Tracking Biomedical Equipment in Virtual Reality

A Project report

Submitted by

Ayyalasomayajula Varun Kumar

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

May 2017

THESIS CERTIFICATE

This is to certify that the thesis titled Tracking Biomedical Equipment in Virtual Reality, submitted by **A.Varun Kumar**, to the Indian Institute of Technology, Madras, for the award of the degree of Bachelor of Technology, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Dr.M Manivannan

Project Guide

Professor Dept. of Applied Mechanics

IIT-Madras, 600036

Place: Chennai

Date:

Prof. Dr.Kaushik Mitra

Project Co-guide

Asst. Professor Dept. of Electrical Engineering

IIT-Madras,600036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I would like to thank my project guide, **Dr. M Manivannan**, for all his supervision and help during the project work. I would also like to thank **Dr. Kaushik Mitra** for being the co-guide of our project. I would also like to extend my sincere thanks to the people at Haptics Lab – Joseph Isaac, Ravalli Gourishetty, Chandrasekkhar Burlee and Prabhu Vasulingam for helping me out at different stages of my project work. Lastly, I thank my friend and batch mate Ashfakh Rithu, for being my project partner and a good friend.

CONTENTS

1. Abstract..... 5

2. Introduction..... 6

2.1 Motivation.....6

2.2 Problem Statement.....6

3. Initial Work.....7

3.1 PNP algorithm for Pose estimation.....11

4. Monocular Pose Estimator.....14

4.1 System Requirements.....14

4.1.1 Hardware.....14

4.2 Algorithm.....15

4.2.1 Overview.....15

4.2.2 LED Detection.....16

4.2.3 Correspondence Search.....18

4.2.4 Prediction.....19

4.2.5 Pose Optimization.....20

5. Evaluation.....21

6. Conclusion.....22

6.1. Limitations.....22

6.2. Future Work.....22

7. References.....23

1 ABSTRACT

The project is aimed at using the RIFT technology of tracking a head mounted display to track other objects in Virtual reality. We use infrared LEDs as markers on target object for an accurate and robust pose estimation system. They are mounted on a target object and are observed by a camera that is equipped with an infrared-pass filter. The correspondences between LEDs and image detections are first determined using a combinatorial approach and then tracked using a constant-velocity model. The pose of the target object is estimated with a Perspective-3-Point algorithm and optimized by minimizing the re-projection error. Since the system works in the infrared spectrum, it is robust to cluttered environments and illumination changes.

2 INTRODUCTION

2.1 MOTIVATION

The Motivation behind this project was to create a proper tracking system for the biomedical equipment as a part of ACLS project in the Haptics lab. We were part of the project which made a 3-DOF tracking system using stereo cameras. But the system was not robust and accurate. So we decided to come up with a robust system based on the tracking technology using Infra Red LEDs as they are easier to detect and provide more accurate tracking.

2.2 PROBLEM STATEMENT

To make an object tracking system with sub-millimetre accuracy so as to track biomedical equipments as objects. Since sub-millimetre accuracy is required we have chosen to use IR LEDs and track the object as this method is being used by many state of the art tracking technologies

2.3 SCOPE OF THE PROJECT

The tracking system we developed is of sub-millimetre precision. It can be used in many on going and future projects at Haptics Lab. This thesis also includes the detailed procedure of working of Oculus Rift DK2, so it will be useful in any future projects which include hacking of Oculus Rift DK2.

3 INITIAL WORK

Our Initial work was based on Oculus Rift technology. The Head Mounted Display (HMD) of the Oculus is tracked using IR LEDs and an Inertial Measurement Unit(IMU). Reverse engineering of Oculus Rift was done to find out how the HMD was tracked. There are 41 IR LEDs and an IMU mounted into the Oculus HMD at predefined positions.

The Synchronised camera communicates via I2C Serial communication. So by giving a specific set of commands to the camera, information about the position of HMD can be extracted. The camera used in Oculus Rift DK2 is Aptina MT9V034, and the commands can be found in the datasheet.

The LEDs on the HMD are controlled by the camera. The camera is the master device and HMD, the slave device. There is a HID feature report (0x0C) that turns on the LEDs. It has a built-in timeout of 10 seconds, meaning the

report must be reset at regular intervals by the driver software or the LEDs turn off again. After sending this feature report, the LEDs will turn on for 10 seconds, and they will show up in the tracking camera image very brightly and flicker at particular blinking frequencies. The Blinking pattern of LEDs can be controlled and this pattern also gives each LED a 10-bit identity.

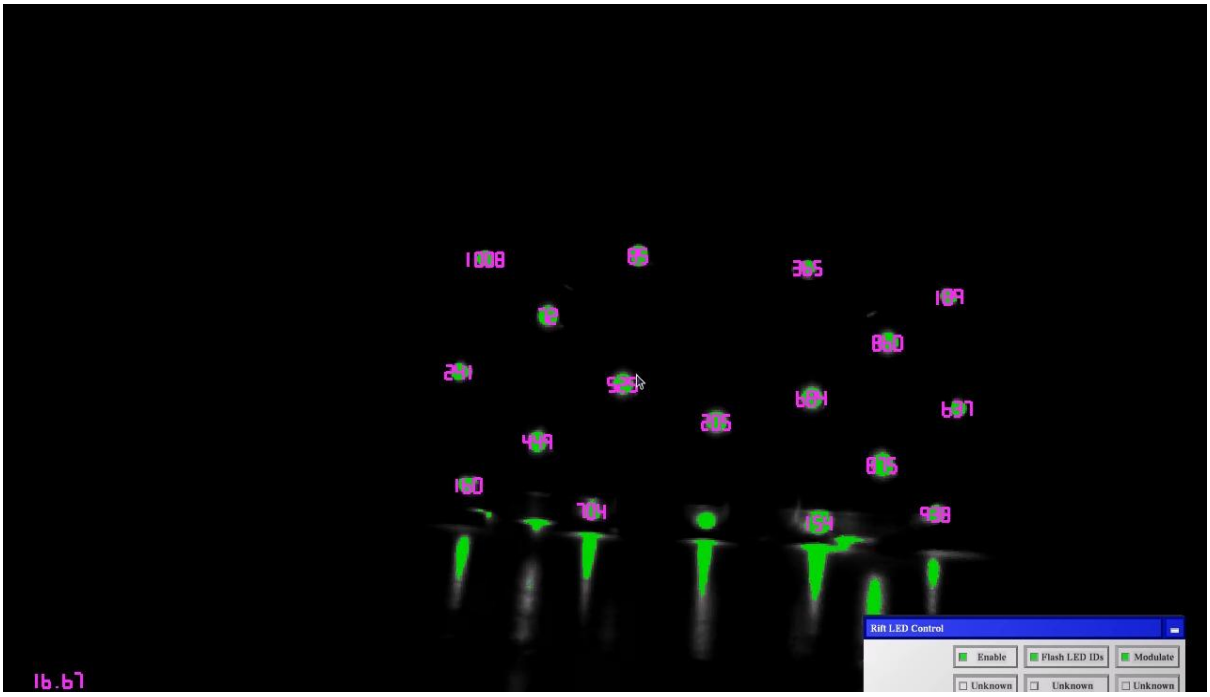


Figure 2: LED tracking and identification algorithm with frame-drop correction. Still frame from "Identifying LEDs Based on Blinking Patterns."

Here is the full list of 10-bit IDs, ordered by 3D marker position index in report 0x0f, from 0 to 39 (see Figure 3 for a picture of the corresponding marker positions in 3D space):

```
  2,  513,  385,  898,  320,  835,  195,  291,  800,  675,
 97,  610,  482,  993,  144,  592,  648,  170,   27,  792,
410,  345,  730,   56,  827,  697,  378,  251, 1016,  196,
165,   21,  534,  407,  916,  853,  727,  308,  182,  119
```

The numbers might look random, but there's a reason why they're spread out over the entire [0, 1024) interval. If we know that the only 10-bit IDs that we expect to find in a video stream are the 40 listed above, we can use the redundancy of assigning 10 bits to encode 40 values to automatically correct the kinds of bit errors. This is based on Information theory concept called Hamming distance. The Hamming distance between two 10-bit binary numbers is the number of bits one has to flip to turn the first number into the second. If the numbers are identical, their Hamming distance is 0; if they differ in a single bit, their distance is 1, if they are bit-wise negations of each other, their Hamming distance is 10. There's a related concept called minimal Hamming distance, which is the minimum of Hamming distances between all pairs of elements of a list. In the case of our 40 elements, their minimal Hamming distance happens to be 3. It means the minimum number of bit flips it takes to turn one valid ID into another valid ID is 3. So if we assume that bit errors are rare enough that it's improbable that more than one occurs in any sequence of ten video frames (and that seems to be the case), then we can not only detect, but correct those errors on-the-fly.

We compare any extracted number to the list of 40, and find the list entry that has the smallest Hamming distance. If that distance is 0 or 1, we know that the number we extracted should be set to the list entry we found. If there happen to be 2 or more bit errors in a sequence of 10 frames, we're out of luck, but in practice, it seems to be working.

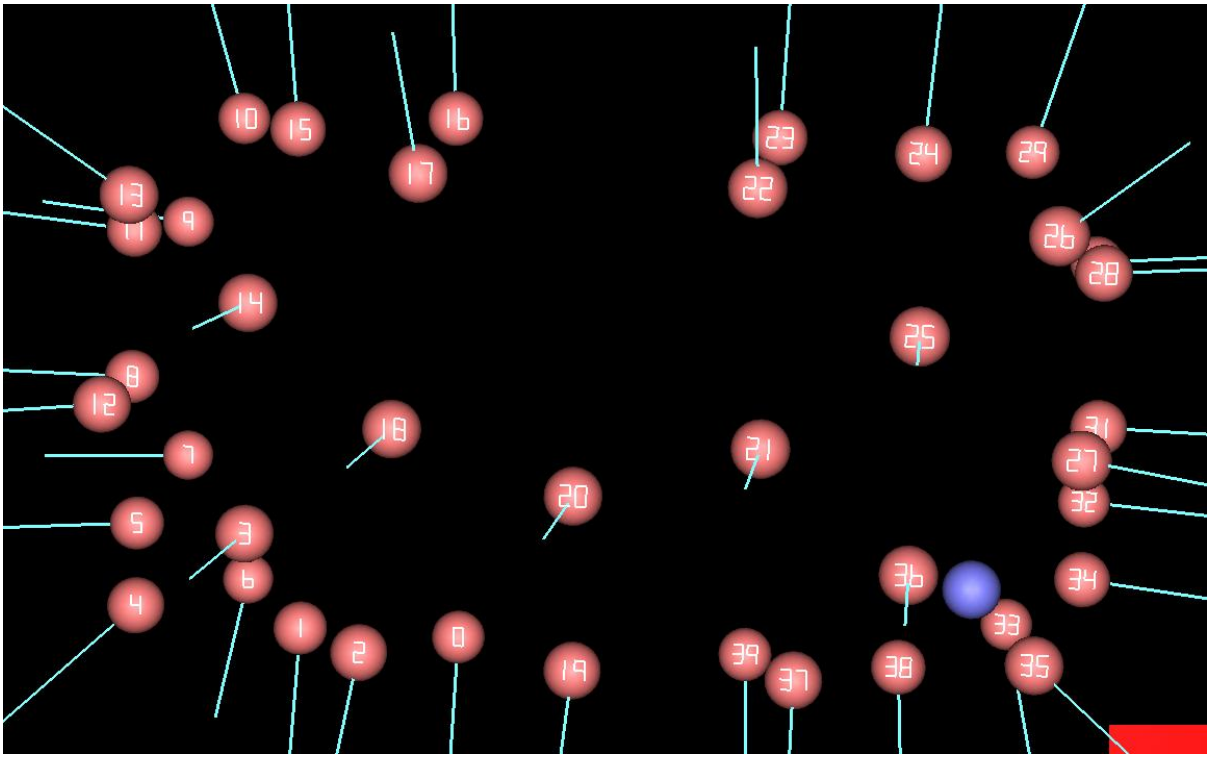


Figure 3: Oculus Rift DK2's 3D LED positions, labeled by marker index in the sequence of 0x0f HID feature reports.

And instead of setting the LED blob ID to the correct 10-bit number we just set it to the index of the associated 3D marker (see Figure 4) That way we can feed it directly into the pose estimation algorithm.

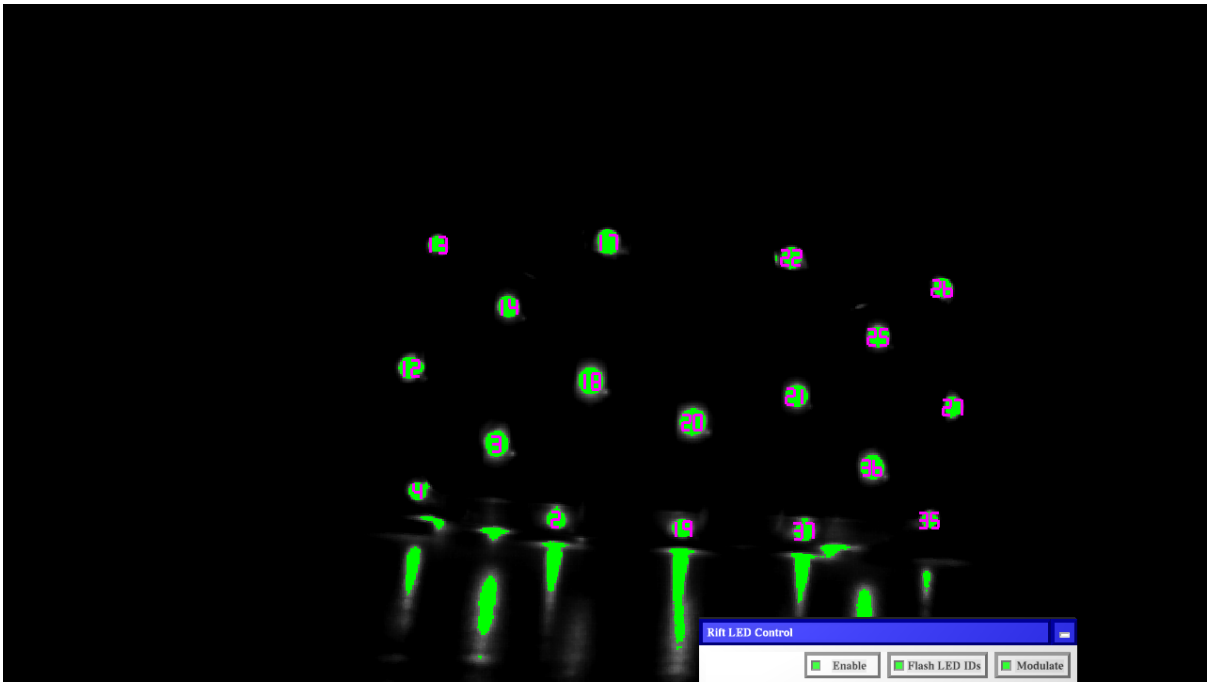


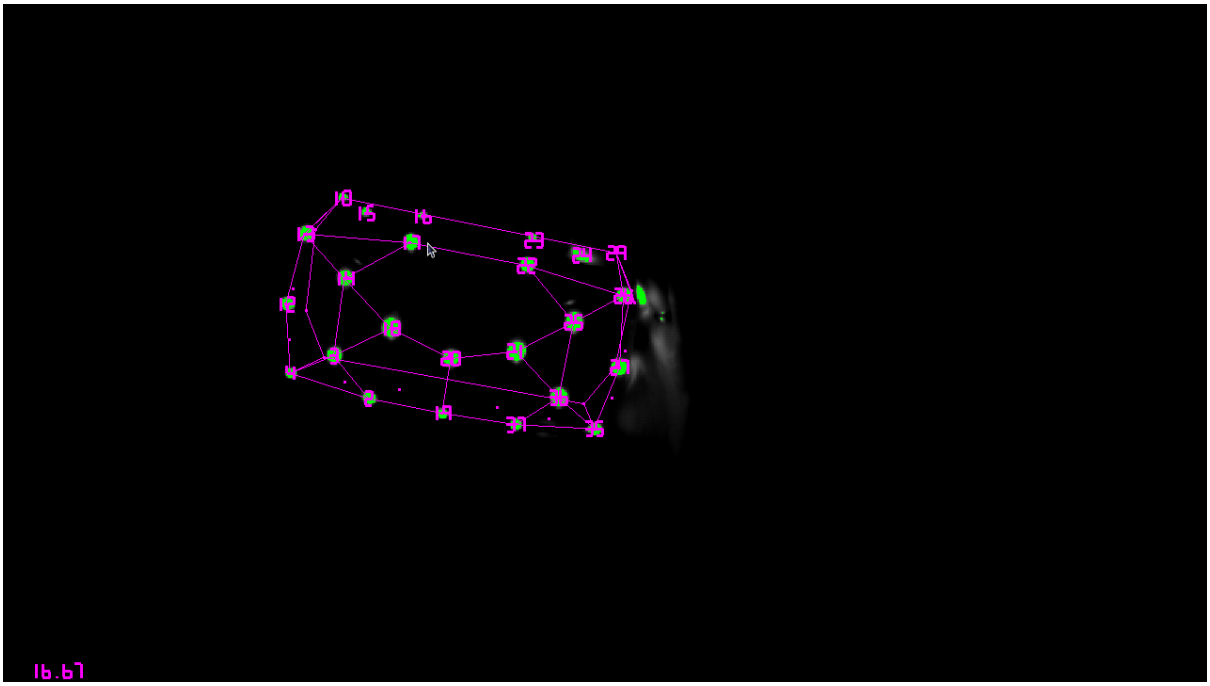
Figure 4: LED blobs extracted from the video stream, labelled with the indices of their associated 3D marker positions. Compare to Figure 3.

3.1. PNP ALGORITHM FOR POSE ESTIMATION

3D pose estimation, or the problem of reconstructing the 3D position and orientation of a known object relative to a single 2D camera, also known as the Perspective-n-Point problem, is a well-researched topic in computer vision. In the case of the Oculus Rift DK2, it is the foundation of positional head tracking. An inertial measurement unit (IMU) by itself cannot track an object's absolute position over time, because positional drift builds up rapidly and cannot be controlled without an external 3D reference frame. 3D pose estimation via an external camera provides exactly such a reference frame.

3D pose estimation is a multi-dimensional non-linear optimization problem. Given a known model, i.e., a collection of 3D points such as the DK2's tracking LEDs, a camera with known intrinsic parameters, and a set of 2D

points in the camera's image, such as the set of extracted LED blobs, one can try to reconstruct the unknown position (tx, ty, tz) and orientation (yaw, pitch, roll) of the model with respect to the camera (in reality, one would never use yaw, pitch, and roll angles to do this, but that's a technical detail). In theory, the approach is simple. Given a candidate set of unknown parameters (tx, ty, tz, yaw, pitch, roll), one takes the set of 3D model points, transforms them by the rigid body transformation defined by the six parameters, projects them into image space using the camera's intrinsic parameters, and then calculates the sum of their squared distances from the true observed image points (this is called re-projection error). This process defines an error function F(tx, ty, tz, yaw, pitch, roll), and the problem is reduced to finding the set of parameters that globally minimizes the value of the error function.



Still frame from pose estimation video, showing a 3D model of the DK2's headset (the purple wireframe) projected onto a raw 2D video frame from the tracking camera based on reconstructed position and orientation.

Given a predicted image point, which one of the observed image points should it be compared to? Without knowing anything else, one would have to test all possible associations of observed and predicted image points, and pick the association which yields the smallest error after optimization.

Unfortunately, there are a lot of possible associations; in general, if there are N predicted image points and M<=N observed image points, then there are N!/(N-M)! possible associations. To pick an example, for N=40 (number of LEDs on DK2) and M=20, there are 335,367,096,786,357,081,410,764,800,000 potential associations to test, and that's a large number even for a computer. There are many heuristic and/or iterative methods to establish associations automatically, but they tend to be rather slow and fragile. The best approach, is to somehow make it possible to identify a-priori which observed image point belongs to which 3D model point, and the DK2's flashing 10-bit patterns do exactly that.

The lack of a proper Linux SDK made our work very difficult. Even though we were able to dump the estimated pose of the oculus into the terminal, we were unable to expand it so that this can be implemented with another object. More over the synchronized camera had a couple bugs which made it very difficult to use it with normal camera packages and calibration procedures. The hardware used by Oculus was difficult to replicate on a mall biomedical object for it to be tracked. So we decided to come up with another tracking system which mimicked the Rift tracking system and was considerably less complex.

4 MONOCULAR POSE ESTIMATOR

4.1 SYSTEM REQUIREMENTS

4.1.1 HARDWARE

Our system consists of infrared LEDs at known positions on the target object and an external camera with an infrared pass filter. With at least four LEDs on the target object and the corresponding detections in the camera image, we can compute the 6-DOF pose of the target object with respect to the camera. To increase robustness, the system can also handle more than four LEDs on the target object. The placement of the LEDs on the target object is arbitrary, but must be non-symmetric. In addition, the LEDs should not lie in a plane to reduce ambiguities of the pose estimation. To increase precision, they should span a large volume. Robustness can be increased if the LEDs are visible from as many view points as possible.

As mentioned above, our system requires prior knowledge of the LED configuration, i.e. the positions of the LEDs in the reference frame of the target object. Since infrared LEDs are detectable by a motion capture system, we can use it to determine the positions of the LEDs with sub-millimetre accuracy. We can track the target object in the motion capture system and read out the positions of the single LEDs, which can be transformed into the target-object coordinate frame. Furthermore, we need to know the intrinsic camera parameters, which we obtain using the camera calibration tools of ROS.

The entire system is implemented in ROS. ROS is a flexible framework for writing robot software. It consists of a collection of tools, libraries and conventions that are very commonly used among developers.

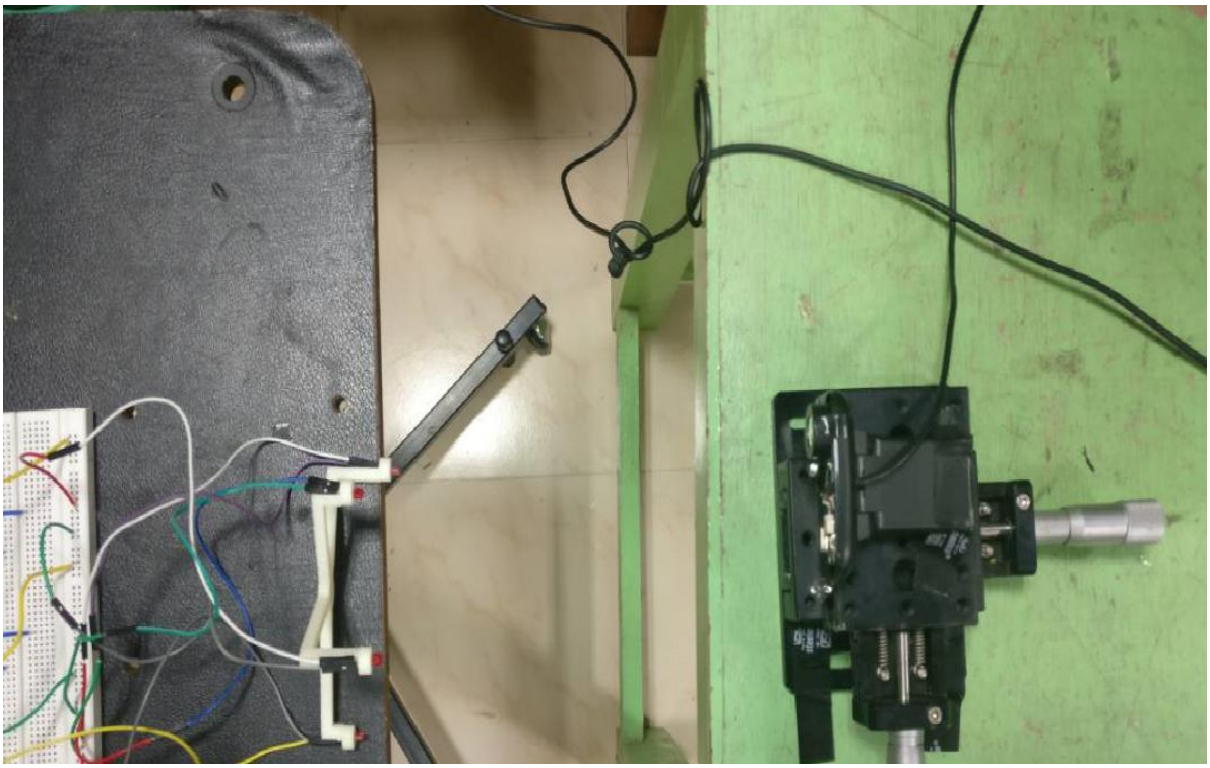


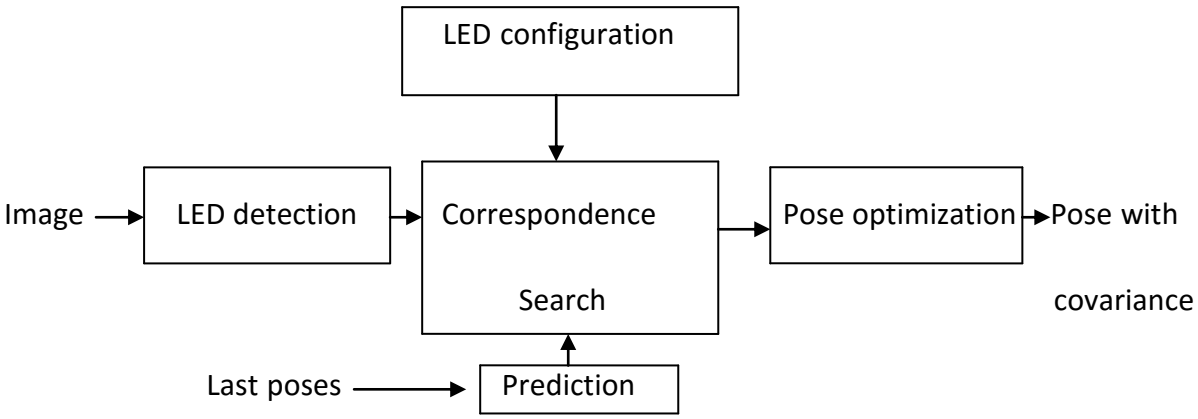
Figure: Experimental Setup

4.2 ALGORITHM

4.2.1 OVERVIEW

The flowchart of our algorithm is presented in Fig. 3. The current camera image, the LED configuration, and previously estimated poses serve as inputs to our algorithm. In a first step, we detect the LEDs in the image. Then, we determine the correspondences using prediction or, if that fails, using combinatorial brute-force approach. Finally, the pose is optimized such that the re-projection error of all detected LEDs is minimized. This optimization also

returns the covariance of the pose estimate, which is crucial information in further processing.



4.2.2 LED DETECTION

Since we are using infrared LEDs whose wavelength matches the infrared-pass filter in the camera, they appear very bright in the image compared to their environment. Thus, a thresholding function is sufficient to detect the LEDs,

$$I'(u, v) = \begin{cases} I(u, v), & \text{if } I(u, v) > \text{threshold}, \\ 0, & \text{otherwise.} \end{cases}$$

This threshold parameter depends on the shutter speed of the camera settings. However, we found that a large range of parameters works well (80–180). We then apply Gaussian smoothing and group neighbouring pixels to blobs. To estimate the centre of these blobs with sub-pixel accuracy, we weigh the pixels with their intensity. The centre is then calculated using first image moments that are defined as

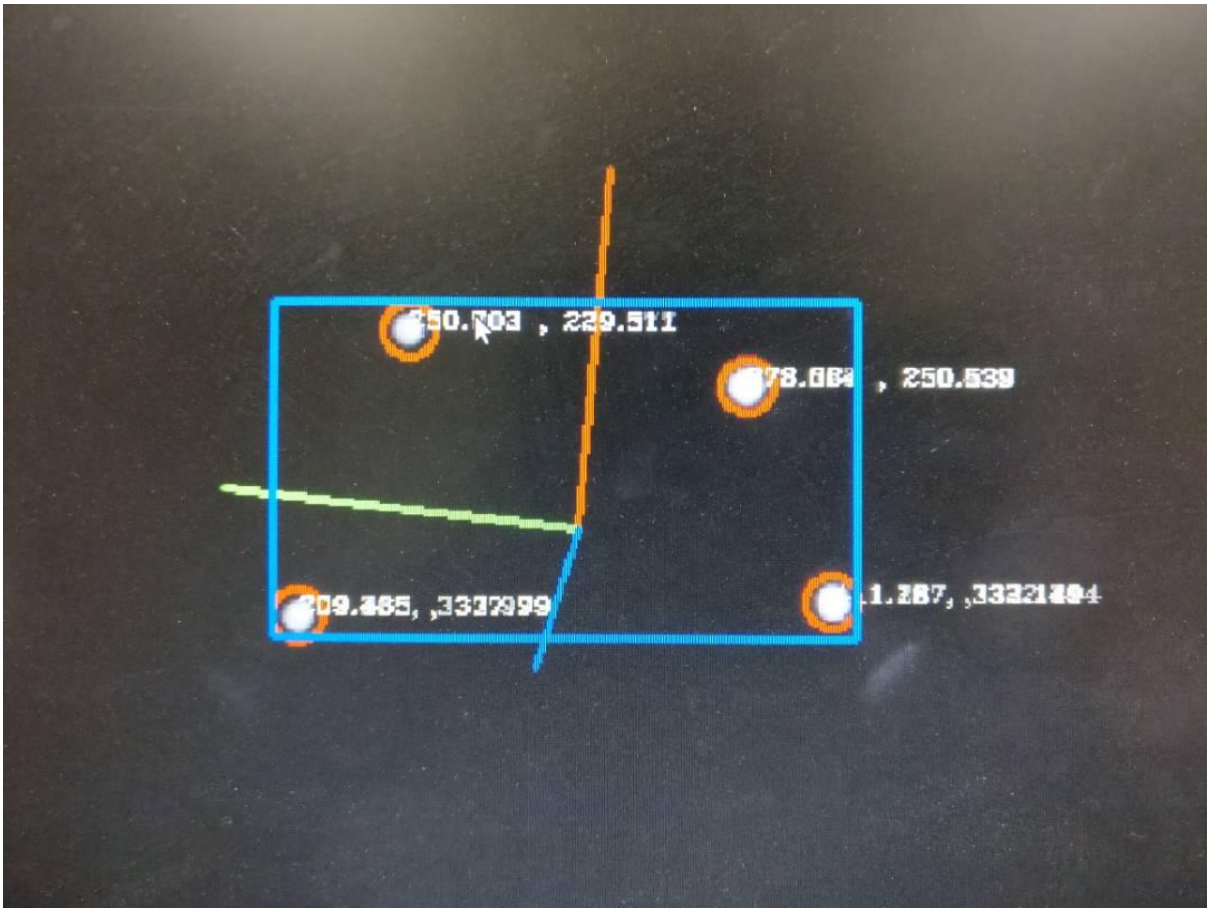
$$M_{pq} = \sum_u u^p \sum_v v^q I'(u, v).$$

The weighted centre, i.e. the (distorted) LED detection in the image, is then

$$\hat{u} = M_{10}/M_{00},$$

$$v\hat{} = M_{01}/M_{00}$$

In all calculations to come, we assume the standard pinhole camera model. Thus, we have to correct the detections for radial and tangential distortion. We do this using the OpenCV library.



4.2.3 CORRESPONDENCE SEARCH

The different LEDs detected in the image is not identified with their corresponding positions in the marker file, so we need to run a correspondence search in order to identify which LED is which one. Since we've 4 LEDs as markers and only 3 LEDs are required for P3P pose estimation, we can find four possible combinations of any three LEDs and every permutation of LEDs in the marker holder. Now we use the fourth LED in each case and compute its pose estimated by the position matrix and re-project it into the input image_raw. If the closest neighbour of the re-projected LED is less than a given threshold value, then that particular pose is used. We used the re-projection threshold to be around 4 pixels. A histogram of each and every possible combination and corresponding re-projected value is stored, and then it is used to find the one which is closest to the original image. If n_D is the number of detections, n_L is the number of possible configurations, then, the number of total possible pose estimations are

$$N = 4 * \binom{n_D}{3} * \frac{n_L!}{(n_L - 3)!}$$

This grows very quickly for increase in n_L and n_D. Since we use only 4 LEDs, this shouldn't be a problem. But in the case of Oculus rift, which uses nearly 41 LEDs, it is very difficult to determine the correct pose using an exhaustive pose estimation search.

For 4 LEDs, a total of 384 pose estimation candidates are present. The search through this is done pretty quickly so as to obtain the correct

corresponding LED-Marker candidates. Once they are determined, they are kept tracked using a constant velocity tracker, so that the brute force search is not necessary all the time.

4.3.4 PREDICTION

Since the brute-force matching in the previous section can become computationally expensive, we predict the next pose using the current and the previous pose estimates. A constant-velocity model is used for prediction. The pose P is parameterized by twist coordinates ξ. We predict the next pose linearly

$$\xi_k+1 = \xi_k + \Delta T*(\xi_k - \xi_{k-1}) \ ,$$

$$\Delta T = \begin{cases} 0, & \text{if } n_p = 1, \\ (T_{k+1} - T_k)/(T_k - T_{k-1}) & \text{if } n_p \geq 2 \end{cases}$$

where T_k is the time at step k and n_p the number of previously estimated poses. Using the predicted pose, we project the LEDs into the camera image. We then match each prediction with its closest detection, if they are closer than a threshold. This condition prevents false correspondences, e.g. if an LED is not detected. We typically use 5 pixels for that threshold. We then check if the predicted correspondences are correct. To do so, we compute the four pose candidates with the P3P algorithm for every combination of three correspondences. We then compute the projection of the remaining LEDs and

check if at least 75 % of them are below the re-projection threshold. If this is true for one of the four pose candidates of more than 70 % of the combinations of correspondences, we consider them as correct. In case we could not find the correct correspondences, we reinitialize the tracking using the brute force method.

4.3.5 POSE OPTIMIZATION

To estimate the target-object pose, P^*, we use all correspondences in C and iteratively refine the re-projection error starting with a solution from the P3P algorithm as an initial estimate, that is

$$\mathrm{P}^* = \arg_P \min \sum_{(\mathrm{l},\mathrm{d})\in \mathrm{C}} \left|\left|\Pi(\mathrm{I},\mathrm{P}) - \mathrm{d}\right|\right|^2$$

Where $\pi : \mathrm{R}^3 \times \mathrm{SE}(3) \rightarrow \mathrm{R}^2$ projects an LED into the camera image. For the optimization, we parameterize the pose using the exponential map and apply a Gauss-Newton minimization scheme. The covariance of the final pose estimate, is a by-product of the Gauss-Newton scheme, since it requires the computation of the Jacobian matrix, J which can be computed in closed form. The covariance of the pose, $\sum_P$, is then obtained by

$$\sum_P = (\mathrm{J}^T \sum_{\mathrm{D}}^{-1} J)^{-1}$$

where $\sum_D \in \mathrm{R}^{2\mathrm{x}2}$ is the covariance of the LED detections, which we conservatively set to $\sum_D = I_{2\mathrm{x}2} \cdot 1\ \mathrm{pixel}^2$

5 EVALUATION

The Evaluation scheme was to find out the accuracy and precision of the tracking system subject to various variable changes. Also different parameters given as input and their corresponding values were also found out for optimum results in a trial and error basis. The optimal values for different parameters are given below and may subject to change based on External conditions (Illumination changes, cluttered environments etc.)

- Threshold parameter needs to be within the range of 80-180
- Gaussian sigma used in smoothing needs to be at or below 1.7. We wanted the range within which it will not false-detect LEDs where they are not present.
- Minimum blob area was found to have an upper limit of 37 (on a scale to 0-100)
- Maximum blob area was found to have a lower limit of 53 (on a scale of 0-1000)
- Max width height distortion was found to have a lower limit of .48 (on scale of 0-1.0)

PERFORMANCE TEST

The time taken for one pose estimation is around 20ms i.e. 50Hz. The initial estimation of pose takes around 1 second and then it does it at 50Hz using constant velocity model.

6 CONCLUSION

We present a robust, accurate tracking system for Biomedical Equipment in VR. Since it is implemented in ROS, it can be easily integrated with other projects. The system can also be implemented across a network so that the pose data can be read and manipulated by a different system furthering the integration into Virtual reality. The Algorithm is fast and reliable and is very easy to implement. There is a need for a lot of general purpose libraries and repositories for virtual reality development.

6.1. LIMITATION

The detection also depends on size of object. Larger the object more the number of IR LEDs required which increase the time to estimate the pose drastically. So keeping the number of IR LEDs used less is a real challenge while using this system.

6.2. FUTURE WORK

Future work is to implement segmentation to track multiple objects at the same time. A solution based on blinking frequency (implemented by Oculus) is proposed and we hope to further the research into the same.

The implementation will also require us to integrate our tracking system with the ACLS project that is happening at the Haptics lab. We hope to implement this system to some of the generally used Biomedical equipment and test their performances.

7 REFERENCES

[1] Matthias Faessler, Elias Mueggler, Karl Schwabe and Davide Scaramuzza “Monocular Pose estimation System”

[2] A. Breitenmoser, L. Kneip, and R. Siegwart, “A Monocular Visionbased System for 6D Relative Robot Localization”

[3] L. Kneip, D. Scaramuzza, and R. Siegwart, “A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Compuation of Absolute Camera Position and Orientation”

[4] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,”

[5] T. Pintaric and H. Kaufmann, “A Rigid-Body Target Design Methodology for Optical Pose-Tracking Systems,”

[6] R. Szeliski, “Computer Vision: Algorithms and Applications”