המחלקה להנדסת תוכנה

שם הפרויקט: מערכת לזיהוי מנח כף יד ואצבעות בזמן אמת

Project Name: Handy

ספר הפרויקט - Final Report

שם הסטודנט: ניר בן דור

שם הסטודנט: דניאל קורניס

תאריך ההגשה: 08/06/21

שם המנחה: אהוד דיין

חתימת המנחה:

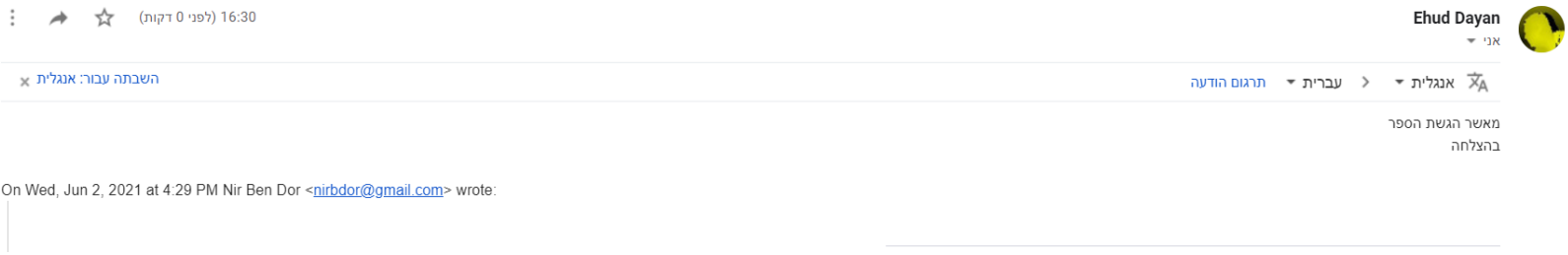


Table Of Contents:

4. Acknowledges

We would like to thank Mr. Ehud Dayan, our advisor, for his expert advice and encouragement throughout the project.

5. Index

6. Summary in Hebrew

לממשק בין אדם-מחשב יש השפעה רבה על הפרודוקטיביות וקלות השימוש בכלים ובמחשבים של ימינו. בשנים האחרונות נעשו מספר ניסיונות לפתח טכנולוגיות חדשות גם בתחומים הדורשים דיוק רב במיוחד במהלך , כמו במציאות מדומה, או שימוש במכונות כבדות. ציוד לכך כיום הוא לעתים מגושם ומלווה בחומרה נוספת שהופכת אותו ליקר. מטרת הפרויקט היא ליצור פתרון עבודה לאינטראקציה בין אנשים למחשבים. פרויקטנו נועד לספק אלטרנטיבה אינטואיטיבית וזולה יחסית המאפשרת תנועה מדויקת בזמן אמת. הוא רותם ארכיטקטורה של למידת מכונה כדי לחלץ תנוחות ידיים ומפרקים מקלט חי של תמונות RGB ואז ממפה אותן לפלט תיאום ג'ויסטיק. המערכת מאפשרת למשתמש להכניס קלט למערכת הדומה לג'ויסטיק על ידי הטייה של הידיים מול מצלמה.

שתי חלופות מערכת נשקלו:

1. שימוש בתמונות גולמיות לאימון מודל רגרסיה באמצעות תמונת RGBD. הבעיות ? של שיטה זו הן הרגישות לשינויים בתאורה, בהגדרות, ושינויים נוספים במערך.

2. שימוש במודל גרפי hand-graph-cnn לחילוץ מיקומי היד והמפרקים מתמונת RGB. שיטה זו כביכול מדויקת, אך אינה מיועדת לזיהויי אינטראקציות של אובייקטים וידיים, דבר שיקשה רבות על אימון המודל השני שלנו, כמו כן היא אינה מייצרת תוצאות בזמן אמת.

לכן עבור המוצר הסופי שלנו החלטנו להשתמש במודל הפועל על המעבד ומספק תוצר למגוון רחב יותר של מכשירים.
המוצר מורכב משתי רשתות - הראשונה היא רשת נקודות ציון ידיים המוציאה את המיקומים התלת מימדיים של מפרקי הידיים מזרם של תמונות RGB בעזרת מצלמה. הרשת השנייה היא רשת וקטור-קואורדינטות המתרגמת את המיקום התלת-מימדי למיקום הדו מימדי של הג'ויסטיק הרצוי באמצעות רשת למידה עמוקה.

ביצועי המערכת טובים מספיק כדי לפעול בזמן אמת, והיא מצליחה לתפקד אפילו בתנאי סביבה שונים - כגון תאורה משתנה, מיקום פיזי, רקע משתמשים שונים שמפעילים אותה.

הנתונים לשלב האימון והבדיקה נאספו באמצעות ג'ויסטיק פיזי המשמש כבסיס לשלבי האימון והבדיקה שלנו. חישבנו את ה- MSE (שגיאת ריבוע ממוצעת) בין מיקומי הג'ויסטיק האמיתיים לבין לתוצאה החזויה על ידי המודל.

המסקנה אליה הגענו בסוף הפרויקט היא ששיטה זו אמינה מספיק לשימוש ביישומים רבים, היא עובדת עם מגוון ידיים ורקעים.

7. Executive Summary

Human-Computer interface has a large influence on productivity and ease of use of today's tools and computers, and in recent years there were several attempts to expand new technologies to fields that require different types of handling, such as Virtual Reality gear, or heavy machinery. Equipment for this today is often clumsy and accompanied by surrounding hardware which makes it more expensive than it could be. The purpose of this project is to create a working solution for interaction between people and computers. Our project aims to provide an intuitive and relatively cheap alternative that allows precise movement in real time. It harnesses Machine-learning architecture to extract hand and joint poses from a live input stream of RGB images then map them to a joystick coordinate output. This allows the user to emulate joystick input to the system by tilting his/her hand in front of the camera.

Two system alternatives were considered:

1. Using raw images for training a regression model using an RGBD Image. The caveats of this method is the sensitivity for changes in the lighting, settings and variations in setup.

2. Using an alternative model hand-graph-cnn for obtaining the hand and joints coordinates from an RGB Image. This method is supposedly accurate, however it is not robust for hand-object interactions, and due to lack of access to the college gpu and work environment, we've decided to use a model which runs on the cpu, and provides a working prediction to a larger variety of devices.

The project consists of two networks.

1. A hand landmark network that extracts the 3d coordinates of the joints of the hands from a stream of RGB images from a camera.

2. A Coordinates-to-vector network which translates the 3d coordinates to the joystick coordinates required.

The system is fast enough to work in real time, and is functional in various conditions. The data for the training and testing phase was gathered using a physical joystick that functions as the baseline for our training phase and results. We calculated the MSE (mean squared error) between the joystick and the produced result.

This method is reliable and accurate enough for usage in many applications. The tracking works in various environments and types of hands, and requires only minor adjustments to be comfortably used by different users.

8. Glossary

General Information and Common terms:
Machine learning:

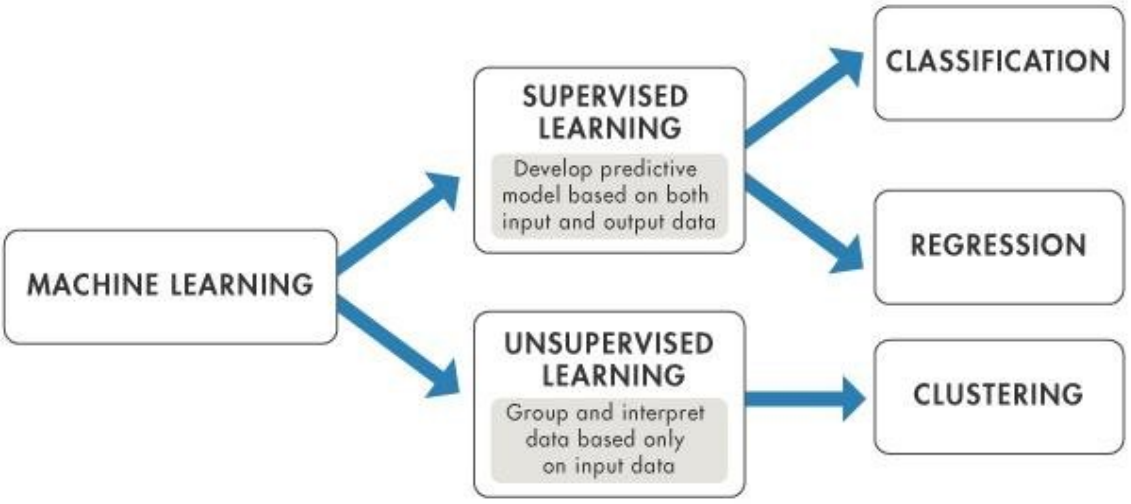


Image 8.1 Machine Learning

Machine learning is a class of methods for automatically creating models from data. Each kind of method solves a different kind of problem and depends on the computing resources available, and the nature of the data.

There are two major categories of problems that are often solved by machine learning: regression and classification. **Regression** is for numeric data while **classification** is for non-numeric data.

Independent of these divisions, there are another two kinds of machine learning algorithms: supervised and unsupervised. In **supervised learning**, you provide a training data set with answers and with the goal of finding a model that could correctly identify a picture that it had not previously seen.
In **unsupervised learning**, the algorithm goes through the data itself and tries to come up with meaningful results.

Mean Square Error:
MSE is the most commonly used regression loss function. MSE is the sum of squared distances between our target variable and predicted values.

$$MSE = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}$$

CNN:

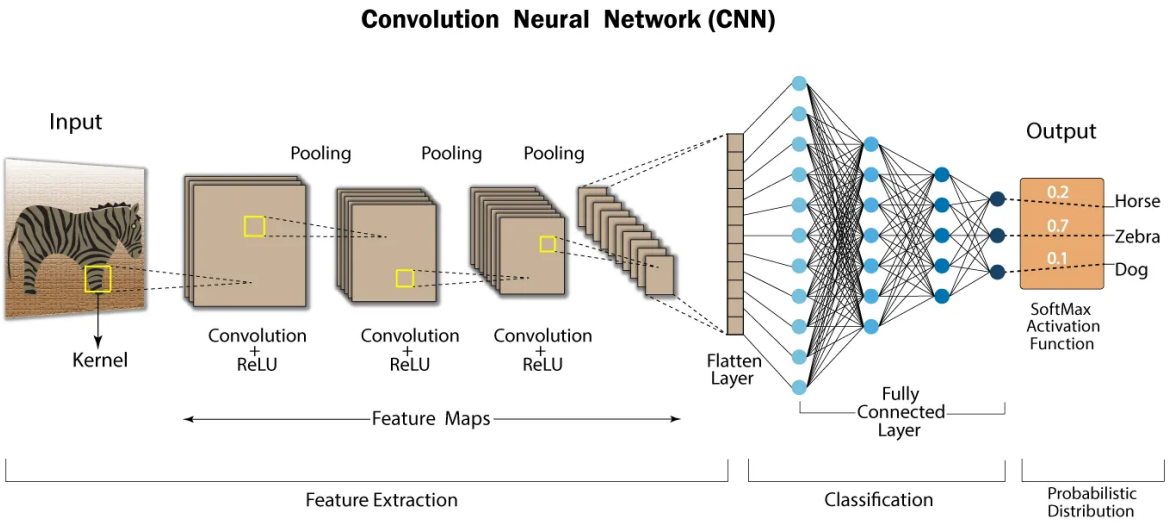


Image 8.2 CNN

Neural nets are a means of doing machine learning, in which a computer learns to perform a task by analyzing training examples. Usually, the examples have been hand-labelled in advance.

A neural network works similarly to the human brain's neural network. A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis. For every batch of examples and labels trained on ut, the network uses a technique called "back propagation" to optimize its predictions.

The methods used in this work are convolutions, multilayer perceptrons.

MANO Hand Model:

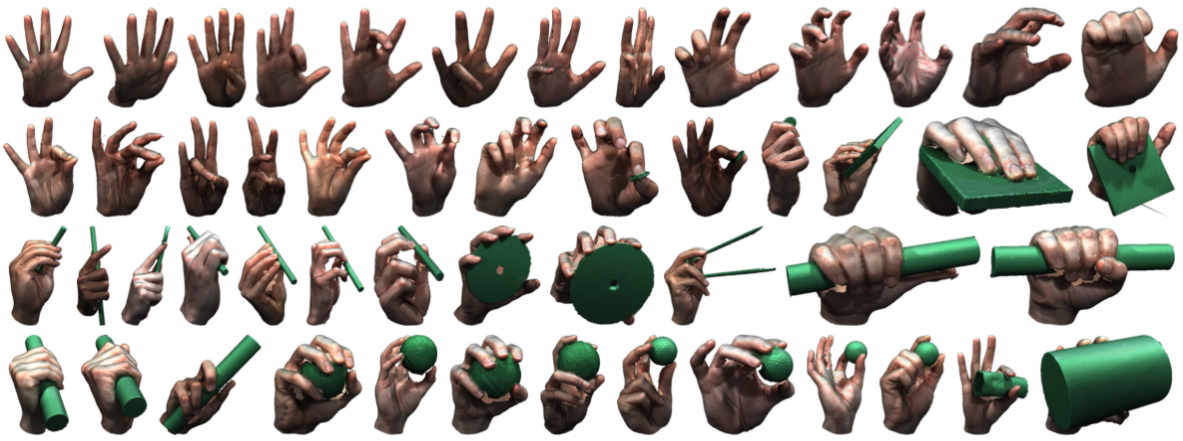


Image 8.3 MANO hand model

A realistic, low-dimensional model that captures non-rigid shape changes with pose, is compatible with standard graphics packages, and can fit any human hand. MANO provides a compact mapping from hand poses to pose blend shape corrections and a linear manifold of pose synergies.

Cosine Similarity:
The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them. This metric is a measurement of orientation and not magnitude, it can be seen as a comparison between documents on a normalized space because we're not taking into consideration only the magnitude of each word count (tf-idf) of each document, but the angle between the documents. What we have to do to build the cosine similarity equation is to solve the equation of the dot product for the $\cos\theta$:

$$\vec{a} \cdot \vec{b} = \|\vec{a}\|\|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

Exponential Moving Average:
An exponential moving average (EMA) is a type of moving average (MA) that places a greater weight and significance on the most recent data points. The exponential moving average is also referred to as the exponentially weighted moving average.

The moving average is designed as such that older observations are given lower weights. The weights fall exponentially as the data point gets older – hence the name exponentially weighted.

The Formula for EMA Is:

$$EMA_{Today} = \left(Value_{Today} * \left(\frac{Smoothing}{1 + Days}\right)\right) + EMA_{Yesterday} * \left(1 - \left(\frac{Smoothing}{1 + Days}\right)\right)$$

An EMA does serve to alleviate the negative impact of lags to some extent. Because the EMA calculation places more weight on the latest data, it "hugs" the price action a bit more tightly and reacts more quickly.

9. Introduction

Our project was aimed to improve the accessibility of human-machine interface while simultaneously making the whole process less expensive and more intuitive to the user.

The goal of our project was to estimate the (x,y) coordinates of a hand-held joystick from a live input stream of RGB images. To achieve this goal we've been using a deep learning model that extracts 3D pose of the hand and it's joints from each

frame of the input stream and then by using a custom trained model we've matched each pose to the corresponding joystick coordinates.

10. Goals

Project Functional Goals:

- Locate the movements of a hand that operates a joystick from RGB images

Project Quantitative Goals:

- Processing the Images at a rate of 10-20 HZ.

Project Measurements :

- Image Processing Rate (in Hertz)
- Precision of the joints extracted compared to the ground truth and the existing competition (in Millimetres).

11. Literature Review and Market Survey

The literature review provides a broad view of the problem at hand - integration between man and machine in the field of hand recognition. Some of the papers propose solutions to similar issues and provide an outlook on how different approaches perform, with a wide perspective on the gap between the performance of current technology trends and the goal at hand.

We will also discuss our approach of combining the separate fields of machine learning and computer vision in order to solve the problems that arise from said integration.

Papers:

- **First-Person Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations:**

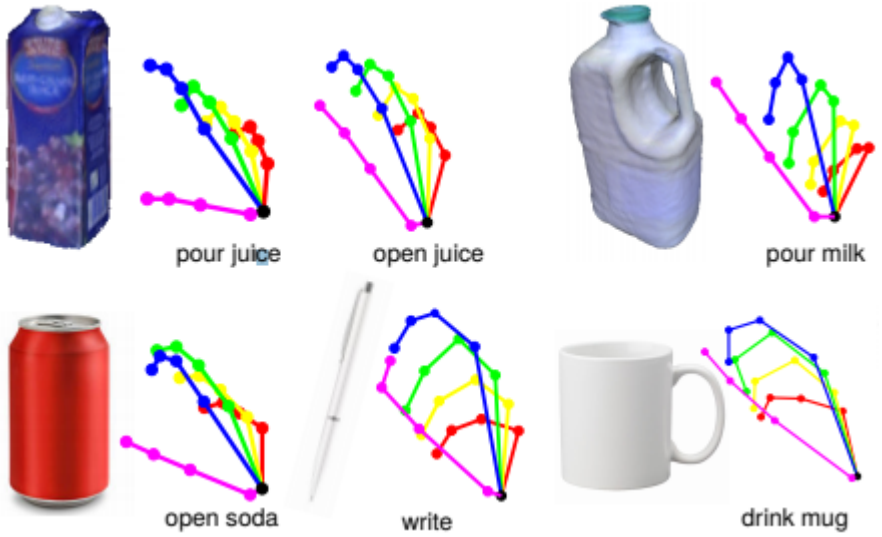


Image 11.1 Article 1

This work presents an experimental evaluation of RGB-D and pose-based action recognition by 18 baselines/state-of-the-art approaches. The impact of using appearance features, poses, and their combinations are measured, and the different training/testing protocols are evaluated. Finally, the researchers have assessed how ready the 3D hand pose estimation field was when hands were severely occluded by ob-jects in egocentric views and its influence on action recog-nition. From the results, they've seen clear benefits of using hand pose as a cue for action recognition compared to other data modalities. Their dataset and experiments can be of interest to communities of 3D hand pose estimation, 6D object pose and robotics as well as action recognition.

- **HOnnotate: A method for 3D Annotation of Hand and Object Poses:**

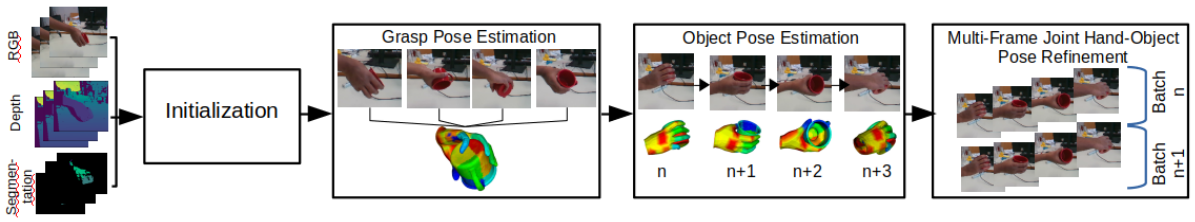


Fig 11.2 Article 2.1

This work presents new methods for training a Deep Network to predict the 2D joint locations of the hand along with the joint direction vectors and then lifting them to 3D by fitting a MANO model to these predictions.

Their work combines the use of both discriminative and generative approaches.

A Discriminative method models the decision boundary between the classes while a Generative Method explicitly models the actual distribution of each class. Both of them are predicting the conditional probability, but through different probabilities.

A Generative model learns the joint probability distribution. It predicts the conditional probability with the help of Bayes Theorem.
A Discriminative model learns the conditional probability distribution. Both of these models are generally used in supervised learning problems.

In their article the research team have discussed the issues related to each method:

Discriminative approaches directly predict the joint locations from RGB or RGB-D images. they show remarkable performance compared to Random Forests, but still perform poorly in case of partial occlusion.

On the other hand, Generative approaches take advantage of a hand model and its kinematic structure to generate hand pose hypotheses that are physically plausible. predict 2D joint locations and then lift them to 3D. They are usually accurate and can be made robust to partial occlusions. They typically rely on some pose prior, which may require manual initialization or result in drift when tracking.

The research team have decided to implement both of them in their work, A generative approach was used within a global optimization framework to generate the pose annotations, while a discriminative method was used to initialize this complex optimization. Furthermore they've trained a discriminative method to predict the hand poses which are robust to occlusions from interacting objects.

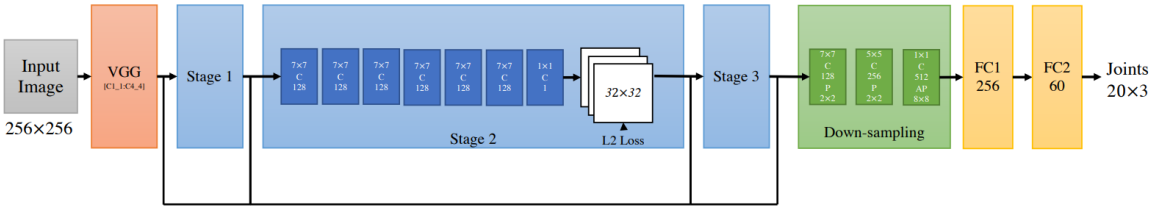


Image 11.3 Article 2.2

- **3D Hand Shape and Pose Estimation from a Single RGB Image:**

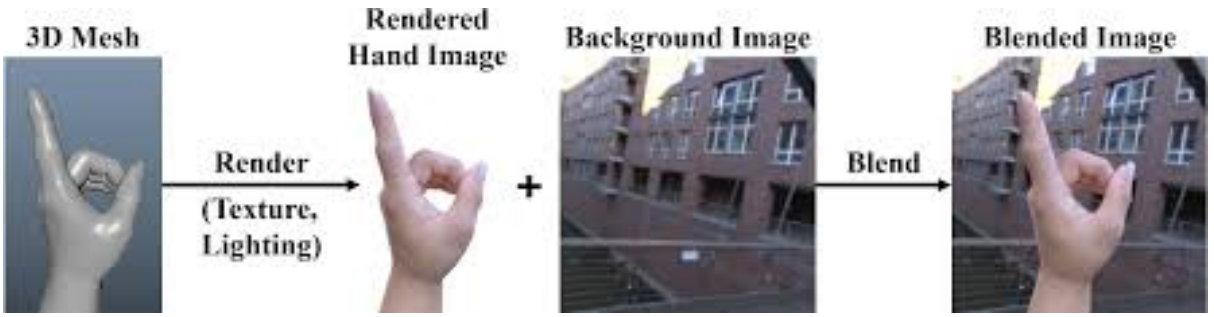


Image 11.4 Article 3

In this paper the researchers argue that the output 3D hand mesh vertices in essence are graph-structured data, since a 3D mesh can be easily represented as a graph. to output such graph-structured data and better exploit the topological relationship among mesh vertices in the graph, they propose a novel Graph CNN-based approach. Specifically, we adopt graph convolutions hierarchically with upsampling and nonlinear activations to generate 3D hand mesh vertices in a graph from image features which are extracted by backbone networks. With an end-to-end trainable framework, the Graph CNN-based method can better represent the highly variable 3D hand

shapes, and can better express the local details of 3D hand shapes.

In their paper, they were trying to solve the challenging task of how to jointly estimate not only 3D hand joint locations, but also a full 3D mesh of hand surface from a single RGB image.
The aim is to estimate 3D hand shape from a monocular RGB image, which has not been extensively studied yet by applying deep neural networks that are trained in an end-to-end manner to recover 3D hand mesh directly from a single RGB image.

In the process of achieving this goal, they came across several challenges: The first challenge is the high dimensionality of the output space for 3D hand mesh generation. Compared with estimating sparse 3D joint locations of the hand skeleton it is much more difficult to estimate 3D coordinates of dense mesh vertices using conventional CNN (21 joints compared to 1280 vertices). One straightforward solution is to follow the common approach used in human body shape estimation, namely to regress low-dimensional parameters of the MANO predefined deformable hand model.

An additional challenge is the lack of ground truth 3D hand mesh training data for real-world images.
To address this issue, they propose a novel weakly supervised method by leveraging depth map as a weak supervision for 3D mesh generation, since depth map can be easily captured by an RGB-D camera when collecting real world training data. When fine-tuning on real-world datasets, they render the generated 3D hand mesh to a depth map on the image plane and minimize the depth map loss against the reference depth map.
During testing, they only need an RGB image as input to estimate full 3D hand shape and pose.

They've conducted comprehensive experiments on their proposed synthetic and real-world datasets as well as two public datasets. According to the results their proposed method can produce accurate and reasonable 3D hand mesh with real-time speed on GPU, and can achieve superior accuracy performance on 3D hand pose estimation when compared with state-of-the-art methods.

- **MediaPipe Hands: On-device Real-time Hand Tracking:**
  In the attached article, the research team has decided to tackle the issue from a different angle. instead of relying on specialized hardware, depth sensors and powerful processors, they've proposed a solution that does not require any additional hardware and performs in real-time on mobile devices.

The solution utilizes an ML pipeline consisting of two models working together:

1. A palm detector that operates on a full input image and locates palms via an oriented hand bounding box.
   since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers the researcher focused on training their model to detect the hand palm instead of the hand itself.
   In addition palms can be modelled using only square bounding boxes while ignoring other aspect ratios, therefore reducing the number of anchors significantly.

   The palm detector is using an encoder-decoder architecture which operates similarly to Feature Pyramid Network, It combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections.

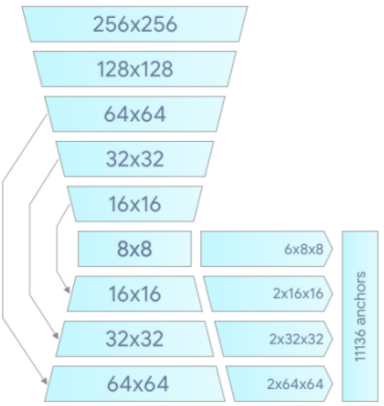


Image 11.5 Article 4

   The aim is to minimize the focal loss during the training process in order to support a large amount of anchors.

2. A hand landmark model that operates on the cropped hand bounding box provided by the palm detector and returns high-fidelity 2.5D landmarks. It performs precise land-mark localization of 21 2.5D coordinates inside the detected hand regions via regression. It learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.
   It produces 3 outputs:
   - 21 hand landmarks consisting of x, y, and relative depth.
   - A hand flag indicating the probability of hand presence in the input image.

- A binary classification of handedness,e.g. left or right hand.

Market Survey

- **Leap Motion**:

Image 11.6 Leap Motion

The Leap Motion **controller** is a small USB peripheral **device** which is designed to be placed on a physical desktop, facing upward. It can also be mounted onto a virtual reality headset.

Using two monochromatic **IR cameras** and three infrared LEDs, the device observes a roughly hemispherical area, to a distance of about 1 meter. The LEDs generate pattern-less IR light and the cameras generate almost 200 frames per second of reflected data.

The data is then sent through a USB cable to the host computer, where it is analyzed by the Leap Motion software. The software uses **calculations** in order to compare the 2d frames generated by the two cameras and synthesize from it the 3D position data.

- **Oculus Quest**:

Image 11.7 Oculus Quest

The Oculus Quest is a wireless virtual reality **headset**. It doesn't need to be tethered to a PC and doesn't require a phone.
It features six degrees of freedom (6DoF), meaning it can track up, down, left, right, forward, and backward movements. It doesn't require any external sensors, Instead, it has **sensors** built into the headset. It also supports two updated **Touch controllers**.
A **camera** in each corner of the headset (total of four) **track** space and motion controllers from the inside out.

The technology behind it makes use of wide angle cameras and **hardware** based image analysis to identify landmarks within a room and then **determine** the location and orientation of the headset based on how those landmarks appear. It uses a similar method to track the controllers, though instead of orienting them relative to the landmarks it orients them relative to the headset.

- **Wrnch**:

Image 11.8 Wrnch

wrench is a computer vision / deep learning software engineering company based in Montréal, Canada, a world-renowned hub for AI and visual computing. The wrnchAI platform enables software developers to quickly and easily give their applications the ability to see and understand human motion, shape, and intent.

The technology behind wrnch AI is based on human motion capture and activity recognition.
Human motion capture digitizes human motion, allowing machines to track or reconstruct human behavior. The main advantage of human motion capture is that large amounts of data can be processed within a few milliseconds. This enables applications to perform in real-time, such as movement analysis for sports and automation involving human-machines interactions.

Motion capture is performed via **joint skeletal tracking**, which tracks humans in a video by creating a virtual skeleton overlay. The skeleton consists of several skeletal joints and segments, representing the body parts and limbs. The number of skeletal joints can vary according to the pixel resolution, which can vary depending on how far an individual is from the camera. The timeline of skeleton point and segment coordinates forms the digitized human motion data from which movement paths and trajectories can be estimated.

Activity recognition involves tracking an individual over time as they perform a series of actions. The machine learning model compares the ongoing action to the set of actions that it was trained on, allowing it to not only recognize the actions but also assess movement deviations by comparing against the average trajectory.

- **<u>Valve Index</u>**:

Image 11.9 Valve

The headset uses LCD panels for each eye - the panels are full RGB and can operate at refresh rates of 80 Hz, 90 Hz, 120 Hz.

Central to the Lighthouse technology are the Base Stations. These Base Stations are small rectangular objects placed in the tracking area**.** They serve as reference points for any positionally tracked devices such as the HMDs and controllers. Base Stations perform this function by constantly flooding the room with a non-visible light.

The receptors on the tracked devices would intercept the light and figure out where they are in relation to the Base Stations. Multiple Base Stations (2 for SteamVR) allow the tracked devices to figure out where they are in the 3D space.

Each Base Station contains an IR beacon called Sync Blinker and 2 laser emitters that spin rapidly. 60 times per second, the Sync Blinker would emit a synchronization pulse and 1 of the 2 spinning lasers would sweep a beam across the room. The receptors, HMDs and controllers, are covered with photo sensors that recognize the synchronization pulse and the laser beams. When it detects a synchronization pulse, the receptor starts to count til one of its photosensors is hit by the laser beam. Lighthouse calculates When the photosensor is hit by the laser and Where that photo sensor is located to find the exact position of the receptor in relation to the Base Station. When there are 2 Base Stations, the position and the orientation of the receptors in the 3D space of the room is established.

Base Stations are vulnerable to occlusion. They require line of sight to the tracked objects. Base Stations are designed to be scalable. 2 Base Stations are placed in opposite sides of the room to minimize this problem. More Base Station can be placed to increase the tracking range.

Category	Leap Motion	Oculus Quest	Valve index	Wrnch	**Handy**
Price					
Spacial stability					
Surrounding hardware					
Ease of use					
Mobility					

Good	**Mediocre**	**Bad**

Table 11.1 Competitors Analysis

From the initial market survey that we've conducted we found out that all of our competitors were using sensors and wearable gear (headset, controllers, gauntlet ,etc.)
In order to track the hand's position and movement.

Our product will nullify the need to use such equipment which will further improve the user's experience and immersion into the virtual environment.
The lack of wearable devices can also contribute to a more natural hand movements and more precise actions that are not available for the user in the present - like grabbing virtual objects, turning keys and tightening a grip around ingame tools and devices.
Moreover, the client no longer needs to rely on wearable/surrounding gear to reduce the total price of VR games and attract new potential customers who couldn't afford it before.

12. Alternative Systems

1. A different CNN other than the one we've used in order to estimate the hand and joints poses.

2. An alternative simulation tool for the use of visualizing our results and demonstrating the capabilities of our product.

3. A different kind of camera for the gathering of both the training and test datasets.

13. System Requirements

Functional Requirements

- The software will accept a live input stream of RGB Images that will contain the user's hand.
- The images will be processed using two neural networks
- The system will produce a predicted 2d coordinates for the joystick along with a 3d vector of 21 hand landmarks.

Non - Functional Requirements

- The predictions of the trained model will be displayed using a visualization tool which won't require any prior knowledge in a specific field.
- The data will be entered in a consistent format and resolution
- User friendly, Simple and intuitive API.

14. Software Specifications

14.1. System Modules

Our system is comprised of several classical Machine Learning modules in addition to a physical camera:

- **RGB Camera -** produces an input stream of images in order to test the trained model and predict the joystick coordinates in real time.

- **Dataset** - composed of RGB images of the user's hand while it's holding a joystick in different poses and angles, in addition to matching joystick coordinates Labels.

- **Palm detection model -** A model to detect the presence of a hand in the image, and it's relative coordinates within it via an oriented hand bounding box

- **A hand landmark** model operates on the cropped hand bounding box provided by the palm detector and returns high-fidelity 2.5D landmarks (2d coordinates and relative depth).

- **Joints to Vector Translation model -** receives the 3D hand pose and estimates from it the Joystick x,y coordinates. The estimation is done by implementing regression in the form of a resnet model with a MSE loss function.

Entity Comprehension: ERD

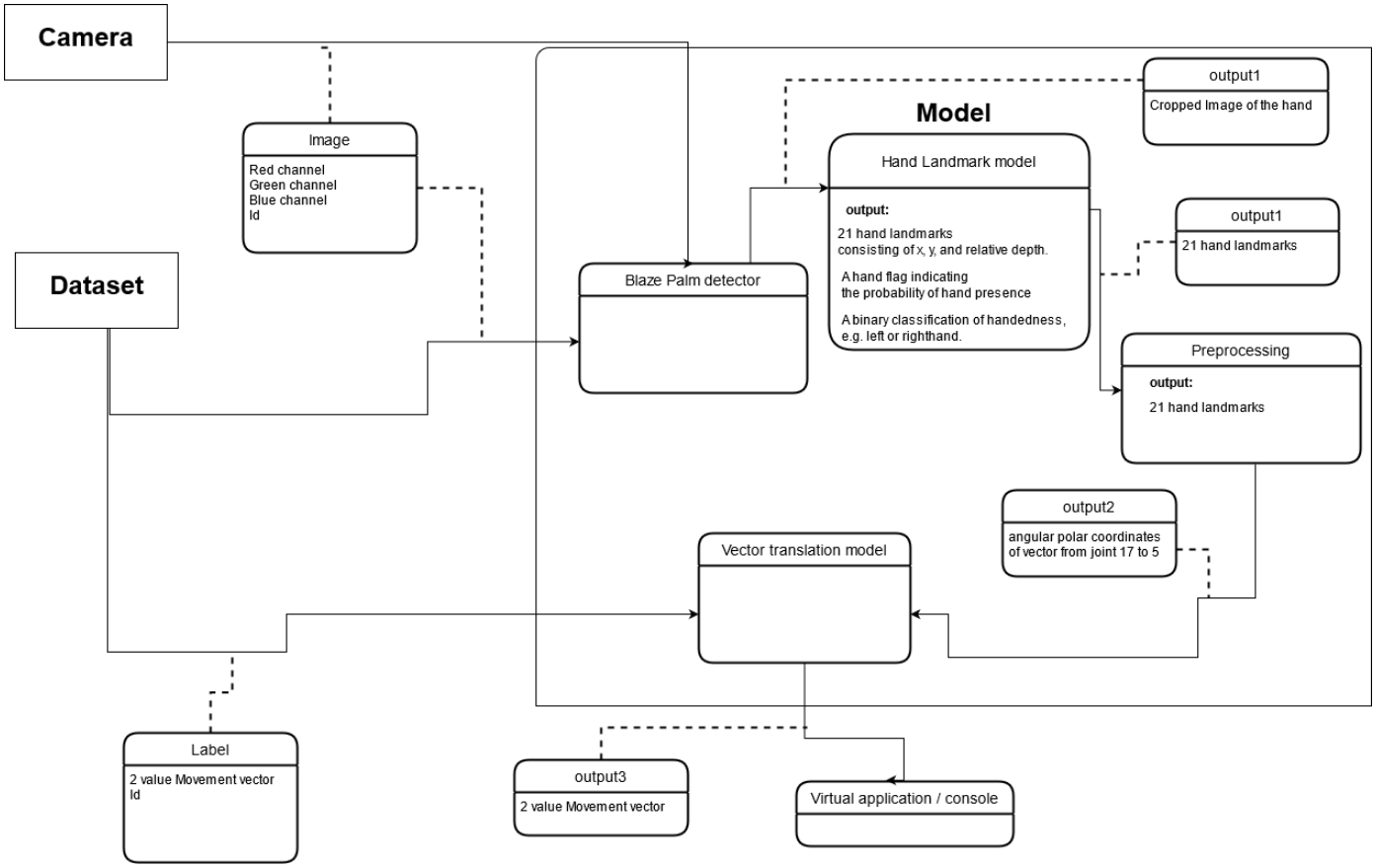


Diagram 14.1 ERD

14.2. Performance

- MSE loss below 1.00 (distance between predicted point and ground truth)
- a prediction rate of 20-30 HZ

14.3. System Constraints

The system has some constraints:

- The system will run on operating systems that support python and pytorch (Windows and Linux)
- The system will have access to a hard drive with enough space to accommodate the predicted results and images.
- The system will not predict accurately in extreme conditions, lack of light source etc.
- inorder to get the optimal results and predictions the user's hand and joystick need to stay at a specified distance from the camera.

15. Alternative Technologies

1) Multiple RGB-D cameras instead of only a single camera for the training stage

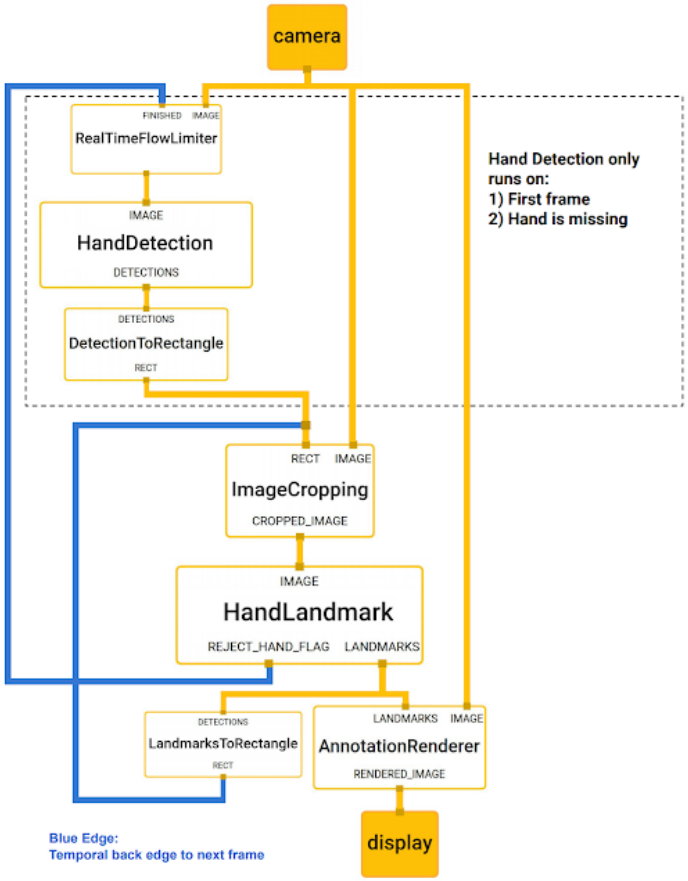
2) We can use a robotic arm that will mimic the poses shown in the input images Instead of using a virtual simulation of the model predictions.

3) We can use a graphic engine software like Unity in order to visualize the results.

4) We can use and test a different object other than the Joystick. For example a keyboard and mouse.

16. Software Design

16.1. System Architecture



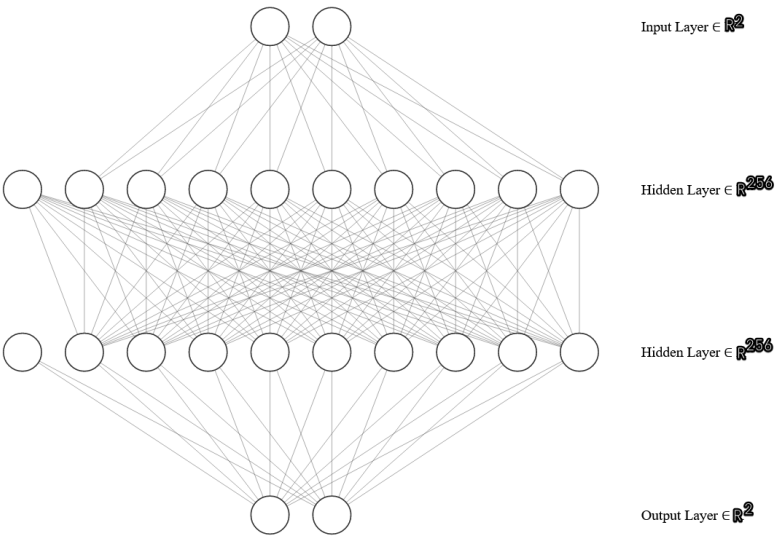


Diagram 16.1.1 System Architecture

The regression model used has 2 hidden layers with hidden dimension of 256, and an activation function of PRelu, The final layer has no activation function.

16.2. System Design

1.1. Hand Pose Estimation Model - Preprocessing, feature extraction and training.
1.2. Camera - Trained Model Interface through live feeding of the images.
1.3. Model for Joystick Coordinates Estimation - Regression model inorder to get the X and Y coordinates of the joystick.
1.4. Interface between the two models - which takes the output from the Hand Model and transfers it as input for the Coordination Model.

Interfaces:

- **Camera Interface**
  Specifications: Intel® RealSense™ Depth Camera D415
  Location: External interface which captures the user's hands as an image and transfers it to the Handy system.
  Timing: Each passing second with a set number of frames (i.e 30 frames per second)

- **Hand-model joystick-estimator interface**
  Specifications: connects the hand and joints pose estimation model to the joystick coordinates estimation model.
  Location: internal interface between the hand -joints pose estimation model to the joystick coordinates estimation model.
  Timing: Each time a batch passes through the first network and into the next one.

16.3. Alternative Designs

Alternative Function	Description	Advantages	Disadvantages	Comparison Criteria	Score
Tensor Flow	An open source library for numerical computation and large-scale machine learning	Better computational graph visualizations. Compatible with many languages.	Not intuitive to use on it's own. has low speed with respect to its competitors has only NVIDIA	Ease of use	7/10

		TensorFlow has compatibility with Keras, which allows its users to code some high-level functionality sections in it	support for GPU and python support for GPU programming does not provide much features for the Windows Operating System users	GPU support Speed	
R	An open source environment for statistical programming and visualization	Widely used in statistical analysis and supports on many systems	Not intuitive to use, doesn't have a wide selection of machine learning models		5/10
Weka	An open source software that contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces	A whole range of data preparation, feature selection and data mining algorithms are integrated. Also comes with a GUI, which should make it easier to use	Doesn't implement the newest techniques and doesn't support numeric estimation. on large datasets, the running time can become quite long	Function variety	4/10
Pytorch	An open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing	Easy to use, compatible with the mesh-pose network in the project and has a wide support of libraries and networks	Has lower visualization capabilities compared to the alternatives		8/10
Jax	A Python library designed for high-performance numerical computing, especially machine learning research	Fast compared to other systems. Support syntax similar to numpy	Relatively new library, has less documentation and open libraries compared to pytorch and tensorflow	Visualization	7/10

Table16.3.1 Alternative Designs

16.4. Block Diagram

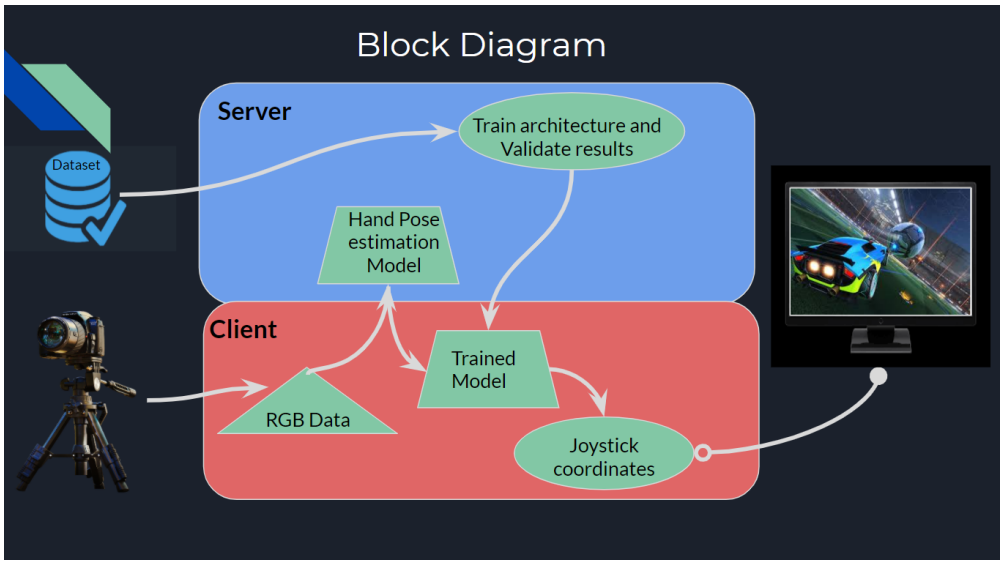


Diagram 16.4.1 Block Diagram

The server trains the custom regression model on a dataset which consists of images of hands moving a joystick and labeled 2D joystick coordinates, The model is then saved to the clients computer and fed inputs from a live RGB stream and displays the predicted 2D coordinates to the client via a visualization tool.

- RGB Camera: A camera with three color channels to record images.
- Server: necessary for the training stage, used to train the network before its deployment at the Client side.
- Client: The user of the software, will stream a live input of images of his hands using a RGB camera to the software in his computer which will output the joystick coordinates.
- Simulation: a visualization of the 2D joystick coordinates alongside 21 hand landmarks that will be presented to the client on each frame that has been taken from the camera.

16.5. Algorithms

Pose Estimation:

Single image hand pose estimation is a very popular problem in computer vision, and approaches can be divided into discriminative and generative methods. Discriminative approaches directly predict the joint locations from RGB or RGB-D images.

Recent works based on deep networks show remarkable performance, compared to early works based on random forests. However, discriminative methods perform poorly in case of partial occlusion.

Generative approaches take advantage of a hand model and its kinematic structure to generate hand pose hypotheses that are physically plausible, predict 2D joint locations and then lift them to 3D.
Generative approaches are usually accurate and can be made robust to partial occlusions. They typically rely on some pose prior, which may require manual initialization or result in drift when tracking.

This model is related to both discriminative and generative approaches: we use a generative approach within a global optimization framework to generate the pose annotations, and use a discriminative method to initialize this complex optimization. This model is trained using a discriminative method to predict the hand poses which are robust to occlusions from interacting objects.

Graph CNNs for Mesh and Pose Estimation:

A 3D mesh can be represented by an undirected graph $M = (V, \epsilon, W), where\, V = \{v_i\}_{i=1}^{N}$ is a set of N vertices in the mesh, $\epsilon = \{e_i\}_{i=1}^{E}$ is a set of E edges in the mesh, $W = (w_{ij})_{NxN}$ is the adjacency matrix, where $w_{ij} = 0$ if $(i, j) \notin \epsilon$ E, and $w_{ij} = 1$ if $(i, j) \in \epsilon$. The normalized graph Laplacian is computed as $L = I_N - D^{-1/2} W D^{-1/2}$, where $D = diag(\Sigma j\ w_{ij})$ is the diagonal degree matrix, I_N is the identity matrix.

Here, we assume that the topology of the triangular mesh is fixed and is predefined by the hand mesh model, i.e., the adjacency matrix W and the graph Laplacian L of the graph M are fixed during training and Testing. Given a signal $f = (f_1, \cdots, f_N)^T \in R^{NxF}$ on the vertices of graph M, it represents F-dim features of N vertices in the 3D mesh. In Chebyshev Spectral Graph CNN, the graph convolutional operation on a graph signal $f_{in} \in R^{NxFin}$ is defined as $f_{out} = \sum_{K=0}^{K-1} T_k(\bar{L}) * f_{in} * \Theta_k$, where $Tk(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ is the Chebyshev polynomial of degree k,

$T_0 = 1, T_1 = x; \tilde{L} \in R^{N \times N}$ is the rescaled Laplacian, $\tilde{L} = \frac{2L}{\lambda_{MAX}} - I_N$, λ_{max} is the maxi-mum eigenvalue of L; $\theta k \in R^{Fin \times Fout}$ are the trainable parameters in the graph convolutional layer; $f_{out} \in R^{NxF_{out}}$ is the output graph signal. This operation is K-localized since Eq. 1 is a K-order polynomial of the graph Laplacian, and it only affects theK-hop neighbors of each central node.

Readers are referred to the SRD appendix [6] for more details.

17. Product

17.1. Detailed Description

Our product is a local computer application that enables the user to interact with external softwares which receive a joystick as input - using his/her hands.

The alpha prototype that we've designed operates as a simulation system that integrates between the various external modules which are available to the user - like the joystick controller and the camera that sends a live feed input stream to the application.

The system relies on the requirements that were described in the system requirements section in order to function properly, and meet the functional requirements that we've set up at the beginning of the work:

- The software accepts input of RGB Images
- The software processes the images in two consecutive deep learning networks
- The first network outputs 3D coordinates of 21 hand landmarks including the palm position, while the second network outputs a 2D coordinate of the joystick position.
- The software requires a small-medium sized dataset of RGB images with labels for the coordinates of the joystick (the training set consists of about 6000 images)

The system receives a stream of RGB Images and as an input, It then uses two independent neural networks:

- The first network receives an RGB image containing a hand and estimates the 3D coordinates of 21 landmarks of that hand.
- The second network receives the coordinates of the joints and, using the relative angle between the knuckles of the hand and the XZ plane, estimates the corresponding point of the "joystick" on a 2D coordinates system.

The result is then displayed on the screen via a visualized simulation of the 21 hand landmarks and the estimated 2D coordinates of the joystick.

17.2. Specifications

Hardware:

- RGB Camera
- A computer which supports pytorch
- A joystick controller

Software:

Python with the installed libraries:

- Pytorch
- Media Pipe
- CV2
- Matplotlib
- Numpy
- Pygame - inorder to gather the initial data from the joystick and establish the training and test sets.

17.3. Software Modules and Infrastructures

Modules:

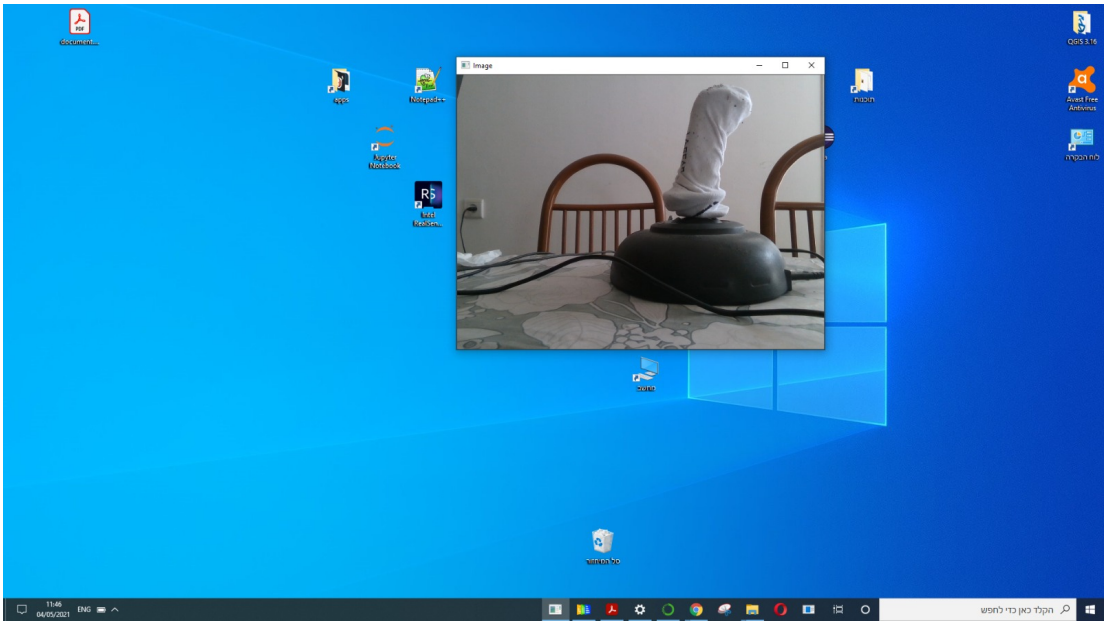
- A module for training and evaluation of our custom trained model.
- A client-side software that runs live on the clients pc and estimates the 2D coordinates of a hand-held joystick using a given input stream of RGB images.

17.4. UI/UX and Use Cases:

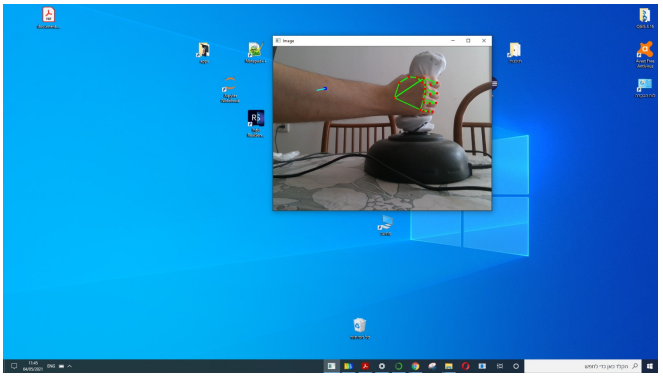
Main Flow: Using unplugged Joystick for a more stable grip

1. The user sets up the scene with the appropriate requirements.
2. The user runs the software
3. The system presents to the user a windowed video that is being updated each frame.

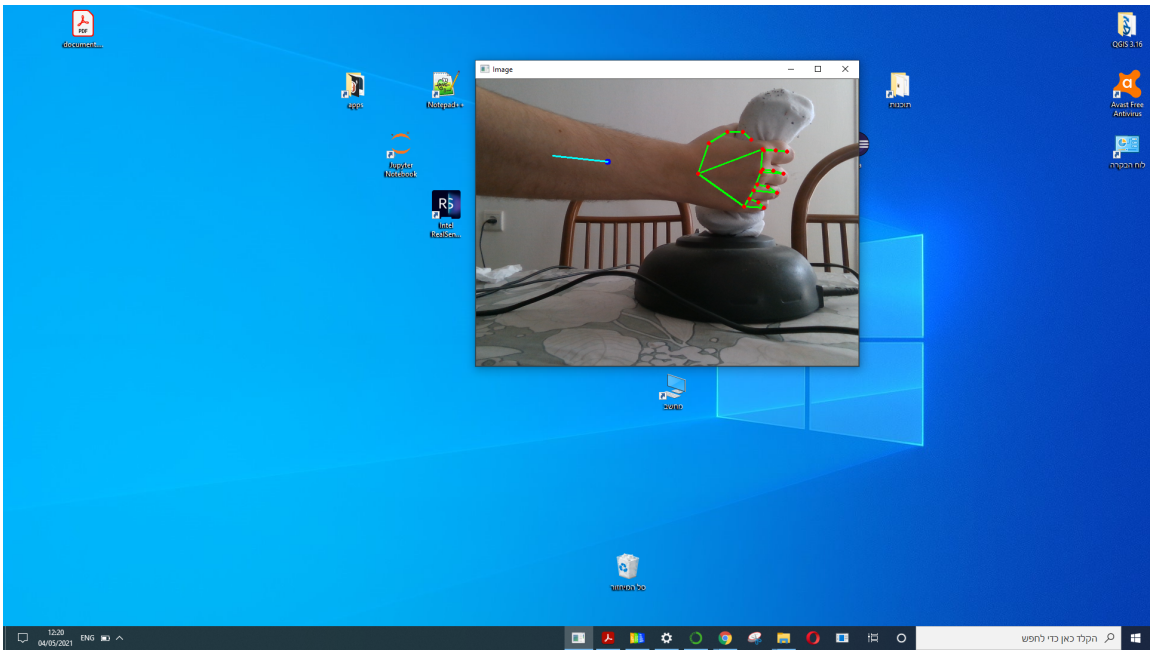
Image 17.4.1 Use Cases



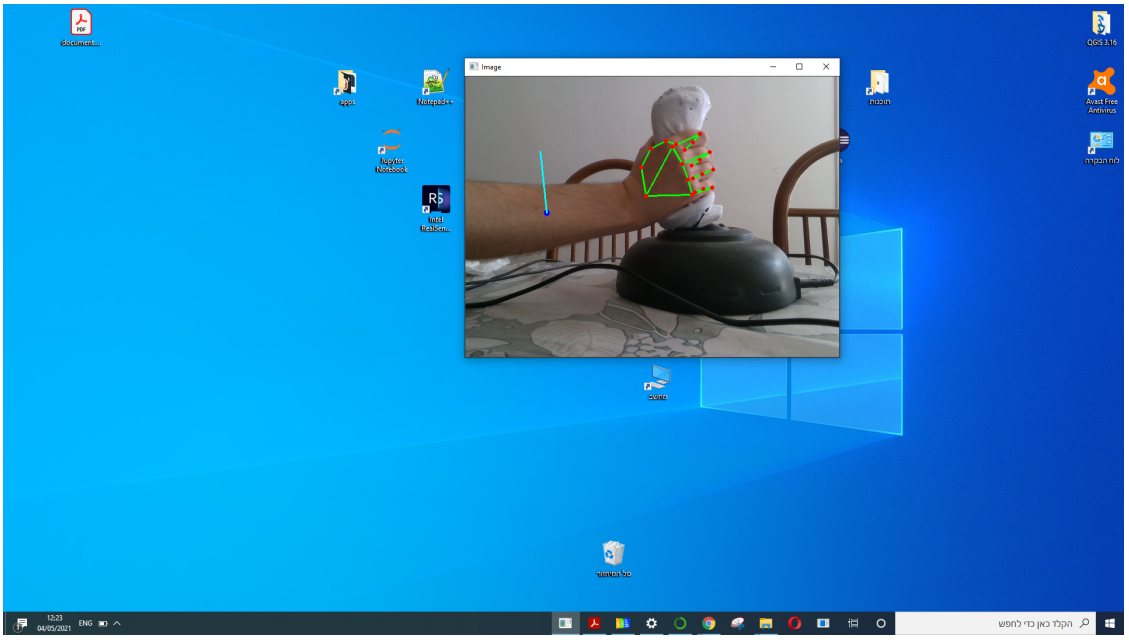
4. The user grabs the joystick and adjusts it so that the marker will align with the center dot.



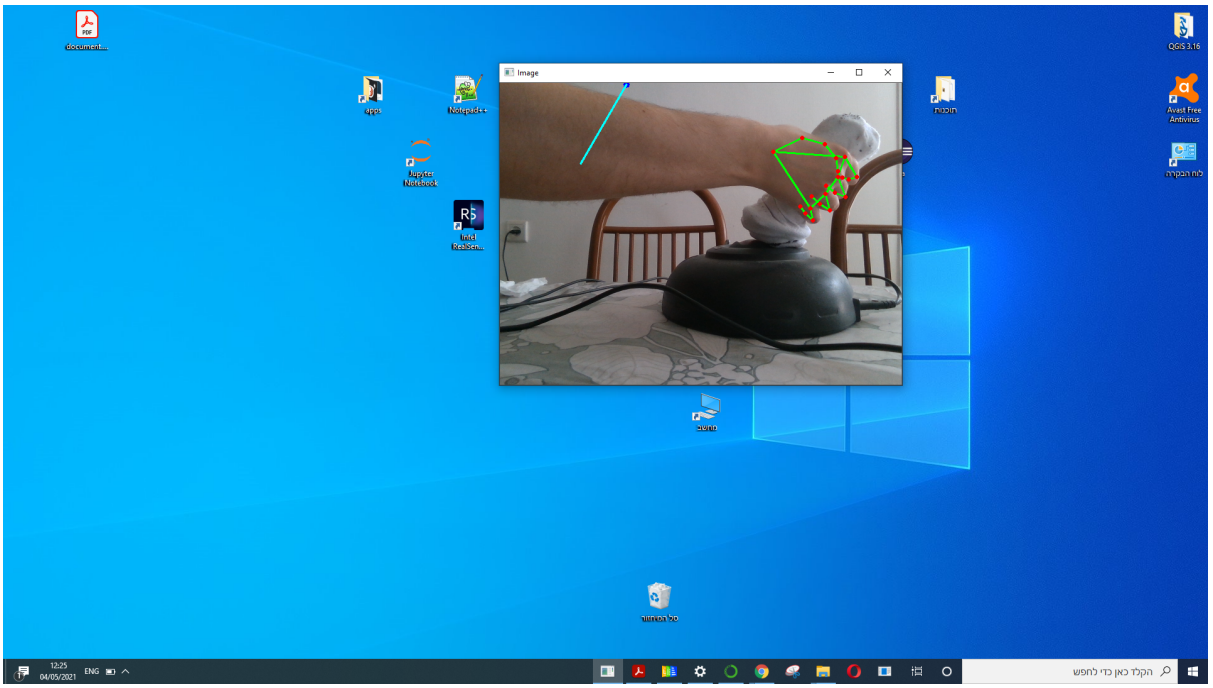
5. When the software detects the user’s hand in a specific frame it displays the predicted 2d joystick coordinates on top of the frame in addition to the 21 hand landmarks.

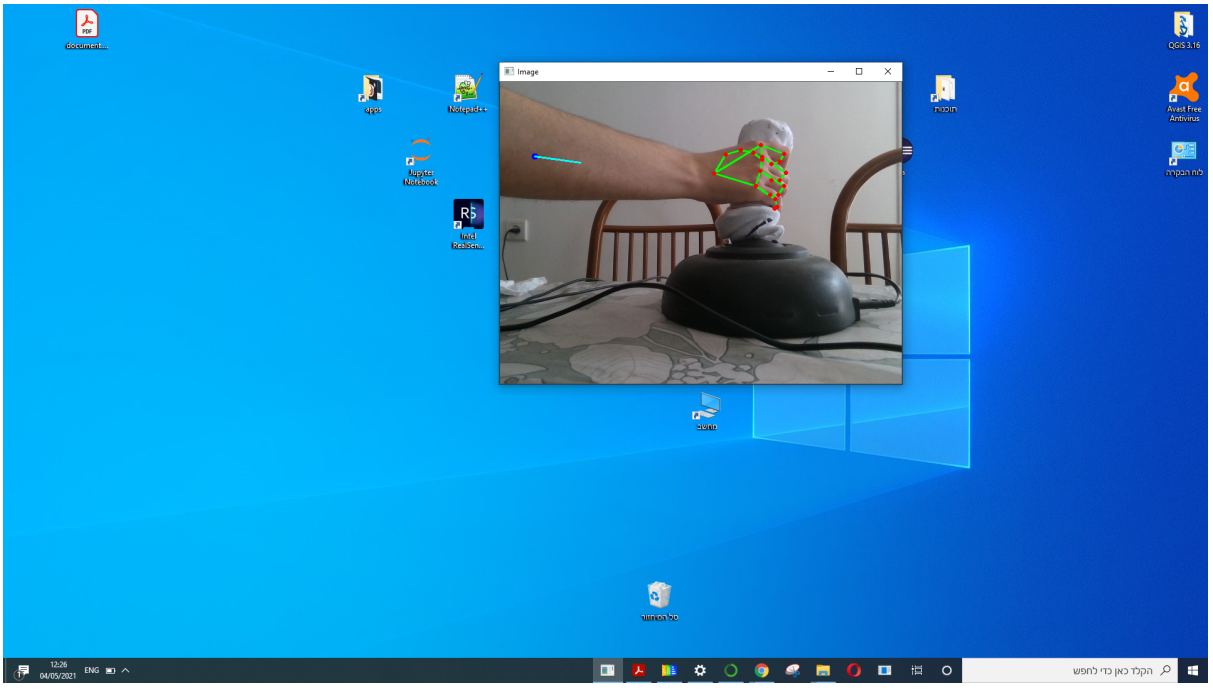


Image



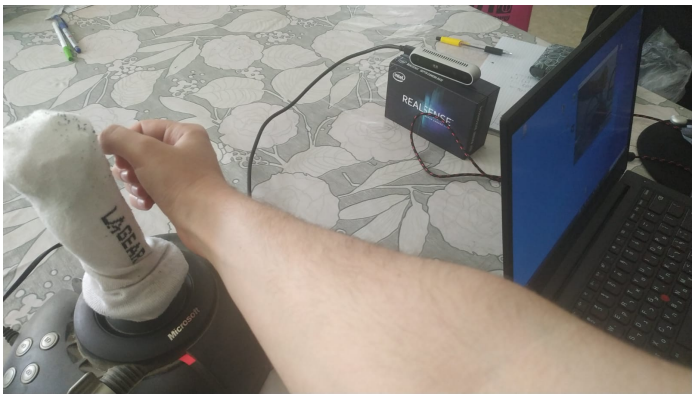
Image

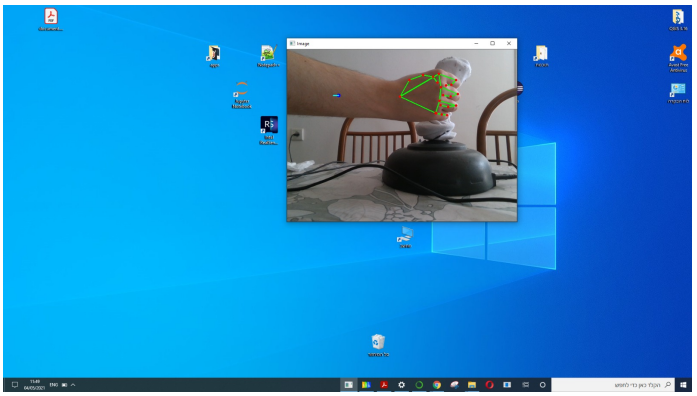




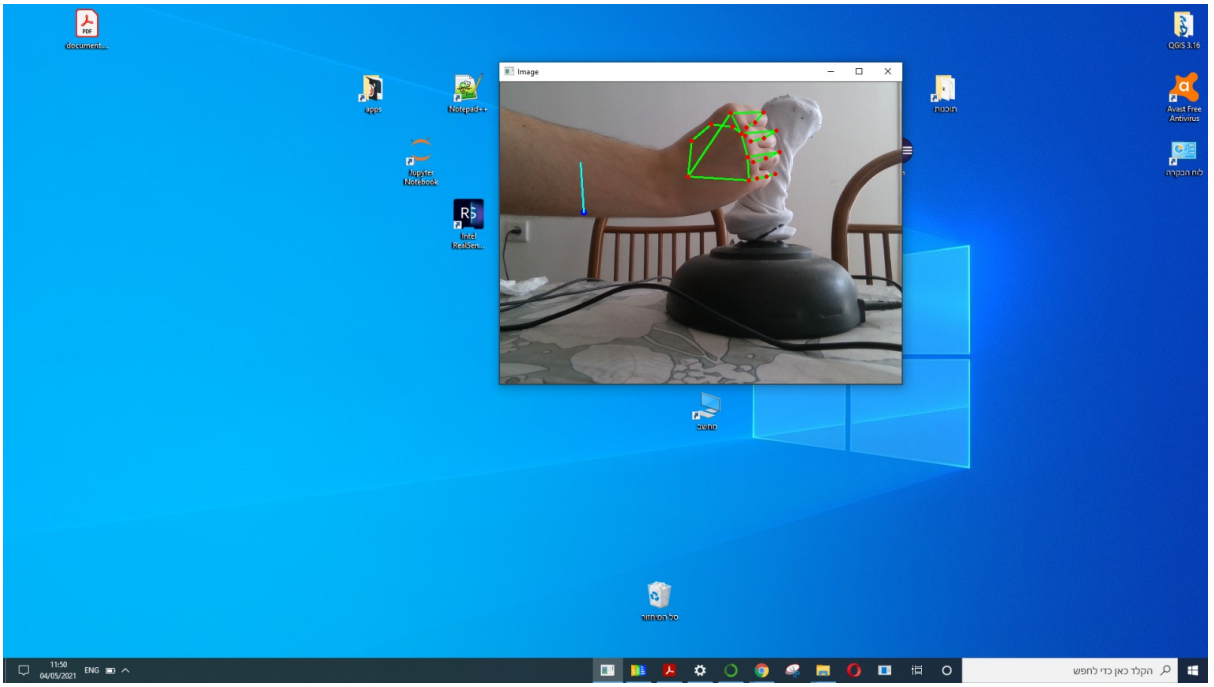
Alternative Flow: Using the user's hand instead of the joystick

4.1 The user imitates the previous action by grabbing an 'invisible' joystick. he/she then adjusts the grip so that the marker will align with the center dot.





5.1 When the software detects the user's hand in a specific frame it displays the predicted 2d joystick coordinates on top of the frame in addition to the 21 hand landmarks.



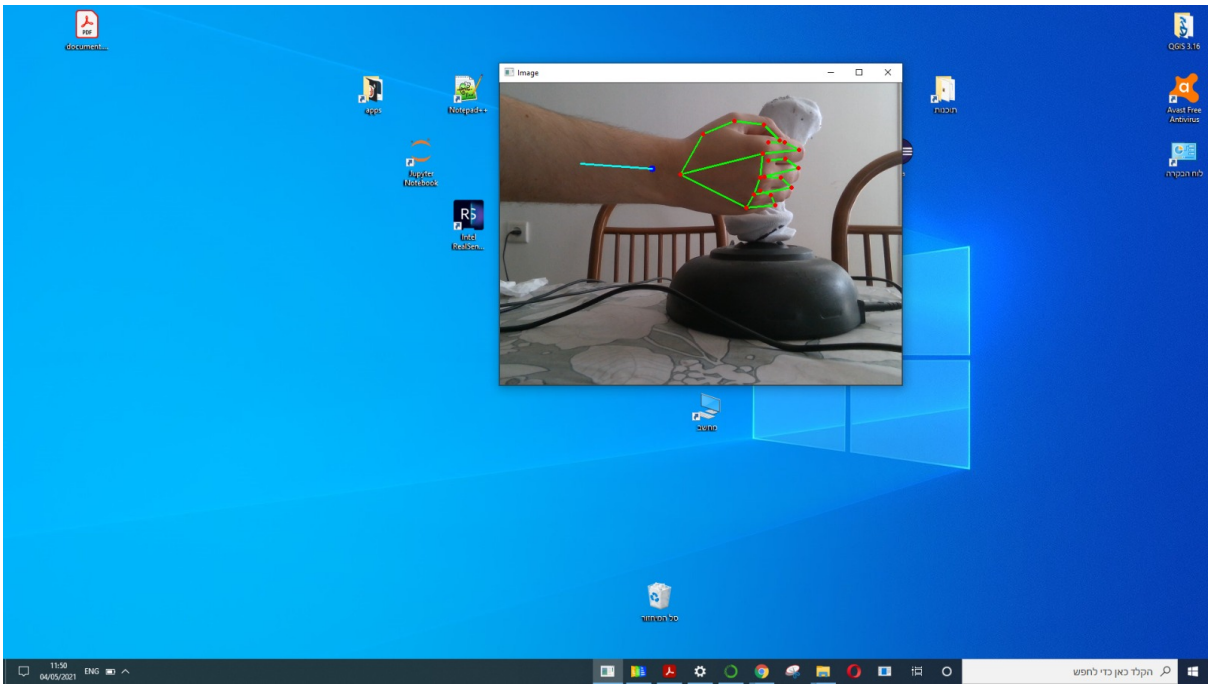


Image
הקלד כאן כדי לחפש

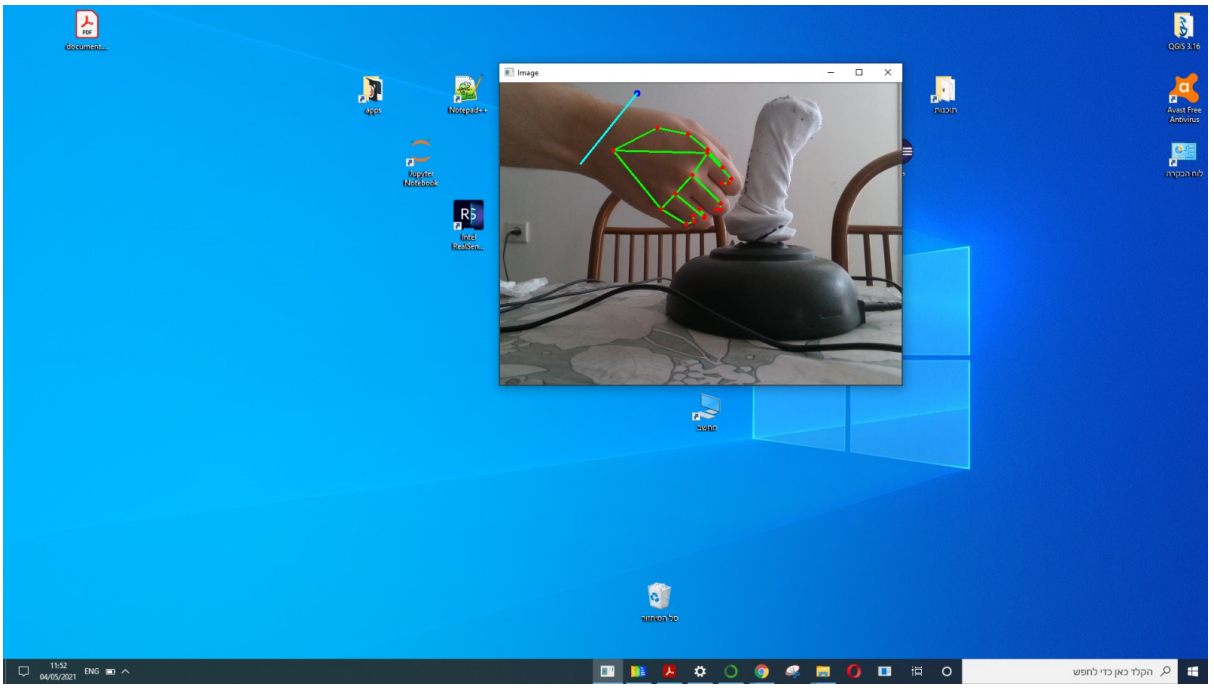


Image
הקלד כאן כדי לחפש

17.5. Final Results/Products

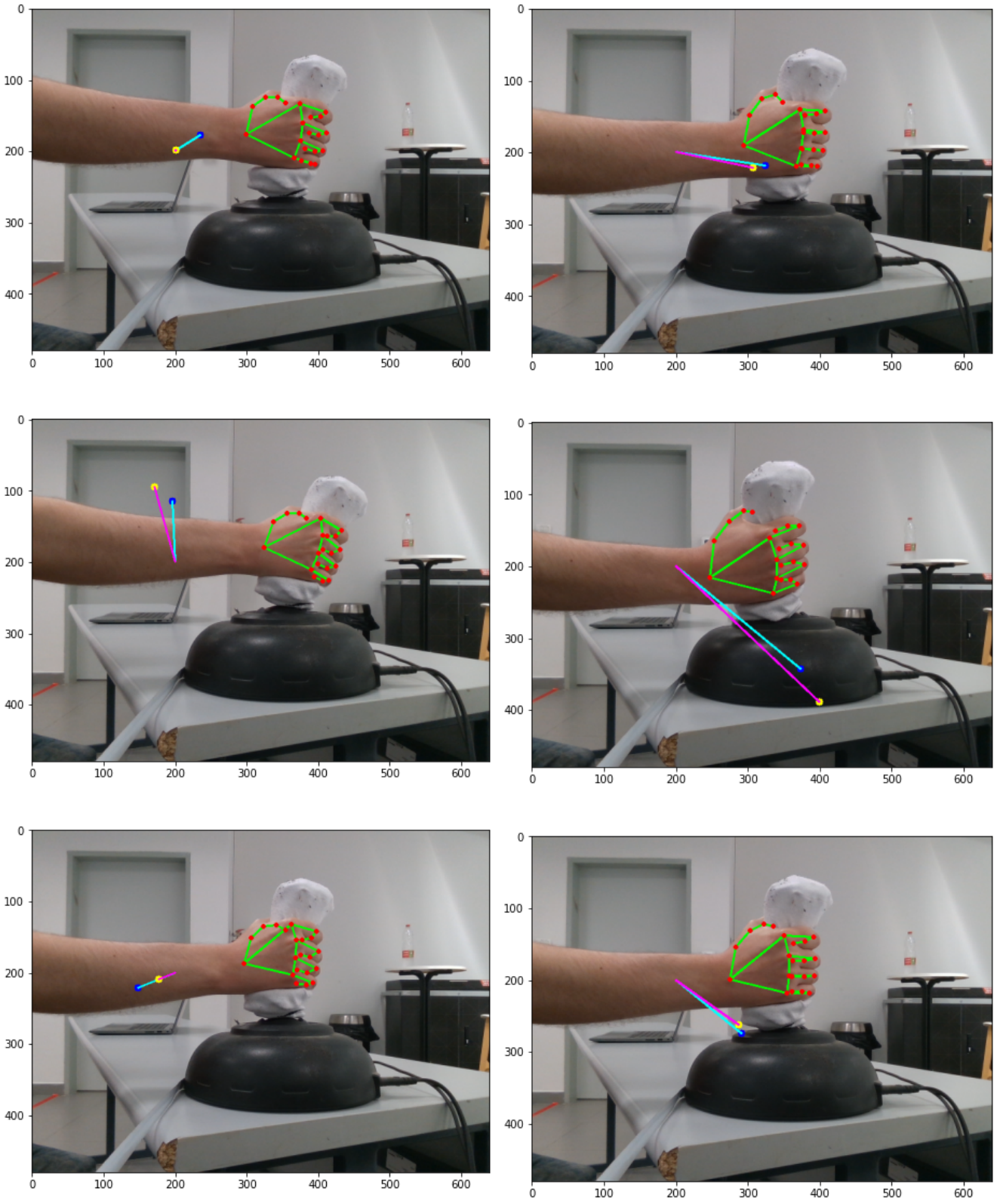


Image 17.5.1 Final Results

Training loss graph:

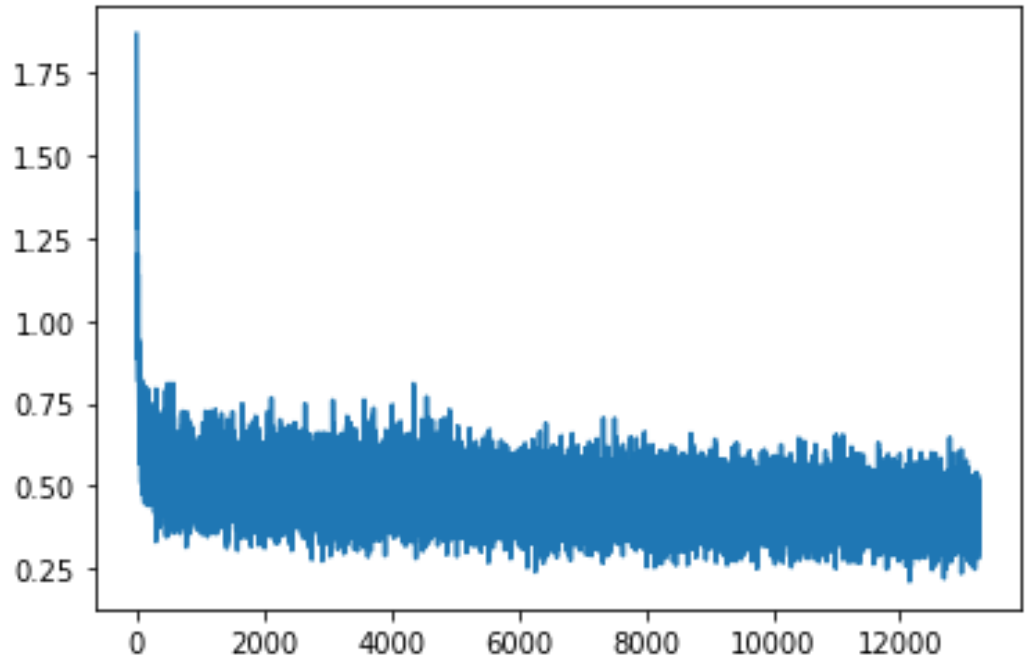


Diagram 17.5.2 Training Loss

Validation loss graph:

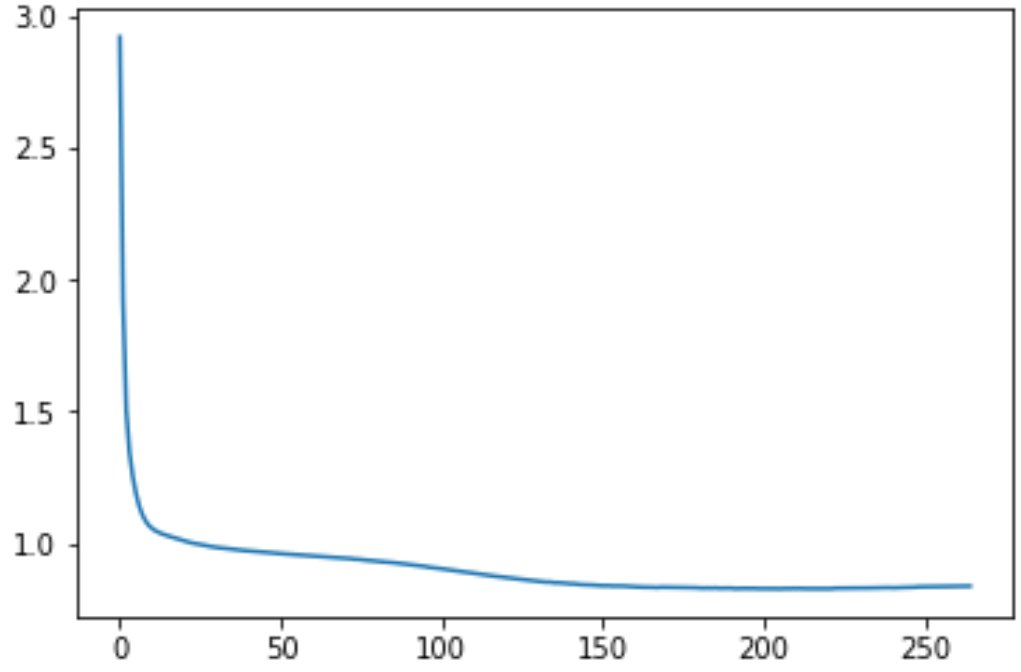
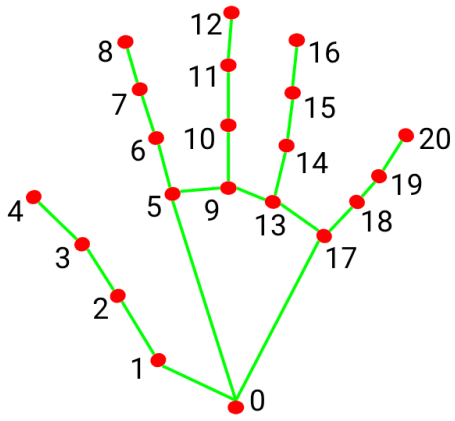


Diagram 17.5.3 ValidationLoss

17.6. Algorithms

First the BGR image is read from a camera and passed to an instance of Hand_detector class. The class extracts and returns a dictionary containing 21 elements. Each element contains a list with 3 floating point numbers that details the 2d pose and relative depth of each of the 21 landmarks of the hand in the image.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Image 17.6.1 Hand Landmarks

We are using joints number 17 and 5. Which are the positions of the knuckles of the Index and the little finger respectively.

To keep the algorithm robust for several different types of hands, camera distances and general setup, we calculate the angular positions of the vector in a polar coordinate system on a unit sphere. This method has been proved to generalize the results to many types of hands, distances and variations in the setup, with the caveat of small variations in the initial (0,0) position of the joystick.

The angular coordinates are inserted to a custom regression model inorder to produce the final x,y coordinates of the joystick.

The coordinates are clumped into a range between 0 and 1 incase of a deviation in the result. to keep the output stable we average the last four xy coordinates that have been received.

18. Project Planning:

18.1. Programme of Work

GANTT CHART

TASK NAME	START DATE	END DATE	START ON DAY*	DURATION* (WORK DAYS)	TEAM MEMBER
Programme Of Work	3/15	5/25	0	437	
Selecting the Subject of the Project	3/15	3/19	0	5	
First team meeting and brainstorming	3/15	3/17	0	3	Nir & Daniel
Initial gathering of information	3/17	3/18	2	2	Nir & Daniel
Advisor Confirmation	3/18	3/19	3	2	Ehud Dayan
Submission of the proposed project	3/19	3/19	4	1	
SOW - Statement Of Work	3/19	7/13	4	117	
Team meeting	3/19	3/19	4	1	Nir & Daniel
Searching for papers and reviewing article:	3/20	3/22	5	3	Nir & Daniel
Objectives, goals and measurements	3/23	3/23	8	1	Nir & Daniel
Initial Elevator Pitch Presentation	5/3	5/3	49	1	Nir & Daniel
Literature Review	5/4	5/7	50	4	Nir & Daniel
Introduction	5/8	5/8	54	1	Nir & Daniel
Current Solutions	5/9	5/9	55	1	Nir & Daniel
Competitors Comparison	5/10	5/10	56	1	Nir & Daniel
Initial Market Survey	5/11	5/11	57	1	Nir & Daniel
System Requirements	5/12	5/13	58	2	Nir & Daniel
Substitute Technologies	5/13	5/14	59	2	Nir & Daniel
Tools and Means	5/14	5/14	60	1	Nir & Daniel
Gaps of Knowledge	5/15	5/15	61	1	Nir & Daniel
Products: General Description	5/15	5/16	61	2	Nir & Daniel
Risk Management	5/15	5/16	61	2	Nir & Daniel
SOW Presentation	5/17	5/17	63	1	Nir & Daniel
Sending the draft to the Project Advisor	7/1	7/1	108	1	Nir & Daniel
Correcting the draft	3/19	3/19	4	1	Nir & Daniel
Advisor Confirmation	3/19	3/19	4	1	Ehud Dayan
Submission of the project's SOW	7/13	7/13	120	1	Nir & Daniel
PR - Progress Report	7/13	9/30	120	80	
Submission of the project's PR	7/13	7/13	120	1	Nir & Daniel
MR - Midterm Report	9/30	1/17	199	110	
Sending the draft to the Project Advisor	1/3	1/3	294	1	Nir & Daniel
Finishing the Alpha	1/3	1/10			Nir & Daniel
Correcting the draft	1/10	1/17	301	8	Nir & Daniel
Submission of the project's MR	1/17	1/17	308	1	Nir & Daniel
Project Journal - Final Report	1/17	5/25	308	129	Nir & Daniel
Initial testing phase	1/18	2/15	309	29	Nir & Daniel
Improvement of the code	2/15	3/15	337	29	Nir & Daniel
QA testing	3/15	5/25	365	72	Nir & Daniel
Submission of the project's Final Report	5/25	5/25	436	1	Nir & Daniel

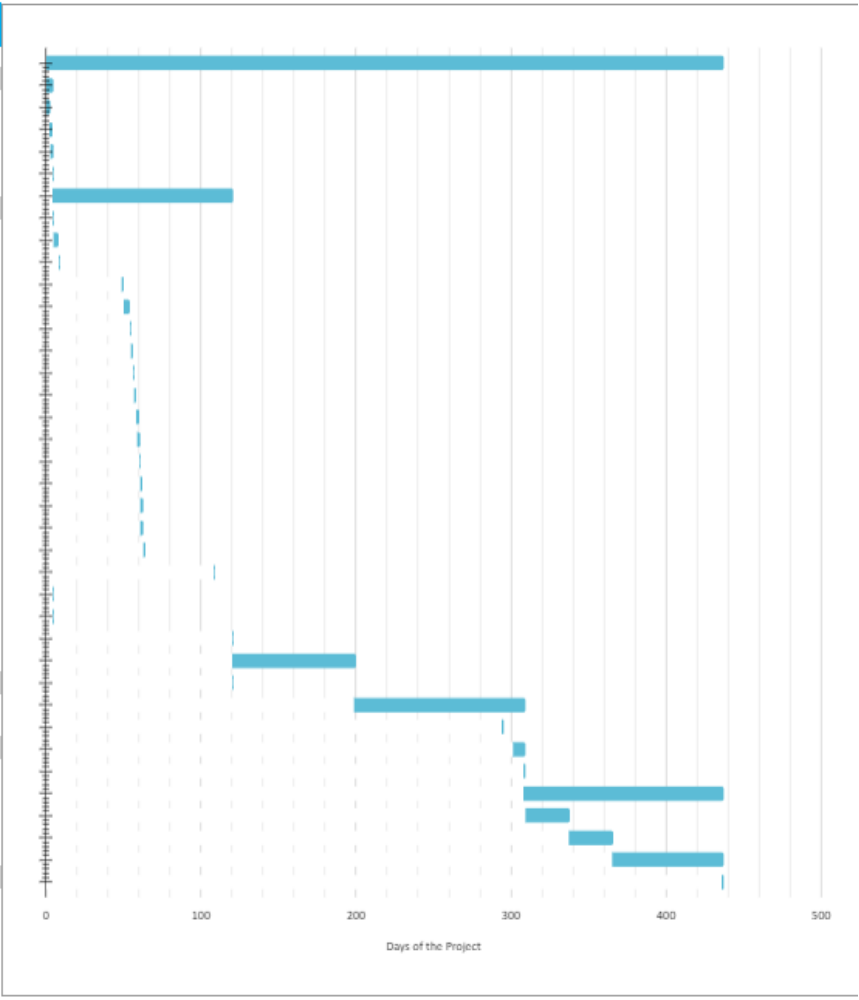


Table 18.1.1 Programme of Work

18.2. Project Updates

Updates	Section	Timing	Initiator	Impact
Added an additional article	Literature Review	Within the Final Report timetable	Students	More refined algorithms
Replaced a layer of the model	Architecture	Within the Final Report timetable	Students	More accurate model

Table 18.2.1 Updates

18.3. Risk Management

Category	Possible Risk	Severity	Likelihood	countermeasure	schedule	Status
Knowledge Gaps	Lack of knowledge in the field of hand pose estimation	0.3	0.1	Intensive research process	Within the SOW timetable	Completed
	Lack of background in the required field	0.3	0.1	Intensive research process	Within the SOW timetable	Completed
	Lack of knowledge in available solutions	0.3	0.2	Intensive research process	Within the SOW timetable	Completed
	Lack of knowledge in relevant algorithms	0.5	0.3	Self-study of the necessary models	Within the Midterm Report timetable	Completed
	Lack of knowledge about advanced functions in the field of hand pose estimation	0.5	0.3	Self-study of the necessary models	Within the Midterm Report timetable	Completed
Issues regarding the development of the Algorithm	Under performing dataset	0.5	0.2	Running multiple simulation before starting the development	Within the Midterm Report timetable	Completed
	Model doesn't correlate well with the ground truth	0.7	0.4	Running multiple simulation while developing the algorithm	Within the Midterm Report timetable	Completed
	Bad choices of HyperParameters	0.5	0.4	Exploring the best hyper parameters via cross validation	Within the Midterm Report timetable	Completed
	Results with low performance	0.5	0.5	Variety of datasets under different conditions	Within the Midterm Report	Completed

	under different conditions				timetable	
General	Non-compliance with deadlines	0.7	0.3	Planning ahead using a schedule	Within the Midterm Report timetable	Completed
	Data Loss	0.6	0.3	Saving data in backup storage	Within the Midterm Report timet	Completed

Table 18.3.1 Risk Management

19. Software Testing and Evaluation

In order to ensure that the product is meeting our performance expectations, we've planned a couple of tests that will check the different modules and components of the system.

The software was tested in a **variety of settings** - different hours of the day, different light sources, and with a number of users.
the variety of settings helped to decipher if the model had suffered from overfitting and to verify the model's strength and adaptability to changes.

In addition we've prepared a **boundary value analysis** and dealt with model predictions which passed the joystick boundary values.
the joystick can move in the horizontal or vertical axes in a set range of values, from $[(-1, -1), (1, 1)]$. the predictions of the model can sometimes contain values that pass that boundary. In that case we've clamped the received results between the two boundary values with a simple function so they would not interfere or alter future predictions.

in regards to the **model reliability** - we've dealt with situations when the user's hand was not present in the input image or in situations when the model did not detect the hand in the image for some reason. in order to prevent an exception in the software we simply default the position to the (0,0) coordinates if a hand is not detected.

Completeness - the product meets all of the requirements and goals we've set up for it in the beginning of the work.

Quality of the results - there is a 'blind spot' in the 1st quadrant in which the model outputs predictions with a much lower quality. in all the other regions the predictions have a significantly higher quality and the results are much closer to the actual positions of joystick

Costs:

- The system contains a small sized dataset - at around 6000 of images and appropriate labels. the space that is required from the user is very limited which does not require the use of any external hard disk in order to contain the software files or data.

- The system requires the use of a single RGB camera only.
  At the initial stages of the work we've parchaused an advanced RGB-D camera with an added depth sensor inorder to extract and gather a refined input stream for our dataset and improve the 3D capabilities of our trained model. As we've progressed farther in the project we came across more sophisticated techniques and machine learning models that could use a regular RGB image and still improve on the overall results and detection accuracy. therefore the purchase of an expensive camera is no longer needed and the overall expenses are significantly reduced.

- The system relies on the use of a joystick controller in order to gather the labels for the training stage of the model. but the final product itself uses only the user's hands as a substitute controller for the joystick and no additional hardware is needed.

- The system relies on free libraries and softwares only (Pytorch, CV2, Media Pipe)

20. Conclusion

This project has been a great opportunity for us to experience the development process of software and machine learning product end-to-end.

We've learned much about the importance of requirements engineering and gained hand-on experience in writing specifications. We've had some difficulties regarding the various technologies and tools that we could choose from in order to accomplish the goals we have set out for ourselves.

working on the project has taught us to perform proper research and academic finding, influencing our design decisions in software, hardware and machine learning algorithms and architectures.
It was also an excellent opportunity to employ new technologies unfamiliar to us and delve in the field of pose estimation and hand tracking.

At the beginning of the work we've conducted a wide-range research of alternative technologies in the hopes of finding the one that could yield the best results in our case. We did find some promising architectures and technologies in the field that

looked perfect for our needs at first glance, but turned out to be less than optimal when dealing with a live input stream of images. The architecture was heavily layered and far too complicated for pose estimation in real time.

During the year new articles were published which had a fresh angle on the issue of hand pose estimation and it's prefered solutions.

The final product was able to produce reliable results for usage in many VR applications, but the existence of a 'blind spot' that the model has needs to be handled and resolved in a future work in order for the product to be truly a replacement for a joystick controller. The limited data received by a single camera setup introduces inaccuracy which could be more easily addressed using a more elaborate setup of multiple cameras.

We do believe that we've developed a good proof-of-concept for a product and hope it can be put to use somehow and are looking forward to expanding on it in further work and research in the future.

21. Propositions for future work

It is possible to further expand the work in the following ways:

1. Although a single camera setup is fairly easy to deploy, it has strong downsides, up to a fourth of the region of control can have large inaccuracies due to the interoperability of the handLandMark detection model. Using an additional camera with this method would improve the accuracy considerably.

2. Gesture controls, positions of the fingers of the hand, or gestures performed while controlling the "joystick", could be used to derive additional input.

3. Due to the covid crysis we lost access to appropriate hardware to test alternative solutions in machine learning models and camera setup. Which could result in more accurate results, and faster frame rate.

22. Bibliography

HOnnotate: A method for 3D Annotation of Hand and Object Poses

Embodied Hands: Modeling and Capturing Hands and Bodies Together

3D Hand Shape and Pose Estimation from a Single RGB Image

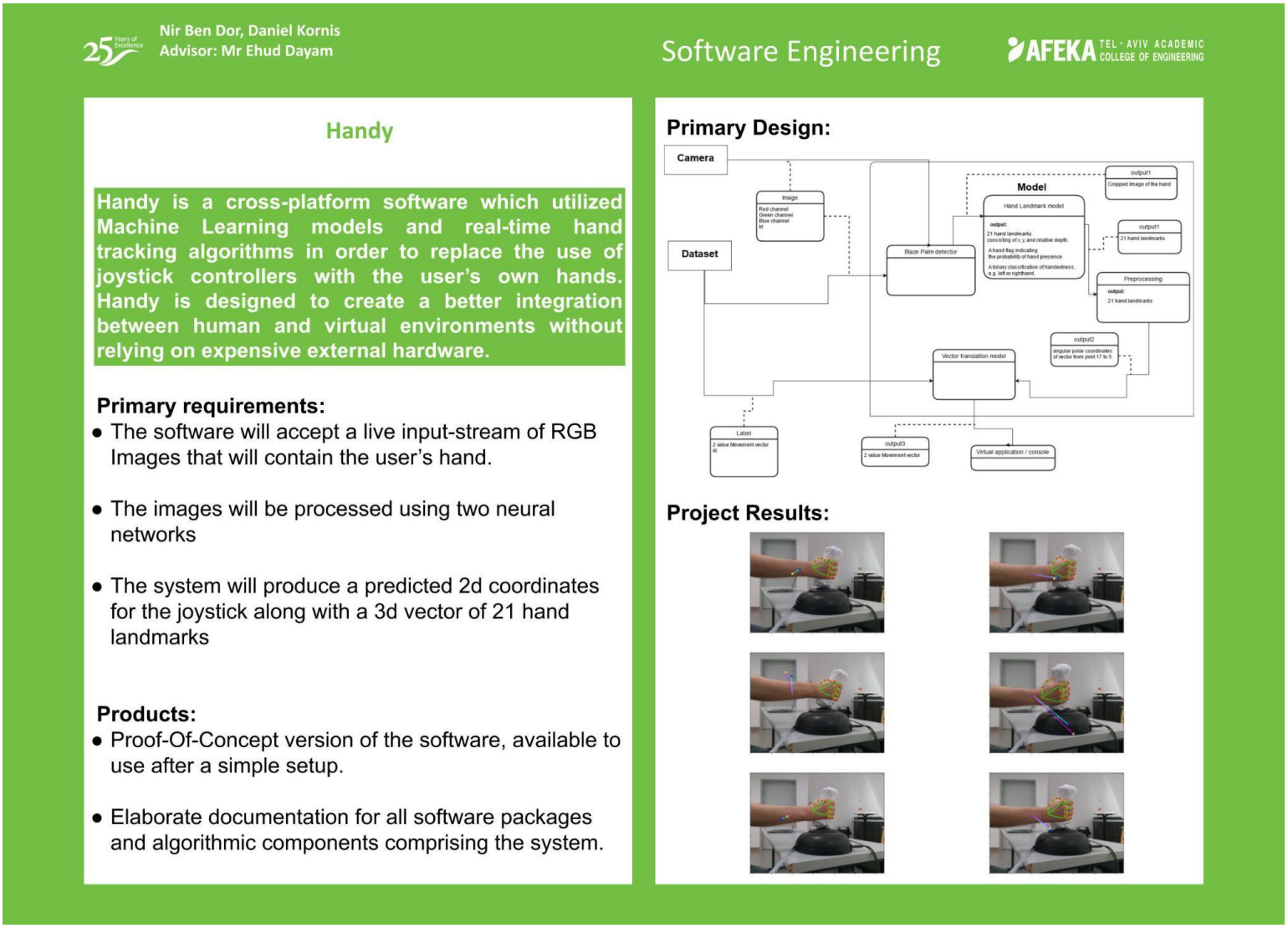
Powell's dog leg method

[CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters]

[MediaPipe Hands: On-device Real-time Hand Tracking]

23. Appendices A

23.1. Project Poster



23.2. Article

MediaPipe Hands: On-device Real-time Hand Tracking

Fan Zhang Valentin Bazarevsky Andrey Vakunov
Andrei Tkachenka George Sung Chuo-Ling Chang Matthias Grundmann
Google Research
1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA
{zhafang, valik, vakunov, atkach, gsung, chuoling, grundman}@google.com

Abstract

We present a real-time on-device hand tracking solution that predicts a hand skeleton of a human from a single RGB camera for AR/VR applications. Our pipeline consists of two models: 1) a palm detector, that is providing a bounding box of a hand to, 2) a hand landmark model, that is predicting the hand skeleton. It is implemented via MediaPipe[12], a framework for building cross-platform ML solutions. The proposed model and pipeline architecture demonstrate real-time inference speed on mobile GPUs with high prediction quality. MediaPipe Hands is open sourced at https://mediapipe.dev.

24. Appendices B

24.1. Appendix B.1 - Software Requirements Document

1. **Introduction**:
   Human-Computer interface has a large influence on productivity and ease of use of today's tools and computers, and in recent years there were several attempts to expand new technologies to fields that require different types of handling, such as Virtual Reality gear, or heavy machinery. Equipment for this today is often clumsy and accompanied by surrounding hardware which

makes it more expensive than it could be. The purpose of this project is to create a working solution for interaction between people and computers. Our project aims to provide an intuitive and relatively cheap alternative that allows precise movement in real time. It harnesses Machine-learning architecture to extract hand and joint poses from a live input stream of RGB images then map them to a joystick coordinate output. This allows the user to emulate joystick input to the system by tilting his/her hand in front of the camera.

2. **Module Specifications:**

Our system is comprised of several classical Machine Learning modules in addition to a physical camera:

- **RGB Camera -** produces an input stream of images in order to test the trained model and predict the joystick coordinates in real time.

- **Dataset** - composed of RGB images of the user's hand while it's holding a joystick in different poses and angles, in addition to matching joystick coordinates Labels.

- **Palm detection model -** A model to detect the presence of a hand in the image, and it's relative coordinates within it via an oriented hand bounding box

- **A hand landmark -** model that operates on the cropped hand bounding box provided by the palm detector and returns high-fidelity 2.5D landmarks (2d coordinates and relative depth).

- **Joints to Vector Translation model -** receives the 3D hand pose and estimates from it the Joystick x,y coordinates. The estimation is done by implementing regression in the form of a resnet model with a MSE loss function.

3. System Requirements:

Functional Requirements

The software will accept input of RGB Images that will be processed in a neural network and output three dimensional coordinates for hand and finger positions. This project will require a small-medium sized dataset of RGB images with labels for the coordinates of the joystick.

Options available to the user would possibly include:

- Different types of preprocessing that may improve detection against different backgrounds.
- Sensitivity settings for the coordinates extracted

Non- Functional Requirements

- The predictions of the trained model will be displayed using a visualization tool which won't require any prior knowledge in a specific field.
- The data will be entered in a consistent format and resolution
- User friendly, Simple and intuitive API.

4. Flow Diagram - Use Cases

Identifying the Actors and Stakeholders:

The Clients - Actor - will be able to interact with virtual objects by moving their own hands while using only a single camera and no additional hardware (apart from a computing unit).

Medical Facilities - Stakeholder - will be able to install the software in operating rooms and transfer the very subtle and precise hand movements of their surgeons to robotic systems.
It can be proved vital in situations where the patient himself needs to be under strict quarantine and direct contact between him and the surgeon is not possible.

Video Game Consumers - Actors, people who do not own a joystick but have a camera. The camera could be used to play games supporting joystick input with our software as a substitution.

Video Game Developers - Stakeholder - will be able to expand the field of Virtual Reality based games by implementing ways to further interact with the environment and virtual objects inside the game.
With the new system the player will be able to make subtler and more precise moves without the need to gears or joysticks which could improve the accessibility of VR games to a much bigger market segment.

The Main Processes / Functions of the project:

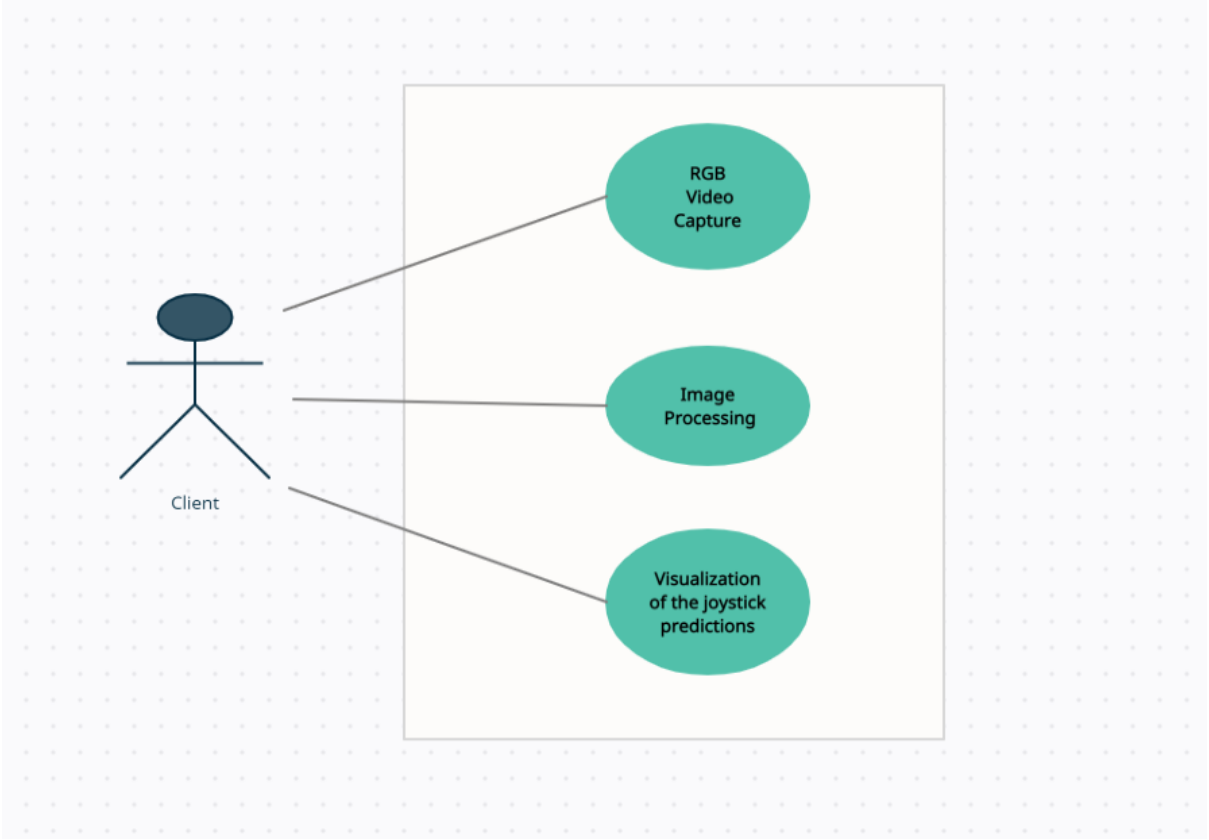


Diagram SRD Use Cases

RGB Video Capture - The external camera sends a live input stream of RGB images of the client's hand to the software which captures the frame for further work.

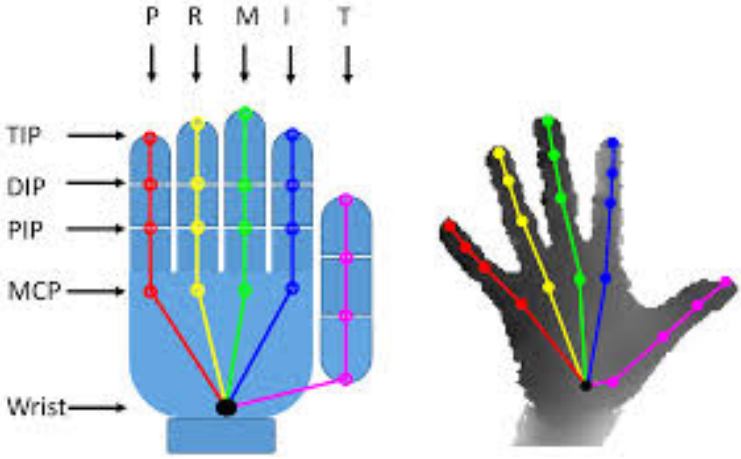
Image Processing - each frame is preprocessed with several computer vision modules and then is being transferred to the neural networks.

Visualization of the joystick predictions - the 2D coordinates output of each image is then visualized to the client on a window display alongside the 21 hand landmarks that were extracted from the frame.

5. Algorithms:

Image SRD Pose Estimation

Pose Estimation:
Single image hand pose estimation is a very popular problem in computer vision, and approaches can be divided into discriminative and generative methods. Discriminative approaches directly predict the joint locations from RGB



or RGB-D images.

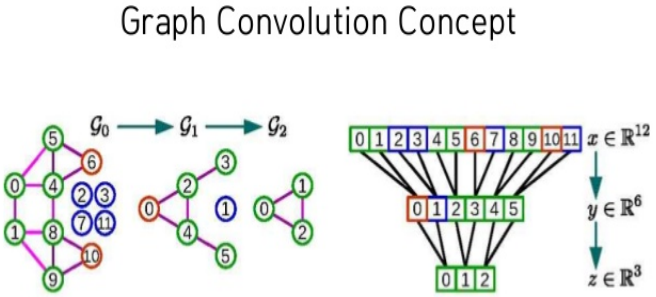
Recent works based on deep networks show remarkable performance, compared to early works based on random forests. However, discriminative methods perform poorly in case of partial occlusion.

Generative approaches take advantage of a hand model and its kinematic structure to generate hand pose hypotheses that are physically plausible, predict 2D joint locations and then lift them to 3D.
Generative approaches are usually accurate and can be made robust to partial occlusions. They typically rely on some pose prior, which may require manual initialization or result in drift when tracking.

This model is related to both discriminative and generative approaches: we use a generative approach within a global optimization framework to generate the pose annotations, and use a discriminative method to initialize this complex optimization. This model is trained using a discriminative method to predict the hand poses which are robust to occlusions from interacting objects.

Graph CNNs for Mesh and Pose Estimation:

Image SRD Graph CNN



A 3D mesh can be represented by an undirected graph $M = (V, \epsilon, W), where\ V = \{v_i\}_{i=1}^{N}$ is a set of N vertices in the mesh, $\epsilon = \{e_i\}_{i=1}^{E}$ is a set of E edges in the mesh, $W = (w_{ij})_{NxN}$ is the adjacency matrix, where $w_{ij} = 0$ if $(i, j) \notin \epsilon$ E, and $w_{ij} = 1$ if $(i, j) \in \epsilon$. The normalized graph Laplacian is computed as $L = I_N - D^{-1/2} W D^{-1/2}$, where $D = diag(\Sigma j\ w_{ij})$ is the diagonal degree matrix, I_N is the identity matrix.

Here, we assume that the topology of the triangular mesh is fixed and is predefined by the hand mesh model, i.e., the adjacency matrix W and the graph Laplacian L of the graph M are fixed during training and Testing. Given a signal

$f = (f_1, \cdots, f_N)^T \in R^{NxF}$ on the vertices of graph M, it represents F-dim features of N vertices in the 3D mesh. In Chebyshev Spectral Graph CNN, the graph convolutional operation on a graph signal $f_{in} \in R^{NxFin}$ is defined as

$f_{out} = \sum_{K=0}^{K-1} T_k(\bar{L}) * f_{in} * \Theta_k$, where

$Tk(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ is the Chebyshev polynomial of degree k,

$T_0 = 1, T_1 = x; \tilde{L} \in R^{N \times N}$ is the rescaled Laplacian, $\tilde{L} = \frac{2L}{\lambda_{MAX}} - I_N$, λ_{max} is the maxi-mum eigenvalue of $L; \theta k \in R^{Fin \times Fout}$ are the trainable parameters in the graph convolutional layer; $f_{out} \in R^{NxF_{out}}$ is the output graph signal. This operation is K-localized since Eq. 1 is a K-order polynomial of the graph Laplacian, and it only affects theK-hop neighbors of each central node.

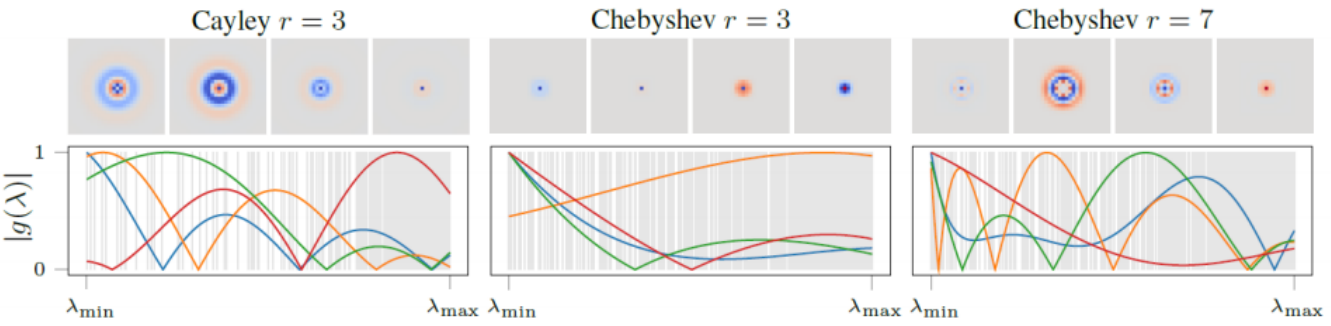


Image SRD Chebyshev

Regression Model:

Regression is a form of statistical analysis in which one finds the line that most closely fits the data according to a specific mathematical criterion. The model used in this project is a neural net model with an activation function of PReLU.

Activation Function:

an activation function is a function that is applied to a neuron and allows non linear functions in the neural network.

PReLU:

The ReLU is the most commonly used activation function in deep learning models.

The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back. So it can be written as: $f(x) = max(0, x)$
PRelu is an evolution of this activation function, that can be written as $f(x) = max(t, x)$ where t is a learnt parameter. From our experiments this activation function has allowed a more stable convergence in training than the commonly used alternatives.

24.2. Appendix B.2 - Software Design Description

1. **Introduction:**

    1.1. Hand Pose Estimation Model - Preprocessing, feature extraction and training.
    1.2. Camera - Trained Model Interface through live feeding of the images.
    1.3. Model for Joystick Coordinates Estimation - Regression model inorder to get the X and Y coordinates of the joystick.
    1.4. Interface between the two models - which takes the output from the Hand Model and transfers it as input for the Coordination Model.

2. **Module Specifications:**
    Our system is comprised of several classical Machine Learning modules in addition to a physical camera:

    - **RGB Camera -** produces an input stream of images in order to test the trained model and predict the joystick coordinates in real time.

    - **Dataset** - composed of RGB images of the user's hand while it's holding a joystick in different poses and angles, in addition to matching joystick coordinates Labels.

    - **Palm detection model -** A model to detect the presence of a hand in the image, and it's relative coordinates within it via an oriented hand bounding box

    - **A hand landmark -** model that operates on the cropped hand bounding box provided by the palm detector and returns high-fidelity 2.5D landmarks (2d coordinates and relative depth).

    - **Joints to Vector Translation model -** receives the 3D hand pose and estimates from it the Joystick x,y coordinates. The estimation is done by implementing regression in the form of a resnet model with a MSE loss function.

3. **Interfaces**:

- **Camera Interface**
  Specifications: Intel® RealSense™ Depth Camera D415
  Location: External interface which captures the user's hands as an image and transfers it to the Handy system.
  Timing: Each passing second with a set number of frames (i.e 30 frames per second)

- **Hand-model joystick-estimator interface**
  Specifications: connects the hand and joints pose estimation model to the joystick coordinates estimation model.
  Location: internal interface between the hand -joints pose estimation model to the joystick coordinates estimation model.
  Timing: Each time a batch passes through the first network and into the next one.

4. **Security:**
   The system is receiving a data stream using the python library cv2. Any vulnerability in the data stream of cv2 will inevitably affect the product.
   our project does not use any sensitive information tho, so there isn't a serious security risk.

24.3. Appendix B.3 - Software Test Documentation

Test Schedule

TASK NAME	START DATE	END DATE	START ON DAY*	DURATION* (WORK DAYS)	TEAM MEMBER
Dataset value testing					Nir & Daniel
Variable Type	3/15	3/29	0	15	Nir & Daniel
dataset setup validity	3/29	4/12	14	15	Nir & Daniel
Model credibility testing					Nir & Daniel
Valid image range	4/12	4/19	28	8	Nir & Daniel
hedge cases of hand poses and model input	4/19	4/26	35	8	Nir & Daniel
Valid results	4/26	5/3	42	8	
Speed performace	5/3	5/17	49	15	
Alpha Testing					Nir & Daniel
Inserting invalid data	5/17	5/25	63	9	Nir & Daniel

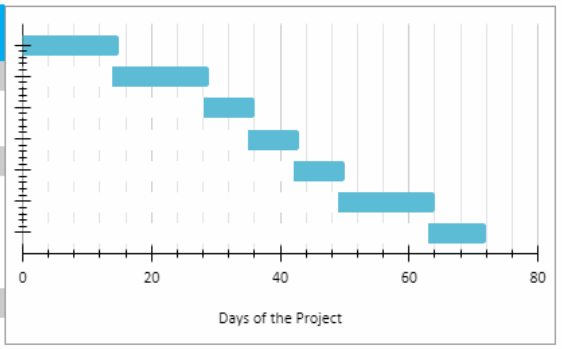


Table STD Test Schedule

In order to ensure that the product is meeting our performance expectations, we've planned a couple of tests that will check the different modules and components of the system.

The software was tested in a **variety of settings** - different hours of the day, different light sources, and with a number of users.
the variety of settings helped to decipher if the model had suffered from overfitting and to verify the model's strength and adaptability to changes.

In addition we've prepared a **boundary value analysis** and dealt with model predictions which passed the joystick boundary values.
the joystick can move in the horizontal or vertical axes in a set range of values, from $[(-1, -1), (1, 1)]$. the predictions of the model can sometimes contain values that pass that boundary. In that case we've clamped the received results between the two boundary values with a simple function so they would not interfere or alter future predictions.

in regards to the **model reliability** - we've dealt with situations when the user's hand was not present in the input image or in situations when the model did not detect the hand in the image for some reason. in order to prevent an exception in the software we simply default the position to the (0,0) coordinates if a hand is not detected.

Completeness - the product meets all of the requirements and goals we've set up for it in the beginning of the work.

Quality of the results - there is a 'blind spot' in the 1st quadrant in which the model outputs predictions with a much lower quality. in all the other regions the predictions have a significantly higher quality and the results are much closer to the actual positions of joystick

Costs:

- The system contains a small sized dataset - at around 6000 of images and appropriate labels. the space that is required from the user is very limited which does not require the use of any external hard disk in order to contain the software files or data.

- The system requires the use of a single RGB camera only.
  At the initial stages of the work we've parchaused an advanced RGB-D camera with an added depth sensor inorder to extract and gather a refined input stream for our dataset and improve the 3D capabilities of our trained model. As we've progressed further in the project we came across more sophisticated techniques and machine learning models that could use a regular RGB image and still improve on the overall results and detection accuracy. therefore the purchase of an expensive camera is no longer needed

and the overall expenses are significantly reduced.

- The system relies on the use of a joystick controller in order to gather the labels for the training stage of the model. but the final product itself uses only the user’s hands as a substitute controller for the joystick and no additional hardware is needed.

- The system relies on free libraries and softwares only (Pytorch, CV2, Media Pipe)