3D Object Measurement using Depth maps by leveraging MiDaS/Depth Anything v2

S Kiran
— Harikrishna S—
Kanishka S

Abstract

The Project Aim's at Identifying/Configuring the dimensions of the "Parcel's/Package's" present in a given image along with some reference height to identify simple parcel shapes such as Cuboid's, Cube's ,Cylinders and Spheres which would be useful for segregation the packages for efficient Delivery/Transport. We leverage Depth Anything V2 model for depth estimation and combine it with classical computer vision techniques for object segmentation and measurement by using pinhole camera approximation and relative depth map, A histogram-based approach is used to identify thresholds for object masks, aided by Gaussian smoothing and KMeans clustering to ensure accurate segmentation. The obtained result's had an average error of 8% in the measurements with accurate segment masks, a Key observation was that there was sudden increase in measurement error when under low-light and unevenly distributed surface's (greater than 15 %).

1. Introduction

Measuring objects in 3D from a single RGB image is vital for robotics, Military, AR, and inventory management, where depth sensors or multiple cameras may not be feasible. The problem tackled by the project is the Inventory Management problem, i.e. to classify objects for delivery so that they could be distinguished from each other, We specifically tackle's the problem of classifying package's based on their size so that they could be efficiently transported to the desired location by maximising the amount of packages that could be carried in a single visit.

1.1. Novelty

The Existing Solution to the above problem relied mainly on using existing YOLO(v-X) Models to segment the image into segments by leveraging feature extracting algorithms of ResNet and VCG backbone, to extract the required keypoints required to find the dimensions of the image. Our

Figure 1. Comparison Between Models of MiDaS with Depth Anything v3

Idea solves the problem by using Existing MiDaS-3.1 and Depth Anything V2 to identify the depth map of the given object and apply our method of histograms to identify the keypoints required for finding the threshold points of the binary mask of the object's and then applying pinhole camera approximation to measure the real world x, y coordinates from image coordinates and leverages relative depth to identify the depth of all the objects

1.2. Overview

Converts the RGB image into a BGR numpy array which is then passed through the Depth Anything v2 Hybrid model which returns normalised grayscale depth map which we pre-process and pass it through the histogram to get the arrival at and get the minima's of both smoothened and DoG histogram to which we apply K-Means Clustering (with K as the number of object's) and get the center's of the respective labels as the threshold points for each of the objects present in the image, then we apply pinhole camera approximation to translate the x, y image coordinates to real world coordinates and then calculate using linear transformation of depth map. pipeline is available in Fig(1).

2. Methodology

2.1. Depth Estimation

Leveraged Depth Anything v2 over MiDaS.v3 due to higher flexibility and accuracy of the models, Moreover compared to MiDaS.v3 depth anything v3 provides significantly better depth resolution which would be essential to calculate depth's accurately.

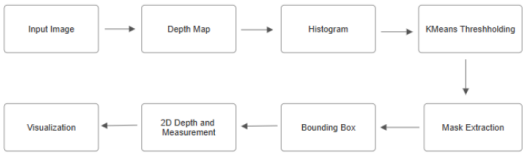

Figure 2. Project PipeLine

Model	RMSE	Rel. Depth Estimation
MiDaS v3.1	—	low(80%)
Depth Anything v2	0.056	High(95%)

Table 1. Benchmark RMSE comparison of monocular depth-estimation models

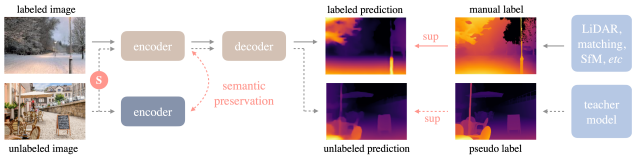

Figure 3. Structured Pipeline of Depth Anything v3

2.1.1. Pre-Processing

We are using Gamma stretching of with $\gamma = 1.03$ Also, contrast coefficient c_r 1.04 for adding a linear boost over the gamma boost.

2.2. Segmenting

First, the image histogram is obtained for analysis. This is done by converting the image to grayscale and segregating the image into various intensity bins(0-255), which are equivalent to various elevation or depth bins. The aim is to segment it by taking multiple thresholds based on the number of objects and the background. A Gaussian ($\sigma = 1.89$.) is passed through the histogram to smooth it and reduce variations.

2.2.1. Thresholds

Threshold values are found by finding minima's since it's evident that the object thresholds occur at the peaks of the plot due to the uniform nature of the objects correlating to clustered point's with the same intensities, and chosen on the basis of the number of objects and relative height. The minima are arranged in ascending order and then clustered using the K-means algorithm. DoG($(\sigma 2_1 = 3.76) - (\sigma 2_2 = 1.8$.)is applied to the histogram. Then it is scaled down and minima are obtained, again repeating the process of K-means. The midpoint of both the thresholds obtained using minima and DoG is taken as the final threshold.

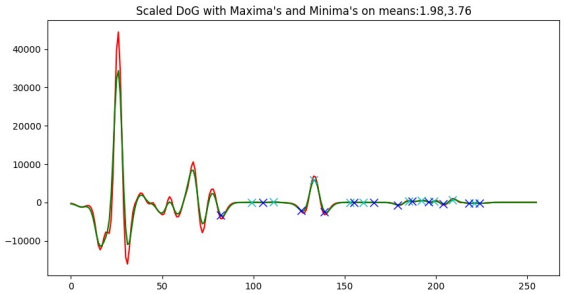

Figure 4. (3DoG) in Red, (3.2DoG) w/($\sigma = 1.5$) in Green

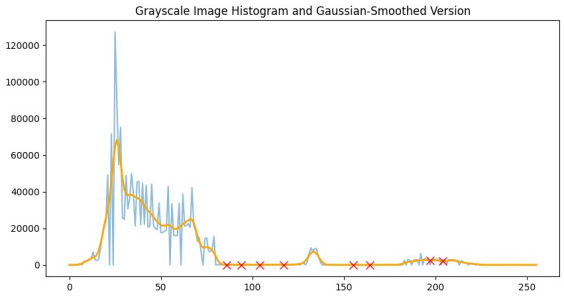

Figure 5. Hist(Image) vs Smoothened Hist(Image) w/Minima's

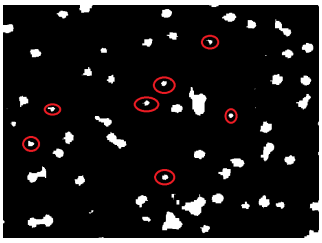

Figure 6. Image highlighting smaller area components by using connectedcomponentswithstats() courtesyStack Overflow

2.2.2. Binary masks

Using the threshold values, binary masks are obtained in which the background is black and the object/Segment is in white. The masks are chosen such that there will be N masks for N objects. An example is shown referencing a cylinder, The Ground Truth segment is obtained by finding the minimum intensity level of the clustered centers which would provide perfect foreground background seperation. each threshold point remove's a singular object from ground truth, We dynamically remove the unnecessary options by using connected component's and removing the smaller area as demonstrated in fig(x)

2.3. Bounding Box

ShiTomasi is used for corner detection and is implemented using OpenCV's goodFeaturesToTrack function. The 2 points are chosen to find the bounding box, such that they

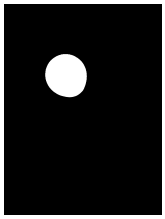

Figure 7. Left Object(cylinder) Mask

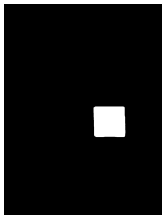

Figure 8. Right_Object(Cube) Mask

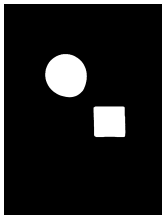

Figure 9. Ground Truth

are the lower right-most and top-left-most corner points of the rectangle. The points are found by calculating the max and min of both x and y coordinates of all corners.

$$\begin{aligned} Top\_left &= (min(corners[x], max(corners[y]) \\ Bot\_right &= (max(corners[x], min(corners[y]) \end{aligned} \tag{1}$$

2.4. Measurement

2.4.1. Length and Width

Taking inspiration from ray tracing algorithms, we implemented a similar model involving just a linear transformation by taking a rectangular pyramid approximation. This is needed to get multiple similar triangles, which are required for scaling the values. It is calculated by obtaining the change in x and y coordinates used for calculating the bounding box. Here, we take the view viewport dimension to be the same as that of the sensor. It is used to get the x distance on the sensor and further scaled using the camera height and focal length.

$$\begin{aligned} sensor\_x &= (\Delta x / totalpixel[x] * viewport\_dist[x] \\ sensor\_y &= (\Delta y / totalpixel[y] * viewport\_dist[y] \\ x &= (cam\_height / focal\_length) * sensor\_x \\ y &= (cam\_height / focal\_length) * sensor\_y \end{aligned} \tag{2}$$

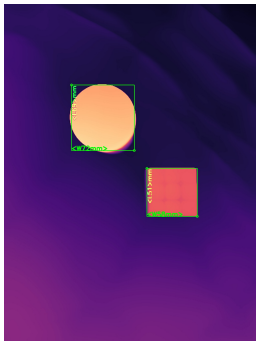

Figure 10. Final 2D Measurement Obtained

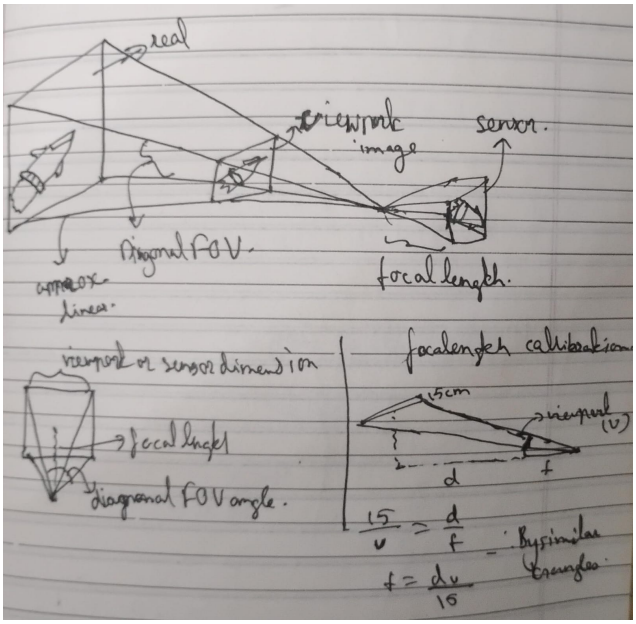

Figure 11. Used Approximation

$$f = \frac{\sqrt{w^2 + h^2}}{2 \tan\left(\frac{\theta}{2}\right)}$$

$$\theta = diagonal\_FOV$$

(Focus calculation when using adaptive Focus or unknown focus with w and h being sensor dimensions.)

2.4.2. Depth

Depth is calculated by using each of the segments of the respective objects to compute the relative mean depth, which could be linearly interpolated by setting a reference mean, which is calculated by finding the mean of the region present between the origin and the top left corner of the highest bounding box and using (3) to arrive at the depth.

$$cal_H = (ref_H)\frac{x - ref_G}{D_h - ref_G} \tag{3}$$

$$ref_H : provides\ the\ reference\ height$$

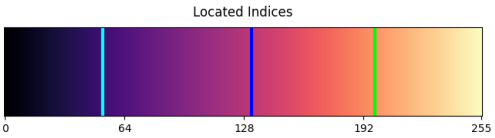

Figure 12. ref_G : cyan, ref_H : blue, cal_h : lime

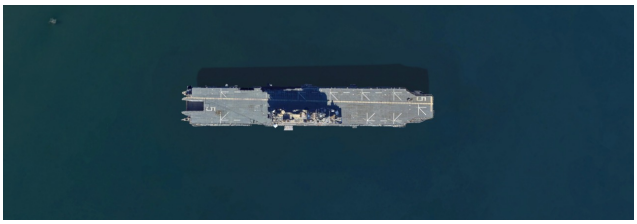

Figure 13. Helicopter Carrier aerial/satellite image

$$ref_G : depth\ value\ ground$$

$$D_h : Depth\ value\ reference$$

$$cal_h : calculated\ height$$

3. Results and Application's

3.1. Results

Final Website with UI implementation: `https://iitkee604.streamlit.app//`. `https://github.com/kanishkas01/iitk.ee604`

3.2. Sample Results

Image	OrgDim (lxwxh)	CalDim (lxwxh)
Sample 1	155x60x35—105x58x62	152x59x35—109x71x52
Sample 2	65x65x110—53x53x53	69x72x97—58x51x53
Sample 3	100x79x52—155x60x35	98x83x73—140x53x35

Table 2. Quantitative comparison of depth estimation methods on sample images and their error calculation

3.3. Design Choices

3.3.1. Bounding box using Shi Tomasi vs Hough Transform

Shi Tomasi was preferred for getting the edges by joining corners as it was more clean and simpler to compute compared to Hough transform which also requires computing the canny edges.

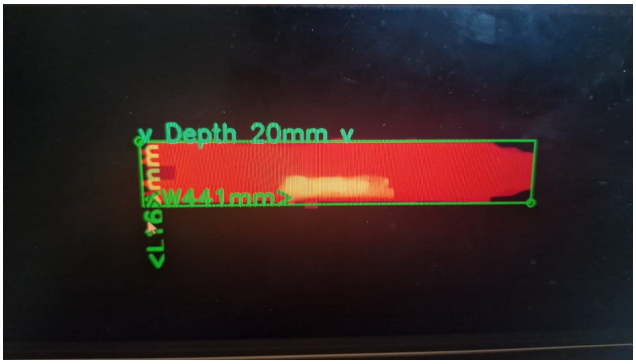

Figure 14. Provide's Approximate Dimension Estimate's(mm to m) of the Carrier, accuracy increases with correct specification of camera used

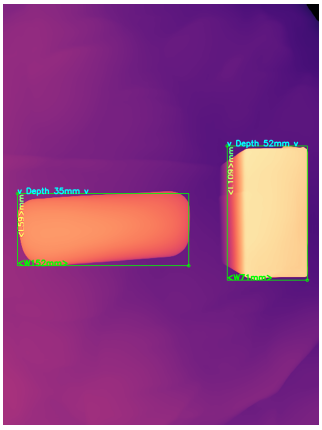

Figure 15. Sample Result (1)

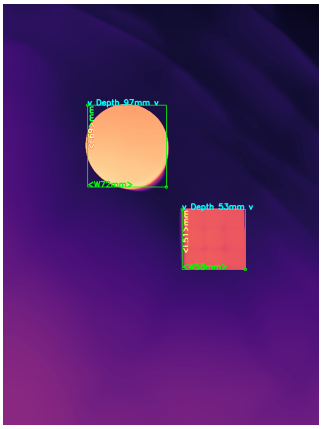

Figure 16. Sample Result(2)

3.4. Improvement

3.4.1. Using DOG

Finding blobs in the histogram and applying otsu threshold at the initial location's of the blob will give better results

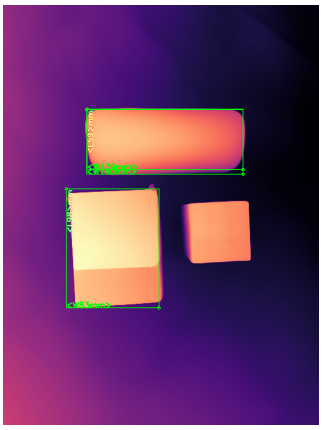

Figure 17. Sample Result(3)

compared to Minima method. Since it dynamically arrives at the best threshold value within the given window for best seperation.

3.4.2. Better Geometric Approximation

Instead of taking a square pyramid, considering an approximation like thin lens or hyperbolic edges would give better results.

3.4.3. Using both Shi Tomasi and Hough Transform

Shi Tomasi and Hough Transform can be implemented by using an extension cum merger algorithm on Hough output and combining it with Shi output by taking only the parallel outputs.

3.5. Applications

3.5.1. Inventory Management

Large scale E-commerce companies can easily use it to classify deliveries dynamically while the packages are moving in the conveyor belt.

3.5.2. Military

It is a Robust algorithm to find the size and height of aircraft carriers and ICBM. It will be useful to differentiate real from mock up infractructure and also gain critical specifications.

3.5.3. AR and Rendering

Provides distinct scales which would prove to be essential for generating 3D object models when given an image, to be used in Augmented Reality devices for various purpose's.

3.5.4. Colab Notebook

link of notebook: `https://colab.research.google.com/drive/18b0tTmatku_wsdoQ9BZfkosRgU7D4Rl4?usp=sharing`

References

1)https://github.com/DepthAnything/Depth-Anything-V2
2)https://stackoverflow.com/questions/42798659/how-to-remove-small-connected-objects-using-opencv
3)https://raytracing.github.io/books/RayTracingInOneWeekend.html