Oakland University

Intelligent Ground Robotics Competition

Students: Leonard Magro
David Muscat
Niels Schouten
Johann Briffa

Date: July 2000

Contents

List of Figures

Introduction

The unmanned vehicle system is divided in three modules. These are the Vision Module (VM), the Path Planning Module (PPM) and the Navigation and Control Module (NCM). Splitting the system in three completely separate modules and defining a fixed communication protocol between them allows for completely independent implementation. This means that if some algorithm is changed in one module, the other modules are not affected as long as the communication protocol is not changed.

In order for the vehicle to move through a path that is specified only by lane markings, it must have a means of sensing the environment. This is the task of the VM. It must extract the positions and directions of lane markings and be able to produce a list of coordinates, relative to the vehicle, which completely define the lane markings. This requires the camera to be previously calibrated.

Once the VM specifies a series of points in vehicle coordinates, the PPM has to use this data and previous data in order to define a safe path that the vehicle can follow. The PPM also acts like a master, with the VM and NCM acting as slaves. Thus the PPM can synchronize the three modules together.

The third module is Navigation and Control. The navigation part of the system keeps track of the position of the vehicle, and the control part controls the position and speed of the vehicle. Wheel encoders are used in order to keep track of position. The control system will control the speed and position of the vehicle such that the path will be followed smoothly, accurately, and fast.

1.0 Vision Module

The main tasks in the VM were calibration and detection of lane markings and obstacles. These will be discussed separately.

1.1 Camera Calibration

Camera Calibration is the procedure to obtain the camera parameters. These parameters define the relation between the image coordinates of a point in a scene and the actual world coordinates. (see [1]). Thus this requires the definition of a frame of reference, relative to which all measurements are taken. The main camera parameters are the x, y, z coordinates of the camera coordinate frame and the angles between the reference frame axes and the camera axes. This results in the following homogenous transformation:

$$\begin{bmatrix} c_{h1} \\ c_{h2} \\ c_{h3} \\ c_{h4} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where a_{ij} contain the camera parameters

X, Y & Z are the world coordinates

C_{h1}/C_{h4} is the x image coordinate

C_{h21}/C_{h4} is the y image coordinate

C_{h31}/C_{h4} is the z image coordinate, but this is of no interest. It is just required in order to simplify the derivation.

Since the z image coordinate is not required, $a_{31} \ldots a_{34}$ are not required. Thus camera calibration reduces to the determination of 12 unknowns. To solve for these unknowns, at least 12 equations are required. These equations can be obtained by knowing the world coordinates of 6 points and their corresponding image coordinates. In practice, in order to get an accurate calibration, more than 6 points are required. This leads to an over-determined system of equations, which can be solved using the least-squares method.

The method to obtain points in world coordinates and their corresponding image coordinates is as follows. First an image of a grid of lines drawn on a flat surface is grabbed using the camera. By knowing the orientation and position of the flat surface and the spacing between the lines, the world coordinates of the

intersections of the grid lines can be found. The image coordinates of the intersections is then found by fitting straight lines to the grid line pixels in the image using the Hough Transform. Then the actual intersections can be easily calculated. Figure 1 shows the flat surface with the grid lines drawn on it. Figure 2 shows lines (blue) fitted to the grid line pixels. The red crosses in figure 2 show the calculated grid intersections.

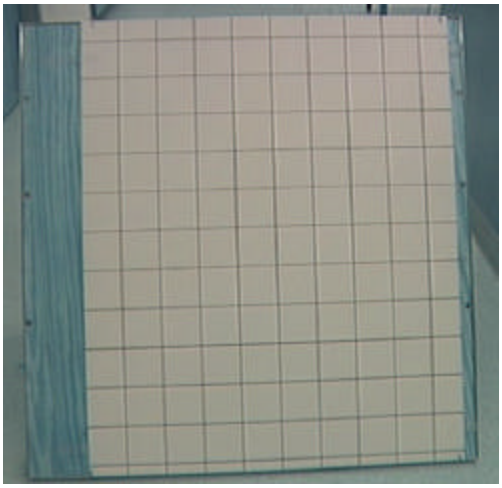

Figure 1 - Shows the flat surface with the grid lines drawn on it.

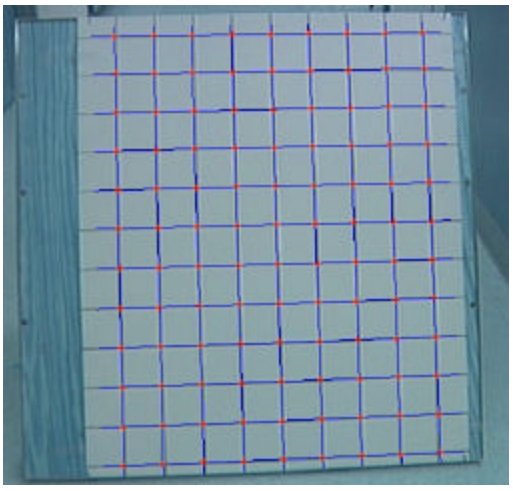

Figure 2 - Lines (blue) fitted to the grid line pixels. The red crosses show the calculated grid intersections.

In order to obtain a large amount of points, the grid shown in figure 1 was placed in front of the camera in 3 different orientations. Each orientation produced 108 points. Matlab was used to solve the resulting over-

determined system of equations. The software is given at the end of Appendix B. The resulting matrix were then stored in a text file which is then read by the lane detection software.

1.2 Detection of Lane Markings

The flowchart in figure 3 shows the process used to detect lane markings. Figures 4 and 5 show intermediate results of the process defined by figure 3. Figure 6 shows the final result. It can be seen that the centerlines of the lane markings were extracted. This process runs at about 1Hz. But this is not a fixed value since the computational time varies according to the content of the current frame.

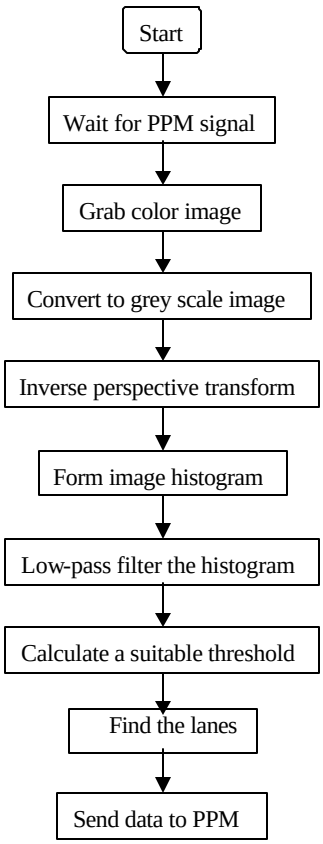

Figure 3 - Flowchart for Lane detection process

Figure4 - Grey scale version of image as seen by the camera

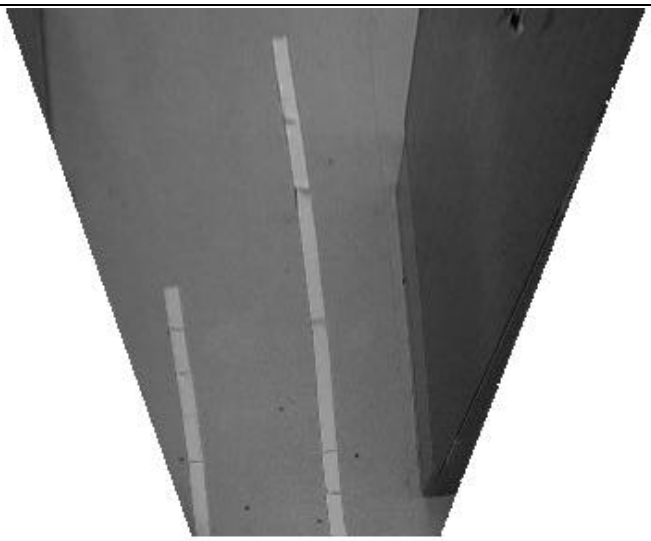

Figure 5 - Inverse perspective of the image in figure 4

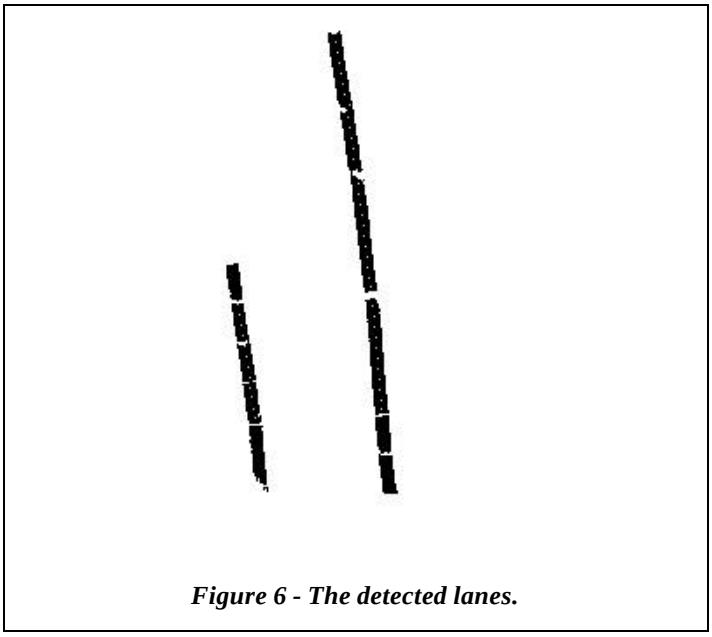

Figure 6 - The detected lanes.

1.3 Obstacle detection

Orange obstacles are detected based on the color. The hue and saturation of the image are first calculated. Then a mask is created based on windows in the hue and saturation scales. Thus the pixels representing the

orange obstacles can be set to zero before detection of lanes is conducted. Also the position of obstacles is calculated using inverse perspective.

2.0 Path Planning Module

Once the VM detects the position of the obstacles and the lane markings, it sends this data to the PPM via ethernet. Basically the PPM has to make the following decisions:

1. Which areas ahead of the vehicle are safe.
2. Which is the best path to follow. Best means a path which is clear of obstacles and has the least turns.

The decision to find a good path seems trivial, but it is not the case. The path planning process can be broken down to the following steps:

1. Conversion of the point coordinates (sent by VM) from vehicle frame to world coordinate frame. This can be done because the PPM can query the NCM for current position and orientation and at the same time instruct the VM to grab a frame. Thus the PPM knows the origin and orientation of the coordinates of the points that are sent by the VM.
2. Decide which areas in front of the vehicle are safe and construct a path.
3. The path data is then sent to the NCM through a serial link.

3.0 Navigation and Control Module

3.1 Approach

The software for the navigation and control system has been designed in three steps, i.e. PC & Matlab, PC & CVI, Microcontroller & C language. Matlab can be used for fast verification of ideas and algorithms, because implementation is fast and a lot of convenient analysis tools are available. PC & CVI can be used to check algorithms while using data from the actual sensors. Compared to the microcontroller, DAQ

boards & CVI software simplify visualization and analysis of signals a lot, which is very convenient for debugging the code. The C-code for algorithms in CVI can be used for microcontroller as well. This C-code has already been checked, which decreases time spend on debugging the microcontroller.

3.2 Navigation System

For our navigation system we define two coordinate systems. The first coordinate system is a non-moving reference frame that is fixed to the obstacle course, and the second coordinate system is the vehicle frame connected to the vehicle. The second frame defines the position and orientation of the vehicle. At the start the reference frame and the vehicle frame are initialized to be the same (see figure 7), when the vehicle starts moving, the position and orientation of the vehicle frame with respect to the reference frame will continuously be updated.

The navigation uses the sensor signals from optical encoders. The big advantage of the encoders is that the output is digital, so that the noise is very low. The quantization noise of the encoders is also very low

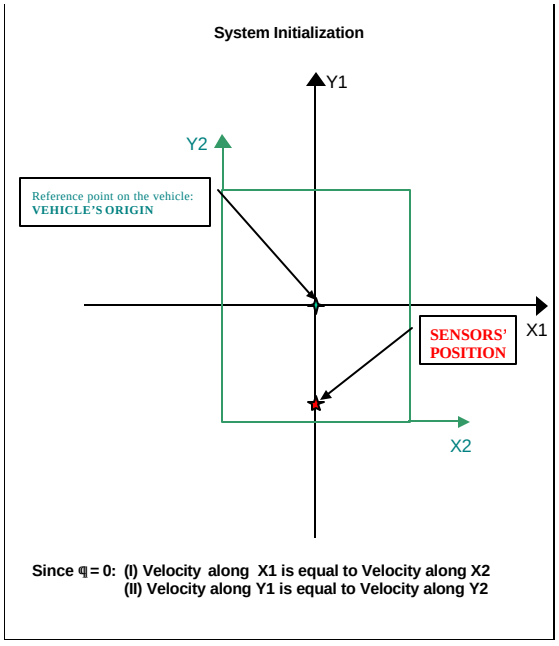

Figure 7 - Initialization of Vehicle and Reference Frames

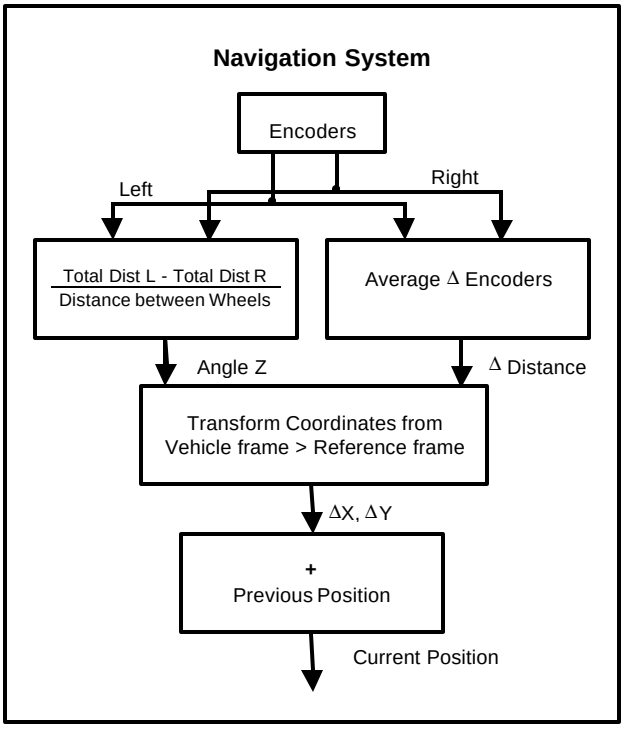

Figure 8 - Block Diagram of Navigation System

because of the high resolution of the encoders (1024 pulses per revolution) and the gear box (6: 1), resulting in 6*1024 = 6144 encoder pulses per revolution of the wheel.

The encoders are feeding Quadrature Decoder/Counter ICs. These quadrature decoders decode the incoming signals into count information that will then be transformed into actual displacement and speed. Figure 8 presents how the position, orientation, and speed of the vehicle frame are being updated using optical encoders mounted on the motor shafts:

1. The encoder signals are transformed to number of wheel revolutions by dividing them by the resolution of the encoders and the gear ratio of the gear box.
2. Convert number of revolutions from the wheels to distance traveled by left and right wheel.

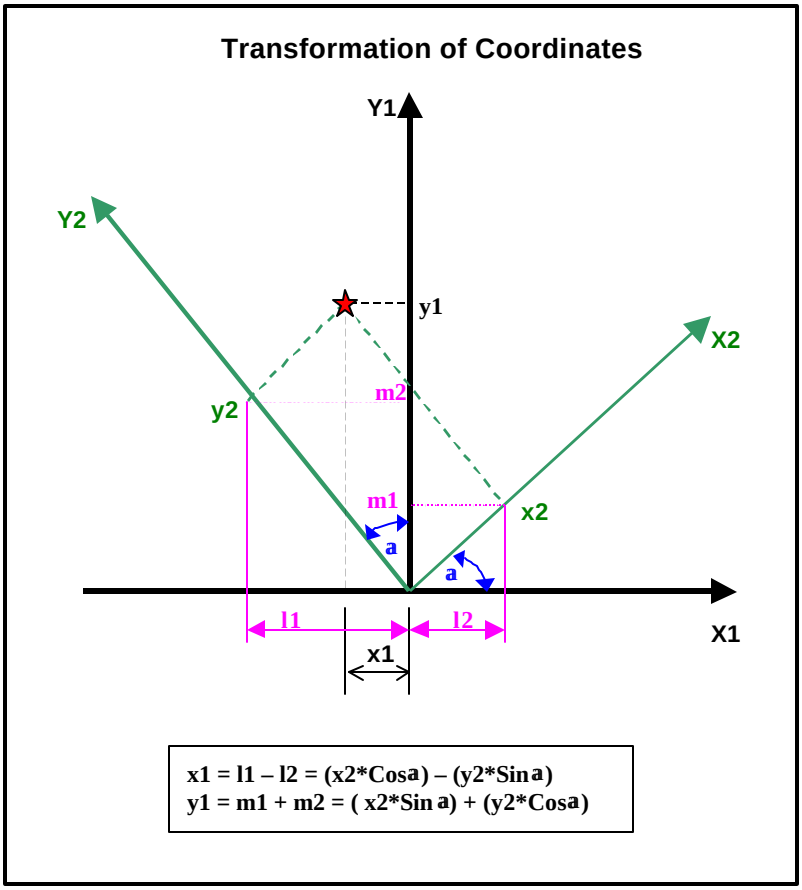

Figure 9 - Coordinate Transformation

3. These signals are fed into the second two blocks in figure 8, which compute the new angle (Angle Z) of the vehicle frame with respect to the reference frame and the distance traveled between the current and the last sample (ΔDistance).
4. The fourth block uses Angle Z to transform ΔDistance from vehicle frame to reference frame. The outputs of the block are the difference in the X and Y coordinate (with respect to the reference frame) of the origin of the vehicle frame. The transformation is explained in Figure 9 (see also [3]).
5. Finally ΔX and ΔY are added to the current position of the origin of the vehicle frame to obtain the new position.

3.3 Control System

The current vehicle position and an array of points sent by the Path Planning Module are used by the control system to control the position of the vehicle. Figure 10 presents a block diagram of the control system.

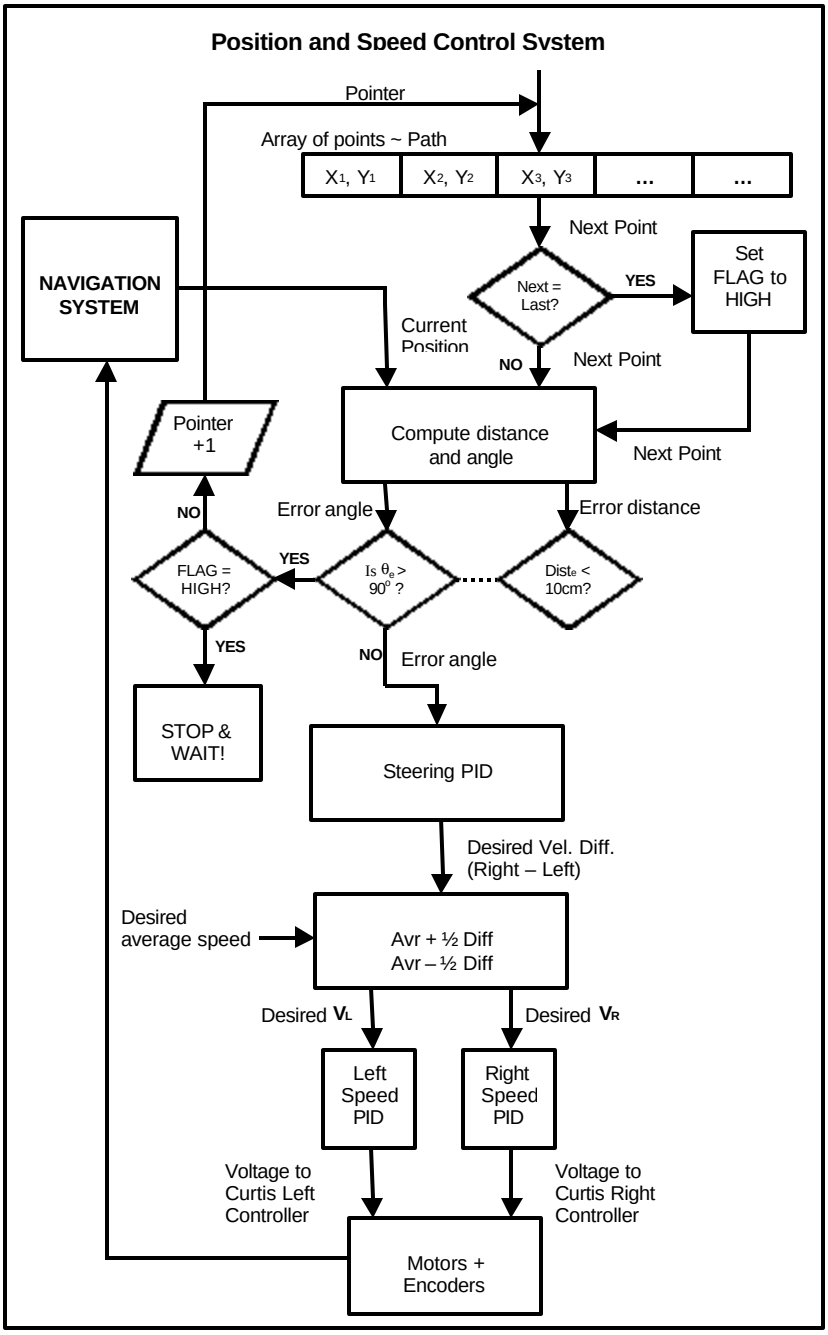

Figure 10 - Block Diagram of Control System

The Path Planning Module (PPM) sends the NCM an array of coordinates that define the path at least once a second. When PPM sends a new array, NCM will replace the previous array with the new array. First the distance and angle between current position and orientation of the vehicle to the first point in the array are computed. If the distance is smaller than a certain tolerance (10 cm in the current set-up) or when the absolute value of the angle is greater than 90 degrees (in this case the point is behind the vehicle), the next point in the array will be chosen by incrementing the pointer of the array. In case the last point in the array has been reached, the vehicle will be stopped and the NCM will wait for new points from the PPM.

In case the last point has not been reached yet, the error angle is send to the steering PID that outputs a desired difference in speed between right and left wheel. The block after the Steering PID converts the desired speed difference and desired average speed into the desired speed for the left and right wheel. The setpoints for left and right speed are subtracted from the actual vehicle speed and fed into two PID that control the speed of the left and right motor separately. The outputs of the PIDs are fed into the Curtis PWM drivers for the motors.

Because the steering PID is connected in series to the speed control PIDs, the steady state error in heading angle is zero. It is possible to tune all 3 controllers with the Ziegler-Nichols tuning rules (second method) followed by some fine-tuning. Tests showed that the vehicle passes through all points sent by the Path Planner at the desired speed.

References

[1] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing*, Addison-Wesley, 1993.

[2] Katsuhiko Ogata, *Modern Control Systems*, Third Edition, Prentice Hall, 1997.

[3] Mark W. Spong and M. Vidyasagar, *Robot Dynamics and Control*, John Wiley & Sons, 1989.

Bibliography

Ramesh Jain, Rangachar Kasturi, Brian G. Schunck, *Machine Vision*, McGraw-Hill, 1995.

W. Bolton, *Mechotronics*, Second Edition, Addison Wesley Longman, New York, 1999.

Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language*, Second Edition, Prentice Hall, 1988.

Jacob E. Goodman and Joseph O'Rourke, *Handbook of Discrete and Computational Geometry*, CRC Press LLC, 1997.

Anil K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.

Massimo Berozzi and Albeto Broggi, GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection, *IEEE Transactions on Image Processing*, vol. 7, no. 1, January 1998.