AN OBJECT-DRIVEN COLLISION DETECTION WITH 2D CAMERAS USING ARTIFICIAL INTELLIGENCE AND COMPUTER VISION

Yang Liu[1], Evan Gunnell[2], Yu Sun[2], Hao Zheng[3]

[1]Department of Mechanical and Aerospace Engineering,
George Washington University, Washington, DC 20052
[2]California State Polytechnic University, Pomona, CA, 91768
[3]ASML, Wilton, CT 06897

ABSTRACT

Autonomous driving is one of the most popular technologies in artificial intelligence. Collision detection is an important issue in automatic driving, which is related to the safety of automatic driving. Many collision detection methods have been proposed, but they all have certain limitations and cannot meet the requirements for automatic driving. Camera is one of the most popular methods to detect objects. The obstacle detection of the current camera is mostly completed by two or more cameras (binocular technology) or used in conjunction with other sensors (such as a depth camera) to achieve the purpose of distance detection. In this paper, we propose an algorithm to detect obstacle distances from photos or videos of a single camera.

KEYWORDS

Autonomous driving, computer vision, machine learning, artificial intelligence, distance detection, collision detection.

1. INTRODUCTION

As a new product of modern society, artificial intelligence [1] is the future direction of development. It can operate automatically in a specific environment according to a present mode. Without human management, the expected or higher goals can be achieved. Autonomous driving[2] is one of the most popular technologies in artificial intelligence. It can bring great convenience to our lives, and at the same time it is a kind of release to people's fatigue when driving. From the perspective of the nature of autonomous driving, it is essentially a fast-reacting robot, and its level of intelligence is relatively high. If autonomous driving can be achieved, it will mean a huge step forward not only in commuting but also in the field of robotics, as the same level of artificial intelligence can be applied to more standard robots. What's more, autonomous driving technology can provide people with a huge benefit of time. After being released from the requirement to drive, people in the car can do what they want. At this time, the car on the journey is a brand-new space for people's lives, and people will have one more means of living their life than before. The supporting facilities of the car will be completely changed, and passengers can work and entertain. This change in the sense of space provides a new service model and a new life experience for social development. Collision detection [3] is an important issue in automatic driving, which is related to the safety of automatic driving.

David C. Wyld et al. (Eds): EMSA, SEA, AIFU, NLCAI, BDML, BIoT, NCOM, CLOUD, CCSEA, SIPRO - 2022
pp. 317-328, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.120626

Nowadays, many collision detection methods have been proposed, but they all have certain limitations and cannot meet the requirements for automatic driving. Ultrasound [4] is an important means of collision detection. The energy consumption of ultrasonic waves is relatively slow, the propagation distance in the medium is relatively long, the realization is convenient, the cost is low, the calculation is simple, and it is easy to achieve real-time control. Ultrasonic radar has great advantages in short-distance measurement. However, ultrasonic radar has certain limitations in measuring distance at high speeds and is greatly affected by the weather. Moreover, the propagation speed of ultrasonic waves is slow, and when the car is running, it cannot keep up with the change of the distance between the cars in real time. In addition, the ultrasonic scattering angle is large, and the directivity is poor. When measuring a long-distance target, its echo signal will be relatively weak, which affects the measurement accuracy.

Another common method is the camera [5]. The camera is generally composed of a lens, an image sensor, an Image Signal Processor (ISP) [6], and a serializer. The general procedure is that the basic information of the object collected by the lens is processed by the image sensor and then sent to the ISP for serialized transmission. Transmission methods can also be divided into LVDS-based transmission on coaxial cable or twisted pair or direct transmission via Ethernet. The camera is mainly used in the automatic driving system for obstacle detection, lane line detection, road information reading, map construction and auxiliary positioning. However, the obstacle detection of the current camera is mostly completed by two or more cameras (binocular technology) [7] or used in conjunction with other sensors (such as a depth camera [8]) to achieve the purpose of distance detection.

The depth camera based on binocular stereo vision is similar to the human eyes, and is different from the depth camera based on TOF [9] and structured light principle. It does not actively project the light source to the outside, and completely relies on the two pictures taken (color RGB or grayscale) to calculate the depth, so it is sometimes called a passive binocular depth camera.

TOF is short for Time of flight, literally translated as the meaning of flight time. The so-called time-of-flight method 3D imaging is to continuously send light pulses to the target, and then use the sensor to receive the light returning from the object, and obtain the target object distance by detecting the flight (round trip) time of the light pulse. This technology is basically similar to the principle of the 3D laser sensor [10], except that the 3D laser sensor scans point by point, while the TOF camera obtains the depth information of the entire image at the same time.

All of these technologies are often combined into one unit with the data being passed through some level of advanced object detection [11] or other machine learning algorithm. However, our aim in this paper is to reduce the number of elements needed to produce an accurate prediction so as to reduce the barrier to entry when utilizing automated object detection.

Generally, humans obtain depth information through their eyes. Binocular technology in particular is based on this idea. However, even when we somewhat limit our view and observe something with only one eye, we can still feel the distance of the object. Even with some loss of depth perception, humans can generally still accurately predict distances with one eye closed. This is because people themselves have a very good understanding of the world where they live (prior knowledge). They have a basic prediction of the size of everyday objects (visual training for many years) [12]. According to common sense, it is indeed possible to infer the distance of the object in the image. In addition, when a person observes an object with a single eye, the human eye is actually frequently moving and scanning its surroundings. This is functionally equivalent to a moving monocular camera [13], which is similar to the principle of the structure from motion. The moving monocular camera compares the difference of multiple frames. It is indeed possible to get in-depth information. It shows that humans can also obtain a certain depth

of information from a single eye. At the same time, the depth information is obtained by comparing the differences of multiple frames. Therefore, it stands to reason that it is feasible to obtain depth information with just a single camera. In this paper, we propose an algorithm to detect obstacle distances from photos or videos of a single camera.

In this project, we did three experiments to finish the distance detection function. First of all, we added distance labels in YOLOv5 [14] so that we can get the real-time information from the screen. Then we collected the dataset of one object to build and test our models. After we prove that the distance information can be obtained. We collected the dataset of different objects to compare the results of different models. Lastly, different cameras were selected to collect the dataset, in order to explore and validate the impact of different camera types.

The rest of the paper is organized as follows: Section 2 gives the details on the challenges that we met during the experiment and designing the sample; Section 3 focuses on the details of our solutions corresponding to the challenges that we mentioned in Section 2; Section 4 presents the relevant details about the experiment we did, followed by presenting the related work in Section 5. Finally, Section 6 gives the conclusion remarks, as well as pointing out the future work of this project.

2. CHALLENGES

Challenge 1: Learning and training the model to predict the distance for specific objects shown in the view based on the size (dimension) of the recognized object is difficult. In different environments, various factors such as the type, appearance, and size of the recognition object are different. For example, in the case of autonomous driving, even in different countries and regions, there are still differences in objects with uniform standards such as traffic lights, buses, and taxis. Therefore, the first step of training and learning, establishing a database, is a big challenge. Although there are already some object recognition databases, for pedestrians, animals, and other objects that have individual differences, the addition of size information data is still a big challenge. For example, the height of an adult man may range from 1.65m to 1.90m, so more detailed classification is needed. In addition, the camera parameters and camera postures in each model are different. Whether this factor will affect the distance detection needs to be further explored. Although in theory a monocular camera can obtain a certain information for depth, it is still doubtful whether the information that is obtained by a monocular camera can meet the requirements for automatic driving.

Challenge 2: The impact on the accuracy of the distance prediction coming from the different camera types is unknown. There are many types of cameras on the market today. The influence of various camera parameters on object recognition and distance detection is unclear. It is also a big challenge to find the influence of the parameters of each camera on the distance detection. Among the various parameters, the change of the focal length will inevitably affect the distance detection, so during use, how much influence the change of the focal length will have on the distance detection remains to be explored. This will determine whether to use a zoom camera or a fixed focus camera in the end. Aperture is another important parameter of the camera, which directly contributes to the sharpness of the image. The aperture must have a certain influence on object recognition, but whether it has a greater influence on distance detection still needs to be explored. So how to choose the type of camera is also a big challenge. Due to how many camera options there are, we will simply have to select one type of camera and use that as the standard for all of our testing.

Challenge 3: Choosing a reference machine learning model [12] involes multiple issues. Relying on the existing reference model to predict the distance of other objects in the same view

is a possible method to solve our problem. There are many related models for object recognition. On this basis, adding size data information to establish a mapping relationship with depth is the main task of this paper. In the object recognition model, the reliability of object recognition is also a major factor affecting distance detection. So how to choose a reference model is also a big challenge. Measuring data input values and model selection accuracy can help us pick the most optimal reference model. However, it would take a lot of time to test every model. We can read some related work about these models so that we know the range of their application. We can also learn their advantages and disadvantages from this work. From the studies that already exist in these papers, we can choose some of the models which might be better than others. Then we can use the same input data, which we know all the information, to test each model we chose and choose the one that gives us the highest accuracy. To measure the accuracy of our model we will provide a dataset of known objects and distances and see how well it performs. We can then also tune the model once it is chosen to tailor it specifically to our problem.

3. METHODOLOGY

In this section, we will introduce the Overview of the System and Models and Feature Selections.

3.1. Overview of the System

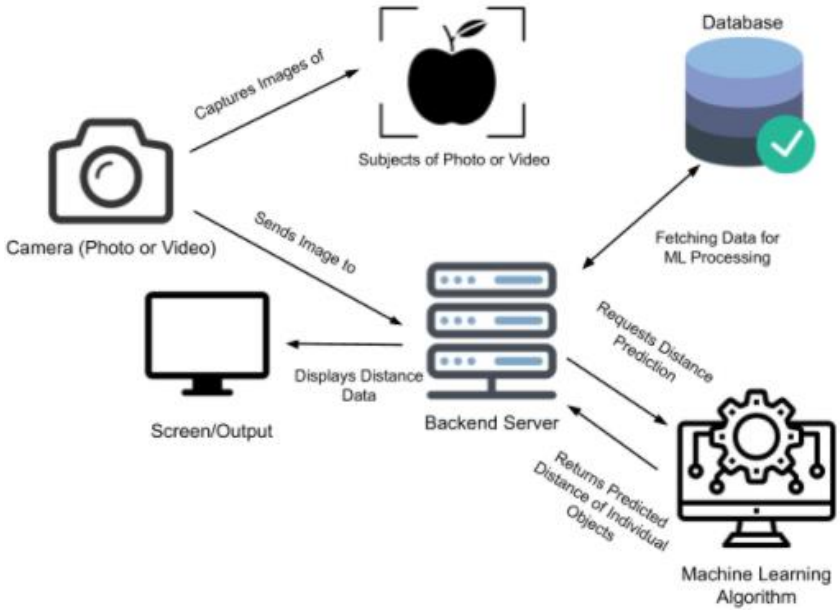

Figure 1. The Overview of the System Architecture

The proposed system framework is shown in the Figure 1. The input can be submitted in two ways: one is to take photos captured by the camera as input, and the other being to use video as input. After getting the specified data, the program will use a Web API/Web service to handle communication between the frontend, the database and the backend server. This can be implemented in a service such as Python Flask. The database we are using in particular is a non-relational database that holds all of our training data and testing data. This data is utilized to obtain the distance information through the machine learning algorithm and heuristic. Aside from just training data, it also contains important information related to the approximate size of certain objects to help better guide the system. The machine learning portion of the system uses modern regressive machine learning libraries. We will show some testing results for a variety of the

common models such as Linear Regression, Polynomial regression, and Random Forest Regression.. After undergoing all of this processing in our backend web server, the picture or video with the distance information is returned to the display screen as output. If it is a video, it dynamically updates the screen with the most recent distance information. For a picture, it will simply estimate the current distance for the recognized objects.

3.2. Models and Feature Selections

The primary focus of the project is the machine learning component, with us a common YOLO machine learning model. YOLO, standing for "You Look Only Once", is a member of the object detection model family. Its first iteration was released in 2016 by Joseph Redmon with multiple modern iterations being implemented. This particular project focuses on the YOLOv5 model, augmenting it with additional machine learning to accomplish our project task of estimating distance of recognized objects. The YOLO family of models consists of three main architectural blocks: the Backbone, the Neck, and the Head. Each component of the models helps to employ a specific feature of it. Combined they allow the model to quickly compute object detection on a variety of objects. The backbone employs CSPDarknet as the primary tool for image feature extraction consisting of cross-stage partial networks. The neck represents the aggregation of training features that will later be used by the head. It uses PANet to generate a feature pyramid network for its aggregation. Lastly, the YOLOv5 head focuses on providing multiple layers that generate predictions from the anchor boxes for object detection.[14]

Our project aims to extend its fundamental object recognition moduel by adding distance information in both photo and/or video predictions. Below is an example of some of the code added to the library.

```
def calculate_distance(xyxy):
    img_height = 640
    img_width = 640

    # Calculate the relative size of the the bounding box for a person. 80% = 2ft, 50% = 5ft
    percentage_of_screen = ((xyxy[2]-xyxy[0]) / img_width) * 100

    # 1ft = 90% +
    # 2ft = 80% screen (0.78)
    # 3ft = 70%
    # 4ft = 60%
    # 5ft = 50% screen (0.5)
    # 6ft = 40%
    # 7ft = 30%
    # 8ft = 22.5% screen (0.225)
    # print(percentage_of_screen)
    print((xyxy[2]-xyxy[0]) / img_width)
    print("LOOKING AT A PERSON")

    return "L = " + str((100 - percentage_of_screen) / 10)
```

Figure 2. The Code Excerpt of Calculating the Distance using a Base Formula

In this paper, we present an algorithm based on the width of the object to get the distance information. We can judge the distance by the width of screen ratio of the object, because when the object goes farther away from the camera, the width of screen ratio of the object will be smaller. For the similar reason, we could also use the height of screen ratio to get the depth information when the width of the object is too small to be measured or exceeds the range of the camera. Utilizing width or height fixes an issue of potential camera angles. When a camera is

higher or lower affects the estimated height while if it is to the left or right of the object it distorts the perceived width. Once we collect the data of the width of image ratio and the distance between object and camera, we could build a model reflecting the relationship between the percentage of the screen and the distance of the current object. To build the model, we could use machine learning algorithms [12] in the width-based detection. Results of testing the various predictive models will be available in Section 4 later in the paper.

Because we get the distance information from the width of screen ratio, camera wide angle will be the most direct factor to influence the distance prediction. Changing the total width view of the camera has a drastic impact on what the image will look like, in turn distorting the potential results of the object detection. For example, a fish-eye camera lens has a significantly larger viewing angle than a standard phone camera does. The focal length also will be a factor to influence the distance prediction. As the focal length decreases, the width view will be larger, and the distortion of the picture will also become larger. Two same things with different distances in photos, the closer one will be much larger than the farther one. However, the closer one in shorter focal length cameras will seem larger than the one in longer focal length cameras. Another factor for cameras which will influence the distance prediction is the aspect ratio of the photo or video. The different aspect ratio will distort the object in video or photo. The shape of the object will be changed. For example, when we watch TV, if the aspect ratio is 4:3, the people will appear shorter and fatter than the aspect ratio is 16:9.

In order to better simulate the effect of human monocular distance measurement, we think that it is better to use video than photos. Due to the existence of waves of movement, the comparison between each frame and the previous frame or several frames can theoretically achieve the effect of binocular imaging. As a result, a 3D space can be better constructed, so that the distance information of the object can be obtained better and more accurately.

After selecting the appropriate model, we can use the information from our database to train the model and get it prepped for potential input. It is then integrated with the rest of the application. It will connect to the API of the backend server, take in potential image/video data, and then return the most accurate distance estimate to the photo or video.

4. EXPERIMENTS

Three experiments have been designed and conducted to illustrate the performance of the proposed distance prediction algorithm.

4.1. Experiment 1: Distance Estimation using Machine Learning Models

In this part, we choose several machine learning models to finish distance prediction. We collect the width percentage of the screen with different distances from a person to the camera.

By using this data to predict the distance, we choose the Linear Regression model [15], Polynomial Ridge Regression model [16], Random Regression model [17], and ElasticNet Regression model [18]. For the Polynomial Ridge Regression model, we select different Poly Features, such as 3 Poly Features and 4 Poly Features. For the Random Regression model, we compare 2 max depth models and no max depth models.

4.2. Experiment 2: The Impact of the Object Type on the Distance Estimation

In this part, we choose several machine learning models to finish distance prediction and compare the influence of object type. Besides the data we collect in experiment 1, we also collect the data of different objects, such as the cell phone and stuffed penguin.

By using this data to predict the distance, we choose the Linear Regression model [15], Polynomial Ridge Regression model [16], Random Regression model [17], and ElasticNet Regression model [18]. For the Polynomial Ridge Regression model, we select different Poly Features, such as 3 Poly Features and 4 Poly Features. For the Random Regression model, we compare 2 max depth models and no max depth models.

4.3. Experiment 3: The Impact of the Camera Type on the Distance Estimation

In this part, we choose several machine learning models to finish distance prediction and compare the influence of camera type. Besides the data we collect in experiment 1 and 2, we also collect the data of the second person using a different camera (Dell G5).

By using this data to predict the distance, we choose the Linear Regression model [15], Polynomial Ridge Regression model [16], Random Regression model [17], and ElasticNet Regression model [18]. For the Polynomial Ridge Regression model, we select different Poly Features, such as 3 Poly Features and 4 Poly Features. For the Random Regression model, we compare 2 max depth models and no max depth models.

4.4. Dataset and Results

As Figure 3 shows the camera information, the object width information and the input data, Figure 4 shows the distance data as the output data, test input data and the machine learning model we use. Figure 5 shows the output of the test input data, including predicting distance and the Cross Validation Average Testing Scores.

```
# 1. change the model (Linear Regression, Polynomial Regression, RandomForestRegression)
# 2. same model, change the parameters

# Data [(camera type), object, width]

#Evan Camera: Razer Kiyo ---camera 0
    #Still image: 4 megapixel
    # Video: 1080p at 30 fps, 720p at 60 fps

    #Object width: 0 - person (20in.), 1 - cell phone (6in. horizontal), 2 - stuffed penguin (13in.)
# Yang Camera:  Dell G5 ---camera 1
    # Still image: 0.92 megapixel (HD)
    # Video: 1280 x 720 (HD) at 30 fps
    #Object width: 0 - person (15.5in.), 1 - cell phone (6in. horizontal)
input_data = [
  [0, 0, 80],
  [0, 0, 60],
  [0, 0, 46.5],
  [0, 0, 40],
  [0, 0, 35],

  [0, 1, 56],
  [0, 1, 28],
  [0, 1, 18],
  [0, 1, 14],
  [0, 1, 11.75],
  [0, 1, 8.5],

  [0, 2, 65],
  [0, 2, 55],
  [0, 2, 47],
  [0, 2, 41.2],
  [0, 2, 30],
  [0, 2, 20],
  [0, 2, 15],

  [1, 0, 67],
  [1, 0, 46],
  [1, 0, 40],
  [1, 0, 32.5],
  [1, 0, 28],

  [1, 1, 54],
  [1, 1, 26],
  [1, 1, 16.7],
  [1, 1, 13],
  [1, 1, 10.5],
  [1, 1, 7]
```

Figure 3. The Input Dataset for the Experiments

```
#Distance in feet
output_data = [
  2,3,4,5,6, # person 1
  1,2,3,4,5,6,# phone 2
  2,3,4,5,6, 7, 8, # stuffed penguin
  2,3,4,5,6, # person 2
  1,2,3,4,5,6,# phone 2
]

input_data.extend(input_data)
input_data.extend(input_data)

output_data.extend(output_data)
output_data.extend(output_data)

input_data.extend(input_data)
input_data.extend(input_data)

output_data.extend(output_data)
output_data.extend(output_data)

test = [[0, 1, 20]]
cv_num = 5

# Linear Regression
model = linear_model.LinearRegression()
model.fit(input_data, output_data)
print(model.predict(test))
print(sum(cross_val_score(model, input_data, output_data, cv = cv_num, scoring = 'neg_mean_squared_error')) / cv_num)


# Polynomial Ridge Regression (4 Poly Features)
model2 = make_pipeline(PolynomialFeatures(4),linear_model.LinearRegression())
model2.fit(input_data, output_data)
print(model2.predict(test))
print((cross_val_score(model2, input_data, output_data, cv = cv_num)))
print(sum(cross_val_score(model2, input_data, output_data, cv = cv_num) / cv_num))

# Polynomial Ridge Regression (3 Poly Features)
model2_5 = make_pipeline(PolynomialFeatures(3),linear_model.LinearRegression())
model2_5.fit(input_data, output_data)
print(model2_5.predict(test))
print(sum(cross_val_score(model2_5, input_data, output_data, cv = cv_num, scoring = 'neg_mean_squared_error')) / cv_num)

# RF Max_depth = 2
model3 = RandomForestRegressor(max_depth = 2, random_state = 0)
model3.fit(input_data, output_data)
print(model3.predict(test))
print(sum(cross_val_score(model3, input_data, output_data, cv = cv_num, scoring = 'neg_mean_squared_error')) / cv_num)

# RF No max Depth
model3_5 = RandomForestRegressor(random_state = 0)
model3_5.fit(input_data, output_data)
print(model3_5.predict(test))
print(sum(cross_val_score(model3_5, input_data, output_data, cv = cv_num, scoring = 'neg_mean_squared_error')) / cv_num)

# ElasticNet Regression
model4 = ElasticNet(random_state = 0)
model4.fit(input_data, output_data)
print(model4.predict(test))
print(sum(cross_val_score(model4, input_data, output_data, cv = cv_num, scoring = 'neg_mean_squared_error')) / cv_num)
```

Figure 4. The Code Excerpt for Dataset Training and Cross Validation

```
[5.27190273]
-1.8569599858280637
[2.71911058]
[0.99693576 0.99663323 0.99688429 0.99680586 0.99652266]
0.9967563586641514
[2.89022235]
-0.02830028786354614
[4.29479788]
-1.4191690835273607
[3.]
0.0
[4.83249015]
-2.034562505455932
[0.47311828 0.43010753 0.43010753 0.44086022 0.43478261]
0.44179523141654 98
[0.03019402 0.17018951 0.79961646]
[0.5483871  0.53763441 0.55913978 0.62365591 0.51086957]
0.5559373539036933
[0.01038713 0.05466782 0.93494505]
[1. 1. 1. 1. 1.]
1.0
```

Figure 5. The Excerpt of the Test Results

Figure 6 shows the Cross Validation Average Testing Scores for different model. From Figure 6, we can know that Random Forest (depth=2) get the highest score and ElasticNet Regression get the lowest score. In this way, we could say Random Forest (depth=2) is the most suitable model for the data.

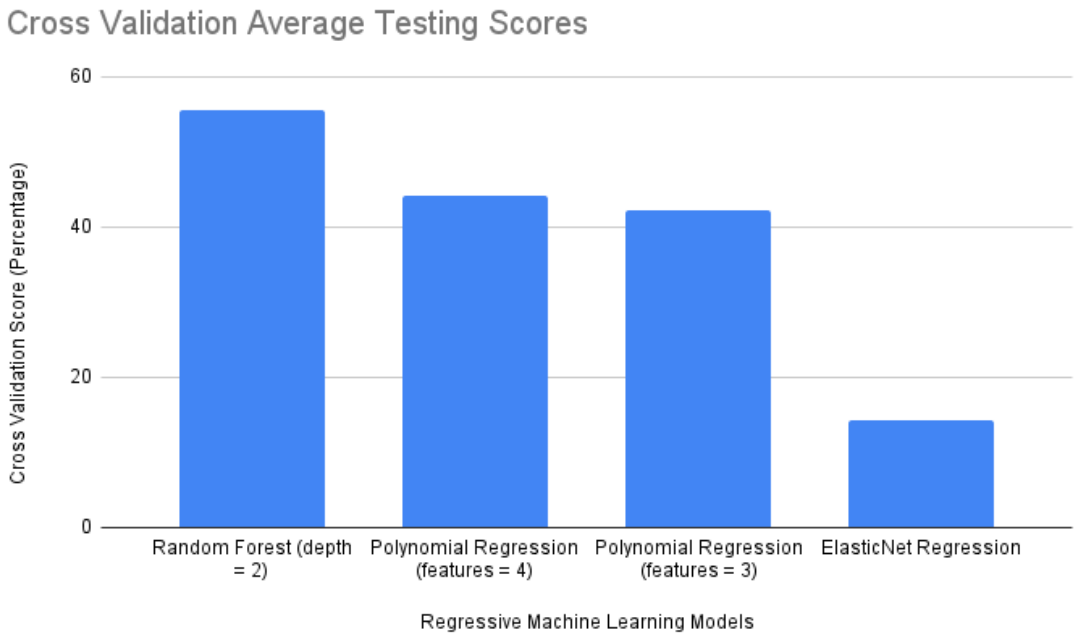

Figure 6. The Accuracy Comparison of Different Machine Learning Models

Figure 7 shows the influence of various factors on distance prediction. We can know that the percentage of screen width is the most important factor to predict the distance. The camera and the object has less influence on prediction.

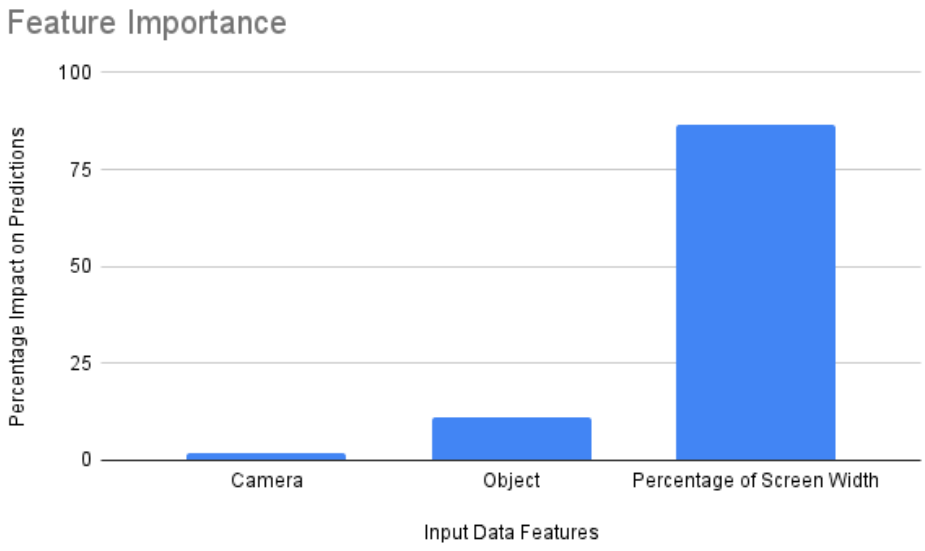

Figure 7. The Impact Comparison with the Different Extra Feature Set

5. RELATED WORK

Iro Laina. et al proposed a fully convolutional architecture to model the ambiguous mapping between monocular images and depth maps. They use MatConvNet, and train on a single NVIDIA GeForce GTX TITAN with 12GB of GPU memory. The network is trained on RGB inputs to predict the corresponding depth maps. They use data augmentation to increase the number of training samples. They model small translations by random crops of the augmented images down to the chosen input size of the network [19]. Compared to our work, they don't get the actual distance but a relative location, while we will point out the exact distance from the camera to the objects.

Michael W. Tao. et al presented a defocus and correspondence algorithm to get the depth information. Their algorithm comprises three stages. The first stage is to shear the EPI and compute both defocus and correspondence depth cue responses. The second stage is to find the optimal depth and confidence of the responses. The third stage is to combine both cues in a MRF global optimization process [20]. However, compared to our work, their algorithm relies on shearing, so that objects that are too far from the main lens's focal plane will have incorrect depth estimations. They also rely on Lytro consumer cameras but we are able to apply in more general cameras.

Mansour, M. et al compared depth estimation performance of motion parallax and binocular disparity visual cues by using two different camera resolutions and feature points locations. Mansour, M. et al also proposed a method to overcome the limitations of the stereo camera by switching between motion parallax and binocular disparity [21]. Compared to our work, their work depends on motion parallax, and binocular disparity visual, which depends on at least two cameras. We are able to get depth information from still photos or a video from a single camera.

6. CONCLUSIONS AND FUTURE WORK

In this project, we proposed an algorithm which is designed to detect the distance of obstacles from a photograph or video from a single camera. The model identified distances for specific objects based on the size (dimension) of an object. We used different machine learning models to predict the distance, and compare the results of different models. We found that the object which is closer from the camera, the width percentage will change quickly. While it is farther, the change will become slower and even there is no change after a certain distance. It is obvious for small size objects. The smaller the object is, the earlier there is no change. For model choosing, we found the Polynomial Ridge Regression model (neither 3 Poly feature or 4 Poly feature) does not work well. The Linear Regression model works better than the Polynomial Ridge Regression model. It predicts correctly for close objects but for farther, it has more room to improve. The Random Regression and ElasticNet Regression model works best in these models.

In addition, one limitation in this project is that it does not suggest the sufficient threshold of training dataset. One thing we plan to improve is to evaluate the accuracy of the training process and collect more dataset to improve the accuracy.

As for the future work, we will investigate other machine learning algorithms to keep improving accuracy of the distance prediction. We also would like to explore the possibility of applying deep learning [22] in this problem domain. We will also continue to study the impact of camera parameters on predictions and select the most suitable camera to use.

REFERENCES

[1] Jackson, Philip C. Introduction to artificial intelligence. Courier Dover Publications, 2019.
[2] Levinson, Jesse, et al. "Towards fully autonomous driving: Systems and algorithms." 2011 IEEE intelligent vehicles symposium (IV). IEEE, 2011.
[3] Hubbard, Philip Martyn. "Collision detection for interactive graphics applications." IEEE Transactions on Visualization and Computer Graphics 1.3 (1995): 218-230.
[4] Mason, Timothy J., et al. "Application of ultrasound." Emerging technologies for food processing. Academic Press, 2005. 323-351.
[5] Sturm, Peter, and Srikumar Ramalingam. Camera models and fundamental concepts used in geometric computer vision. Now Publishers Inc, 2011.
[6] Wu, Chyuan-Tyng, et al. "VisionISP: Repurposing the image signal processor for computer vision applications." 2019 IEEE International Conference on Image Processing (ICIP). IEEE, 2019.
[7] Cao, Zhi-Le, Zhong-Hong Yan, and Hong Wang. "Summary of binocular stereo vision matching technology." Journal of Chongqing University of Technology (Natural Science) 29.2 (2015): 70-75.
[8] Izadi, Shahram, et al. "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera." Proceedings of the 24th annual ACM symposium on User interface software and technology. 2011.
[9] Remondino, Fabio, and David Stoppa, eds. TOF range-imaging cameras. Vol. 68121. Heidelberg, Germany: Springer, 2013.
[10] Konolige, Kurt, et al. "A low-cost laser distance sensor." 2008 IEEE international conference on robotics and automation. IEEE, 2008.
[11] Viola, Paul, and Michael Jones. "Robust real-time object detection." International journal of computer vision 4.34-47 (2001): 4.
[12] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. "Machine learning basics." Deep learning 1.7 (2016): 98-164.
[13] Haseeb, Muhammad Abdul, et al. "DisNet: a novel method for distance estimation from monocular camera." 10th Planning, Perception and Navigation for Intelligent Vehicles (PPNIV18), IROS (2018).
[14] Introduction to YOLOv5 Object Detection with Tutorial, https://machinelearningknowledge.ai/introduction-to-yolov5-object-detection-with-tutorial/,2021
[15] Yao, Weixin, and Longhai Li. "A new regression model: modal linear regression." Scandinavian Journal of Statistics 41.3 (2014): 656-671.
[16] Cheng, Xi, et al. "Polynomial regression as an alternative to neural nets." arXiv preprint arXiv:1806.06850 (2018).
[17] Misztal, Ignacy. "Properties of random regression models using linear splines." Journal of Animal Breeding and Genetics 123.2 (2006): 74-80.
[18] Hans, Chris. "Elastic net regression modeling with the orthant normal prior." Journal of the American Statistical Association 106.496 (2011): 1383-1393.
[19] Laina, Iro, et al. "Deeper depth prediction with fully convolutional residual networks." 2016 Fourth international conference on 3D vision (3DV). IEEE, 2016.
[20] Tao, Michael W., et al. "Depth from combining defocus and correspondence using light-field cameras." Proceedings of the IEEE International Conference on Computer Vision. 2013.
[21] Mansour, Mostafa, et al. "Relative importance of binocular disparity and motion parallax for depth estimation: a computer vision approach." Remote Sensing 11.17 (2019): 1990.
[22] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.