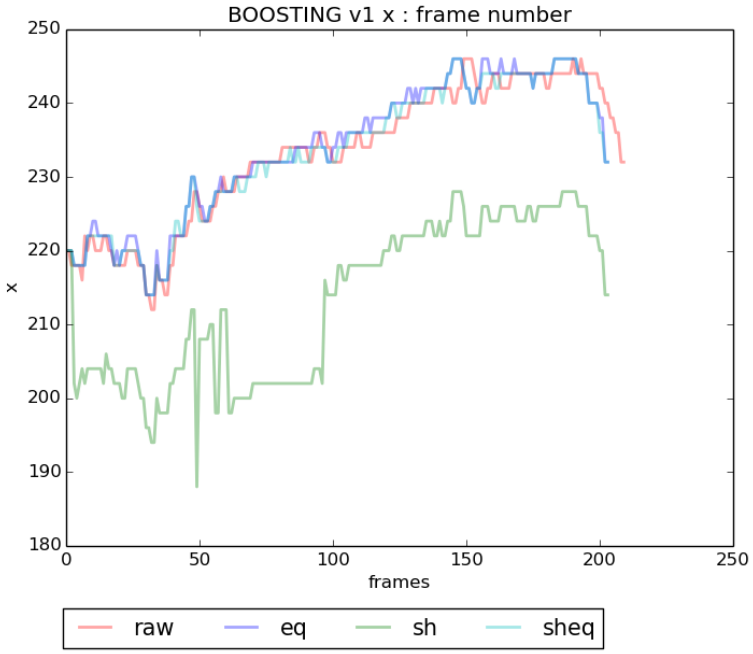


Figure 9.21 Ada Boost tracking v1

The **CMT** tracker does not display any difference in trajectory while processing any kind of enhanced video.

Median Flow yields trajectories that are improved greatly on low-texture slow motions when equalizer or sharpener and equalizer are used. Pre-processing helps the tracker recover from loss of object and yield the robust trajectory (Figure 9.22).

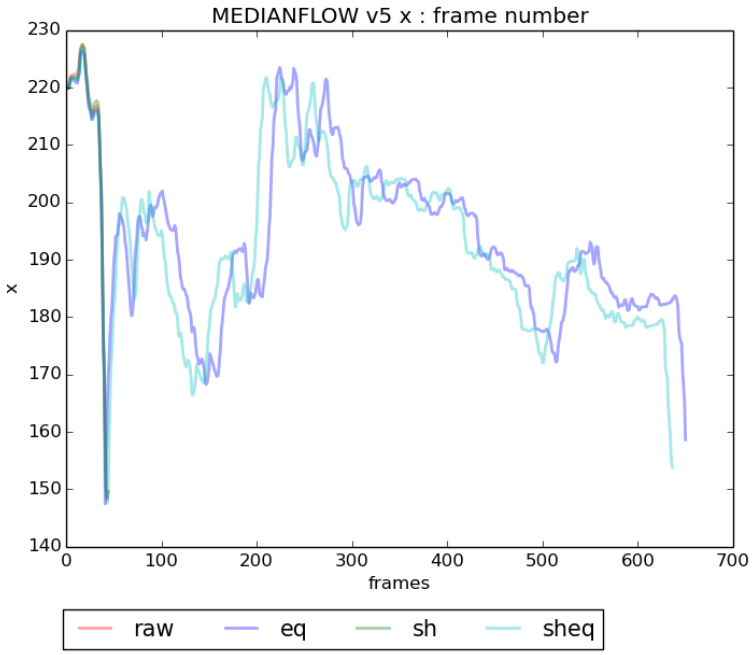


Figure 9.22 Median flow v5 tracking

MIL tracker shows improvement in all 3 pre-processing modes across several videos (e.g. Figure 9.23), but does not change the trajectory when it is already robust.

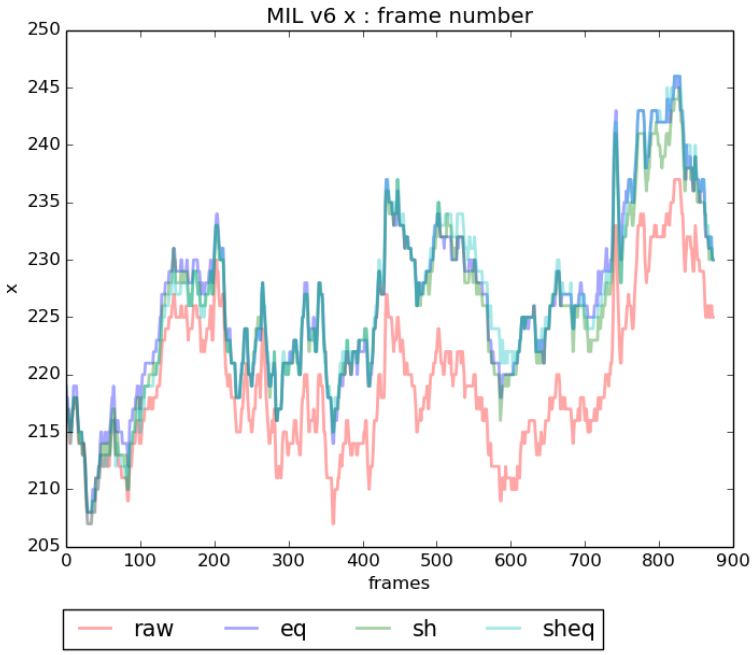


Figure 9.23 MIL v6 tracking

TLD's performance is demonstrated to be improved by different combinations of pre-processing techniques by wither removing a few tracking failures or providing complete recovering from severe failures (e.g. Figure 9.24).

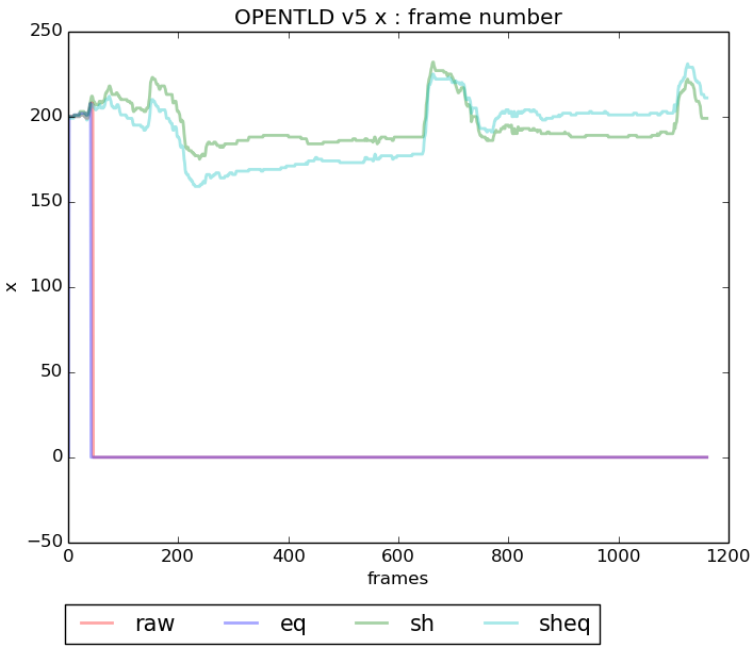


Figure 9.24 TLD v5 tracking

9.3. Summary

The application was tested performance-wise and the surgical residents reported that the system conforms to the system requirements specified earlier, while at the same time proposing further improvements to make the system more user-friendly.

The system remains experimental to a vast degree and is suitable for collecting user feedback and tracking statistics.

The analysis of the pre-processing techniques has proven that they do impact the quality of image analysis and can in many cases be used to enhance the tracking results.

Among the trackers tested, TLD has been proven to be more tolerable towards scenes with lower texture and subtle movements, while being weak towards fast movements (even in a highly texturized scene). The other trackers tend to behave similarly in different situations and fail tracking far less often than TLD does (in scenes with fast motion).

Sharpening and histogram equalization have been proven to affect the immediate results of tracking of all the trackers besides the CMT, which is, however, weak towards large numbers of features to track. Image sharpening has been demonstrated to increase the quality of tracking for TLD.

10. Conclusions and Future Work

10.1. Conclusions

The motivation behind this project was to propose and architecture and develop a prototype of a system for telestration in surgical telementoring that could be used as a test-bed for improvement of camera motion estimation and the telestration tool itself.

The aim of this work was to test effectiveness of conventional object trackers in a surgical context, estimate and compare their performance with regards to different surgical videos and various methods of image enhancement such as histogram equalization and image sharpening.

The system was successfully designed and implemented using C#(.NET/WPF)/Windows for the GUI and C++/Ubuntu for the stabilizer component. The system components communicate with each other using TCP sockets which means that the communication is reliable and the components may be replaced by newer versions or better alternatives as long as they conform to the same communication standard.

The system provides the functionality described in requirements specification and gives the user an opportunity to perform basic telestrating operations over a surgical video streamed from a source of choice (e.g. camera, video file).

The system's distributed nature and the way the two modules communicate with each other allows the user GUI module to be implemented for mobile platforms using either native applications or a browser-based video and telestration streaming using WebRTC. The fact that the stabilizer module that has to do all the heavy work can reside on a powerful computing device, means that the GUI-machine is no longer a computational bottleneck and tracking algorithms of great complexity can be used as long as the "stabilizer" node is fit for the task and the network bandwidth allows real-time video transmission.

In order to analyze the performance of a set of 5 trackers(TLD, CMT, MIL, Ada Boost, Median Flow), a data set of 6 laparoscopic surgery videos has been collected and processed by image enhancement techniques, resulting in total of 24 videos. The image enhancement techniques' performance has also been measured, indicating that the two techniques used in this work can be used in real-time without impairing the system's quality of service.

FAST and BRIEF feature detectors were used to compare the amounts of features detected in a raw and enhanced image. Enhanced images have shown dramatic boost in the number of features with image histogram equalization giving a much more significant boost than the image sharpening enhancement.

The FPS of trackers in question was calculated while estimating a position of camera in the whole data set. All the trackers but CMT displayed the FPS close to or significantly higher than the real-time FPS (24 fps).

The tracking quality has also been estimated based on the output from the trackers and the data set of 24 videos. The trackers' trajectory has been compared within each other

and with regards to the video being enhanced (pre-processed) or not. Plotting the latter relationship allows us to see the weak spots of certain trackers based that depend on the video property.

It has been discovered that image sharpening in some cases leads to significantly increased tracking results, even recovers the tracker from failure (TLD). In general, comparing the trajectories produced by trackers on raw and enhanced (pre-processed) videos has shown that the pre-processing methods presented in this work provide a boost in trajectory robustness. However, the trackers that do not have limits on the number of features that represent the tracked area, and the trackers that use more computationally heavy feature detectors or trackers, suffer a performance (FPS) loss because of a greatly increased amount of trackable features in enhanced videos.

10.2. Future Work

Future work would first of all mean introducing improvements to the system's usability. This means that it has to become less experimental and more ready to be deployed with ease.

Although the telestrations' position is adjusted based on the tracker, surgical videos contain movements from which it is not always possible to recover after a failure (e.g. extreme tissue folding, tissue removal, removing camera from the patient and introducing it back).

A tracker failure should be included into the application's routine instead of being an exception. It might be enough to hold a telestration in place while the mentee is observing the telestration and the tissue and is planning the next move. When the move (e.g. cutting, electrocuting) itself is performed, an instrument will obstruct the field of view and the tracker might fail. It may make sense to assign a new attribute to each telestration being created that would state how a telestration should behave in case of tracker failure.

Every telestration instance may be given a "time-to-live" after the failure has occurred and will fade away gradually, while residing in the position where it was last left by the stabilizer. The surgeon would then have less worries about the object tracking failures.

With the latter improvement in mind, some upgrades might be needed for a communication protocol.

Also, there are many more tracking algorithms including object trackers and SLAM-based camera motion estimators that can be ported for a suitable software platform or language in order to be tested within the surgical video setting.

Furthermore, the system could be extended to support mobile platforms while still relying on a "powerhouse" stabilizer module that would reside on a stationary PC. This would allow to test the system in a remote setting.

And finally, a more systematic user feedback from a wider audience could be collected in the future, which would allow to tailor the telestration tool itself for the needs of operating surgeons.

Appendix A Source code

See included file Source code.zip or

https://dl.dropboxusercontent.com/u/18799254/Source%20code.zip

The archive includes the GUI module implementation (VS 20013 C# project) and a C++ implementation of the stabilizer.

Appendix B Raw videos

Raw videos can be found on

https://dl.dropboxusercontent.com/u/18799254/Raw%20videos.zip

A set of raw videos and their pre-processed copies can be found on

https://dl.dropboxusercontent.com/u/18799254/All%20videos.zip

Appendix C Preprocessing scripts

See attached file “Preprocessing scripts.zip” or

https://dl.dropboxusercontent.com/u/18799254/Preprocessing%20scripts.zip

The attached file and the file in the URL specified above contain the scripts used for preprocessing of image sequences (histogram equalization, image sharpening, and applying the latter two at the same time).

After an image or a video is pre-processed, a prefix is appended in order to differentiate it from the raw version.

eq_ prefix is added to assets that have undergone histogram equalization.

sh_ prefix is added to assets that have undergone sharpening.

sh_eq_ prefix is added to assets that have undergone histogram equalization and sharpening after that.

Appendix D Video set and Naming Conventions

Entries of the Name column are clickable hyperlinks and lead to www.youtube.com.

The first prefix denotes the fragments' belonging to a certain video. The following pairs (triplets) of numbers denote the beginning and the end of an interval, from which a segment has been cut(mm-ss or hh-mm-ss). The trailing numbers stand for video resolution, and the extension is for the video encoding format.

Simple video name aliases are used throughout the work for better readability (Table 4). The data sets included may contain either simple aliases or aliases with time codes.

Name	Alias with time code	Simple alias
Laparoscopic left radical nephrectomy	y1_50-44_50-55_640x480.mp4	v3.mp4
Laparoscopic left radical nephrectomy	y1_51-04_51-08_640x480.mp4	v2.mp4
Laparoscopic left radical nephrectomy	y1_1-10-17_1-10-57_640x480.mp4	v5.mp4
Laparoscopic uterine fibroid surgery	y2_00-13_00-21_640x480.mp4	v1.mp4
Laparoscopic uterine fibroid surgery	y2_00-23_00-30_640x480.mp4	v4.mp4
Video provided by Etai Bogen (UNN)	el1_00-33-40_00-34-15.mp4	v6.mp4

Table 4 Video aliases

Appendix E Tracking plots

See attached file “Tracking plots.zip” or

https://dl.dropboxusercontent.com/u/18799254/Tracking%20plots.zip

The *contrib_output_tracker_video_plots* folder contains the plots created for every tracker/video combination, which integrate the results of raw videos tracking, and the pre-processed ones. These plots can be used to see if there is any difference in tracking a certain video by a certain tracker with or without a certain pre-processing method.

The *contrib_output_video_plots* folder contains the plots created for every video (raw and every kind of pre-processing separately) that integrate results for all trackers in one plot. These are good for comparing tracking results produced by different trackers for the same video.

In both folders, the vertical axis is either x or y coordinate of a bounding box, and the horizontal axis is the number of frame in the video.

References

1. Viola P, Jones M, editors. Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition, 2001 CVPR 2001 Proceedings of the 2001 IEEE Computer Society Conference on; 2001: IEEE.
2. Crow FC. Summed-area tables for texture mapping. ACM SIGGRAPH computer graphics. 1984;18(3):207-12.
3. Viswanathan DG. Features from Accelerated Segment Test (FAST). nd; 2009.
4. Lindeberg T. Scale invariant feature transform. Scholarpedia. 2012;7(5):10491.
5. Bay H, Tuytelaars T, Van Gool L. Surf: Speeded up robust features. Computer vision–ECCV 2006: Springer; 2006. p. 404-17.
6. Verri A, Poggio T. Motion field and optical flow: Qualitative properties. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 1989;11(5):490-8.
7. Hoetter M. Differential estimation of the global motion parameters zoom and pan. Signal Processing. 1989;16(3):249-65.
8. Optical flow [cited 2015 05.02.2015]. Available from: http://en.wikipedia.org/wiki/Optical_flow.
9. Lucas BD, Kanade T, editors. An iterative image registration technique with an application to stereo vision. IJCAI; 1981.
10. Lucas–Kanade method [cited 2015 06.02.2015]. Available from: http://en.wikipedia.org/wiki/Lucas%E2%80%93Kanade_method.
11. Baumberg A, editor Reliable feature matching across widely separated views. Computer Vision and Pattern Recognition, 2000 Proceedings IEEE Conference on; 2000: IEEE.
12. Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM. 1981;24(6):381-95.
13. Visual odometry [cited 2015 10.02.2015]. Available from: http://en.wikipedia.org/wiki/Visual_odometry.
14. Budrionis A, Augestad K, Patel H, Bellika J, editors. Towards Requirements for Telementoring Software. Proceedings of Scandinavian Conference on Health Informatics (SHI2012); 2012.
15. Bogen EM, Augestad KM, Patel HR, Lindsetmo R-O. Telementoring in education of laparoscopic surgeons: An emerging technology. World journal of gastrointestinal endoscopy. 2014;6(5):148.
16. Nebehay G, Pflugfelder R. Consensus-based Matching and Tracking of Keypoints for Object Tracking.
17. Leutenegger S, Chli M, Siegwart RY, editors. BRISK: Binary robust invariant scalable keypoints. Computer Vision (ICCV), 2011 IEEE International Conference on; 2011: IEEE.
18. Hamming RW. Error detecting and error correcting codes. Bell System Tech J. 1950;29:147--60.
19. Kalal Z, Mikolajczyk K, Matas J, editors. Forward-backward error: Automatic detection of tracking failures. Pattern Recognition (ICPR), 2010 20th International Conference on; 2010: IEEE.
20. Bouguet J. Pyramidal implementation of the lucas-kanade feature tracker: Description of the algorithm, opencv documentation. Santa Clara, CA: Intel Corp, Microprocessor Research Labs. 1999.
21. Nickels K, Hutchinson S. Estimating uncertainty in SSD-based feature tracking. Image and vision computing. 2002;20(1):47-58.
22. Shi J, Tomasi C, editors. Good features to track. Computer Vision and Pattern Recognition, 1994 Proceedings CVPR'94, 1994 IEEE Computer Society Conference on; 1994: IEEE.
23. Grabner H, Grabner M, Bischof H, editors. Real-Time Tracking via On-line Boosting. BMVC; 2006.
24. Hager GD, Belhumeur PN. Efficient region tracking with parametric models of geometry and illumination. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 1998;20(10):1025-39.
25. Comaniciu D, Meer P, editors. Mean shift analysis and applications. Computer Vision, 1999 The Proceedings of the Seventh IEEE International Conference on; 1999: IEEE.

26. Dalal N, Triggs B, editors. Histograms of oriented gradients for human detection. Computer Vision and Pattern Recognition, 2005 CVPR 2005 IEEE Computer Society Conference on; 2005: IEEE.
27. Porikli F, editor Integral histogram: A fast way to extract histograms in cartesian spaces. Computer Vision and Pattern Recognition, 2005 CVPR 2005 IEEE Computer Society Conference on; 2005: IEEE.
28. Levi K, Weiss Y, editors. Learning object detection from a small number of examples: the importance of good features. Computer Vision and Pattern Recognition, 2004 CVPR 2004 Proceedings of the 2004 IEEE Computer Society Conference on; 2004: IEEE.
29. Ojala T, Pietikainen M, Maenpaa T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 2002;24(7):971-87.
30. Kalal Z, Mikolajczyk K, Matas J. Tracking-learning-detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 2012;34(7):1409-22.
31. Bouguet J-Y. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. Intel Corporation. 2001;5:1-10.
32. Forster C, Pizzoli M, Scaramuzza D, editors. SVO: Fast Semi-Direct Monocular Visual Odometry. Proc IEEE Intl Conf on Robotics and Automation; 2014.
33. Thrun S, Leonard JJ. Simultaneous localization and mapping. Springer handbook of robotics: Springer; 2008. p. 871-89.
34. Unsharp mask [cited 2015 05.02.2015]. Available from: http://docs.gimp.org/en/plug-in-unsharp-mask.html.
35. Unsharp masking [cited 2015 05.02.2015]. Available from: http://en.wikipedia.org/wiki/Unsharp_masking.
36. Gaussian blur [cited 2015 05.02.2015]. Available from: http://en.wikipedia.org/wiki/Gaussian_blur.
37. Histograms: Construction, Analysis and Understanding. Available from: http://quarknet.fnal.gov/toolkits/ati/histograms.html.
38. Image histogram [cited 2015 06.02.2015]. Available from: http://en.wikipedia.org/wiki/Image_histogram.
39. Acharya T, Ray AK. Image Processing: Principles and Applications: Wiley; 2005.
40. Histograms [cited 2015 10.02.2015]. Available from: http://docs.opencv.org/modules/imgproc/doc/histograms.html?highlight=equalizehist#equalizehist.
41. Histogram Equalization [cited 2015 10.02.2015]. Available from: http://docs.opencv.org/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html.
42. Yang Q, Tang J, Ahuja N. Efficient and Robust Specular Highlight Removal. IEEE Transactions on Pattern Analysis & Machine Intelligence. 2014(1):1-.
43. Arnold M, Ghosh A, Ameling S, Lacey G. Automatic segmentation and inpainting of specular highlights for endoscopic imaging. Journal on Image and Video Processing. 2010;2010:9.
44. Ginhoux R, Gangloff J, de Mathelin M, Soler L, Sanchez MMA, Marescaux J, editors. Beating heart tracking in robotic surgery using 500 Hz visual servoing, model predictive control and an adaptive observer. Robotics and Automation, 2004 Proceedings ICRA'04 2004 IEEE International Conference on; 2004: IEEE.
45. Chai L, Hoff WA, Vincent T. Three-dimensional motion and structure estimation using inertial sensors and computer vision for augmented reality. Presence: Teleoperators and Virtual Environments. 2002;11(5):474-92.
46. Krupa A, Gangloff J, Doignon C, de Mathelin MF, Morel G, Leroy J, et al. Autonomous 3-D positioning of surgical instruments in robotized laparoscopic surgery using visual servoing. Robotics and Automation, IEEE Transactions on. 2003;19(5):842-53.
47. Aggarwal J, Nandhakumar N. On the computation of motion from sequences of images-a review. DTIC Document, 1988.

48. Wei G-Q, Arbter K, Hirzinger G. Real-time visual servoing for laparoscopic surgery. Controlling robot motion with color image segmentation. Engineering in Medicine and Biology Magazine, IEEE. 1997;16(1):40-5.
49. Uecker DR, Wang Y, Lee C, Wang Y. Laboratory investigation: Automated instrument tracking in robotically assisted laparoscopic surgery. Computer Aided Surgery. 1995;1(6):308-25.
50. Ortmaier T, Groger M, Boehm DH, Falk V, Hirzinger G. Motion estimation in beating heart surgery. Biomedical Engineering, IEEE Transactions on. 2005;52(10):1729-40.
51. Sauvée M, Noce A, Poignet P, Triboulet J, Dombre E. Three-dimensional heart motion estimation using endoscopic monocular vision system: From artificial landmarks to texture analysis. Biomedical Signal Processing and Control. 2007;2(3):199-207.
52. Teber D, Guven S, Simpfendörfer T, Baumhauer M, Güven EO, Yencilek F, et al. Augmented reality: a new tool to improve surgical accuracy during laparoscopic partial nephrectomy? Preliminary in vitro and in vivo results. European urology. 2009;56(2):332-8.
53. Bourger F, Doignon C, Zanne P, de Mathelin M, editors. A model-free vision-based robot control for minimally invasive surgery using esm tracking and pixels color selection. Robotics and Automation, 2007 IEEE International Conference on; 2007: IEEE.
54. Mirota DJ, Ishii M, Hager GD. Vision-based navigation in image-guided interventions. Annual review of biomedical engineering. 2011;13:297-319.
55. Chen S. Kalman filter for robot vision: a survey. Industrial Electronics, IEEE Transactions on. 2012;59(11):4409-20.
56. Cavusoglu MC, Rotella J, Newman WS, Choi S, Ustin J, Sastry SS, editors. Control algorithms for active relative motion cancelling for robotic assisted off-pump coronary artery bypass graft surgery. Advanced Robotics, 2005 ICAR'05 Proceedings, 12th International Conference on; 2005: IEEE.
57. Bader T, Wiedemann A, Roberts K, Hanebeck UD, editors. Model-based motion estimation of elastic surfaces for minimally invasive cardiac surgery. Robotics and Automation, 2007 IEEE International Conference on; 2007: IEEE.
58. Stoyanov D, Mylonas GP, Deligianni F, Darzi A, Yang GZ. Soft-tissue motion tracking and structure estimation for robotic assisted MIS procedures. Medical Image Computing and Computer-Assisted Intervention–MICCAI 2005: Springer; 2005. p. 139-46.
59. Baker S, Matthews I. Lucas-kanade 20 years on: A unifying framework. International journal of computer vision. 2004;56(3):221-55.
60. Hoff L, Elle OJ, Grimnes M, Halvorsen S, Alker HJ, Fosse E, editors. Measurements of heart motion using accelerometers. Engineering in Medicine and Biology Society, 2004 IEMBS'04 26th Annual International Conference of the IEEE; 2004: IEEE.
61. Groeger M, Ortmaier T, Sepp W, Hirzinger G, editors. Tracking local motion on the beating heart. Medical Imaging 2002; 2002: International Society for Optics and Photonics.
62. Lee S-L, Lerotic M, Vitiello V, Giannarou S, Kwok K-W, Visentini-Scarzanella M, et al. From medical images to minimally invasive intervention: computer assistance for robotic surgery. Computerized Medical Imaging and Graphics. 2010;34(1):33-45.
63. Noonan DP, Mountney P, Elson DS, Darzi A, Yang G-Z, editors. A stereoscopic fibroscope for camera motion and 3D depth recovery during minimally invasive surgery. Robotics and Automation, 2009 ICRA'09 IEEE International Conference on; 2009: IEEE.
64. Mountney P, Stoyanov D, Davison A, Yang G-Z. Simultaneous stereoscope localization and soft-tissue mapping for minimal invasive surgery. Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006: Springer; 2006. p. 347-54.
65. Kalman RE. A new approach to linear filtering and prediction problems. Journal of Fluids Engineering. 1960;82(1):35-45.
66. Mountney P, Lo B, Thiemjarus S, Stoyanov D, Zhong-Yang G. A probabilistic framework for tracking deformable soft tissue in minimally invasive surgery. Medical Image Computing and Computer-Assisted Intervention–MICCAI 2007: Springer; 2007. p. 34-41.
67. Mountney P, Yang G-Z. Soft tissue tracking for minimally invasive surgery: Learning local deformation online. Medical Image Computing and Computer-Assisted Intervention–MICCAI 2008: Springer; 2008. p. 364-72.

68. Mountney P, Yang G-Z, editors. Dynamic view expansion for minimally invasive surgery using simultaneous localization and mapping. Engineering in Medicine and Biology Society, 2009 EMBC 2009 Annual International Conference of the IEEE; 2009: IEEE.
69. Mountney P, Yang G-Z. Motion compensated SLAM for image guided surgery. Medical Image Computing and Computer-Assisted Intervention–MICCAI 2010: Springer; 2010. p. 496-504.
70. Wang H, Mirota D, Ishii M, Hager GD, editors. Robust motion estimation and structure recovery from endoscopic image sequences with an adaptive scale kernel consensus estimator. Computer Vision and Pattern Recognition, 2008 CVPR 2008 IEEE Conference on; 2008: IEEE.
71. Lowe DG. Distinctive image features from scale-invariant keypoints. International journal of computer vision. 2004;60(2):91-110.
72. Delponte E, Isgrò F, Odone F, Verri A. SVD-matching using SIFT features. Graphical models. 2006;68(5):415-31.
73. Hu M, Penney G, Figl M, Edwards P, Bello F, Casula R, et al. Reconstruction of a 3D surface from video that is robust to missing data and outliers: Application to minimally invasive surgery using stereo and mono endoscopes. Medical image analysis. 2012;16(3):597-611.
74. Mirota DJ, Wang H, Taylor RH, Ishii M, Gallia GL, Hager GD. A system for video-based navigation for endoscopic endonasal skull base surgery. Medical Imaging, IEEE Transactions on. 2012;31(4):963-76.
75. Yip MC, Lowe DG, Salcudean SE, Rohling RN, Nguan CY. Tissue tracking and registration for image-guided surgery. Medical Imaging, IEEE Transactions on. 2012;31(11):2169-82.
76. Agrawal M, Konolige K, Blas MR. Censure: Center surround extremas for realtime feature detection and matching. Computer Vision–ECCV 2008: Springer; 2008. p. 102-15.
77. Calonder M, Lepetit V, Strecha C, Fua P. Brief: Binary robust independent elementary features. Computer Vision–ECCV 2010: Springer; 2010. p. 778-92.
78. Lourenço M, Stoyanov D, Barreto JP. Visual Odometry in Stereo Endoscopy by Using PEaRL to Handle Partial Scene Deformation. Augmented Environments for Computer-Assisted Interventions: Springer; 2014. p. 33-40.
79. Grasa OG, Bernal E, Casado S, Gil I, Montiel J. Visual SLAM for handheld monocular endoscope. Medical Imaging, IEEE Transactions on. 2014;33(1):135-46.
80. Chang P-L, Handa A, Davison AJ, Stoyanov D. Robust real-time visual odometry for stereo endoscopy using dense quadrifocal tracking. Information Processing in Computer-Assisted Interventions: Springer; 2014. p. 11-20.
81. Moore R, Adams J, Partin A, Docimo S, Kavoussi L. Telementoring of laparoscopic procedures. Surgical endoscopy. 1996;10(2):107-10.
82. Schulam PG, Docimo SG, Cadeddu JA, Saleh W, Brietenbach C, Moore RG, et al., editors. Feasibility of laparascopic telesurgery. CVRMed-MRCAS'97; 1997: Springer.
83. Wachs JP, Gomez G. Telementoring systems in the operating room: a new approach in medical training. Medicina. 2013;73(6):539-42.
84. Augestad KM, Bellika JG, Budrionis A, Chomutare T, Lindsetmo RO, Patel H, et al. Surgical telementoring in knowledge translation--clinical outcomes and educational benefits: a comprehensive review. Surgical innovation. 2013;20(3):273-81.
85. Alverson DC, Bowyer M, Darzi A, FMedSci H, Denstedt JD, Dev P, et al. Surgical Telementoring News. 2014.
86. Budrionis A, Augestad KM, Patel HR, Bellika JG. An evaluation framework for defining the contributions of telestration in surgical telementoring. Interactive journal of medical research. 2013;2(2).
87. Santomauro M, Reina GA, Stroup SP, James O. Telementoring in robotic surgery. Current opinion in urology. 2013;23(2):141-5.
88. Budrionis A, Augestad KM, Bellika JG, editors. Telementoring as a Service. Scandinavian Conference on Health Informatics 2013; 2013.
89. Budrionis A, Augestad KM, Bellika JG, editors. Telestration in Mobile Telementoring. eTELEMED 2013, The Fifth International Conference on eHealth, Telemedicine, and Social Medicine; 2013.

90. Comer DE, Gries D, Mulder MC, Tucker A, Turner AJ, Young PR, et al. Computing as a discipline. Communications of the ACM. 1989;32(1):9-23.
91. Robertson J, Robertson S. Volere. Requirements Specification Templates. 2000.