Alternative Distance Detection with Stationary Computer Vision

Jayden Chen

ABSTRACT

With more advancements in robotics, efficient communication through computer vision has become a subject of interest. Distance detection is vital for robotic systems such as self-driving cars and automatic delivery services. Existing approaches for distance detection focus on parallax or sonar with visible light, or infrared lasers, which are expensive or close ranged. Other approaches focus on highly trained neural network systems that require tremendous amounts of data and energy. In this paper, I present a method that uses two aligned cameras with object detection, creating a more accessible approach that relies less on the quality of the device and/or the trained machine learning model. Using Google's Mediapipe for computer vision and two web-cameras, this method managed to achieve high success rates within three tests. This technology was tested with ROS2 and other computer vision technology, proving it to be a good, efficient component in robotic systems. This technology could be used for smart city infrastructure or other line-like places that require detailed object locating.

Keywords

Machine Vision, Distance Ranging, Object Detection, Smart Infrastructure

INTRODUCTION

Devices with significant vision computing power are necessary for more efficient autonomous systems. Many uprising, revolutionary robotic systems absolutely depend on complex systems of computer vision and location, such as automatic delivery systems. It is vital for these models to be perfected and therefore valuable to develop powerful camera detection models. However, it is difficult to extract 3-dimensional and distance data from purely visual cameras.

Many models have been developed to extract dimensional and distance data. Computer stereo vision that uses parallax and computer vision is used massively to do this very well (Fan et al. 2012). However, it is expensive and requires both cameras to be synchronized. Laser detecting and ranging, or Lidar (Wadinger et al. 2005) is another popular method of distance tracking, which calculates the distance through the rebounded light and the time it takes for the light to rebound. This method is also expensive and needs high processing power. Other approaches use highly trained neural networks, which require excessive amounts of data, processing power, and energy (Patterson et al. 2021).

Since the techniques above are either expensive or not easily accessible, this method is not viable for street and other types of infrastructure that require prominent levels of implementation, high accessibility, and low costs. Smart cities that are optimized and automatic have been theorized to reduce threats such as urbanization and population growth (Drepaul 2020). Thus, developing accessible distance-finding technology is appealing.

In this paper, I present a method that uses two aligned cameras with object detection, creating a more accessible approach that relies less on the quality of the device and/or the trained machine learning model. The contributions of this paper are:

1. Describe an accessible approach to detect distances through computer vision and object detection (Figure 1).

2. Assess the accuracy of such an approach.

3. Prove that it is viable for high-performing robotic systems and other approaches.

4. Discuss the limitations and implementations of such method.

METHODOLOGY

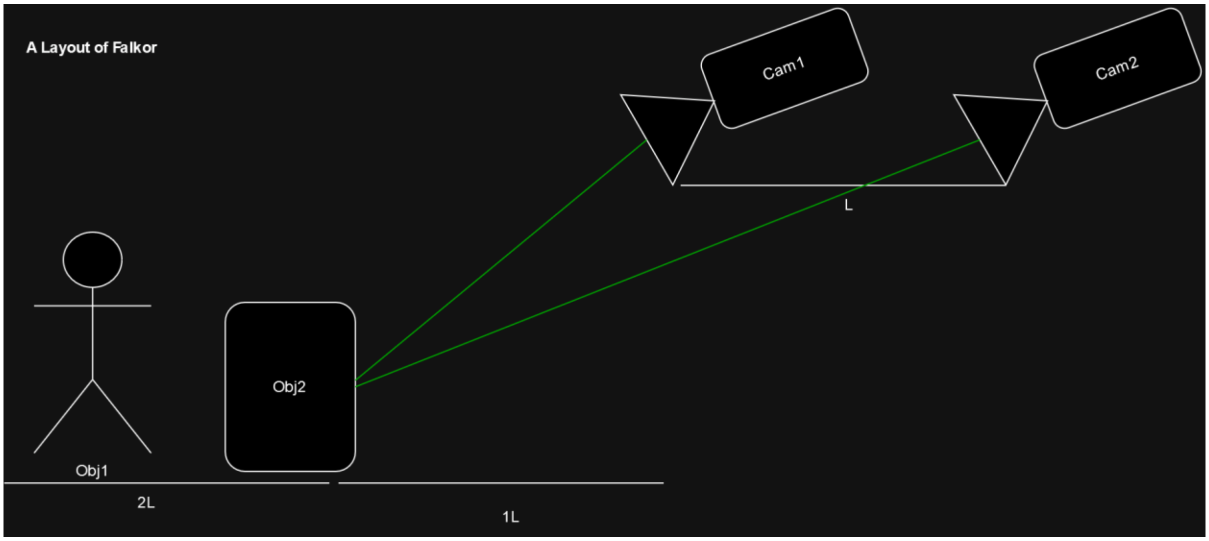


Figure 1. A visual demonstration of the approach depicted in this paper. The two cameras are facing the same direction on the same axis. ***L*** is the distance between the two cameras, which is used as a measuring unit. In the demonstration, it is detecting the distance of Object Two which is around 1***L*** away from Camera One, while ignoring Object One.

This method ranges distance through a ratio of two aligned, forward facing camera images (shown in Figure 1). For convenience in this paper, I will nickname it Falkor.

Falkor's Distance Sensing

Shown in Figure 1, **L** is the distance between Camera One and Camera Two. ***L*** will be used as a measuring unit for the distance. Both cameras input a frame into the Falkor program. With computer vision, Falkor can detect Object Two and get a bounding box of Object Two with a pixel width and height value, which is visually shown in Figure 2 (more information on this is below). Thus, we have two object detection results from Object Two. Let us define ***P1*** as the result from Camera One and ***P2*** as the result from Camera Two. The ratio of ***P2***'s height to ***P1***'s height is ***P1/P2***, which Falkor can gain the distance between Object Two and Camera One using it.

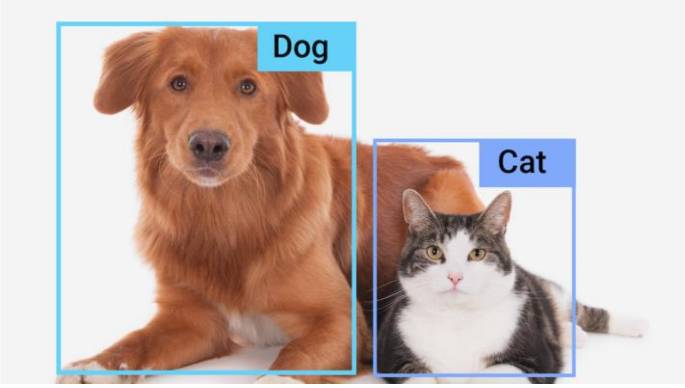


Figure 2. Image from https://developers.google.com/mediapipe/solutions/vision/object_detector#efficientdet-lite0_model_recommended. A short demonstration of mediapipe's object detection, which is used for computer vision in this paper. Mediapipe supplies a trained object detection model, which returns an object label, bounding box, and probability of success.

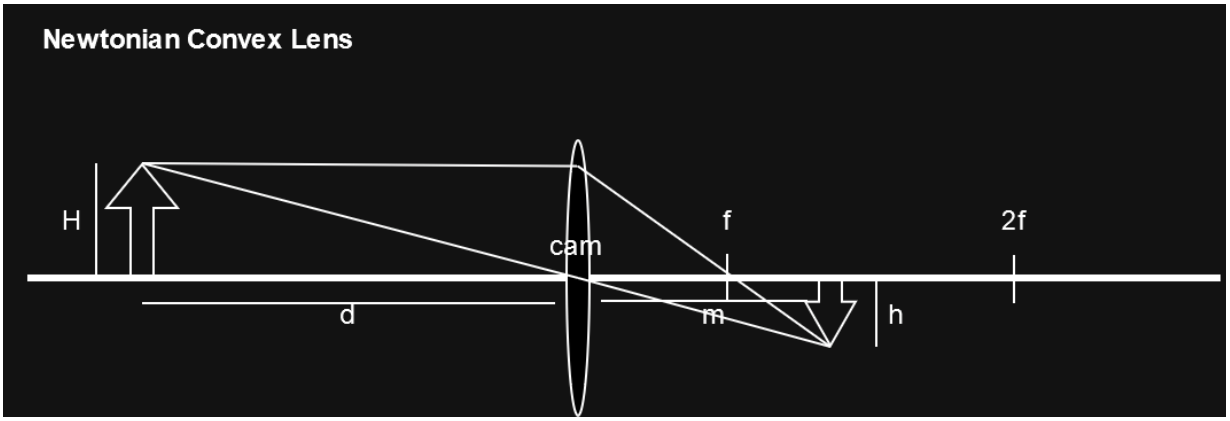


Figure 3. This (Newtonian Lens) illustrates how the ray changes direction both as it enters and as it leaves the camera's convex lens. The triangles formed by ***H, d*** and ***h, m*** are similar right triangles.

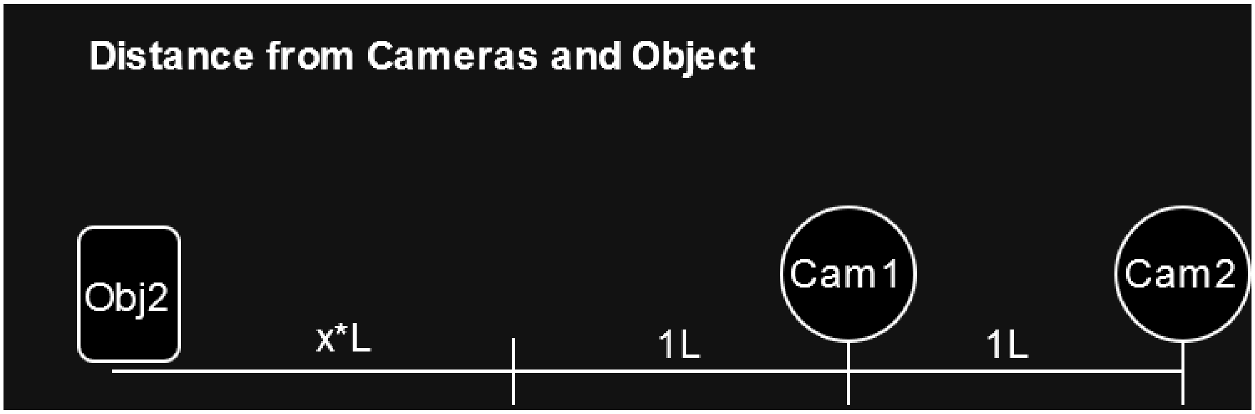


Figure 4. This shows the distance between Object Two and Camera One and Two. ***L*** is used as a distance unit so that different lengths between Cameras could be implemented.

Most cameras use convex lens for optical perception. Figure 3 shows this convex lens in a Newtonian model. The physical height of the object is ***H***, and the refracted height within the lens is ***h***. The distance between the physical object and the lens is ***d***. The distance between the refracted object and the lens is somewhere between ***f*** and ***2f*** or defined here as **m *(1f<m<2f)***. This ignores the depth of the convex lens, but it is neglectable above 1L. Using the similar triangle rule, we get the equation:

$$\frac{H}{h} = \frac{d}{m}$$

This in terms of ***h*** is:

$$h = \frac{H \cdot m}{d}$$

Assuming that the pixel height of the image, which is ***P1*** and ***P2,*** is the ***h*** times an unknown value of ***α***, the equation of the height of ***P1*** or ***P2*** is:

$$P = \frac{H \cdot m \cdot \alpha}{d}$$

Figure 4 shows the physical distance between Object Two and Camera One and Camera Two. The distance between Object Two and Camera One (d_{P1}) is ***x + 1***, where ***x*** is the distance using ***L*** as a unit; $\boldsymbol{d_{p2}}$ is ***x + 2***. Using this, assuming that the two cameras are the same device, the equation for ***P1*** to ***P2*** is:

$$\frac{P1}{P2} = \frac{\frac{H \cdot m \cdot \alpha}{x+1}}{\frac{H \cdot m \cdot a}{x+2}} = \frac{x+2}{x+1} = \frac{x+1+1}{x+1} = 1 + \frac{1}{x+1}$$

In terms of ***x + 1*** (distance of Object from camera one in ***L*** units), the distance equation is:

$$x + 1 = \frac{1}{\frac{P1}{P2} - 1}$$

Evaluation of Falkor

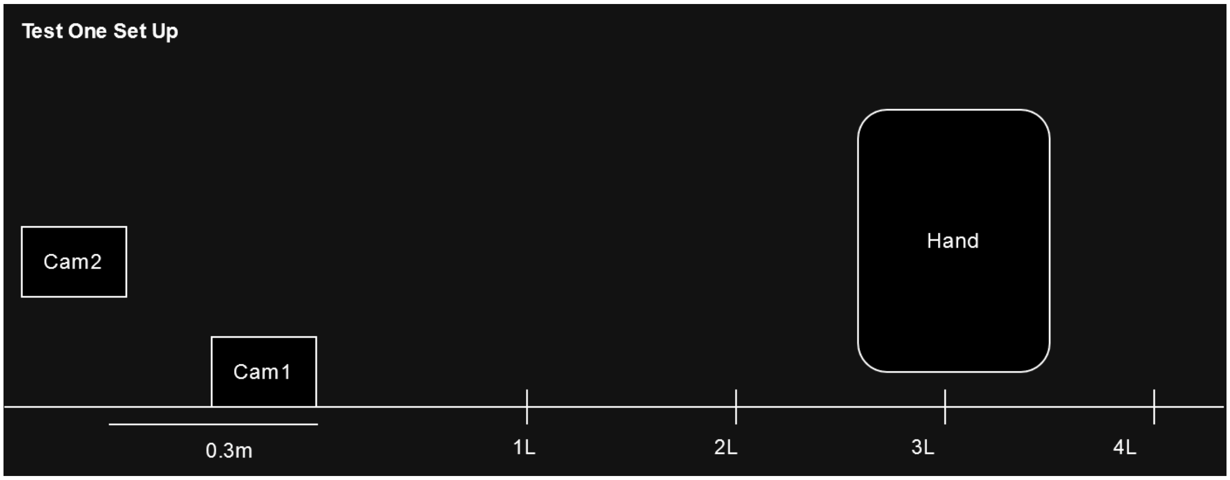
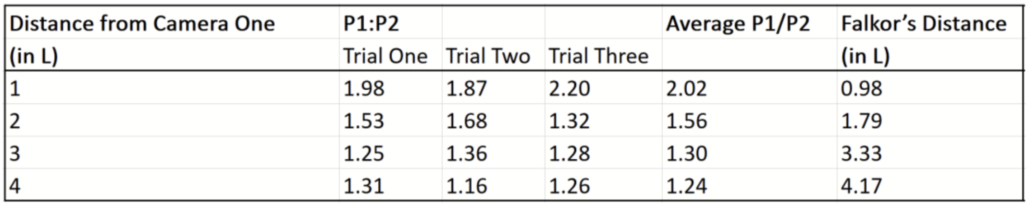


Figure 5. This shows the set up for Test One. The cameras were set 0.3 meters or 30 centimeters apart. The hand was put on intervals 1, 2, 3, 4***L***.

Small data was gathered to support this claim. The set up is shown in Figure 5, where two Logitech C270 cameras that were connected to a laptop running Mediapipe and Falkor had an ***L*** of 0.3 meters (30cm). Using a human hand as an object to detect, Falkor return the following data and graph:

Results From Initial Test Against Equation Derived



Distance from Camera One	P1:P2			Average P1/P2	Falkor's Distance
(in L)	Trial One	Trial Two	Trial Three		(in L)
1	1.98	1.87	2.20	2.02	0.98
2	1.53	1.68	1.32	1.56	1.79
3	1.25	1.36	1.28	1.30	3.33
4	1.31	1.16	1.26	1.24	4.17

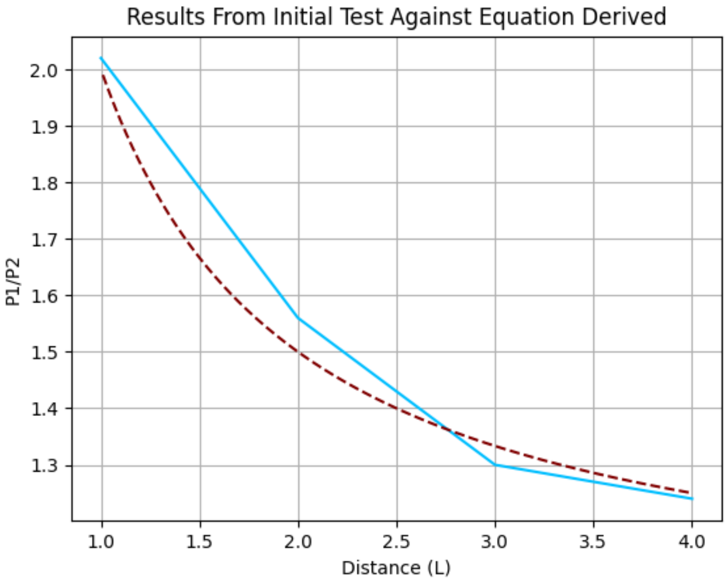


Figure 6. Table and graph from the results of Test One. The graph evaluates Falkor's equation.

Images made by matplotlib.

From the graph, we can conclude that the relationship between the ***P1/P2*** and the real distance is a reciprocal relationship, which follows the equation above. And from the table, we can see that the distance given by Falkor has a maximum range of 0.33***L*** away from the real distance. Thus, the equation does work. There is a great range of data, which can be explained by the depth of the lens and the instability of object detection for a hand. With a small distance of 0.3 meters, the depth of the lens will have a bigger impact on Falkor. In scenarios where Falkor is needed, ***L*** should be at 1 meter or above. A bigger factor would be the bounding box of the hand, which is constantly changing and moving since human hands do move quite often. This explains the big margin of error. Therefore, it is concluded that

Falkor is a practical method of distance detection, but it requires stationary object and an ***L*** of greater or equal to 1 meter.

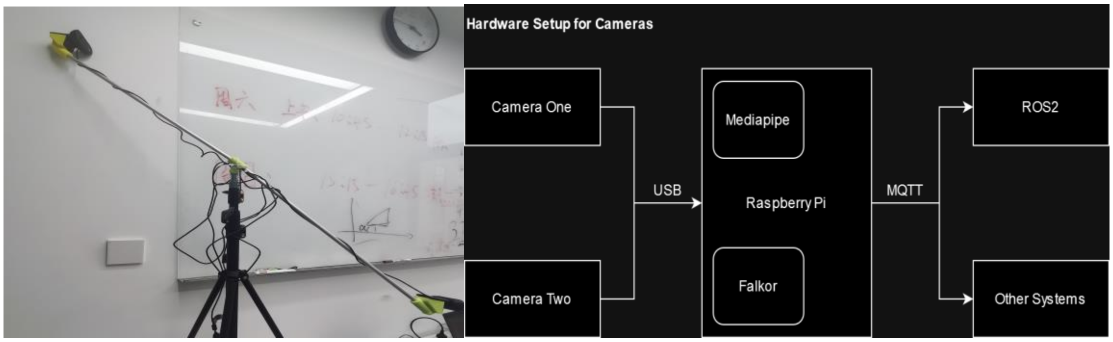


Figure 7. Hardware set up. Two cameras were placed 1 meter apart with a 0.4-meter height difference. The device sustained some damage after testing and the angle was curved. The right is a mind map of such hardware.

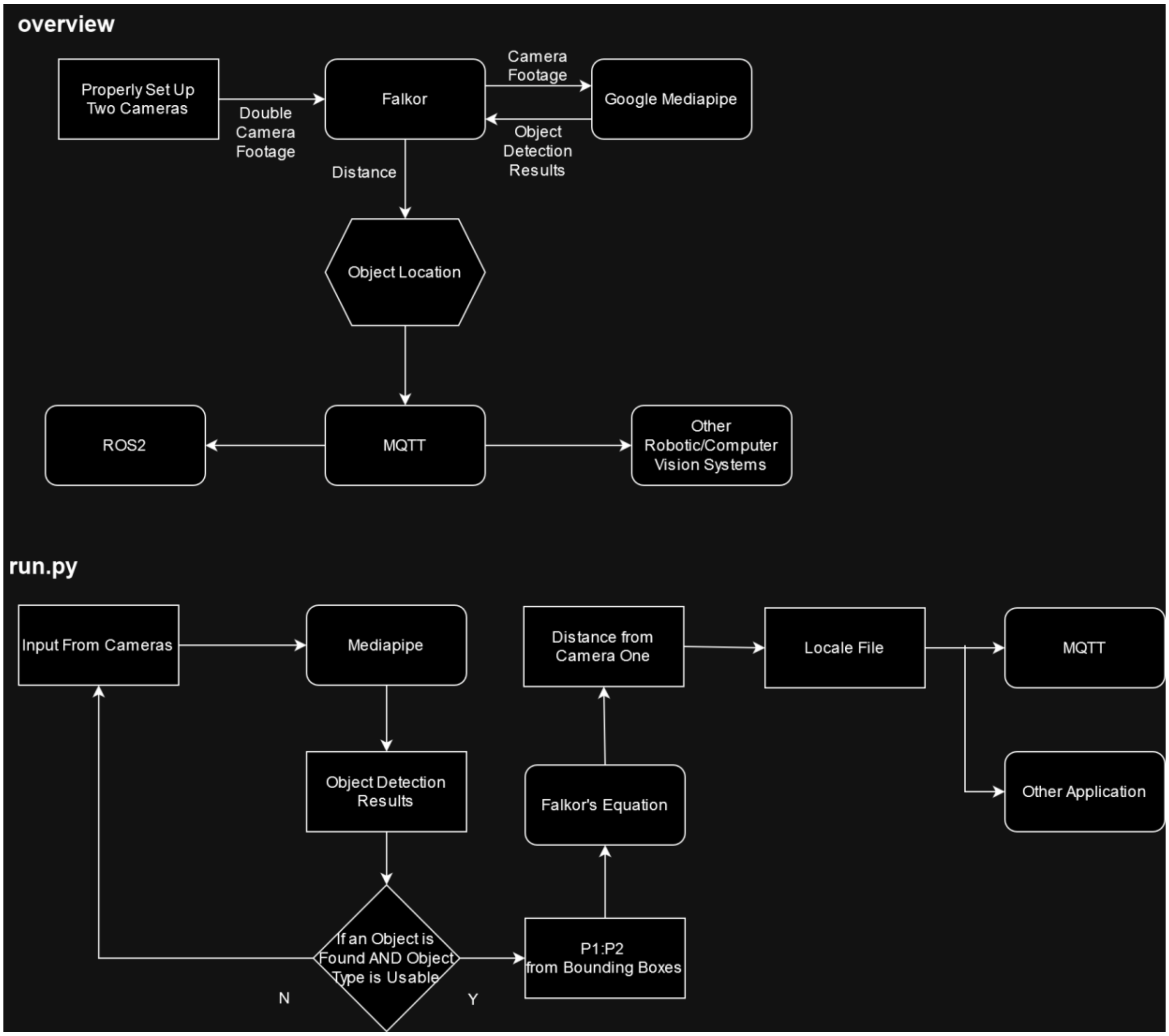


Figure 8. mind maps of the code made for Falkor. The first image shows an overview of the entire project. The second image shows a more detailed description of the main program or run.py.

Implementing Falkor

Continuing, an improved set up was made. Two Logitech C270i cameras are stationary placed 1 meter apart (***L*** = 1 meter) with a 0.4-meter height difference on an adjustable camera stand (Figure 6). They are connected to a Raspberry Pi 4 with the 64-bit Raspberry Pi OS. Falkor's code was downloaded to the Raspberry Pi. Mediapipe for Python was downloaded to the Raspberry Pi, with the packages necessary for it. This is vital for Falkor, since Mediapipe has systems for object detection with pre-trained models, so that we could get the bounding boxes for ***P1*** and ***P2*** (Figure 2). The program's overall structure is shown in Figure 7. The program would send the camera footage to Mediapipe and gain the results for ***P1*** and ***P2***, then using the equation to gain the object's location. More details are shown in Figure 7. The objects' location would be saved to a locale file. Any other program would use MQTT to publish the location of the object. MQTT works on a publisher-subscriber model, where the results are sent to a server's directory and then to any subscribers subscribed to such directory. This would allow robotic systems like ROS2 to gain the location of the object.

RESULTS

After the first test, two tests were done on this technology. Test Two aimed to validate Falkor's ability to gain the location of an object. Test Three aimed to show that such technology could be used with robotic systems or other computer vision systems.

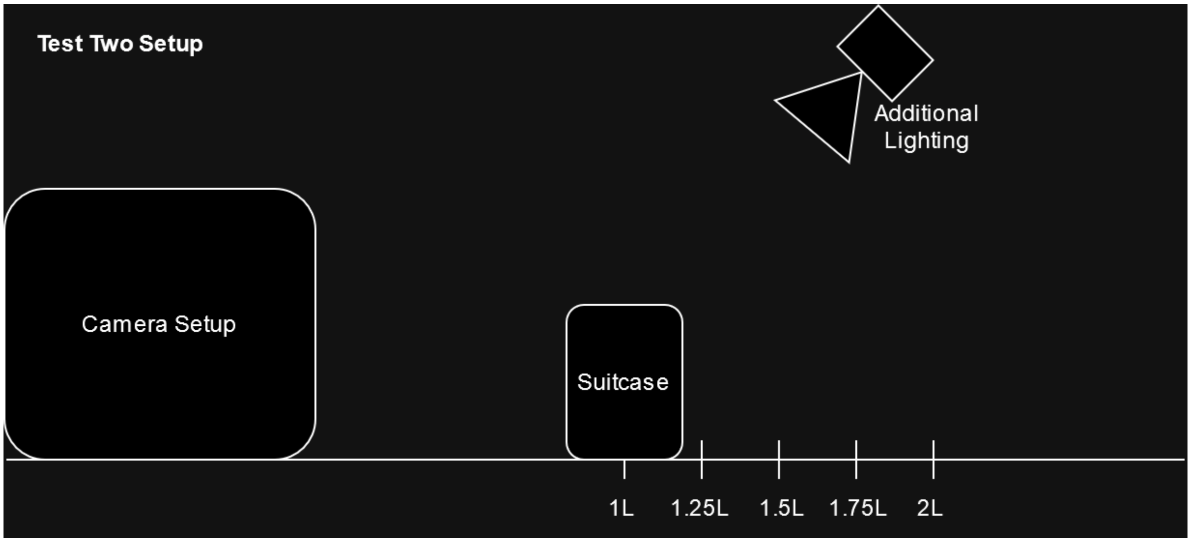
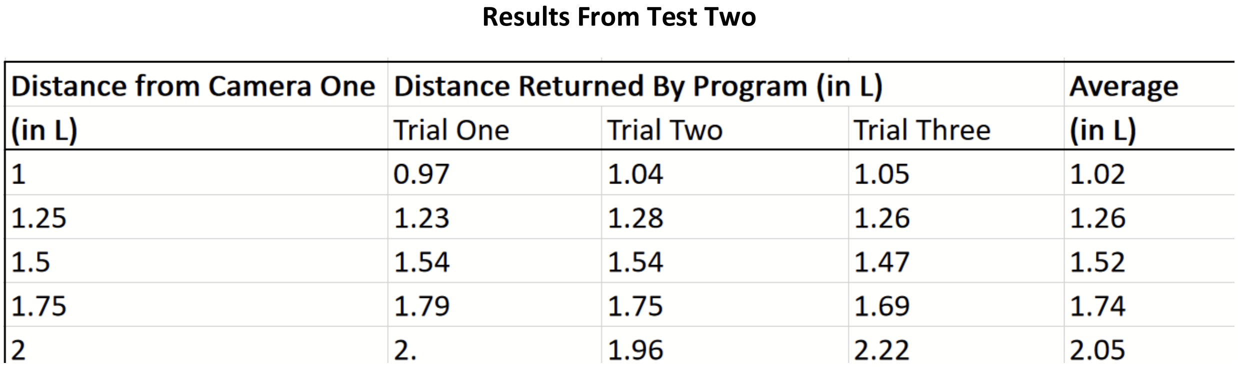


Figure 9. A demonstration of the set up for Test Two. The suitcase was put on intervals 1, 1.25, 1.5, 1.75, 2***L***. Additional lighting was needed.

The setup shown in Figure 6 was put in front of a 4-meter aisle. Using 1. meter (1***L***), 1.25 meters (1.25***L***), 1.5 meters (1.5***L***), 1.75 meters (1.75***L***), and 2 meters (2***L***) from Camera One as intervals, a small suitcase was put under adequate lighting. This is shown in Figure 9. For each interval, the suitcase was rotated 90, letting the cameras detect different sides. This is shown in Falkor gained the data shown below:



Results From Test Two

Distance from Camera One	Distance Returned By Program (in L)			Average
(in L)	Trial One	Trial Two	Trial Three	(in L)
1	0.97	1.04	1.05	1.02
1.25	1.23	1.28	1.26	1.26
1.5	1.54	1.54	1.47	1.52
1.75	1.79	1.75	1.69	1.74
2	2.	1.96	2.22	2.05

Figure 10. Table and graph showing all the data from Test Two.

The results from Figure 10 are promising. Most of the distances were close to the physical distances, with the greatest average range being only 5 centimeters away from the physical distance. Some observations were made during this test too. The suitcase needed additional lighting for object detection to return processible results. And many of the results were not fairly accurate, as the object label was constantly showing "refrigerator" instead of suitcase. This could be explained by how the program needed to compress the camera footage into mjpg because of the limited processing power of the Raspberry Pi. Thus, the quality of the object detection would decrease compared to using quality images. This compression could also explain some of the inaccuracies in the table in Figure 10.

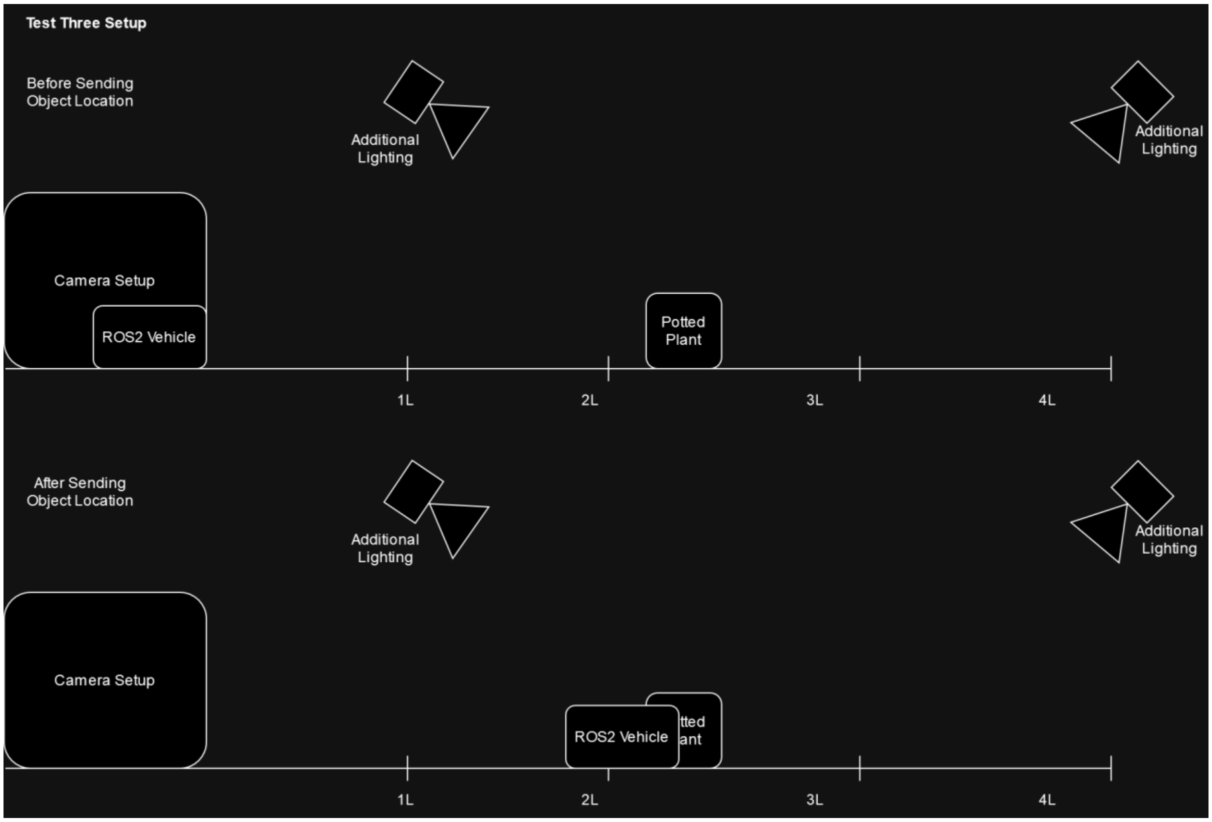


Figure 11. A demonstration of the setup for Test Three. The plant was placed randomly between 1***L*** and 4***L***. Additional lighting was needed for better accuracy.

The third test focused more on quantitative data to show how Falkor could be a great part for robotic systems. Using the same environment shown in Figure 9, a vehicle running ROS2 was set beside the cameras. A potted plant (plants gave more correct object detection results) was set at random distances from Camera One. This is shown in Figure 11. Falkor would publish the object's location through MQTT. After three tests, the vehicle was successful in finding the plant, moved towards its location, and parked either at the plant's exact location or close to it. The distance equation ran smoothly and returned exact results even with some human interference in the background. Thus, proving that Falkor could be run with robotic systems with efficiency.

DISCUSSION

Evaluation of Falkor's Results

Restating the equation: $\frac{P1}{P2} = 1 + \frac{1}{x+1}$

This reciprocal equation brings us the distance between an object and the camera set up, which allows us to find the object, which is used in intelligent robotic systems. The first test shown in figure 6 supports this despite its ranging data. The inaccuracies in the data could be explained by the unstable movement of the hand, resulting in varying bounding boxes, which is the reason the remaining tests used un-moving objects.

The remaining two tests gave promising results. The table of Figure 10 shows even under low-quality photography and object detection, using 1 meter as ***L*** returns functional distance results. The inaccuracy of the data is neglectable, as the distances returned were close to the object's location. Thus, proving that Falkor is workable for real-life scenarios. Test Three proved that Falkor is viable as an element in complex robotic and computer vision systems. It correctly returned the result of distance, sent the results through MQTT, and the ROS2-based autonomous vehicle was able to locate the object and arrive next to the object. It was executed with gesture-based computer vision, despite the quality of the gesture detection. To conclude, Falkor is a viable solution for robotic systems, especially in smart city/building infrastructure.

Limitations

Falkor seems inconvenient and almost useless because of its line-like set up. This technology can only be used for specific line-like areas, since it is better for one-dimensional distancing with plenty of light. It could only be used with stationary cameras; so, it cannot be used in autonomous vehicles that require constant quality performance. This technology could only be used on stable objects at rest, as shown in Test One.

Falkor and Robotic Systems in Smart Cities and Buildings

It is true that this row of cameras would be useless in a 2D layout, however modern infrastructure, such as streets and alley ways have a line-like layout for vehicles. Falkor takes advantage of this, because it is built as computer vision for line-like layouts and it works with low-cost, low-quality cameras and computers, which makes it more accessible for smart city layouts. In addition, this technology does not require the cameras to be completely synchronized, therefore city streets and building hallways could implement long rows of these cameras, allowing this distance finding throughout entire streets.

In the preliminary stages, a first idea of airport smart infrastructure was proposed. Suppose Person One was dropped off at the gate of the airport with his suitcase. However, let us say that the suitcase is too heavy for Person One to carry. If Person One left to obtain a trolley for the suitcase, the suitcase could have been stolen, or someone else could have mistaken the suitcase for their own. It would be helpful if the trolley could almost-magically arrive at Person One's suitcase. This can be done with intelligent robotic systems, where the system detects a person in need, exactly locates the person, and executes a trolley to arrive at the person's or suitcase's location. Falkor could act as a locating system because it can be perfectly integrated with airport streets and find the exact location of the person. Furthermore, the robotic technology used in Test Three, shown in Figure 11, could be adapted for this situation.

This method would require less resources than other methods, as they would need expensive hardware or highly trained neural network systems. Without a doubt, funds for public infrastructure would be limited (Roy 2016), thus low-expense hardware should be used. Neural networks create a substantial portion of carbon emissions and professionals have vouched for "greener" neural network systems (Patterson et al. 2021); thus, simpler methods of distance detection should be considered instead of machine learning models.

With more innovative, futuristic cities, this method could be used for many robotic systems, such as delivery services, transportation, and cleaning systems. A notable example of this would be the in-development city, The Line, as Falkor is designed to work in "a line." Strong infrastructure is one of the key drivers of thriving smart cities (Veselitskaya et al. 2019). Internet of things and similar technologies is an innovative way to reduce many problems that would arise in smart cities, such as overcrowding, citizen stress, and safety (Drepaul 2020). Some developing countries lack the resources for infrastructure (Roy 2016) and Falkor allows intelligent robotic systems to be more accessible. Falkor and its associating technologies could be strong components for these smart systems and thus should be considered in these projects.

ACKNOWLEDGMENTS

All code can be found and resources in this GitHub repository: https://github.com/fa1k0or/Distance-Detection-with-Stationary-Computer-Vision

REFERENCES

Drepaul, Nicole A. "Sustainable Cities and the Internet of Things (IOT) Technology: IOT Technology Improves the Development of Smart Cities' Infrastructures and Reduces over-Population Stresses." *Consilience*, no. 22, 2020, pp. 39–47. *JSTOR*, https://www.jstor.org/stable/26924960. Accessed 6 Nov. 2023.

Fan, Rui, et al. *Computer Stereo Vision for Autonomous Driving*. arXiv, 16 Dec. 2020. *arXiv.org*, https://doi.org/10.48550/arXiv.2012.03194.

Patterson, David, et al. *Carbon Emissions and Large Neural Network Training*. arXiv, 23 Apr. 2021. *arXiv.org*, https://doi.org/10.48550/arXiv.2104.10350.

Roy, Souvanic. “The Smart City Paradigm in India: Issues and Challenges of Sustainability and Inclusiveness.” *Social Scientist*, vol. 44, no. 5/6, 2016, pp. 29–48. *JSTOR*, http://www.jstor.org/stable/24890283. Accessed 6 Nov. 2023.

VESELITSKAYA, Natalia, et al. “DRIVERS AND BARRIERS FOR SMART CITIES DEVELOPMENT.” *Theoretical and Empirical Researches in Urban Management*, vol. 14, no. 1, 2019, pp. 85–110. *JSTOR*, https://www.jstor.org/stable/26590931. Accessed 6 Nov. 2023.

Wandinger, Ulla. “Introduction to Lidar.” *Lidar: Range-Resolved Optical Remote Sensing of the Atmosphere*, edited by Claus Weitkamp, Springer, 2005, pp. 1–18. *Springer Link*, https://doi.org/10.1007/0-387-25101-4_1.