Distance Determination for an Automobile Environment using Inverse Perspective Mapping in OpenCV

S.Tuohy, D.O'Cualain, E. Jones, M.Glavin

Electrical & Electronic Engineering

College of Engineering and Informatics

National University of Ireland, Galway

email{**s.tuohy2 d.ocualain4 edward.jones martin.glavin**}*@nuigalway.ie*

***Abstract*—This paper presents a novel real-time distance determination algorithm using an image sensor for use in an automobile environment. The system uses a forward facing camera placed within the vehicle.**

From a single forward facing image, it is difficult to determine distances to objects in front of the vehicle with any degree of certainty. There is a non linear relationship between the height of an object in a forward facing image and its distance from the camera. This paper presents a method which uses Inverse Perspective Mapping (IPM) to overcome this problem. Using IPM, we can transform the forward facing image to a top-down "bird's eye" view, in which there is a linear relationship between distances in the image and in the real world.

The algorithm is implemented in the C language using the OpenCV libraries. Implementation in OpenCV leads to a high performance, low overhead system that could be implemented on a low power embedded device in an automotive environment.

***Keywords* – Inverse Perspective Mapping, Distance Detection, OpenCV**

I INTRODUCTION

In 2008, rear end collisions accounted for almost 25% of all injuries sustained in road traffic accidents on Irish roads [1]. Collision detection systems can provide drivers with a warning prior to a potential collision allowing them to take preventative action [2]. They can also be incorporated into safety systems, as seen in Mercedes Pre Safe [3] which pre-charges brakes and tightens slack on seatbelts if an imminent collision is detected.

Currently available systems on the market from manufacturers such as Audi, Mercedes Benz, and Nissan use RADAR or LIDAR sensors to implement collision detection. However, since metallic parts of vehicles reflect RADAR much more effectively than human tissue, detection of pedestrians can be difficult with RADAR, particularly at long distance. Furthermore, RADAR and LIDAR are active systems, potentially leading to higher power requirements and interference issues, in comparison with optical camera sensors [4].

A large amount of research has been done in the area of collision detection using forward facing cameras [5]. A multi-camera setup as employed by Toyota in some Lexus models, provides depth information by establishing feature correspondence and performing triangulation [6]. However, it also carries severe processing and configuration overheads, which are undesirable in the cost and power sensitive automotive market.

The proposed system consists of a single forward facing video camera mounted on a vehicle, capturing video images at 30 frames per second (fps). This setup distinguishes the system from similar systems which use either multiple cameras or active devices such as RADAR or LADAR.

Figure 1 below shows an overview of the proposed system with a forward facing camera ideally mounted inside the vehicle, just behind the rear-view mirror.

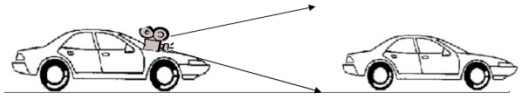

Figure 1. Forward facing camera detects objects ahead of the vehicle.

Section II of this paper discusses distance determination technologies currently found in many vehicles, while section III describes Inverse Perspective Mapping which allows the image perspective to be linearised. Section IV discusses the requirement for object detection and section V

details the calibration issues. The technological platform is described in section VI and results are presented in section VII. Some discussion and future work concludes the paper in section VIII.

II Distance Determination

Distance determination in an automobile could allow an in-car system to provide feedback and alerts to the driver, by either prompting the driver to take preventative action, or prepare the vehicle's safety systems for an imminent collision.

Distance determination using active systems such as RADAR or LADAR is a relatively simple concept: signals transmitted from an antenna reflect back to the transmitter from the target object. The distance can be calculated based on the length of time taken for the signal to travel to and from the target.

The use of a single, forward-facing optical camera does not directly provide depth information in a scene. A means of compensating for the non linearity between an objects position in the image and its position on the road surface in the real world is necessary. Figures 2 and 3 below demonstrate the difficulties that working with a front mounted optical camera can present.

Figure 2. Distant object as shown by forward facing camera.

Figure 3. Near object as shown by forward facing camera.

As can be seen from the white arrows in Fig. 2 and Fig. 3, the relationship between the height of an object in the image and its distance from the camera is somewhat nonlinear in nature. The object is much closer to the camera in Fig. 3 than it is in Fig. 2, however, the change in the length of the white arrow does not proportionally reflect this difference in distance. A method by which we can compensate for this perspective effect is required.

III Inverse Perspective mapping

Inverse Perspective Mapping (IPM) [7] is a mathematical technique whereby a coordinate system may be transformed from one perspective to another. In this case we use IPM to remove the effects of perspective distortion of the road surface in the forward facing image (an example of which can be seen in Fig. 2), to an undistorted top-down view.

Transforming the image in this manner removes the non linearity of distances, as can be seen in the sample Inverse Perspective Mapped view in Fig. 4

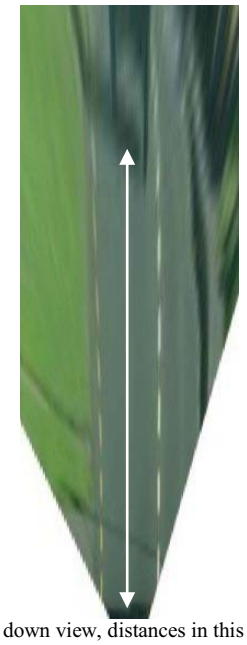

Figure 4. Top down view, distances in this image may be measured directly.

In order to create the top-down view, we first need to find the mapping of a point on the road surface (X_w, Y_w, Z_w) to its projection on the image plane (u, v). From Fig. 5 we can see that this requires a rotation about θ, a translation along the camera's optical axis, and a scaling by the camera parameter matrix.

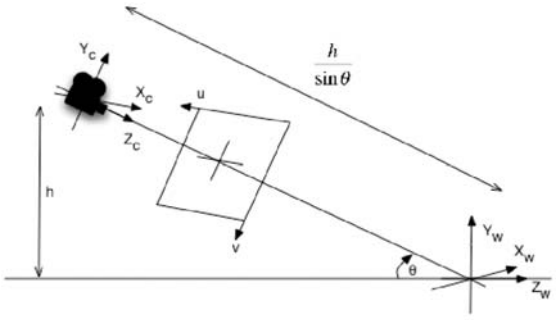

Figure 5.Image coordinate system in relation to world coordinate system.

This mapping can be expressed as:

$$(u, v, 1)^T = \boldsymbol{KTR}(x, y, z, 1)^T \quad (1)$$

where $\boldsymbol{R}$ is the rotation matrix:

$$\boldsymbol{R} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2)$$

$\boldsymbol{T}$is the translation matrix:

$$\boldsymbol{T} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\dfrac{h}{\sin\theta} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3)$$

and $\boldsymbol{K}$ is the camera parameter matrix:

$$\boldsymbol{K} = \begin{pmatrix} f \times ku & s & u_o & 0 \\ 0 & f \times kv & v_o & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (4)$$

Here, f is the focal length of the camera, s is the skew parameter [8] of the pixels, and $(ku \times kv)$ is the aspect ratio of the pixels. Eq. 1 can also be expressed as:

$$\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \quad (5)$$

As we are only interested with data on the road plane ($Y_w = 0$), this simplifies to:

$$\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{13} & p_{14} \\ p_{21} & p_{23} & p_{24} \\ p_{31} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} X_w \\ Z_w \\ 1 \end{pmatrix} \quad (6)$$

Using this equation, it is possible to map each pixel on the image plane to a top down view.

IV Object Detection

In order to determine the distance between the camera and an object in front of the camera, we must determine where the object lies on the road surface in the Inverse Perspective Mapped view (as shown in Fig. 4).

Most monocular object detection algorithms require a priori knowledge of the potential objects they will be expected to detect [9]. However, due to the high variability in potential objects on the road, an algorithm of this kind would require a very large database of detectable object characteristics. A more efficient method of doing this is using a road-surface subtraction algorithm. Considering the scene in Fig. 4, the object we must detect is deemed to be the nearest part of the image that is not road surface directly in front of the vehicle. Therefore the easiest way to accurately detect the object is by removing the road surface from the image, leaving only the required object. A flowchart of the road removal algorithm is shown in Fig. 6. The algorithm removes the road surface from the image as follows:

1. The image is separated into its constituent Red Green and Blue channels.
2. Taking one of the RGB channels, the algorithm calculates the average pixel intensity from the area immediately in front of the vehicle.
3. The algorithm loops though every pixel in the image, creating a new binary image where 0 signifies a road pixel, and a 1 is a non-road pixel. A road pixel is identified if it is within ±35 intensity of the average intensity calculated from step 2.
4. The algorithm repeats steps 2 and 3 for the remainder Green and Blue Channels.
5. Finally, the algorithm merges the 3 new binary images into a new RGB image.

Sampling the road pixels allows the algorithm to dynamically adapt to different lighting and road surface conditions. Using a sample scene prior to road removal shown in Fig. 7, the resultant image after road removal is shown in Fig. 8.

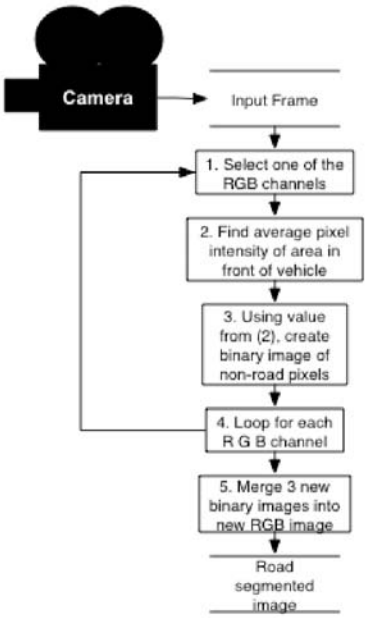

Figure 6. Flowchart of road removal algorithm

Figure 7. Road scene prior to removal of road surface

Figure 8. Scene from figure7 after application of the road removal algorithm

The scene in Fig. 8 is now ready for IPM transformation and subsequently, detection of the vehicle in front. The transformed image resulting from Eq. 6 is shown in Fig. 9.

Before we can calculate the distance of the object from the host vehicle, we need to calculate the height of the object in pixels from the bottom of the image as shown in Fig. 9. Beginning at the bottom of the binary image in each RGB colour space, we calculate the average pixel intensity for each row of pixels in the image. From experimentation, we found that value larger than 100 on the RGB scale indicated the presence of an object of interest. From this we can ascertain how high in the image the nearest object lies. Only the area directly in front of the host vehicle was analysed. This was to minimise the rate of false positive object detections caused by vehicles in neighbouring lanes.

V CALIBRATION

We now have a location (measured in pixels in the transformed image) for the position of the nearest object in the image. For a practical implementation, this value needs to be converted into meters. There are several methods by which the algorithm can be calibrated.

a) Manual Calibration

Typically, the proposed system would be implemented with a fixed camera in a fixed position on the chassis of a car. Placement of a 1m white stripe in the line of sight of the camera, along a flat surface, would allow for the measurement of the number of pixels equivalent to 1 meter. Using this value, the number of pixels in the image could be converted to a real world value in meters.

b) Automatic Calibration

Automatic calibration of a bird's eye view projection system can be achieved using a chessboard pattern of known dimensions placed in the line of sight of the camera as explored by Bradski & Kaehler [8].

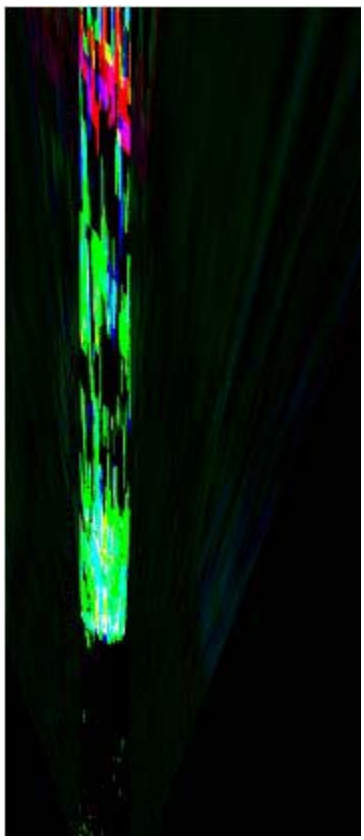

Figure 9. Section of transformed image, potential object can be seen in green

c) Road marking online calibration

The methods described here illustrate calibration for a fixed camera position with a consistent transformation matrix. Distances were calibrated based on road markings in the image, which were measured to be 1.85m in length. Calibrating in this fashion greatly simplified testing of the algorithm, as the algorithm was not restricted to a single configuration, and a wide variety of sample images could be used.

VI IMPLEMENTATION TECHNOLOGIES

The algorithm presented here was first investigated and then implemented to function in real time.

For this purpose, the C language, complemented by the OpenCV library of functions was chosen for implementation. OpenCV is an open source library of functions for use in image processing [10]. Originally developed by Intel, the project is now open source and maintained by the community, supported by the Willow Garage company. The purpose of the library is to provide optimised implementations of common image processing functions.

OpenCV allows for the rapid development of image processing algorithms without the need to develop algorithms from first principles every time. Data structures are provided to simplify handling of images and video. Numerous functions are provided to help carry out common tasks, such as thresholding, transforming, matrix multiplication among other commonly used image processing functions.

VI Testing and Results

This section details the results obtained by the algorithm. Successful distance determination was obtained from a database of video samples in real time operation. Samples of these successful tests can be seen below. In Fig. 10, the algorithm has detected the vehicle in front. A white line overlaid at the point of detection indicates this.

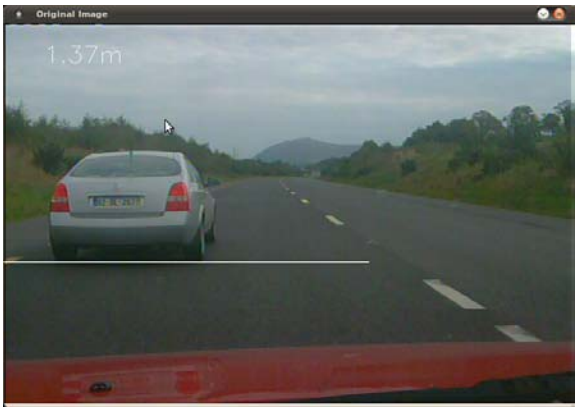

Figure 10. Successful object detection and distance calculation.

In Fig. 10 above the calculated distance is displayed in the top left hand corner of the image where there is typically little or no activity. This allows the information to be conveyed to the driver unobtrusively. The algorithm was tested with approximately 3 minutes of video with varying levels of traffic and was found to function well. The ratio of frames with successful detection to the total number of frames sampled was regularly measured at 100% with a tolerance of approximately 5%. Another example of successful object detection is displayed below in Fig 11. Here, only the vehicle directly in front of our vehicle triggers detection and the algorithm is not confused by multi-lane traffic.

VII Performance

This section explores system performance and practicality for implementation in a real time system. A database of video samples was captured at 30 frames per second (fps), at a resolution of 640 x 480 pixels. The algorithm was implemented on a standard Intel processor without any sandboxing or process optimisation applied. Testing was carried out on a 2.5GHz Intel Core 2 Duo processor with 4GB of RAM, running Linux kernel version 2.6.32.

Figure 11. Successful object detection

The code footprint was 93KB and the time taken to process each frame was found to be 0.053 seconds which would not allow this algorithm to be run on every frame of video at 30fps. Considering that the change in distance of potential objects is relatively slow compared to the full frame rate of the system, a slower frame rate was considered for the purposes of calculating distance.

Using a sampling rate of every 10 frames produced smooth and reliable results. 10 frames was chosen as it provides a good trade-off between computation time and number of calculations per second. Fig. 12 below demonstrates the computation time taken for a 386 frame video at a number of different sampling rates.

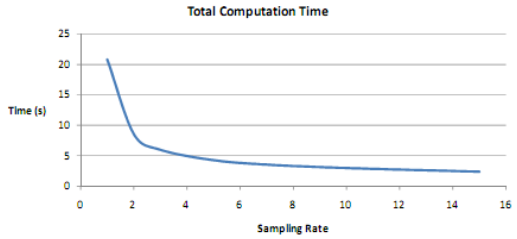

Fig 12. Computation time for 386 frame video for a number of sampling rates

Further performance improvements are possible by using highly optimised routines provided by microprocessor manufacturers. Intel provides a set of routines in their IPP (Intel Performance Primitives) [11] package which OpenCV can natively use instead of its own process implementations. Using a dedicated processor, with these optimisations can provide very significant performance increases over a multipurpose, multi tasking processor such as the one on which testing was carried out. This means that real time implementation of the algorithm on even lower powered devices is well within reach.

Situations where the car tilts due to acceleration, deceleration or undulations in the road surface could lead to short term distance estimation errors. Compensation for these effects could be implemented in two ways: (1) by detecting regular shapes in the image (e.g. road markings) and using

their known shape and size to calibrate the images or (b) use sensors, such as tilt sensors, to indicate the instantaneous angle of the camera and use this information to recalculate the IPM parameters.

The system would also be vulnerable to environmental factors such as fog, heavy rain and lighting conditions that reduce the clarity of the image available to the camera. While RADAR may be better capable of "seeing" through visual obstructions, it presents an additional cost to the car manufacturer and can only be used for a small number of applications. Cameras can be used for a wide range of applications, and are already being placed in high-end vehicles at several locations around the vehicle.

VIII Conclusion

Distance determination coupled with robust object detection routines could provide innovative and life saving systems to aid in driver safety and awareness.

In the past, commercially available microprocessors have been unable to cope with the strain of real time image processing algorithms, however this is changing. With low level C code, and the help of optimised routines, real time image processing systems are now a feasible proposition for the automotive market. The versatility of a forward facing camera, and its low cost make it an ideal addition to automobile safety systems. The success of this algorithm opens up exciting new possibilities in the field of road safety. Its potential uses are numerous. It could be used as a pre-collision safety system to warn drivers of safe following distances. It could also be used to detect pedestrians and other objects that RADAR and LADAR cannot.

Future work in this project will involve making the system more robust to lane markings and other road surface artefacts such as manhole covers, shadows and other artefacts. Other possibilities include using the angle of the steering wheel to predict the path to be taken by the vehicle to allow the system to handle lane-change situations.

References

[1] Irish Road Safety Authority. "RSA Road Collision Factbook 2008",Road Safety Authority, Primrose Hill, Dublin Road, Ballina, Co. Mayo, 2008 (http://www.rsa.ie/Documents/Road%20Safety/Crash%20Stats/RCF20083-2-2010.pdf) (Accessed April 2010)

[2] D. O Cualain, M. Glavin, E. Jones, P. Denny:"Distance Detection Systems for the Automotive Environment: A Review", In *Irish Signals and Systems Conf. 2007*, Derry, N. Ireland, 2007

[3] S. Arduc, R. Justen, T. Unselt, "Motor Vehicle with a Pre-Safe System", U.S. Patent 2005/0080530, Assignee: DaimlerChrysler AG, April 14, 2005 (http://www2.mercedes-benz.co.uk/content/unitedkingdom/mpc/mpc_unitedkingdom_website/en/home_mpc/passengercars/home/new_cars/models/cls-class/c219/overview/safety.html) (Accessed April 2010)

[4] D. M. Gavrila, "Sensor-based pedestrian protection", in *IEEE Intelligent Systems*, 16(6):77–81, 2001.

[5] T.A. Williamson,"A high-performance stereo vision system for obstacle detection", PhDthesis, Carnegie Mellon University, Pittsburgh, Pennsylvania,1998.

[6] V. Vijayenthiran, "Opel Insignia to feature front camera safety system", *Motor Authority,* 18th June, 2008 (http://www.motorauthority.com/blog/1023797_opel-insignia-to-feature-front-camera-safety-system) (accessed 2010)

[7] H.A. Mallot, H.H. Biilthoff, J.J: Little, S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle detection", in *Biological Cybernetics,* pp. 177-185, 1991

[8] R. Hartley and A. Zisserman, "MultipleViewGeometryinComputerVision",Cambridge University Press, pp. 175, 2000

[9] Z. Sun, "On-road vehicle detection", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 694–711, 2006.

[10] G. Bradski, A. Kaehler, "Learning OpenCV: Computer Vision with the OpenCV libraries". O. Reilly Media. pp. 409, 2008

[11] S. Taylor, "Intel Integrated Performance Primitives: Howto Optimize Software Applications Using Intel IPP", Intel Press. 2008, (http://software.intel.com/en-us/intel-ipp/) (accessed April 2010)