

Computer Science and Mathematics
COD290
B922435

**PROJECT TITLE THAT COULD
GO OVER THREE
LINES**

by Jacob Toller

Supervisor: Dr. H. Cai

Department of Computer Science
Loughborough University

June 2023

Abstract

abstract goes here

Contents

1	Introduction	3
2	My Background	3
3	Hardware Review	3
3.1	Nvidia Jetson Nano	3
3.2	Loughborough University's In-House Lab Robots	4
4	Literature Review	6
4.1	6DRepNet	6
4.2	MTCNN	7
4.3	OpenAI & ChatGPT	8
5	Gap Analysis	10
6	Methodology	10
6.1	Language & Framework Choice	10
6.1.1	Robot Operating System (ROS2)	10
6.1.2	Python	13
6.2	My Design & Development Strategy	13
6.3	Software & Supporting Systems	13
6.3.1	VSCode	13
6.3.2	GitHub	13
6.3.3	Ubuntu	13
6.3.4	L ^A T _E X & Overleaf	13
7	Appendix	15

Disclaimer

Note that throughout this report, I may refer to 'ROS2' simply as 'ROS'. Robot Operating System 1 (ROS1) has no relevance in this project or report.

All images and diagrams in this document are my own; unless specified otherwise.

1 Introduction

2 My Background

3 Hardware Review

3.1 Nvidia Jetson Nano



Figure 1: Image of the NVIDIA Jetson Nano Developer Kit.

Sitting at the heart of this project is the Jetson Nano. It is an ARM-architecture, low-power microcomputer originally released by NVIDIA in 2019 (NVIDIA Newsroom 2019). Compared to its competitors; such as the beloved Raspberry Pi and its derivatives or the Arduino family of products; the Jetson is suited to my project as it has the following advantages:

- Enough processing power and memory to support the additional overhead of a 'normal', non-specialised operating system.

- Dedicated GPU. The utilisation of a dedicated GPU is important for applying machine-learning based models; as it allows for significantly faster processing than what would otherwise be possible on just a CPU, especially that of an ARM-based single-board computer.
- On-board WiFi chip & antenna. This allows for wireless connectivity to the robot.

However, the Jetson Nano does not come without its downsides; which have to be worked around:

- Due to lack of support from NVIDIA regarding the Nano itself; the latest version of CUDA that can be used is 10.2; and must be used in a Ubuntu 18.04 environment (NVIDIA Developer 2022).
- ARM-based CPU architecture means that we have to source libraries & software that have either been distributed with an ARM build; or build one from the source code.

3.2 Loughborough University's In-House Lab Robots

If the Jetson Nano is the heart of this project; then the Lab Robots are the body of it.

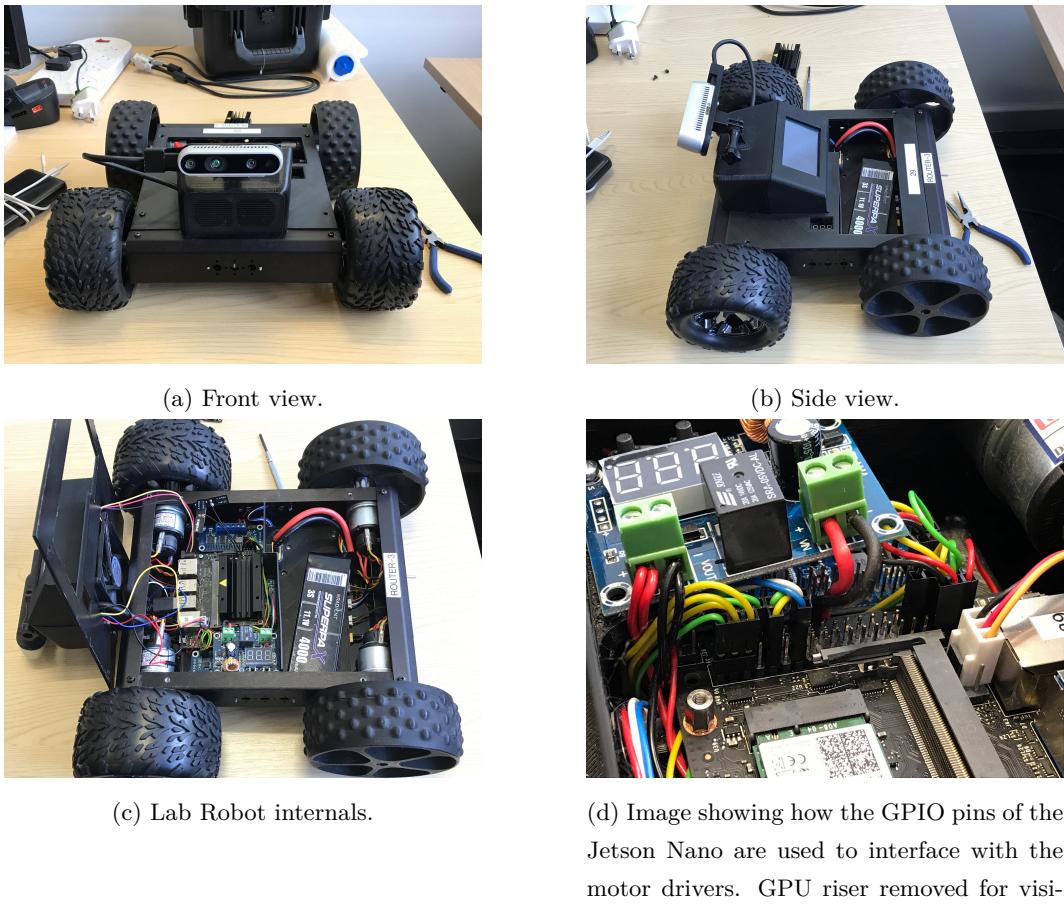


Figure 2: Images of the Lab Robot and its components.

As pictured in Figure ; the Lab Robots use a four-wheeled configuration, where all four wheels are driven by individually addressable, geared, reversible motors. This allows the robot to easily drive forwards and backwards; and can steer using a skid-steer configuration¹. The robot is powered by a 11.1V, 4000mAh battery; which is used to power both the Jetson Nano and the two motor drivers. This is all enclosed in an aluminium and acrylic casing, which protects the internal components. Atop the body of the robot is a raised compartment which houses a fan to provide airflow to the enclosure; and a mounting for the Intel RealSense D435i camera system. The fan is important as while the passive heat sink of the Jetson Nano is enough to cool the board when exposed; it needs forced airflow when enclosed.

The Intel RealSense D435i is an all-in-one sensing module; that has the following features

¹For example; for the robot to turn left, the wheels on the right hand side of the robot can be driven forwards, and the wheels on the left hand side driven backwards.

(Intel RealSense 2019):

- Connectivity via USB-C
- Standard RGB camera; similar to that of a normal webcam. It is able to produce a maximum image size of 1920x1080, at a maximum rate of 30 frames per second.
- Depth sensing; via a stereoscopic pair of cameras. It is able to produce a depth image with a 1280x720 resolution at up to 90 fps.
- Internal IMU (Inertial Measurement Unit) to allow for translational and rotational movement sensing.

Most of these features will go unused in this project; as we will mostly be using the RealSense for the fact that it is what is supplied with the Lab Robots; although they do offer nice opportunities that could be explored in the future.

4 Literature Review

4.1 6DRepNet

One of the sub-systems that I needed for this project was to have a way of determining whether or not a person was looking at the robot. The idea being that someone who would be looking at the robot is paying attention to it and wants to interact.

In the paper “6d Rotation Representation For Unconstrained Head Pose Estimation”, (Hempel, Abdelrahman, and Al-Hamadi 2022a); the authors propose an unconstrained, landmark-free approach, ‘6DRepNet’, to the task of head pose estimation; where ‘unconstrained’ means that it should be able to estimate the direction that a person’s head is facing from any reasonable direction, not just a narrow angle at the front. The two main approaches for this task are either landmark-based, or landmark-free:

- A model/approach/implementation that uses a landmark-based approach will operate by taking the image of the head and first attempt to locate a series of landmarks on the face (this could be eyes, nose; but also abstract points on the face). These landmarks are then used to attempt to predict a facing direction by comparing it to a reference frame. The accuracy of this type of model is dependent on accurate localisation of the reference landmarks.
- A landmark-free model will attempt to resolve the facing direction without the intermediate step; estimating the facing direction straight from the input image.

Specifically, how their approach differs from other approaches is that their model predicts direction as represented by two 3D column vectors, a_1 and a_2 ; where the 6 elements that make up these vectors are predicted by the model, and is where the model gets its namesake. These vectors are then converted into a 3D rotation matrix with the following equations:

$$\begin{aligned} b_1 &= \frac{a_1}{\|a_1\|} \\ b_2 &= \frac{u_2}{\|u_2\|}, \quad u_2 = a_2 - (b_1 \dot{a}_2) b_1 \\ b_3 &= b_1 \times b_2 \end{aligned} \tag{1}$$

Finalised rotation matrix as:

$$B = \begin{pmatrix} | & | & | \\ b_1 & b_2 & b_3 \\ | & | & | \end{pmatrix} \tag{2}$$

as taken from Eq's. (2-3); (Hempel, Abdelrahman, and Al-Hamadi 2022a). Whereas a typical head-pose detection approach might just directly predict pitch, yaw and roll angles.

This system is applicable to my project if we make the assumption that 'if someone's head is facing towards the robot, then they must be looking at the robot'. My reasoning for using this method over a gaze detection model is that, in theory, it should offer more accurate prediction at range, or lower camera resolution²; as the head is a much larger target to determine direction from compared to using the direction of the eyeballs.

In terms of implementation; this paper comes with an accompanying GitHub repository (Hempel, Abdelrahman, and Al-Hamadi 2022b); containing a PyTorch implementation of the model and its weights. The disadvantage to this approach is that the model is only functional when the input image is cropped to just the face. Hence it must be paired with a method of detecting a face/faces from an input image, to generate the cropped images automatically.

4.2 MTCNN

As established in the previous section; I needed a face detector that could be executed on an image to feed bounding boxes into the head-pose detector.

The face detection model that I chose was MTCNN; as published by (Zhang et al. 2016).

²This could be due to hardware limitations of the camera, or by processing a lower resolution image for transmission/prediction.

MTCNN operates in a three stage process. In between each stage, Non-Maximum Suppression (NMS)³ is applied to the generated bounding boxes to remove repeated results.

1. The first stage takes the input image and generates a rough set of candidate boxes.
2. The second stage is used to sample and discard boxes from the first stage; and to improve the quality of the remaining boxes.
3. The third, final stage repeats the process of the 2nd stage to produce a finalised prediction. It also, ironically, detects facial landmarks; although they are not numerous enough to be usable for direct head-pose estimation.

To use in my system; I use the python implementation available from the <https://github.com/timesler/facenet-pytorch> repository, (Esler 2019). This specific implementation of the model is specifically a PyTorch implementation. My exact reasoning for this choice will be discussed in the

4.3 OpenAI & ChatGPT

Open AI is an American company which develops a variety of Artificial Intelligence models; the purpose of which can vary from 'Synthesising images from a text prompt' (Ramesh et al. 2021), to 'Defeating professional e-sports players in the game Dota 2' (Brockman et al. 2018). They also perform research into the environment surrounding AI; such as *Preparing for malicious uses of AI* (Clark, Page, and Amodei 2018). This goes to show that they have a diverse portfolio/roster of research in the field of AI and its applications.

However by far their most renowned⁴ product is ChatGPT; a Generative Pre-trained Transformer model(Brown et al. 2020) packaged as an online chat application, releasing in November of 2022 (OpenAI 2022). Its appeal comes from the fact that users can provide it with a prompt or question; and it will synthesise a reply to assist/answer the user. This is best demonstrated with an example (Figure 3):

³This is a pruning technique that is used to remove duplicated bounding boxes; where if the Intersection over Union score exceeds a set threshold, the box with the lower confidence score is discarded.

⁴in terms of academia and education, 'infamous' is probably more accurate

USER: What is the appeal of ChatGPT?

CHATGPT: The appeal of ChatGPT, and similar language models, lies in their ability to understand and generate human-like text, making them valuable tools for a wide range of applications. Here are some key aspects that contribute to the appeal of ChatGPT:

- Conversational Interaction: ChatGPT is designed to engage in natural language conversations. Users can ask questions, seek information, discuss topics, or have interactive exchanges. The model's ability to understand context and generate relevant responses makes it feel more like conversing with a human.
- Versatility: ChatGPT can handle various topics and domains, drawing upon its training on a diverse range of internet text. It can discuss science, technology, literature, entertainment, history, and more. The broad knowledge base allows for exploration of different subjects within a single platform.
- Information Retrieval: Users can obtain quick and concise information from ChatGPT. It can provide definitions, explanations, summaries, or factual details. This feature is particularly useful for those seeking answers or clarifications on specific topics.
- Creative Writing: ChatGPT can assist with generating text creatively. Whether it's brainstorming ideas, writing stories, or composing poems, ChatGPT can offer suggestions, help overcome writer's block, and act as a collaborative writing partner.
- Learning and Education: ChatGPT can serve as an educational resource, helping users grasp complex concepts or explaining topics in a simplified manner. It can also provide examples, references, or additional resources for further exploration.
- Accessibility: ChatGPT is accessible through various platforms and interfaces, such as web applications, APIs, or integrations with other software. This availability enables users to engage with the model using familiar tools and frameworks.
- Continuous Improvement: ChatGPT is a part of an ongoing research and development process. Feedback from users helps in refining the model and addressing its limitations. This iterative approach ensures that ChatGPT evolves and improves over time.

It's important to note that while ChatGPT can produce human-like responses, it's still an AI model and may occasionally provide incorrect or nonsensical answers. Care should be taken to verify information obtained from any AI-based system.

Figure 3: Example dialogue with ChatGPT. Fresh conversation instance (no history), queried 27/05/2023.

The concept of a Generative Pre-trained Transformer is that it is a model capable of generating a series of tokens⁵ from a set of input tokens; where the 'pre-trained' aspect comes from the fact that these models are trained on an expansive data-set of published media. This includes books, papers and 'formal' publications; but also other sources such as personal websites and social media.

This type of model is applicable to my project, as in concept, we could have the robot listen to what an interactee is saying to it; parse this through a GPT model to generate a response; and then have the robot speak that response aloud for the person to hear.

Regarding actually implementing this; the pre-trained model cannot be locally downloaded and used; due to its immense size and the fact that it is not available for download. Instead, OpenAI offer an API (and accompanying Python wrapper) which can be used to interact and generate results from their various models; including sending prompts to their GPT models for it to generate a reply.

5 Gap Analysis

6 Methodology

6.1 Language & Framework Choice

6.1.1 Robot Operating System (ROS2)

I needed a solution that could allow for code to be executed simultaneously ros does this as it runs nodes as their own individual process; allowing the operating system to allocate the

needed a way to transmit images from the jetson nano to a control system ros allows for seamless operation across devices given that they are on the same network little note on the domain ID here,

wanted a framework that allows me to w

note that I use ROS2 galactic on my laptop, dashing on the jetson board; these two can communicate; and will simply reduce to ROS throughout the report. One of the things that

⁵These tokens are either words, or parts of words; and is how a block of text is split up into components to be used as an input and produced as output.

I did at the start of this project was to establish a series of requirements that the backbone of the system would have to fulfil to allow for this project to succeed:

1. I needed a solution that could allow for code to be executed simultaneously. The reasoning for this is simple, as going into the project I recognised that it would have several components that need to operate at the same time; such as face recognition and
2. I wanted a solution that allowed for the code running on the Lab Robot to easily communicate with another computer. This is because I wanted a backup plan incase something didn't perform well enough/couldn't be run on the Jetson Nano. This would also open up future possibilities for robots which don't have as much processing power available to them locally.
3. I wanted something that could be modular and adaptable. For example could have a centralised control system, that isn't robot specific; and then have a 'client' or something that interfaces with the robot and its specific components. This would mean that the whole project wouldn't have to be 'ported' to a different type of robot, just the inter-facial component.

I chose to use ROS2 for this project as it offers a solution to each of the problems.

Robot Operating System 2 (ROS2), as published by (Macenski et al. 2022), is an open-source robotics framework built from the successes and shortcomings of ROS1; and operates on a system of nodes; where each individual node can be its own program, written in any language of the developer's choice⁶. These nodes can then communicate between each-other through several different communication types:

- Topics; where Nodes can publish a message to the topic (publisher Nodes); and the message will be distributed to all the subscriber nodes of that topic.
- Services; where server nodes can lie in wait until a client node sends a request to perform a task. Once completed, the server node will send a reply in the form of a response message.
- Actions; which are intended for tasks which take a long time to complete and allow for progress to be reported as it progressed. Functionally, they operated like a combination of a Service and a Topic, with the latter used to publish task feedback.

⁶By default, ROS2 comes with ROS Client Libraries for C, C++ and Python; but community RCL's allow for nodes to be written in other languages.

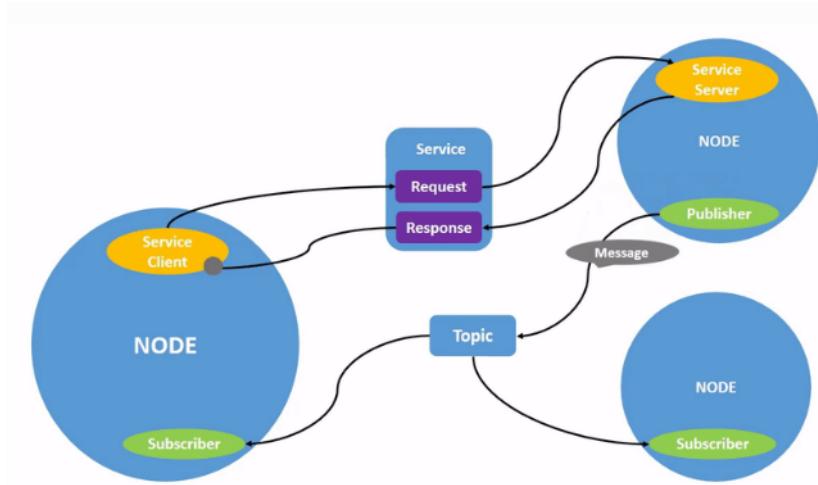


Figure 4: Diagram comparing the functionality of Services and Topics in ROS. Image from (Open Robotics 2021).

It is noted that a singular Node can make use of multiple different communication types, or even multiple of the same type. For example, Figure 4 shows how a possible node could act both as a publisher for a topic, and as a server for a service.

The node system of ROS solves two of the problems specified earlier; namely allowing for simultaneous processing and modularity. This is because each node functions as its own individual program, which allows for the Operating System of the computer it is deployed upon to manage the allocation of processing time and cores for the process. Having multiple nodes means we can establish the system across a series of 'core' nodes, and then have a node (or even multiple nodes) which is written specifically for the robot it is deployed on, which acts as the interface between the rest of the ROS system and the robot's non-standardised components.

The final issue that ROS solves is that of allowing communication between multiple devices. ROS solves this by allowing for nodes to communicate with nodes that are running on a different device on the same network; as if they were both operating on the same device. Seamlessly. This is big as it not only allows for parts of the system to perform their processing off of the robot itself; but also allows for nodes to be tested on the main system as they are developed; as the system could be running entirely on the Jetson of the robot, with the exception of the in-development node being tested and iterated from my laptop. The requirements for this to function is simply having the devices on the same network, and having the same `ROS_DOMAIN_ID`⁷.

⁷This is used to allow for two separate ROS systems to operate on the same device/devices/network

6.1.2 Python

6.2 My Design & Development Strategy

6.3 Software & Supporting Systems

6.3.1 VSCode

VSCode is my text-editor/IDE of choice, regardless of project. Its suitable here as it supports all of the filetypes of my project, with syntax highlighting; notably .py, .json, .cfg and .xml files.

6.3.2 GitHub

Git & GitHub were used extensively in order to manage the code-base of this project. I elected to manage the project across two separate repositories; with one repository used for holding all of my prototyping work and report-writing materials, and the other used to hold the collection of ROS packages. My reasoning for this is that it allowed me to keep one of the repositories purely ROS-based so that it could easily be deployed onto the Jetson Board from development on my laptop.

<https://github.com/plebble/human-robot-interaction>
(For prototypes and non-ROS code.)

<https://github.com/plebble/ros2-project>
(ROS packages for deployment on the robot.)

6.3.3 Ubuntu

6.3.4 L^AT_EX & Overleaf

In order to write and typeset this thesis; I am making use of L^AT_EX and the Overleaf editor/platform. I've worked with a local installation of the L^AT_EX environment in the past, on Windows and it was one of the most painful installation experiences in my life. Overleaf bypasses this as it is a fully comprehensive editor and compiler accessible from a browser.

References

Brockman, Greg et al. (June 2018). *OpenAI Five*. URL: <https://openai.com/research/openai-five> (visited on 05/28/2023).

without interference.

- Brown, Tom B. et al. (May 2020). *Language Models are Few-Shot Learners*. arXiv: 2005.14165 [cs.CL].
- Clark, Jack, Michael Page, and Dario Amodei (Feb. 2018). *Preparing for malicious uses of AI*. URL: <https://openai.com/research/preparing-for-malicious-uses-of-ai> (visited on 05/28/2023).
- Esler, Tim (May 2019). *facenet-pytorch*. URL: <https://github.com/timesler/facenet-pytorch> (visited on 05/27/2023).
- Hempel, Thorsten, Ahmed A. Abdelrahman, and Ayoub Al-Hamadi (Oct. 2022a). “6d Rotation Representation For Unconstrained Head Pose Estimation”. In: *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE. DOI: 10.1109/icip46576.2022.9897219. URL: <https://doi.org/10.1109%5C%2Ficip46576.2022.9897219> (visited on 05/27/2023).
- (Feb. 2022b). *6DRepNet*. URL: <https://github.com/thohemp/6DRepNet> (visited on 05/27/2023).
- Intel RealSense (Feb. 2019). *Intel RealSense Depth Camera D435i*. URL: <https://www.intelrealsense.com/depth-camera-d435i/> (visited on 05/25/2023).
- Macenski, Steven et al. (May 2022). “Robot Operating System 2: Design, architecture, and uses in the wild”. In: *Science Robotics* 7.66. DOI: 10.1126/scirobotics.abm6074. URL: <https://doi.org/10.1126%5C%2Fscirobotics.abm6074> (visited on 05/25/2023).
- NVIDIA Developer (June 2022). *JetPack SDK 4.6 Release Page*. URL: <https://developer.nvidia.com/embedded/jetpack-sdk-46> (visited on 05/25/2023).
- NVIDIA Newsroom (Mar. 2019). *NVIDIA Announces Jetson Nano: \$99 Tiny, Yet Mighty NVIDIA CUDA-X AI Computer That Runs All AI Models*. URL: <https://nvidianews.nvidia.com/news/nvidia-announces-jetson-nano-99-tiny-yet-mighty-nvidia-cuda-x-ai-computer-that-runs-all-ai-models> (visited on 05/25/2023).
- Open Robotics (May 2021). *Understanding Nodes*. URL: <https://docs.ros.org/en/galactic/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html> (visited on 05/25/2023).
- OpenAI (Nov. 2022). *ChatGPT / OpenAI*. URL: <https://chat.openai.com/> (visited on 05/28/2023).
- Ramesh, Aditya et al. (Jan. 2021). *DALL-E: Creating images from text*. URL: <https://openai.com/research/dall-e> (visited on 05/28/2023).
- Zhang, Kaipeng et al. (Oct. 2016). “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks”. In: *IEEE Signal Processing Letters* 23.10, pp. 1499–1503. DOI: 10.1109/lsp.2016.2603342. URL: <https://doi.org/10.1109%5C%2Flsp.2016.2603342> (visited on 05/27/2023).

7 Appendix