3D-guided Biopsy

Jasper Dijt, Alexandros Karalidis, Haluk Sahin
{J.W.Dijt, A.Karalidis, H.Sahin}@student.tudelft.nl

July 15, 2014

Abstract

Systems providing assistance during a biopsy or an autopsy can be found in various medical centers and hospitals, today. Traditionally, CT(computed tomography) and MRI(magnetic resonance imaging) scans are used to produce detailed 3D models and offer support during this procedures. Unfortunately, the high doses of radiation which are involved during a CT scan make the procedure less favorable for a continuous operation.

At the Erasmus MC the departments of Pathology, Radiology and Bio-informatics are working on a prototype for guiding biopsies in a Minimal Invasive Autopsy (MIA) procedure using a tracking camera.

For this prototype, an existing piece of software for rendering and manipulating data from CT and MRI scans, V-Scope, developed at the Erasmus MC has been extended to provide the needed functionality for this.

In this report, we describe the design, implementation and testing of this prototype as a project for our bachelor thesis. Furthermore, we provide recommendations for further enhancements of the prototype.

Contents

1 Introduction

At the Erasmus MC the departments of Radiology, Pathology and Bio-informatics are working together on a prototype guidance system, for taking tissue samples with a biopsy needle during a MIA procedure. A MIA is an autopsy procedure that involves making several detailed scans (e.g. MRI, CT) of a cadaver and taking tissue samples of specific organs and other areas of interest, with a biopsy needle. For this project, Anton Koning of the Bio-informatics department was our primary contact at the Erasmus MC. Additionally, Ivo Wagensveld was our contact at the department of Pathology that will use and evaluate the finished prototype.

In this report, we describe our project for creating and refining this prototype. This prototype has been build by extending the existing software, at the Erasmus MC. We have build a working prototype that can be used to track a biopsy needle using an optical tracker. In this prototype, the needle is tracked and rendered within a 3D model of the real world target. Testing proved that our tracking algorithm is accurate enough to reliably hit targets with a size of at least 0.5cm.

The structure of this report is as follows: In Section 2, a problem analysis is presented with an introduction to the existing software, followed by a detailed description of the problem. Section 3 describes the research that has been done in the beginning of the project and the theoretical background on which we have based many of our choices, during this project. In Section 4, the design of our extension is presented followed by a detailed description of the implementation, in Section 5. The testing of various parts of the product, including the testing of the final product as a whole, is discussed in Section 6. Section 7 describes the analysis of the final product. Section 8 presents recommendations for future enhancements. Finally, section 9 outlines the process of the project during this bachelor project. The report is concluded with section 10.

2 Problem Analysis

Currently, a CT-guided procedure is used for taking biopsies during a MIA at the Erasmus MC. In this procedure, the location of the needle is checked with additional CT-scans. The main disadvantages of this approach are that the additional CT-scans are time-consuming and that the metallic needle introduces artifacts (e.i. noise), in the each scan. This effect is known as beam hardening [1]. The consequence is that small targets may not be distinguished in the scan, due to the presence of the needle.

In the 3D guided biopsy prototype, a single 3D model of the subject, based on a CT-scan, will be used for guiding the needle to the target. By using an optical tracker, the needle will be followed and its orientation and tip will be displayed,

within the model, on a 3D computer's screen. It is expected that the 3D guided biopsying procedure will be faster, easier to perform and will provide more accurate results than the current process used.

We will first review the existing software and the prototype at the Erasmus MC. Afterwards, an overview of the problem which our group has been assigned to solve, will be provided.

2.1 Existing Software & Prototype

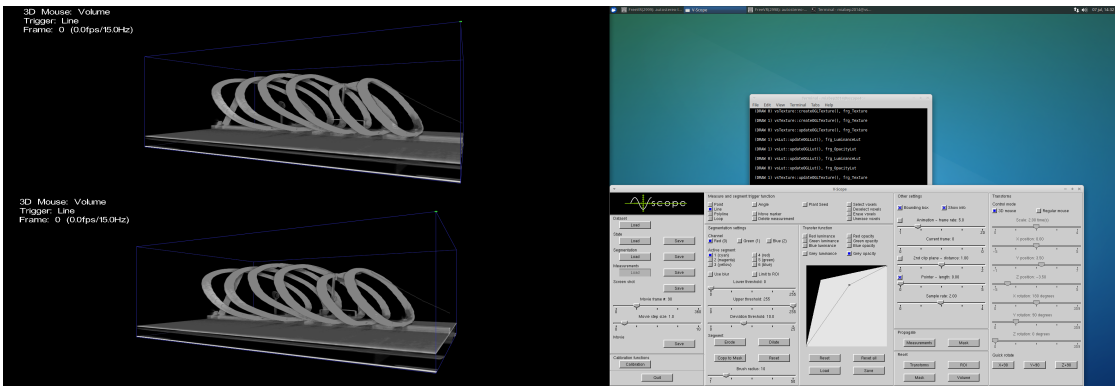

Figure 1: V-Scope: left screen is for a stereoscopic monitor.

The Bio-informatics department at the Erasmus MC already has software in place, for rendering and manipulating data from 3D scans. Many sources of 3D data can be used, such as CT, MRI and 3D ultrasound. The program used for this, V-Scope, is primarily written and maintained by our supervisor at the Erasmus MC, Anton Koning.

V-Scope has initially been developed to operate in a virtual reality environment, similar to CAVE. An example of such an environment is the I-Space, in the Bio-informatics department[1]. However, a version to be used on desktop computers with a stereoscopic 3D monitor, also exists. V-Scope and its associated utility programs can be used to convert scanned data to 3D volumes, render these volumes and manipulate them. A more extensive review of V-Scope can be found in our Research report (Appendix B).

Additionally, a preliminary prototype exists for tracking a biopsy needle. Unfortunately, this prototype often fails to calibrate correctly and this makes the tracking of the needle impossible.

[1] `http://www.erasmusmc.nl/amie/meet-scientists/anton-koning/`

2.2 Problem Description

The main goal of our project is to create a working prototype that can be tested in an actual MIA. To achieve this, it is vital to make the calibration for the tracking of the needle operational.

Once the calibration can be performed, the accuracy of the tracking can be evaluated with the help of a phantom[2]. If the tracking is not accurate enough, we will attempt to improve it within the scope of our project, as defined in the work plan (Appendix A). In addition, we will improve and refine the whole prototype and make it ready for testing during a real MIA procedure. The desired improvements are primarily based on the feedback of Anton Koning and Ivo Wagensveld.

2.3 Planning

This project has been executed in several phases. Firstly, a 2 to 3 weeks orientation & design phase. In this period, the research report has been produced. We have also become familiar with the problem space and we have selected several tools to use during the implementation (e.g. Issue tracking and source control). Second a 6 to 7 weeks implementation phase. During this period, we implemented the chosen approach and integrated this into the existing software at the Erasmus MC. Lastly, a finalization phase in which we produced the final report and made our final adjustments to the system. More information concerning the planning of the project can be found in our work plan (Appendix A).

3 Research and Theoretical Background

During the first weeks of this project, a research report has been produced which can be found in the Appendix B. In this section, the most important subjects described in the research report will be highlighted.

3.1 V-Scope

For our assignment, V-Scope will have to be modified and extended. An important challenge during this operation is that the existing code is poorly documented: there are few comments and not much documentation available. Therefore, an analysis of the current code-base, the dependencies between the various modules and the functionality provided by these modules, has been made.

[2]The subject is represented by a predefined object for which a scan (e.g. data for a 3D model) is available . See fig. 13

The analysis has been made down to the module level but not all classes have been documented. This choice has been made because providing a full and correct description of the existing class hierarchy would have taken a significant amount of time. In practice, the global overview and analysis proved to be very helpful to us, in finding useful modules and classes in the existing code-base.

3.2 Similar Systems & Techniques

To get a better understanding of the problem domain, we have made a short overview of other projects concerning the tracking of such instruments. We have found several examples in the literature, of tracking being used in neurosurgery. We have also read about various methods which are used to implement such a tracking.

In addition to this overview, we visited the Erasmus MC mortuary to look at a tracking device, used for doing brain-biopsies. We also visited the CT-scanner room which is the place in which the system will be used, during a MIA procedure.

The detailed overview and the visit to the mortuary have given us a good insight into how other people approach this problem. In addition, the visit to the CT-scanner room and the discussion with Ivo Wagensveld have given us a good indication of of challenges, we might encounter. For instance, the camera position and the visibility of the needle-marker are especially challenging, in case of obese subjects

3.3 Matching & Calibration

The most important aspect of the software is the calibration, since the tracking is impossible without it. For this reason, an in-depth literature review on the topic rigid-body point-based registration has been made. From this, the relevance of our problem with the so called absolute orientation problem has been revealed. Therefore, we have made a list of possible solutions to this problem. The solution-approaches that have been primarily taken into consideration are: Horn's algorithm [2], Thompson's approach [5], Oswal's - Balasubramanian's method [3] and Schut's approach [4].

Afterwards, the most suitable approach for our case has been selected and argumentation has been provided, concerning this choice. We have decided that the approach of Horn for solving the absolute orientation problem, was the best choice in our case. Horn provides a closed form solution that works for any number of point pairs, greater than three. The other approaches are iterative in nature or cannot use more than three points to get a solution. The algorithm calculates the transformation using unit quaternions, which can give the orthonormal matrices

that contain the transformation. More information concerning this algorithm and its implementation can be found in the Appendix B.

3.4 Implementation details

Finally, we have presented the supporting tools, the programming language and the libraries which have been used in this project.

Some choices had already been made for us, such as the choice of the programming language (C++). In other areas, we were free to choose our own approach. For example, we have chosen to use the Eigen library for many operations in the matching algorithm. These operation were related to linear algebra. Furthermore, Git has been selected as our source control program. Finally, a set of guidelines concerning the coding style has been chosen. This is mainly based on the Google C++ style guide.

4 Design

In this section, we take a closer look at the design of our extension to V-Scope. Firstly, the relevant use cases and the work-flow of the calibration will be discussed. Secondly, the specifications & features of the tracking camera and how these influence the design will be discussed. Furthermore, a technical design will be given which highlights where in the V-Scope's ecosystem our code will be placed.

4.1 Use Cases

There are not many distinct use cases for our extensions because it is designed to assist a specific procedure, namely the guidance of biopsies with a biopsy needle. However, the user has to perform several steps in order to get the actual tracking. The use-case diagrams and state-diagram below give a good insight into the different tasks, which the user has to perform.

The first use-case relates to the preparation of the CT-scan data in order to be used, and the initiation of the V-Scope program.. The second one relates to the calibration of the system and the actual biopsy's procedure. The third figure is a state diagram illustrating the work-flow for (re)calibrating the tracking of the needle.

4.1.1 Converting & Importing Scan Data

The first use case (fig. 2) is for starting V-Scope and importing the data from the CT-scanner. When the system is used on an actual subject, this will be done in the

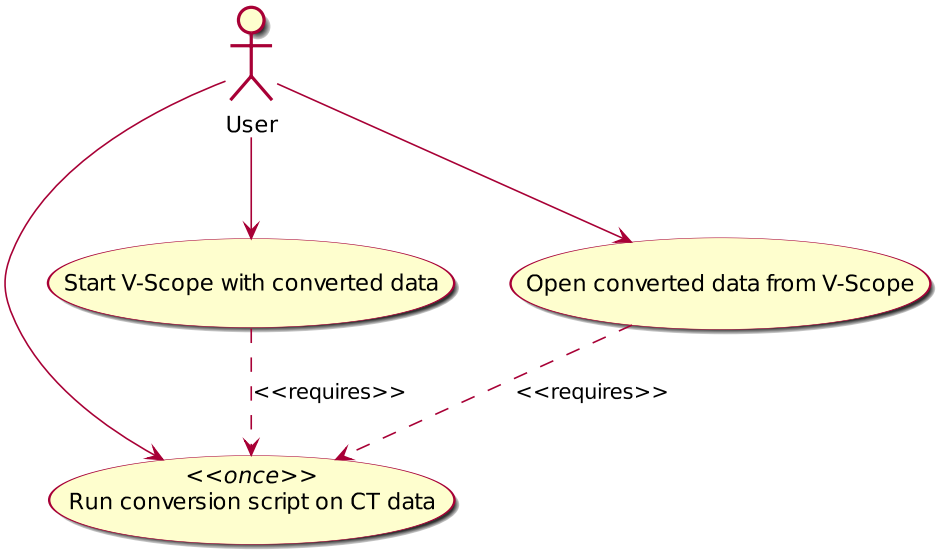

Figure 2: V-Scope start & CT Data import

same room as the CT-scanner. A scan of the subject will be made, which results in a large 'stack' of 2D images, representing slices of the subject. This stack of images must be converted to a 3D volume before it can be loaded into V-Scope. A script for performing this conversion is available at the Erasmus MC. After the conversion, the 3D volume can be loaded by starting V-Scope with the corresponding data set (i.e. "Open file with program") or by opening it from a dialogue, within V-Scope.

4.1.2 Calibrating, Tracking & Performing Biopsies

This use case (fig. 3) displays the tasks related to the tracking of the needle, and to the tissue-sampling. To track the needle, the system must be calibrated first. A calibration can be performed in two ways, a full calibration or a recalibration. A recalibration is a faster way of calibrating that can be used after an initial full calibration. This facilitates the moving of the camera, in case which the needle is not in the field of the camera's vision during the MIA procedure. The work-flow for calibrating the system will be discussed, in depth, in the following section.

After the calibration, the tracking of the needle can be enabled and used to perform biopsies. V-Scope already contains some useful features to manipulate 3D models (e.g. erase parts of them) that makes it more easy to view the needle on screen.

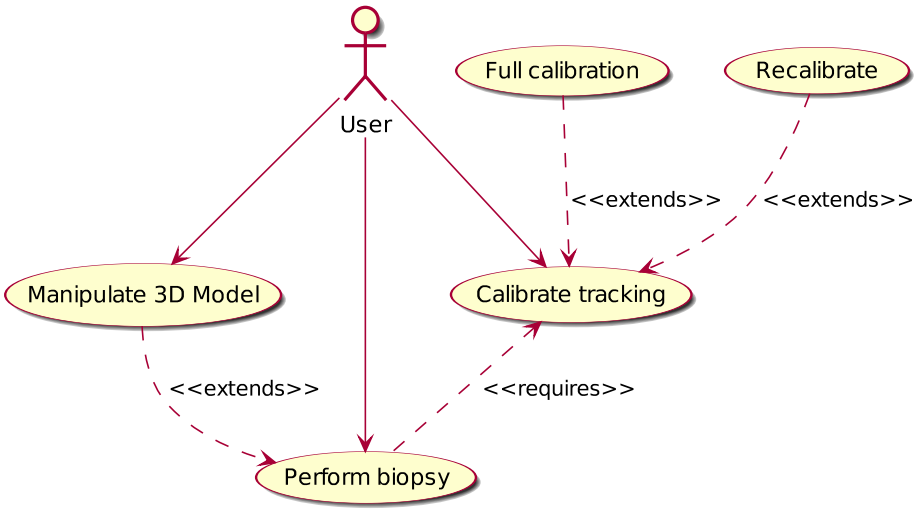

Figure 3: Tracking & Calibration

4.1.3 Calibration workflow

The calibration of the system follows a specific workf-low which is represented by the state machine in fig. 4. A breakdown per top level state:

- **Calibrating**
  In this state the system is calibrated. Calibration is performed by the user. He/she will select a point in the rendered volume and press a button. Afterwards, he/she points out the corresponding point in the actual subject with a traceable pointer. Examples of points, also called landmarks, which can be used in the human body are the hip bones and the sternum.

  This is repeated until at least three pairs of points have been entered. After the third pair of points, sufficient information has been provided to calibrate the system and start tracking

- **Tracking**
  In this state the needle is tracked and rendered on screen, showing its position in the 3D volume.

- **Recalibrating**
  When the calibration is no longer valid (e.g. due to a movement of the camera or to a change to the subject's position), the user can initialize a recalibration. This is a shortened calibration procedure in which only the points on the actual subject have to be pointed out again. This is possible because the points in the rendered volume will not have moved and can be reused.

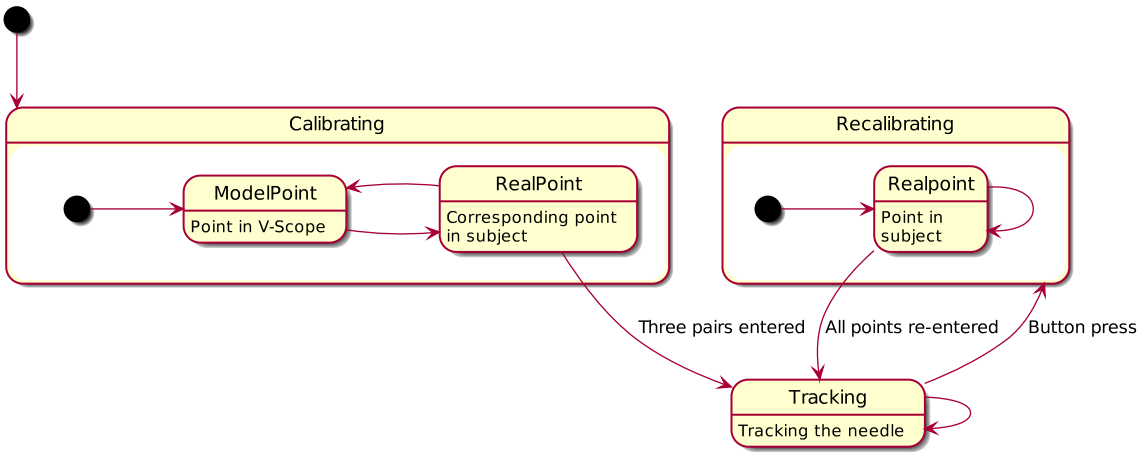

Figure 4: Calibration state diagram

Once the new position of all points on the subject is known the system is re-calibrated and the needle can be tracked again.

4.2 Tracking camera

Figure 5: SMARTTRACK camera

For this project an Advanced Realtime Tracking (ART) SMARTTRACK[3] (fig. 5) camera will be used for tracking the markers. The SMARTTRACK communicates with the computer via Ethernet. In this section the camera's specifications, configuration-software and features will be discussed.

4.2.1 Specifications

The SMARTTRACK is a tracking device that uses infra-red cameras to track markers. It can track a maximum of 4 markers, up to 2.5 meters away from the

[3] http://www.ar-tracking.com/products/tracking-systems/smarttrack/

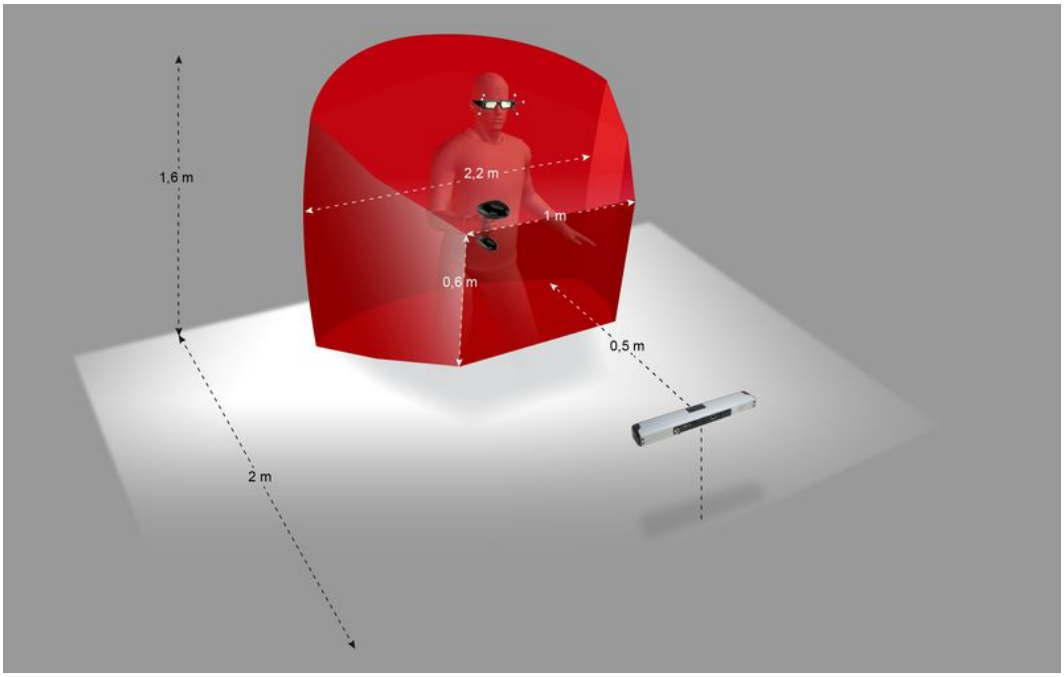

Figure 6: SMARTTRACK tracking volume

camera, see fig. 6 for a visualization of this.

The position of the markers is given as a translation vector that gives the position of the marker relative to the tracker in mm, and a 3x3 rotation matrix describing the oriëntation of the marker. Occasionally, distinct objects outside the tracking volume (i.e. further than 2.5m away) but in the field of view, can be tracked. However, this usually comes with a considerable accuracy-downfall.

4.2.2 Software

Delivered with the camera is a program called DTrack2[4] for calibrating new markers and reading camera data. In addition to this software a Software Development Kit (SDK) is available, this SDK can be used to communicate with the camera and retrieve marker positions.

An important feature of the SMARTTRACK camera we are using, is the 'measurement tool' feature. This is an extended calibration function that allows the user to calibrate the camera to return the location of the tip of the tool, instead of the location of the attached marker. The tip calibration procedure is easy to perform and indicates the residual error after the calibration.

Due to the additional expense of the license for this feature we performed a

[4] `http://www.ar-tracking.com/products/software/dtrack2/`

comparison between this and statically configuring the translation from marker to tip. This was possible because ART provided us with a temporary testing license for the duration of the project.

The result of this comparison was that both methods work for getting the location of the tip. However the measurement tool functionality proved to be significantly more reliable in terms of accuracy and the detection of measurement errors.

Its easy calibration method also allows for quickly setting up new configurations, e.g. a different needle, a different marker, etc. and may streamline further tests performed with the prototype at the Erasmus MC in the future. Therefore we built the system to work with the measurement tool feature.

From a technical standpoint this feature also has another welcome effect. The calibration changes the internal axis system of the marker to the oriëntation displayed in fig. 7. The specific rules for this are as follows:

1. The tip is the origin of the system.
2. The z-axis goes from the marker furthest away from the tip to the tip.
3. The z/y plane is then defined by the marker closest to the tip.

As the coordinates are right handed this gives enough information for determining the x axis. For normal bodies a different set of rules is used based on the distance between individual balls on the marker.

The consequence of this is, that if the marker furthest away from the tip is positioned in line with the needle, then so is the -Z axis. This simplifies finding the direction of the pointer / needle relative to the axis system of the marker for the configuration of new tools, especially compared to using normal tracked bodies where one first has to determine the orientation of the axis system.

Figure 7: Axis system after calibration

4.3 Technical design

The technical design choices we made during this project will be discussed in four parts. First the matching algorithm. Then the standalone calibration test tool we developed, to test the algorithm before moving to integration with V-Scope. After that the design of the extension written for the Simple Virtual Reality Daemon (SVRD), the daemon reading data from the tracker and finally the integration with V-Scope.

4.3.1 Matching algorithm

The matching algorithm which is used to map points from the tracking camera to the scanned (3D) model is an implementation of Horn's solution for the absolute orientation problem [2]. The C++ template library Eigen is used for performing the linear algebra operations that are required for this. Given coordinates of points (x,y,z) measured in two different Cartesian coordinate systems, the algorithm calculates the transformation between these systems. After the transformation, consisting of the scale s, the 3x3 rotation matrix R and the translation vector t, has calculated it is returned so that the caller can reapply it to new data received from the tracking camera.

This transformation from a point from the camera p_r to a point in the volume p_v is performed as follows:

$$p_v = R(p_r * s) + t$$

It is important to note that the algorithm can only handle uniform scaling, i.e. the scale factor must be the same in all directions.

This algorithm is then compiled into a static library that can be linked with other C/C++ programs. This library exports only one symbol: `computeTransform`. This is the function that computes the transformation between the camera and the model coördinate systems. The choice for a separate library was made because the algorithm code was to be used in more than one program, namely the standalone calibration and later V-Scope itself.

4.3.2 Standalone Calibration

The standalone calibration tool is built as a small program that facilitates testing the calibration in a very simple setup. It was developed to test the matching algorithm with the camera without spending a lot of time already integrating the matcher with V-Scope. It takes a list of points from a file, representing points measured in the model. After that the user can measure the same number of points with the camera. Then the program performs a calibration and displays the position of the pointer in the terminal.

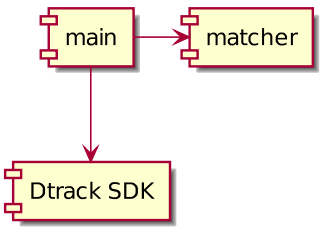

Figure 8: Standalone calibration Component diagram

As can be seen in fig. 8 the standalone implementation does not use SVRD for getting the camera points. Rather, it uses the Dtrack SDK for this. The reason for this choice was because using the camera SDK allowed us to get a better feeling of the possibilities for using this SDK and the camera. In addition to that the SDK is almost as easy to use as SVRD.

4.3.3 Extending SVRD

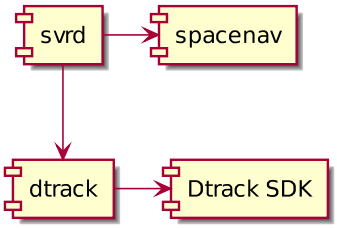

Figure 9: SVRD Component diagram

Before the project SVRD could read only one marker position from the camera. In our case, we need to read at least two. One for the needle, and one for the pointer used in the calibration. An additional constraint is that SVRD has to remain compatible with older implementations of v-scope.

Luckily the existing data structure used for sharing input data from SVRD with V-Scope has room for extensions. SVRD uses a four item long array of a generic type(fig. 10) for passing on input data in shared memory. The existing implementation only uses the first two elements leaving the last two free for us to use.

Therefore we extended SVRD to allow for the tracking of up to three markers using the SMARTTRACK, taking all available space in the array.

```
typedef struct
{
        double valuator[16];
        int button[32];
} svrInput;
```

Figure 10: SVRD Input data struct

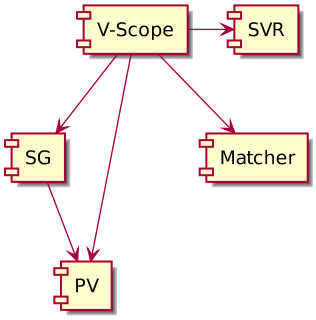

Figure 11: V-Scope component diagram

4.3.4 V-Scope integration

The final task for our group was integrating the matcher with V-Scope. As there are already several special versions of V-Scope in the source directory, each with a separate main source file that links with the rest of the V-Scope object.

We chose to integrate our code as another version of V-Scope, therefore we cloned the basic desktop version file and worked in that clone. This has the benefit of keeping the original desktop version intact. Additionally we put almost all of the code we have written for the integration in the new file. Only where this was illogical or V-Scope provided an alternative location we deviated from this. This, combined with the fact that most of the additional logic was placed in separate functions that hooked into the V-Scope event loop will make it easier to keep track of changes and differences between the original desktop version and our version for future developers.

5 Implementation

In this section we will discuss the implementation phase, the additional design choices made in this period and the challenges encountered.

This will be split up in several parts. First, the setting of the working environment will be discussed. In the existing code base the system is build using the,

primarily command line, tool Make. Not all group members prefer to work from the terminal, so this was a small challenge we had to deal with. Second, the implementation of the matching algorithm is discussed. Third the final integration into V-Scope and the workflow / user interface modifications will be discussed. This is also, roughly, the chronological order in which these items have been implemented.

5.1 Setting of a Working Environment

The working environment consisted of Eclipse, an IDE, in combination with some predefined makefiles. The existing system was build by using a terminal and running those makefiles from there. An alternative was to allow Eclipse to automatic generate makefiles for imported project. The latter choice was found less preferable because the old system (prototype) already had several makefiles to be used. Besides, one of our tasks was to expand and maintain these makefiles when needed.

By using Eclipse the project's overview becomes clearer and potential changes can easier be made. Moreover, using git for version control is well integrated with Eclipse. By installing the appropriate plugins (i.e. EGit) group members less experienced with the terminal could resolve merge conflicts as easily as those having more experience working in the terminal, for instance. Finally, we have exploited other useful tools and features of Eclipse (such as spell-check while typing) which have offered us assistance during the development.

5.2 Matching Algorithm

Not many challenges were encountered during the implementation of the matching algorithm. This is mainly due the fact that a thorough literature search had taken place beforehand. In addition, a detailed design of how to efficiently implement and test our solutions has been developed.

The collected literature offered a manner to implement this functionality and the Eigen package provided some extra operations for efficient developing (e.g. quaternions). Initially the result was initially returned using matrix and vector types from Eigen. After testing in the standalone calibration tool this was rewritten to returning a plain double array. This was done because double arrays are what is used in V-Scope and this prevents us from having to perform the conversion from Eigen matrices to plain arrays in V-Scope. It also keeps all references to the Eigen package within the matcher code.

5.3 Integration

Once the standalone version of the matching algorithm worked and was proven accurate enough we integrated it with V-Scope. The main challenge with the

integration was the existing code base we had to work with. Especially the way V-Scope handles matrices internally caused us some headaches. With both the integration of the algorithm and the implementation of the use cases and calibration workflow this cost a lot of time.

Another problem was that adding support for the additional markers to SVRD increased the time it took to update other input devices SVRD polls as well, such as the 3D mouse. This was quite noticeable when interacting with V-Scope, causing major delays in the responsiveness. The solution implemented for this is to poll the camera less often in SVRD. This restored the latency to the low level needed for the 3D mouse and did not make the response time of the camera too slow.

5.3.1 User interface

The process is primarily controlled using the buttons of the 3D mouse. There are many utility buttons that V-Scope does not use on these devices, something we took advantage of, an overview of the buttons and their functions can be viewed within V-Scope.

The Erasmus MC also made available two new 3D mice for us to use, these were of a newer type with a different key map which made them incompatible with SVRD. We extended SVRD to work with the new 3D mice as well. This was time consuming, because we needed to extract the key & event codes sent by the new mouse.

5.4 SIG Feedback

During this project our code was submitted to the Software Improvement Group (SIG) two times, once roughly midway during the implementation and another time at the end. The SIG rates the maintainability and quality of the code we have produced on a scale of 1 to 5 stars (more stars meaning better maintainability).

5.4.1 First submission

In the initial submission our code got 4 out of 5 stars. Because some files where mostly existing V-Scope code and not our own work not all source code was submitted to the SIG. Eric Bouwers from the SIG noted that this makes it more difficult to give specific recommendations for our code base. The following are specific items pointed out by the SIG for us to take into account:

- There is LGPL code in the 'tracker/sdk' directory. It is unclear whether this code as been edited or is there for reference purposes. This should be documented and/or removed to prevent confusion for future developers.

- There is a lower score for Unit Size and Unit Complexity, meaning that some parts of our code contain very long and/or complex functions. These functions should be split up into smaller and less complex functions. As an example of this a function in the matching algorithm implementation is named. According to Eric Bouwers from the SIG a reduction in Unit Size generally also creates a reduction in Unit Complexity.
- There are no unit-tests for the code base. It is strongly recommended to at least make tests for the most important functionalities of the program.

Based on these recommendations we made the following changes.

- Added a readme to the tracker/sdk directory, explaining the purpose of the code there.
- Further split the matching algorithm function into separate functions.

In addition to these changes we checked the length and complexity of functions in the code written after the submission to the SIG.

6 Testing

This section describes the tests we have performed to evaluate the accuracy and quality of the calibration. Most tests performed were black box tests: check if the program as a whole performs well with a given input.

In this section we we first discuss the tests we performed for the standalone calibration and the integrated system. After that a short overview of the test results is given.

The section starts with an overview of the various testing methods used to estimate the performance of the product. For each test, a description is provided: which functionality is meant to be evaluated and how this has been accomplished. The section continues with the test results and the measurements taken, when those were not as desired.

6.1 Testing Methods

6.1.1 Standalone Calibration

For the tests of the standalone matcher we used a piece of paper with an x and y axis on it, for varying the z-position of the points we used plastic cups, or any other appropriate item we could put the pointer on. The calibration has been performed with various positions and orientations of this piece of paper relative to the camera (see fig. 12).

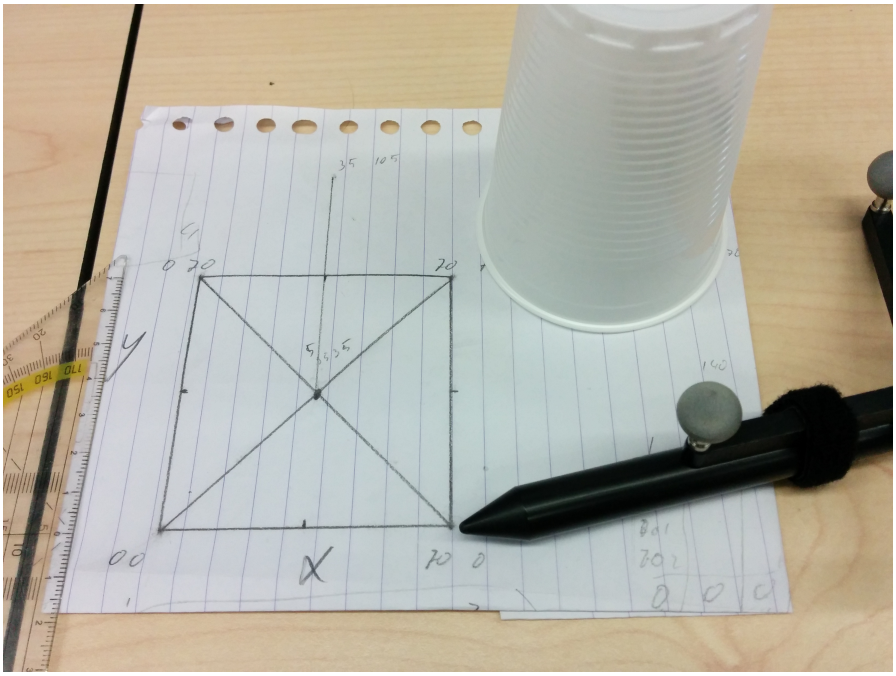

Figure 12: Pen and paper testing setup

The tests with the standalone method quickly proved that the tracking and calibration was accurate enough: in all cases the position returned after calibration was never more than a few millimeters off target. We also found that calibrating on three points often already provided an accurate enough transformation and that matching on more than 5 points rarely provided further improvement. We think this is because that while each new point does provide more information for getting an accurate transformation, it generally also introduces an additional measurement error.

6.1.2 Integration & User tests

After the integration with V-Scope testing on a larger scale was possible. For this we used a phantom (fig. 13) provided by Ivo Wagensveld. This phantom was scanned in a CT-scanner and the dataset converted to V-Scope's file format.

For the phantom we used the procedure as described in section 4.1.3. For these tests the calibration was performed using three points, one in the middle of the 'ribcage' and two at the corners of the plate the phantom is resting on. We chose these points to simulate a calibration using the sternum and the hip bones of a real cadaver. Also in practice, most likely these points will also be used.

After the calibration we tried to hit the targets marked in fig. 13 using a biopsy

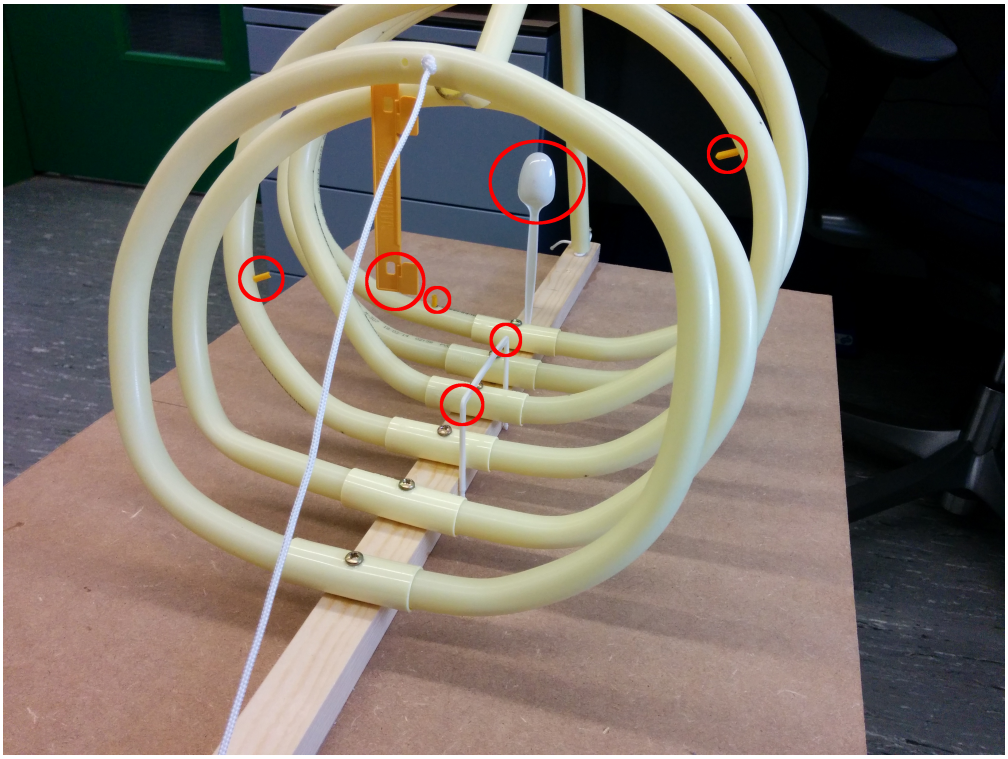

Figure 13: Phantom, red circles indicate targets

needle (fig. 14) similar to the needles used in actual MIA procedures. This was done with varying camera positions, at the front, back and side from the phantom. The camera height was also varied, depending on the height it looked straight forward at the phantom or down at an angle on the phantom.

In this stage the recalibration feature was also tested, this was done by moving the camera or phantom and then performing a recalibration. After that we again tried to hit the targets in the phantom.

Finally a full test run (calibration, tracking and recalibration) was done together with Ivo Wagensveld. From this test we got a lot of useful feedback on the software, mainly related to the usability of the system during a real MIA. However, most importantly, Ivo was satisfied with the achieved accuracy.

6.2 Results

The tests using the system showed that we were able to hit all targets in the phantom reliably. Also the orientation of the needle was rendered correctly, tested by holding the needle next to the ribs of the phantom.

This was also the conclusion of the tests performed together with Ivo Wagensveld.

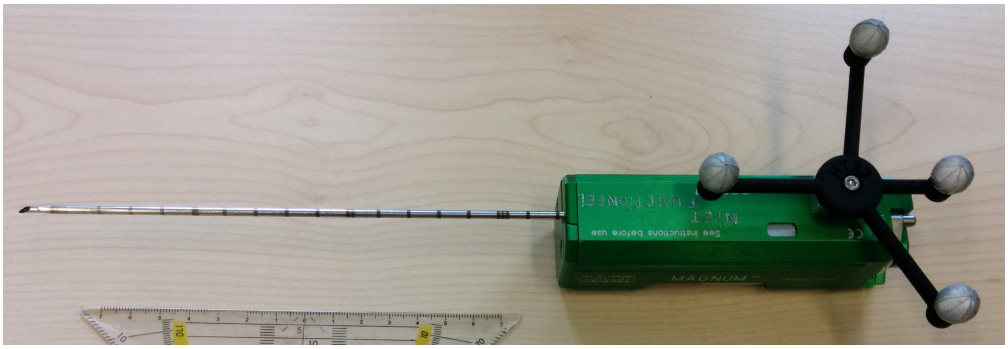

Figure 14: Bioptgun with needle and marker

He thinks the tracking is accurate enough for him to hit the desired targets in a real MIA.

What also came forward in the user test was that the calibration may be a lot easier on the phantom compared to an actual subject. While it is true that the sternum and hip bones are the most easy landmarks to point at. It is still hard to target a specific point on these bones trough the skin. This is especially the case with an obese cadaver. The solution proposed by Ivo Wagensveld for this is to put screws in the hip bones and sternum, as these will be clearly visible on the scan and are always easy to target. The feasibility of this is something he will look into.

6.3 Recommendations for Future Testing

The primary recommendation for future testing, is to use a more realistic phantom. The phantom we used was realistic when it came to the size of the targets we had to hit. However, because the phantom was only 'bones' and see-trough it was sometimes possible to track the needle where this would not have been possible with a real cadaver. The fact that it was only bones, also made calibrating the phantom easier than it will be in reality.

We think that a phantom that is more human-like in these aspects, with some sort of skin-like covering over the bones, will put the difficulty of performing the procedure on the phantom closer to the difficulty of performing a real MIA, and thus provide more accurate test results.

7 Analysis of Final Product

In this section we will provide an analysis of the end product looking back on the requirements specified in the work plan. A very important requirement, not mentioned in the work plan, that was met was customer satisfaction. Both Ivo

Wagensveld and Anton Koning are satisfied with the way the prototype functions and the accuracy of the tracking displayed in the tests.

7.1 Requirements evaluation

Most of the requirements that are mentioned in the workplan are met. Per requirement:

- *System documentation for the prototype must be provided, including improvement opportunities.*
  Our code has been properly documented and commented, the design is discussed in this report and the research report. In addition to that a user manual and installation manual for the system is provided. Improvement opportunities are provided in section 8 in this report.

- *The prototype must be developed for the Linux operating system.*
  The prototype has been developed on the Linux system. However, it is worth noting that we do not rely on Linux specific system calls, therefore it should be possible to port the code to other (semi) POSIX compliant systems.

- *The registration & matching algorithm must work for real world cases (e.g.biopsying a tumor).*
  The registration & matching algorithm works in real world cases, although there has been no testing done on a real cadaver. During tests Ivo Wagensveld was confident that it was accurate enough to perform well in a real case.

- *The tracking accuracy of the biopsy needle must be as high as possible, preferably no more than 1mm.*
  The tracking camera is capable of tracking this accurately. However the actual accuracy depends on the calibration of the system and the precision of the user in pointing out the landmarks during this procedure.

- *The calculation of the matching and the tracking of the needle must be in real time.*
  The calculation of the matching and tracking of the needle are done in real time, with very little lag or stutter.

- *Usability improvements.*
  In the end we did not get to implement a lot of usability improvements. This is in part because many of the suggestions we got are hardware-related and concerning the final use and setup of the physical system in an actual MIA. The kind of changes that don't require a change in the software but do require the purchase of a sturdy tripod, for instance.

We did get some suggestions from Ivo Wagensveld regarding the way we display the needle. These have been incorporated into the software.

8 Recommendations for Future Enhancements

In this section recommendations for system enhancements will be discussed. The first part presents proposals for enhancing the usability of the overall system. In the second part of this section, enhancements for the system's performance are suggested.

8.1 Usability

Below are several suggestions for enhancing the usability and performance of the system in the future.

- **Streamline the calibration process**
  The current calibration process is fast and not difficult to perform. However, if the approach with screws in the hip bone and sternum is followed it is an easy enhancement to attach markers to those points. The SMARTTRACK camera is capable of tracking up to 4 markers, which leaves enough room for a needle and 3 calibration points, the pointer would not be needed in this setup.

  Using this method the only thing that needs to be done for calibration is pointing out the three points on screen and then enough is known to calibrate. When the camera detects movement in one of the calibration markers, it can automatically recalibrate. Additionally, this method makes it harder to introduce measurement errors.

- **User interface**
  Currently the prototype uses the original V-Scope configuration interface. The second way of enhancing the usability of this system is by making a user interface that is clearer and easier to learn. An interactive, modal, user interface will increase usability performance while keeping the current functionality as is. The new interface will contain all the options available on the current interface but it will show only the information a user needs, at certain moment.

- **Needle visualization**
  Currently the system does not automatically position the volume in a way that is practical. The user will have to do this by hand using a 3D mouse. A solution for this would be to attempt to automatically rotate the rendered

volume to an orientation that maximizes the visibility of the needle. This way the user can focus more on performing the biopsy procedure.

8.2 Performance

The main performance metric of the prototype is the accuracy of the tracking for the needle. One approach for improving the accuracy we have not explored during our project, that might prove to be valuable in the future, is using different markers on the needle.

The stability of the tracking is influenced by the shape and size of the marker, perhaps a different shape or size is more optimal for the use case of the prototype. This is something that can be researched, perhaps in collaboration with ART, to further enhance the accuracy.

9 Project Process & Execution

In this section a comparison between the schedule planned in the workplan and the actual progress during this project will be shown. In the planning of the workplan the first two weeks of this project were reserved for the research report. The research consisted of analyzing the technologies to be used in the system and approaches for matching the 3D points in different coordinate systems for the calibration process. The research report took one week longer than planned because we also had to spend time setting up our working environment, e.g. doing fresh installs on the computers and installing V-Scope. Additionally the papers we read for the calibration algorithm took a lot of time due to the 'density' of the material.

The amount of sprints was shortened to 6 sprints for implementing most of the functionality, with 4 sprints until the initial code review by SIG. The first three sprints we focused on further becoming familiar with the tracking camera and V-Scope, implementing the matching algorithm for the calibration and performing the initial tests for the algorithm. In the three sprints that followed the integration with V-Scope was done and the prototype was made ready to be used as a whole.

During sprint 3 and 4 we got some new hardware, a tracking camera and new 3D mouse. Due to a communication error we were initially not aware we were going to get this, some extra time had to be spend to make the needed modifications to use this hardware.

Examples of the suggestions from the user test, mentioned at sprint 6, are: display an indication of the measurement error and increase the visibility of the needle.

Here follows a short overview per sprint:

- **Sprint 1**
  - Further Analyze V-Scope and camera.
  - Begin implementing matching algorithm.
- **Sprint 2**
  - Implement matching algorithm.
  - Start work on standalone calibration test tool.
- **Sprint 3**
  During this sprint we received a new tracking camera.
  - Adjust standalone matcher for new camera.
  - Finalise standalone calibration test tool.
  - Test calibration.
- **Sprint 4**
  During this sprint we got the new 3D-mice.
  - Modify SVRD for new 3D mice.
  - Modify SVRD for multiple marker tracking.
  - Integrate calibration algorithm into V-Scope.
  - Send code to SIG.
- **Sprint 5**
  During this sprint we performed a user test with Ivo Wagensveld.
  - Finalize calibration workflow in V-Scope.
  - Perform user test.
  - Process feedback from SIG.
- **Sprint 6**
  - Implement suggestions from user test.

Despite the initial delay and some of the complications encountered (e.g. new hardware), the project has finished within its predefined duration by taking the appropriate mitigation measures. Although the adjusted planning caused by the delays did leave us with less time to refine the prototype, only sprint 6 was left for this.

10 Conclusion

The goal of this project was to implement 3D-guided biopting capabilities in existing software at the Erasmus MC, i.e. V-Scope. To solve this problem we had to solve different sub problems such as getting the calibration to work and integration our approach.

Despite some of the delays encountered during the project the group delivered a working prototype on time. The final prototype has been tested on an example target and has been deemed accurate enough in user tests and will be tested on actual cases soon. The other requirements have been met as well, as is showcased in section 7 and our contacts at the Erasmus MC are satisfied.

The members of our group have shown that they can perform well individually but also in a team. Despite small conflicts, our team acted professionally and always found a compromise in order to continue. We are happy to see that the prototype will likely be tested in a real case soon.

Additionally this project provided us with hands on experience with the use of computer based visualisations in a medical environment. Furthermore working with a large existing code base has been a challenge during the programming, but experience with this is definitely an asset for the future.

The resulting system is already operational and can be optimized to a high value product.

References

[1] Julia F Barrett and Nicholas Keat. Artifacts in ct: Recognition and avoidance 1. *Radiographics*, 24(6):1679–1691, 2004.

[2] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987.

[3] H L Oswal and S Balasubramanian. An exact solution of absolute orientation. *Photogramm. Eng*, 34:1079–1083, 1968.

[4] G. H. Schut. On exact linear equations for the computation of the rotational elements of absolute orientation. *Photogrammetria*, 16:34–37, 1960.

[5] E. H. Thompson. A method for the construction of orthogonal matrices. *Photogramm*, 3:55–59, 1959.

Appendix A Workplan

This is the workplan our group created at the start of the project.

Workplan 3D-guided biopting

Jasper Dijt, Alexandros Karalidis, Haluk Sahin
{J.W.Dijt, A.Karalidis, H.Sahin-1}@student.tudelft.nl

June 13, 2014

Contents

1 Introduction

At the Erasmus MC the technology of 3D-CT guided biopsies is being researched and a prototype system is under development. It is expected that this technology will make the process of biopting easier, faster, and more accurate. The task to improve the prototype and to make it more usable has been assigned to us.

This work plan will be written in collaboration with Anton Koning, our supervisor at the Erasmus MC. In addition to that it will be approved by our project coach at the Tu Delft, Elmar Eisemann.

In this work plan the following subjects will be discussed. First the project assignment will be described in detail, the requirements and scope of the project will be defined here. After that the methodologies that will be used for work in the project will be discussed. Finally this document will contain a planning for the project.

2 Project Assignment

At the Erasmus MC CT guided biopsies are currently performed using a biopsy needle with repeated radiological checks to verify if the needle is still on target. With 3D-CT navigation these checks will not be redundant, 3D-CT navigation will most likely allow the procedure to be more accurate, faster and easier to perform.

Initially the technique is going to be used for Minimal Invasive Autopsy (MIA) procedures. In a MIA first a detailed CT-Scan is made of the (deceased) subject. After that several tissue samples are taken from the subject to investigate the cause(s) of death. The method for 3D-CT navigated biopsy in a MIA should be as follows:

1. A detailed CT scan of the subject is made.
2. Several markers (at least 3) are placed on the 3d model of the CT scan.
3. Using a trackable pointer those same three points are touched on the actual subject.
4. These points are then matched by the system to get a match between the displayed 3D model and the subject.
5. Now a biopsy needle, again trackable, can be used to retrieve tissue samples from the subject. Using the match made by the system the needle is visualized on screen and the pathologist can see exactly where in the subjects body his needle is.

Already in place are an application for rendering the CT data in 3D and manipulating it (v-scope). The assignment will primarily focus on getting the matching between the subject and the CT data right. At the moment this part of the process does not work.

2.1 Goal

The goal of this project is to create a working prototype that matches the CT data to the actual subject, tracks the biopsy needle that is used to take a tissue sample

and finally visualizes the position of the needle on screen within the CT data. Additionally, any spare time will be used to increase the usability, maintainability and accuracy of the program.

2.2 Scope

The following items are in the scope of this project :

- Provide documentation for the overall system (e.g. diagrams).
- Fix matching functionality of the current system and optimize the algorithm to reach the desired accuracy.
- Improving input data (e.g. taking more features into account).
- Improve the maintainability of the system (i.e. SIG control).
- Improve the graphical user interface(i.e. v-scope GUI).
- Formulate test cases for quality testing.

These items are out of scope for this project:

- Changes or optimizations in the drivers and/or libraries related to the hardware that is used.

2.3 Deliverables

In week two of the project (calendar week 19) a small research report containing at least a detailed description of the existing software and several options for tracking algorithms will be delivered.

In week 11 of the project a working and optimized version of the prototype will be delivered, along with proper documentation and a final report.

2.4 Requirements

The following requirements have been formulated:

- System documentation for the prototype must be provided, including improvement opportunities.
- The prototype must be developed for the Linux operating system.
- The registration & matching algorithm must work for real world cases (e.g. biopting a tumor).

- The tracking accuracy of the biopsy needle must be as high as possible. Depending on the specifications of the provided hardware the maximum tracking error should not be over 1mm.
- The calculation of the matching and the tracking of the needle must be in real time.
- Usability improvements once registration & matching work (i.e. GUI extensions, performance improvements). The specific requirements for this will be determined and documented once the amount of time available for usability improvements is known.

3 Project layout

In this section the project layout will be discussed. First the relevant persons and their responsibilities in relation to the project will be identified. After that our approach will be discussed per phase: research & design, implementation, and product delivery.

3.1 Persons & Responsibilities

In this section we will first discuss the expertises of the group members, after that an overview of key persons for the project will be given listing their roles and responsibilities.

Each member has some main expertises. Based on these expertises the main responsibilities are distributed to the team members. In our case Alex and Haluk have taken the courses for the media and knowledge technology variant of the TI bachelor. Their experience in computer graphics can be exploited by taking a leading role in all the relevant processes of this project. Jasper has taken the courses of the software technology variant of the TI bachelor, additionally he has extensive experience with Linux. His skills will be of importance during the integration of the matching algorithm into the current prototype and with other low level programming or optimization tasks.

In addition, Alex has a good background in project and process management from a minor and therefore can undertake all the relevant responsibilities. Haluk is the only group member having good insight of medical sciences via his minor, this will most likely be very useful during this project.

3.1.1 Overview of persons

In this list an overview of relevant persons is given with an overview of their specific roles.

Alexandros Karalidis Bachelor project group member.

Anton Koning Project coördinator.

- Primary contact at the Erasmus MC.
- Creator of the existing system. (v-scope).

Elmar Eisemann TuDelft project coach.

- Document approval (work plan, research and final report).

Ivo Wagensveld Contact at pathology department.

Jasper Dijt Bachelor project group member.

Haluk Sahin Bachelor project group member.

3.2 Research & design

The research phase of the project is where we explore the existing system. During this phase, we are going to make a list of requirements that must be fulfilled for the end of the project. In addition, we will formulate all the improvements that are desired and implementable, concerning the current system. We will at least look into the following subjects:

1. Analysis of current system (i.e. Architecture,diagrams)
2. Analysis of multiple tracking methods.
3. Formulation of optimization opportunities
4. Preliminary implementation Details (i.e Diagrams,Use cases)

3.3 Implementation

During the implementation phase we will implement as many of the required changes as possible. We will use the Scrum methodology, issues/features will be tracked with an issue tracking system. The issue tracking system will be selected during the research phase. In this phase we will also select which tool we will use for source control.

Due to the relatively short duration of the project a tight feedback loop is desirable. Therefore we will be doing sprints of one week and at the end of each

week a feedback session will take place in order to evaluate the extent to which the requirements are fulfilled.

Because we are working with an existing code base without unittests implemented we will have to be careful when refactoring. If possible we will write test cases for existing functions before modifying them.

In addition to (re)writing the code every week an amount of time will be dedicated to documenting and working on the (final) report.

3.4 Product delivery

Product delivery is in the final two weeks. During these weeks a final sprint will be done, the presentation will be prepared and the report will be finalized.

4 Planning

In this section a per week overview of deadlines and goals is given. For each week the date of the Monday is listed.

Wk 1 – 28 April
- Work plan should be completed ASAP.
- Research current implementation, libraries used, devices used.

Wk 2 – 05 May
- Create design & requirements.
- Get report reviewed.
- DEADLINE 09/5: Research report done.

Wk 3 – 12 May
- Sprint 1

Wk 4 – 19 May
- Sprint 2

Wk 5 – 26 May
- Sprint 3

Wk 6 – 02 June
- Sprint 4

Wk 7 – 09 June
- Sprint 5
- DEADLINE 13/6: Initial code review by SIG

Wk 8 – 16 June
- Sprint 6

Wk 9 – 23 June
- Sprint 7

Wk 10 – 30 June

- Send in code for second review (some moment this week)
- DEADLINE 5/7: Turn in final report.

Wk 11 – 7 Juli

- Presentation!

Appendix B Research report

This is the full text of the research report our group created in the first few weeks of the project.

Research Report 3D-Guided Biopting

Jasper Dijt, Alexandros Karalidis, Haluk Sahin
{J.W.Dijt, A.Karalidis, H.Sahin-1}@student.tudelft.nl

June 13, 2014

1 Introduction

Three dimensional navigation systems are emerging in healthcare, because there is a need for accurate minimal invasive surgery and autopsy techniques. For this bachelor project a prototype 3D navigation system for taking biopsies will be developed.

In this report several subjects will be discussed: First, the existing system for rendering 3D images will be described since we have to integrate our final approach with the existing system (V-Scope) that is used at the Erasmus MC. Second comparable systems and techniques for 3D navigated surgery will be analysed in order to get a better idea of the methods and solutions pursued by others. Third, the mathematical background for calibrating the tracking for the navigation is provided, because without this accurate navigation is impossible. Finally, the practical details of the implementation will be discussed in order to get a good overview of the possibilities and potential problems we might face during the implementation phase.

2 Existing Software

In this project an extension to existing software at the Bio-informatics group at the Erasmus MC (EMC) will be made. This section provides a short overview of the existing functionality and use cases of the software. Afterwards, a short review of the various modules, their functions and inter-dependencies will be given.

2.1 Current Functionality

The existing software, named V-Scope, is used to render and manipulate 3D images. These images can range from CT, MRI or 3D ultrasound scans to very detailed scans of microscopic samples. V-Scope can be used for rendering these images in an immersive environment, such as the I-Space facility at the EMC Bio-informatics department[1], which is a CAVE-like environment. However, it can also be used on a desktop computer using a stereoscopic 3D monitor. For our project the desktop version will be used. The 3D rendering can be navigated using, among others, a normal mouse, a 3D mouse and trackable pointers using a tracking camera.

Included in this report are two screen-shots from the V-Scope system: Figure 1 shows the configuration & control window of the V-Scope software. Figure 2 shows an example rendered data set, the top half is for the left eye and the bottom half for the right eye. These separate images are overlaid by the 3D monitor to get a 3D image that van be viewed using special polarized glasses.

[1] `http://www.erasmusmc.nl/bioinformatica/virtualreality/4519666/`

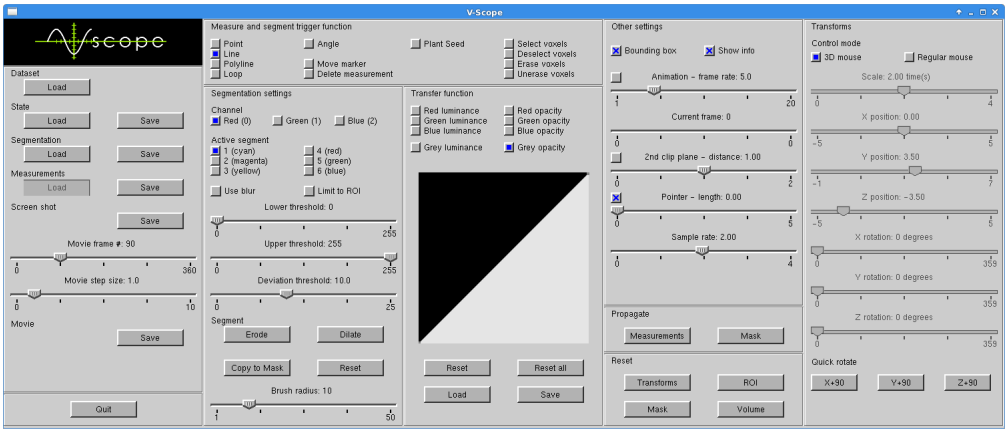

Figure 1: The configuration & control window in V-Scope.

The visualization provided by V-Scope allows users to more easily see details in a scan compared to a rendering on a normal computer screen. It also allows for 'erasing' parts of a scan to isolate specific sections and it facilitates making accurate 3D measurements within a scan.

2.2 System architecture

V-Scope is written in C/C++ and is designed to run on Linux. V-Scope has primarily been developed by the project coördinator at the EMC: Anton Koning. It runs as two separate programs that communicate via shared memory, see Figure 3. The Simple Virtual Reality Daemon (SVRD) program has to be run as root and reads the input from the 3D mouse and tracking camera. V-Scope itself may be run by a regular user, V-Scope reads the user input from the shared memory segment, loads the selected data set and manages the rendering of it.

2.3 SVRD

SVRD is a small daemon written in C++ that processes the input from a 3D-mouse and tracking cameras. Additionally it provides a client library that is used by V-Scope to connect to SVRD. At present, there is support for a 3DConnexxion[2] 3D mouse and various tracking cameras. In our project a SMARTTRACK tracker from Advanced Real time Tracking will be used[3].

Currently, SVRD is made up of three modules relevant for our project, modules for controlling different tracking camera's are not included, see Figure 4. The main

[2] `http://www.3dconnexion.eu/index.php`
[3] `http://www.ar-tracking.com/products/tracking-systems/smarttrack/`

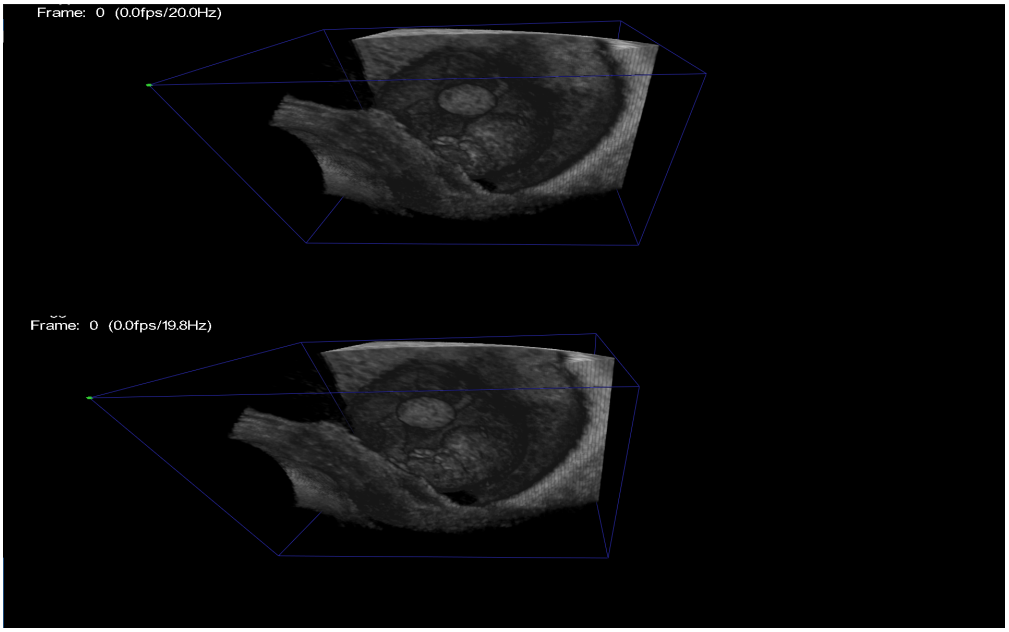

Figure 2: 3D data display in V-Scope (desktop version).

module, svrd, manages the shared memory block and polls the modules for the tracking camera, dtrack, and 3D mouse, spacenav. In the current implementation the data in the shared memory is not locked when it is written to or read from.

2.3.1 Dtrack

The dtrack module is the component of SVRD that uses the library provided with the tracking camera to retrieve the position of the trackable markers. Upon initialization, a handle for the tracking camera's is acquired. Once the handle is acquired the library is configured to track the markers, see Figure 5 for an example of a marker.

When polled by the main module, the dtrack module retrieves the current position of the marker from the library if it is availablei. This position is then stored as a translation vector and rotation matrix in the shared memory, the numbers in the vector and matrix are stored as double precision floats. The translation vector gives the position of the marker relative to the origin of the tracking camera in mm in a right handed coördinate system. In the case of the current tracker the default origin is in the middle between the cameras. So a translation vector $\boldsymbol{v_t} = [45.1, 50.1, 800.02]^\intercal$ indicates that the marker is at 45.1mm on the x axis, 50.1mm on the y axis and 800.02mm on the z axis. The rotation matrix is used to indicate the orientation of the marker. For instance: if the marker

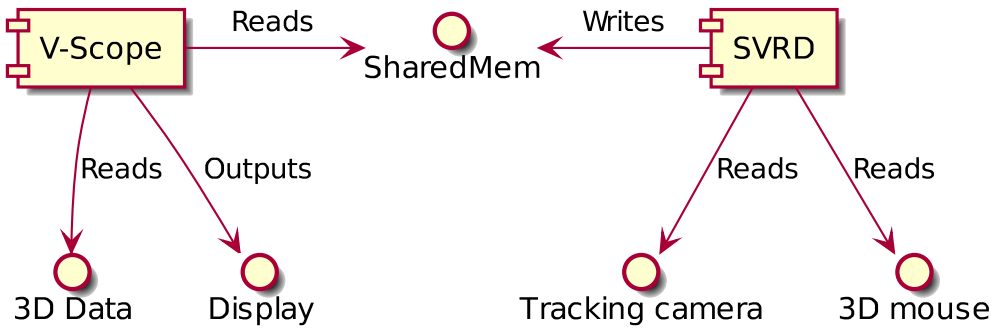

Figure 3: V-Scope system overview

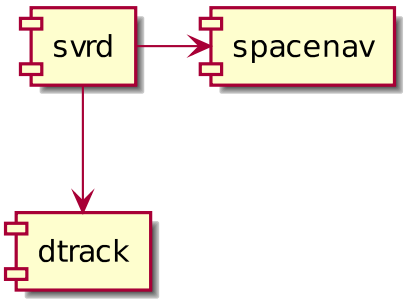

Figure 4: SVRD overview

is attached to a pen this could be used to determine in which direction it is pointing.

2.3.2 Spacenav

The spacenav module manages the input from the usb 3D mouse. When polled by the main module spacenav retrieves io events from the mouse. These can be changes to the position of the mouse 'joystick' or button presses. Position changes are returned via the shared memory in a list of 6 double precision floats between 1 and -1 indicating how far the joystick is pushed or rotated in a certain direction. The first three are movements across the x, y and z axes the second three values are rotations around the x, y and z axes. The coördinate system used in this driver again conforms to the right hand rule, with the x axis pointing right, the y axis pointing toward the user and the z axis pointing down using the middle point of the joystick as the origin.

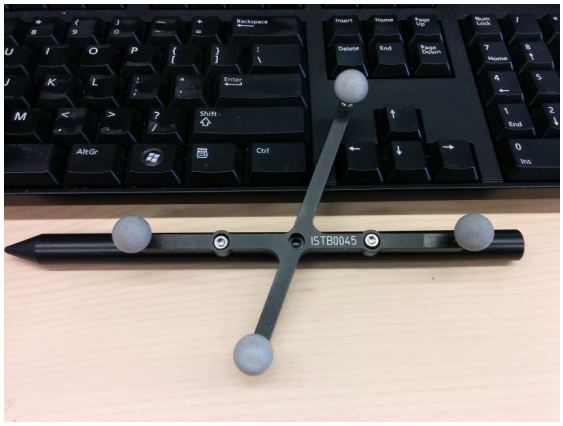

Figure 5: An example of a marker.

2.4 V-Scope

V-Scope is the program that manages the rendering and editing of the 3D data. As shown in Figure 6 V-Scope is made up out of several modules. Svr is the client library for SVRD, SceneGraph (SG) is a library that implements a scene graph and wraps OpenGl, Portable Volume (PV) is a library that defines the file format used by V-Scope and provides functions for various operations on these files and finally, V-Scope is the main application that uses these modules to render the data.

In this section a more detailed description of V-Scope and SG will be given. Because it is extremely unlikely that we will be doing any work within PV and because the svr client only provides direct access to the shared memory and does nothing else. These modules will not be reviewed in depth.

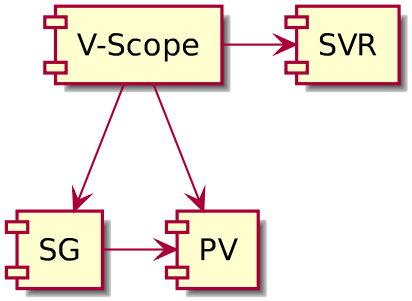

Figure 6: V-Scope overview

2.4.1 SceneGraph (SG) & V-Scope in depth

SG is a library that provides various utilities for V-Scope. Primarily it provides a scene graph that is used by V-Scope to manage the various objects (nodes) in a rendered scene. It provides a base node class that is extended throughout SG and V-Scope to define various node types such as transformations, shapes, markers, etc. The SG library then tracks where shapes intersect and what part of the scene actually has to be rendered on screen.

In addition to the scene graph functionality SG provides utilities for performing linear algebra operations, shared memory management and locking for the scene graph data structure that are used extensively throughout SG and V-Scope.

V-Scope, in addition to extending SG classes, also contains the code for opening the window with the final render (Figure 2) and the configuration window (Figure 1).

3 Similar Systems & Techniques

Computer-aided navigation systems are researched since 1995 [3] and have been a great assistance in surgeries. There are several fields in which computer-aided navigation systems are used. In this section, several applications and techniques used for computer-aided navigation systems will be presented.

In the medical sector, the most common technique to implement 3d reconstruction of a subject is to use images taken from CT-scans. The advantages of CT over other imaging techniques are the high speed of taking images (high frame rate), the high resolution of images that it captures. However, CT-scans have also their disadvantages in relation to other techniques. For instance, it generates high levels of radiation.

A good alternative for CT-scans is the magnetic resonance imaging (MRI). MRI-scans are broadly used in three dimensional imaging technologies since they produce low error rates (about 0.56 mm in relation to 0.55 error of a ct-scan) [1]. One advantage of MRI over CT is that is has no radiation at all. In addition, it has an higher resolution than CT. Another advantage of the MRI is that it can make contrast between different type of tissues. However, isosurfaces are difficult to be produced with MRI in contrast to CT.

In the following paragraphs a number of cases is presented, offering a better insight on computed tomographic systems.

In the study of Ewers et al. a set up of computer-aided navigation system has been researched. That system was meant to support a dental surgeon during the surgery [3]. The duration of this study was 12 years, during which the set up had been optimized multiple times with hardware changes and software updates.

In more detail, the system consists of a 3D Model of the CT -scan of a patient

which is enhanced with graphical structures (e.g. points, lines and planes) made before the operation. The tracking of the patient's body position and the tracking of the surgical equipment is done by using an IR-digitizer. The digitizer sends an infrared signal and records the reflection made by markers that are placed on the patient and the tools. With this system the surgeon can observe the location of the surgical equipment on the patient by a monitor or an augmented reality glass. The conclusion of this study is that the computer-aided navigation technology is helpful in most indications (i.e. brain tumour biopsy).

In another study they used computer aided navigation in neurosurgery [4]. In this paper they discuss different methods for tracking the equipment used for neurosurgery. One way to keep track of the position of the equipment on the patient in 3D space is with an arm-based pointer. This arm contains joints equipped with sensor that measure the angles, which can be used to measure the relative distance of the surgical equipment to the target. Another system is based on an armless pointer that can be either tracked by sonic waves, magnetic field or optic sensors. Depending on the technique different sensors can be used which differ in accuracy and speed [4].

4 Matching & calibration

As described in the project assignment section of the work plan, the method for taking a 3D guided biopsy contains a step where the 3D tracking is calibrated. This must be done to establish the relation between the space seen by the tracker and the space displayed on screen. The method for doing this is described in the work plan and consists of identifying at least three pairs of points on both the 3D rendering of the CT data and the actual subject. So for every point p_i in the 3D rendering there will be a matching point p_i^* in the space of the tracker. Using these point pairs, a transformation matrix can be derived which can be used to accurately display trackable tools within the application. In its simplest form the following equation will have to be solved for this [4].

$$XT = X^*$$

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} = \begin{bmatrix} x_1^* & y_1^* & z_1^* \\ x_2^* & y_2^* & z_2^* \\ x_3^* & y_3^* & z_3^* \end{bmatrix}$$

In this equation X is a matrix that consists of the x, y, z coordinates of three points in the space defined by the tracker and X^* the coordinates of three points in the space of the 3D rendering. T is the transformation matrix that has to be determined. Generally this equation has no solution if more than three point pairs

are used [4]. However it is possible to use additional point pairs to select the three best point pairs to reduce the overall error.

Another solution for determining the transformation that works with any number of point pairs greater than three also exists. This solution is Horn's method using unit quaternions [5] or a similar method proposed by Horn using orthonormal matrices [6]. Several example implementations of Horn's algorithm can be found, primarily for Matlab. However implementations can be found in other languages as well. The Point Cloud Library[4] implements functions for this in C++, to name an example.

Because the basic method described does not use all available information to calculate the rotation matrix we think that Horn's solution will produce a more accurate transformation. This is stated by Horn in his articles as well [5, 6].

5 Implementation considerations

As stated in the work plan how to track issues and what tool to use for source control will be discussed in this document. Other implementation concerns are the used programming language, available libraries and available development tools for the selected language. Because the implementation will have to be integrated into an existing code base the choice of libraries or programming language will not always be completely free. In this section first source control & issue tracking will be discussed, after that programming language and libraries.

5.1 Issue tracking & source control

Various web based issue tracking systems have been discussed within the group. Several free & open source alternatives exist such as: Trac[5], Redmine[6] and Bugzilla[7]. The same goes for source control, two of the most commonly used tools for this are Git and Subversion. Regardless of the selected tool a method for using it has to be determined.

5.1.1 Issue Tracking

For issue tracking most packages provide the needed features, i.e. basic workflow, creating and tracking issues. One of the group members, Jasper Dijt, already has extensive experience with the Redmine issue tracking system and already has it

[4] `http://www.pointclouds.org/`
[5] `http://trac.edgewall.org/`
[6] `http://www.redmine.org/`
[7] `http://www.bugzilla.org/`

installed on a server. This makes Redmine the fastest to set up and use for this group as it is already installed and one of the group members knows the system well. Therefore Redmine will be used for for issue tracking.

5.1.2 Source Control

For source control primarily Git and Subversion have been compared. Git has been selected because it is faster and has significantly better branching support (see also the next section). And additional advantage is that every project member will have a full copy of the repository, next to the central one. This works as a (semi-)automatic distributed backup and allows group members to work offline, in contrast to Subversion which requires a connection to the central repository to commit.

5.1.3 Branching Workflow

The Git Book [2] has a chapter on branching workflow for Git. Based on the recommendations from this chapter we will use a master branch for stable versions, a development branch for developing and topic branches for commits related to specific features. If the work on a feature is complete it will be merged into the main development branch. When the code in the develop branch is in a stable state, as it should be at the end of a sprint, it can be merged into master as the current stable version.

5.2 Language, Libraries and Development Tools

The existing system, V-Scope, is written in C/C++. Because the tracking will have to be integrated within V-Scope it is a natural choice to develop this using C++ as well. In addition to that this will make it easier for other people working on V-Scope to later modify and use our code because it is written in a language already familiar to them.

This brings up several topics to consider such as the libraries that will be used, what kind of coding style guidelines will be used and language specific features we should be aware of. In some cases the choice will already have been made for us, for instance the SG library and V-Scope heavily depend on OpenGl for rendering and on functions from the POSIX standard for general system operations. This limits us to (semi-)POSIX compliant operating systems, such as Linux.

C++ is an object oriented language which is known for high performance. This is one of the reasons the prototype of the 3d navigation system has been implemented in C++. However some C++ features deserve some extra attention, for instance the runtime does not provide a garbage collector.

5.2.1 Libraries for C++

As mentioned before in some cases there is but one feasible option for the libraries to use in the project, for instance OpenGl for rendering and the device specific library for the tracking camera.

For the matching algorithm discussed in Section 4 we will have to perform operations on matrices and vertices. C++ does not natively support these kind of operations, thankfully several C++ libraries for performing such operations exist. It is vital that the selected library provides good performance because it is required that we reach real time speed for matching and tracking.

Several libraries exist for that, such as Armadillo[8] and Eigen[9]. Both are open source libraries used in several open and closed source projects and both provide similar performance[10].

An advantage of Armadillo is that it allows for working with matrices and vectors in a way similar to Matlab. Since all group members are familiar with Matlab, this library should be easier to use in comparison to Eigen.

On the other hand, Eigen has its advantages such as an easy syntax which is not so difficult to learn. Moreover, another important advantage of Eigen over Armadillo is that it includes quaternion functions. These build-in functions can become very usefull during the implementation of the Horn's algorithm which uses quaternions to calculate the transformation matrices.

5.2.2 Coding Style

When programming in a group it is important to maintain a consistent coding style. This makes the code more readable for both the group members and people who have to work with the code after the project.

Various guides for coding style and best practises for C++, examples of this are Bjarne Strousup's C++ Style and Technique FAQ[11], the Boost library guidelines[12] and the Google C++ style guide[13]. It is a good idea to base the coding style used for the project on an existing guide to save time.

For this project the C++ style guide from Google will be used. This style guide explains structurally how to write C++ code in a uniform way. In addition it disallows or restricts the use of some C++ features that may act as enablers for programmer errors (due to poorly readable code). A rationale is provided for

[8] http://arma.sourceforge.net/
[9] http://eigen.tuxfamily.org/
[10] http://nghiaho.com/?p=1726
[11] http://www.stroustrup.com/bs_faq2.html
[12] http://www.boost.org/development/requirements.html
[13] http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml

all conventions listed and a script that checks code for adherence to many of the guidelines is available. Below a short overview of the guidelines is given.

- **Header files**
  In the header file section guidelines for writing header files and including them are given. It contains important readability rules such as a guideline for function parameter ordering (inputs, then outputs) and the order of includes.

- **Scoping**
  In this section guidelines for the use of namespaces are discussed. Using namespaces to reduce the amount of symbols in the global namespace is encouraged.

  The usage of the '`using namespace <namespace>`' to make all symbols in a namespace available is discouraged to prevent namespace pollution. However '`using <function-or-class-in-other-namespace>`' within classes, functions and .cc files is ok, as is using namespace aliases. The most important rationale for these rules is to keep it as clear as possible which functions from what namespace are actually used in the code.

- **Classes**
  The guidelines for classes are fairly straightforward and mainly go into member variable initialisation, the providing of a copy constructor and/or assignment operator and access control (member variables should be private).

  When using classes the use of multiple inheritance is discouraged, however it is allowed if only one of the parent classes has an implementation. The other parents must be pure virtual classes (like Java's interface declaration).

- **Naming**
  The primary guideline for naming is that names should be descriptive, i.e. "`int num_errors;`" vs "`int nerr;`".

  In addition to the primary guidelines naming rules for types, variables, constants, etc. are defined.

- **Comments**
  Comments are important to improve the readability of the code. They are required above function declarations (describing use) and function implementations (describing operation). For variable comments the use of a descriptive variable name is encouraged, however if needed a comment should be placed. Both block (`/* */`) and line (`//`) comments are allowed.

  Finally all comments should be written using proper grammar, spelling, and punctuation.

Naturally the most important part is that the coding style is consistent across the project. Therefore the style guide will be treated as a guideline that should be followed but can be deviated from.

5.2.3 Development Tools

The existing system can be build using GNU Make and GCC. However it is also possible to use an Integrated Development Environment (IDE), like Eclipse[14], for developing C++ software. While it takes a bit of time to properly set up an IDE it can improve productivity (i.e. improved syntax checking and automatic inclusion of header files). On the other hand, some group members are not familiar with IDEs and prefer to work in the terminal.

Backwards compatibility with the existing build system has to be maintained. In order to do this, we will import the project with its own makefile into Eclipse. This approach allows us to use as many of the features of Eclipse as possible in combination with the excisting build system (GNU Make). A consequence of this is that the makefile has to be maintained by hand.

References

[1] Joyce Van den Broeck, Evie Vereecke, Roel Wirix-Speetjens, and Jos Vander Sloten. Segmentation accuracy of long bones. *Medical engineering & Physics*, 36(5):563–575, 2014.

[2] Scott Chacon and Junio C Hamano. *Pro git*, volume 288. Springer, 2009.

[3] R Ewers, K Schicho, G Undt, F Wanschitz, M Truppe, R Seemann, and A Wagner. Basic research and 12 years of clinical experience in computer-assisted navigation technology: a review. *International journal of oral and maxillofacial surgery*, 34(1):1–8, 2005.

[4] P Grunert, K Darabi, J Espinosa, and R Filippi. Computer-aided navigation in neurosurgery. *Neurosurgical Review*, 26(2):73–99, 2003.

[5] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987.

[6] Berthold KP Horn, Hugh M Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135, 1988.

[14] `http://www.eclipse.org/cdt/`