Multi-Camera 3D Mapping with Object Detection, Similarity Matching and Depth Estimation

Emilio Montoya [1] [2], David Ramirez [2], Dr. Andreas Spanias [2]
Cochise College [1], SenSIP Center [2], Arizona State University [2]

***Abstract* - The purpose of this research is to develop a room mapping and an animal and human tracking algorithm. We will use a 360-degree camera to get a broad point of view (POV). Then a simulated robotic agent will move throughout the room to show a different POV where the 360-degree camera cannot see. With our two cameras we can run deep learning object detection to locate animals and humans. We then use ORB points for feature matching across the two cameras. By using two POVs, we can build a 3-D model of the room using Simultaneous Location and Mapping (SLAM). This 3D model will be analyzed for changes over time comparing the actual room before and after. This room mapping will increase the awareness of the environment.**

Index Terms: SLAM, Structure from Motion, Deep Learning, Object Detection, SSD, MobileNet, ORB, GMS, Omni-Directional Camera, DenseDepth, Midas

I. Introduction

The cat is up to trouble! Animals alter the environment in which they live. During the Covid-19 pandemic, there was a considerable increase in animal adoption from animal shelter facilities. Pets are living beings that require proper care and attention. However, some pets require more attention than others. Unfortunately, owners cannot always be around for their pets, it can be difficult to keep track of the activities their pets do throughout the day. Monitoring animal activity and behavior can help prevent accidents.

We want to implement a fusion of both pet tracking and room mapping. The purpose is to be able to track what the pet has done and where it has been. Even though there are GPS trackers for pets in the market, they do not work accurately indoors or at small scale. Nor does GPS have the ability to track what pets have done while they are on their own wandering around the house. Our system will be capable of combining images from the actual room to form a 3D model. This model will be generated by using a stationary camera and a simulated robotic agent that will provide more information to compare along with a different point of view (POV). Over time different views can provide contrast to show changes to the environment over time.

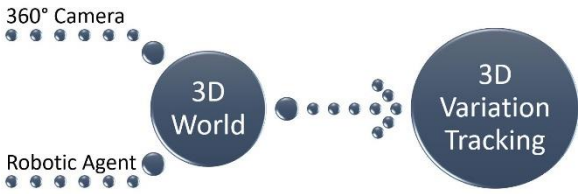

Figure 1 Simple representation of steps to take.

II. Related Work

Over the years, object detection has attracted the attention from many scholars. Y. Zou et al [1] found that Single Shot Detector (SSD) with MobileNet V2 classifier is faster to other convolutional neural networks (CNN) modules, such as the Faster Region-CNN (R-CNN) with Inception V2 classifier. Even though Faster R-CNN with Inception V2 was better at detecting test image examples, the SSD MobileNet V2 can identify and detect objects faster which is ideal for real time detection and recognition.

For real time detection on live video sources, methods vary from an input image, because with video the feature vector can be generated by the use of temporal information of the adjacent frame images. The application of real time detection has captivated the attention of various researchers. In one example F. Zhang developed a face expression recognition [2]. As a result, over the last decade multiple facial expression recognition programs were developed with a high recognition precision.

In addition, the Grid-based Motion Statistics (GMS) [3] algorithm enables us to be able to see and determine similarities between images. Feature matching is the most basic type of input for many computers vison programs. According to J. Bian, this technique *"*it encapsulates motion smoothness as the statistical likelihood of a certain number of matches in a region." [3]

III. Approach

The goal is to run one frame at a time from two live video sources through our detection and mapping system. We use Single Shot Detector (SSD) for the object detection, and MobileNet for classification. To avoid the costs of training a convolutional neural network (CNN), we will be using a pre-trained TensorFlow module. Additionally, we will use Midas Depth Estimation [4] neural network to determine how far away the object is from the cameras POVs. Also, we run a feature point matching algorithm to determine the similarity across both POVs, and by doing that the computer will be able to know

if the objects are the same or not. Finally, combine all SSD MobileNet, Midas depth estimation, and feature matching to determine the location of an object based on stationary camera and robotic agent location. From this work we expect to be able to generate a 3D model mapping a room.

For our purposes and resources, we will be using a cat to be the object to track in the room. We will also use a 360-degree camera as the main camera to gather images from the room. The object detection algorithm will be able to follow the cat's movements around the room. With [5], we will implement an object detection CNN to detect and identify objects, in our case cats and humans.

Even though the 360-degree camera will give a wide image of the whole room, it will have some blind-spots in which the cat might hide. To account for this, we will introduce a simulated robotic agent into the room. As a result, this will give another POV of the room, which will also provide more information on the blind-spots that the main camera might encounter. Since we will be using a simulated robotic agent in an in-door environment, a SLAM algorithm [6] [7] will be used to map the room and also give a precise location of the whereabouts of the simulated robotic agent as shown below.

Figure 2. A 3D reconstruction of the room is generated from 2D camera imagery [8].

We began with Object Detection. This is a critical part that would likely take the longest to set up and is also crucial for the overall project. This was chosen as the first step towards competition of the project. At the beginning of this step, we had two options: train a CNN on object detection or use an already trained object detection CNN. The first option was great because, we get to train it only on the objects we are interested in. However, it is also time consuming, and it requires a considerable amount of training images and a test to check its overall performance. On the other hand, the second option would not require any training whatsoever. The downside of this option is the object detection CNN is trained in a more generalized manner an alternate application. By taking into consideration what was mentioned above, we choose option number two.

We started to research TensorFlow object detection modules. We found a whole list of modules that included CenterNet [9] HourGlass [10], SSD [11] MobileNet v2 [12], Faster R-CNN [13], SSD [11] ResNet [14], and EfficientDet [15]. It is worth noting that, in this list of modules includes variations of the CNNs mentioned above with different version types and image resolution inputs. We then tested various modules of each type to determine which module would be ideal for a real-time object detection task. The speed performance results of the modules will be introduced on the result portion of the paper.

While we were using an object detection CNN from TensorFlow, it seems that there were some updates to the website. Unfortunately, these updates led to an error in the code because it could not find the specified module to download. Then we tried to run the code with other object detection modules, but the required input types were different from the previous module. Consequently, we had to re-define functions and adjust some of our code. Also, since we were downloading the object detection model every time we used it, there were some complications in which the computer could not locate the file for the module. We solved it by downloading the module and placing the file in the work directory. TensorFlow updates hyperlinks and application programming interfaces (API) without regard for past software releases. This causes a lot of headaches for thousands of people.

After the object detection algorithm was complete, we moved onto feature point matching. Feature point matching is the basic input of many computer vision algorithms. For this specific part we ran GMS [3]. This type of algorithm looks for similarities across images by focusing onto specific key points, ORB points [16], and its neighboring points. Then it looks at the other images looking for the ORB point at work to determine a true match or a false match. The purpose for this algorithm on this project is so that the computer will be able to determine if the objects across the multiple perspectives are the same or not.

Afterwards, we implement a depth estimation algorithm. This is required to determine the distance of the objects in the room from the perspective of the different cameras. For this we will be using a depth generator network for monocular depth estimation. This is mainly to estimate the distance of the multiple objects on the room, from the objects to the main camera or the simulated robotic agent. Moreover, this is a great addition to room mapping because with this tool now we can perceive where the different objects on a room are, not only the simulated robotic agent using a SLAM algorithm.

IV. Results

To run this experiment, we used an Ultrabook laptop with an Intel Core i7 8th Gen CPU @ 2.00 GHz, 16.00 Gigabytes of RAM. For the main camera we used a 360-degree Garmin VIRB Camera, and for the simulated robotic agent we used the Razer Kiyo Webcam. Moreover, to run all of the various software modules of this project, we created various virtual environments on Anaconda3 [17] for each specific feature. We also had to get Jupyter [17] notebook to run all the different codes involved. At the end, when all the features were running properly, we combined all the codes into one single application in which all features ran together at the same time.

A. *Object Detection*

From our findings on Table 1 and Chart 1, we learned that the best module for real time object detection was the SSD MobileNet v2 running at 0.1129 seconds. The next best option was the EfficientDet D0 512x512 running slower by 0.7 seconds. However, it was also more precise on its detections. It appears that there is a tradeoff between speed and accuracy when it comes to object detection modules. The faster the neural network runs, its precision is generally lower and vice versa.

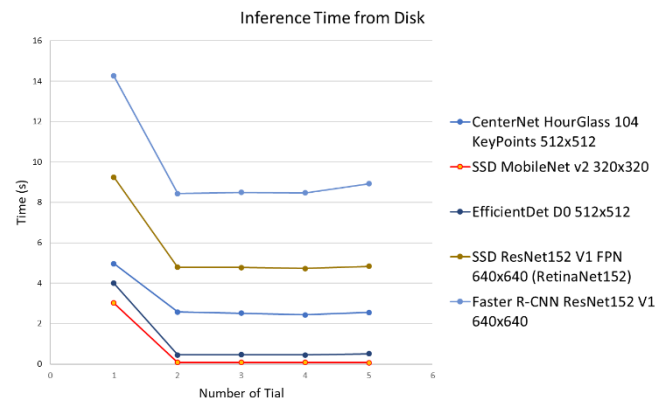

Chart 1. Comparison of CNN TensorFlow modules performing object detection from images.

Object Detection CNN Module Type	Average Inference Time from Video
SSD MobileNet v2	0.1129 seconds
EfficientDet D0 512x512	0.8080 seconds
CenterNet HourGlass 104 KeyPoints 512x512	5.6910 seconds
SSD ResNet152 FPN 640x640 (RetinaNet152)	14.8599 seconds
Faster R-CNN ResNet152 v1 640x640	22.9677 seconds

Table 1. Average frame per second of TensorFlow modules performing object detection from two different live video source.

Also, as shown on Figure 3 we found that background on images also affect the performance of the modules at detecting objects.

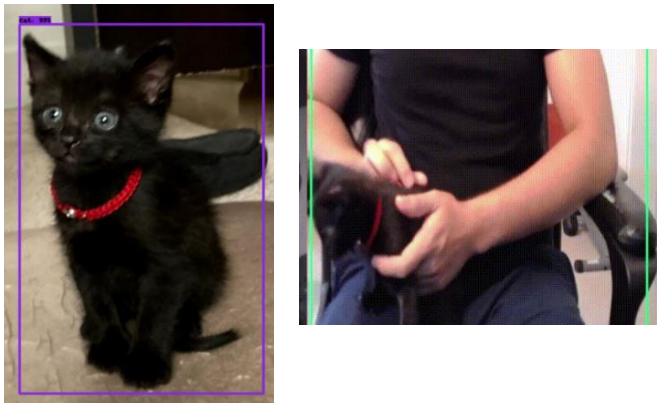

Figure 3. A black cat being detected and identified SSD MobileNet V2 (image on the left). With same color on background the black cat was not detected (image on the right).

B. *Feature Matching*

We could successfully implement GMS feature point matching across live video with an average time per frame of 0.3378 seconds. In Figure 4, we can see the different ORB points matches across different POVs

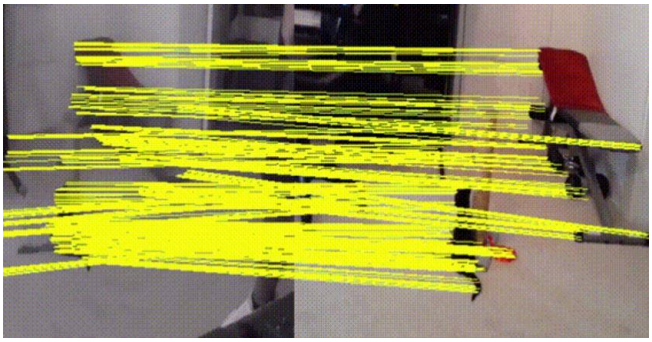

Figure 4. GMS feature point similarity matching (yellow lines)

C. *Midas Depth Estimation*

We were able to successfully run Midas depth estimator across live videos from the multiple cameras. The output image given by the neural network is in a grey-scale format. As shown in Figure 5, the objects closest to the camera are white and the objects further away are black.

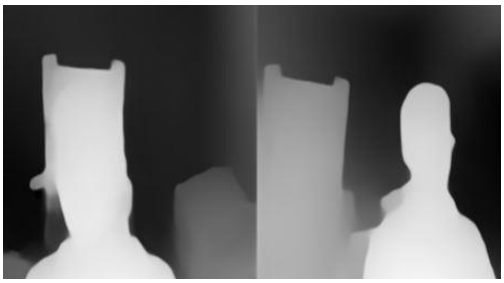

Figure 5. Midas depth estimator depth map output generated by RGB input data from multiple cameras

This algorithm was able estimate depth on video with an average of 0.8561 seconds per frame.

D. DenseDepth

DenseDepth [18] was a last-minute addition to the project with the purpose of generating a 3D model based on depth estimation. It is important to mention that this algorithm takes about 3.2 seconds to run per frame. However, it can be used to create a 3D model of a specific frame in particular as shown in Figure 6.

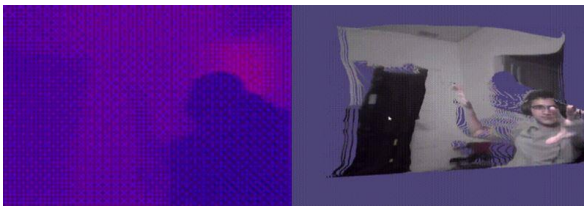

Figure 6. Depth map output (image on the left) combined with the RGB input image to generate a 3D moveable model (image on the right).

Findings

When we were combining the different algorithms to run at the same time, we found some interesting interactions with neural networks. On Table 2, we estimated a time when combining all algorithms. However, when we used the SSD MobileNet v2 combination, we ran it faster with an average of 0.8236 seconds per frame. On the other hand, the SSD MobileNet v1 combination ran slower with an average time of 3.8265 seconds.

Modules Used	Average Time from Video
SSD MobileNet v2	0.1129 seconds
MiDas v2	0.8561 seconds
Grid-based Motion Statistics	0.3378 seconds
Combined Time	1.3068 seconds

Modules Used	Average Time from Video
SSD MobileNet v1 FPN 640x640	1.6190 seconds
MiDas v2	0.8561 seconds
Grid-based Motion Statistics	0.3378 seconds
Combined Time	2.8129 seconds

Table 2. Expected time when combining SSD MobileNet v1 [19] FPN [20] and SSD MobileNet v2 different modules and running them at the same time

Acknowledgment

This project was funded in part by the NSF CISE award number 1659871.

References

[1] Y. Zou, et al, “Ship target detection and identification based on SSD_MobileNetV2,” 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), 2020, pp. 1676 – 80, doi: 10.1109/ITOEC49072.2020.9141734.

[2] F. Zhang, et al, “An Expression Recognition Methods on Robots Based on Mobilenet V2-SSD,” 2019 6th International Conference on Systems and Informatics (ICSAI), 2019, pp. 118-22, doi: 10.1109/ICSAI48974.2019.9010173.

[3] J. Bian, W. Lin, Y. Matsushita, S. Yeung, T. Nguyen and M. Cheng, "GMS: Grid-Based Motion Statistics for Fast, Ultra-Robust Feature Correspondence," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2828-2837, doi: 10.1109/CVPR.2017.302.

[4] Ranftl, Rene, et al. “Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer.” arXiv preprint arXiv: 1907.01341.2019.

[5] F. Chollet, “Basic Classification: Classify Images of Clothing : TensorFlow Core,” TensorFlow, 2017, www.tensorflow.org/tutorials/keras/classification.

[6] Y. Latif, “SLAM in the Era of Deep Learning,” Medium, Towards Data Science, 17 Nov. 2019, www.towardsdatascience.com/slam-in-the-era-of-deep-learning-e8a15e0d16f3.

[7] R Mur-Artal, JD Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. IEEE Transactions on Robotics, 2017.

[8] J. F. Henriques, “Mapping Environments with Deep Networks - Visual Geometry Group Blog.” Information Engineering Main/Home Page, 2018, www.robots.ox.ac.uk/~vgg/blog/mapping-environments-with-deep-networks.html.

[9] Duan, Kaiwen, et al. “Centernet: Keypoint Triplets for Object Detection.” Proceedings of IEEE/CVF Intenational Conference on Computer Vision. 2019.

[10] Melekhov, Iaoroslav, et al. “Image-based Localization Using Hourglass Networks.” Proceedings of the IEEE International Conference on Computer Vision Workshops. 2017.

[11] Liu, Wei, et al. “SSD: Single Shot Multibox Detector.” European Conference on Computer Science Vision. 2016.

[12] Sandler, Mark, et al. “Mobilenetv2: Inverted Residuals and Linear Bottlenecks.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[13] Ren, Shaoqing, et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” Advances in Neural Information Processing Systems 28. 2015.

[14] He, Kaiming, et al. “Deep Residual Learning for Image Recognition.” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016

[15] Tan, Mingxing, Rouming Pang, and Quoc V. Le. “Efficientdet: Scalable and Efficient Object Detection.” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

[16] Rublee, Ethan, et al. “ORB: An Efficient Alternative to SIFT and SURF.” The IEEE International Conference on Computer Vision. 2011.

[17] E. Nxumalo and K. O. Awodele, "A Review on the Use of Recommendation and Value Estimation Systems to Determine Protection Based Distributed Generation Penetration Limits," 2020 International SAUPEC/RobMech/PRASA Conference, 2020, pp. 1-6, doi: 10.1109/SAUPEC/RobMech/PRASA48453.2020.9041048.

[18] Alhashim, Ibraheem, and Peter Wonka. "High Quality Monocular Depth Estimation via Transfer Learning." arXiv preprint arXiv:1812.11941. 2018.

[19] Howard, Andrew G., et al. "Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications." arXiv preprint arXiv:1704.04861 (2017).

[20] Lin, Tsung-Yi, et al. "Feature Pyramid Networks for Object Detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.