הטכניון - מכון טכנולוגי לישראל

הפקולטה להנדסת חשמל ע"ש אנדרו וארנה ויטרבי

המעבדה לבקרה, לרובוטיקה וללמידה חישובית

**ספר פרויקט**

**Connecting a Camera to a Prosthetic Hand**

מבצעים:

נוואף סלמאן Nawaf Salman

נאהל עוידאת Nahel Awidat

Supervisor:

קובי קוחי Koby Kohai

סמסטר רישום: אביב תשפ"ג

תאריך הגשה:

**תוכן**

[**תקציר** 2](#_Toc191742491)

[**Abstract** 3](#_Toc191742492)

[**Introduction** 4](#_Toc191742493)

**Algorithm**……………………………………………………………………………………………………………….………………………...4

[**Technologies Used** 5](#_Toc191742494)

[**Haifa 3D Robotic Hand:** 8](#_Toc191742495)

[**General Architecture:** 9](#_Toc191742496)

[**Complete Implementation:** 11](#_Toc191742497)

[**Failed Attempts:** 12](#_Toc191742498)

[**Possible Future Expansions:** 13](#_Toc191742499)

[**References** 14](#_Toc191742500)

# **תקציר**

האפשרות לשלב את היכולת לנצל את העוצמה של בינה מלאכותית ולמידת מכונה ברובוטיקה לצורך הבנת נתונים נכנסים מהסביבה, באמצעות חיישנים ומצלמות מסוגים שונים – כפי שנעשה, למשל, בסגמנטציה של תמונות וחפצים – מצביעה על כך שאנו צועדים לעבר עתיד של תלות גוברת במכשירים חכמים ומחוברים בחיי היומיום שלנו.  
מטרת הפרויקט היא לתכנן ולפתח **יד רובוטית אוטונומית** המסוגלת להסתובב ולאחוז בחפצים כדי לסייע לקטועי ידיים. זהו דוגמה מצוינת לאופן שבו רובוטיקה ובינה מלאכותית יכולים לתרום לחברה. על ידי אוטומציה של פעולת האחיזה, היד הרובוטית מפחיתה את העומס הפיזי על קטועי ידיים ומאפשרת להם לחיות ולתפקד בקלות רבה יותר ביום-יום. הפרויקט מדגים את הפוטנציאל של רובוטיקה ובינה מלאכותית לשיפור איכות החיים של אנשים עם מוגבלויות, ומדגיש את החשיבות של פיתוח והטמעה אחראיים של טכנולוגיות אלו.  
היד הרובוטית, אשר פותחה על ידי עמותת **Haifa3D**, מצוידת במצלמת **Oak-D Lite** ומופעלת על ידי **מיקרו-בקר Jetson Nano**. ה-Jetson Nano מעבד את הנתונים המתקבלים מהמצלמה ומפעיל מודל סגמנטציה לזיהוי חפצים, קביעת כיוונם ומרחקם. לאחר שהאובייקט מזוהה, ה-Jetson Nano שולח פקודות ליד הרובוטית כדי לסובב אותה לזווית מתאימה או לאחוז בחפץ.  
היד הרובוטית נועדה להחליף את היד הקטועה, והפרויקט הזה מהווה צעד משמעותי לקראת השבת תפקודיות ועצמאות לקטועי ידיים, תוך שיפור איכות חייהם באמצעות טכנולוגיית פרוסתטיקה מתקדמת.

# **Abstract**

The prospect of integrating the ability to utilize the power of AI and machine learning into robotics for understanding incoming data collected from the surrounding environment by providing them with various types of sensors and cameras, as is the case in image and object segmentation for example, means that we are heading towards a future of even greater dependency on intelligent connected devices in our daily lives.   
The aim of this project is to design and develop an autonomous robotic hand capable of rotating and grabbing objects to help hand amputees. This is an excellent example of how robotics and AI can be used to benefit society. By automating the task of grabbing objects, the robotic hand reduces the physical burden on hand amputees and allows them to live and function easier daily. This project demonstrates the potential of robotics and AI to improve the quality of life for people with disabilities and highlights the importance of responsible development and deployment of these technologies.   
The robotic hand, which was made by Haifa3D association and the Oak-D Lite camera mounted to it, are controlled by a Jetson Nano microcontroller. The Jetson Nano processes the input from the camera and utilizes a segmentation model to detect objects, their orientation and distance. Once the object is detected, the Jetson Nano sends signals to the robotic hand to either rotate to an appropriate angle or grab the object. The robotic hand is designed to substitute the amputated hand. This project is a step towards restoring functionality and independence for amputees, improving their quality of life through advanced prosthetic technology.

# **Introduction**

All the code of this project is open source and available for the general public on GitHub NahelAwi/ProjectB repository.

The project was suggested in the CRML department in the Technion. The goal is to automate a robotic 3D printed hand capable of rotating and grabbing objects to help hand amputees. While similar projects were done in the past, it had very different architecture, we approached the problem using a camera attached to the prosthetic hand.

Similar projects built previously in the faculty:

- Prosthetic Hand Control via Shoulder Movement. Done by Nitay Ozer and Itay Mal.

-Controlling a robotic arm using sensors on foot. Done by Rony Zlatokrilov and Rotem Posti.

Our project did 3 things differently:

1- Our robotic hand must be fully autonomous and there is no need for humans to give it commands its goal was predetermined.

2- The robotic hand had to interact with the real world differently since it can grab real objects on its own.

3- We supplied the robot with the ability to understand its surroundings using AI-based and non-AI-based algorithms for object segmentation, angle calculations and distance measuring.

# **Algorithm**

This section describes, in detail, the algorithm powering our autonomous robotic hand. We explain the data processing pipeline—from image acquisition to motion control—highlighting the techniques and methods we used, such as fastSAM segmentation, depth estimation filtering, and PCA-based angle calculation. In addition, we describe the communication between our two main threads, detailing how commands are synchronized and transmitted.

Our algorithm is designed to process real-time image data and determine the optimal hand movement to interact with objects. The overall process is divided into two main components:

* **calc\_angle Thread:** Processes incoming camera frames, computes the depth and segmentation masks, calculates the target angle for the hand, and decides when to trigger a grasp.
* **hand\_control Thread:** Listens for commands from the calc\_angle thread, rotates the robotic hand by the computed relative angle, and executes a closing (and subsequent reopening) action when necessary.

We integrated the fastSAM segmentation model into our pipeline to quickly and accurately isolate objects of interest from the background. In parallel, we apply filtering techniques (moving average, median filtering) to smooth noisy depth data from the Oak-D Lite camera.

**Frame Acquisition and Preprocessing**

* **Capture and Preprocess:**  
  The process begins with capturing frames from the Oak-D Lite camera. Each frame is preprocessed to remove noise using a combination of a moving average filter and median filtering. This preprocessing is essential for obtaining reliable depth measurements, especially when objects are close (e.g., around 12cm).
* **Depth Calculation:**  
  Once the frame is preprocessed, we compute the depth map using the camera’s Time-of-Flight (ToF) data. This step includes addressing challenges such as sensor noise and dynamic lighting conditions. The calculated depth is smoothed over time using a moving average to avoid abrupt changes that could lead to incorrect hand movements.

**Object Segmentation Using fastSAM**

* **fastSAM Segmentation:**  
  The preprocessed frame is then fed into the fastSAM segmentation model. fastSAM, an optimized version of the SAM (Segment Anything Model), quickly generates multiple segmentation masks corresponding to different objects and regions within the frame.
* **Mask Selection:**  
  From the set of segmentation masks, our algorithm selects the optimal mask based on criteria such as:
  + **Mask Area:** Preference is given to larger, more prominent segments.
  + **Confidence Scores:** In cases where fastSAM provides a confidence metric, the mask with the highest score is chosen.
  + **Shape Consistency:** We ensure the selected mask represents a clear object boundary, minimizing false detections.

**Angle Calculation via PCA**

* **Principal Component Analysis (PCA):**  
  With the optimal segmentation mask identified, we extract the coordinates of the mask’s contour. PCA is then applied to these points to determine the principal axis—the line along which the object’s shape is elongated. The orientation of this axis gives us the target angle for the hand to align with.
* **Relative Angle Determination:**  
  The calculated angle is compared with the current orientation of the robotic hand. To prevent excessive adjustments, a threshold margin (e.g., 10°) is set. If the change in angle is below this margin, the algorithm sends a command of 0 (i.e., no adjustment). Otherwise, it computes the relative angle (difference) and sends that value.
* **Close-Proximity Trigger:**  
  If the measured depth indicates that the object is within 12cm (or below the defined threshold), the calc\_angle thread issues a command for the hand\_control thread to initiate the grasping (closing) sequence. In this case, the calc\_angle thread will block further angle updates until the hand\_control thread signals that the grasping operation is complete.

**Inter-thread Communication Details**

* **Queue Mechanism:**  
  The calc\_angle thread enqueues commands (rotation or grasp signals) that are read by the hand\_control thread. This queue is implemented using Python’s queue.Queue, ensuring that message ordering and thread safety are maintained.
* **Blocking and Unblocking:**  
  When the depth threshold is met (object is close enough), the calc\_angle thread sends a “GRASP” command and then enters a blocked state. The hand\_control thread, upon receiving the command, executes the grasp sequence (closing and then opening the hand) and sends an acknowledgment back. This acknowledgment unblocks the calc\_angle thread so that it can resume processing new frames.
* **Margin for Angle Adjustments:**  
  To prevent unnecessary movements, the calc\_angle thread compares the newly computed angle with the current hand orientation. If the difference is less than a predefined margin (e.g., 10°), the thread sends a zero adjustment, ensuring that only significant orientation changes prompt motor actions.

# **Technologies Used**

**NVIDIA Jetson Nano Developer Kit:**The NVIDIA Jetson Nano Developer Kit is a compact single-board computer designed for AI and robotics applications. It features an NVIDIA Maxwell GPU and is capable of running multiple neural networks in parallel. The developer kit is a low-cost platform for developing and testing AI projects, and is suitable for makers, students and hobbyists. It is based on NVIDIA Jetson Nano System-in-Module and its GPU includes 128 CUDA cores. Also includes a quad-core ARM A57 CPU, 4 GB LPDDR4 memory, Gigabit Ethernet, USB 3.0 and HDMI.

In addition to its AI capabilities, the Jetson Nano Developer Kit is also equipped with GPIO and UART interfaces that make it suitable for a wide range of robotic projects.

A close-up of a computer chip

AI-generated content may be incorrect.

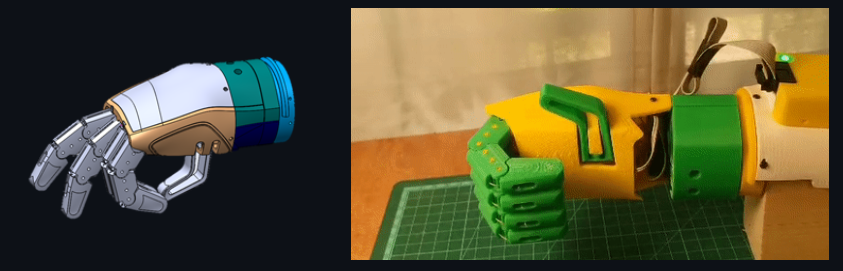
In the previous picture we can see the Nvidia minicomputer (Jetson Nano). The jack on the lower left side of the board is the power supply input: 5V DC power with at least 4 amps is needed to smoothly power the device.

**Oak-d lite Camera:**We used the oak-d Lite camera. The Oak-D Lite camera is a small depth-sensing camera designed for computer vision applications. It is built by the company Luxonis, which specializes in creating computer vision tools and hardware. The Oak-D Lite camera features an Intel Movidius Myriad X VPU (Visual Processing Unit) and a Sony DepthSense IMX556PLR back-illuminated ToF (Time of Flight) depth sensor. These components enable the camera to capture depth information and process it in real-time, making it ideal for applications such as object detection, tracking, and recognition. The Oak-D Lite camera also includes a variety of software tools and libraries to facilitate development, including the Luxonis DepthAI Python API, which allows developers to easily integrate the camera into their projects and applications. Additionally, the camera is designed to be compatible with popular machine learning frameworks, such as TensorFlow and PyTorch, enabling developers to create custom computer vision models and deploy them on the device. Overall, the Oak-D Lite camera is a powerful and versatile tool for computer vision applications, with a compact and lightweight design that makes it suitable for use in a wide range of settings and environments.

A black rectangular object with a black rectangular object

AI-generated content may be incorrect.

# **Haifa 3D Robotic Hand:**

The Oded Hand can perform a variety of user-defined hand configurations using an autonomous adaptive grip along with a variety of grip force levels. The mechanical design consists of four fingers with four different tendon-driven mechanisms actuated by four different micro-DC motors, a passive thumb with two discrete positions using two magnets and wrist rotation driven by planetary gears and a DC motor. Each finger is an independent module, hence from a developer point-of-view, you can check new finger designs and change the number of fingers in your hand model. For more details, see the <https://github.com/Haifa3D/hand-mechanical-design>.  


# **General Architecture:**

The project consists mainly of 3 parts:

* 1. • NVIDIA Jetson Nano developer kit
     + 1. Since the Jetson Nano does not include a Wireless Bluetooth, we added a USB Bluetooth dongle.
  2. • A camera we used the oak-d lite camera.
  3. • A mechanical Hand (attached to it is the camera).

Let’s delve a bit more into the general architecture. Further details about the technologies and implementation will be discussed in later parts of the booklet.

**NVIDIA Jetson Nano Developer Kit**

Jetson Nano is a powerful small computer used for building embedded applications. In our project, it was used as “the brain” of the robotic hand. On this small computer, an object segmentation model and a controller are running. The object segmentation model is used to detect objects in the frame that the camera captured and measured its distance from the hand (abiding by certain thresholds). The controller is used to control the robotic hand which is responsible for rotating and grabbing.

A camera is connected to the Jetson Nano from which the running controller gets further information about the surrounding environment.

**Oak-d lite Camera:**

using its depth estimation feature given by the oak-d Lite camera and the library depth-ai, we measured the distance from the center of the frame to the camera in order to determine need for rotation/aligning and grabbing. It’s connected to the jetson nano via a (USB-A -> USB-C) cable.

**Robotic hand:**

We used the Haifa 3D Bluetooth operated hand, following the manual on the movements of the fingers and how to activate each engine which is found on their Github, after the camera determines a distance from the center of the frame as close enough, we activate our segmentation model on the jetson nano, which then sends commands to the hand via Bluetooth, either to rotate to a certain angle or to grab or to remain idle.

**Overall:**

Frame Acquisition & Preprocessing (Noise Filtering)

Camera

Depth Caculation (Moving average, Median filtering)

fastSAM Segmntation (Edge detection & objet Mask Generation

Optimal Mask Selection (Based on Area, Confidence, shape)

Angel Calculation (PCA on Mask Points)

Command Queue:  
Send Angle/Grasp Signal to Hand.

# **Complete Implementation:**

All the code for this project is open source and available on the GitHub repository [NahelAwi/ProjectB](https://github.com/NahelAwi/ProjectB).

A hand holding a device

AI-generated content may be incorrect.A hand holding a device with orange and green connectors

AI-generated content may be incorrect.

**Measuring Distance**

The implementation begins with capturing image frames from the Oak-D Lite camera. Each frame undergoes preprocessing to reduce sensor noise using a combination of a moving average filter and median filtering. This filtering is crucial for obtaining a reliable depth map from the camera’s Time-of-Flight (ToF) sensor data. The depth information, which is continuously smoothed over time, allows the system to accurately determine whether an object is within the critical 12cm threshold needed to initiate a grasping action or within the 18cm threshold to initiate alignment action.

**Object Segmentation**

Once the depth map is computed, the preprocessed frame is passed into the fastSAM segmentation model. The segmentation process involves two key steps:

* **Edge Detection:**  
  The system initially identifies the boundaries within the image to locate potential objects.
* **Mask Generation and Selection:**  
  The fastSAM model quickly produces several candidate segmentation masks. The algorithm then selects the optimal mask based on:
  + **Mask Area:** Larger, more prominent segments are favored.
  + **Confidence Scores:** When available, the mask with the highest confidence is chosen.
  + **Shape Consistency:** The selected mask must accurately delineate the object’s contour to avoid false detections.

This optimal segmentation mask serves as the basis for determining the object’s orientation.

**Angle Calculation via PCA**

With the best segmentation mask identified, the next step is to calculate the target angle for the hand:

* **Contour Extraction and PCA:**  
  The contour points of the optimal mask are extracted, and Principal Component Analysis (PCA) is applied to determine the principal axis along which the object is oriented.
* **Relative Angle Determination:**  
  The computed orientation is compared with the current angle of the robotic hand. To prevent unnecessary adjustments, a threshold margin (e.g., 10°) is applied. If the difference is below this margin, no rotation command is issued; otherwise, the system computes the relative angle and sends the appropriate adjustment command.

**Controlling the Robotic Hand with the Jetson Nano**

The Jetson Nano serves as the control hub for the robotic hand. Based on the computed relative angle:

* **Rotation Commands:**  
  If the object’s orientation requires adjustment, the Jetson Nano sends a rotation command via Bluetooth to reposition the hand.
* **Grasping Command:**  
  When the depth measurement confirms that an object is within the 12cm threshold, a “GRASP” command is issued to close the hand. After the grasping action, the hand reopens to resume monitoring for new commands.

**Integration via Multithreading**

To achieve real-time performance, the project leverages Python’s threading library by dividing tasks into two primary threads:

* **calc\_angle Thread:**
  + Continuously captures and preprocesses frames.
  + Computes depth maps and runs the fastSAM segmentation model.
  + Calculates the object’s orientation via PCA and determines if a rotation or grasp command is needed.
  + If the object is within the grasping threshold, this thread sends a “GRASP” command and then enters a blocked state.
* **hand\_control Thread:**
  + Monitors a command queue for instructions from the calc\_angle thread.
  + Executes received rotation commands by adjusting the hand’s orientation accordingly.
  + When a grasp command is received, it triggers the closing sequence of the hand and, once complete, sends an acknowledgment to unblock the calc\_angle thread.

Inter-thread communication is managed through Python’s queue.Queue, ensuring that commands are processed in order and that both threads operate safely and concurrently. This design allows for continuous, real-time control where image processing and mechanical actuation occur in parallel.

This comprehensive implementation integrates real-time image acquisition, depth measurement, segmentation, angle computation, and synchronized control of the robotic hand, ensuring robust autonomous operation in assisting hand amputees.

# **Failed Attempts:**

**The Built-in Neural Network in the Oak-d Lite:**

The Oak-d Lite Camera comes with a built-in neural network model to identify objects, using the python library depth-ai

Although the network works well on its own, when running it to try and detect objects we need the object to be predefined in the models training, which doesn’t work well with us since we are searching for any object and only care about its shape and not its label, the code isn’t open source so we couldn’t modify it to our liking.

In addition to that the camera comes with depth estimation, which is achieved using stereo vision, which involves capturing two images of a scene from slightly different perspectives and then using computer vision algorithms to calculate the depth information from the disparity between the two images. This allows the camera to create a 3D representation of the scene. At first, we got to precise measurement of as close as 18cm, which was considered too far for grabbing objects, this problem was later resolved by changing some of the camera settings.

# **Possible Future Expansions:**

**Enhanced depth detection:**Improving depth accuracy is crucial for better object interaction. Future iterations could incorporate more advanced depth estimation techniques, such as infrared sensors.

**Upgrading the project to consider the object 3D dimension:**Currently, the system relies on 2D segmentation and depth estimation. Expanding to full 3D object analysis would enhance object grasping**.**

**A more versatile arm:**Enhancing the robotic hand to handle a wider variety of objects and tasks would significantly improve usability. Possible upgrades include:

* Additional Degrees of Freedom: Implementing wrist and finger articulation for more complex grasping actions.
* Grip Pressure Modulation: Allowing the hand to adjust grip strength based on object material and size.

**A more powerful GPU for enhanced performance:**Processing power limitations can restrict real-time AI operations. Upgrading to a more powerful GPU would allow:

* Faster Object Segmentation: Running more complex AI models for improved detection accuracy without lag.
* Higher Resolution Processing: Enabling more detailed depth and segmentation analysis for better precision.
* Multitasking Capabilities: Running multiple neural networks simultaneously for more sophisticated real-time decision-making.

# **References**

TODOs:  
1- add a BIG section for the “Algorithm” (3-4 pages)  
 - introduction to the algorithm, what we used (fastsam,…)  
 - very detailed explanation of how the algorithm works (suggestion below).  
 - add flow diagram and sequence diagram of the algorithm  
2- fix the high level flow diagram (Camera -> Jetson NANO -> hand). Current one is correct but missing details.  
3- short explanation about the 2 threads communication – queues/block/unblock/…  
  
  
suggestion:   
separate explanation for calc\_angle thread and hand\_control thread.  
  
calc angle:  
1- take a frame and process it (more details)  
2- calc depth (with more details: challenges (12cm), moving average, median filtering, …)  
3- fastsam segmentation (with more details)  
4- pick the optimal segmentation mask from fastsam results (explain how)  
5- calc angle (how? - PCA…)  
6- communicate the angle to the hand\_control thread (explain that we send the relative angle to the current one, explain that we have a margin (10?), if angle didn’t change more than a margin, we send 0)  
7- if depth <= 12cm (?), tell the hand\_control thread that it should close the hand and block current thread.  
8- …  
  
hand control:  
1- receive the angle from the queue, if angle > 0, rotate hand by this angle  
2- if need to close the hand, close the hand, then open it, and tell the main thread to unblock/continue.  
3 …