Methods to Obtain Bird's Eye View Using Inverse Perspective Mapping for Small Scale Embedded Applications.

Shishir Pokhrel
University of Siegen.
Department of Electrical Engineering and Computer Science
shishir.pokhrel@student.uni-siegen.de

ABSTRACT

Images obtained from cameras for processing in Autonomous vehicles like automobiles, mobile robots, Unmanned Arieal vehicles and etc. often need to be corrected. Conventional pin hole cameras as well as fish eye cameras take images in perspective, meaning that parallel lines converge to a point due to the perspective view. For fast moving autonomous objects, this poses a problem in curves, as well as in unsmooth terrain. A well known solution for this problem is the inverse perspective correction. Various methods are available for performing this correction, however, it needs to be simplified for small embedded devices such that the perspective correction can be done with minimal resources. This paper discusses four different ways how it can be obtained namely Geometric method, Homography transform, perspective correction and rotation correction. The former two are done in Rasperry pi, and the latter in Arduino Nicla Vision, using Python.

Keywords

Homography (inverse perspective) Transform, Image processing, Embedded devices, Autonomous driving.

1. INTRODUCTION

Image based control systems are increasing in today's autonomous systems. The general purpose is to map terrain, detect obstacles, or lanes and signs in the field of operation. Often times, the field of view can range up to large distances, introducing the problem of perspective as well as distortion correction (especailly in the case of fish eye cameras) for these images.

Bird eye view generation can be obtained from inverse perspective mapping and is particularly useful in mobile robots, or autonomous driving applications for drive assistance. For instance vehicular control in dynamic situations like curves or even in Bumpy roads is often difficult in perspective view. Curvatures on roads must be accurately identified by the mobile vehicle, for the controller to maneuver correctly, requiring filtering for smooth operation [7].

There are various methods available for generating a Birds eye view. A geometric approach uses geometry relationships to transform images from the perspective view to bird eye view [5, 2, 3], a homography based method estimates the relationship between a bird eye view with a perspective view of a scene to transform the image. The two views must be known and the relationship is an estimate using algorithms like RANSAC [1]. Moreover, image processing tools such as OpenCV and OpenMV provide an option to select points on an image, and to stretch them to cover the screen, effectively allowing careful selection of region of interest, which can be stretched over the full image frame to depict a Bird eye view.

In addition to image processing techniques, practioners and researchers have used Deep Learning techniques to obtain bird eye views. Using prior information from lane and other markings on roadways, researchers have mapped the waypoints for vehicles in roadway in Bird eye view [6]. In addition, bird eye view has also shown to be effective for detection of potholes on the road via the use of stereo vision [8].

The application of a virtual top down bird eye view is wide spread in the autonomous driving domain. However, processing images in real time requires faster computation platforms. In the case of rapid prototyping and in smaller devices, some practical methods to generate useable bird eye view are required. This paper tries to achieve this with short guide to generate birds' eye view from a camera in real time, for small embedded applications. Four approaches are described here namely, Geometric transformation, Homography transformation, Perspective correction and Rotation correction.

2. BACKGROUND INFORMATION AND RELATED WORK

2.1 Geometric transformation

Transforming the image plane to a top down view from is depicted in Figure 1. The idea of this method is to determine the geometric relationship between the image view and the desired transformed view (Bird eye view).

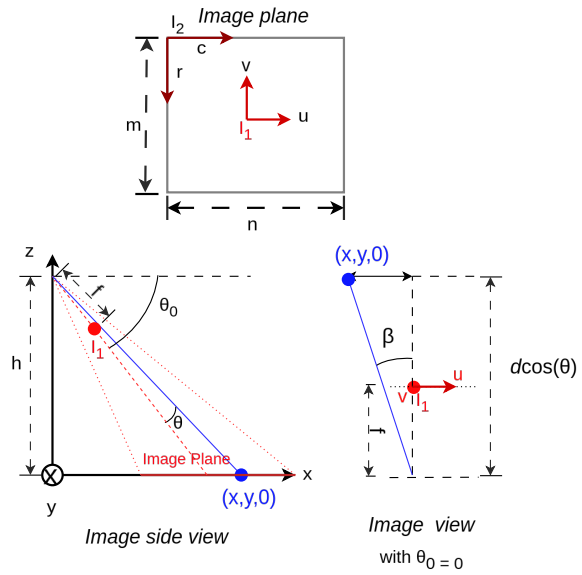

Figure 1: Depiction of geometric relation of image on image plane [3]

Utilising the modelling in Figure 1, Turker and Calligaris [3, 5] have derived the positions for x and y in the transformed image frame with the following relationships.

$$x = \frac{h}{\tan\left[\theta_0 - \arctan\left(\frac{\frac{m+1}{2} - r}{f}\right)\right]} \tag{1}$$

The position for y is then given by;

$$y = \frac{\left(\frac{n+1}{2} - c\right) d \cdot \cos\left(\arctan\left(\frac{\frac{m+1}{2} - r}{f}\right)\right)}{f} \tag{2}$$

2.2 Homography transformation

In image processing, when multiople images are taken of the same scene from the same camera in different poses, there exists a homography between them, which is the relationship between these images. Figure 2 depicts this idea. The extrinsic parameters, rotation and translation of the camera is captured in the homography.

Such projective transform from one image plane to another, as introduced by [4] a nonsingular $3x3$ matrix H represents a vector x with the given relationship holds.

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$x' = Hx \tag{3}$$

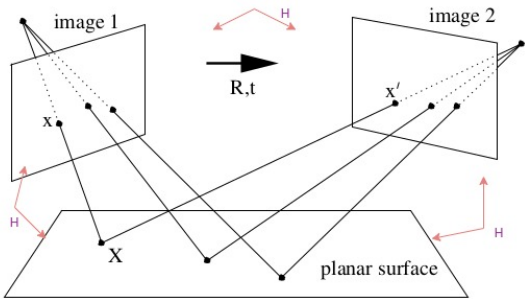

Figure 2: Depiction of Homography between image planes [4]

The image plane is related to the object plane with the following relationship, where q_{image} is the image plane, C_m is the camera intrinsic matrix, $[R, t]$ is the extrinsic matrix or the Homography matrix, and q_w represents the world coordinates.

$$q_{image} = C_m \cdot \begin{bmatrix} R & t \end{bmatrix} \cdot \begin{bmatrix} q_w \\ 1 \end{bmatrix} \tag{4}$$

Acknowledging the scaling, the equation can be expanded to:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix}$$

In small scale prototyping, the z axis can be assumed to be the perpendicular axis where image is formed implying that the rotation and translation is 0 in this axis when the camera frame is concerned. This can be assumed to be 0, and therefore a 3x3 homography matrix H like Equation 1 is obtained.

This relationship can be used to estimate a homography between two views namely, perspective and top-down view. Two images with checkerboard patterns akin to Figure 3 is utilized. The checkerboard corners is used by a sampling algorithm RANSAC, which detects points on both image and estimates the relative transformation between them, effectively providing the homography matrix, which is then used to warp perspective view to a top down view or the bird eye view.

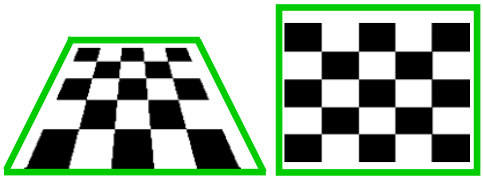

Figure 3: Chessboard pattern views for estimating the homography. Left: perspective view, right: Top-down view.

2.3 Perspective Correction

Perspective correction uses the idea of the vanishing points in a perspective image. Usually, parallel lines vanish towards the centre of the image frame, tending to converge. Using

this idea, a trapezoid can be drawn with shorter side somewhat in the top half of the screen. This trapezoid is then transformed to a square.

In implementation phase, this can be done using four corner points of the trapezoid as input or source points. These four points are stretched to the corners of the original image or to a set of four destination points, effectively stretching the image. A homography between the two view points can also be obtained (in openCV), however, it is not as robust as using a checkerboard pattern due to the intermediate points (checkerboard corners). This provides a view similar to a top down view, as depicted in Figure 3. It must be noted that only the green boxes are relevant for this approach.

2.4 Rotation Correction

A top down view can be seen as an image, taken from a virtual camera pointing from the top to the image plane. With this idea, a camera can be rotated (virtually) to ensure that the ground plane normal, and the optical axis of the camera are parallel. In other words, the camera is arranged in such a way that it looks at the ground plane in top down view.

With the use of an Inertial Measurement Unit (IMU), the camera angle can be obtained, its translation parameters calculated using trigonometry corresponding to the angle obtained from the IMU. This information is then used to rotate the image to obtain a quasi bird eye view.

3. IMPLEMENTATION AND RESULTS

The geometric as well as the homography transformation approaches are implemented on a Raspberry pi board, with the latter utilising openCV library. The results of the transformation is depicted in Figure 4.

Figure 4: Bird Eye View with Homography and Geometric methods

Table 1 compares the results of the two approaches. Both of these methods run on Raspberry Pi, while the perspective and rotation correction methods are run on the Arduino Nicla Vision.

Method	Remark
Geometric	Direct computation using a rotated camera angle and distance, as well as focal length. Camera requires calibration as focal length is a parameter. Has higher computation load (required 14 seconds on a 1.8GHz 4-core processor)
Homography-Based	Faster and less complex, fitting for Raspberry Pi Can also be used for live video capture with smooth BEV generation. However, estimated homography matrix H needs to be accurate.

Table 1: Comparison of geometric and homography based methods

Moreover openMV library is used for the Rotation and perspective correction approaches and are implemented in Arduino Nicla Vision board. The resultant transformation is shown in Figures 5 and 6.

Figure 5: Perspective Correction

Figure 6: Rotation Correction

4. SUMMARY

Table 4 summarises the four methods for obtaining inverse perspective transformation according to the image quality, real time abilities, library used and system it is implemented in.

5. CONCLUSIONS

The four different approaches discussed are useful for different applications. For roboustness and real time capability,

Table 2: Summary of the methods to obtain inverse perspective transformation

Approach	Image quality	Real time performance	Tested on	Libraries used
Geometric	Satisfactory	Works on simulated environment, although with selective features being transformed	Raspberry pi, GAZEBO	OpenCV
Homography	Good	Works in real time environment.	Raspberry Pi	OpenCV
Perspective correction	Good (in grayscale)	Works in real time environment	Arduino Nicla Vision.	OpenMV
Rotation correction	Satisfactory	Works in real time, Image perspective is corrected, Bird eye view is not generated.	Arduino Nicla Vision	OpenMV

the Homography method has produced the better results. This can be implemented for small scale applications, as well as in simulated environment as it has been found that the method is faster than the geometric method. The geometric method requires that the image is largely preprocessed, and only certain essential, dynamic and constantly varying features are transformed. These two approaches are performed using openCV. Moreover, the perspective correction and homography transform methods are similar in the sense that they both use a similar envelope of image. When estimating homography matrix, the outline of checkerboard pattern in perspective view makes a trapezoid, however, due to the identifiable points on the checkboard pattern (using RANSAC), the estimation of homography is better.

The perspective correction method is useful when the homography between two images is not known or cannot be known, and when the system does not require accurate or feature detection. The image quality after the transformation is not comparable to the transformation from the homography estimation, however, it has been possible to use images transformed with this method (using openMV) on a small scale lane following robot with satisfactory performance in lane detection. Rotation correction, also implemented using openMV has proven to be useful in transforming the perspective distortion, namely the converging of parallel lines to keeping them parallel in the transformation (see figure 6. This however, did not provide a complete birds eye view like the other approaches.

6. REFERENCES

[1] OpenCV documentation, 2024.

[2] A. Abbas and A. Zisserman. A geometric approach to obtain a bird's eye view from an image. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2024.

[3] T. Alper. Illumination-robust lane marker detection for stereo inverse perspective mapping, 2016.

[4] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, USA, 2 edition, 2003.

[5] C. Marvin. Analyse und implementierung mathematischer methoden für die beschreibung des fahrbahnmarkierungsverlaufs zur realisierung automatischer spurhaltesysteme, 2022.

[6] M. A. Rafique, M. I. Hussain, S. Dubey, K. Sayfullokh, and M. Jeon. A monocular camera bird-eye-view generation using lane markers prior. In *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, pages 1137–1142, 2022.

[7] J. Zhang, W. Sui, Q. Zhang, T. Chen, and C. Yang. Towards accurate ground plane normal estimation from ego-motion. *Sensors*, 22(23):9375, 2022.

[8] T. Zhao, L. Yang, Y. Xie, M. Ding, M. Tomizuka, and Y. Wei. Roadbev: Road surface reconstruction in bird's eye view, 2024.