Journal of UAB ECTC
Volume 15, 2016
Department of Mechanical Engineering
The University of Alabama, Birmingham
Birmingham, Alabama USA

COMPARISON OF STOP SIGN DISTANCE DETECTION USING 2D AND 3D CAMERAS

Mahbubul Islam
Kennesaw State University
Marietta, GA, USA

Dr. Kevin McFall
Kennesaw State University
Marietta, GA, USA

ABSTRACT

In this paper we illustrate traffic sign detection and distance measurement by monocular and stereo camera. We have introduced this method to detect traffic signs and calculate their distance by applying a Haar cascade classifier which is unambiguous and obtains optimally calculated results. We have obtained distance by mathematical calculations from 2D images obtained from a monocular camera. 3D traffic stop signs can be detected using a Point Feature Histogram descriptor from a point cloud map. The distance to the detected traffic signs can be measured by a depth map obtained from a stereo camera. After proper calculation, we can draw a clear idea comparing stop sign detection in 2D and 3D cameras or stereo cameras.

INTRODUCTION

This paper introduces the detection of traffic stop signs and compares the distance measurements obtained by 2D and 3D cameras. Conventionally we can detect traffic signs or any object from captured image data as per our given reference or given classifier. But if we determine the distance from the object to the camera, then it will be much easier for us to use the result in any real-life application and so on. We can get mathematical results of distance from a 2D image, and we get analytical values for distance from 3D images by measuring a depth map in a point cloud environment. By comparing values for stop sign distance detected by 2D and 3D cameras, we can make a clear point that a calculated depth map result can be more accurate than a theoretical result in a real environment.

2D OBJECT DETECTION

Two dimensional images contain RGB color values for every pixel. To detect a sample object from an image, we need to contain the 2D colored picture RGB values in every single pixel, and the computational intensity of feature calculation makes the detection process slow, complex and computationally expensive. This kind of problem was addressed by Paul Viola [5], who introduced Haar-like features, a method of rapid object detection from digital images.

Haar-like Features: The Haar-like features process is divided by three key contributors.

Integral Image: Allows the process to compute in a quicker and optimized way by detector sampling. To integrate an image, a rectangle feature is used to compute very rapidly using an intermediate representation for image.

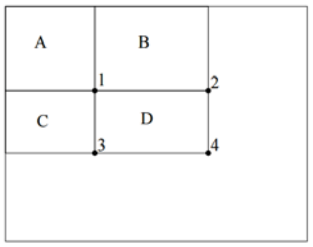


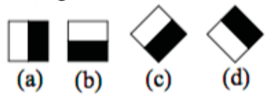
Figure 1: Representation of Haar-like feature

$$W(D)= L(4) + L(1) - L(2) - L(3) \quad (1)$$

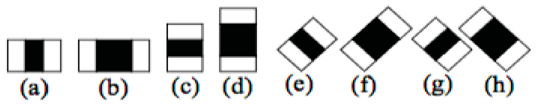
Equation (1) represents W(D) as a weight of the image and L as specific point area. The sum within D is computed using four lookups, as it contains the sum of all pixels on its upper-left region.

Learning Classification: Allows the system to select a minimum number of visual features from a large set of pixels and yields the process to compute the detection optimally [2]. OpenCV can be used for object detection using a cascade classifier that uses the following Haar-like features:

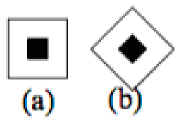
1. Edge Feature



2. Line Feature



3. Center-surround Feature



The feature used in a particular classifier is specified by its shape and position within the region of interest and by the scale (this scale is not the same as the scale used at the detection stage, though these two scales are multiplied). For example, in the case of the third line feature (2c) the response is calculated as the difference between the sum of image pixels under the rectangle covering the whole feature (including the two white stripes and the black stripe in the middle) and the sum of the image pixels under the black stripe multiplied by 3 in order to compensate for the differences in the size of areas. The sums of pixel values over rectangular regions are calculated rapidly using integral images

Cascade: Allows background regions to be discarded based on the integral image and the learning process. The overall form of detection process generates a decision tree by boosting process that we call a ‘cascade’ [1]

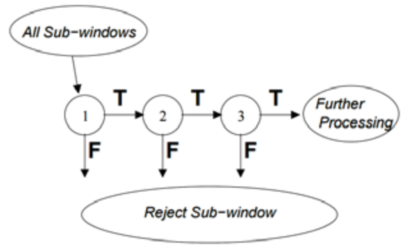


Figure 2: representation of a cascade decision tree

A positive result introduces the evaluation of the second classifier, which is adjusted to achieve high detection rates. A negative result leads to immediate rejection of images. Currently the process uses Discrete Adaboost, and the basic classifier is a decision tree with at least 2 leaves.

Training Cascade classifier

A classifier is trained with a few hundred sample views of a particular object, called positive example images containing traffic stop sign, that are scaled to the same size (say, 20x20), and negative examples - arbitrary images of the same size to distinguish from a particular object. In this way, the classifier can build a decision tree of the classifier for the image environment.

Cascade stages are built by training classifiers using Discrete AdaBoost. Then the system needs to adjust the threshold to minimize false negatives. In general, a lower threshold yields higher detection rates from positive examples and high false position rates from negative examples. After several stages of processing, the number of sub-windows reduces radically.

After cascade classifier training is fully accomplished, it can be applied as a given reference to detect an object of interest from input images. The classifier shows as “detected” if the region is likely to show the object, and “not detected” otherwise. Classifier is also designed in such a way that it can be easily “resized” in order to find the objects of interest at different sizes, which is more efficient than resizing the image itself.

2D OBJECT DISTANCE

The central imaging model of cameras projects a point P in real space to point p in the image plane using the focal length f, as appears in Figure 1. Points in space map from (X,Y,Z) to (x,y) according to

$$x = f\frac{X}{Z} \text{ and } y = f\frac{Y}{Z} \tag{2}$$

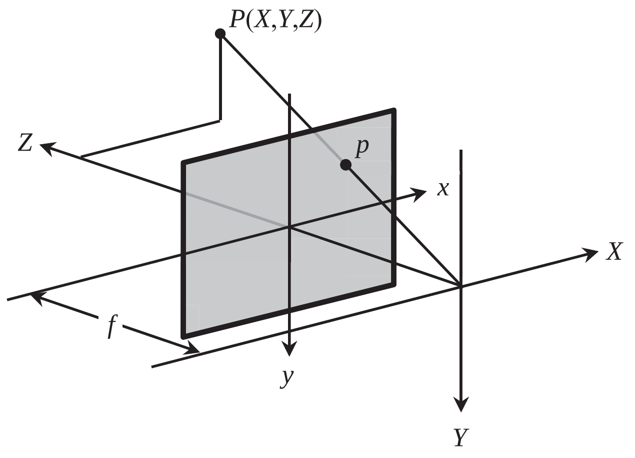


Figure 3: Geometry for projecting a point *P* in *XYZ* space to its location *p* on the *xy* camera image plane.

Assuming a stop sign is parallel to the image plane, the sign height ΔY in real space and Δy in image space measured in length units are related by

$$\Delta y = f\frac{\Delta Y}{Z} = h\frac{\Delta y_{\text{pix}}}{N_r} \tag{3}$$

The corresponding height Δy_{pix} measured in pixels involves the vertical resolution in pixels N_r and the image height h in length units. Knowledge of the focal length f is required, which varies among cameras and is not always documented. Alternatively, the diagonal angle of view ψ, defined as the angle between lines from opposite corners of the image to the *XYZ* origin, is approximately 50° for most cameras. From geometry, image height can be expressed as

$$h = \frac{2f\tan\left(\frac{1}{2}\psi\right)}{\sqrt{R^2+1}} \quad (4)$$

where R is the aspect ratio of image width over height. Stop signs are detected in the image and enclosed in a circle of radius r_{pix} where $\Delta y_{\text{pix}} = 2r_{\text{pix}}$. Combining this with Equations (2) and (3) results in

$$Z = \frac{N_r \Delta Y \sqrt{R^2+1}}{4r_{\text{pix}}\tan\left(\frac{1}{2}\psi\right)} \quad (5)$$

where ΔY is 30 in. for a standard stop sign. The distance Z to a detected stop sign is therefore determined by measuring the sign size in the image r_{pix} and applying the image size in N_r and R, and constants ΔY and ψ.

3D OBJECT DETECTION

The task of 3D object detection is to find a set of corresponding values of point clouds between two different clouds with a given or reference or feature point cloud object. The system compares points that store XYZ coordinates, RGB Color value etc., which takes too much time and is computationally ambiguous[3].

For optimal 3D recognition of an object, there are 3 conditions given below:

1. **Robust transformations**: Rigid transformations (the ones that do not change the distance between points) such as translations and rotations must not affect the feature. Even if we move the cloud a bit, there should be no difference. In any place the object is taken or put, it should be recognized by the system, rather than requiring that the features have the same size or angle of view.
2. **Robust noise**: Measurement errors that cause noise should not change the feature estimation much. Even if the 3D point cloud creates noise with a feature, then it should be detected.
3. **Resolution invariant**: If sampled with different density (like after performing down sampling), the result must be identical or similar.

Descriptor:

A descriptor is a complex and precise signature of a point that stores information about the surrounding environment's geometry. It identifies a point across multiple point clouds for multiple point clouds, robust noise or resolutions and viewpoints.

There are many 3D descriptors for 3D object detection. Each has its own method for computing unique values for point clouds. Some use the difference between the angles of the normal of the point and its neighbors, etc.

Point Feature Histograms (PFH) descriptors:

Surface normal and curvature estimations are basic in their representations of the geometry of a specific point. To get a better result and optimal detection, a descriptor should not capture too much detail, as they approximate the geometry of a point's k-neighborhood with few values [6].

The purpose of PFH is to encode a point's k-neighborhood geometrical properties by generalizing the mean curvature around the point in a multi-dimensional histogram of point clouds, which provides an informative signature for feature representation. It is an invariant to 3D presentation of the surface and performs well with different densities or noise levels or any resolution. It attempts to capture the best possible sampled surface variations by all interactions between the directions of estimated normal.

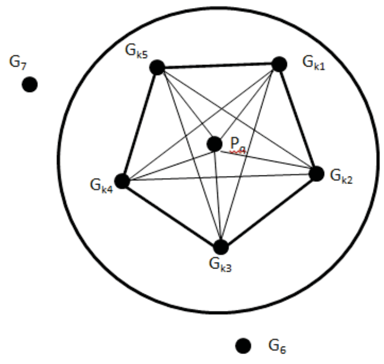


Figure 4: representation of PFH computation

Figure 4 is a diagram of PFH computation for a query point (G_q) whose radius is r and and that has k neighbors (where the distances are smaller than radius r) interconnected with a mesh. The PFH descriptor computed a histogram of relationships between all pairs of points of the neighborhood anthe d computational complexity ***$O(K^2)$***

To compute the relative difference between two points G_i and G_j and their associated normal n_i and n_j, we define a fixed coordinate frame at one of the points (see the figure below).

$$x = n_s \quad (6)$$

$$y = x \times \frac{(Gt-Gs)}{||Gt-Gs||_2} \quad (7)$$

$$z = y \times x \quad (8)$$

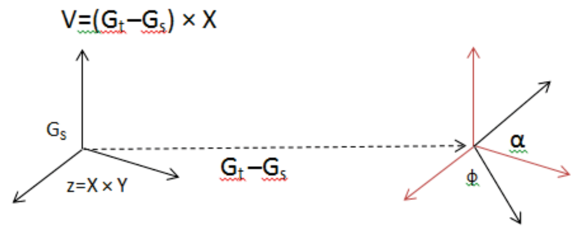


Figure 5: representation of descriptor measurement in xyz frame

Using Figure 5's **xyz** frame, the difference between the two normals n_s and n_t can be expressed as a set of angular features as follows:

$$\alpha = y.n_t \quad (9)$$

$$\emptyset = x.\frac{(G_t - G_s)}{d} \quad (10)$$

$$\theta = \arctan(z.n_{t,} \ x.n_{t,} \ ) \quad (11)$$

where **d** is the Euclidean distance between the two points G_s and G_t, d=$||G_t - G_s||_2$. The quadruplet ($\alpha, \theta, \emptyset$) is computed for each pair of two points in k-neighborhood

Set of quadruplets is binned into a histogram. The binning process divides the feature values range into subdivisions. The process counts the number of occurrences in each subinterval. The measured angles between normals can easily be calculated to the same interval on a trigonometric circle.

RESULT & EXPERIMENT

We have established an experiment regarding this topic. We have run a program using OpenCV, PCL (Point Cloud Library) and a stereo camera. The results are given in Table 1:

Table 1: Experimental Results

Case	2D Distance (meter)	3D distance (meter)	Difference (%)
1	17.23	17.5	0.015
2	24.52	25.3	0.031
3	8.41	9.77	0.162
4	34.05	36.84	0.082

As we have seen from the result, the results obtained by of 2D camera and 3D camera differ by about ~0.02%.

FUTURE RESEARCH

We are also thinking about extending this research by using different parameters and real life situations, like weather conditions, image noise reduction to recognize traffic signs, low light recognition of traffic signs etc. We hope to find more accurate and successful experiments using different parameters and real life conditions.

REFERENCES

[1] Zhao, Y., Gu, J., Liu, C., Han, S., Gao, Y., & Hu, Q. (2010). License Plate Location Based on Haar-Like Cascade Classifiers and Edges. *2010 Second WRI Global Congress on Intelligent Systems*. doi:10.1109/gcis.2010.55

[2] Lienhart, R., Kuranov, A., & Pisarevsky, V. (2003, September). Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In Joint Pattern Recognition Symposium (pp. 297-304). Springer Berlin Heidelberg.

[3] Huang, J., & You, S. (2013). Detecting Objects in Scene Point Cloud: A Combinational Approach. *2013 International Conference on 3D Vision*. doi:10.1109/3dv.2013.31

[4] Huang, J., & You, S. (2013, June). Detecting objects in scene point cloud: A combinational approach. In 2013 International Conference on 3D Vision-3DV 2013 (pp. 175-182). IEEE.

[5] Viola, P and Jones, Michael. (2001). Rapid Object Detection Using a Boosted Cascade of Simple Features, *Conference On COMPUTER VISION AND PATTERN RECOGNITION 2001*, 1-9.

[6] Kim, I., Kim, D., Cha, Y., Lee, K., & Kuc, T. (2007). An embodiment of stereo vision system for mobile robot for real-time measuring distance and object tracking. *2007 International Conference on Control, Automation and Systems*. doi:10.1109/iccas.2007.4407049

[7] Li, C. T. (Ed.). (2009). *Handbook of Research on Computational Forensics, Digital Crime, and Investigation: Methods and Solutions: Methods and Solutions*. IGI Global.