Stereoscopic Binary Obstacle Detection

Michael Bass, Drupad Khublani, Khaled Nakhleh, Venkata Pydimarri

Abstract

Obstacle detection and avoidance is critical for several autonomous systems and will become more important with future technologies. We have investigated stereoscopic binary obstacle detection to determine if an image contains an obstacle. To do this we modified a remote-controlled rover to perform stereoscopic image collection. We then used a convolutional neural network (CNN) to analyze the images. Our CNN was able to achieve 99.9% accuracy on the training data, and over 90% accuracy on validation data. Ultimately our experiments proved that a CNN was capable achieving high levels of performance after being exposed to a small subset of data that was previously unseen.

I – Introduction

Obstacle detection and avoidance is a critical component in autonomous vehicles (AV), driver assistance technologies, unmanned aerial vehicles (UAV), and mobile robots. In our project, we have assessed different methods for obstacle detection and analyzed the accuracy of various convolutional neural networks (CNN). We have approached the problem of obstacle detection using stereoscopic (SS) imagery and have performed binary classification on whether images contain an obstacle.

In addition to analyzing data, we decided to do manual data collection. By collecting our own data, we would be able to obtain the data we desired, as well as learn about the process of handling raw unfiltered data. To do this we modified a remote-controlled (RC) rover to include a SS camera setup. The rover then acquired images through a Python script as it was driven in different environments and in front of various obstacles.

Once the data was acquired we tested multiple CNNs to perform binary obstacle detection (BOD). We tried models of varying complexity and ultimately compared the accuracy of a complex and simple model. Once we determined that our CNN was successful at BOD we tried to answer other questions and comparison. First, we compared our SS model to a model that used monoscopic (MS) images. After this we evaluated the CNNs ability to detect obstacles that were not included in its training data. Finally, we exposed the CNN to a very small number of previously unseen images, and then reevaluated its accuracy.

Through our evaluations we gained several insights into the performance and abilities of MS and SS models performing BOD. To our surprise MS and SS performed nearly identical and were both very successful at detecting obstacles. However, when attempting to detect obstacles that were previously unseen, every model performed poorly. Lastly, we exposed the model to 3 images of an obstacle that it was pervious untrained on. We were very excited to see the models perform much better after being trained on only 3 additional images.

We think our results are promising and could lead to several future studies. The most exciting result is the performance of a model that included a small amount of previously unseen data. With this result, we believe models designed for facial recognition may be able to quickly identify subject of interests after only one or two images. This could be very impactful in several technologies that utilize machine learning.

II – Monoscopic vs Stereoscopic

Initially our project focused on performing BOD with SS images. However, as we were successful in our objective we realized the need for a comparison and naturally turned to MS images. This section discusses some of the differences between MS and SS images and how they are currently used.

Monoscopic (MS) and stereoscopic (SS) imagery are different methods used to acquire and analyze images. MS imagery uses a single camera, and therefore a single vantage point, while SS imagery uses multiple cameras to capture the same images from multiple viewpoints. MS images tend to require more processing due to having less data available, while SS is believed to having inherent benefits due to its resemblance to human vision.

Many techniques were developed to detect obstacle presence and depth estimation from a single MS image. This is a challenging problem, since local features alone are insufficient to estimate depth at a point, and one needs to consider the global context of the image. However, a discriminatively-trained Markov Random Field that incorporates multiscale local- and global-image features could be used [1]. It captures depth at individual points as well as the relation between depths at different points. But that approach is reasonable only for constrained environments. Also, every MS image method requires computational complexity that could be avoided by using SS methods.

SS vision is a technique from which one can extract 3D information and distance of a scene from at least two images (left and right). Conventionally, obstacle detection systems must identify and compare the position of obstacles. For the obstacle detection using SS images there are a few crucial parameters such as size, position of obstacle, accuracy and speed of the algorithm. SS imaging technique provide advantages in terms of accurate position estimation as well as fast computation with low cost solution.

Due to these differences, we have selected MS imagery to be a strong comparison for our SS solution. Intuitively we expected SS models to perform better due to their ease of extracting depth. Although both forms proved to perform equally well, we believe that a similar approach to estimate depth using a CNN may favor SS models.

III - Data Collection

a. *Educational Benefit*

We decided to begin our project by collecting data manually. Several machine learning projects and studies start with previously collected data. However, we decided to perform manual data collection to better understand the process and some of the issues that may come up. Through this process we incurred issues such as the rover burning a fuse and the expected tablet platform being underpowered. We overcame these issues and gained valuable experiences in working with a data acquisition system.

b. *Platform Construction*

To capture the SS images, we enhanced an RC rover, and designed a custom SS image capturing system. The rover was modified to include two 1080p webcams, one mounted on the left and right sides of the rover. First, we mounted two prototyping T-rails to the left and right sides of the rover. Then we mounted two perpendicular T-rails near the front of the rover point upwards. We attached a PVC pipe-based webcam mount to the upward pointing T-rails. The webcams were mounted about a foot and a half off the ground with each webcam pointed forward. The PVC webcam mount was designed to rotate the camera in 3 dimensions. The modified rover can be seen in Image 3.1 and 3.2. The rover's main function was to support, point, and move the cameras. The mounted cameras would then capture frames of the rover's path, with and without obstacles. Initially we planned on mounting a small Windows based tablet to the rover capture the images. However, the tablet proved to be incapable of powering both webcams. We resorted to connecting the webcams to a MacBook and walking behind the rover.

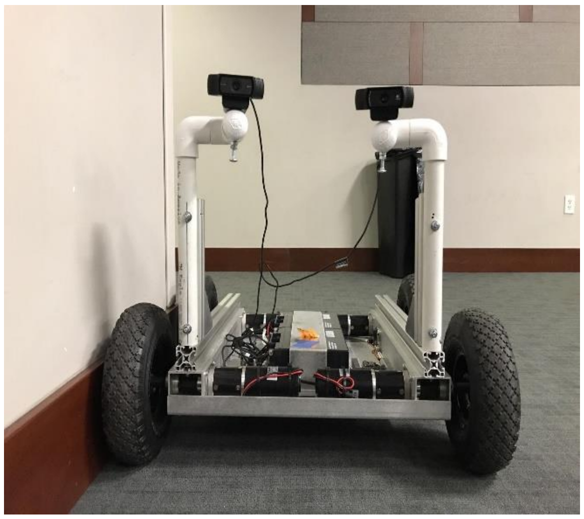
Image 3.1: Profile image of the modified RC rover. The webcams are mounted on the top left and right PVC arms.

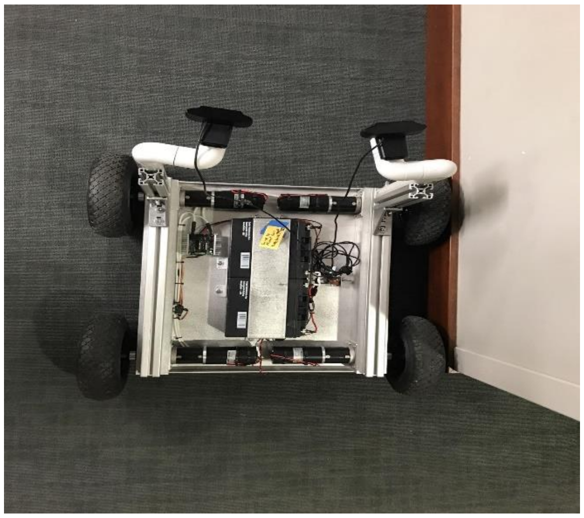
Image 3.2: Aerial view of the modified rover.

c. *Software*

To capture SS images, we wrote a Python script that utilized OpenCV [2], an open-source library released under the BSD license. OpenCV offers C++, Python, and Java interfaces, implements over 2500 algorithms, and is used primarily to enable machine perception, computer vision, and machine learning applications. Additionally, OpenCV implements object tracking, facial recognition, 3D model extraction, and object identification algorithms.

In our project we used OpenCV to access the two webcams, capture frames, transform the frames from RGB to grayscale, and save the images. We did not use any of OpenCV's algorithms or CUDA implementations. The frames were stored in separate obstacle and no obstacle folders, with each of these containing separate folders for left and right images. Therefore, we used a total of four folders. The frame capturing process was done using a single Python script. At the start of each run the Python script asked the user whether obstacles were being recorded and which run the user was performing. The script would then use OpenCV to continually capture frames from the two webcams using the *VideoCapture.read()* function. The frames were transformed to grayscale images using the OpenCV function *cvtColor()*. All frames were then saved, including frame and run numbers, into the designated folders using the *imwrite()* function. Full code implementation, along with comments, can be accessed on GitHub [3].

d. *Using the Rover*

Each continual data capturing iteration for the rover was recorded as a run. Runs were conducted in two separate locations on Texas A&M's campus; first the Zachry Engineering Education Complex's second floor side hallway, and second the Engineering Quad. Having a combination of indoor and outdoor environments would enable a more robust model due to variations in lighting conditions, surface reflections, building structures, and projected path perception. For the indoor portion, an empty hallway was chosen within the Zachry education complex to reduce foot traffic. For the outdoor portion, runs were made at the Engineering Quad due to its proximity to the Wisenbaker building where the rover could be repaired if needed. The Engineering Quad also offered standard objects for a set as obstacles (light pole, fire hydrant, etc..). All runs had the same lighting and weather conditions, which means no weather abnormalities could change model training behavior. Unfortunately, the location had a large amount of foot traffic during the data collection session. Some runs had to be restarted in order to ensure clean data samples.

Runs with obstacles lasted approximately 30 seconds to complete, while no obstacle runs lasted approximately 20 seconds. We used 10 objects as obstacles and placed them in front of the rover's path. The obstacles used varied in size, length, material, and reflectivity. This was done to ensure a wide variance in the data collected. The following items were used as obstacles:

1) Two metal water bottles
2) Matte black backpack
3) Bright yellow cotton-textured chair
4) Bright red metal fire hydrant
5) Two reflective silver trash cans
6) Matte black light pole
7) Human being
8) Reflective multi-colored sign
9) Thin bright orange road cone
10) Medium sized tree

Starting each run the cameras would begin to continuously capture frames for approximately 15 meters. With this approach the rover would capture obstacles from various distances while the rover approached. Runs with obstacles would end with the rover closely approaching the obstacle within 30 cm, and the cameras pointed closely towards the object. Our hope is that capturing obstacles from different distances would enable the model to learn an object's volumetric size and location within the space it occupies. Runs without obstacles would be stopped after the rover had proceeded an appropriate distance.

In addition to capturing images with and without obstacles, we decided that obstacles placed far away from the rover should not yet be considered obstacles. Due to this, these frames were labeled as having no obstacles. With this approach we hoped that the model would learn not to classify far objects as obstacles, but instead only detect an obstacle once it was within a certain distance.

e. *Data Collected*

A total of 34 runs were conducted, with 5,021 images pairs being captured and stored. 13 runs had obstacles in front of the cameras, while 22 had no obstacles. 1,824 image pairs contained obstacles, and 3,197 image pairs had no obstacles. Table 3.1 below provides an overview of each run, along with the number of stored frames.

Number of Images Captured Per Run

215	207	206	213	206	121
209	10	66	210	35	302
85	120	254	198	217	89
120	161	106	152	62	154
225	94	112	239	115	101
80	94	81	160		

Table 3.1. Depicting the number of photos captured per run. Runs with a white background did not have an obstacle, while runs with black background did. Non-underlined runs were done in Zachry, while underlined runs were performed in the Engineering Quad.

Images 3.3 and 3.4 below were taken during runs inside Zachary. In these images no obstacle has been placed in front of the rover. Images 3.5 and 3.6 were taken outside in the Engineering Quad. During this run a backup placed on a chair was used as the obstacle. The images shown were taken very close to the obstacle.

IV – Machine Learning

a. *Machine Learning Applicability*

A general approach to object detection is to locate and separate the obstacle from the background. On a broad-scale this is addressed by computer vision. Lately, image analysis has been addressed by machine learning approaches based on CNNs. Therefore, we used these approaches to address BOD.

Image 3.3: Image taken from the left camera while driving the rover down the hallway in Zachry. There is no obstacle in the image.

Image 3.4: Image taken from the right camera at the same moment as Image 3.3.

Image 3.5: Image taken from the left camera while driving the rover in the Engineering Quad. The obstacle is the black backpack placed on the chair. The rover was close to the obstacle when the image was taken.

Image 3.6: Image taken from the right camera at the same moment as Image 3.5.

b. *Models and Approaches Studied*

i. Computer Vision

Computer vision is a branch of artificial intelligence, where computer algorithms are designed to mimic human visual perception. Traditional computer vision algorithms use physical or geometrically-based techniques which analyze the details of images based on specific features such as nodes or edges. SIFT or SURF [4] algorithms combined with support vector machines are used to solve many computer vision problems such as facial recognition and object detection. However, for this project, computer vision is not a viable choice because they rely on handcrafted features and do not perform well in complex background.

ii. Siamese Networks

We had been encouraged to investigate the use of Siamese neural networks [5] for this problem. Siamese networks are used for one shot image recognition due to their fast computation performance. However, Siamese networks also have low accuracy rates. Since we had an adequate dataset of over 5,000 image pairs, we believed a CNN would be a better alternative.

iii. Inception Networks

There are many elegant image detection CNN architectures that exist and can be used for BOD. A model that was particularly appealing is the Inception model [6]. Inception is a module which passes the input layer through 4 different filters, such as 1x1, 3x3, 5x5 and max-pooling. Inception models perform well because it can learn different feature levels at the same stage. However, most of the architectures have multiple CNN layers (above 15) which can be computationally very expensive. Google Colaboratory (Colab) limits its user to 12 GB of memory, so using a model with 15+ layers on a dataset of 5,000 image pairs is not possible.

c. *Chosen Model*

Due to the memory limitations of Colab and the poor performance of Siamese networks, we decided to use a basic CNN. Although we tried several different architectures we ultimately analyzed 2 different CNN architectures. The first architecture, referred to as the complex model, consisted of 5 convolutional layers (each 32 to 64 filters wide), 5 max pooling layers, and 3 fully connected dense layers (500, 200, and 2 neurons). The second model, referred to as the simple model, had 3 convolutional layers (each 16 filters wide), 3 max pooling layers, and 2 fully connected dense layers (20 and 2 neurons). In both the architectures dropout layers were used to attempt to reduce overfitting. In our Python script we used the Sequential model from the KERAS framework. After performing a grid search relu was determined to be the best activation function. Since we are working on a classification problem, we chose the softmax activation function for the last layer in both models.

d. *Preprocessing*

Unlike many machine learning solutions, our solution required very little preprocessing of the data. The acquired images were transformed to grayscale images, and we reduced the resolution from 1920x1080 pixels to 320x180 pixels. Lastly, we stitched the two SS images side by side into a single image. This is the only preprocessing we performed.

e. *Binary Classification*

Since there can be many different obstacles that obstruct a vehicle's path we were uncertain how well our CNN would be able to detect obstacles. Therefore, we decided to keep the initial problem binary and focus on identifying whether or not an obstacle is present. Although it would be preferred to classify the obstacle, this would require much more data and neural network experimentation. After experimenting with different models and structures we were successful in achieving BOD.

f. *Stereoscopic vs Monoscopic*

After our success with SS BOD we felt that we needed a better benchmark for comparison. For this we decided to examine how a MS camera would perform. To do this we simply disregarded the data from the right camera, and solely used the data from the left camera. We used the same CNN structure, but due to using half of the input data, the MS CNNs had roughly half of the training parameters. We had hoped that the SS models would outperform the MS models, however the two performed nearly identical.

g. *Recognizing New Obstacles*

Next, we began to wonder how well our CNN would perform when exposed to new obstacles. One of the big problems with neural networks is they tend to perform very poorly when it must make an inference in a scenario that it has not been exposed to during training. If something is not present in the training data and is presented while testing, it is likely that the model will not recognize it and give bad results. So, to examine how the simple and complex models performed on new obstacles we removed 3 runs from the training data and used them solely as the testing data. Both models performed very poorly on the unseen testing data. The results aligned with our knowledge that CNNs require the training data to strongly resemble the test data to achieve high accuracy levels.

h. *Performance with Minimal Exposure*

After the model performed poorly on new obstacles we wondered how well it would perform with minimal exposure to the previously unseen data. This thought was motivated because many times it is not possible to get large amounts of images of a subject, but there are often some images available. With this in mind, we moved 3 images from the unseen test data that contained the new obstacle to the training data. Surprisingly it performed much better than the previous test, and the model performed only slightly worse than those trained on the complete dataset. We believe this is a strong achievement because it proves that even a small amount of data from a particular scenario is sufficient to correctly classify the image.

V – Results

To study SS BOD and the accompanying questions, we trained multiple models each for 50 epochs.

a. *Binary Classification*

Our first task was to determine if we could achieve BOD with SS imagery. We used 2,778 image pairs of our collected data and had a 5% validation split. This resulted in training using 2,640 images and validation using 138 images.

On the training data we achieved 77.8% accuracy after this 1st epoch, 93.3% after the 2nd epoch, and 96.2% after the 3rd epoch. Our model reached 99% accuracy after the 12th epoch and achieved over 99.6% accuracy on the training data. The validation data did not perform as well. By the end of the training the validation accuracy reached 91.06%. The results for the 50 training epochs are presented in Figure 5.1 below. From these results it is clear that our CNN was able to successfully classify images with and without obstacles.

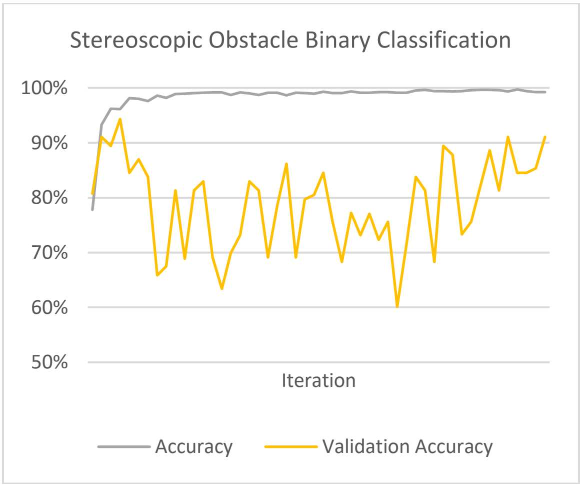

Figure 5.1. Testing basic obstacle binary classification.

Our validation accuracy was lower than expected, reaching 80% to 90%, but we believe that a much larger dataset and more training epochs may improve the validation accuracy. Another possibility is that the CNN only learns the data presented to it, and nothing else. Adding some gaussian noise may improve the performance in this scenario.

After successfully achieving binary classification, we tested how complex and simple models compared. Both models quickly achieved high accuracy percentages on the training data and displayed similar patterns in the validation data. These results are shown in Figure 5.2. We concluded that there is very little performance difference between the simple and complex models. For an embedded or restricted computing environment the simpler model would be greatly preferred due to the greatly reduced number of calculations.

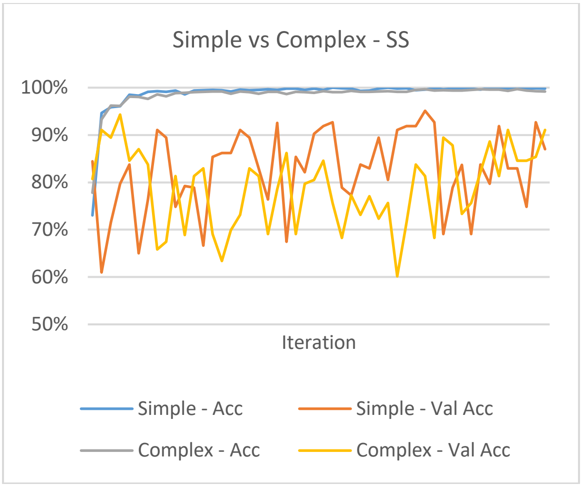

Figure 5.2. Comparing the performance of a simple and complex CNN.

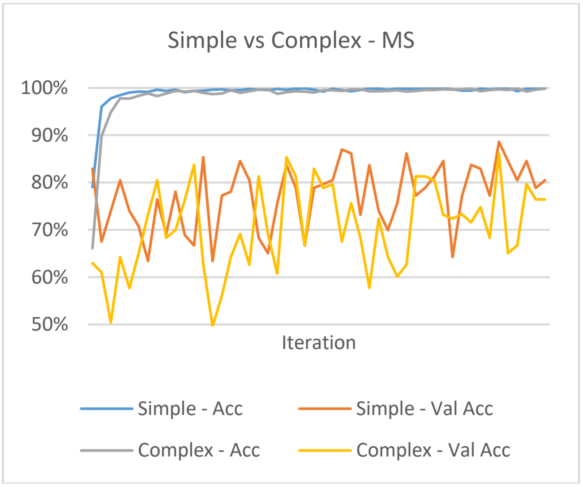

Figure 5.3. Monoscopic performance.

b. *Stereoscopic vs Monoscopic*

Our next set of experiments focused on the performance on SS versus MS BOD, and whether SS vision would have higher performance. The

MS imagery was obtained by using the left camera data of the SS images. Both SS and MS achieved high accuracy on the training data quickly. The MS validation accuracy was usually 5% to 10% less than the SS, but still achieved over 80%. We decided that SS and MS had very close performance and SS did not have a clear advantage in binary obstacle detection.

c. *New Obstacles*

Next, we tested the performance of the CNN to classify new obstacles that it had not seen previously. We tested this by removing three runs from the training data; two runs from the no obstacle data were removed, with one of those containing far obstacles, and one run with an obstacle was removed. All other data was used for training.

To our dismay the CNN performed very poorly. The SS version achieved 65% to 70% accuracy on the unseen obstacle while the MS version achieved 60% to 65% accuracy. This is shown in Figures 5.4 and 5.5. Cleary the CNN performs much better on the training data than when tested with a new obstacle. Figure 5.6 shows the SS and MS performance on a new obstacle on a single graph. From this graph it is seen that again there is little difference in the performance and patterns between SS and MS.

d. *Minimal Exposure*

After the CNN's poor performance detecting new obstacles, the last experiment we ran was to test the CNN's performance with a very minimal exposure to an obstacle. With just a little bit of data the CNN performed drastically better. The SS and MS models performed very similar again. The test performance was on average 20% more accurate than the new obstacle performance. The SS result is shown in Figure 5.7 and the MS result is show in Figure 5.8. Figure 5.9 shows a comparison between all forms of minimal exposure and new obstacle testing. This figure illustrates the improvement minimal exposure made in accuracy.

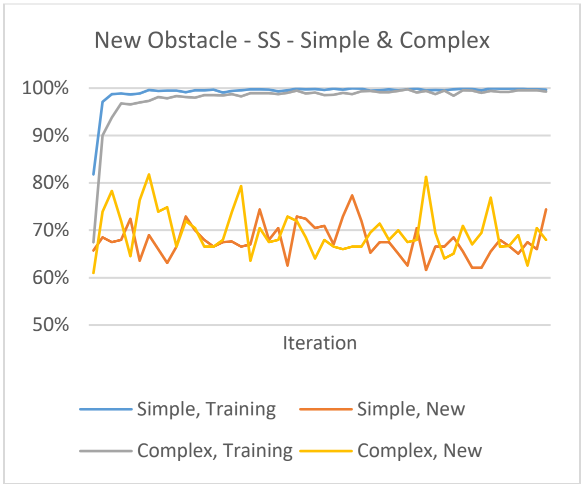

Figure 5.4. SS performance of CNN seeing a new obstacle.

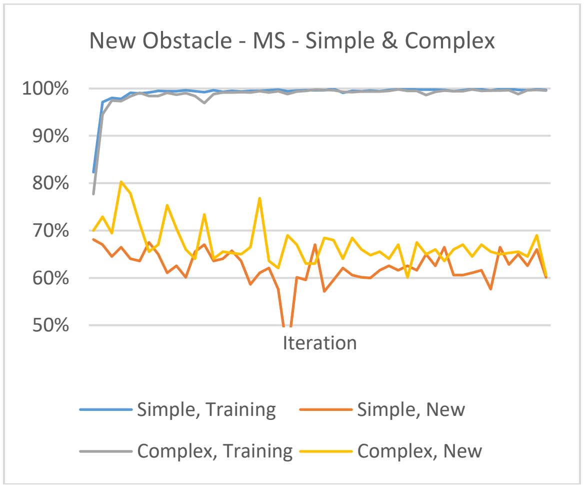

Figure 5.5. MS performance of CNN seeing a new obstacle.

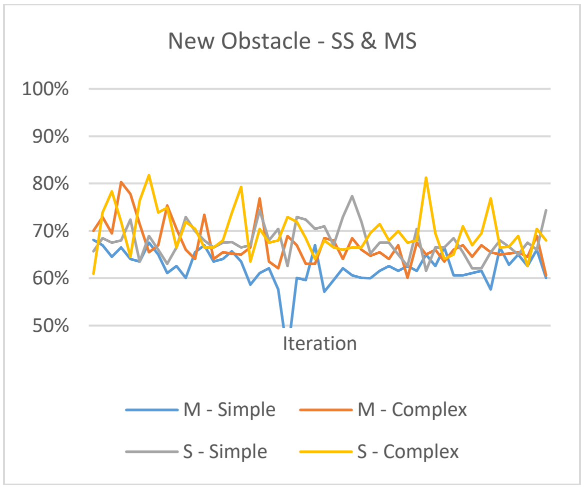

Figure 5.6. Performance of SS and MS on seeing a new obstacle overlapped on a single graph.

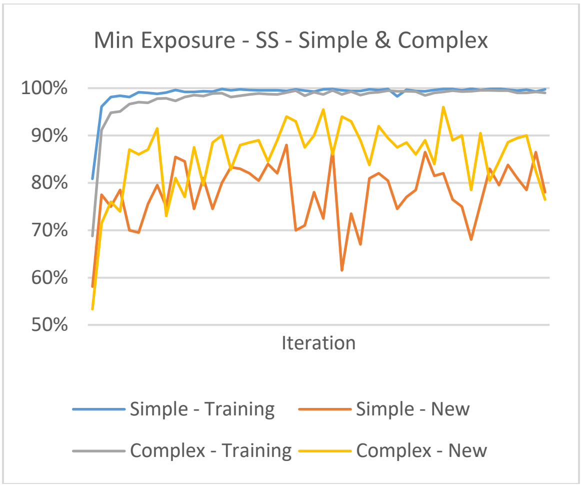

Figure 5.7. S performance of CNN seeing a new obstacle with seeing 3 images of the obstacle.

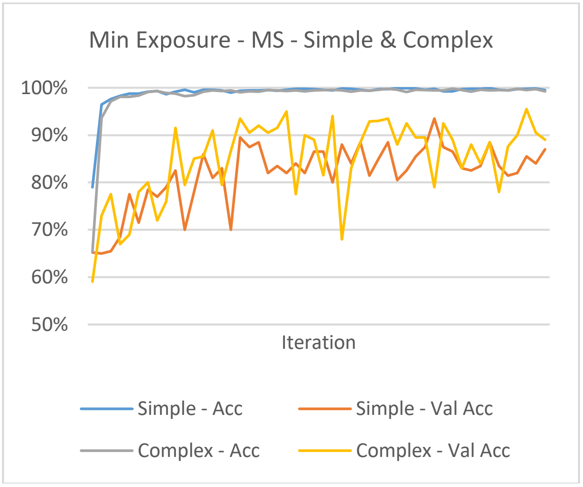

Figure 5.8. S performance of CNN seeing a new obstacle with seeing 3 images of the obstacle.

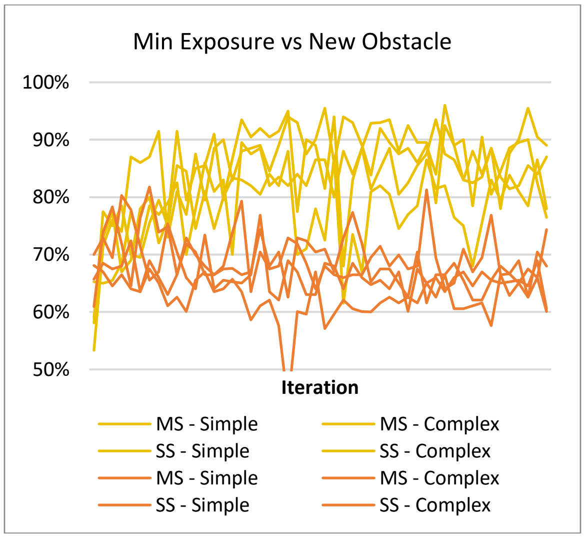

Figure 5.8. All SS and MS testing with minimal exposure and new obstacles. The Minimal exposure data is yellow, and the new obstacle data is orange. The minimal exposure performed drastically better.

VI – Impact

a. *Performance of Low Cost Hardware*

There is currently a strong movement within the automobile industry to move toward AVs. AVs are still in the research and development phase and therefore use very costly equipment to collect real time data to use as inputs in neural networks. One such example is Lidar. However, in our experiments we used very low-cost hardware in comparison. We only used two USB webcams to collect HD images, but due to the memory constraints of Colab, the resolution was reduced to 320x180. Although our project did not perform perfectly, producing such a good result with low cost hardware is definitely an achievement, and a step in the right direction.

b. *Limited Preprocessing*

Large amounts of preprocessing are typically required to be successful in most data science environments. It is often said that "garbage in, garbage out". In our project the raw images were preprocessed only by reducing the resolution, grayscaling the image, and placing the two images next to each other. There are many environments and situations, such as real time and embedded solutions, where large amounts of preprocessing are infeasible. The success that we had with limited preprocessing is promising for the given resource constrained environments.

c. *Monoscopic vs Stereoscopic*

During our experiments we tried several different configurations and strategies to extract meaningful comparisons. One of the most surprising results was the comparison between SS and MS models. First, we experimented with SS images to replicate human vision. Second, MS image models were tested to give us a baseline. To our surprise, there was not much difference between the results. We thought that by classifying objects at far distances as not containing an obstacle would strongly favor the SS models. However, SS and MS performed with almost equal levels of accuracy. If compared with human vision, it is known that individuals with only one eye are still able to recognize obstacles very well if there is something in the road. Although SS and MS performed very similarly here, there may be other applications where this is not the case. Continuing the comparison to human vision, we know that individuals with one eye do not judge distance as well as individuals with both eyes. Therefore, for estimating distance, SS model may perform much better than MS models.

d. *Performance with Minimal Exposure*

It was disheartening when the models performed very poorly to unseen obstacles, but it was very delightful when the models performed very well after being exposed to a small number of images. There are many applications where a good network has minimal exposure opportunities. One such example is looking for suspects with only a single picture. With success in this case, it paves the way for networks that need to be quickly trained/enhanced with very minimal data. This could be an exciting area to explore, especially in the case of autonomous vehicle where there can be a lot of change in surroundings and a very wide range of potential obstacles.

VII – Future Improvements

While working on the project we envisioned several improvements and methods that could carry the project forward.

a. *Depth Estimation Using CNN*

Depth prediction and distance estimation from images is an important problem in robotics, virtual reality, and 3D modeling of scenes. With the use of SS and CNNs, we believe that our models could excel in estimation the depth of objects when compared to SIFT and SURF implementations.

b. *Obstacles Detection Using Obstacle Images Outside of Any Environment*

Building upon the success of our model after minimal exposure, a model could be trained to recognize obstacles in a green screen environment. Then, the obstacles could be placed in real world environments, and the model could be evaluated on its performance.

c. *Night Time Detection*

We were very excited to produce promising results using low quality hardware. To extend our success to dark environments, a night vision camera could be used.

d. *Obstacles vs Intentional Objects*

It would be wrong to identify another vehicle as an obstacle. It would be another step forward to teach the model to recognize certain objects as obstacles, but other objects as acceptable.

e. *Obstacle Classification*

We thought about incorporating obstacle identification into our project. Obstacle identification can be done using traditional computer vision techniques [7]. We believe that a cleaner solution may be obtained using a CNN.

f. Grayscale vs Color

Currently we have used only grayscale images to test the quality of our model. One can further extend the domain of this project by choosing various color schemes such as RGB images.

VIII – Conclusion

In this project we examined the performance of CNNs using SS and MS data to perform BOD. To do this we first modified an RC rover to capture the SS images. We then fed the images through a CNN to infer whether the image contained an obstacle. We experimented with previously unseen obstacles, only to see the CNN perform poorly. Our last experiment focused on providing the CNN a small amount of data on the new obstacle, and to our surprise it performed much better.

We believe that our results could lead to some exciting new work, primarily in applying minimal new information to trained models. This area could have significant impact on machine learning systems. Along with our use of minimal preprocessing, we believe this platform could provide useful for real time embedded machine learning solutions.

Acronyms

AV	Autonomous vehicle
BOD	Binary Obstacle Detection
CNN	Convolutional Neural Network
Colab	Google Colaboratory
MS	Monoscopic
RC	Remote Controlled
SS	Stereoscopic
UAV	Unmanned Aerial Vehicle

References

[1] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng, "Learning Depth from Single Monocular Images", Computer Science Department, Stanford University, Stanford, CA 94305.

[2] Opencv.org. (2018). About - OpenCV library. [online] Available at: https://opencv.org/about.html [Accessed 11 Dec. 2018].

[3] Python image capturing solution. https://github.com/khalednakhleh/ECEN689Project2

[4] Andy Lee, "Comparing Deep Neural Networks and Traditional Vision
Algorithms in Mobile Robotics"

[5] Siamese Neural Networks for One-Shot Image Recognition. https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf

[6] Inception networks. https://arxiv.org/abs/1512.00567

[7] Understanding of Convolutional Neural Network (CNN) — Deep Learning. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148