Monocularly Generated 3D High Level Semantic Model by Integrating Deep Learning Models and Traditional Vision Techniques

Steven Alsheimer
Data Science
The CUNY Graduate Center
New York, United States
salsheimer@gradcenter.cuny.edu

Zhigang Zhu
Computer Science
The CUNY City College and Graduate Center
New York, United States
zzhu@ccny.cuny.edu

***Abstract*—Scene reconstruction using Monodepth2 (Monocular Depth Inference) which provides depth maps from a single RGB camera, the outputs are filled with noise and inconsistencies. Instance segmentation using a Mask R-CNN (Region Based Convolution Neural Networks) deep model can provide object segmentation results in 2D but lacks 3D information. In this paper we propose to integrate the results of Instance segmentation via Mask R-CNN's, CAD model Car Shape Alignment, and depth from Monodepth2 together with classical dynamic vision techniques to create a High-level Semantic Model with separability, robustness, consistency and saliency. The model is useful for both virtualized rendering, semantic augmented reality and automatic driving. Experimental results are provided to validate the approach.**

***Index Terms*—Monocular Depth Inference, Instance Segmentation, Scene Reconstruction, Deep Models, Augmented Reality.**

I. Introduction

We seek to infer an informative and robust 3D semantic model from a single RGB image or set of video frames. Performing these estimations seems impossible without a second image or without a stereo system. Yet humans do possess the capability to learn via navigation and infer not only depth but accurate and complete model shaping for classes of objects on a novel scene (Fig. 1).

On one hand, general classical SfM (Structure from Motion) methods require point triangulation from many pictures of varying angles followed by sparse bundle adjustment [12] to assemble a scene reconstruction that can then undergo semantic segmentation. On the other hand, to be able to infer depth using only a single image would require a large and varied set of ground truth data to train a deep neural network. Recent approaches to training such a network using a large real world dataset have used a self-supervised system. There are two approaches, taking either synchronized stereo or monocular video pairs from various scenes to train the monodepth networks [1]. Monocular video pair training, where an image is compared to the previously taken image is attractive due to its simplicity of hardware, but the stereo training methods where a right image is compared to its left counterpart have more simplistic depth networks for an increase in the training cost, with some occlusion issues [1]. Both monocular and stereo based training are available models for Monodepth2. While each method has its trade-offs, the overall self-supervised monodepth approach is a state of the art system.

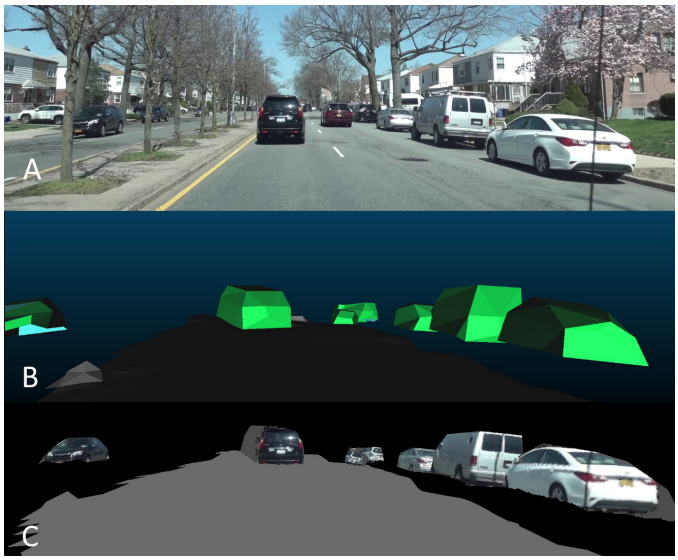

Fig. 1. A: Video frame when driving down a street. B: Its High-level Semantic Model (HLSM). C: Its HLSM Augmented Reality Version.

However, integration of multiple local depth maps created via a monocular depth inferencing method using Perspective-n-point (PnP) with Random sample consensus (RANSAC) and Global Optimization demonstrate the lack of consistency and noise with Monodepth2. One cannot readily tell where the cars, pedestrians and roads are by looking at this model, as shown in Fig. 2. This demonstrates the variance in the Monodepth2 depth prediction. the backbone of Monodepth2 (DispNet) used synthetic 3D models to train for disparity via rendered scenes [5]. Self-supervised monocular depth methods intend on using real images to solve for synthetic versus real life mismatches, but there is still a gap between self-supervised monodepth and the synthetic DispNet, as evidenced by the

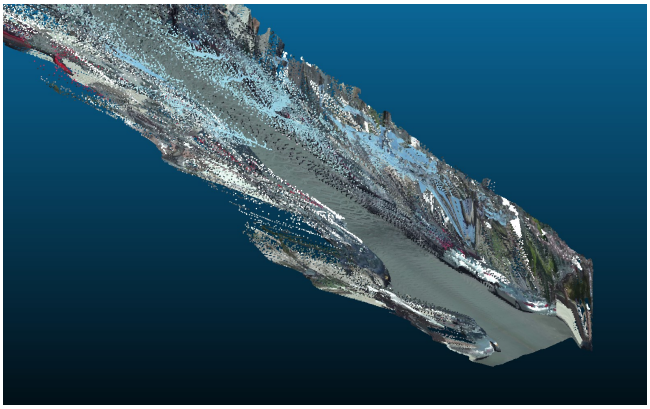

Fig. 2. Noisy point cloud from SfM based reconstruction.

variance and consistency issues in Monodepth2 prediction.

These inaccuracies and the limitations revealed by trying to get a scene reconstruction from this approach led us to the High-level Semantic Model (HLSM). We wanted a geometrically robust and separable model that would be more useful for either virtual rendering or autonomous driving, but still taken from a single lens. We decided to continue using the Monodepth2 network as our base model with the hope that we could ultimately use our HLSM in a refinement process as well as a virtual rendering pipeline for fast semantic depth predictions.

The HLSM will also have labels on it for the 4 primary components: the Cars, Road, Sky and Side-walls. The side walls give us the boundaries beyond which are the myriad of complex 3D models that often exist outside of the road environment. This model will have robust 3D shapes of the pertinent objects such as cars and roads. Having such a model is useful for resource selective rendering of a virtual scene. Where certain objects are rendered in lower quality or at a lower refresh rate to conserve computational expense. As well as an interactive augmented street scene, with information that could be useful for disabled people.

II. Related Work

A. Self-Supervised Monocular Depth Inference

Niantic and Toyota are two groups that have made impressive progress in the field of self-supervised monocular depth inferencing. Niantic's Monodepth system [1] and the Toyota Research Institute's (TRI) Superdepth system [3] both are SOA monocular inferencing systems. Both use the powerful DispNet [5] as their backbone. DispNet is a fully convolutional deep neural network that was created to infer disparity maps from synthetic 3D scenes and CAD (Computer Aided Design) models in Blender. [5].

TRI's Superdepth takes an input image and pushes it forward through the monocular depth network where a depth prediction is made. That depth prediction is then used to estimate what the image should look like from the other camera in the stereo trained system or the previous frame for the monocular video trained system. Then a series of Appearance Matching loss, Disparity Matching loss and, for Niantic, Left-Right Consistency matching loss functions are used to back-propagate and train the system. [2] [3]. The Left-Right Consistency matching loss is needed to resolve the issue of the network learning depth parameters for the opposite image, thus loosing input image alignment in the final product.

Niantic uses both the left and right images of the stereo training pair to estimate the depth map and predict the translated opposite image [2]. TRI only uses one image but uses a flipped disparity augmentation layer to handle the left-right consistency issue [3]. TRI Superdepth also uses a Sub-pixel convolution layer to allow for higher resolution predictions improving benchmarks. Both models have improved the SOA KITTI Eigen Test Split Benchmarks, but still have to close the gap to fully-supervised methods. In this paper we use the pre-trained Niantic Monodepth2 network out of the box since it was readily available on github [1], uses a similar architecture to Superdepth (which we want to study) and is ranked 16th on the KITTI Eigen Test Split Benchmark. We specifically use the stereo-1024x320 model.

B. Instance Segmentation

The matterport Mask R-CNN available on github is a widely used instance segmentation network [6]. It is trained off of the COCO dataset [19] and based on the Feature Pyramid Network with a ResNet101 backbone [14]. The matterport Mask R-CNN gives very accurate masking for cars and pedestrians walking on the street. It is however a computationally heavy process and forbids us from real time processing of the HLSM. Still, the performance benefits outweigh the computational expense.

C. 3D Object Detection and Mesh Deformation

Estimating the location of cars in the 3D world, including their orientation, is necessary for autonomous driving. 3D bounding boxes can be used to impute canonical 3D object models. Filder et. al [11] used a deformable cuboid model to estimate the 3D bounding boxes for the location of cars from a singular monocular image. In the end of the paper they augment CAD models into those bounding boxes as a solely visual representation of the effectiveness of their methods. However, for more accurate car shape alignment we can look towards Takeo Kanade's paper on the subject. There they trained a Random Forest Classifier to choose vehicle landmarks, these were then used to create a hypothesis of the cars position in 3D space via a minimization technique. In that paper the car can also be deformed to better minimize the reprojection error. For our paper we will be using a pre-trained Neural Network from Pirazh Khorramshahi to choose vehicle landmarks and then will use the PnP method to align the cars with further steps for shaping [18]. This network was chosen since it fit our needs of attaining 20 vehicle landmarks and was available on Github.

III. Methodology

The HLSM Pipeline comprises the following stages (Fig. 3). An image is fed into both the Monodepth2 [1] Inference and

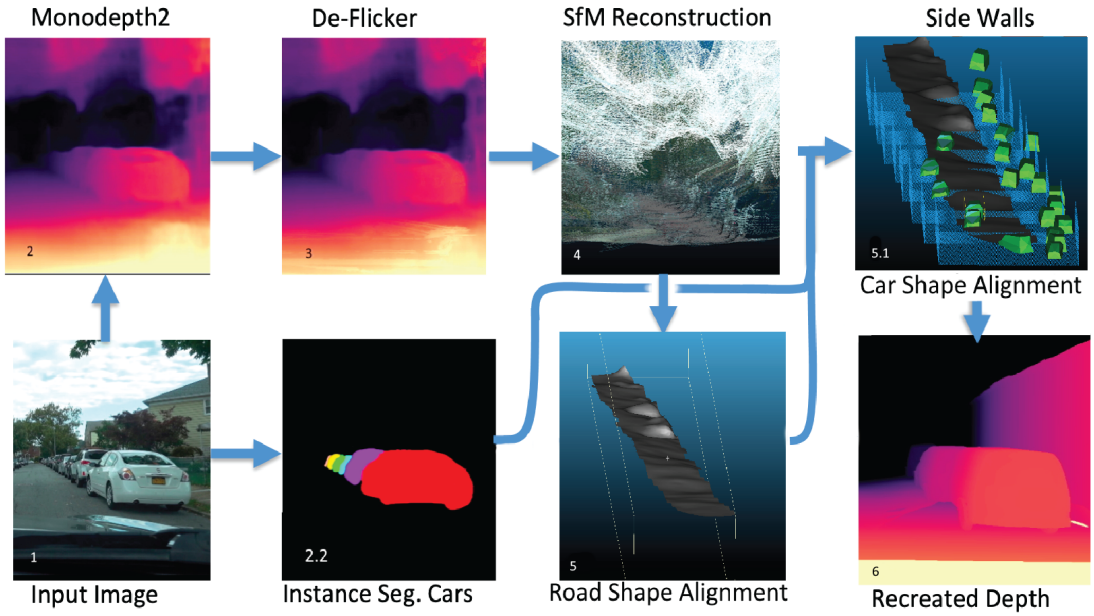

Fig. 3. Left to right shows the multiple stages to get the HLSM.

Mask R-CNN Instance segmentation networks [6] as well as Pirazh Vehicle Landmark Network [18] to generate its depth map and car detection mask map, respectively. After the depth map passes through a de-flicker stage and a SfM reconstruction stage, the two maps are integrated to generate a 3D semantic segmentation of road, cars, and other parts of the scene. Then the models of cars, road, roadside and sky are combined to create the high-level semantic model. For demonstration purposes, a clean HLSM depth map is generated, and a HLSM + Monodepth2 prototype combination depth map is also provided.

A. RGB Image to Depth Map

For the development of the pipeline, we used a video of a car driving down a block. For our data collection we used a Sony RX100V using manual settings, at 24 mm focal length (35 mm equivalent) and shot the video at 30 FPS. All videos were taken around various areas around Queens, NY. These videos do not include any pedestrians, intersections or turns. As of now the system is limited from turning, in the future we will incorporate GPS and/or gyroscopic data into the system to create a more accurate reconstruction system addressing these issues. Using the Monodepth2 pre-trained model (stereo 1024x320 model) [1] we get the depth output and the visualization counterpart. Monodepth2 takes 0.03 s to complete a singular depth prediction. A magma color-mapped depth map is the visualization counterpart to the depth prediction.

B. De-Flickering

In testing we noticed camera jitters induce hallucinations of objects. By this we mean Monodepth2 predicts an object close to the camera that does not exist. It is possible that the KITTI dataset (Monodepth2 used for training) had a more stable environment for the camera, or road. When the camera we used was jolted by a bump on the road the jitters from the camera caused the Monodepth2 network to erroneously predict depth. It would often hallucinate a large blob close to the car (Fig. 4). In some instances the hallucination hung around for a few frames greatly affecting reconstruction accuracy.

To resolve the flickering without adding additional and expensive hardware we created a software to minimize the effect. We used the visual counterpart of the depth prediction, taking frame by frame pairs. We use homographies to provide temporal matching of the input image series and depth images, finding areas to be excluded in reconstruction. Below is the algorithm:

- If the current depth image and the previous depth image show no significant differences over large areas, both images are real.
- If the current depth image shows a large area that does not match the homography transformed previous depth image the different area is classified as not-real and purged.
- If the previous depth image also had a flicker detected the frame four images ago is used.
- To avoid pixel drift every ten frames the homography prediction is reset to the Monodepth2 predicted depth image.

In the case that multiple frames have a consistent flicker issues arise. In those cases we back off 4 frames as in the algorithm. This was determined experimentally as enough frames to handle long flickers, yet not too long that the camera has moved too far for homography to work without significant pixel drift. We call pixel drift the corresponding corruption of quality as pixel information changes and errors propagate over the course of de-flickering. This is a result of the current image being partly based on the previously altered depth image, it will therefore carry errors. For this reason we also reset the

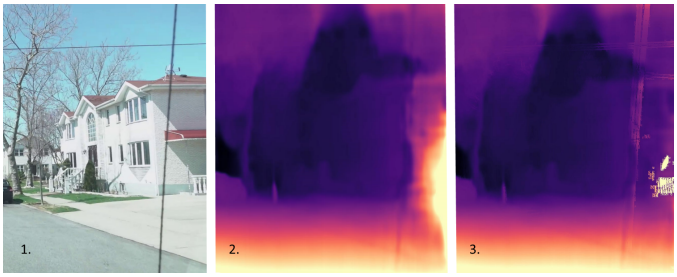

Fig. 4. From left to right: an input image, its depth map from Monodepth2 that has a hallucinated blob on the right and the de-flickering result.

image every 10 frames to take not from an altered depth image but from the original depth map to curtail these corruptions. The OpenCV library was used to implement this stage [7]. We only use the de-flickered depth images to ascertain where on the Monodepth2 depth map data should not be included in the reconstruction phase.

C. SfM Reconstruction

Though Monodepth2 is a singular frame depth prediction, we can use SfM to enforce consistency and reduce noise on longer depth prediction sequences. Here we use the previous output and textbook techniques via multiple video frames and Solve-PnP-RANSAC from OpenCV [7]. The standard approach is to make the first depth prediction and use it's 3D output as the base. PnP then uses the 3D-2D corresponding pairs of the first color images pointcloud and the previous 2D image to determine the camera pose. Since we have depth data here we do not perform triangulation adding to a global point set for future registration. Instead we perform the PnP iteratively on each pair of images, this yields a dense output. We then continue this for each (t-1,t) pairs of images. We use Open3D's Global Optimization method (Levenberg–Marquardt algorithm) to get a final reconstructed 3D model. Implemented with OpenCV and Open3D [8] [7]. We do allow for a non-Globally optimized model, and a Colmap type model as options. As mentioned previously this output is very noisy, however it is needed for the next steps (Fig. 2).

D. Road Shape Alignment

Monodepth2 generates a relatively noisy road, especially the part of the road immediately in-front of the car. This issue combined with general errors in prediction can make the road depth prediction inaccurate, and after reconstruction attempts inconsistencies of road depth between different frames becomes apparent. To solve this we generate a road surface mesh to create a consistent and flattened singular road surface. We first take the SfM reconstructed point cloud data from the previous step and with the Camera Extrinsic information provided by PnP we begin scanning along the road surface. The user can set the number of intervals, the more intervals the more accurate the road surface (to a limit) with added processing time. We settled at 25 intervals for our system (i7-4790k, 32 GB RAM, RTX 2080). For each interval a camera is chosen, in this cameras frame the small lower region right in front of the car (a 30x400 region was used here, user can change these dimensions) is scanned to collect the road height. That road height is used as a local constraint for finding a plane via the three point RANSAC method, the plane found should be the road. This is ensured via a small distance to plane threshold per inlier (0.5) and a total inlier threshold of 1000. The important assumption here is that roads are planes at a very local scale. Finally we perform alpha shape surface reconstruction to get our road mesh. The figure below shows the road shape alignment scanning method described above. Parameters are found experimentally as the optimum between accuracy and cost.

E. Car Shape Alignment

1) Primary Car Hypothesis: In order to get geometrically proper car shapes we use a car shape alignment method similar to the one Takeo Kanade proposed [17]. However, we use a neural network and PnP for the hypothesis and minimization method respectively. We first run the Matterport instance segmentation network over the RGB video frames to detect individual cars. Using Monodepth2's depth data we cut off cars beyond a distance threshold, 40 meters in our case. This parameter was found experimentally since beyond this distance car transformation approximations tend to be inaccurate. These parameters can be changed by the user. Masked images cropped to focus on the car are then sent to the Pirazh Car Landmark NN for landmark point detection. There are 20 potential landmarks to be detected. This NN does not take into account where landmarks are in relation to each other in its first stage. It's vehicle orientation predictions (2nd stage) are sub-par. We therefore built a method to detect and filter landmarks depending on the order of special landmarks (licence plates, and mirrors). We now have our landmarks filtered for car orientation.

We have 3 canonical CAD models with their landmarks labeled in 3D coordinates. The 3 types of canonical models are Sedan, Van and Truck (more to be added later), they are standard with the HLSM pipeline and do not change per scene. Using PnP we minimize the reprojection of the 3D Car Landmarks against the 2D landmarks found. We used the SolvePnP-EPNP(EPNP) [7] method here as opposed to the more typical RANSAC method used for the SfM Reconstruction part of the pipeline. EPNP gave the best results (most cars placed with smallest reprojection error) out of all the available methods. Out of the 3 types of cars we select the car type based off of minimal reprojection error. Next we allow 3D landmark morphing to better fit the unique and individual cars in the dataset. The amount of morphing is constrained, which is why initial car type estimation is important. Symmetry constraints are then applied to this model to ensure geometrically proper car shapes.

To decide if a cars shape alignment is good we need more than just reprojection error. If 3 landmarks have a low reprojection error we found that more often than not the fit was not as accurate as a higher reprojection error with 9 points.

We devised the formula $Threshold > \frac{rep-error}{n-landmark-points^2}$ to handle the reprojection error and number of landmark points relationship. The threshold value of 3.6 was determined experimentally by maximizing the number of aligned cars but minimizing poor fits. To finish up we texture the cars with their associated car image for an augmented reality system. In the case that a car is not imputed, likely due to too much occlusion in the RGB images we have a back-off car alignment hypothesis method described below.

2) Back-off Car Hypothesis: The back-off method does not use Car Landmarks, we assume a car we see is facing away from us along the Z-axis. Usually this assumption would be inaccurate, but since our model right now is constrained in straight lines its a sufficient plan B. (Fig. 3 row 2, col 2). In this case we chose a step of every 20 frames to reduce cost. Instance segmentation and the distance threshold are the same as the above method.

The Monodepth2 depth map is then referenced over the cropped region containing a certain instance segmentation of one of the cars. We assume that car as viewed from the point of view that the car will be facing away from the users car, creating an L shape as viewed from the top. Where the back and side are showing to some extent. With that assumption we take the 6 points defining this L shape (top and bottom) to build a 3D Bounding Cube where a car should be located. A dot product between the direction of the cube and the actual coordinate location of the max Z point of the L shape to properly orient the box. We label key-points of the boxes and CAD models and use SVD (Singular Value Decomposition) and the basic RMSE (Root Mean Square Error) minimization technique used in the closed form solution to ICP (Iterative Closest Point) to find the transformation matrix to place and orient the CAD model of the car within the determined bounding box [15].

We then iterate through the video sequence, using every 20 frames and continue to impute the cars. The 20 frame interval was considered a compromise between getting redundant car alignments, compute cost and getting the most amount of unique cars, balancing against the aforementioned distance threshold. To ID the cars using color of the car given by the RGBD segmented area around the car. We use this ID and a centroid distance checker to locate duplicate car imputations and combine redundant car models. There is no shape morphing in the back-off method.

F. Side-Walls

The noisiest and most inaccurate parts of the Monodepth2 prediction are the areas outside of the roadway (Fig. 2). To work around this (since these areas are not important to an automated car, and would be cumbersome for an AR system) we create a side-wall and floor to bound the entire High-level Semantic Model within the confines of the road environment.

Using the car models and the road from the previous steps to algorithmically identify the boundaries of the road we then add a few meters to either side of the road beyond that to define the road area. This is because the road shape found in Section III-D often neglects parts of the road where cars are parked. We build side walls for demonstration purposes. The walls are usually generated in 10 segments, this number is arbitrary and optional. The boundary information is useful for the augmented reality scene boundary.

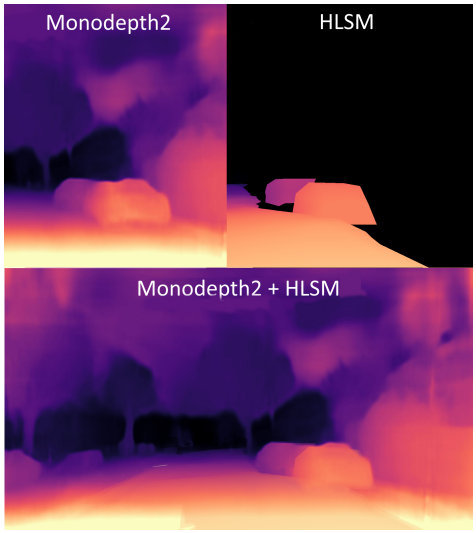

Fig. 5. Top shows the Monodepth2 and HLSM Output. Bottom is a prototype of the combined depth information.

G. Final Output

We give two final outputs. One of the final outputs includes the green colored or textured cars, the side-walls colored in blue, the road colored in grey and a Sky Dome colored in Sky-Blue to finish off the augmented reality environment. We also generate depth maps from the new model to show the geometric robustness of the HLSM system. We provide a prototype of the HLSM + Monodepth2 Depth Map, no KITTI benchmark testing has been conducted yet. We ultimately believe we can use this more geometrically robust model with less variance and accurate shapes of important 3D objects such as cars and the road to refine the output of the Monodepth2 network and improve the depth estimation benchmarks (SILog and Stereo KITTI) for the cars and road, in particular [16]. In its current state the model is not ready to be tested on KITTI dataset and we do not have Lidar equipment for ground truth testing, therefore at the moment observations and qualitative reports of the system are used as report cards for the system. Reprojection error and other similar metrics are used wherever possible. In the future when the model is tested on KITTI data we can test it against the SILog and Stereo benchmarks.

IV. Results

Fig. 5 demonstrates the Monodepth2 stereo 1024x320 output, and the HLSM output on the top, for the same scene and camera pose. We then provide a prototype of the depth averaging that can be used to improve the Monodepth2 predictions. We can see how the more robust and proper car and road shapes affect the Monodepth2 output. The windows of the car, the road area right at the bottom of the frame and the boxy

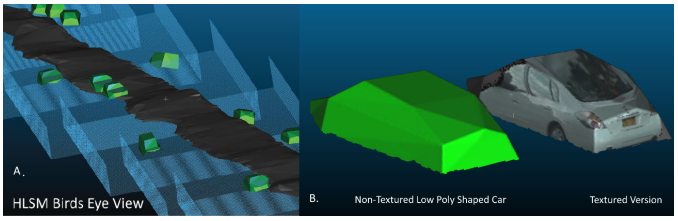

Fig. 6. A: A Birds Eye View of the whole HLSM model with the cars highlighted in green. The Sky dome is omitted here since it would block the screen. B: A close up of a shape aligned car and its textured version.

definitions of the cars are all improvements in consistency and smoothness.

Fig. 1B shows the HLSM model in its separable augmented reality format where the various classes of the model are given their distinct ID color tags. This model format is useful for 3D detection, interaction, and piece-wise rendering as described previously. Fig. 1C Shows the Augmented Reality final version with textured cars. It is a more forensic core model. Fig. 6A shows a birds eye view of the reconstructed core model scene in orthographic view. The Shape Aligned cars are highlighted in green, this view demonstrates the success of HLSM in creating a separable and clean street scene with proper road object shapes. Fig. 6B also shows a close up of a shape aligned car and its textured version.

V. Conclusion and Future Work

We have shown that the combination of classical computer Vision algorithms and more advanced deep learning frameworks can resolve the noisy, inconsistent output of the Monodepth2 system to create a High-level Semantic Model (HLSM) with separability and saliency in mind, with the help of Instance Segmentation via Mask R-CNN. We have shown that all of this can be done without needing any external knowledge of the scene that is not provided from the Monodepth2 system. This, we hope, means that the Monodepth2 system can be retrained with the HLSM on a second pass to be refined and be made more robust, with better shape definition, and less noise. The primary goal of our work is to improve the KITTI depth prediction benchmarks on elements specific to the road environment (Vehicles, Pedestrians and the Road). By retraining the system in a Self-Semi-Supervised manner, we may be able to achieve improvements in these areas, or at the least create an inference model to perform this same HLSM creation at a fast rate. The system as it stands right now can be used in the creation for augmented reality focused on street scenes, this could be useful for people with disabilities or for mapping projects.

For future work on this system we intend on incorporating moving cars. We also want to incorporate GPS data and/or gyroscopic data and Kalman filters to handle turns better. As well as looking for ways to make the system more efficient, including alternatives to the expensive Mask R-CNN models. In the far future we eventually intend to add pedestrians.

Acknowledgment

The work is supported by the National Science Foundation through Awards #1827505 and #1737533, the Air Force Office for Scientific Research (Award #FA9550-21-1-0082) and the Intelligence Community Center of Academic Excellence (IC CAE) at Rutgers (Awards #HHM402-19-1-0003 and #HHM402-18-1-0007).

References

[1] C. Godard, O. M. Aodha, M. Firman and G. Brostow, "Digging Into Self-Supervised Monocular Depth Estimation," In Proc. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 3827-3837

[2] C. Godard, O. Mac Aodha and G. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency", CVPR, Honolulu, Hawaii, USA, 2017.

[3] S. Pillai, R. Ambruş and A. Gaidon, "SuperDepth: Self-Supervised, Super-Resolved Monocular Depth Estimation," In Proc. International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 9250-9256

[4] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos and A. Gaidon, "3D Packing for Self-Supervised Monocular Depth Estimation", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2020.

[5] N. Mayer, E. Ilg, P. Hausser, P. Fisher, D. Cremers, A. Dosovitskiy, and T. Bronx, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation", IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016.

[6] W. Abdulla, "Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow", Github repository, Github, [https://github.com/matterport/Mask_RCNN], 2017.

[7] ITseez, *Open Source Computer Vision Library*, Github repository, Github,[https://github.com/itseez/opencv], 2015.

[8] Q.Y. Zhou, J. Park and V. Koltun, "Open3D: A Modern Library for 3D Data Processing", arXiv:1801.09847, 2018.

[9] D. Haggerty et al, *Trimesh*, [https://trimsh.org/], 2019.

[10] M. Matl, *Open Source Python Rendering*, [https://github.com/mmatl/pyrender], 2018.

[11] F. Sanja, D. Sven and U. Raquel, "3D Object Detection and Viewpoint Estimation with a Deformable 3D Cuboid Model", In Proc. NeurIPS 2012 Conference, Lake Tahoe, NV, USA, 2012.

[12] J. L. Schönberger and J. Frahm, "Structure-from-Motion Revisited," In Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 4104-4113

[13] S. McCann, "3D Reconstruction from Multiple Images", [Online] Available: https://cvgl.stanford.edu/teaching/cs231a_winter1415/prev/projects/CS231a-FinalReport-sgmccann.pdf, 2014.

[14] K. He, G. Gkioxari, P. Dollár and R.B. Girshick, "Mask R-CNN", CoRR, Volume abs/1703.06870, 2017.

[15] B.K.P Horn, H.M. Hilden, S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices", J. Opt. Soc. Am. A, pages 1127-1135, OSA, 1988.

[16] M. Menze and A. Geiger, "Object scene flow for autonomous vehicle", In Proc. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 3061-3070

[17] Y. Li, L. Gu and T. Kanade, "A robust shape model for multi-view car alignment," The IEEE Computer Vision and Pattern Recognition (CVPR) 2009, Long Beach, CA, USA, October 2019.

[18] K. Pirazh, K. Amit, P. Neehar, R.S. Saketh, C. Jun-Cheng and C. Rama, "A Dual-Path Model With Adaptive Attention for Vehicle Re-Identification", The IEEE International Conference on Computer Vision (ICCV) 2019, Seoul, Korea, October 2019.

[19] T.Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C.L. Zitnick, P. Dollár, *Microsoft COCO: Common Objects in Context*, In Proc, Computer Vision – ECCV 2014, Zürich, Switzerland, 2014. Springer International Publishing, Pages 740-755.