UNIVERSITY OF CALIFORNIA SAN DIEGO

Optical Tracking via Stereo-vision for Surgical Navigation

A Thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Bioengineering

by

Ananya Rajan

Committee in charge:

Professor Frank E. Talke, Chair
Professor Geert Schmid-Schoenbein, Co-Chair
Professor Alyssa C Taylor-Amos

2024

Copyright

Ananya Rajan, 2024

All rights reserved.

The Thesis of Ananya Rajan is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

TABLE OF CONTENTS

THESIS APPROVAL PAGE……...iii

TABLE OF CONTENTS……iv

LIST OF FIGURES…….v

LIST OF TABLES……..vii

ACKNOWLEDGEMENTS ……..……viii

ABSTRACT OF THE THESIS…….x

Chapter 1 SURGICAL NAVIGATION SYSTEMS…….12

Chapter 2 OPTICAL TRACKING WITH COMPUTER VISION……...22

Chapter 3 EXPERIMENTAL SETUP……..31

Chapter 4 EXPERIMENTS……. 42

Chapter 5 RESULTS AND DISCUSSION……..49

Chapter 6 CONCLUSION AND FUTURE WORK……57

REFERENCES…….65

LIST OF FIGURES

Figure 1.1: Surgical navigation workflow comprising preoperative imaging and planning, registration, and intraoperative navigation……13

Figure 1.2: NuVasive's Pulse Platform's instrument reference array [13] ……..……14

Figure 1.3: Pulse Platform's Navigation Software displaying screw trajectories [13] ……15

Figure 1.4: Stryker's surgical navigation solution showing the SpineMask non-invasive tracker on the back and the SpineMap 3D navigation software [14] ……16

Figure 1.5: 7D's Flash Navigation system [16] ……17

Figure 1.6: Augmedics XVision based on augmented reality [19]……18

Figure 1.7: Passive 'sleeve' type marker attached to instrument [23] ……..19

Figure 2.1: Design of Surgical Navigation System ...……...21

Figure 2.2: ArUco Markers with different patterns and IDs ……23

Figure 2.3: 3D ArUco Markers for continuous detection of instruments ……24

Figure 2.4: Steps to calibrate camera…….25

Figure 2.5: Pinhole Camera Model [34]……26

Figure 2.6: Finding corners in the checkerboard pattern for calibration……...26

Figure 3.1: Stereoscopic Camera ……..31

Figure 3.2: Images for calibration ……32

Figure 3.3: The XY positioning platform ……34

Figure 3.4: 5DOF robotic arm holding marker……..36

Figure 3.5: Programming robot ……37

Figure 3.6: Some trajectories considered for the robot: (a) Circle (inclined), (b) 'S', (c) Parabola and (d) Infinity……...38

Figure 3.7: (a) Surgical Needle, (b), (c) Needle with marker attachment…….41

Figure 4.1: Marker movement on XY platform…….43

Figure 4.2: Inclined or 3D circular path……..…44

Figure 4.3: Different marker colors (white, pink, orange and yellow) …….45

Figure 4.4: Various marker sizes tested ……45

Figure 4.5: RGB color space [49] …….46

Figure 4.6: HSL (left) and HSV (right) color spaces [52] ……47

Figure 5.1: Experimental vs. theoretical data points obtained from tracking marker on XY platform…….52

Figure 5.2: Patterns (a) before and (b) after SVD ……52

Figure 5.3: Correlation between experimental and theoretical data (a) before and (b) after SVD……...53

Figure 6.1: An example of a disparity map created from stereo images [61] ……..58

Figure 6.2: From left to right - 5-sided [63], 12-sided and 6-sided ArUco dice ……..60

Figure 6.3: A novel 9-sided ArUco die in different sizes ……61

Figure 6.4: Marker attached to surgical instruments ……61

Figure 6.5: Needle bending under the weight of the white resin die ……62

LIST OF TABLES

Table 1.1: Summary of existing technologies …….18

Table 1.2: Summary of tracking methods and accuracies ……20

Table 3.1: Calibration angles for robot …….37

Table 3.2: Scaling factors for the robot ……40

Table 4.1: Experimental parameters …….42

Table 5.1: Detection % of colored markers moving in the XY platform in RGB, HSL and HSV color spaces averaged over 3 trials using homography [42] ……50

Table 5.2: Root mean square error (mm) of tracking colored markers moving in the XY platform in RGB, HSL and HSV color spaces averaged over 3 trials using homography [42] …….50

Table 5.3: Root mean square error (mm) of tracking 30 mm, 20 and 15 mm markers in RGB, HSL and HSV at 500 mm from camera ……..51

Table 5.4: Root mean square error (mm) of tracking 30 mm, 20 and 15 mm markers in RGB, HSL and HSV at 600 mm from camera ……..51

Table 5.5: Root mean square error (mm) of tracking 30 mm, 20 and 15 mm markers in RGB, HSL and HSV at 700 mm from camera ……..51

ACKNOWLEDGEMENTS

I would like to thank Dr. Frank Talke for his unwavering support ever since the time I joined his lab two years ago. He always encouraged me to explore new areas of research, and taught me to enjoy the journey, despite the many hurdles faced along the way. I am ever grateful to him for believing in my work and instilling in me the confidence to do the same.

I would like to thank Dr. Farshad Ahadian, anesthesiologist at the VA Medical Center, for allowing me to shadow him multiple times while he performed spinal procedures. Without his advice and helpful insights into surgical navigation and its needs, the project would not have progressed as much as it has.

I would like to express my deepest appreciation to Darin Tsui who introduced me to this project. He has been very patient with answering all of the questions I had about computer vision, and was instrumental in helping me navigate the project. Darin and Mitsuhiro Jo-Zee, both undergrads at the time, gave me much needed support in planning out the flow of the project and my thesis. I am also thankful to Bryan Nguyen, a previous Masters student, who started this project and made it possible for me to join and continue from his work.

I am very grateful to the entire AR team, Songyuan Lu, Steven Hui and Capalina Melentyev for supporting me during my time at the lab. Their contributions to various discussions we had as a team helped me design experiments appropriately. Both Songyuan and Steven, still a part of the team, would gladly offer their help whenever I needed their assistance in planning and executing tests. They are great researchers, and I look forward to seeing how they will shape the project in the future.

I would also like to thank Brian Li and Po-han Chen for creating a friendly and positive environment to work in. I have benefited greatly from their guidance, and it was a true pleasure working with them.

I am thankful to my good friend, Shivaramakrishna Srinivasan, for being kind enough to proofread my thesis, ensuring that every chapter was comprehensible and easy to understand.

Lastly, I would like to thank my family and friends who motivated me and encouraged me throughout my time at UC San Diego and made it possible for me to come this far.

Chapters 3, 4 and 5 in part, are a modified version of the material from Tsui, Darin., Melentyev, Capalina., Rajan, Ananya., Kumar, Rohan., & Talke, Frank E., An Optical Tracking Approach to Computer-Assisted Surgical Navigation via Stereoscopic Vision, 2023. The dissertation author was one of the co-authors of the paper.

ABSTRACT OF THE THESIS

Optical Tracking via Stereo-vision for Surgical Navigation

by

Ananya Rajan

Master of Science in Bioengineering

University of California San Diego, 2024

Professor Frank E. Talke, Chair
Professor Geert Schmid-Schoenbein, Co-Chair

Recent advances in cutting-edge computer-assisted surgical navigation systems resolve the critical need for safer and more efficient surgical procedures. State-of-the-art navigation systems incorporate invasive retro-reflective markers that can be detected with the help of infrared cameras, providing real-time guidance to clinicians about patient anatomy and instrument trajectory. While these systems have revolutionized computer assisted surgeries, the cost ranges from $250,000 to 500,000, making it prohibitively expensive for small healthcare centers.

Optical tracking, an alternative to infrared tracking, is explored in this paper. It relies on visible light to be able to track specific objects or markers. Using two low-cost web cameras, a stereoscopic camera was calibrated and programmed to detect the 3D position of moving fiducial ArUco markers, non-invasive tags that store positional information. First, using a positioning platform, the markers were moved in the X and Y directions with the objective of tracking minor movements of patients during surgery. Next, a marker attached to a surgical needle was moved in 5DOF using the Arduino Tinkerkit Braccio Robotic Arm to mimic the handling of surgical instruments. Movements were recorded and the videos were processed in 3 color spaces, red-green-blue (RGB), hue-saturation-lightness (HSL) and hue-saturation-value (HSV) on OpenCV to calculate the tracking accuracy of the cameras. On the positioning platform, differently colored markers (white, pink, orange and yellow) were tested for the highest accuracy of detection. Using various post-processing techniques, it was found that an accuracy of 5.5 mm could be achieved in the RGB color space using a white colored marker. When the marker was moved using the robot, it was possible to obtain 2.27 mm accuracy in the RGB color space with a white colored marker.

Chapter 1

SURGICAL NAVIGATION SYSTEMS

1.1 Surgical Navigation Systems

Surgical navigation combines medical imaging and visualization, providing surgeons with positional information about the patient and instruments on an auxiliary display. This allows operators to plan the trajectory of instruments in real-time as they penetrate the skin [1]. Some systems provide guidance to the clinician during procedures, while others can perform parts of the procedure, such as when combined with robotic navigation [2]. They have been used since the 1990s [3] offering advantages such as the possibility of minimally invasive procedures and increased precision during operation [4]. For example, in a study performed to compare conventional fluoroscopy-based and navigation-based pedicle screw placement, it was found that the former resulted in 16% misplacement rates, while the latter only in 4.6% [5].

Surgical navigation is achieved by using trackers (known as reference arrays) that help identify patient and instrument location in space, which is then displayed on a screen. There are various tracking methods, but in existing systems, the trackers commonly used are reflective infrared spheres which can be localized when illuminated by infrared light. The workflow of surgical navigation consists of preoperative imaging and planning, registration, and intraoperative navigation. Patient scans are collected during preoperative imaging, and the intervention is planned. Registration refers to trackers being visible on the patient scans and on the navigation system so that they can be matched to each other. In the final step, tracked instruments can be navigated on the scans using a computer, enabling surgeons to plan the path of intervention to reach the location of interest [6].

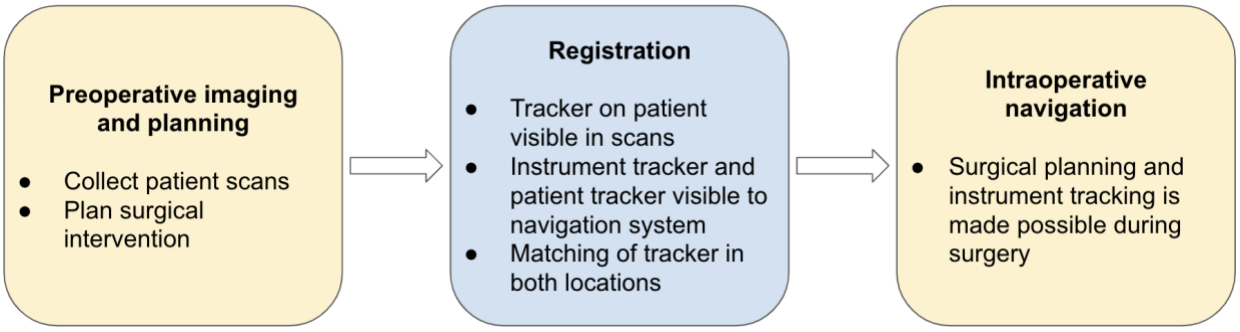


Figure 1.1: Surgical navigation workflow comprising preoperative imaging and planning, registration, and intraoperative navigation (adapted from [6])

1.2 Back Pain and the Need for Surgical Navigation in Pain Management

Research indicates that as many as 23% of adults globally experience chronic low back pain, with one-year recurrence rates between 24% and 80% [7]. While some of the causes of lower back pain are non-specific, conditions such as intervertebral disc disease are the cause of lower back pain in 26-42% of the patients and result in disc herniation [8, 9]. For such conditions, treatments like pain medication, physiotherapy, epidural steroid injection and, in more serious cases, surgical intervention are prescribed and performed.

Epidural steroid injection is a non-surgical treatment for lower back pain arising as a result of herniated disc, degenerative disc disease, or spinal stenosis. During the 30-minute procedure, a steroid is injected into the dural sac, the epidural space surrounding the spinal cord. The clinician relies on fluoroscopy to accurately advance the needle to the target of interest. The pain relief stemming from epidural steroid injection is temporary, and thus, for long-term treatment, more of such procedures are recommended [10]. Further procedures mean an increase in exposure to radiation for the patient and the clinician who performs multiple such procedures a day.

In spine surgeries, computer-assisted navigation has been gaining momentum rapidly with increased adoption and technical advancements [11]. The next section details various systems and their technology that are in use today.

1.3 Current Surgical Navigation Systems in the Market

NuVasive's (San Diego, CA, USA) Pulse Platform offers the potential to enhance the accuracy of spine surgeries through intraoperative image-based guidance. With the help of infrared (IR) reflective markers affixed to surgical instruments, along with fiducial infrared markers anchored to bony locations through an incision in the back, precise tracking of instruments by an IR camera is made possible. This tracking mechanism delivers real-time information to the surgeon, offering insights into the depth of penetration, and the potential trajectory of the instruments. Subsequently, the trajectory of these instruments can be superimposed onto 3D radiographic images. The resulting mapped trajectory can be viewed on a separate monitor during the surgical intervention. The Pulse Platform costs anywhere between $365,000 and $505,000 [12].

Figure 1.2: NuVasive's Pulse Platform's instrument reference array [13]

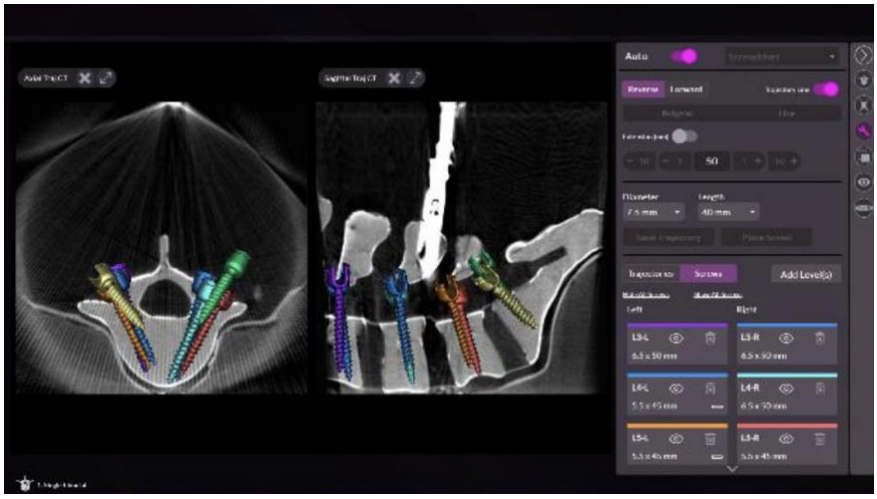


Figure 1.3: Pulse Platform's Navigation Software displaying screw trajectories [13]

With the idea of avoiding additional incisions in the skin to insert markers, Stryker Navigation (Stryker Corp, Kalamazoo, Michigan) released a system for pedicle screw placement procedures wherein a non-invasive adhesive rectangular marker, the SpineMask, equipped with 31 infrared LEDs would be applied to the patient's back to serve as a reference for the patient's position. If the patient's position changes, or if there is deep muscle retraction, the system compensates for the deformation. Smaller markers, with IR LEDs attached, would be affixed to instruments to track them. Their proprietary camera, using active optical technology, is capable of detecting the infrared LEDs, and thus the movement of instruments. Akin to the Pulse Platform, information on the instruments is overlaid on a CT image projected on a separate monitor. The cost of Stryker's spine navigation solution starts from $215,000 and can go up to $350,000 [12].

Figure 1.4: Stryker's surgical navigation solution showing the SpineMask non-invasive tracker on the back and the SpineMap 3D navigation software [14]

7D Surgical was the first company to use 'machine vision navigation' technology in spinal surgical navigation. Flash Navigation is priced at about $450,000 and uses a stereoscopic video camera to obtain a 3D image of the patient's surface anatomy. This image is integrated with a CT scan made either before or during the surgery, and various depths in the anatomy are calculated. Additionally, an infrared camera is used to track surgical tools, and images are projected real-time showing trajectories of the many instruments to be used. Augmented reality (AR) can optionally be used in this system to visualize safe zones of the pedicle screw trajectories, for example, when tools cannot be tracked [15].

Figure 1.5: 7D's Flash Navigation system [16]

The Augmedics XVision is a complete AR navigation system that allows the user to view an overlaid CT image over the patient while wearing a wireless headset with near eye display [17]. The headset is customized to match the user's focal length. First, a registration marker is affixed to a part of the spine and a CT scan is obtained and 3D segmented. The registration marker is then swapped for navigation markers - these, in addition to the reference CT image aid instrument tracking through the method of triangulation. The price of this product is around $160,000 and it can be used for open, as well as minimally invasive surgeries [18].

Figure 1.6: Augmedics XVision based on augmented reality [19]

Table 1.1: Summary of existing technologies

Company/ Product	Technology	Display	Application	Cost
NuVasive - Pulse Platform	IR reflective markers, external display	External	Spine surgeries	$365,000 - $505,000
Stryker - SpineMask	IR LED mask on patient, external display	External	Spine surgeries	$215,000 - $350,000
7D Surgical - FLASH Surgical	Machine Vision for creating 3D image mapping, external display, AR (optional)	External	Spine and cranial navigation	$450,000
Augmedics - XVision	Infrared markers, Augmented Reality, Head Mounted Display	Head Mounted Display	Neuronavigation, pedicle screw insertion	$160,000

1.4 Research in Tracking Methods for Surgical Navigation

Researchers have been developing different tracking techniques for surgical navigation, such as mechanical tracking, electromagnetic tracking, ultrasound tracking, and others [20].

Electromagnetic (EM) tracking can overcome the limitation of requiring constant marker visibility, a necessity for optical tracking. EM tracking works by modeling the disturbances of the magnetic field between a sensor and the object being tracked [21]. Monotonic variations in magnetic fields along the X, Y, and Z axes were generated to establish magnetic field gradients. Sharma et al have developed microchips to perceive these gradient fields [22]. By attaching one such microchip to an implant within the body and another to a surgical tool, the devices concurrently measured and relayed magnetic field information from their respective locations to an external receiver. Through this method, they achieved a localization accuracy of <100µm.

For ultrasound guided surgeries, Stoll et al. have utilized ultrasound imaging to track the tip of instruments by developing a passive marker in the form of a sleeve to slip onto instruments [23]. Ridges on the sleeve follow a specific pattern, and when imaged by an ultrasound probe, they provide the position of the instrument tip. On processing the marker images, a positional accuracy of 0.22 mm was achieved.

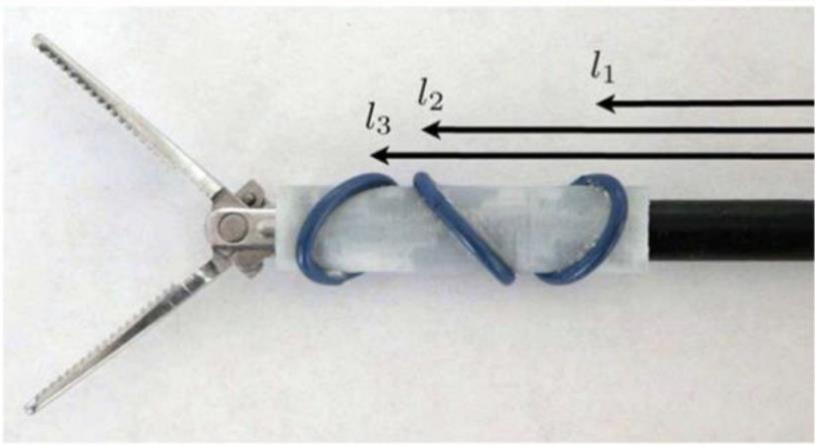


Figure 1.7: Passive ‘sleeve’ type marker attached to instrument [23]

Optical tracking systems are considered the gold standard for minimally invasive surgeries [24]. The technology used can be categorized into infrared (IR) and videometric tracking. IR systems can either use reflective spheres that are illuminated by IR light, or an IR light-emitting diode (LED) which can be captured with a charge coupled device (CCD). In videometric tracking, instead of using IR markers, light from the visible range is employed to identify markers which are tracked by applying computer vision algorithms to real-time videos. This method provides an accuracy of 0.2 - 0.5 mm [24, 25].

Table 1.2: Summary of tracking methods and accuracies

Tracking Method	Accuracy
Electromagnetic	<100μm [22]
Ultrasound	0.22 mm [23]
Optical	0.2 - 0.5 mm [24,25]

1.5 Limitations of Existing Methods

As seen in Table 1.1, the cost of current state-of-the-art navigation systems is exorbitant, especially due to the use of IR. Smaller healthcare organizations are hence unable to afford such products. Another drawback in systems like the Flash Surgical, and NuVasive's Pulse platform is that they require the surgeon to look up at a screen away from the patient, which does not allow for seamless workflow during surgery. Lastly, many of the systems in the market are dependent on fluoroscopy due to its high resolution and ability to provide great contrast when using dyes.

The disadvantage is that it emits ionizing radiation, a cause of concern for patients and the clinical team as it can lead to the development of cancer. Especially in spinal surgeries, the radiation exposure to the operator is very high, and close to 12 times that of other procedures [26]. With respect to research in tracking methods, in electromagnetic tracking, the movement of nearby metallic objects and devices causes distortion, leading to error accumulation [25]. Ultrasound tracking, on the other hand, is susceptible to error from image artifacts caused by specular reflection when instruments are imaged along with markers [27]. Optical, or videometric tracking, is expensive and requires constant line of sight of markers, despite being the most commonly used [28].

The solution proposed in this thesis is the use of augmented reality in surgical navigation systems with real-time overlay of MRI scans and instrument trajectory planning. The system is made cost-friendly by utilizing web-cameras to facilitate optical tracking, and non-invasive fiducial skin-markers called ArUco markers to paste on the skin of the patient. ArUco markers, too, are easily available at a low cost. The cameras are programmed to detect the markers and locate the patient with OpenCV's open-source ArUco library. When the patient's pose (position and orientation) is known, an augmented-reality MRI scan can be overlaid above the patient, providing visual, real-time guidance to the clinician. The use of MRI reduces the need for heavy fluoroscopy during surgery, thus preventing much radiation exposure. Additionally, these markers are also attached to surgical needles to calculate their position with the goal of instrument trajectory planning. 3D dice, with an ArUco marker on each face of the die, are used as an alternative to planar ArUco markers during instrument tracking to enable continuous marker visibility for constant instrument detection.

Chapter 2

OPTICAL TRACKING WITH COMPUTER VISION

2.1 Overview

This thesis focuses on the feasibility of using optical tracking for surgical navigation systems by employing concepts of computer vision to calibrate and program a stereoscopic camera. The camera is built for the optical tracking of ArUco markers placed on the skin of the patient and on the instrument to be tracked. It is first calibrated to adjust for its inherent distortion and programmed on OpenCV to detect the markers. Tests are performed to verify the stereoscopic camera's accuracy in detecting moving ArUco markers in 2DOF and 5DOF.

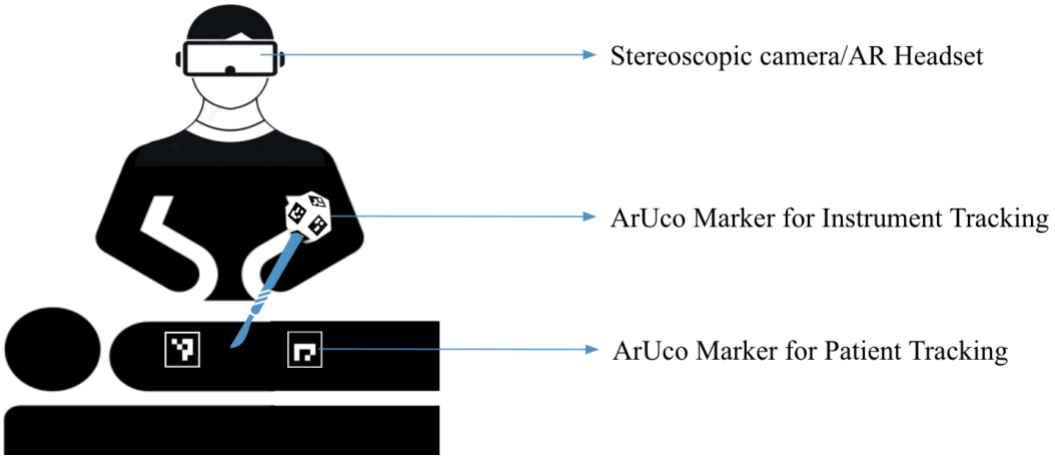


Figure 2.1: Design of Surgical Navigation System

2.2.1 ArUco Markers

ArUco markers are binary square fiducial markers used for detection in computer vision. They have a black border surrounding an inner matrix comprising black and white squares forming several unique patterns. Each pattern is defined by a marker ID, ranging from ID 0 to 999, and when detected by a camera, provides information about its position and orientation (pose) [29].

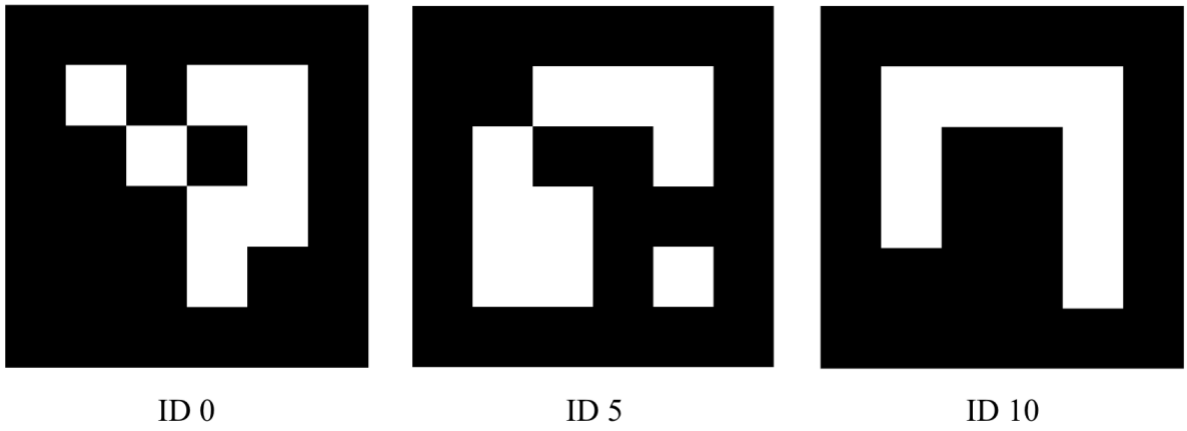


Figure 2.2: ArUco Markers with different patterns and IDs

ArUco markers are generally 2D planar markers as seen in figure 2.2. It is possible to locate a patient lying on the OR table by pasting these markers on the patient's skin. When the camera detects the markers, it receives positional information about the markers, which in turn helps determine the same for the patient.. Similarly, its use can be expanded into tracking surgical instruments when an ArUco marker is attached to the end of the instrument to be tracked. Since an instrument could potentially move in 6DOF, the planar ArUco marker cannot be visible to the camera at all times. To circumvent this issue and establish an uninterrupted line-of-sight, a 3D n-sided ArUco die has been created. Each side of the dice contains an ArUco marker, as seen in figure 2.3. With a 3D marker attached to a surgical instrument, such as a scalpel, the instrument can be constantly tracked leaving no gaps in detection.

Figure 2.3: 3D ArUco Markers for continuous detection of instruments

2.2.2 Stereoscopy, Stereovision and the Stereoscopic Camera

Stereoscopy is an imaging technique that puts 2D images of a single scene together to create the sensation of depth for 3D perception. With a single camera (monocular vision), it is impossible to estimate depth. Stereo vision relies on two cameras, each capturing an image from the left and right perspectives of a scene, to create a 3D representation. A stereoscopic camera (also called ‘stereo camera’) is a set of two cameras that work in tandem to reconstruct a single image that is a 3D version of the scene they are viewing.

2.2.3 Disparity

The two cameras are separated by a certain distance on the X axis, but not on the Y or Z axes. Stereo vision algorithms that program cameras search for a pair of pixels, or pixel blocks in the two images that describe the same point (in this case, the center of the visible ArUco marker). There are many ways to determine pairs of pixels called ‘matching’ algorithms. In this thesis, matching is performed by directly comparing one pixel of a frame with one pixel of another frame Once a pixel conjugate is found, the horizontal difference, known as disparity, is estimated. This

process is simplified since the pixel coordinates vary only in the X-axis, thus reducing the 2D matching problem to a 1D problem. Next, by applying triangulation, the depth of the point can be estimated [30]. The 3D coordinates of the center of the detected ArUco marker, described by (x_{left}, y_{left}) on the left camera and (x_{right}, y_{right}) on the right camera, are calculated as follows:

$$z = \frac{fb}{d} \quad \ldots\ldots (2.1)$$

$$x = \frac{x_{left} z}{d} \quad \ldots\ldots (2.2)$$

$$y = \frac{y_{left} z}{d} \quad \ldots\ldots (2.3)$$

where d is the disparity given by either $d = |x_{left} - x_{right}|$,

f is the focal length of the cameras

b is the baseline distance between the two cameras

2.2.4 Calibration

The purpose of calibration is to map 3D world coordinates to 2D image coordinates and remove any distortion in the images. Calibration is performed in four steps: (a) converting world coordinates (X_w, Y_w, Z_w) to camera coordinates (X_c, Y_c, Z_c) to image coordinates (u, v), (b) undistorting images, (c) stereo calibration, and (d) stereo rectification.

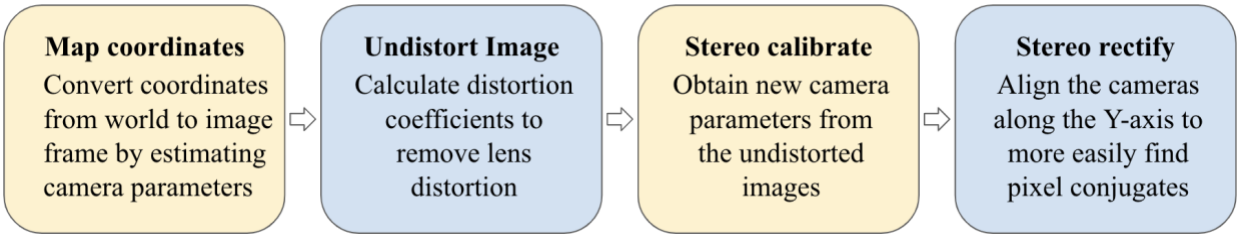


Figure 2.4: Steps to calibrate camera

Converting world to camera coordinates relies on extrinsic parameters of the camera, which are rotation (R) and translation (T) matrices. These define the camera's pose with respect to the real world. Converting camera to image coordinates uses intrinsic parameters of the camera, which

are its focal length (f_x, f_y) and optical center (c_x, c_y). The output is a projection matrix that maps 3D world coordinates to 2D image coordinates [31, 32].

The checkerboard pattern is commonly used for calibration. It is placed at different positions and orientations for the cameras to capture. Following Zhang's method of calibration [33], a pinhole camera model is assumed for the cameras since it provides a linear relationship between 3D and 2D coordinates. The advantage of using a linear model is that the projection matrix formed can be easily deconstructed as needed, as will be described next.

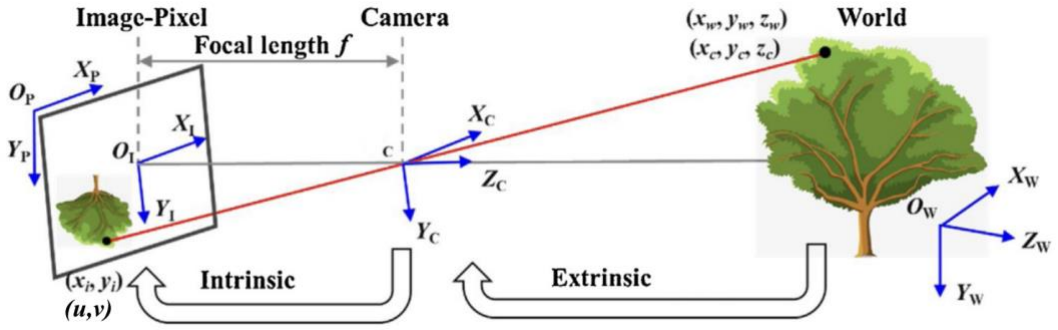


Figure 2.5: Pinhole Camera Model [34]

When the corners of all squares are detected, their 3D world coordinates can be matched with their corresponding 2D image coordinates since the checkerboard pattern is simple and repetitive.

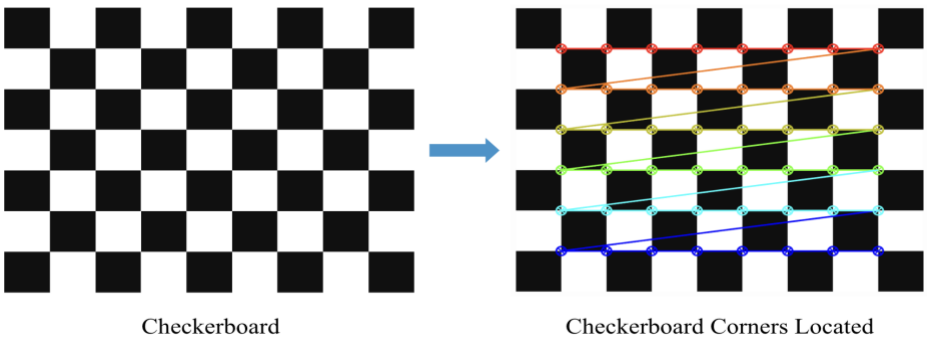


Figure 2.6: Finding corners in the checkerboard pattern for calibration

The projection matrix, ‘P’, thus formed (eqn. 2.4) is decomposed into the camera’s extrinsic and intrinsic parameter matrices using the method of QR factorization (eqn. 2.5 and 2.6) [35].

$$\underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\text{Image Coordinates}} = \underbrace{\begin{bmatrix} p_{11} & p_{21} & p_{31} & p_{41} \\ p_{21} & p_{22} & p_{32} & p_{42} \\ p_{31} & p_{32} & p_{33} & p_{42} \end{bmatrix}}_{\substack{\text{Projection} \\ \text{Matrix} \\ \text{(P)}}} \underbrace{\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}}_{\text{World Coordinates}} \quad \ldots\ldots (2.4)$$

P from the above equation is decomposed as follows,

$$\underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}}_{\substack{\text{Projection} \\ \text{Matrix} \\ \text{(3 x 3)}}} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix}}_{\substack{\text{Intrinsic} \\ \text{Matrix}}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}}_{\substack{\text{Rotation} \\ \text{Matrix} \\ \text{(R)}}} \quad \ldots\ldots (2.5)$$

$$\underbrace{\begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}}_{\substack{\text{Projection} \\ \text{Matrix} \\ \text{(3 x 1)}}} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix}}_{\substack{\text{Intrinsic} \\ \text{Matrix}}} . \underbrace{\begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}}_{\substack{\text{Translation} \\ \text{Matrix} \\ \text{(T)}}} \quad \ldots\ldots (2.6)$$

The extrinsic matrix is formed from the R and T matrices. Camera coordinates are calculated with extrinsic parameters using the equation below,

$$\underbrace{\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}}_{\text{Camera Coordinates}} = \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Extrinsic Matrix}} \cdot \underbrace{\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}}_{\text{World Coordinates}} \quad \ldots\ldots (2.7)$$

With the camera coordinates and intrinsic parameters, the image coordinates can be derived as follows,

$$\underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\text{Image Coordinates}} = \underbrace{\begin{bmatrix} f_x & 0 & c_y & 0 \\ 0 & f_y & c_x & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{Intrinsic Matrix}} \cdot \underbrace{\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}}_{\text{Camera Coordinates}} \quad \ldots\ldots (2.8)$$

Equations 2.7 and 2.8 can also be written as

$$\underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\text{Image Coordinates}} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix}}_{\text{Intrinsic Matrix}} \cdot \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{\text{Extrinsic Matrix}} \cdot \underbrace{\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}}_{\text{World Coordinates}} \quad \ldots\ldots (2.9)$$

With equation 2.9, mapping of 3D to 2D coordinates is complete. This needs to be repeated for the second camera as well. Ultimately, there will be two rotation and translation matrices for the left and right cameras (R_{right}, R_{left}, T_{right}, T_{left}).

The next issue to address is the impact of lens distortion on the images, which is applicable to the web cameras since they have a lens. Two types of distortion are considered - radial and tangential. Radial distortion is caused as a result of light rays bending more near the edges of the lens than at the center. Tangential distortion on the other hand is a result of the lens not being parallel to the scene being viewed.

The pinhole camera model does not have a lens, and so does not account for distortion, which is a non-linear effect. Instead of using a different camera model to factor in distortion, existing equations can be modified to model distortion and produce distortion coefficients - these remove distortion in images that are considered to be 'black' areas carrying no information [36, 37]. The distortion coefficients thus used to rectify the effects of distortion are radial (k_1, k_2, and k_3) and tangential (p_1, p_2) distortion coefficients.

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$r^2 = x_{distorted}^2 + y_{distorted}^2$$

where ($x_{distorted}$, $y_{distorted}$) are the distorted coordinates in pixels, (x, y) are the radially undistorted coordinates in the image, and k_1, k_2 and k_3 are the radial distortion coefficients.

To remove tangential distortion, the following formula is implemented:

$$x_{distorted} = x + 2p_1 xy + p_2(r^2 + 2x^2)$$

$$y_{distorted} = y + 2p_2 xy + p_2(r^2 + 2y^2)$$

$$r^2 = x_{distorted}^2 + y_{distorted}^2$$

where ($x_{distorted}$, $y_{distorted}$) are the distorted coordinates in pixels, (x, y) are the tangentially undistorted coordinates in the image, and p_1 and p_2 are the tangential distortion coefficients.

Once this is done, stereo calibration is performed - it provides the spatial relationship between the two cameras [38] based on the undistorted images. Its output is a new set of extrinsic and intrinsic camera parameters, called the essential matrix and fundamental matrix, respectively.

The essential matrix(E) gives the relation between the real world-coordinates of a point P with respect to the left and right cameras (P_l and P_r) as follows,

$$P_r^T . E . P_l = 0$$

The fundamental matrix (F) relates pixel coordinates of the left and right image (p_l and p_r) to each other.

$$p_r^T . F . p_l = 0$$

Using the outputs of stereo calibration, the fundamental and essential matrices, stereo rectification is applied to the images to ensure that they are at the same position on the Y-axis. Similar rows from each image are compared and aligned. This is done so that the images lie on the same plane and have no difference in height [39].

The calibration process provides camera parameters and the distortion coefficients for each of the cameras. With this process complete, the stereo camera is capable of tracking ArUco markers by computing the disparity between its two images.

Chapter 3

EXPERIMENTAL SETUP

3.1 Stereoscopic Camera Setup

The stereo camera used for this research project was initially built using two HZDQLN HD web cameras which had a resolution of 720p and frame rate of 30 fps, which cost $40 each. For later experiments, Logitech C920x HD Pro cameras were used which had a resolution of 1080p and a frame rate of 30 fps, costing $70 each. The cameras were accessed via OpenCV, an open-source computer vision library in Python.

Figure 3.1: Stereoscopic Camera

The cameras were separated by a baseline distance of 123 mm and were placed at the same height from the ground. Other distances from 90 mm through 130 mm were tested, but the highest tracking accuracy could be achieved only with 123 mm baseline distance. This is related to the lens's focal length - higher the focal length, larger is the baseline distance.

3.2 Calibration

Calibration was performed using OpenCV. 30 sets of left and right images of a 9x6 checkerboard were captured at different angles and distances using the stereo camera. The calibration steps explained in section 2.2.2 were implemented to obtain and store intrinsic and extrinsic parameters and the distortion coefficients of the camera.

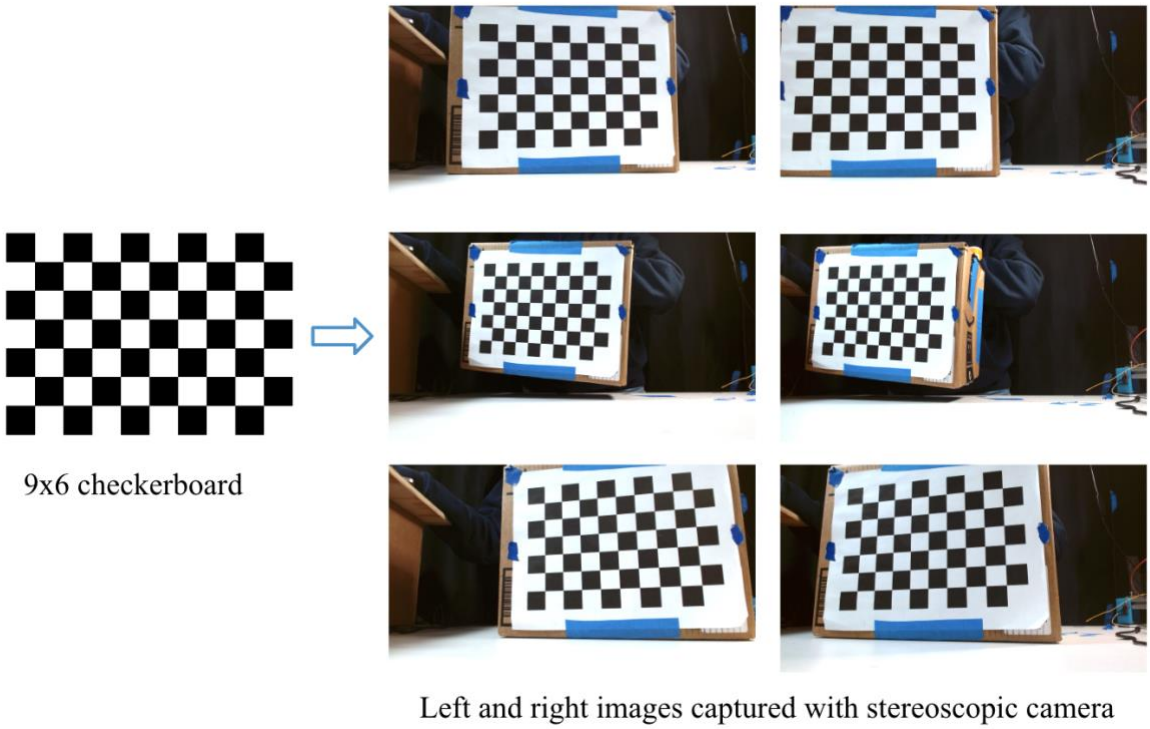


Figure 3.2: Images for calibration

Best practices for calibration include using a checkerboard that has an odd number of squares on one side and an even number on the other. It is important to ensure that the entire checkerboard is visible to both cameras with no squares being cut off from their vision.

3.3 Marker detection

Marker tracking provides the 3D position of any marker that is detected by the camera. The code was written in Python using various functions from the OpenCV and ArUco libraries.

The camera records left and right videos as the marker moves along a pattern (square or 3D circle). Marker detection begins with loading these videos and converting them into frames. Each frame is remapped to a new frame by applying camera and distortion coefficients using the remap() function.

Next, with cv.detectMarkers(), the markers in the left and right frames can be detected, and then, their centers estimated (center_point_right and center_point_left). The centers are used to calculate the disparity between the two frames, as discussed in section 2.2.2. The inputs required for calculating disparity are the x-coordinates of the marker in both frames (x_l and x_r), the baseline distance between the two cameras (b), and their focal length (f). Finally, the x, y, and z coordinates of the marker in the world frame are found at every instance of its detection.

Algorithm 1 Marker tracking

Inputs: Left and right videos, calibration coefficients, baseline distance (b), focal length (f)

```
Convert videos to frames (frame_l, frame_r)
Remap frames with cv.remap() and calibration coefficients
Load required ArUco dictionary
for each set of frame_l, frame_r do
    Detect marker using cv.detectMarker() and ArUco dictionary
    if markers detected then
        Calculate marker center (x_left, x_right, y_left, y_right)
        d = x_left - x_right
        z = fb/d
        x = (x_left * z)/d
        y = (y_left * z)/d
    end if
end for
```

3.4 XY Positioning Platform

A linear positioning platform, capable of moving in the X and Y directions was used to move the marker in a 2D fashion [40]. The purpose of using the platform was to mimic minor movements of the marker caused by a patient's involuntary movement during surgery, for example, due to shivering, breathing or reacting to needle insertion. The size was 50 cm x 50 cm, which is suitable for simulating humans since the biacromial width for men and women does not, in general, exceed 41.1 cm and 36.7 cm, respectively [41]. It has two stepper motors controlled by an Arduino, allowing for linear movement. The platform moves at a speed of 20 mm/s and completes a square pattern. A single marker is attached to its center. Four ring lights are placed above the platform to provide illumination for the marker to be visible to the camera.

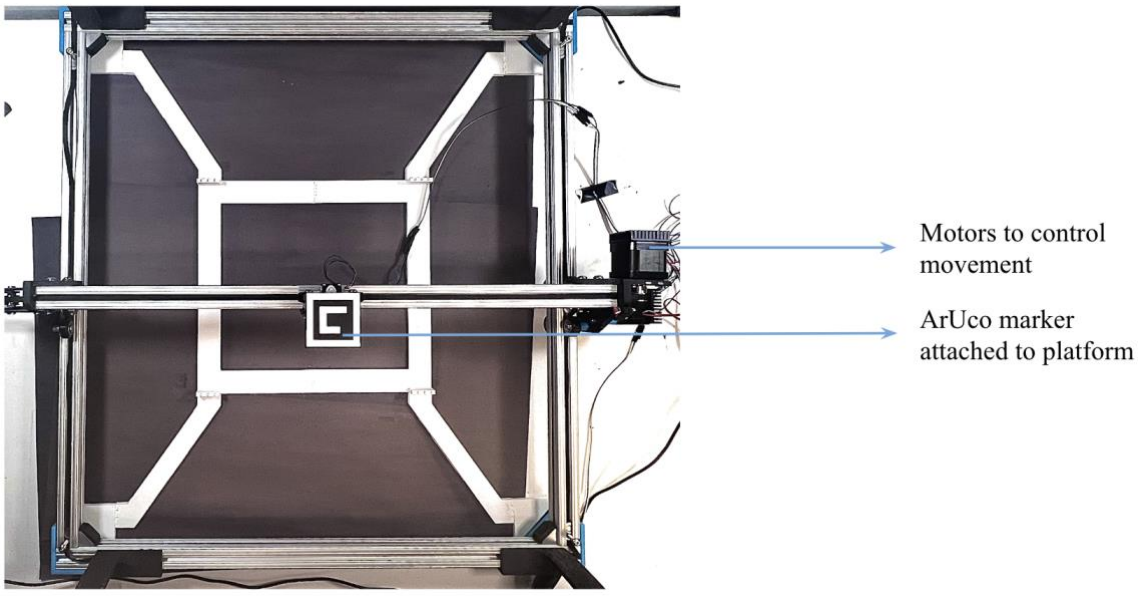


Figure 3.3: The XY positioning platform

3.5 Processing Experimental Data from the XY platform

For processing experimental data generated from the XY platform, a technique called homography was implemented. Homography maps coordinates from one 2D plane to a different 2D plane; here, it would map 2D camera coordinates to corresponding coordinates in the world frame. To begin, the corners of the square pattern in the world frame and in the image frame need to be specified using 2D coordinates (X and Y). With this, a homography matrix (H) is developed.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x'_1y_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x'_2y_2 & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x'_3y_3 & -y_3y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -y_4x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x'_4y_4 & -y_4y'_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix} \quad \ldots\ldots (3.1)$$

Here, x_1 , y_1 , x_2 , y_2 , x_3 , y_3 , x_4 and y_4 are the camera frame coordinates of the square pattern. x'_1 , y'_1 , x'_2 , y'_2 , x'_3 , y'_3 , x'_4 and y'_4 are its coordinates in the world frame. Since both sets of values are known, the homography matrix can be derived. It can then be used to map all other coordinates from camera frame to world frame using the following equation -

$$\underbrace{\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}}_{\text{World Coordinates}} = \underbrace{\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}}_{\text{Homography Matrix}} \underbrace{\begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix}}_{\text{Camera Coordinates}} \quad \ldots\ldots (3.2)$$

With this, the distance that the marker moves in the camera frame is established in the world frame. Any deviation of this distance from the real-life distance that the marker moves across contributes to tracking error [42].

3.6 Programming the Robot

A robotic arm was assembled to hold and move a surgical instrument with an ArUco marker attached to it. It was used to add depth and rotation to marker movements since it could move in 5DOF. The robot was programmed with an Arduino by inputting the necessary joint angles and speed.

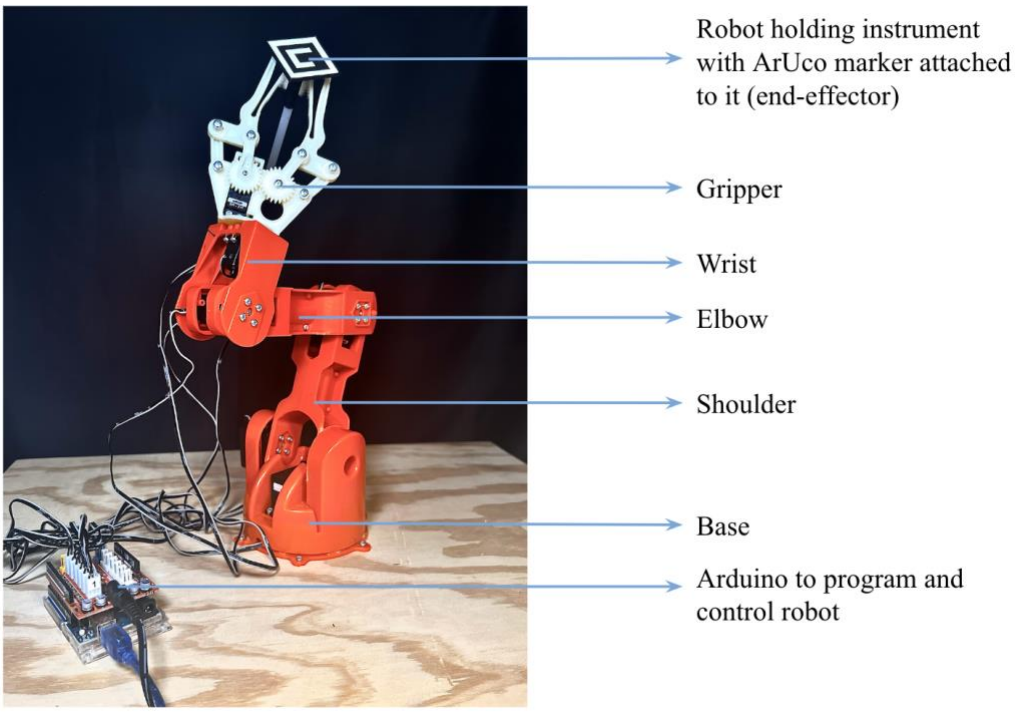


Figure 3.4: 5DOF robotic arm holding marker

Each joint (base, shoulder, elbow, wrist, and gripper) of the robot was calibrated to ensure that its motors moved the joints accurately. The joints were set to their maximum and minimum possible angles, and these were recorded as their joint limits. Similarly, a central angle was also

recorded. These angles were used to calibrate the robot - every input joint angle was fed to the robot with reference to these calibration angles.

Table 3.1: Calibration angles for robot

Joint	Minimum (°)	Maximum (°)	Center (°)
Base	0	180	80
Shoulder	15	165	85
Elbow	7	173	85
Wrist	12	168	80
Gripper	15 (close)	80 (open)	40

A MATLAB code [43] was adapted to generate the necessary joint angles from given waypoints based on inverse kinematics. The code extrapolated a trajectory for the robot from a set of waypoints. Too many waypoints meant that the robot would make sharp transitions, while too few resulted in unintended patterns. Optimal points were decided based on the output of the robot and user discretion. Joint angles were then extracted from each point the end effector traveled through using the Inverse Kinematics (IK) toolbox in MATLAB. A series of angles were generated and fed to an Arduino that was interfaced with the robot.

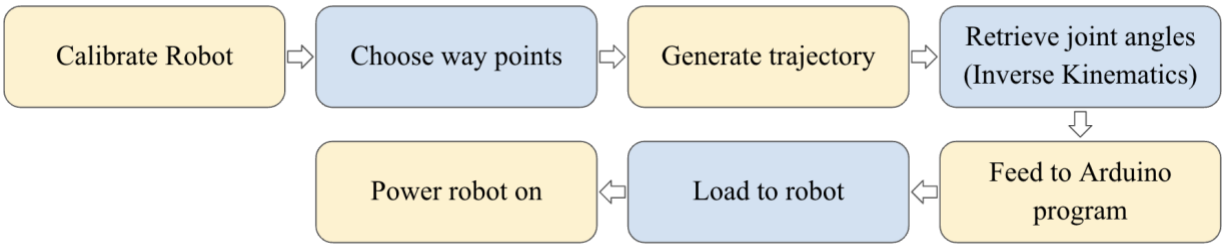


Figure 3.5: Programming robot

The Arduino library for controlling the robot [44] was developed further to accept multiple joint angles and move the end-effector. A delay of 3000 ms was added between every movement to avoid jitter that could cause the camera to capture erroneous data.

The waypoints were chosen in a way such that the marker attached to the robot would move in a 3D manner. Additionally, since the robot moves in 5DOF, any pattern chosen would cause the marker to adopt different angles with respect to the camera as it moves. This means that the marker would not always be parallel to the camera, but would oftentimes be angled away from the camera. This can be expected of an instrument being handled by a surgeon.

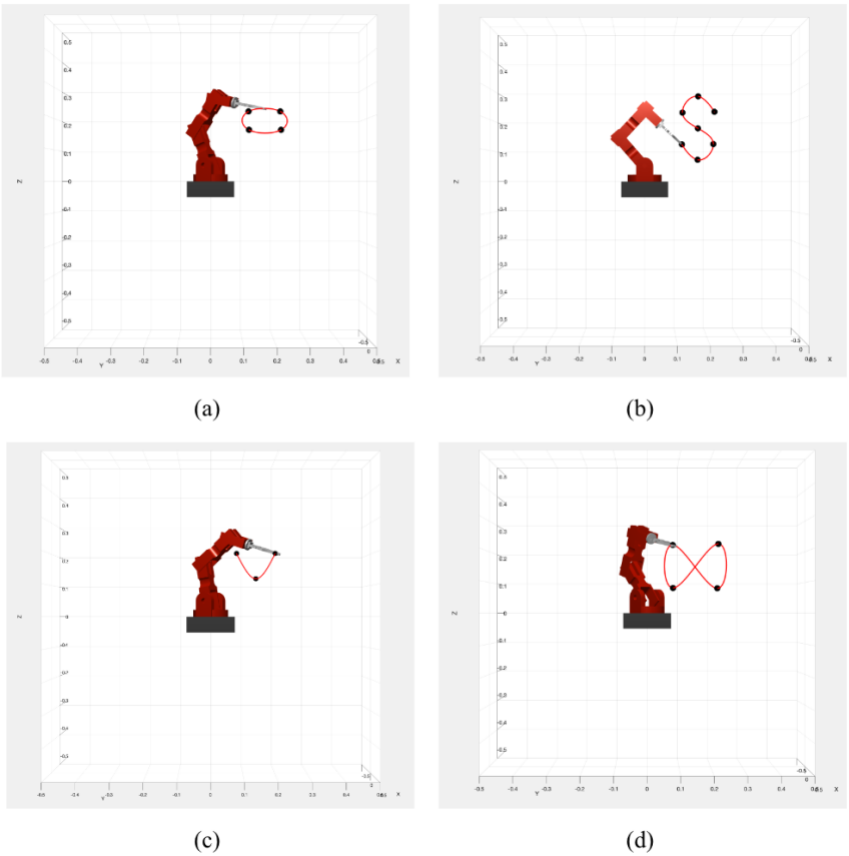


Figure 3.6: Some trajectories considered for the robot: (a) Circle (inclined), (b) ‘S’, (c) Parabola and (d) Infinity

Multiple sets of waypoints forming various shapes were investigated, but the pattern that was found to be ideal to use for verification was an inclined/3D circle. The pattern provided enough depth to perform reliable accuracy tests for depth estimation, while also being simple enough to keep the marker visible at all times.

3.7 Processing Experimental Data from the Robot

This section discusses the conversion of data collected through experiments with the robot to its corresponding real-life distances. This is accomplished by finding a scaling factor that relates experimental data (from the camera) to theoretical values (ground-truth) of the robot.

A scaling term was computed to convert coordinates derived from the camera's vision in the camera frame to the world frame. To do this, an ArUco marker was first moved in the X-direction over a known distance as the stereo camera captured the movement. This was repeated in the Y and Z directions too. For each direction, a scaling factor was calculated by comparing the x, y and z coordinates derived from equations 2.2, 2.3 and 2.1 with the actual distance the marker moved in the X, Y and Z directions, respectively.

One of the challenges faced when working with the robot data was that the waypoints that were specified in the MATLAB code (robot-frame) were not absolute real-life distances (world-frame). To transform robot-frame coordinates into world-frame coordinates, a scaling factor, different from the previous factors, had to be calculated. The method used was fairly straightforward - a linear pattern was assigned to the robot. The length of this linear movement was calculated in the robot frame by finding the difference between the start and end waypoints.

Then, the distance traveled by the robot was also measured in the world-frame. Comparing the two, it was easy to obtain the robot-to-world scaling factor. The processes described above convert all frames into the world-frame, allowing for easy comparison of data points. This is done

to calculate the deviation (error of tracking) between ground truth data and experimental data of the ArUco markers position, obtained from the camera.

Table 3.2: Scaling factors for the robot

Scaling Factor	Value
Camera to World - X axis	1.31
Camera to World - Y axis	1.33
Camera to World - Z axis	1.89
Robot to World	0.0746

3.8 Fixing an Instrument and a Marker to the Robot

An attachment was created to fix onto the end of a surgical needle and hold an ArUco marker for tracking. For different marker sizes, different sizes of attachments were printed. They were made as light as possible to prevent an increase in the load on the robotic arm. Another issue to address was the marker moving out of sight of the camera as the robotic arm moved. Placing the instrument along the length of the gripper ensured that the marker was at 90° to the end-effector, meaning that it could always be seen by the camera.

Chapter 3, in part, is a modified version of the material from Tsui, Darin., Melentyev, Capalina., Rajan, Ananya., Kumar, Rohan., & Talke, Frank E., An Optical Tracking Approach to Computer-Assisted Surgical Navigation via Stereoscopic Vision, 2023. The dissertation author was one of the co-authors of the paper.

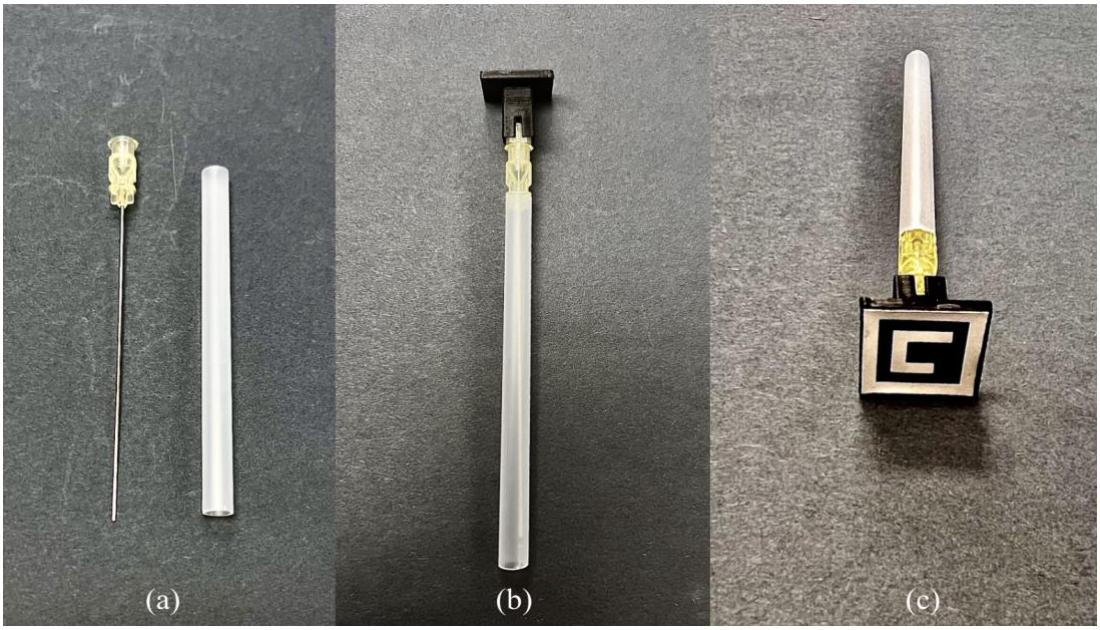


Figure 3.7: (a) Surgical Needle, (b), (c) Needle with marker attachment

Chapter 4

EXPERIMENTS

4.1 Overview of Marker Tracking Experiments

A series of experiments to quantify the accuracy of the stereoscopic camera was conducted. Variations were made in both the type of movement of markers, as well as in the type of markers used. Additionally, different processing techniques were implemented after videos of the moving marker(s) were recorded. While conclusions drawn from 2D experiments influenced some of the experiments conducted in 3D, most of the results from either set of movements are independently considered.

Table 4.1: Experimental parameters

Parameters	Value
Camera Resolution	720p, 1080p
Frame Rate	30 fps
Marker Width(s)	40 mm, 30 mm, 20 mm, 15 mm
Light Intensity (4 ring lights)	37 lux
XY Platform Speed	20 mm/s
Robot Speed (Servo Motor Speed)	0.14s/60°
Distance of Camera from Device	600 mm

4.2 2D Movements

2D maneuvers of the marker, as seen in section 3.4, were performed using the XY platform first. The camera was placed right above and parallel to the platform at a height of 60 cm from it to capture marker movements. The cameras used here were the HZQDLN cameras that had a lower resolution (720p). The marker moved along the path of a square and its diagonals, as seen in figure

4.1. Three such videos were recorded and processed to understand the camera's accuracy in tracking linear movements with no variation in depth.

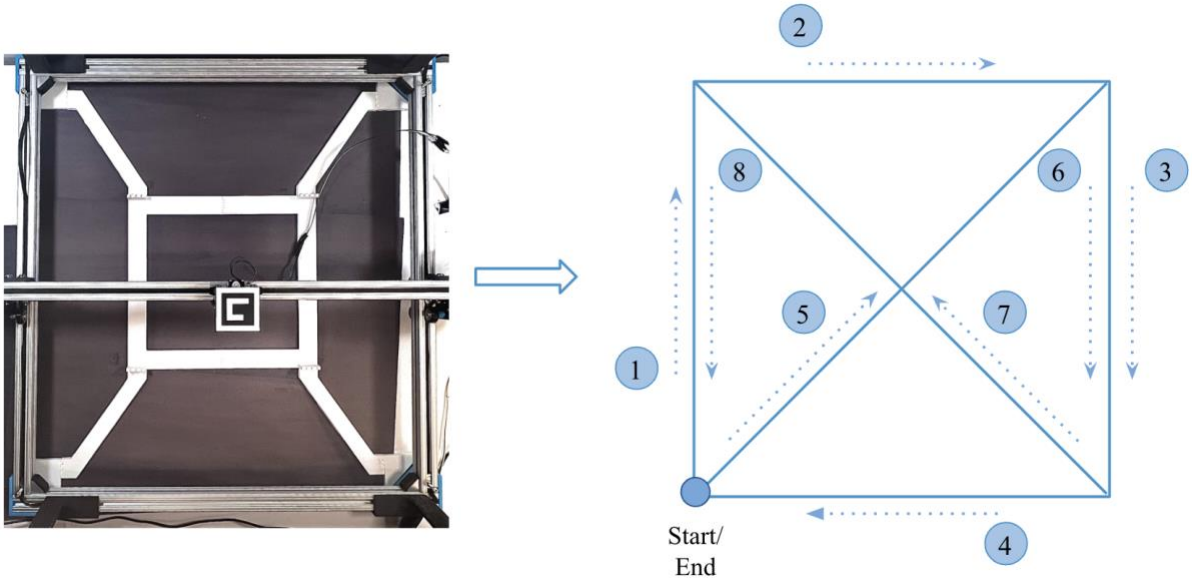


Figure 4.1: Marker movement on XY platform

4.3 3D Movements

The robot was used to provide 3D movements for tracking instruments. Here, a surgical needle was attached to the robot while it had an ArUco marker attached to its end. The camera was placed 600 mm away from the robot, and at a height of 250 mm. As in 2D movement tracking, three sets of videos of the robot's circular motion were captured and processed. The cameras used for tracking the robot movements were the Logitech C920x HD Pro cameras with resolution 1080p.

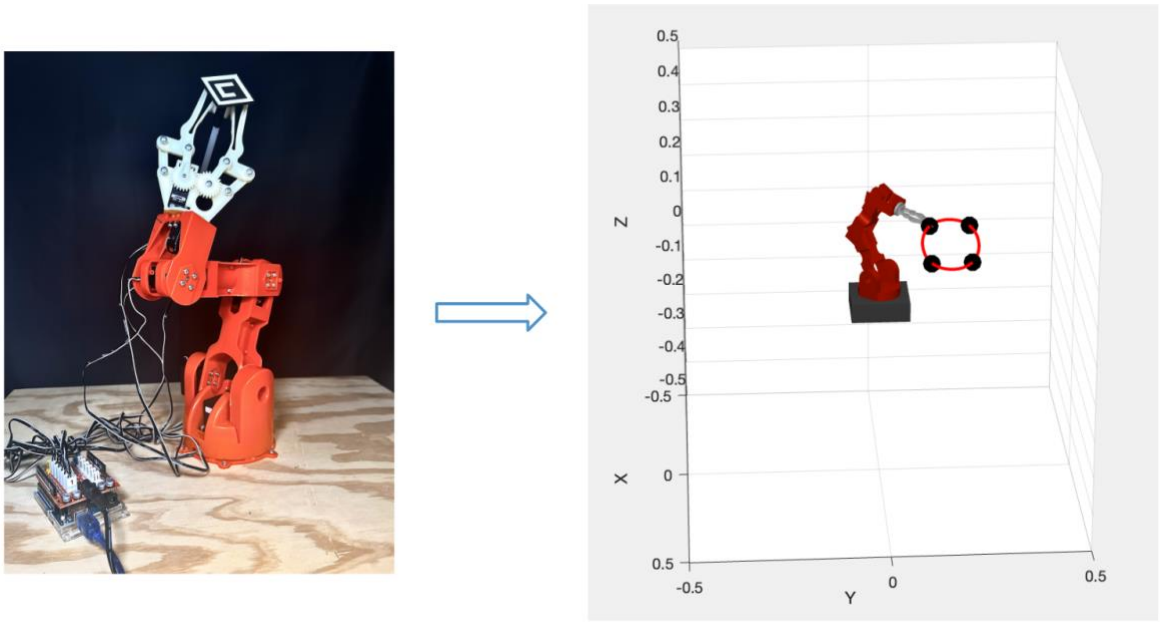


Figure 4.2: Inclined or 3D circular path

4.4 Marker Colors

The tests conducted on the XY platform were performed using markers of 4 different colors. ArUco markers are available in a standard white-and-black combination. By switching white with fluorescent pink, orange and yellow, the influence of more reflective colors on camera accuracy was tested. Considering that fluorescent colors reflect more energy than what is incident on them, unlike regular colors [45], an assumption that the camera would detect the markers more frequently was made. Another three videos of each colored marker were captured as they moved on the XY platform. These tests were not conducted on the robot since they were performed only as a way to understand if other colors could be tracked better than, or as well as the regular ArUco marker.

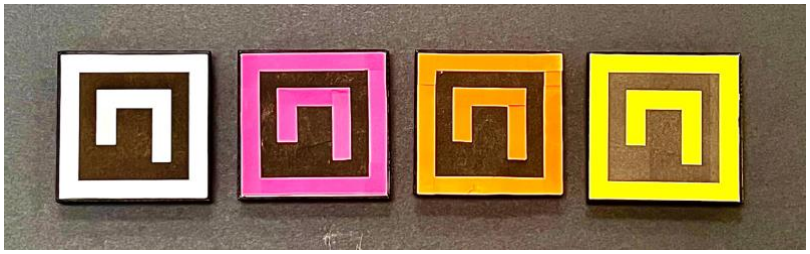


Figure 4.3: Different marker colors (white, pink, orange and yellow)

4.5 Marker Sizes

The size of a marker can greatly affect the accuracy of its detection and tracking. A bigger marker size always means higher accuracy, and a smaller one can drastically reduce the detection percentage. In this project, four marker sizes were tested, and the accuracy of tracking after processing the videos was compared. The largest marker, 40 mm x 40 mm in size was tested only on the XY platform. The remaining three (edge lengths 30 mm, 20 mm and 25 mm) were attached to the surgical instrument (and the robot for) testing.

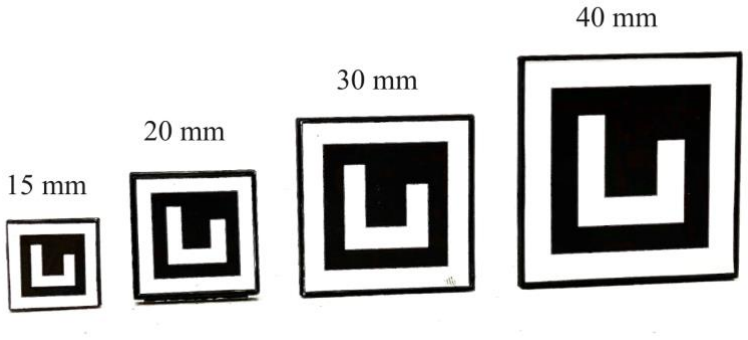


Figure 4.4: Various marker sizes tested

4.7 Color Spaces

A color space is a mathematical model of color organization in an image [47]. Three color spaces were experimented with in this project - RGB (red, green, blue), HSL (hue, saturation, lightness) and HSV (hue, saturation, value).

RGB is the most commonly used color space and is defined by red, green and blue channels. It best represents colors produced through digital means, such as with a computer. It is typically visualized in the form of a cartesian coordinate system, with each axis corresponding to one of the three colors. The colors can take up values anywhere between 0 and 255 - in combinations of different values, the channels together describe a single color [47, 48].

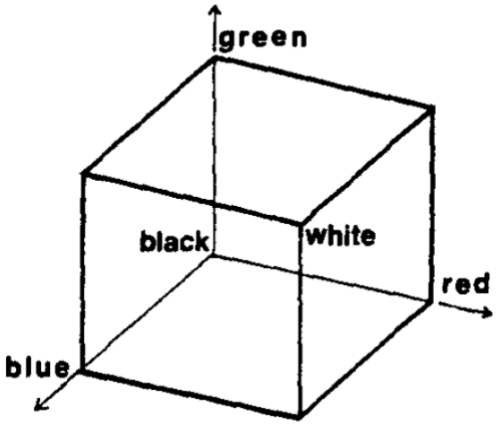


Figure 4.5: RGB color space [49]

HSV and HSL are derived from the RGB color space and are more intuitive to humans since they represent colors the way our eyes view them [46]. They have the ability to separate luma and chroma, which are light and color information, respectively, while RGB is not capable of this. This makes the color spaces independent of ambient light changes, which is useful for surgical applications [50, 51].

The HSL color space comprises the Hue, Saturation and Lightness parameters. It is depicted by a cylinder where any point on it represents a color. Hue corresponds to the angle around the central axis, saturation to the distance from the center of the cylinder, and lightness to the location along the length of the cylinder. It can be observed that white is achieved only at 100% lightness, and black at 0% lightness. The HSV color space, very similar to HSL, comprises a V

component denoting ‘Value’ and describing brightness, instead of an L component. White, in this case, is a result of 0% saturation and black, 0% value [49].

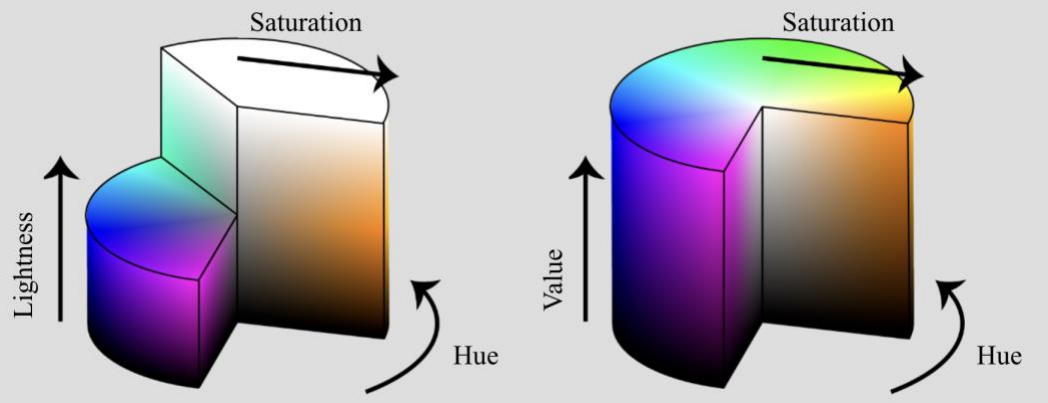


Figure 4.6: HSL (left) and HSV (right) color spaces [52]

The videos are by default in the BGR (blue, green, red) color space, which is similar to RGB, except for the order of the color channels. But, when converting to HSV or HSL, the command BGR2HSV() or BGR2HLS() needs to be applied individually to the left and right videos. By providing an upper and lower bound for a color of interest, a mask is created that thresholds videos to pass only that color [53]. For example, while tracking the white marker, thresholding will filter out all colors and allow only white to pass through. The computer vision function inRange() creates the mask when provided with a combination of three numbers representing HSL/HSV components. bitwise_and() is then applied to the original videos to threshold them.

Chapter 4, in part, is a modified version of the material from Tsui, Darin., Melentyev, Capalina., Rajan, Ananya., Kumar, Rohan., & Talke, Frank E., An Optical Tracking Approach to Computer-Assisted Surgical Navigation via Stereoscopic Vision, 2023. The dissertation author was one of the co-authors of the paper.

Algorithm 2 Color space conversion

Inputs: Left and right videos

1: Convert videos to frames (frame_l, frame_r)
2: Provide upper and lower thresholds as needed
lower_bound = [0, 0, 0]
upper_bound = [255, 255, 255]
3: Convert each frame to HSL with cv.cvtColor(frame, COLOR_BGR2HLS) or HSV with cv.cvtColor(frame, COLOR_BGR2HSV)
4: Create mask
mask = cv.inRange(converted_frame, lower_bound, upper_bound)
5: Apply threshold to frame_l and frame_r
thresholded_frame = cv2.bitwise_and(converted_frame, converted_frame, mask = mask)

Chapter 5

RESULTS AND DISCUSSION

5.1 Analysis and Processing of Videos

Videos were recorded for every experiment and the trials that were conducted. They were processed using Python in OpenCV using the computer vision and ArUco packages. After data points were generated, MATLAB was used for further post-processing and visualization of data. In either device, XY platform, or robot, it was necessary to convert camera distances derived from its videos to real-life distances to obtain theoretical (ground-truth) data as discussed in Chapter 3. The root mean square error (RMSE) was calculated between theoretical data from the platform or robot movement and their respective experimental data.

While calculating the RMSE for the robot, it was observed that the experimental and theoretical patterns were differently aligned, though in the same frame (world-frame). The same problem was not encountered with the XY platform's data, possibly due to no rotational movements or a change in depth. To align the experimental data with the ground truth, it was necessary to apply another post-processing technique, single value decomposition (SVD).
SVD minimizes the error between two sets of data, thus bringing them closer together. It takes care of both rotational and translational differences between the sets, which is otherwise difficult to calculate manually [54].

5.2 XY Platform (2D Tracking) Results

Table 5.1: Detection % of colored markers moving in the XY platform in RGB, HSL and HSV color spaces averaged over 3 trials using homography [42]

Color space	**Color**			
	White	**Pink**	**Orange**	**Yellow**
RGB	99.7	98.1	99.5	97.1
HSV	97.5	73.5	80.6	78.4
HSL	99.5	81.3	87	89.7

Table 5.2: Root mean square error (mm) of tracking colored markers moving in the XY platform in RGB, HSL and HSV color spaces averaged over 3 trials using homography [42]

Color space	**Color**			
	White	**Pink**	**Orange**	**Yellow**
RGB	5.48	5.62	5.37	12.35
HSV	5.61	6.98	5.88	6.17
HSL	5.38	6.80	6.26	5.96

5.3 Robot Experiment (3D Tracking) Results

Table 5.3: Root mean square error (mm) of tracking 30 mm, 20 and 15 mm markers in RGB, HSL and HSV at 500 mm from camera

Color Space	Marker size (mm)		
	30	**20**	**15**
RGB	2.40	2.32	2.57
HSV	2.44	2.45	2.51
HSL	2.45	2.41	2.53

Table 5.4: Root mean square error (mm) of tracking 30 mm, 20 and 15 mm markers in RGB, HSL and HSV at 600 mm from camera

Color Space	Marker size (mm)		
	30	**20**	**15**
RGB	2.27	2.48	2.53
HSV	2.27	2.56	2.53
HSL	2.32	2.51	2.55

Table 5.5: Root mean square error (mm) of tracking 30 mm, 20 and 15 mm markers in RGB, HSL and HSV at 700 mm from camera

Color Space	Marker size (mm)		
	30	**20**	**15**
RGB	2.63	2.74	2.78
HSV	2.65	2.72	2.67
HSL	2.66	2.63	2.89

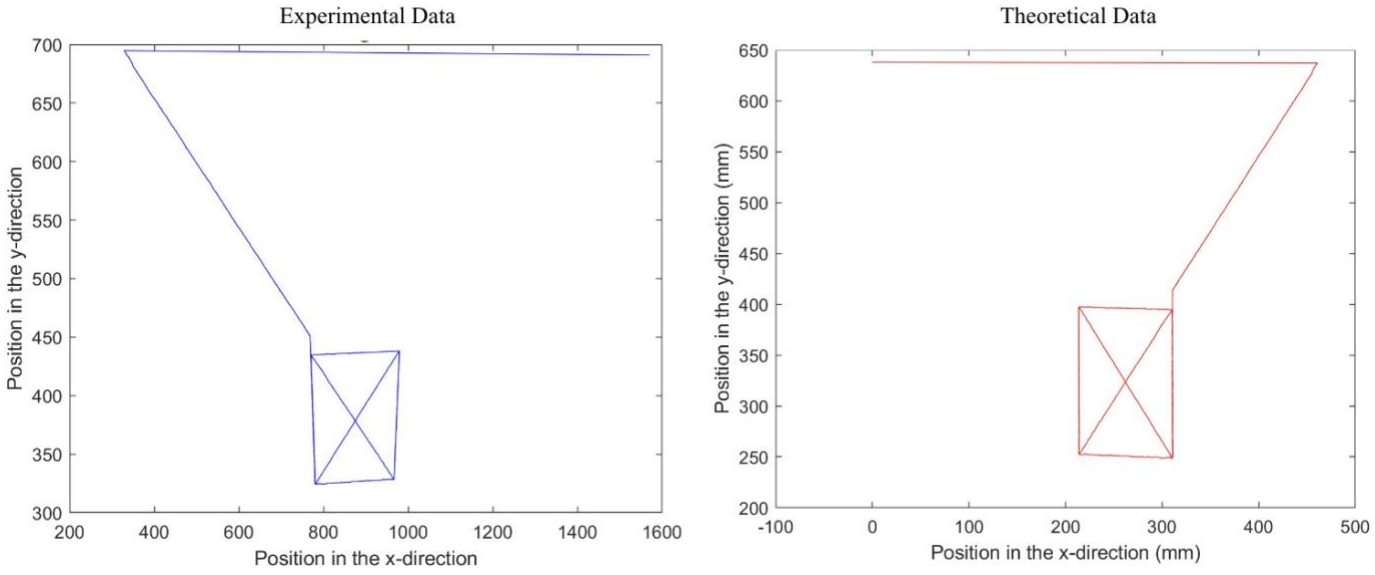


Figure 5.1: Experimental vs. theoretical data points obtained from tracking marker on XY platform

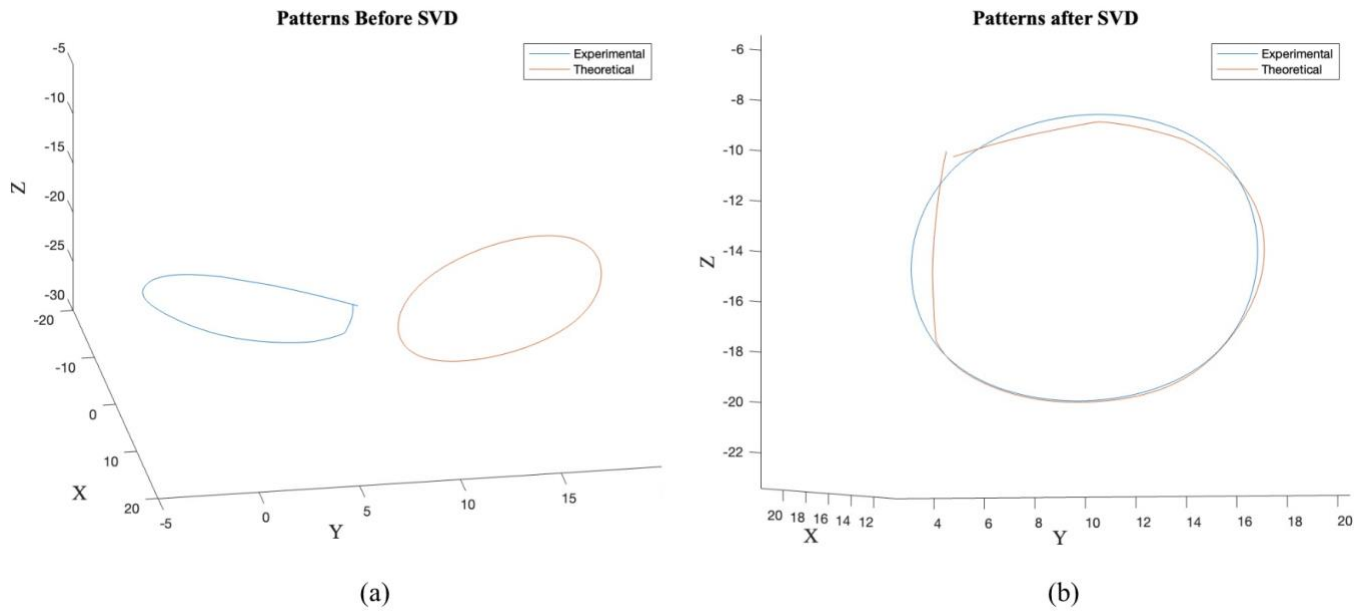
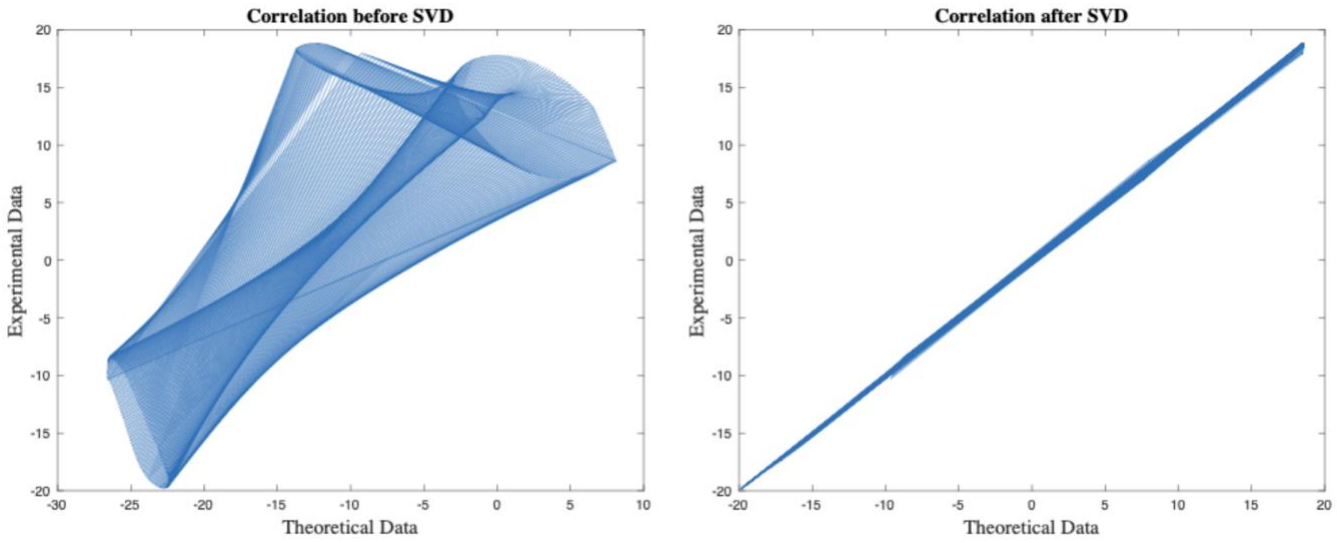


Figure 5.2: Patterns (a) before and (b) after SVD while tracking marker on robot



(a) (b)

Figure 5.3: Correlation between experimental and theoretical data (a) before and (b) after SVD

5.4 Discussion

From the results, it is clear that using optical tracking for detecting ArUco markers is promising. The detection percentages from Table 5.1 show that the cameras are frequently able to detect ArUco markers, as often as 100% of the time. In all color spaces, for the white color marker, the detection percentage is consistent. But, the same cannot be said for the pink, orange and yellow markers. This could be attributed to the fact that the ArUco markers were specifically designed for their white-and-black bits to be identified through computer vision. The black border in particular allows for fast detection of the markers [29]. ArUco markers are identified through a contour detection process, which helps detect the presence of an ArUco marker, and its marker ID [55]. This process might be affected when different colors take up parts of the marker (the white portions) that provide highest contrast to the black bits and enable efficient boundary detection. As a result, it would disrupt the in-built process of marker detection. Similar reasoning can be applied to answer why higher deviational errors are observed when colored markers are tracked.

It can thus be inferred that the standard white-and-black ArUco markers are the most suitable to be tracked by a stereo camera. This result is consistent across all three color spaces too, meaning that for future experiments, it is best to continue working with the white marker. For this reason, no colored markers were used for experiments conducted on the robot.

Tracking experiments conducted with the robot show even lower deviation errors. This may be because SVD was used in the processing of the data for these experiments, which is customized for every dataset that is being compared against the ground truth. The patterns obtained before and after SVD seen in figure 5.1 are similar because SVD minimizes squared errors between every pair of datasets. The subsequent figure, figure 5.2, shows the correlation between actual movement of the robot and the experimental data. In numbers, before SVD the correlation was 0.78, and after, it was 0.99. Although the patterns might not match exactly, such a high correlation indicates that the experimental data is closely following the trend of the ground truth. This means that the method of optical tracking using the tracking algorithms explained earlier is effective and any error can be attributed to insufficiencies of the camera (resolution or frame rate).

The error from 3D tracking increases as the distance of the camera from the robot increases. This is expected, since optical tracking is most accurate when the cameras are closest to the target. Despite this, in Table 5.3, when the camera is only 500 mm away from the robot, the tracking error obtained is higher than when the camera is 600 mm away while tracking the 30 mm marker. It was noticed that when the camera was placed very close to the object of interest, the camera's vision gets blurry, causing gaps in detection. This effect is amplified when the marker is bigger, since the 30 mm marker, at a 500 mm distance, had the lowest tracking accuracy when compared to that of the 20 mm and 15 mm markers. But, as the distance increases (600 mm and 700 mm), we can see that the 30 mm marker is uniformly tracked more accurately than the remaining two markers.

The difference in error across the three distances is not pronounced, meaning that even at larger distances, the marker can be expected to be detected. This can be said of different marker sizes too. In an ideal scenario, the marker would need to be as small as possible with as much tracking accuracy as possible. The bigger the size, the higher are the chances of the marker on the needle obstructing the surgeon's vision.

When we look at the different color spaces that were tested with, the errors are more or less the same during 3D tracking, with no one color space performing significantly better than another. But, from the experiments conducted on the XY platform, the RGB color space was noticeably better in terms of detection percentage and tracking accuracy. This is not very surprising since RGB is the most common color space used in digital image processing and is the one that is understood best by computers [46].

An important point to consider while conducting tracking experiments is that the baseline distance between the two cameras strongly influences tracking accuracy. The baseline distance and the focal length and the distance of the camera from the device(s) are related to one another. Thus, a balance needs to be struck between the three. All the above experiments worked most accurately when the camera was placed at a distance of 600 mm, and when the baseline distance was 122 - 125 mm. Ultimately, the navigation system is to be used by a surgeon, and so, it must also be mentioned that the distance between human eyes, or the interpupillary distance (IPD), is only 5.56 - 6.8 cm [56]. In this work, the aim was to prove the capabilities of optical tracking using web cameras, but any stereo camera built later would have to closely follow the IPD to obtain more relevant results.

Overall, the 2D tracking error (RMSE) was 5.48 mm and the 3D tracking error was 2.27 mm in the RGB color space. The error can further be reduced by using higher resolution cameras,

or by experimenting with different tracking algorithms that might work better for the purpose of surgical navigation.

Chapter 5, in part, is a modified version of the material from Tsui, Darin., Melentyev, Capalina., Rajan, Ananya., Kumar, Rohan., & Talke, Frank E., An Optical Tracking Approach to Computer-Assisted Surgical Navigation via Stereoscopic Vision, 2023. The dissertation author was one of the co-authors of the paper.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

This project provides the foundation for building a low-cost surgical navigation system that relies on optical tracking using affordable cameras. The stereoscopic camera was calibrated and programmed to track ArUco markers effectively in OpenCV. The markers were meant to be placed on the skin of the patient, to track their movements and overlay an MRI image during surgery. The second use of the marker was attaching it to a surgical instrument to track it and provide real-time guidance to the surgeon.

To verify the accuracy of tracking, an XY platform and a robot were employed for each of the above mentioned cases, respectively. The XY platform would simulate a patient undergoing surgery, and the robot, the arm of a surgeon performing surgery and maneuvering the surgical instrument.

All data from the experiments was transformed into the world frame for performing relevant comparison. Additionally, filtering and post-processing were also done to improve tracking accuracy. The deviational error calculated from tests with the XY platform was 5.48 mm, and 2.27 mm with tests with the robotic arm in the RGB color space.

Other tests proved that tracking accuracy reduced over distance, although the increase in error was small (0.2 - 0.4 mm). A final observation was that the smaller the marker becomes, the lower is the tracking accuracy. Thus it is important to maintain the correct ratio of size to distance, with the goal of making the markers as small as possible, so as to not disturb the surgeon's workflow.

6.2 Future Work

Improving Tracking Accuracy

There are various algorithms that can be used to detect an ArUco marker - the one performed in this project, local matching, belongs to a set of matching algorithms that find similarities between two frames by calculating the disparity between them. There are other matching algorithms that can be implemented that produce 'disparity maps'. A disparity map subtracts common features of one frame from another and produces a grayscale image that is a result of the difference. Features can be detected by finding pixel similarities, or pixel block similarities by computing a cost function [57, 58]. Cost function, or cost computation can be as simple as minimizing the sum of absolute difference between two pixel/pixel blocks. To improve accuracy, cost aggregation can be performed to reduce differences along all directions in the image. When implemented in OpenCV, there are several parameters that can be modified to generate a smooth disparity map with distinct differences between light and dark areas, and objects that are closer and farther from the camera. These parameters include the size of the matching window, the minimum disparity value, and so on, which facilitate fine-tuning the output [59, 60].

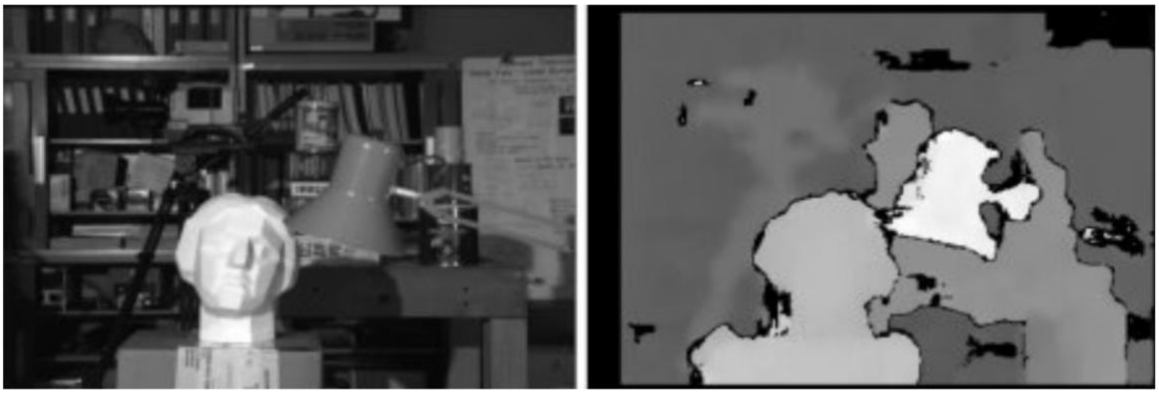


Figure 6.1: An example of a disparity map created from stereo images [61]

Another method that can be considered is Perspective-n-Point computation. This uses the function solvePnP in OpenCV; it provides rotation and translation vectors that can be used to localize the ArUco marker, obviating the need for triangulation [62, 63]. It does so by matching known 3D corners of the ArUco marker with its 2D projection in the image [64], forming a 'correspondence' between the two. The rotation and translation vectors that are the output can be used to convert the marker's pose from its own frame to camera frame [65].

Instrument Tracking

While instrument tracking from the experiments conducted in this project were favorable, there are some limitations associated with the kind of marker that was used. An ArUco marker is generally 2D in shape, and this is ideal in a situation where its movement is limited to 3DOF. The movement of a surgical instrument is naturally not fixed to 3DOF and moves in 6DOF. The robotic arm that was used for conducting experiments in 5DOF indeed moved the marker through various rotations and translations, but in all cases, the marker was always easily visible to the camera. Thus, the effect of more dramatic rotational and translational changes of the marker's position with respect to the camera need to be considered. For example, flipping the instrument away from the camera will result in the marker no longer being visible to the camera. This describes the problem of occlusion faced during optical tracking.

This issue can be resolved by using a 3D ArUco marker attached to the surgical instrument. Similar work has been presented in the past using 3D ArUco markers (referred to as dice here onwards) for various purposes, such as use with multiple cameras arranged around the object to be tracked 69], tracking forceps during surgery [63] or to track the movement of a stylus [67]. Each of these uses a different algorithm for tracking the dice, which is not quite suitable for stereovision, though its use can be explored in other areas. Also, due to the bulky design of the

dice used for research, they are not suitable for the purpose of instrument tracking and counterintuitively occlude the surgeon's vision.

Thus, there is a need to design a die shape that is compact, lightweight, and allows for constant line-of-sight. The last consideration also means that the angles of each face of the die from its center need to carefully be selected. There is more advantage to a die whose faces are angled outwards, for example, 110° from the die center than a smaller angle such as 90°, as in a cube, since a narrow angle creates situations where only one marker is clearly visible at a time.
Many kinds of 3D ArUco markers were designed and printed with different materials (PLA, white resin and black resin) using the Formlabs 3D printer.

Figure 6.2: From left to right - 5-sided [63], 12-sided and 6-sided ArUco dice

The 12-sided marker was observed to be too big. But, if it was scaled down, the size of faces became too small to fix ArUco markers of a detectable size. The problem with the 6-sided ArUco cube, as explained above, was that the angles were all 90°, which is not ideal.
The new design of the ArUco die was 9-sided. It was angled 105° outwards from its center, offering seamless line-of-sight. This angle also ensured that the size of the marker was contained within a reasonably small area and would not disturb the surgeon.

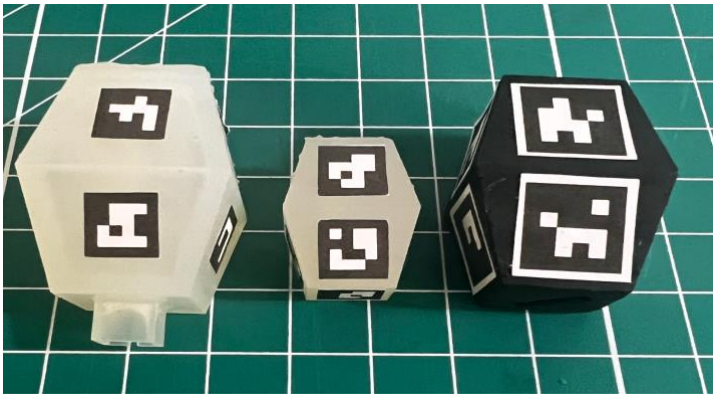


Figure 6.3: A novel 9-sided ArUco die in different sizes

Different materials were used for printing the die - PLA, white resin and black resin. The lightest material, found only by holding the dice, was PLA. Both the resins were heavy and caused the surgical needle to bend.

Figure 6.4: Marker attached to surgical instruments

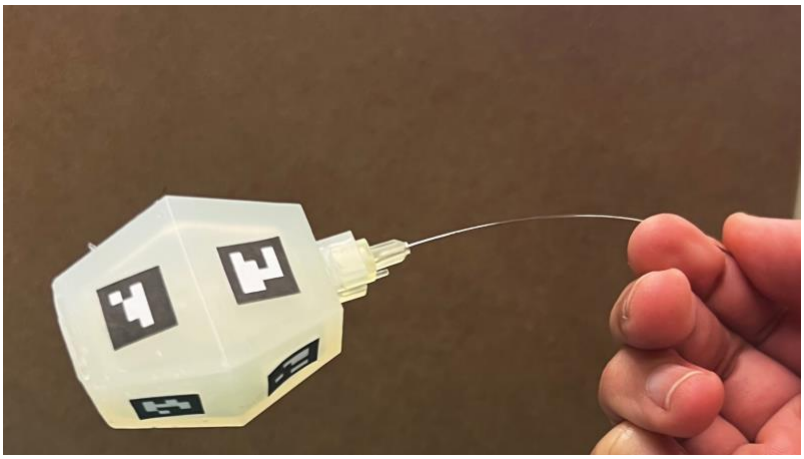


Figure 6.5: Needle bending under the weight of the white resin die

Theoretically, the 9-sided die seems to be ideal, but only future tests can tell if its shape is superior to other shapes in terms of tracking accuracy. Similar verification can be conducted with the robot, as was done for the 2D markers. This leads to a new challenge, which is developing a new algorithm that is capable of detecting multiple ArUco markers at once, and also finding the center of the die relative to the position of these markers. Research has been done in this domain [63, 68, 69], with each work outlining a different method. But the key takeaway is that the relative distance depends on the shape of the die and can be found if the die's corners in world coordinates are provided to the program. The possibility of each marker rotating and translated will also have to be factored into the code. The ultimate goal of instrument tracking is to find the position of the instrument's tip. Again, with relative positions, it is possible to extract the pose of the instrument tip.

MRI Image Overlay

By detecting and tracking ArUco markers on the patient's body, virtual objects can be overlaid. When markers are detected on the patient, a virtual MRI image is to be aligned above the patient and visible through the AR headset.

When overlaying MRI images, it is important to note that they are usually captured when the patient is in a supine position, but during spine procedures, the patient is in the prostate position. This would lead to a mismatch between the overlaid MRI image and the actual position of the patient's spine. One way to resolve this discrepancy is by scanning the patient in the prostate position. If this is not possible, the scan must be rotated such that it matches the patient's orientation. It is also necessary to find the ideal positions to place ArUco markers on the patient's back for MRI overlay. Some positions might provide a more accurate alignment than others depending on the curvature of the back.

There are other concerns such as the scan getting warped, or not being in the same plane as the patient. To prevent this, it needs to be ensured that the transformation from coordinates of the 3D MRI image to image coordinates is accurate and does not lead to any information loss.

Integration with Augmented Reality (AR)

The next stage in building a surgical navigation system is integrating all programs into an augmented reality headset. The same programs written in Python in OpenCV may not be directly transferable to the AR headset. There are other engines that can use the primary codes as a base and convert them to a language that is used to communicate with the AR headset. Unity is one such engine, commonly used for game development in the field of AR and VR (virtual reality) [70]. Unity supports the OpenCV package, so the same algorithm used in this project can be implemented in Unity as well.

With augmented reality in place, the final stage of overlaying virtual objects (MRI and surgical instruments) will be completed. There are many advantages an AR headset can provide to a clinician, such as navigating complex anatomies, and visualizing real-time instrument navigation. Another benefit is the undisturbed focus the surgeon can give the patient, without

needing to look up at a screen during surgery, unlike many of the existing technologies that use an external display.

Clinician Testing

When the AR headset is capable of performing the above mentioned functions, the device must be tested by surgeons. Their input would be valuable in further optimizing the product to fit their needs. Similarly, patient feedback is equally important in understanding how the device can be made more patient-friendly. This would finally complete the design process of the surgical navigation system.

REFERENCES

[1] Langø, T., Vijayan, S., Rethy, A., Våpenstad, C., Solberg, O. V., Mårvik, R., Johnsen, G., & Hernes, T. N. (2011). Navigated laparoscopic ultrasound in abdominal soft tissue surgery: technological overview and perspectives. *International Journal of Computer Assisted Radiology and Surgery*, *7*(4), 585–599. https://doi.org/10.1007/s11548-011-0656-3

[2] Musahl, V., Plakseychuk, A., & Fu, F. H. (2002). Current Opinion on Computer-Aided Surgical Navigation and Robotics. *Sports Medicine*, *32*(13), 809–818. https://doi.org/10.2165/00007256-200232130-00001

[3] Kraus, M. W., Gert Krischak, Keppler, P., Gebhard, F., & Schuetz, U. H. (2010). Can Computer-assisted Surgery Reduce the Effective Dose for Spinal Fusion and Sacroiliac Screw Insertion? *Clinical Orthopaedics and Related Research*, *468*(9), 2419–2429. https://doi.org/10.1007/s11999-010-1393-6

[4] Mezger, U., Jendrewski, C., & Bartels, M. (2013). Navigation in surgery. *Langenbeck's Archives of Surgery*, *398*(4), 501–514. https://doi.org/10.1007/s00423-013-1059-4

[5] Yu, X., Xu, L., & Bi, L. (2008). Spinal navigation with intra-operative 3D-imaging modality in lumbar pedicle screw fixation. *Zhonghua Yi Xue Za Zhi*, *88*(27), 1905–1908. https://pubmed.ncbi.nlm.nih.gov/19040004/

[6] Paulo Waelkens, Oosterom, van, van, Nassir Navab, & Leeuwen, van. (2016). *Surgical Navigation: An Overview of the State-of-the-Art Clinical Applications*. 57–73. https://doi.org/10.1007/978-3-319-26051-8_4

[7] Hoy, D., Brooks, P., Blyth, F., & Buchbinder, R. (2010). The Epidemiology of low back pain. *Best Practice & Research Clinical Rheumatology*, *24*(6), 769–781. https://doi.org/10.1016/j.berh.2010.10.002

[8] Carassiti, M., Pascarella, G., Strumia, A., Russo, F., Papalia, G. F., Cataldo, R., Gargano, F., Costa, F., Pierri, M., De Tommasi, F., Massaroni, C., Schena, E., & Agrò, F. E. (2021). Epidural Steroid Injections for Low Back Pain: A Narrative Review. *International Journal of Environmental Research and Public Health*, *19*(1), 231. https://doi.org/10.3390/ijerph19010231

[9] Peng, B.-G. (2013). Pathophysiology, diagnosis, and treatment of discogenic low back pain. *World Journal of Orthopedics*, *4*(2), 42. https://doi.org/10.5312/wjo.v4.i2.42

[10] Patel, K., Chopra, P., & Upadhyayula, S. (2021). *Epidural Steroid Injections*. PubMed; StatPearls Publishing. https://www.ncbi.nlm.nih.gov/books/NBK470189/

[11] Rawicki, N. L., Dowdell, J., & Sandhu, H. S. (2021). *Current state of navigation in spine surgery*. *9*(1), 85–85. https://doi.org/10.21037/atm-20-1335

[12] Rossi, V. J., Wells-Quinn, T. A., & Malham, G. M. (2021). Negotiating for new technologies: guidelines for the procurement of assistive technologies in spinal surgery: a narrative review. *Journal of Spine Surgery*, *0*(0). https://doi.org/10.21037/jss-21-107

[13] *The Pulse platform - NuVasive*. https://www.nuvasive.com/surgical-solutions/pulse/

[14] *Spine Navigation*. (n.d.). Stryker. https://www.stryker.com/us/en/portfolios/orthopaedics/spine--ortho-/spine-navigation.html

[15] Kalfas, I. H. (2021). Machine Vision Navigation in Spine Surgery. *Frontiers in Surgery*, *8*. https://doi.org/10.3389/fsurg.2021.640554

[16] *FLASH Navigation - 7D Surgical*. (n.d.). https://7dsurgical.com/flash-navigation/

[17] Rush, A. J., Shepard, N., Nolte, M., Siemionow, K., & Phillips, F. (2022). Augmented Reality in Spine Surgery: Current State of the Art. *International Journal of Spine Surgery*, *16*(S2), S22–S27. https://doi.org/10.14444/8273

[18] Dibble, C. F., & Molina, C. A. (2020). Device profile of the XVision-spine (XVS) augmented reality surgical navigation system: overview of its safety and efficacy. *Expert Review of Medical Devices*, *18*(1), 1–8. https://doi.org/10.1080/17434440.2021.1865795

[19] *Augmented Reality Spine Surgery - X-Ray Vision | AugMedics*. (2024, June 11). Augmedics. https://augmedics.com/

[20] Enrique, George & Urbano, Luis & Acero, A. & Huerta, Mónica & Castro, Miguel. (2014). Surgical Navigation Systems: A Technological Overview. http://dx.doi.org/10.13140/RG.2.1.2449.1043

[21] Wilson, J. P., Fontenot, L., Stewart, C., Kumbhare, D., Guthikonda, B., & Hoang, S. (2024). Image-Guided Navigation in Spine Surgery: From Historical Developments to Future Perspectives. *Journal of Clinical Medicine*, *13*(7), 2036. https://doi.org/10.3390/jcm13072036

[22] Sharma, S., Aditya Telikicherla, Ding, G., Fatemeh Aghlmand, Arian Hashemi Talkhooncheh, Shapiro, M. G., & Emami, A. (2021). Wireless 3D Surgical Navigation and Tracking System With 100µm Accuracy Using Magnetic-Field Gradient-Based Localization. *IEEE Transactions on Medical Imaging*, *40*(8), 2066–2079. https://doi.org/10.1109/tmi.2021.3071120

[23] Stoll, J., Ren, H., & Dupont, P. E. (2012). Passive Markers for Tracking Surgical Instruments in Real-Time 3-D Ultrasound Imaging. *IEEE Transactions on Medical Imaging*, *31*(3), 563–575. https://doi.org/10.1109/tmi.2011.2173586

[24] Xu, L., Zhang, H., Wang, J., Li, A., Song, S., Ren, H., Qi, L., Gu, J. J., & Meng, M. Q.-H. . (2023). Information loss challenges in surgical navigation systems: From information fusion to AI-based approaches. *Information Fusion*, *92*, 13–36. https://doi.org/10.1016/j.inffus.2022.11.015

[25] Sorriento, A., Porfido, M. B., Mazzoleni, S., Calvosa, G., Tenucci, M., Ciuti, G., & Dario, P. (2020). Optical and Electromagnetic Tracking Systems for Biomedical Applications: A Critical Review on Potentialities and Limitations. *IEEE Reviews in Biomedical Engineering*, *13*, 212–232. https://doi.org/10.1109/rbme.2019.2939091

[26] Rampersaud, Y. R., Foley, K. T., Shen, A. C., Williams, S., & Solomito, M. (2000). Radiation Exposure to the Spine Surgeon During Fluoroscopically Assisted Pedicle Screw Insertion. *Spine*, *25*(20), 2637. https://journals.lww.com/spinejournal/fulltext/2000/10150/radiation_exposure_to_the_spine_surgeon_during.16.aspx

[27] Stoll, J., & Dupont, P. (2005). Passive markers for ultrasound tracking of surgical instruments. *Medical Image Computing and Computer-Assisted Intervention: MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention*, *8*(Pt 2), 41–48. https://doi.org/10.1007/11566489_6

[28] Hassfeld, S., & Mühling, J. (2001). Computer assisted oral and maxillofacial surgery – a review and an assessment of technology. *International Journal of Oral and Maxillofacial Surgery*, *30*(1), 2–13. https://doi.org/10.1054/ijom.2000.0024

[29] *OpenCV: Detection of ARUCO markers*. (n.d.). https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

[30] Lefebvre, S., Ambellouis, S., & Cabestaing, F. (2011). A 1D approach to correlation-based stereo matching. *Image and Vision Computing*, *29*(9), 580–593. https://doi.org/10.1016/j.imavis.2011.05.003

[31] Tsai, R. (1987). A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, *3*(4), 323–344. https://doi.org/10.1109/jra.1987.1087109

[32] *OpenCV: Camera calibration and 3D reconstruction*. (n.d.). https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html

[33] Zhengyou Zhang. (1999). Flexible camera calibration by viewing a plane from unknown orientations. *Proceedings of the Seventh IEEE International Conference on Computer Vision*. https://doi.org/10.1109/iccv.1999.791289

[34] Lei, T., Rong, Y., Wang, H., Huang, Y., & Li, M. (2020). A review of vision-aided robotic welding. *Computers in Industry*, *123*, 103326–103326. https://doi.org/10.1016/j.compind.2020.103326

[35] *First Principles of Computer Vision.* https://fpcv.cs.columbia.edu/

[36] *What is camera calibration? - MATLAB & Simulink*. (n.d.). https://www.mathworks.com/help/vision/ug/camera-calibration.html

[37] *OpenCV: Camera Calibration*. (n.d.). https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

[38] *cv.stereoCalibrate - mexopencv*. (n.d.). https://amroamroamro.github.io/mexopencv/matlab/cv.stereoCalibrate.html

[39] *cv.stereoRectify - mexopencv*. (n.d.). https://amroamroamro.github.io/mexopencv/matlab/cv.stereoRectify.html

[40] Nguyen, B. (2022). *Position Localization for Interventional Pain Management Therapies* [Master's Thesis, University of California, San Diego].

[41] *Summary health statistics for U.S. adults: National Health Interview Survey, 2009*. (2010, December 1). PubMed. https://pubmed.ncbi.nlm.nih.gov/21905346/

[42] Tsui, D., Capalina Melentyev, Rajan, A., Kumar, R., & Talke, F. E. (2023). *An Optical Tracking Approach to Computer-Assisted Surgical Navigation via Stereoscopic Vision*. https://doi.org/10.1115/isps2023-111020

[43] Angelo Damante. (n.d.). *GitHub - AngeloDamante/arm-manipulator-5dof: Implementation of 5 DOF Arm Manipulator with Simulink.* GitHub. https://github.com/AngeloDamante/arm-manipulator-5dof

[44] Severinghaus, L. (2024, April 19). *kk6axq/BraccioV2*. GitHub https://github.com/kk6axq/BraccioV2?tab=readme-ov-file

[45] Setchell, J. S. (2012). Colour description and communication. *Elsevier EBooks*, 219–253. https://doi.org/10.1533/9780857095534.2.219

[46] Gowda, S. N., & Yuan, C. (2019). ColorNet: Investigating the Importance of Color Spaces for Image Classification. *Computer Vision – ACCV 2018*, 581–596. https://doi.org/10.1007/978-3-030-20870-7_36

[47] Ledley, R. S., Buas, M., & Golab, T. J. (2002). *Fundamentals of true-color image processing*. https://doi.org/10.1109/icpr.1990.118218

[48] Joy, D. T., Kaur, G., Chugh, A., & Bajaj, S. B. (2021). COMPUTER VISION FOR COLOR DETECTION. *International Journal of Innovative Research in Computer Science & Technology, 9*(3). https://doi.org/10.21276/ijircst.2021.9.3.9

[49] Joblove, G. H., & Greenberg, D. (1978). Color spaces for computer graphics. *ACM SIGGRAPH Computer Graphics*, *12*(3), 20–25. https://doi.org/10.1145/965139.807362

[50] Ibraheem, N., Hasan, M., Khan, R., & Mishra, P. (2012). *Understanding Color Models: A Review*. *2*(3). https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=75e9c6dcaeaa31fd60e61a69aaba7e706b97e4ad

[51] Plataniotis, K. N., & Venetsanopoulos, A. N. (2000). Color Spaces. *Digital Signal Processing*, 1–49. https://doi.org/10.1007/978-3-662-04186-4_1

[52] *HSL and HSV*. (2024, July 4). Wikipedia. https://en.wikipedia.org/wiki/HSL_and_HSV

[53] *OpenCV: Changing Colorspaces*. (n.d.).Docs.opencv.org. https://docs.opencv.org/4x/df/d9d/tutorial_py_colorspaces.html

[54] Sorkine-Hornung, O., & Rabinovich, M. (2017). Least-squares rigid motion using svd. Computing, 1(1), 1-5.

[55] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, *47*(6), 2280–2292. https://doi.org/10.1016/j.patcog.2014.01.005

[56] Hayat, N., Alkhairy, S., Cheema, A., Ehsan, M., & Khan, M. A. (2019). Normal interpupillary, inner canthal distance and outer canthal distance in a normal population of Pakistan. *Pakistan Journal of Medical Sciences*, *35*(1). https://doi.org/10.12669/pjms.35.1.288

[57] Konolige, K. (1998). *Small Vision Systems: Hardware and Implementation*. 203–212. https://doi.org/10.1007/978-1-4471-1580-9_19

[58] Hirschmuller, H. (2008). Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(2), 328–341. https://doi.org/10.1109/tpami.2007.1166

[59] *OpenCV: cv::stereo::StereoBinaryBM Class Reference*. (n.d.). Docs.opencv.org. Retrieved July 10, 2024, from https://docs.opencv.org/3.4/d7/d8e/classcv_1_1stereo_1_1StereoBinaryBM.html

[60] *OpenCV: cv::StereoSGBM Class Reference*. (n.d.). Docs.opencv.org. https://docs.opencv.org/4.x/d2/d85/classcv_1_1StereoSGBM.html

[61] *OpenCV: Depth Map from Stereo Images*. (n.d.). Docs.opencv.org. https://docs.opencv.org/4.x/dd/d53/tutorial_py_depthmap.html

[62] *OpenCV: Perspective-n-Point (PnP) pose computation*. (n.d.). Docs.opencv.org. https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html

[63] Stenmark, M., Edin Omerbašić, Magnusson, M., Andersson, V., Abrahamsson, M., & Tran, P. (2022). Vision-Based Tracking of Surgical Motion During Live Open-Heart Surgery. *Journal of Surgical Research*, *271*, 106–116. https://doi.org/10.1016/j.jss.2021.10.025

[64] Marchand, E., Uchiyama, H., & Spindler, F. (2016). Pose Estimation for Augmented Reality: A Hands-On Survey. *IEEE Transactions on Visualization and Computer Graphics*, *22*(12), 2633–2651. https://doi.org/10.1109/tvcg.2015.2513408

[65] López-Cerón, A., & José María Plaza. (2022). Accuracy analysis of marker-based 3D visual localization. *Actas de Las XXXVII Jornadas de Automática 7, 8 Y 9 de Septiembre de 2016, Madrid*. https://doi.org/10.17979/spudc.9788497498081.1124

[66] Sarmadi, H., Munoz-Salinas, R., Berbis, M. A., & Medina-Carnicer, R. (2019). Simultaneous Multi-View Camera Pose Estimation and Object Tracking With Squared Planar Markers. *IEEE Access*, *7*, 22927–22940. https://doi.org/10.1109/access.2019.2896648

[67] Wu, P.-C., Wang, R., Kin, K., Twigg, C., Han, S., Yang, M.-H., & Chien, S.-Y. (2017). *DodecaPen: Accurate 6DoF Tracking of a Passive Stylus*. https://doi.org/10.1145/3126594.3126664

[68] Benjumea, E., Sierra, J. S., Meza, J., & Marrugo, A. G. (2021). Multi-target Attachment for Surgical Instrument Tracking. *Lecture Notes in Computer Science*, 345–354. https://doi.org/10.1007/978-3-030-77004-4_33

[69] Masanao Koeda, Yano, D., Shintaku, N., Onishi, K., & Hiroshi Noborio. (2018). Development of Wireless Surgical Knife Attachment with Proximity Indicators Using ArUco Marker. *Lecture Notes in Computer Science*, 14–26. https://doi.org/10.1007/978-3-319-91244-8_2

[70] Unity Technologies. (2019). *Unity*. Unity. https://unity.com/