stryker

CentraleSupélec

Internship report

Clement Wang

Computer vision R&D intern

Feb 2023 - Aug 2023

University supervisor:
Bich-Liên Doan
bich-lien.doan@centralesupelec.fr

Company supervisor:
Christian Koch
christian.koch@stryker.com

CentraleSupélec | université PARIS-SACLAY

Abstract

This internship report presents my work conducted at Stryker in Freiburg, Germany, focusing on the precision tracking of surgical instruments in the context of surgical procedures. The report encompasses two projects: calibration of surgical instruments and improving the subpixel precision of instrument tracking.

The first project revolves around the calibration of surgical instruments to ensure accurate tracking during surgery. Currently, the calibration process involves using a dedicated calibration device and manually calibrating each instrument individually, which can be time-consuming. In this project, the objective was to explore an alternative method using deep learning and a single RGB camera. The 3D geometry of the instruments is obtained by analyzing a short video sequence of a few seconds. The achieved calibration accuracy showed a 3 mm difference compared to the current method, which claims an accuracy of +/- 2 mm.

The second project focuses on enhancing the subpixel precision of instrument tracking using near-infrared (NIR) images and deep learning techniques. Instruments are tracked based on detecting trackers in the NIR images. The goal was to improve the precision of tracker detection. Through the use of simulated data, the deep learning approach achieved an error of 0.005 pixels, whereas the existing method exhibited an error of 0.05 pixels.

Overall, this internship report showcases the development and evaluation of novel methods for instrument calibration and tracking using deep learning techniques.

Acknowledgement

I would like to express my heartfelt gratitude to the following individuals for their invaluable support and guidance throughout my internship at Stryker in Freiburg, Germany.

First and foremost, I extend my sincere appreciation to Christian Koch, my company supervisor. His introduction to the subject, engaging discussions on machine learning, and continuous advice have been instrumental in refining our ideas and making significant progress in the projects. As an ignorant French guy in Germany, his patience and assistance were invaluable.

I am deeply thankful to Max Sirkin, my manager, for his unwavering support and assistance whenever I needed help. Whether it was showing me around or navigating through paperwork troubles, his availability and willingness to help were truly commendable.

I am indebted to Emeric Umbdenstock and Paul Hoekstra for generously sharing their expertise on the camera and providing valuable guidance whenever I sought more information on its workings. I also extend my gratitude to Yannik Guth for the explanations on the Spine application and the manual calibration process.

I would like to acknowledge Rami Mayer and Alexander Kuck, the other members of my team, for the enjoyable time we spent together and the fruitful meetings we had. Even though our time in the office was limited, it was a pleasure working with them.

A special mention goes out to all the interns who were there with me during my time at Stryker: Sophie, Senil, Oliver, Rebeca, Mauricio, Lena, Jonathan, Florian, Konstantin, Jonas, Anika, Bjorn, Martina, Adrian, Isabelle, Karolina, Leyre, and Ilary. Your warm acceptance and camaraderie made my experience in Germany truly memorable. From lunchtime chats at Mira to walks around the building after lunch, bouldering at Blockhaus, playing volleyball and spikeball at Seepark or Dietenbach Park, and chilling at the lake, the time we spent together was filled with joy and laughter. I never imagined that I would form such strong bonds, party together, and even travel with all of you.

Lastly, I am grateful to the entire team at Stryker for fostering a welcoming and supportive work environment that allowed me to grow both professionally and personally.

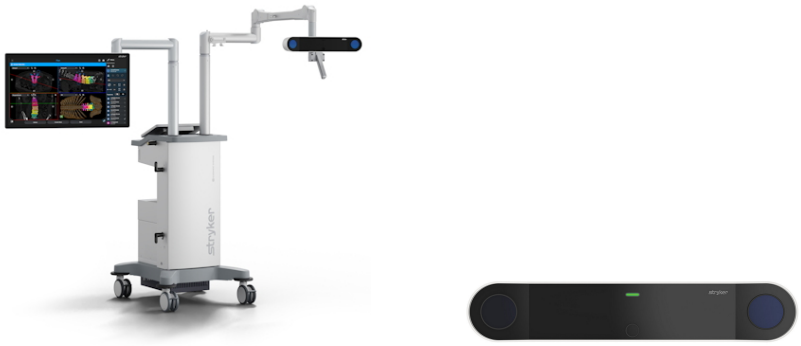
Thank you all for making my internship experience unforgettable.

Contents

1 Introduction

The accurate tracking of surgical instruments plays a vital role in ensuring the success and safety of surgical procedures. With advancements in medical technology, tracking systems have become increasingly sophisticated, enabling precise monitoring and control of surgical instruments within the operation room. This internship report focuses on the development and improvement of tracking methodologies for surgical instruments, specifically in the context of the High-Speed Drill (HSD) instrument used surgeries.

One of the key components of the tracking system utilized in this study is the FP8000 camera system. This camera system comprises two near-infrared (NIR) cameras on the sides and a high-resolution RGB camera.



(a) Navigation platform (b) FP8000 camera

Figure 2: Navigation system

The NIR cameras enable the tracking of three-dimensional (3D) coordinates of trackers. Only passive trackers are used in this study. They consist of fiducial points designed to reflect NIR light. The FP8000 camera captures the reflected NIR light through its dedicated NIR sensors. With the presence of two NIR cameras in the system, classical triangulation techniques are employed to determine the 3D positions of the passive trackers. These fiducial points serve a critical role in enabling the precise localization and motion tracking of surgical instruments throughout various surgical procedures.

This internship was focused on the High-Speed Drill (HSD). The HSD consists of three main components: the tracker, the attachment, and the bur. The tracker, as previously mentioned, allows for real-time tracking of the HSD's position and orientation. The attachment, which can be either straight or twisted, provides the necessary stability and maneuverability for surgical operations. Finally, the bur is the head of the tool, specifically designed for cutting, drilling, or shaping bones with exceptional precision.

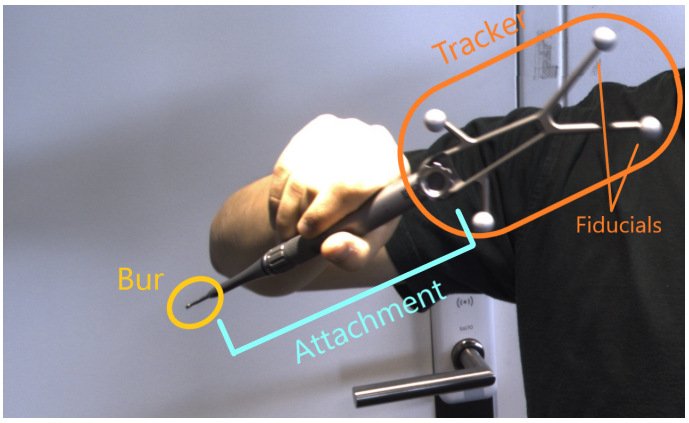


Figure 3: High speed drill

Throughout this internship, efforts were directed towards two main projects: calibration of surgical instruments and improving the subpixel precision of instrument tracking. The calibration project aimed to streamline the calibration process, replacing the manual calibration method with a video-based approach and deep learning techniques. The second project focused on enhancing the subpixel precision of instrument tracking by leveraging NIR images and employing deep learning algorithms.

By addressing these challenges and proposing innovative solutions, this internship aimed to contribute to the ongoing advancements in surgical instrument tracking, ultimately improving the accuracy and efficacy of surgical procedures. The subsequent sections of this report will provide an in-depth analysis of the projects undertaken, methodologies employed, results obtained, and the implications of the findings.

2 High Speed drill 3D calibration

2.1 Problem statement

The calibration of surgical instruments is a critical step in achieving accurate tracking during surgical procedures. However, the existing calibration process is labor-intensive and time-consuming, requiring manual calibration for each individual instrument. This inefficiency arises due to the considerable variability in parameters such as the attachment form and length, as well as the shape and size of the bur.

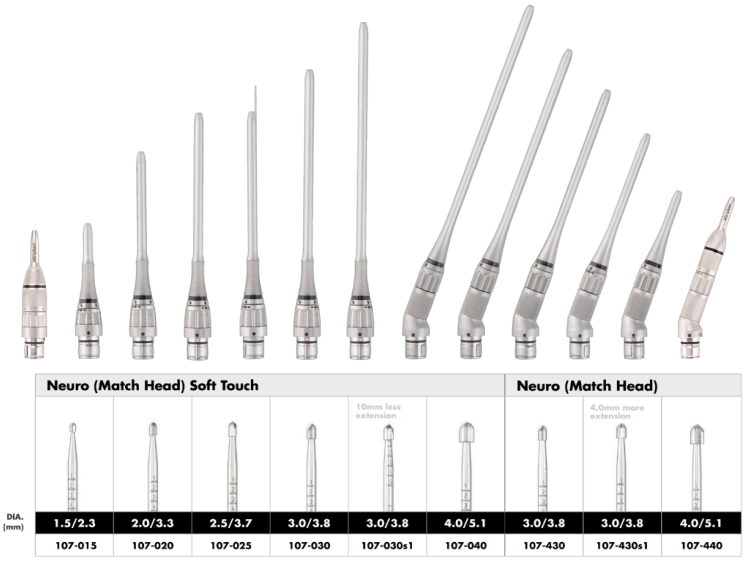


Figure 4: Top: Attachment samples. Bottom: Bur samples.

To address this issue, the primary objective is to develop a calibration solution that minimizes the input required from the user while maximizing precision. The proposed solution aims to leverage the available guidance system, specifically the FP8000 camera system, to streamline the calibration process.

To achieve this, the use of a deep learning algorithm is envisioned. The challenge lies in determining the necessary data inputs for the deep learning algorithm, as well as establishing a methodology for obtaining annotated data required for training.

2.2 Related works

Currently, the calibration of surgical instruments has been accomplished using a dedicated calibration device. This device typically includes a 3D tracker, and the calibration process involves manually touching specific points on the device with the tip of the instrument to establish the 3D transformation between the tool tracker and the tool tip. However, this method has several limitations. It requires multiple

inputs from the user, is repetitive, time-consuming, and demands a high level of precision.

In the context of the FP8000 camera system used in this project, calibration is achieved through triangulation from the two NIR cameras. These cameras provide a binocular view of the surgical scene, enabling the localization of fiducial points. The process involves detecting the corresponding centers of fiducials in both camera images and applying triangulation techniques. This methodology relies on the intrinsic and extrinsic parameters of the cameras, as well as lens distortion parameters. By leveraging principles of linear algebra, each camera image provides two constraining equations for a 3D point, represented as a 3D line in real space. With two images and the standard method of optimization, the precise 3D position of the instrument can be obtained by finding the point that optimizes the distance to each line. This approach, utilizing binocular view and 3D triangulation, is well-known for its exceptional precision.

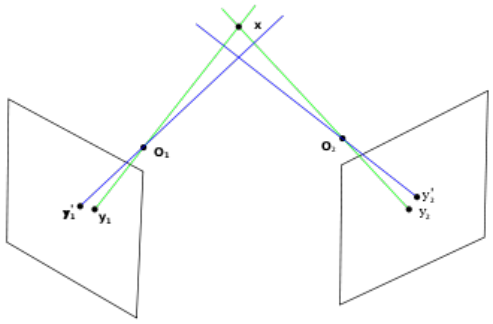


Figure 5: Triangulation

An alternative approach to obtaining 3D positions involves the use of polarized light and polarized sensors. While not directly applicable to the current project, this method offers an interesting avenue for future exploration and potential advancements in surgical instrument calibration.

Deep learning techniques have proven to be valuable in various computer vision tasks and can also be leveraged in the context of surgical instrument calibration. Two specific areas of deep learning that are relevant to this project are key point detection on 2D images and depth estimation from monocular views. Key points detection is a fundamental task in computer vision, involving the identification of specific points of interest on a 2D image. Deep learning models can be trained to accurately detect these points, even in complex and cluttered scenes. It is possible to automate the process of identifying fiducial centers or the instrument tip without the use of trackers.

Figure 6: Left: Key points detection. Right: Depth estimation.

Depth estimation is another area where deep learning has demonstrated significant advancements. Traditionally, depth maps have been obtained using stereo vision techniques that rely on multiple cameras or a pair of stereo images. However, deep learning approaches have enabled the estimation of depth maps from single monocular images. By training deep neural networks on large datasets with ground truth depth information, these models can infer depth information from a single image.

2.3 Method

The proposed method aims to achieve precise calibration of surgical instruments by leveraging deep learning techniques to accurately predict the 2D positions of the instrument tip on the captured images. With the guidance system's navigation software, the 3D positions of the instrument's tracker can be obtained at each frame. By combining the predicted tip positions and the 3D tracker positions, it becomes possible to triangulate and reconstruct the 3D geometry of the surgical tool using multiple frames.

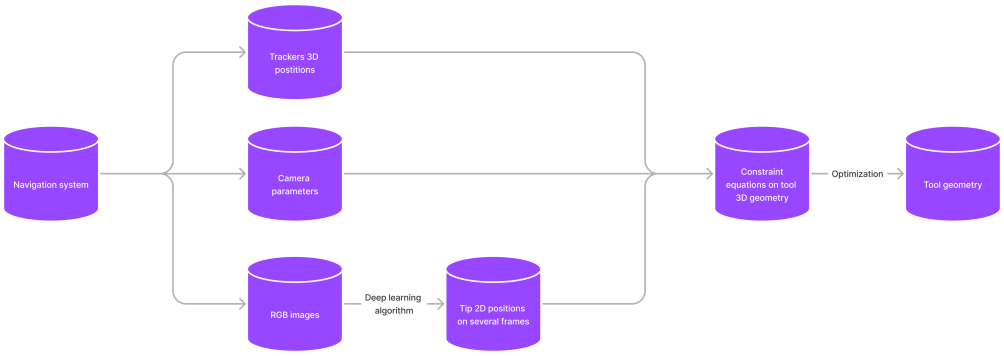


Figure 7: Pipeline overview

This solution is justified by considering the limitations of existing deep learning approaches. Depth estimation with deep learning models is typically not intended to achieve millimeter-level precision. Similarly, key point detection in the literature often focuses on semantic key points rather than precise localization. However, predicting the 2D positions of the instrument tip seems more feasible in terms of achieving the desired precision. Additionally, the method of triangulating the 3D geometry of the tool using multiple frames can be viewed as an optimization problem. Therefore, by incorporating a large number of images, it is possible to minimize and compensate for errors in the calibration process.

Another advantage of the proposed method is the availability of training data. Depth estimation using deep learning models requires ground truth depth data, which can be challenging to obtain. On the other hand, for 2D key point detection, the tools can be manually calibrated, and the corresponding 2D projections can be accurately determined. This allows for the creation of annotated data for training the deep learning models.

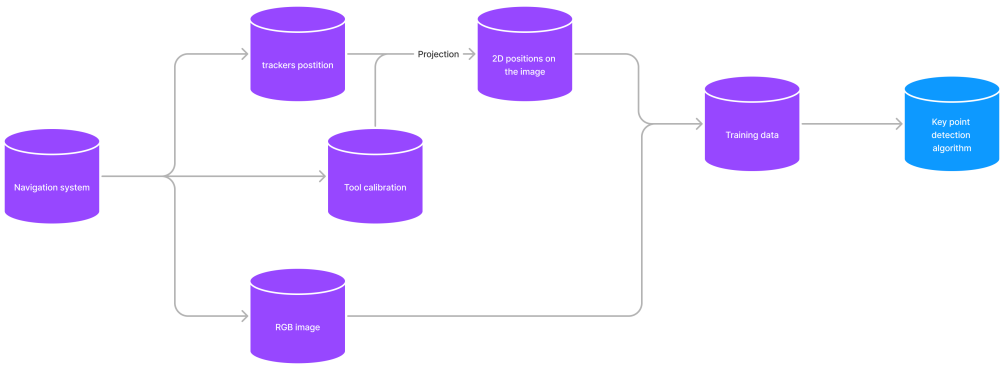


Figure 8: Training pipeline

By combining accurate 2D tip position predictions, 3D tracker positions, and the triangulation process, the proposed method offers a promising approach to achieving precise calibration of surgical instruments. The subsequent sections of this report will provide a detailed account of the specific methodologies employed, the experimental setup, and the results obtained during the implementation of this method.

2.4 Key point detection

The first part of the pipeline consists in accurately getting the 2D positions of the tip on an RGB image.

2.4.1 Literature review

Key point detection is an extensively studied field within computer vision, with a significant focus on semantic key point detection. Semantic key points refer to specific points of interest on an object that carry semantic meaning or significance. Examples of semantic key points include joint positions in human pose estimation, facial landmarks such as the eyes, nose, and mouth in face key points detection, and the positions of fingers in hand pose estimation.

When dealing with multi-object key point detection, two common approaches are employed: bottom-up and top-down. In the bottom-up approach, a heatmap is generated to predict all potential key points in the image, which are then associated with specific instances using a matching algorithm. On the other hand, the top-down approach bypasses this matching process by first using a detection algorithm to identify instances of the object of interest. Subsequently, a crop containing the instance is fed into the network for key point detection, simplifying the association step. In our case, the detection is not relevant. We will never have two overlapped instances during calibration time and the detection is already done with the navigation system.

To achieve accurate and robust 2D key point detection, state-of-the-art architectures often employ the prediction of heatmaps instead of directly predicting the coordinates of the key points. Heatmaps provide a spatial representation of the likelihood or presence of a key point at each location in the image. Convolutional neural networks (CNNs) are leveraged to effectively exploit the inherent spatial dependencies and patterns in the images, allowing for more precise and reliable detection of key points.

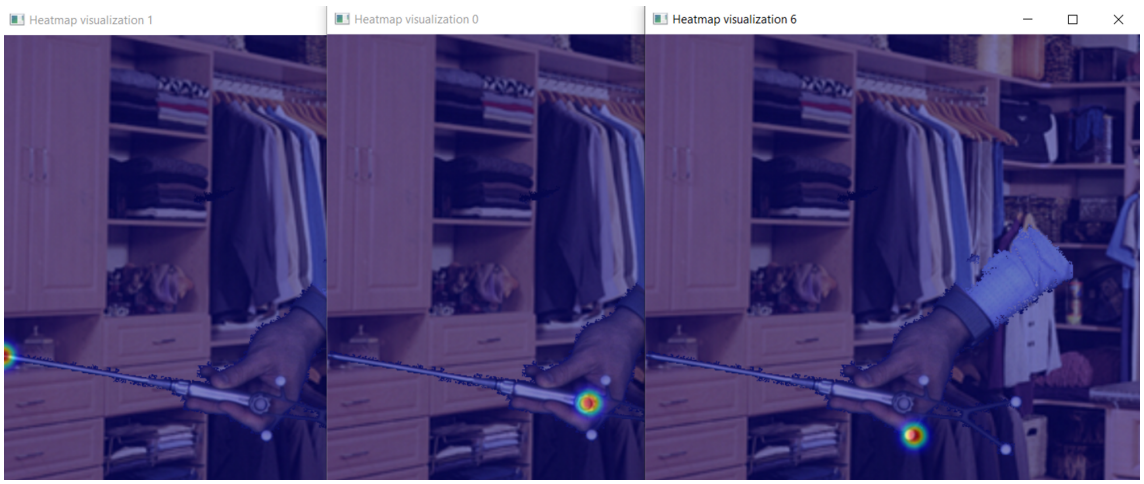


Figure 9: Ground truth heatmap visualization

An essential aspect of key point detection is the design of effective architectures that can leverage both global and local information. The combination of global con-

textual understanding and local fine-grained details plays a crucial role in achieving accurate and precise key point detection.

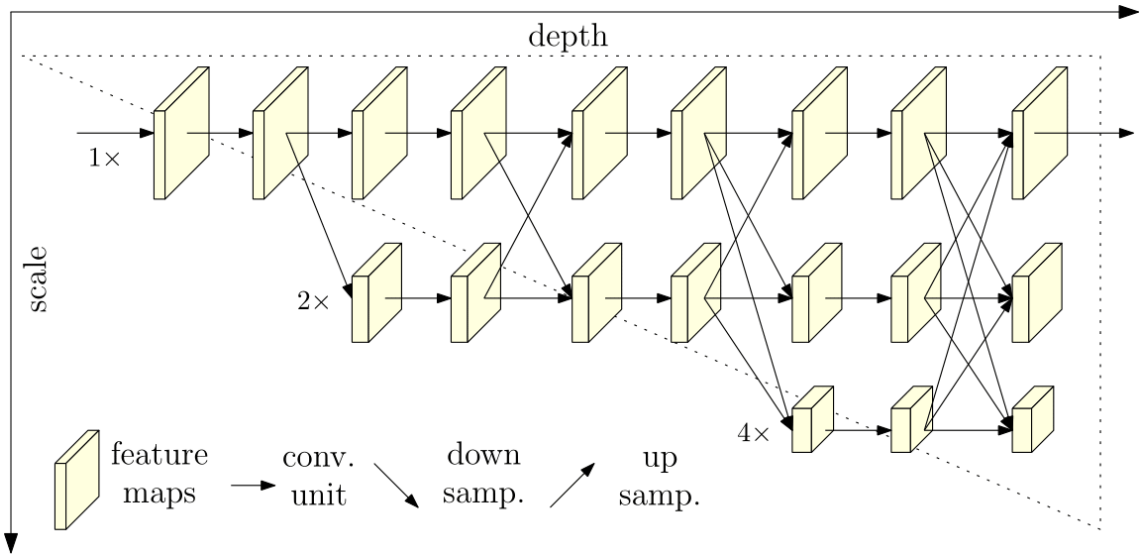


Figure 10: High-resolution network

Among the various architectures for key point detection, High-resolution Network (HRNet) has emerged as the dominant performer. This architecture excels in capturing the connections between local and global features, enabling it to deliver superior accuracy while maintaining computational efficiency. In this study, I only implemented and trained HRNet.

2.4.2 Dataset

Acquiring a diverse and representative dataset is crucial for training and evaluating the key points detection model. Significant efforts were invested in obtaining a dataset that encompasses various aspects, including different attachments, burs, background settings, and lighting conditions.

Projection In the data creation process, a crucial step involves projecting the available 3D information onto the 2D image. This transformation requires applying fundamental principles of linear algebra to convert the coordinate system to the camera's coordinate system and subsequently projecting it onto the 2D plane of the image. To account for distortion, the OpenCV implementation is used, utilizing distortion parameters: radial coefficients $(k_1, k_2, k_3, k_4, k_5, k_6)$, tangential distortion coefficients (p_1, p_2), and thin prism distortion coefficients (s_1, s_2, s_3, s_4).

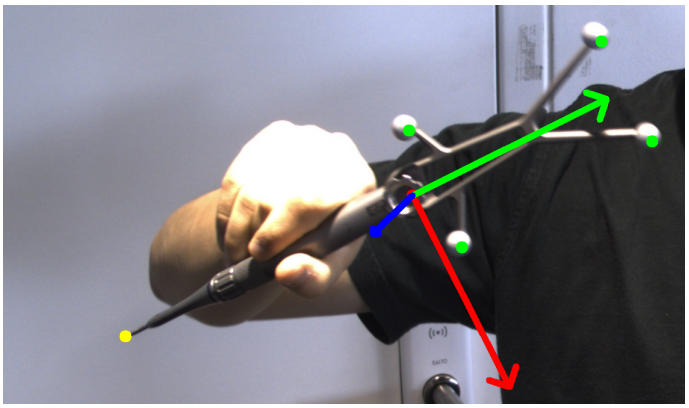


Figure 11: Projected points

Data cleaning Data cleaning is an essential process to ensure the quality and reliability of the dataset used for training and testing. Several methods were explored to identify and eliminate blurry images. Initial attempts involved testing blurriness scores obtained from Laplace kernels convolution or Fast Fourier Transform (FFT) methods, which assessed the sharpness of edges in the image. However, since the focus was specifically on determining the clarity of the tip, these scores did not yield satisfactory results.

An alternative approach was adopted, leveraging the high-frequency tracker information provided by the navigation system, which operates at a rate exceeding 300 Hz. By computing a motion score based on the tracker's movement, an approximation of the tip's motion could be obtained. This method proved to be effective in identifying and discarding blurry images.

Additionally, the navigation system provided a quality score for the tracker's position. Samples with a low quality score were discarded using a threshold to ensure that only reliable data was retained.

To further ensure accuracy, each sample underwent manual validation to verify the correct placement of all key points. The dataset contained seven key points: the four fiducials, the instrument center, the instrument bur center, and the instrument bur tip. While the bur tip's exact pixel alignment was often challenging, samples were retained if they had an error of under 2 pixels from manual annotation. The final training set consisted of 1,047 images, while the test set comprised 645 images.

Cropping and resizing To ensure efficient processing and take advantage of GPU parallelization, the input images were standardized to a fixed size. A resolution of 512x512 pixels was chosen as a suitable trade-off between maximizing input resolution and computational cost. An approximate bounding box is derived using the tracker position. Since the specific attachment being used is unknown, an estimation of the bounding box that considers all possible scenarios was generated. By projecting this approximation onto the 2D image, an estimation of the bounding

box was obtained. The image was then cropped, padded to create a square aspect ratio, and resized accordingly. The reduction ratio is about 0.4 to 0.7.

Data augmentation To enhance the diversity of the dataset, several augmentation techniques were applied on the fly before resizing the training images. Random flipping, contrast and brightness adjustments, random rotation, and random cropping were employed to introduce variations in the dataset. These augmentations helped to expand the dataset and improve the model's ability to handle different scenarios.

Photoshop augmentation To augment the dataset and increase its diversity, a set of images was captured in front of a blue drape. Color detection and the grab cut algorithm were utilized to generate segmentation masks, enabling the removal of the background and the hand of the user. Subsequently, random indoor images were used to replace the background through a process of photoshopping after random rotation and scaling.

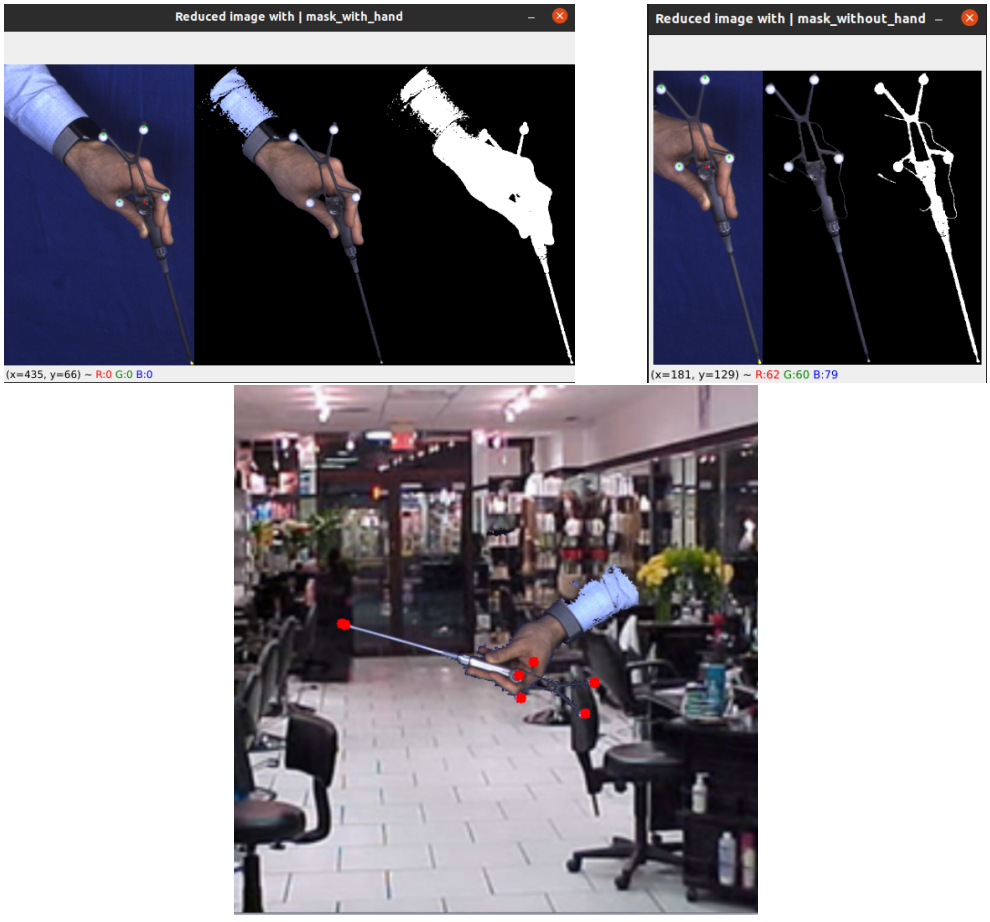


Figure 12: Segmentation masks from color detection and photoshop augmentation

2.4.3 Method and experiments

Architecture The key points detection model was implemented using the HRNet architecture in PyTorch, utilizing the original repository. However, to adapt the model for key point detection in surgical instruments, the last layer was modified to match the required number of output channels.

Initialization Traditionally, the HRNet model is trained in a two-step process, starting with classification and then fine-tuning for key point detection. In this project, the focus was solely on key point detection, without incorporating a classification pipeline. Hence, the training was directly performed for key point detection. However, the weights were initialized with pre-trained weights from human pose estimation. Starting from trained weights is advantageous compared to randomly initializing the network, as it allows for the transfer of knowledge and facilitates faster convergence during training.

Ground truth and maximal precision The ground truth key points obtained through the data cleaning pipeline are expected to exhibit subpixel precision despite the presence of noise. In order to preserve this subpixel precision during the generation of ground truth heatmaps, a technique was employed to maintain continuous values before discretizing them into a Gaussian heatmap. As a result, the generated ground truth heatmaps possess an infinite range of value possibilities. During training, the network learns to predict the mean of the Gaussian heatmap rather than specific pixel-centered values. Considering a 512x512 input image, the output is a 64x64 heatmap, which is four times smaller on each dimension. To approximate the mean value, the maximum pixel value is utilized. To compensate for the reduction in size, a small offset of 0.25 is added before multiplying by 4 to bring the values back to the scale of the original image.

The application of postprocessing techniques directly on the ground truth heatmap results in an error range of +/- 1 pixel. This error range serves as an indication of the limitations of HRNet.

On the following figure, red > orange > yellow. The purple point is the point of maximum value. The argmax is corrected into the green point by comparing the values of the 4 adjacent pixels.

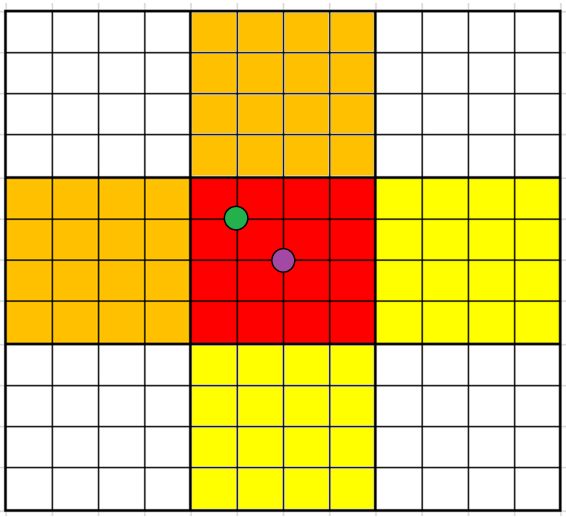


Figure 13: Heatmap postprocessing

Metrics In the existing literature, the focus of key point detection is primarily on semantic key points, resulting in evaluation metrics that heavily emphasize precision and recall of the overall detection. Conversely, in this study, it was assumed that instrument detection would not pose significant challenges. Therefore, the primary objective was to measure the precision of the predictions directly. To achieve this, the mean squared error (MSE) between the ground truth key point and the predicted key point was selected as the chosen metric. This metric provides a straightforward indication of the precision of the predictions, enabling a more focused evaluation.

W&B tracking and fine-tuning The *Weights and Biases* platform is a useful tool for keeping track of experiments and allows for flexible visualization of metrics. This is especially beneficial when conducting a large number of experiments. Additionally, the platform provides the capability to automate experiments using Random search. Initially, Random search was employed to explore a range of hyperparameters and gain an understanding of the optimal settings. Subsequently, a manual fine-tuning approach was adopted to refine the hyperparameters based on the acquired knowledge.

Figure 14: Weights & Biases

Losses Traditionally, the Mean Squared Error (MSE) loss is commonly used to measure the discrepancy between predicted and ground truth heatmaps. This loss calculates the average squared difference between corresponding pixel values in the predicted and ground truth heatmaps.

$$MSE(y_{pred}, y_{gt}) = \frac{1}{n}\sum_{i=1}^{n}(y_{pred}^{(i)} - y_{gt}^{(i)})^2 \tag{1}$$

In addition to the MSE loss, I also implemented several derived losses that have shown effectiveness in previous research. One such technique is Online Hard Keypoint Mining (OHKM), which computes the MSE loss separately for each key point in the image and only considers the k highest loss values. This encourages the network to focus on the most challenging key points in each image. Another approach, Global Hard Keypoint Mining, follows a similar principle but selects the k' highest loss values across the entire batch. Furthermore, I experimented with a weighted loss that assigns greater importance to the tip key point, as it is the primary focus of our method. It is worth noting that the official HRNet paper utilizes the OHKM loss.

After evaluating different loss functions, it was found that the weighted loss performed the best in terms of metric performance on the tip key point. This is likely because achieving optimal results for all key points simultaneously proved to be challenging for the network. Interestingly, convergence was noticeably slower when training without considering all key points together. Training on multiple key points simultaneously helped the network gain a better understanding of the instrument being detected.

In the original implementation, when a key point is absent from the image, the loss is not computed for that specific key point. However, including a loss for predicting the absence of a key point, rather than not optimizing it at all, yielded slightly improved results.

Half-precision and mixed precision The concept of half-precision involves converting all weights and data to float16 before performing computations. However, it is well-known that half-precision can be unstable, particularly when Batch Normalization is utilized. To address this issue, a workaround called mixed precision is employed instead of pure half precision. In mixed precision, certain operations are performed in float16, while others are executed in float32. The implementation remains the same, as the division of precision is handled automatically.

Both half-precision and mixed-precision prove to be beneficial when working with large neural networks that require extensive fine-tuning. Mixed precision, in particular, can save significant training time, approximately 30% with HRNet. However, during my experiments, I encountered a problem where the networks did not converge at all when using mixed precision. While this issue may have required further investigation, I decided to continue using full precision.

Training set distribution and data augmentation As previously mentioned, the training set comprises a combination of real images and synthetic data generated from images captured in front of a blue drape. Initially, I utilized a ratio of 1/3 real data and 2/3 synthetic data. However, this led to significant instability in performance when evaluated on the test set. The instability can be attributed to a substantial distribution shift caused by the synthetic data. The photoshopped backgrounds and imperfect segmentation masks introduced dissimilarities not found in real images, such as remaining blue pixels.

To mitigate this issue, I conducted training experiments using different proportions of synthetic data. Training the model without any synthetic data resulted in poorer performance on the test set. After fine-tuning, I discovered that incorporating 2/5 synthetic data struck a good balance between stability and generalization.

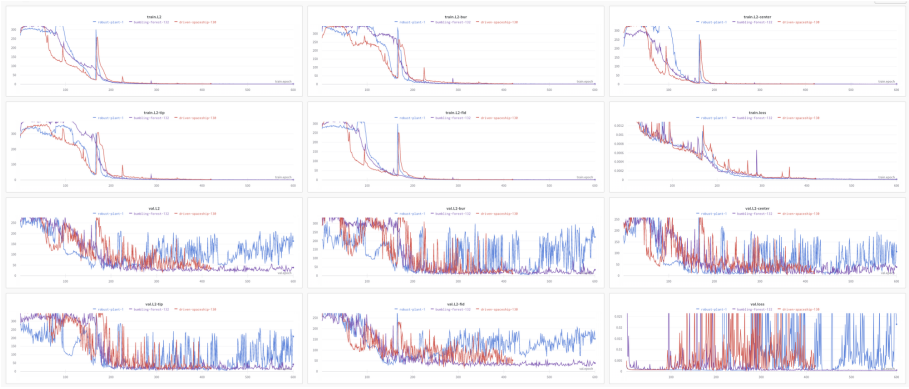


Figure 15: Runs with different proportions of synthetic data

Furthermore, the data augmentation techniques described earlier have a significant impact on the metrics when evaluated on the test set. It is evident from the metrics that the models achieved a 2-pixel error on the train set, regardless of whether data augmentation was applied or not. However, the performance on the test set greatly improved with the inclusion of data augmentation. Rotation, brightness, and contrast adjustments had a notable impact on the results. This can be attributed to the presence of a distribution shift between the training and test data.

Confidence threshold In the specific application I am designing, it is not necessary to use all frames of the video. I have the flexibility to discard frames in which the prediction does not meet the desired level of accuracy.

The mean of the Gaussian distribution is estimated using the Argmax operation, but there is additional information available in the predicted heatmap. When dealing with challenging samples, it is observed that the output values in the heatmap tend to be lower. Therefore, the maximum value of the heatmap can serve as a confidence score for the prediction.

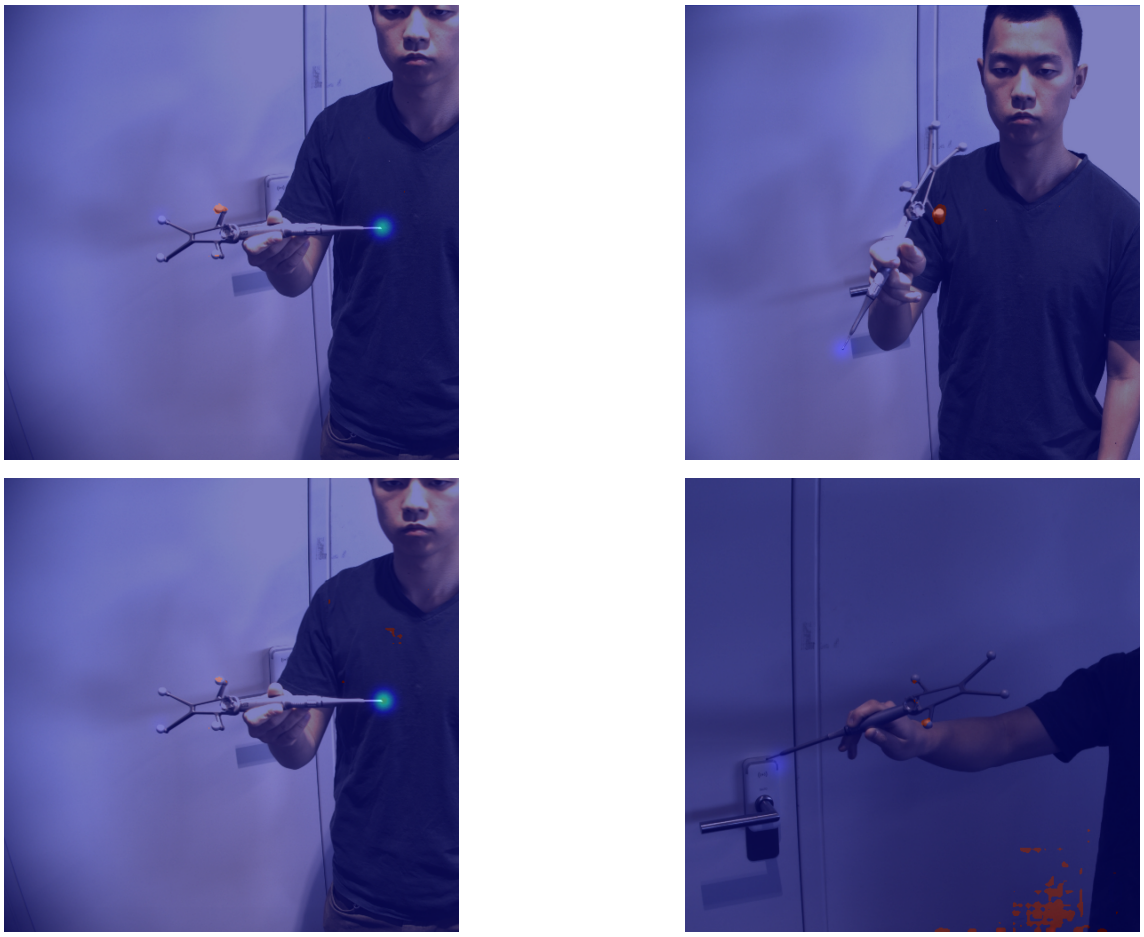


Figure 16: Tip heatmap visualization

However, it should be noted that relying solely on the maximum value may not always provide the most accurate confidence score, especially for out-of-distribution samples where the predicted heatmaps may not resemble a Gaussian distribution. The discussion section on future improvements presents an alternative approach for obtaining a confidence score.

Multi flipping inference and training The network output is designed to be independent of flipping. Ensembling, which involves predicting multiple times with slight variations and then averaging the predictions, is known to be effective in reducing errors. Therefore, during inference, I adopted a multi-flipping approach where I inferred on the four flips, averaged the heatmaps, and obtained the final predicted coordinate from the average heatmaps.

The visualization of the error revealed that multi-flipping inference had minimal impact on the errors. Mathematically, averaging predictions corresponds to reducing variance. The lack of significant difference in the errors suggests that the errors are

primarily influenced by bias rather than variance.

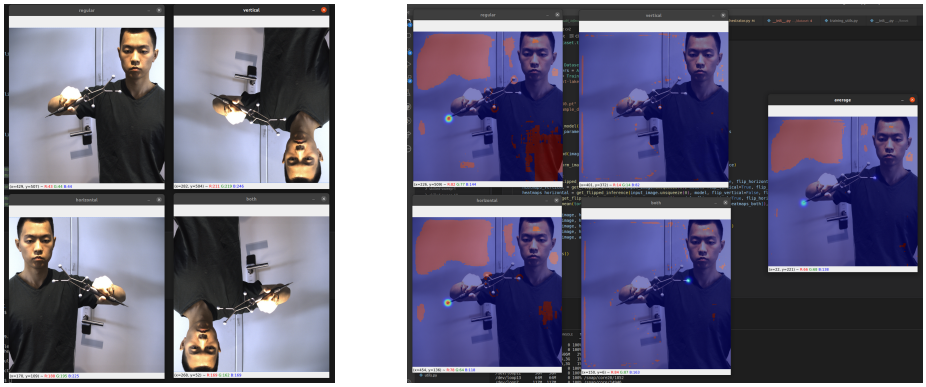


Figure 17: Multi-flipping inference

I also explored modifying the training process. One approach was to train the network on all four flips simultaneously for each sample, which required reducing the batch size. Surprisingly, this approach yielded improved results for all key points, even when using OHKM loss instead of the weighted loss. Another approach, called siamese training, involves taking the average before computing the loss. However, I did not implement this method as I had already reached the limit of what could be achieved with HRNet and the available data annotations.

2.4.4 Results

The best-performing model utilized all the techniques discussed in the preceding section, including OHKM and multi-inference. It achieved an MSE of 2.6 pixels for the tip.

When examining the qualitative visualization of the worst samples, it is evident that the network struggles more in extreme lighting conditions, such as overly bright or dark environments. This issue is likely attributed to the automatic brightness and contrast calibration of the camera. To address this, one potential solution would be to correct the brightness and contrast before inputting the images into the network. Additionally, augmenting the data further may help mitigate this problem and result in a more robust network.

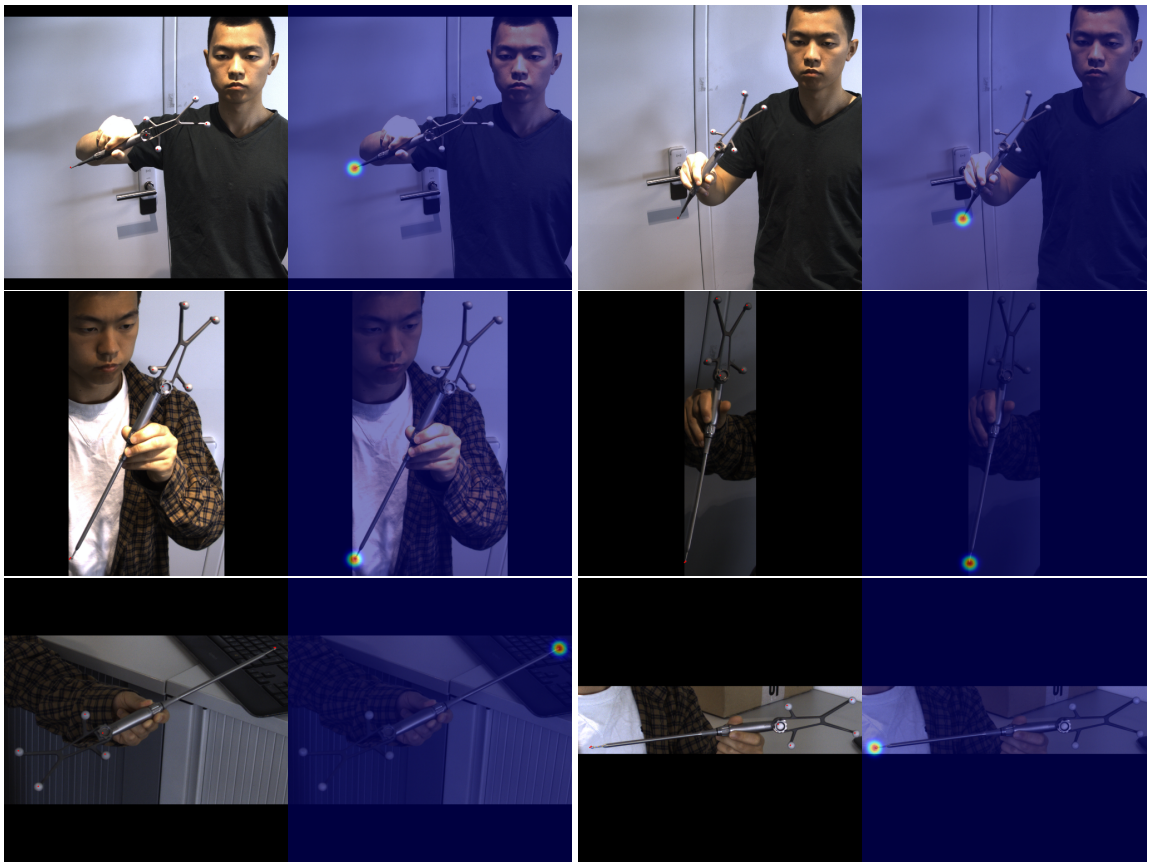


Figure 18: Random test set samples visualization

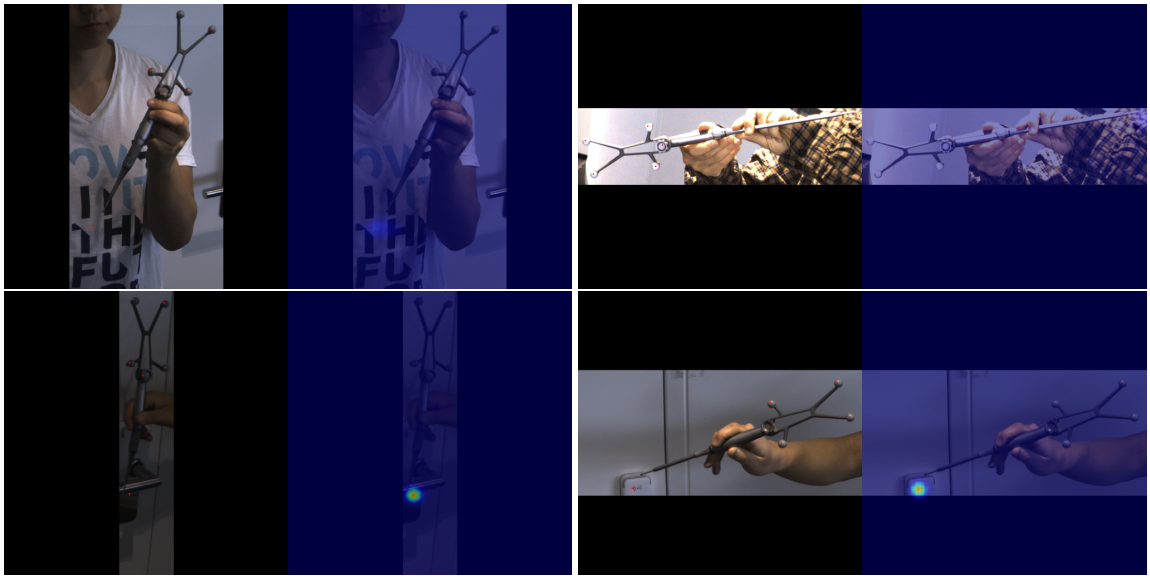


Figure 19: Worst test set samples visualization

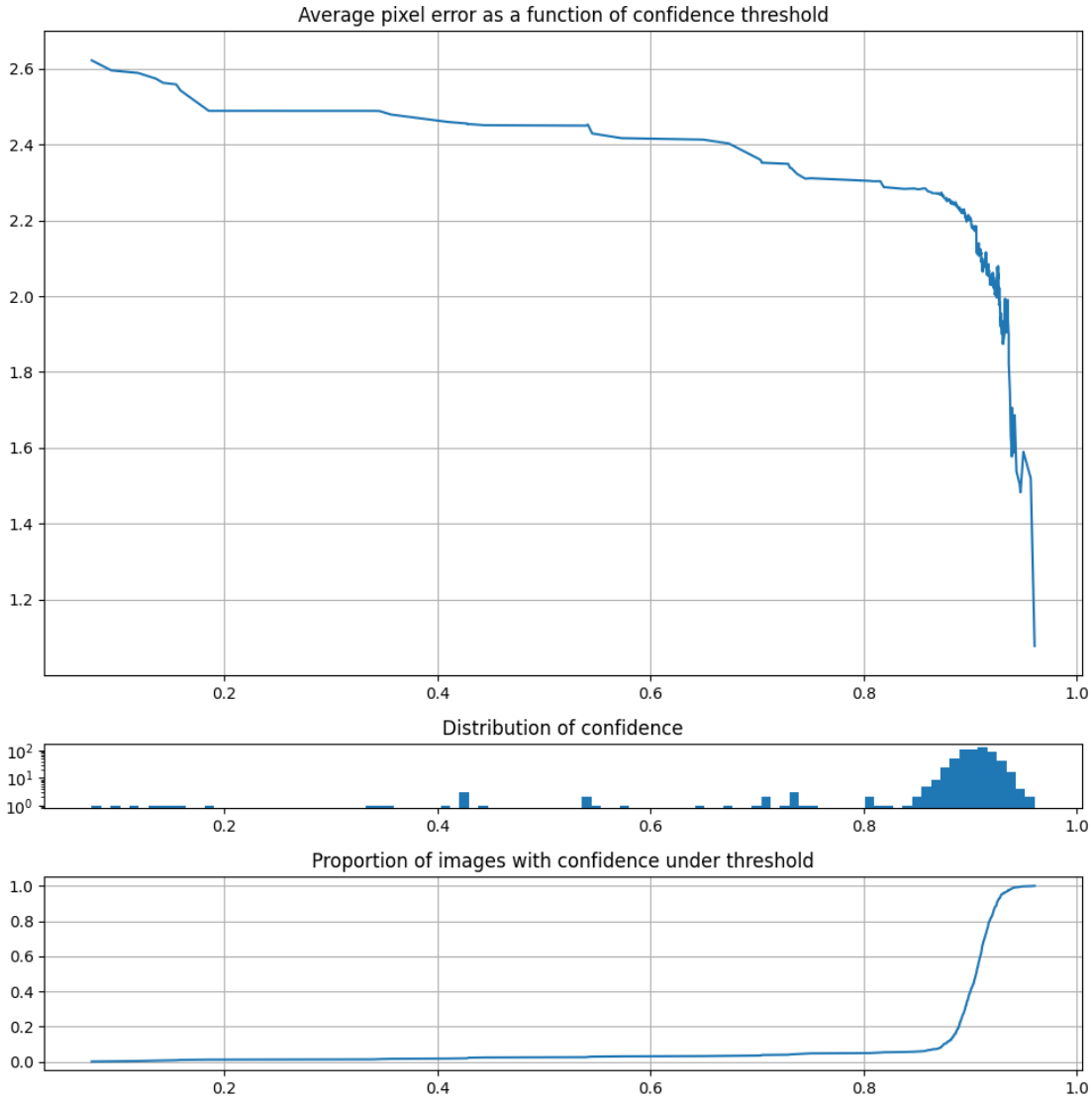


Figure 20: Confidence distribution

The presented figure displays the distribution of confidence scores and their association with the error. The curve demonstrates the network's robustness, as the average error remains relatively stable across different confidence threshold values.

2.5 Triangulation through time

2.5.1 Idea

To create the key points annotation, I used the camera's extrinsic and intrinsic matrices, along with its distortion coefficient, the 3D coordinates of the tracker, and the transformation from the tracker to the tip. Now, I am attempting to do the

opposite: having all the necessary information, including the projected position of the tip on the image, and determining the 3D transformation from the tracker to the tip.

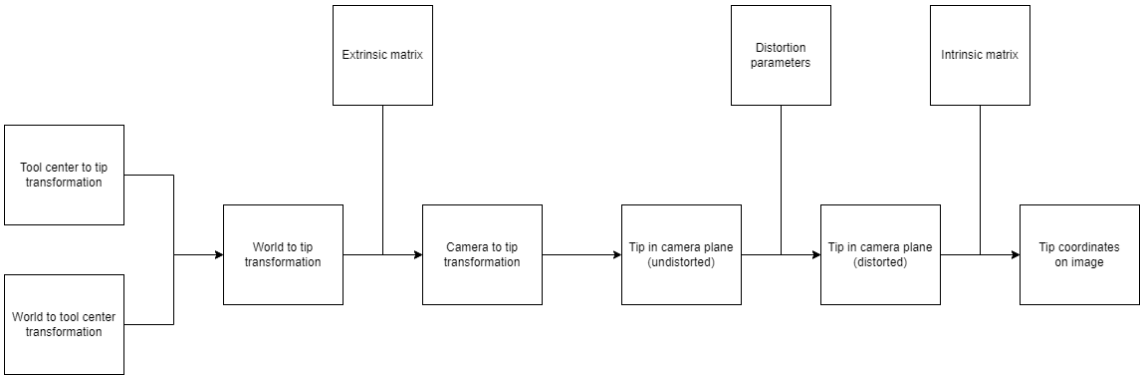


Figure 21: Projection pipeline

This process is quite similar to 3D triangulation from two cameras, except that we are working with just one camera. In essence, each image provides a 3D line along which the tip should lie. Mathematically, this corresponds to having two constraining equations. Analyzing multiple frames makes it possible to compute the tip's position by minimizing the distance to all the lines.

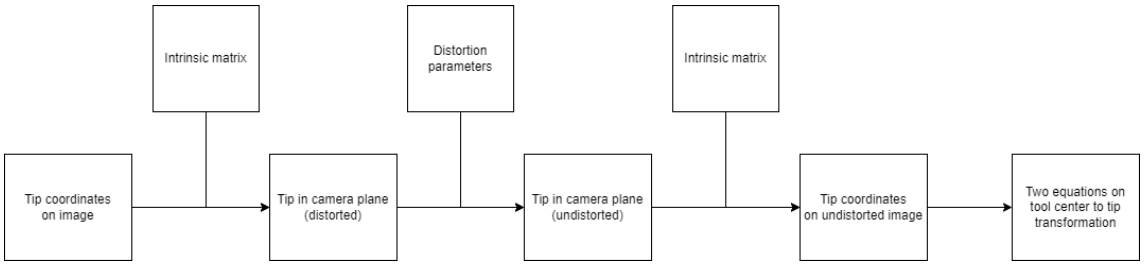


Figure 22: Calibration pipeline

2.5.2 Method

Undistortion All the calculations involved in the process can be considered affine algebra, with the exception of camera distortion. Hence, the initial step is to eliminate the effects of camera distortion.

Thankfully, there are highly accurate computational models available for this purpose. I experimented with Brown's Conrady Model, but it did not perform as well as the OpenCV formula. This is because Brown's Conrady Model only considers radial and tangential distortion parameters, whereas OpenCV also incorporates thin prism distortion coefficients.

In my experiments, (*distort*) *o* (*undistort*), yielded results very close to the identity, with an error of approximately 0.1 pixel. It would be beneficial to conduct further extensive testing to thoroughly evaluate its performance.

Writting the equations

- Let F represent the intrinsic matrix of the camera, given by

$$F = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- T_c and R_c denote the translation and rotation, respectively, of the tool center in the camera's coordinate system.
- t represents the unknown translation from the tool center to the tip of the tool.

According to projective geometry, we know that

$$\psi \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = F(T_c + R_c t)$$

where x and y are the coordinates of the tip on the image, and ψ is unknown.

By multiplying both sides by $\begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$, we obtain

$$\psi = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} F(T_c + R_c t) = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} (T_c + R_c t)$$

Let's note

$$B = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Therefore, we have

$$\psi \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = B(T_c + R_c t)$$

Additionally, we consider

$$D = \begin{pmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thus, we can express

$$\psi \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = D\psi \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = DB(T_c + R_c t)$$

By replacing this expression in the first equation, we obtain

$$DB(T_c + R_c t) = F(T_c + R_c t)$$

As a result, we have

$$(DB - F)T_c + (DB - F)R_c t = 0$$

It can be verified that the last row of $DB - F$ is always zero, meaning that we have exactly two non-zero equations. The other rows are non-zero since the diagonal of $DB - F$ is negative, ensuring that the rank is always 2.

Optimization problem For every image, these equations remain valid. As a result, when dealing with more than two images, we encounter an overdetermined problem. Each image provides a 3D line parametrized by the two equations. As the number of lines exceeds two, we can determine the optimal point by minimizing the L2 distance to all lines. This optimization problem is convex, allowing for a unique and analytically solvable solution.

Indeed, we can define

$$M = \begin{pmatrix} [(DB - F)R_c]_{1:2} \\ \vdots \end{pmatrix}$$

$$b = \begin{pmatrix} [(DB - F)T_c]_{1:2} \\ \vdots \end{pmatrix}$$

Here, M is a $(2n, 3)$ matrix and b is a $(2n, 1)$ vector.

It is important to note that all the matrices involved are in the world coordinate system, which differs from the camera coordinate system. Hence, it is necessary to use the following transformations:

$$R_c = R_{cam} R_{center}$$

$$T_c = R_{cam} T_{center} + T_{cam}$$

In an ideal scenario, we would have

$$Mt + b = 0$$

The optimization problem aims to minimize $\|Mt + b\|^2$.

$$\nabla_t \|Mt + b\|^2 = 2M^T Mt + 2M^T b = 0$$

Solving this equation yields

$$t = -(M^T M)^{-1} M^T b$$

The matrix $M^T M$ is a square symmetric non-negative matrix with a rank of at least 2. In practical scenarios, it is highly improbable for this matrix to be non-invertible. The condition for ensuring its invertibility is that the positions should not align on a 3D line from the camera's perspective.

Mathematical interpretation Mathematically, the optimization problem we are considering can be viewed as a hyperplane optimization rather than a line optimization problem. Each equation represents a hyperplane in 3D space, which corresponds to an infinite 2D plane in the 3D world. Having two such equations is equivalent to having a line in 3D space, which is defined as the intersection of the two planes.

In linear algebra, a hyperplane is defined by a vector a and a scalar b. A point x belongs to the plane if and only if $a^T x + b = 0$. The minimal distance from a point x in 3D space to the plane is given by $\frac{|a^T + b|}{|a|}$. In this context, I chose not to divide by the norm of the normal vector because the calculations are already in millimeters, and the scale holds significance. However, dividing by the norm could also be an interesting option to explore.

Minimizing the L2 distance is equivalent to minimizing the mean squared error (MSE), which can be interpreted as imposing a Gaussian prior on all hyperplanes. The L2 distance aims to distribute the error among all samples, rather than having perfect accuracy on some and worse accuracy on others. It might also be worthwhile to consider testing the L1 distance, which is more robust to outliers.

Weighting Additionally, one can consider assigning weights to each line based on the uncertainty score of the prediction.

This can be achieved by minimizing $\|M't + b'\|^2$ with $M' = WM$ and $b' = Wb$, where W is a weight matrix

$$W = \begin{pmatrix} w_1 & 0 & 0 & \cdots & 0 \\ 0 & w_1 & 0 & & \\ 0 & 0 & w_2 & & \\ & & & w_2 & \\ \vdots & & & \ddots & \\ & & & & \\ 0 & 0 & & & w_n \end{pmatrix}$$

Here, w_1, w_2, ..., w_n represent the weights for each image.

Although I implemented this approach, it did not yield any significant changes. However, it may be worth evaluating its potential in some scenarios.

2.5.3 Results

To create a realistic scenario, I assessed the pipeline using a fixed number of N consecutive frames (after post-processing), ensuring that the pipeline is executed precisely on N predictions surpassing the confidence threshold.

Throughout the rest of the study, the ground truth is determined using the calibration method, which may have an inherent deviation of up to 2 mm.

Video duration The time required to capture these N frames can be estimated as

$$T(N) = \frac{N}{\text{filtering\_proportion} \times FPS}$$

where

$$FPS = 10$$

and

$$\text{filtering\_proportion} = \text{filter\_qual\_blurry} \times \text{filter\_postprocess} = 0.3 \times 0.8 = 0.24$$

Consequently,

$$T(N) = 0.4 \times N\text{seconds}$$

Number of frames influence As mentioned previously, the utilization of more than two frames effectively incorporates a Gaussian prior for each prediction, mitigating errors. The following provides an overview of the error distribution for various numbers of frames:

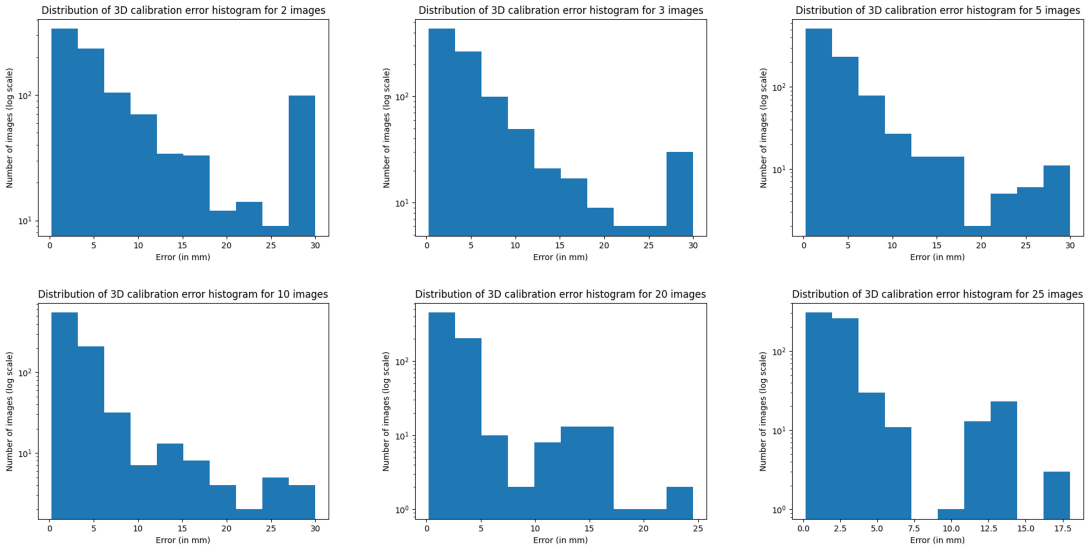


Figure 23: Calibration error distributions for different number of frames

Considering the average error as a singular metric for each sample size:

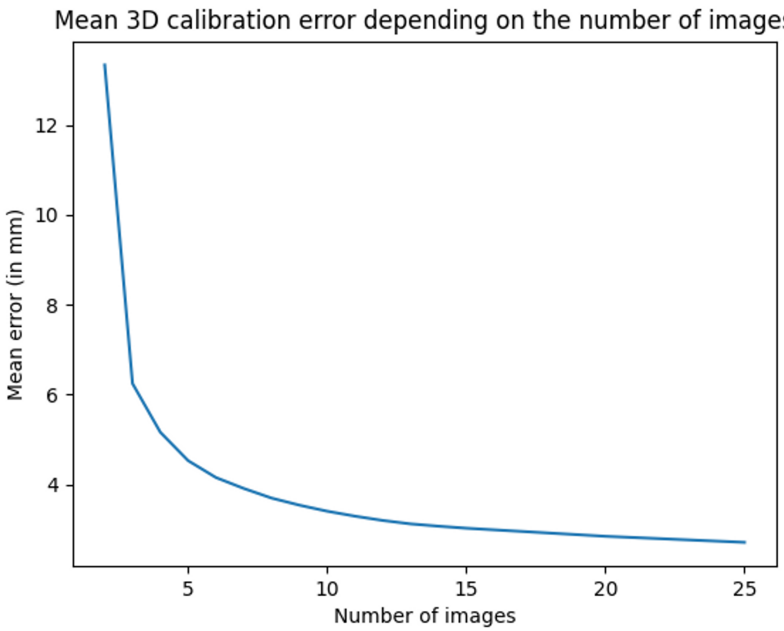


Figure 24: Average calibration error for each sample size

The average error experiences a significant reduction and reaches less than 3.0 mm for 25 frames, corresponding to a 10-second video duration.

2.6 Future improvements

This section focuses on potential enhancements that can be implemented in the pipeline.

2.6.1 Errors analysis for better annotations

The annotated samples obtained from the real calibration procedure and projection exhibit an error of up to 5 pixels, equivalent to approximately 2.5 millimeters. The challenge arises from the fact that 5 pixels represent a significant deviation on an image, potentially causing confusion for the network during training.

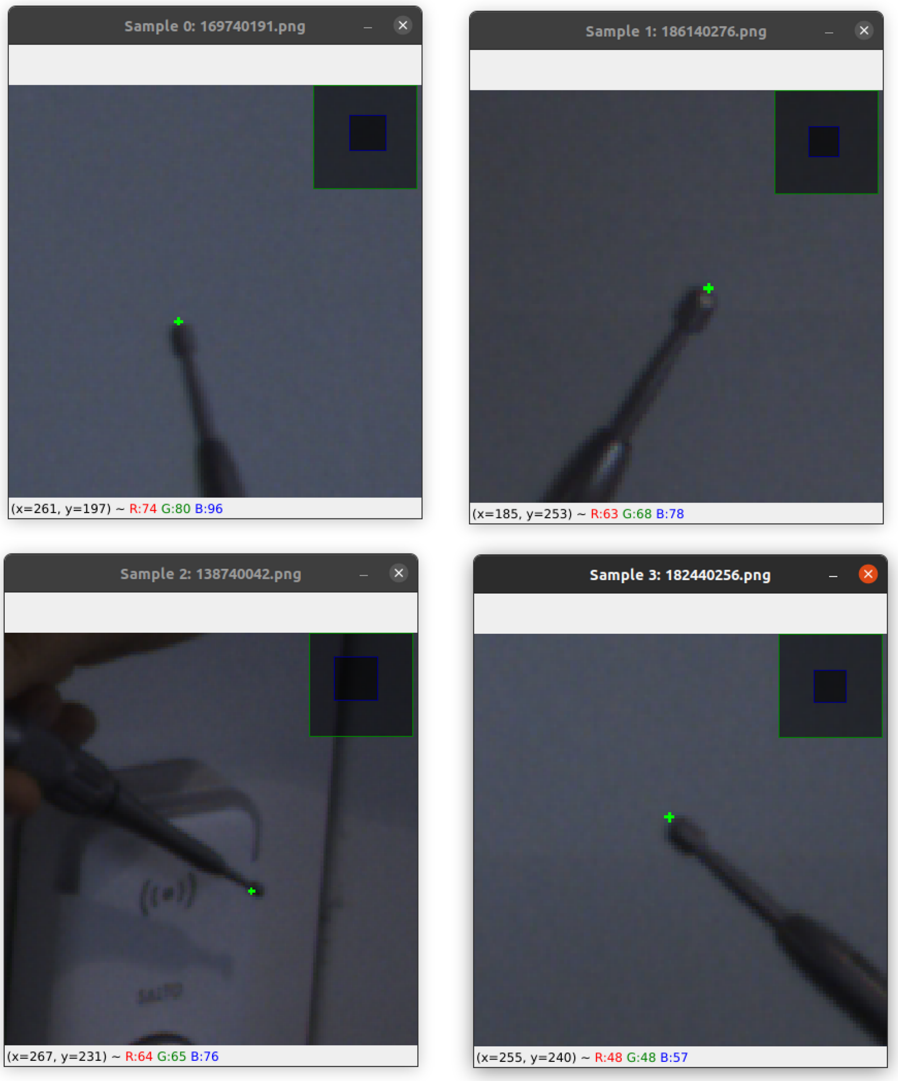


Figure 25: Annotation error visualization

Sources of error Here is a breakdown of potential sources of error in the pipeline:

- Camera calibration: The accuracy of the NIR camera calibration is determined by the quality factor, which is measured by the mean reprojection error of the chessboard corners. This factor reflects the confidence in the accuracy of the intrinsic matrix, extrinsic matrix, and distortion parameters. The RGB camera can also undergo calibration.
- Fiducials and instrument center: The 3D positions of the trackers are obtained by triangulating the fiducials using the NIR cameras. The tool center is then determined based on the known relative positions of the fiducials. This process is integrated into the existing navigation pipeline, which has been verified to provide a precision of less than +/-2mm.
- Instrument calibration error: The previous calibration procedure uses fiducials along with a simple algorithm to get the 3D tip position with two calibration points in the 3D space.
- Timestamp offset: A small offset in the timestamps between the RGB frame and the fiducial tracking can occur due to the rolling shutter mechanism of the camera. The image is captured line by line on the sensor, leading to a slight temporal misalignment. This offset should be taken into account when processing the data.

Static instrument This initial experiment aimed to identify the sources of errors in the system. To eliminate the error caused by the timestamp offset, a technique was employed whereby the instrument was placed on a stationary table, and data was captured with the static tool.

Multiple calibrations were performed to assess if the error could be attributed to the calibration procedure. However, the same video capture was used for all calibrations.

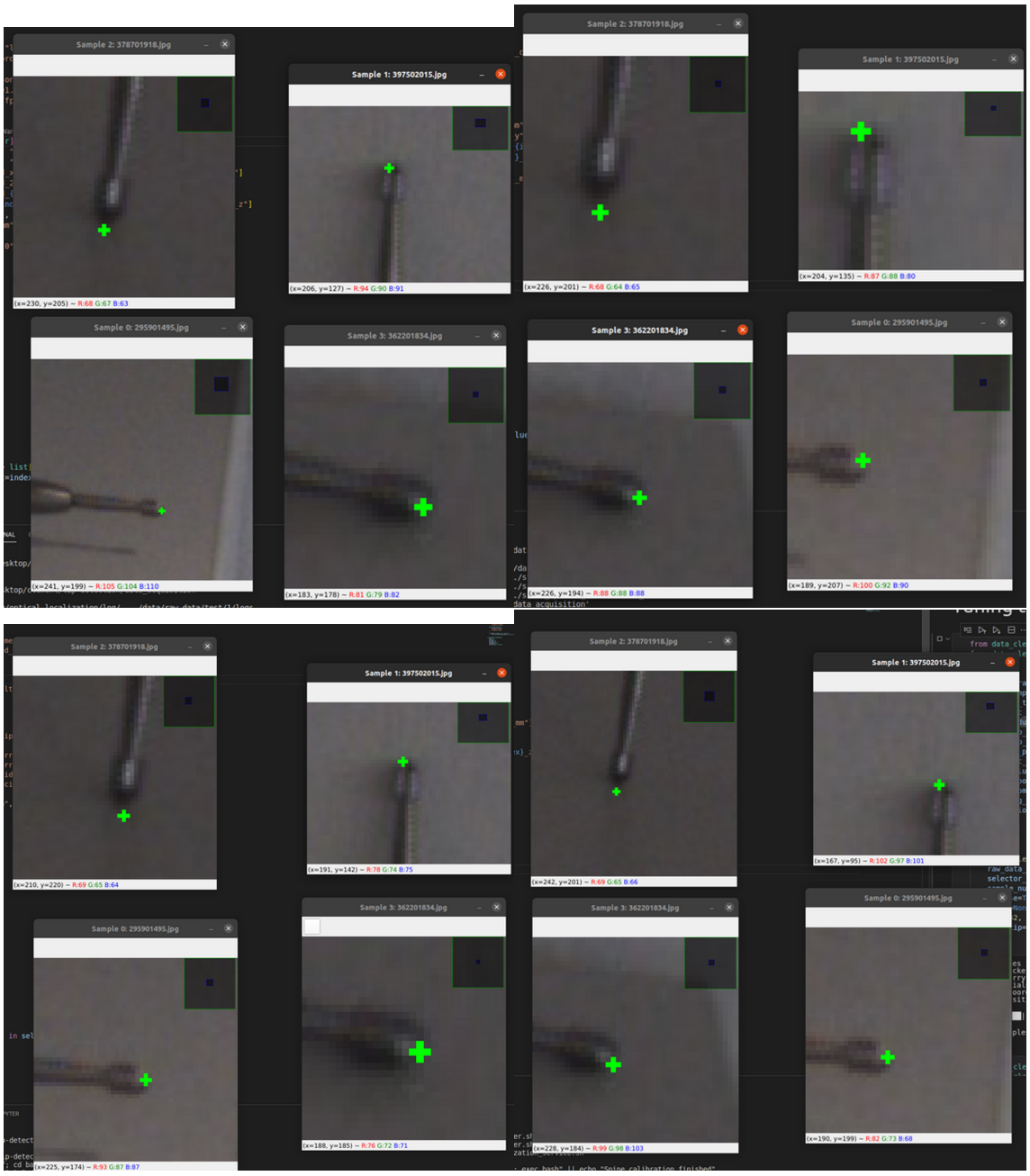


Figure 26: Reprojection visualization for independent calibrations

Upon examining the top-right image, it can be observed that the error remains consistent across all calibrations. This suggests that the error may stem from factors such as the tool center or camera parameters. Overall, the variations between different calibrations were found to be negligible.

Moving timestamp As mentioned previously, the camera used in the system employs a rolling shutter mechanism to capture images. The image acquisition process occurs line by line, with each line requiring a specific amount of time known as the "integration time". This integration time can vary depending on factors such as sensor saturation.

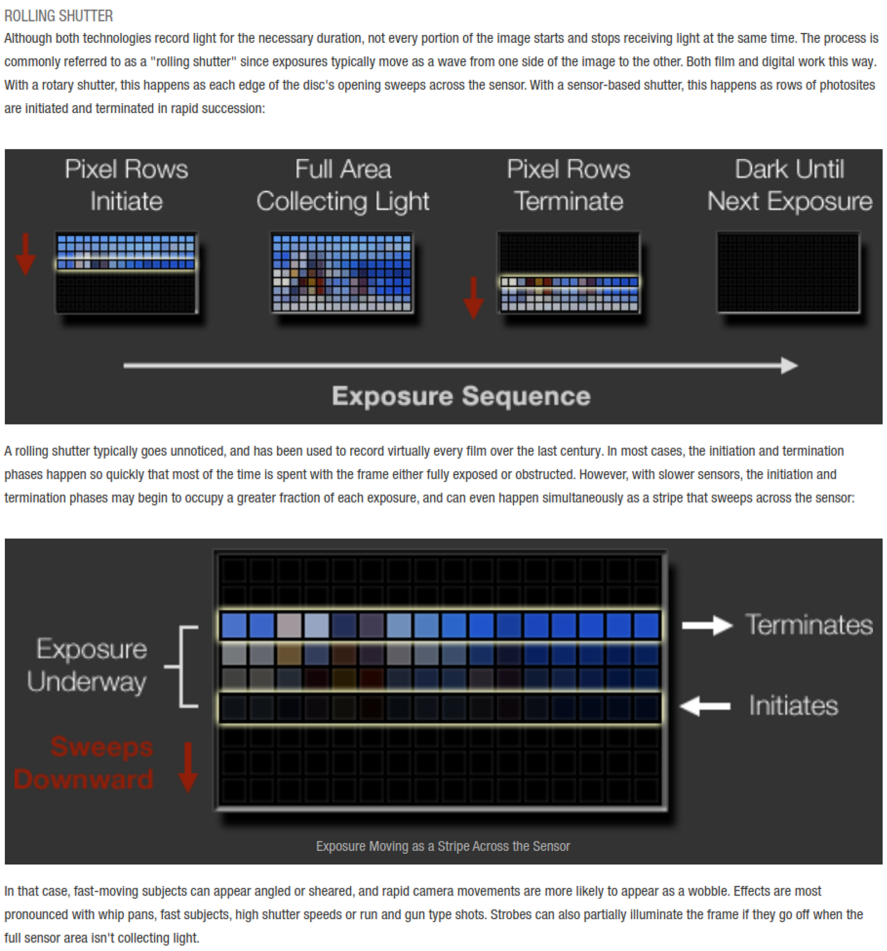


Figure 27: Rolling shutter

Under normal lighting conditions, the expected integration time typically ranges between 20 to 30 milliseconds. The remaining values necessary for the calculation are already known or available.

$$\text{tot\_time} = 1/\text{fps} = 0.1s$$

$$\text{tot\_lines} = 3840$$

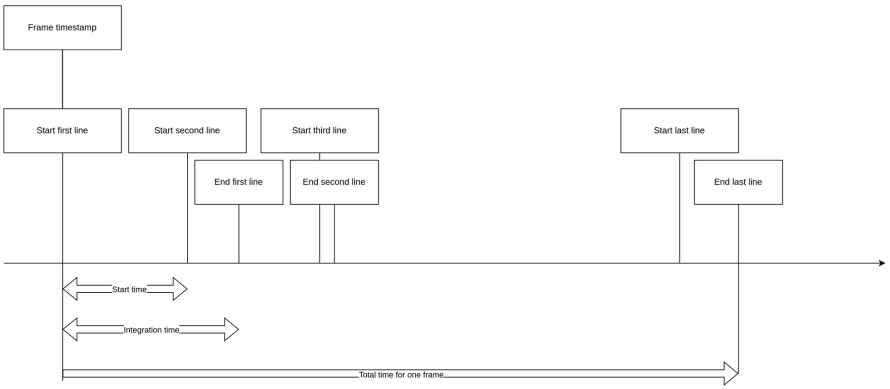


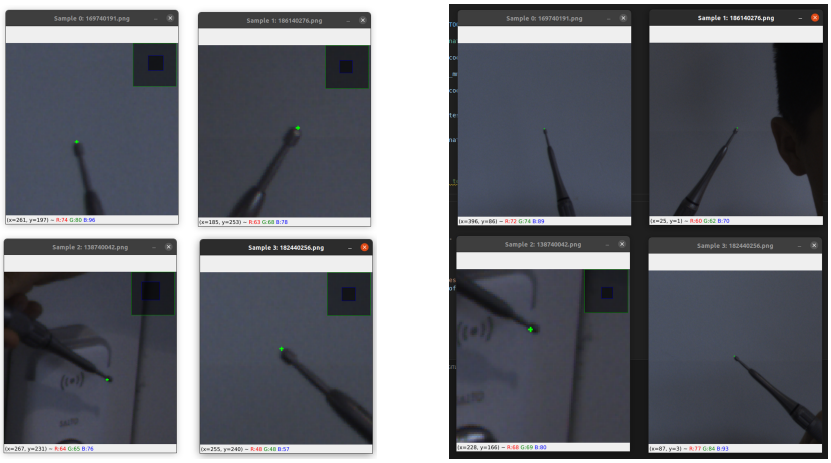
Figure 28: Acquisition schedule

$$\text{tot\_time} = \text{start\_time} \times \text{tot\_lines} + \text{integ\_time}$$

$$\text{line\_offset} = \text{start\_time} \times \text{line\_number} + \text{integ\_time}/2$$

$$\text{line\_offset} = (\text{tot\_time} - \text{integ\_time}) \times \text{line\_number}/\text{tot\_lines} + \text{integ\_time}/2$$

In practical implementation, obtaining the line number requires a two-step process. Initially, a projection is performed to approximate the line number, followed by a second projection using the correct timestamp.



(a) Manual constant offset (b) Automatic dynamic offset

Figure 29: Offset comparison

Interestingly, adjusting the timestamp with a dynamic offset did not result in significant changes compared to a constant offset. This implies that the error is likely originating from other factors. Nevertheless, I have not conducted a comprehensive analysis to quantify the disparity between manual offset and automated dynamic

offset. It is possible that the difference would have been more pronounced if the instrument tip was located closer to the edge of the image.

Based on the two aforementioned experiments, it can be inferred that the majority of the errors stem from the movement of the fiducials during instrument motion.

Noisy labels in deep learning The utilization of noisy labels in deep learning, known as weakly supervised learning, has become increasingly prevalent in the literature. This approach involves training a large-scale model on a substantial amount of noisy data, conducting inference, and refining the annotations by retaining the top predictions. Notable examples of this technique include whisper in speech-to-text, CLIP, and the GPTs. In the most successful scenarios, the model is capable of mitigating the effects of noise and achieving remarkable robustness. However, it is important to note that these models may still produce unexpected outputs, highlighting the need for careful evaluation and analysis.

2.6.2 Postprocessing with likelihood

Applying the Argmax operation to estimate the mean and using the Maximum to obtain a confidence score is not the most optimal approach for postprocessing Gaussian heatmaps. Instead, we can leverage mathematical and statistical techniques to compute a likelihood score that quantifies the degree of Gaussianity in the output heatmap.

A more accurate method involves fitting a truncated continuous 2D Gaussian distribution based on likelihood estimation. To accomplish this, some mathematical calculations are necessary.

Let us denote the output heatmap of the network as (H_{ij}), and let K_h be a constant such that $K_h \cdot \sum_{ij} H_{ij} = 1$.

The general k-dimensional Gaussian distribution can be expressed as follows:

$$g(y|M,\Sigma) = \frac{1}{\sqrt{(2\pi)^k \det(\Sigma)}} \exp[-\frac{1}{2}(y-M)^T\Sigma^{-1}(y-M)]$$

for $y \in \mathbb{R}^k$.

If our intention was to directly fit the distribution, the mean would be estimated using:

$$M = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} = \bar{x}$$

However, this approach would yield poor results when the key point is located near the image border.

Instead, we will attempt to fit a truncated Gaussian distribution. The density function of this distribution should be proportional to:

$$g(y|M,\Sigma).\mathbf{1}_{[0,a]^2}(y)$$

In order for the density to satisfy the requirement of summing to one and representing a probability distribution, we need to introduce a constant term $K(M,\Sigma)$ such that

$$\int_{[0,a]^2} K(M,\Sigma)g(y|M,\Sigma)dy = 1$$

After simplification and renaming, we obtain the following expression:

$$f(y|M,\Sigma) = K(M,\Sigma).g(y|M,\Sigma).\mathbf{1}_{[0,a]^2}(y)$$

$$K(M,\Sigma) = \frac{1}{\int_{[0,a]^2} \exp[-\frac{1}{2}(y-M)^T\Sigma^{-1}(y-M)]dy}$$

$$g(y|M,\Sigma) = \exp[-\frac{1}{2}(y-M)^T\Sigma^{-1}(y-M)]$$

Now that we have everything in place, we can compute the likelihood by treating the heatmap as a histogram of observations.

$$L(Y|M,\Sigma) = \prod_{i,j} f(y_{ij}|M,\Sigma)^{H_{ij}}$$

It is important to note that K_h disappears from the likelihood equation because it is a global constant.

$$\log L(Y|M,\Sigma) = -\frac{1}{2}\sum_{ij} H_{ij}(y-M)^T\Sigma^{-1}(y-M) - (\sum_{ij} H_{ij}) \log(\int_{[0,a]^2} \exp[-\frac{1}{2}(y-M)^T\Sigma^{-1}(y-M)]dy) \quad (2)$$

By maximizing the log-likelihood, we can estimate the mean and variance of the distribution. The maximum value obtained corresponds to a score indicating how Gaussian the output is.

Alternatively, another approach is to estimate the parameters using the Wasserstein distance, which aligns more closely with human intuition.

2.6.3 Refinement network

One approach to improving the results is to enhance the annotations. Manual annotations could be explored as a potential solution. It is important to note that HRNet has a maximum precision of +/- 1 pixel due to its output being four times smaller than the input in all dimensions. Simply adjusting the size of the output heatmap to match the input size could potentially improve the precision. However, this adjustment may also increase the difficulty of training, and the image would still need to be resized to the fixed size of 512×512, resulting in a loss of pixel precision during the resizing process.

A more comprehensive approach involves utilizing HRNet as a key point detector and employing a secondary network to refine the predictions within a square region of the original resolution surrounding the HRNet prediction. The detector network could directly regress the position, leveraging the smaller resolution of the region for more accurate results.

2.6.4 Multi view deep learning based calibration

Instead of relying on frames at different timestamps for triangulation, a more direct approach would be to utilize NIR cameras for the triangulation process. However, this approach would require transmitting all NIR images instead of solely sending the coordinates of the fiducials. Transmitting the complete NIR images would decrease the precision of the navigation pipeline. Additionally, this approach would be limited to obtaining only three measurements for the tip, which may restrict the accuracy of the triangulation.

3 Subpixel detection for trackers

3.1 Problem statement

To determine the position of instruments, trackers and NIR cameras are utilized. In NIR images, the trackers' reflected light appears as colored blobs, and accurately determining the centroid of these blobs is crucial. This project focuses on evaluating the precision of deep neural networks in calculating these centroids. Due to the scarcity of annotated data, a feasibility study is conducted using synthetic data.

The investigation into this problem was pursued as a side project due to the promising prospects of deep learning in this context.

3.2 Data generation

To simplify the task for the network, it is assumed that the ground truth centroid is located within a 1x1 pixel square in the middle of the image. This assumption facilitates training by providing a defined target location. During the inference stage, a simple algorithm is already capable of achieving a precision of 0.05. This algorithm can be utilized to correct any shifts greater than 1 pixel from the center.

To simulate a sample, the following steps are performed:

1. Randomly select a center coordinate with subpixel precision.
2. Generate a 2D Gaussian heatmap centered on this point, with a specified spatial standard deviation.
3. Multiply the heatmap by a given amplitude.
4. Apply Gaussian noise to each pixel.
5. Introduce saturation.
6. Quantize the image.

The standard deviation and amplitude are drawn from a uniform distribution within a specific range.

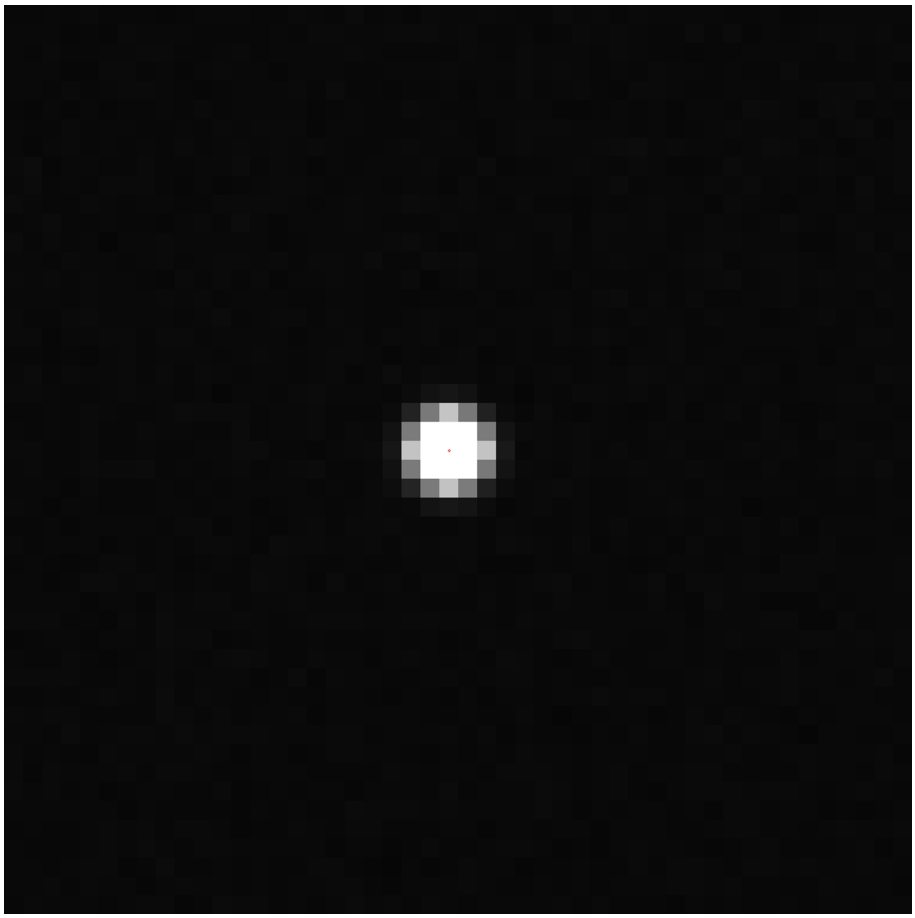


Figure 30: Simulated data sample

To mitigate overfitting and enhance generalization capabilities, the points were sampled from a set of equally spaced points with a step size of 1/10000. This approach significantly increases the number of potential points, resulting in a large number of possibilities. However, during evaluation and training, only 10000 points are randomly sampled each time, rather than iterating over all possible points.

Due to this random sampling, each evaluation may yield slightly different scores. However, after conducting multiple tests, it was observed that the variation introduced by this sampling approach is several orders of magnitude smaller than the actual error of the solution. As a result, the variation can be considered irrelevant noise.

3.3 Previous works

The previous approach involved computing the squared sum of the moment with weighted values: $\sum_{ij}(H_{ij} - \text{offset})^2 \begin{pmatrix} i \\ j \end{pmatrix}$. This method achieved a precision of 0.05

on synthetic data.

Most of the existing literature on key point detection focuses on semantic key points. However, heatmap-based networks do not provide sufficient precision.

Attaining subpixel precision is challenging with heatmap-based methods because precision is closely linked to resolution. Increasing input size is limited due to hardware constraints. Achieving the desired precision would require direct end-to-end optimization. While the deep learning literature has largely overlooked certain architectures due to their performance with semantic key points, in our specific context, it might be possible to utilize them since the problem is not purely semantic and does not require highly powerful features for detection.

3.4 Experiments & results

Existing literature lacks comprehensive research on achieving subpixel precision in key point detection. Consequently, I conducted extensive testing on various architectures to address this gap. The key components of these architectures consist of a feature extractor and a regressor head.

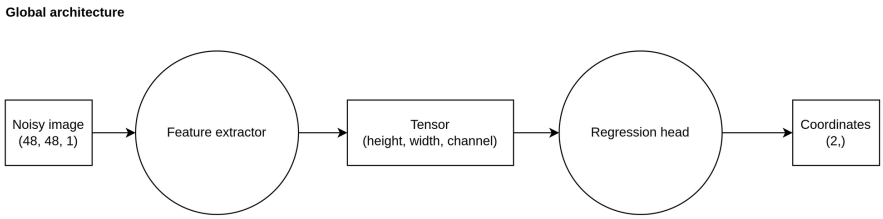


Figure 31: Architecture design for coordinates regressors

Regression head As mentioned previously, achieving subpixel precision with an error of up to 0.05 pixels requires directly predicting exact coordinates and optimizing them using a loss function. Several approaches were explored:

- Softargmax: The softmax function is applied to the 2D heatmap, followed by the calculation of the moment using the resulting heatmap. The coordinates are computed as $\begin{pmatrix} x \\ y \end{pmatrix} = \sum_{ij} H_{ij} \begin{pmatrix} i \\ j \end{pmatrix}$, where H_{ij} represents the softmaxed heatmap.
- Local soft argmax: This approach employs a two-step regression process. Initially, the argmax is used as an approximate mean estimator of the heatmap. Then, the softargmax is computed on a square region of k pixels around the argmax. This method has been used in face landmarks detection.

- Soft moment max:Similar to the currently employed method, this approach calculates the moment by computing the sum of squared values: $\begin{pmatrix} x \\ y \end{pmatrix} = \sum_{ij} H_{ij}^2 \begin{pmatrix} i \\ j \end{pmatrix}$.
- Global average pooling: This method involves using a convolutional layer to obtain 2 channels, followed by global average pooling to determine the x and y coordinates.
- Multipectron head: This method involves flattening the heatmap into a vector and passing it through a Multi-Layer Perceptron (MLP) to obtain the coordinates.

For the MLP and global average pooling solutions, the network is expected to compute normalized coordinates between 0 and 1, using a sigmoid activation function as the final layer instead of direct coordinates.

Among these approaches, the Multipectron head outperformed the others significantly. This could be attributed to the fact that while the other methods first predict a heatmap and then reduce it to a vector in a single operation, the Multipectron approach directly processes the vector without intermediate steps, allowing for better optimization.

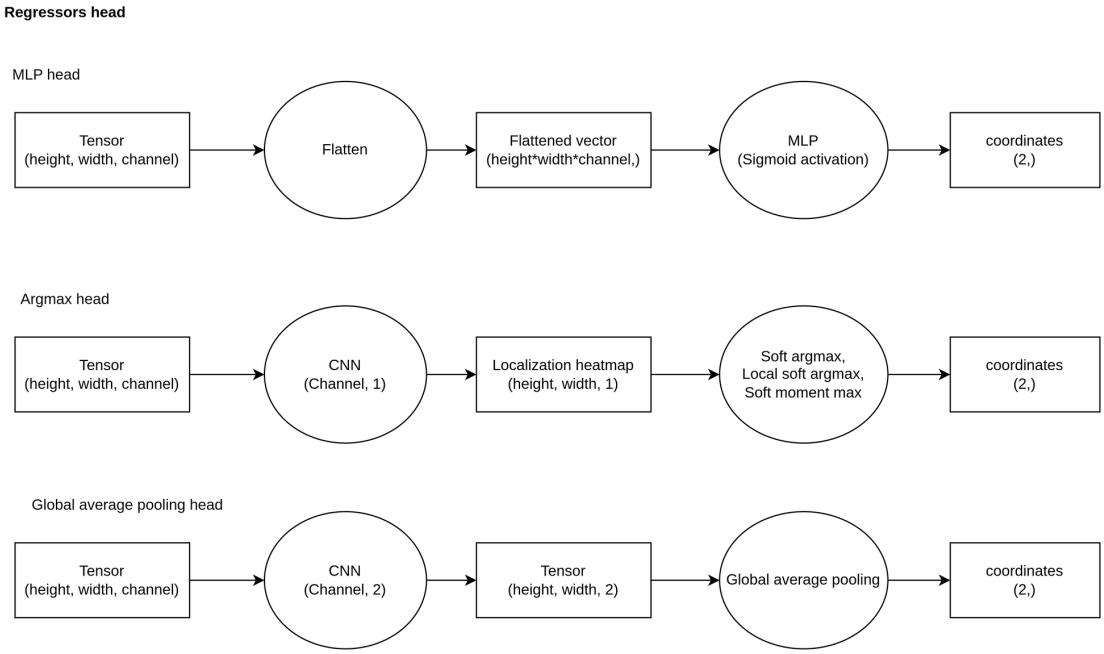


Figure 32: Regressor heads

Convolutive features extractor In the context of this problem, we are dealing with regression on images. Convolutional neural networks (CNNs) are widely recognized for their effectiveness in image-related tasks due to their inherent bias. However, since our goal is regression, it is necessary to progressively reduce the size of the image throughout the network. There are three options to achieve this:

- Utilizing stride in convolutions: By applying stride in convolutional layers, we can downsample the image and reduce its size while retaining relevant information for the regression task.
- Employing Max pooling layers: Max pooling is a common technique that downsamples the input image by selecting the maximum value within each pooling region. This process helps reduce the spatial dimensions of the image.
- Using Average pooling layers: Similar to Max pooling, Average pooling downsamples the input image but by taking the average value within each pooling region instead.

These options allow for the gradual reduction in image size while preserving the relevant features required for the regression task.

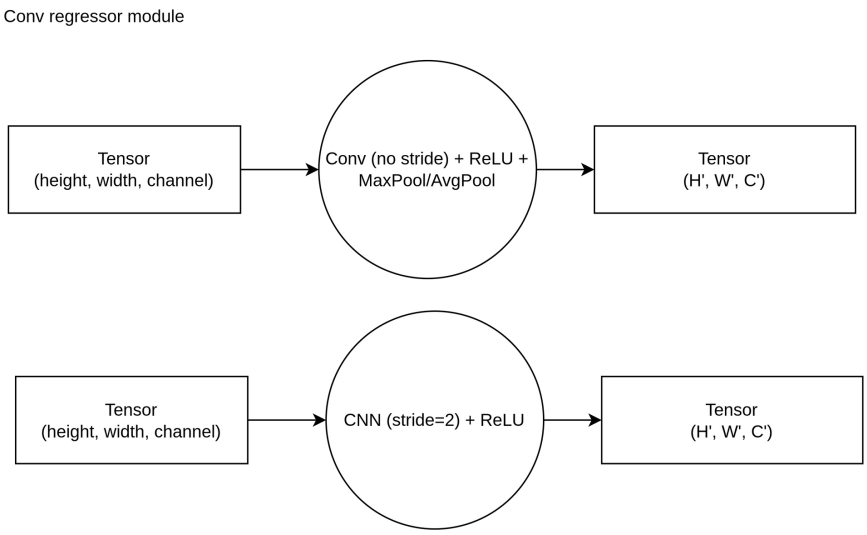


Figure 33: Convolutive modules

Regardless of the reduction method used, the experimental results demonstrated remarkably similar outcomes.

Denoising heatmap and regressor A more advanced approach involves training the initial part of the network to denoise the Gaussian heatmap and subsequently

regress the positions of the centroids. Instead of training the networks separately, it has been established that training them together through intermediate supervision yields better results. Two methods can be employed to implement this:

1. Optimizing the latent space representation: The first method involves directly optimizing the latent space representation to resemble the denoised image.

2. Feature extraction and separate heads: The second method incorporates a feature extractor that maintains the dimensions of the latent space representation. This is followed by one head that utilizes the latent space to compute the denoised image and another head that performs coordinate regression.

The second method is slightly more complex to implement but is generally considered superior since the optimal features for regression may not necessarily align with the denoised image. This allows the network to discover its own optimal representation.

However, in practice, these networks did not yield significantly better results. While they did show slight improvements, this can likely be attributed to their larger size and additional layers. This suggests that the networks are already proficient at denoising the image without requiring intermediate supervision.

Another idea that was considered was employing intermediate supervision with a higher resolution image. However, it is unlikely that this would have resulted in significantly better outcomes.

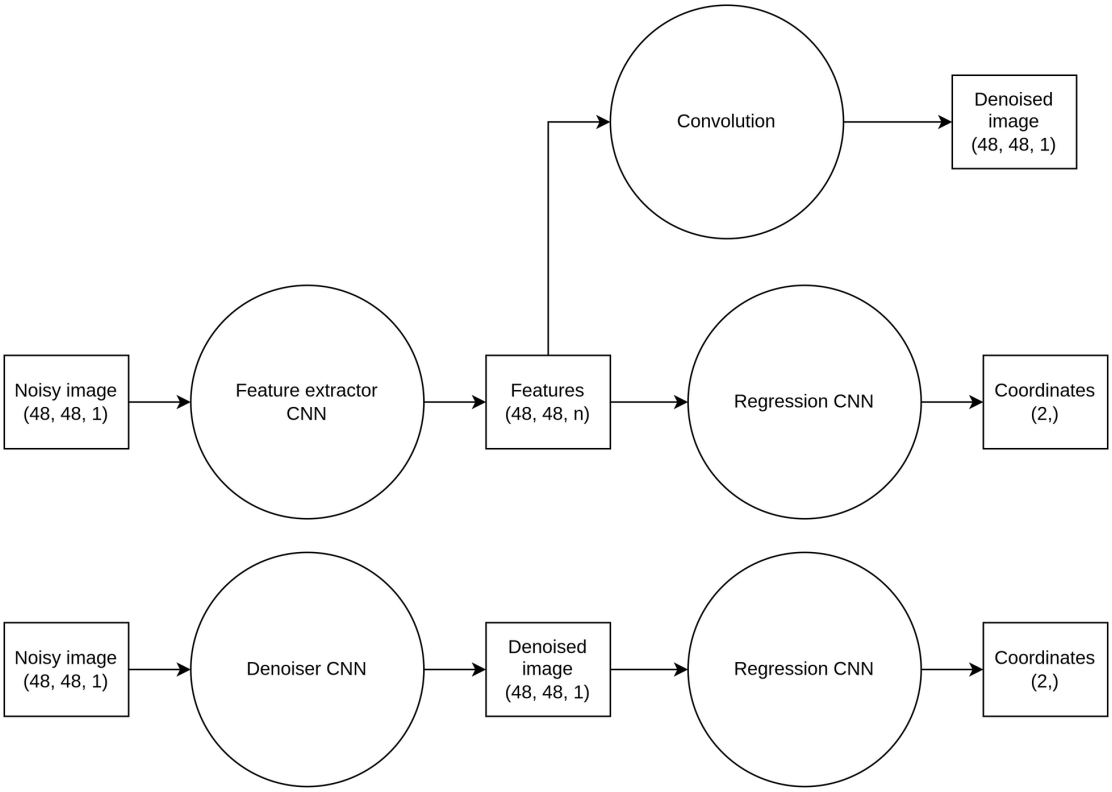


Figure 34: Intermediate supervision

Mobilenet modules In order to enhance the expressiveness and power of the networks while maintaining a lightweight architecture, MobileNetV2 modules were implemented. As part of this implementation, the Batch Normalization layers were removed due to their destabilizing effect on the training process. It is hypothesized that Batch Normalization may compromise localization features in contrast to semantic features.

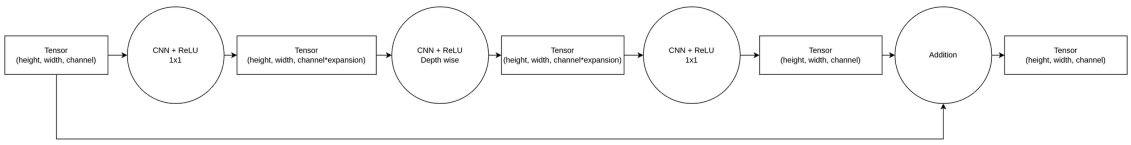


Figure 35: MobileNetV2 modules

Multi-flipping inference Similar to the previous project, for evaluation purposes, the average prediction of all four flips is computed for each sample. This approach has resulted in a significant improvement in accuracy.

Results Below is a table summarizing all the conducted experiments. For both training and evaluation, a fixed amplitude and standard deviation were used to achieve faster convergence.

Features extractor	Regressor head	Interm. sup.	Multi-flipping	MSE score
None	Softargmax	No	No	0.38
None	Local soft argmax	No	No	0.27
None	Soft moment argmax	No	No	0.3
None	MLP	No	No	0.00890
Conv Maxpooling	MLP	No	No	0.00771
Conv Avgpooling	MLP	No	No	0.00796
Conv w/ stride	MLP	No	No	0.00761
Conv w/ stride	Softargmax	No	No	Not conv.
Conv w/ stride	Global avg pooling	No	No	0.39
Deep conv w/ stride	MLP	No	No	0.00471
Conv	MLP	1st meth.	No	0.00511
Conv	MLP	2nd meth.	No	0.00478
MobileNet v2	MLP	No	No	0.00464
MobileNet v2	MLP	No	Yes	**0.00334**

Table 1: Experiments metrics

The experiments demonstrate the superiority of MobileNetV2 modules for feature extraction, resulting in more accurate predictions of centroids. The incorporation of multi-flipping during inference proves to be a significant factor in improving the model's accuracy. Moreover, the experiments suggest that the model's performance does not heavily rely on hyperparameters, and a more exhaustive hyperparameter search might be relevant to further fine-tune the model.

After conducting experiments on a simpler case with fixed amplitude and standard deviation, I proceeded to train the best architecture using random values and achieved an error of 0.0053 pixels. Subsequently, I explored a few other architectures, but the overall performance order remained consistent. Larger architectures tended to yield better results, albeit requiring longer training times. The study demonstrated that the selected architecture was highly effective, even with random parameter values.

3.5 Future improvements

Train and evaluate on real data Directly evaluating the model on real data is expected to yield poor performance due to the distribution shift between the synthetic data it was trained on and the real-world scenarios it will encounter. Incorporating real data into the training process would be highly beneficial for enhancing

the model's ability to generalize. For example, quick movements might cause the blobs to appear more oval than round, which can be better captured by real data in the training set.

Weakly supervised and partial annotations Although the available real data may be limited, we can fully exploit the invariance to rotations to augment the dataset and create additional training samples.

Furthermore, the loss function can be tailored to address specific requirements. For example, if we only have ground truth coordinates along the x-axis, we can compute the loss solely for that axis.

Exploring the potential of weakly supervised learning methods, which involve incorporating noisy or incomplete annotations, could be highly advantageous, especially when manual annotation of data is not feasible. By utilizing weak supervision techniques, the model can leverage a large amount of available data, even if the labels are noisy or incomplete. This approach allows for a more scalable and efficient training process, enabling the model to learn and generalize effectively despite the challenges posed by imperfect annotations.

Uncertainty estimation Incorporating uncertainty estimation techniques into the model's architecture would enable it to provide confidence scores alongside its predictions. These scores can be utilized in various ways, such as down-weighting or discarding predictions with lower confidence levels. By incorporating uncertainty estimation, the model can make more informed decisions and enhance its overall performance and reliability.

Better architectures Investigating alternative architectures, such as U-net, Stacked Hourglass net, and incorporating attention layers, could potentially enhance the model's ability to extract features and improve its representation capabilities. These architectures have demonstrated promising results in various computer vision tasks and might prove valuable in enhancing the accuracy and efficiency of the centroid prediction network. Given the limited duration of this study, the exploration was focused on basic approaches, but further investigation into these advanced architectures could be fruitful for future improvements.

4 Conclusion

Throughout the course of this internship at Stryker, I have been actively engaged in two compelling projects aimed at enhancing the precision tracking of surgical instruments. The first project centered around instrument calibration and sought to accelerate the existing process. Leveraging the power of deep learning, I designed a novel approach that involved detecting key points on RGB images and then triangulating the tip's position using the tracker's position and the detected key points. This innovative method showcased remarkable results, achieving an impressive calibration error of only 3 mm with just a short 10-second video.

In the second project, my efforts were directed towards advancing the subpixel precision of instrument tracking using near-infrared (NIR) images and innovative deep learning approaches. Through the creation and utilization of simulated data, I successfully developed a robust deep learning model that achieved a remarkable error of 0.005 pixels, surpassing the existing method's error of 0.05 pixels.

Both projects have demonstrated the remarkable capabilities of deep learning techniques in the context of surgical instrument tracking. The outcomes of these projects present significant opportunities for enhancing surgical procedures, ultimately leading to improved patient outcomes and more efficient surgical workflows.