Sonar based Outdoor Vehicle Navigation and Collision Avoidance

Dirk Langer and Charles Thorpe
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15232

Abstract - **Detecting unexpected obstacles and avoiding collisions is an important task for any autonomous mobile system. This paper describes an approach using a sonar system that we implemented for the autonomous land vehicle Navlab. The general hardware configuration of the system is shown, followed by a description of how the system builds a local grid map of its environment. The information collected in the map can then be used for a variety of applications in vehicle navigation like collision avoidance, feature tracking and parking. A simple algorithm was implemented that can track a static feature such as a rail, wall or an array of parked cars and use this information to drive the vehicle. Methods for filtering the raw data and generating the steering commands are discussed and the implementation for collision avoidance and its integration with other vehicle systems is described.**

I. INTRODUCTION

The autonomous land vehicle Navlab has already successfully been driven on roads and cross country. Different sensors are used to perceive the structure of the environment and navigate the vehicle under a variety of conditions as described for example in [8]. The sensors mainly employed so far were colour video cameras and the ERIM 3-D laser range finder. Detecting obstacles and taking an appropriate decision is an important task for any mobile system in order to navigate safely through its environment. Obstacle detection is possible using colour video images or ERIM range images. However, owing to the data aquisition process and the required intensive image processing, these types of perception systems are generally not very well suited for a quick reaction to unexpected obstacles. Especially in the case of collision avoidance, a sensor is needed that can supply the relevant information fast with little data processing overhead and interact with actuators at the level of the vehicle controller (See also [1]). Sensors that satisfy these requirements are for example sonars, infrared sensors, pulsed 1-D laser range finders or microwave radar.

Compared to light based sensors, sonars have the advantage that they do not get confused by transparent or black surfaces. On the other hand, the wavelength of ultrasound is much larger than the wavelength of light, i.e. usually around 4 mm as compared to 550 nm for visible light. Therefore, unless the transducer faces the reflector surface in a normal direction, only rough surfaces or edges can reflect sound waves. However, most real world outdoor surfaces almost always have a type of surface roughness that enables a sonar to detect the object. It was therefore decided to use sonar sensors for the collision avoidance system of the autonomous land vehicle Navlab. The following sections describe the sonar system and its performance in an outdoor environment. Some novel results were obtained in using the system for vehicle navigation by itself and by integrating it into other vehicle navigation systems. The system is configured in such a way that more sensors can be added easily in the future. These sensors do not necessarily have to be sonars but can be any other type of point range sensor. In the future it is intended to integrate at least one other type of range sensor into the system, most probably laser or radar based.

II. HARDWARE CONFIGURATION

Sonar sensors have already been used successfully for indoor mobile robots as described in [2] and [4]. An outdoor environment, however, adds some additional constraints on the type of sensor that can be used. Specifically the sensor should be robust against moisture, dust particles and noise from the vehicle engine and other sound sources.

Therefore an open type electrostatic transducer such as the Polaroid cannot be used. Instead a closed type piezoceramic transducer operating at a frequency of 80 kHz was selected. A detailed description of this sensor is given in [5]. The high operating frequency makes this sensor fairly robust against acoustic noise, while still providing an operating range up to 6 m. The beam angle of the sensor is approximately 5°, i.e. at the 3 dB intensity fall off from the major axis. Based on these characteristics, a total of five sensors was chosen in order to provide a good area coverage in front of the vehicle with a reasonable spatial resolution. The sensors are mounted on a guide rail such that their position and orientation can be freely adjusted. A typical sensor arrangement is shown in Fig. 1. Certain sensor configurations or environments can lead to acoustic interference between individual sensors. Therefore the hardware provides the ability to choose the exact trigger time of each sensor. In most circumstances the sensors are mounted such that they point away from each other. In this case all sensors are triggered at the same point in time. At present a measurement rate of 9 Hz is used, which is based on the following calculations: For very good reflectors we can assume a maximum operating range of 8 m, which corresponds to a time of flight of sound in air of approximately 50 ms. Thus echoes are considered during a receiving period of T_{rec} = 50 ms after triggering the sensor. In order to avoid measurement errors due to multiple echoes, only the range of the first echo is measured. The sensors are retriggered after an additional wait period of T_{wait} = 60 ms, which ensures that all detectable echoes from previous pulses are attenuated below the detection threshold. Thus the total cycle time $T = T_{rec} + T_{wait}$ = 110 ms. Each sensor measurement is tagged with the position of the vehicle. At present the Navlab uses dead reckoning to estimate its position relative to some initial point. The distance travelled is provided by an optical encoder on the drive shaft and vehicle orientation in 3-D space is provided by the gyroscope of an inertial navigation system. The measurements are combined to give the x-y position and orientation of the vehicle with respect to world coordinates. For the position tag of a sonar measurement only the following three

parameters are used: x, y and φ, where φ = yaw (Refer to Fig. 1).

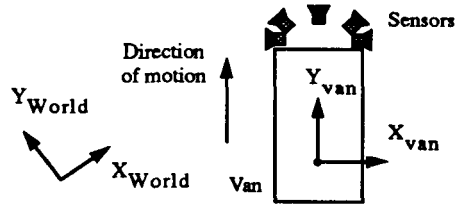


Fig. 1 Sensor Configuration

The hardware of the sonar system consists of an interface module which triggers the sensors and measures the time of flight to the first echo returned. The interface module is accessed via VME bus from a 68020 based CPU. This processor runs as a slave of the vehicle controller processor in a real time environment and takes care of data aquisition, conversion to range, position tagging and proper timings. The map building and tracking algorithms are presently implemented on a Sun SPARC station which communicates with the vehicle controller via ethernet. Ethernet communication time uncertainties can be neglected because of the comparatively long cycle time of the sonar system.

III. SONAR MAP

The previously described sensor system can now be used to build a local grid map. The grid map is called local because it contains only information about the immediate surrounding of the vehicle. The vehicle position is kept at a fixed point in the map. As the vehicle moves, objects in the map are moved from cell to cell relative to vehicle position. Once an object falls outside the map boundary it is discarded and the information is lost. Using just a local map has the advantage that error accumulation owing to dead reckoning is kept small, since only relative movements are considered. On the other hand the disadvantage is that information is lost and thus no global information is available. However, if desired, sequences of the output from the local map could be combined and included in a larger global map. At present the area covered by the local map is 16.4 m x 16.4 m. Each grid cell is taken to cover an area of 0.4 x 0.4 m^2. Hence the map consists of 41 x 41 cells (See Fig. 2). .

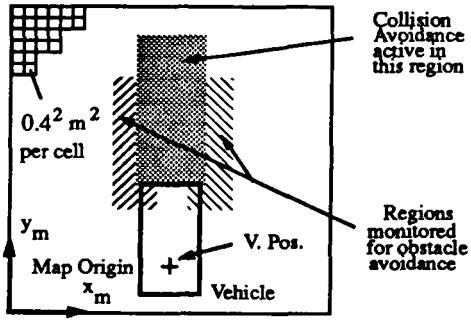


Fig. 2 Sonar Map

The reason for taking such a coarse resolution for each cell was that most applications of the system do not require a high accuracy. Also, the sensor itself does not always provide a highly accurate measurement. For the particular sensor chosen, the measurement accuracy is about ±1 cm. However, depending on the environment, the sensor may also deliver noisy results. The reason lies in the poor angular resolution of the sensor. Echoes may be detected from an object at far range in the main lobe or from a good reflector at closer range in a side lobe. Depending on the relative signal strenghts and movement of the sensor, the range reading may oscillate. A similar effect can also happen when an echo reflected multiple times is received.

Each cell has a set of parameters or annotations associated with it, which are described below:

1. **Object Type**
   This parameter is used to indicate if the object in that cell was seen at the current sensor reading, or if it was seen at a previous reading. If it was seen only at a previous reading, then the object type indicates that it must have moved to that particular cell due to vehicle motion only.

2. **Position**
   Indicates the x-y position of the object with respect to vehicle position.

3. **Count**
   This parameter counts the number of times an object was detected in a particular cell.

The resolution of the grid is fairly coarse and hence a position parameter (X_{obj}, Y_{obj}) is kept to avoid gross error accumulation when objects are transformed in the map. Only one object is kept per grid cell. Thus measurement uncertainty is part of the grid cell representation and any object detected within an area covered by a particular cell is taken to belong to the same object.

Following, a short description of object transformation within the map is given: Vehicle position and orientation are kept constant within the map. Therefore objects in the map move with respect to the vehicle. The vehicle's positioning system returns vehicle position and orientation with respect to a global frame of reference (x, y) that is determined at the time of initialization of the system. Since we are interested only in the movement of objects with respect to the vehicle's coordinate system (x_m, y_m), the appropriate transformation is obtained by using the increment in travelled distance δs and orientation $\delta\varphi$. Depending on the type of positioning system of the vehicle, errors due to dead reckoning are reduced and the positioning system can be initialized independently by using this method. Hence if the vehicle moves from a position in a plane (x_1, y_1, φ_1) to a new position (x_2, y_2, φ_2), then an object in the map at position (x_{m1}, y_{m1}) is transformed to position (x_{m2}, y_{m2}) as follows (Refer to Fig. 3):

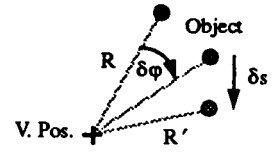


Fig. 3 Object Transformation in Sonar Map

Position and orientation increment:

$$\delta s = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\,, \qquad \delta\varphi = \varphi_2 - \varphi_1$$

Transformation parameters:

$$R = \sqrt{x_{m1}^2 + y_{m1}^2}\,,\ \alpha = \operatorname{atan}\left(\frac{y_{m1}}{x_{m1}}\right),\ \alpha_T = \alpha - \delta\varphi \qquad (1)$$

New object position (vehicle moving forward):

$$x_{m2} = R \cdot \cos\alpha_T$$

$$y_{m2} = R \cdot \sin\alpha_T - \delta s \qquad (2)$$

After the position of objects in the map is updated, new objects detected by the sensors are added. A sensor measures the range R to an object. Position and orientation of each sensor on the vehicle are known. Hence, using the transformation 'vehicle position → sensor position → map', a new object is placed in a cell in the map (Refer to Fig. 4). If that particular cell is already occupied, then only the cell parameter 'Count' is updated, otherwise all cell parameters are updated.

The map parameter **Count** is used for differentiating between moving and stationary objects and for filtering the data. **Count** can also be used to evaluate the confidence that a particular cell is occupied by an object. A higher value of **Count** indicates a higher confidence. In the case of collision avoidance for example it is desirable to slow down the vehicle if there is an obstacle in front and to resume driving if the obstacle moves away, like a car or person could. Hence objects in the map in a sensor's field of view are deleted if a sensor does not detect them anymore. However an object will not be deleted instantly, but only after it was not seen by the sensor for a certain time period. This time period is defined by a parameter called **Life Time**, which is given as the number of cycles of the update sequence (Refer to Fig. 6). We can assume that the cycle period is approximately constant. The parameter **Count** is therefore updated as follows:

If an object is detected, then

1. Initially, Count_t = Life Time
2. Otherwise, $\text{Count}_t = \text{Count}_{t-1} + 1$
3. Also, Decay Amplitude = 0

If no object is detected by the sensor and Count is not equal to zero, then

1. Calculate Decay Amplitude initially, i.e. if it is zero:
   Decay Amplitude = Count_t / Life Time

   Decay Amplitude is calculated only once at the beginning of a decay sequence. It ensures that each object disappears after the same amount of time, i.e. Life Time, has passed.
2. $\text{Count}_t = \text{Count}_{t-1}$ − Decay Amplitude

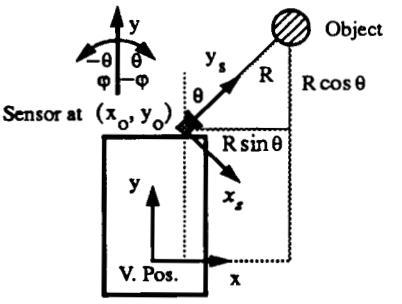


Fig. 4 Sensor to Object Transformation

At present, the decay algorithm is applied only to objects appearing in the area directly in front of the vehicle. Moving objects appearing in this area are of most concern for safe vehicle navigation. Therefore the velocity of the vehicle is reduced as a function of range to the closest object detected in this area. The velocity is set to zero if the closest range is less than a certain minimum distance. The decay algorithm ensures that the vehicle resumes driving when the obstacle moves away. Fig. 5 shows a plot of the percentage of vehicle velocity set versus closest range. D_{max} corresponds to the maximum sensor operating range.

The parameter **Count** can also be used to eliminate spurious echoes such as the ones returned by rain droplets. In this case an object is supposed to be actually present only, if it has been seen consecutively for a certain number of cycle times. Since rain droplets return echoes at random ranges as time progresses, these returns can be filtered out and will not appear as ghost objects in the map.

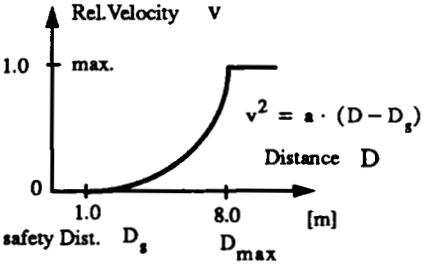


Fig. 5 Velocity control

Of course this procedure does not work in a heavy downpour since the number of rain drops in the environment just becomes too large.

The following figure shows the cycle for map operations:

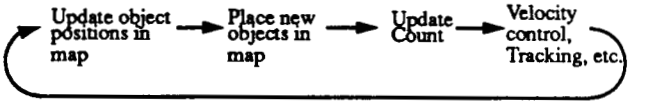


Fig. 6 Updating sequence for sonar map

The parameter **Object Type** is used to indicate the type of operation that was performed on an object in a particular grid cell. These operations can be transformations as mentioned before, or filtering operations like the ones decribed in the next section.

At present, each grid cell is associated with only three parameters. The map structure allows an extention to other annotations (pointers to data structures) in the future, such as object characteristics or triggers. In this way the sonar map could easily be integrated into an *Annotated Map* such as the one described in [7].

IV. FEATURE SELECTION AND TRACKING

The data collected by the sonar map can now be used for autonomous vehicle navigation functions. A basic navigation function is the tracking of features in the environment and using this information to determine a path that the vehicle can drive. The following paragraphs describe a method by which the vehicle uses its sonar sensors to drive on a path parallel to a feature such as a wall, railroad track or parked cars.

Fig. 9 shows data collected in the sonar map when tracking cars parked on the right hand side of the road. As can be seen in the figure, the side of the cars facing the road is fairly well detected. Usually sonar does not detect smooth surfaces very well because of specular reflections. However, in most real world environments such as this one, there are no perfectly smooth surfaces. In this case the sonar receives echo returns from corners and projections like door handles, mirrors, wheels, etc..

The vehicle is to be driven on a path parallel to the curve formed by the parked cars, keeping a constant distance from the cars (usually around 3 m). Therefore the parameters of that curve have to be calculated first, using the data from the sonar map. For reasons of computational simplicity and decreased noise sensitivity a least square fit of a straight line was chosen. All computations are performed with respect to the vehicle origin which is at a fixed position in the sonar map. Data points for the line fit are selected by choosing only data points that appear in a specific area in the sonar map. Thus for the environment represented in Fig. 9, only the right half of the map is searched for data points. The **Position** parameter gives a data point in the map in terms of vehicle coordinates. Since the direction of vehicle motion is along the y-axis, a line parallel to the vehicle would have a slope of $m = \infty$ (Refer to Fig. 2; Note that this coordinate system is defined by the vehicle's position system). To avoid this inconvenience, the vehicle coordinate system is rotated anticlockwise by 90°. All following computations are now performed with the transformed coordinates $y = -x_{old}$ and $x = y_{old}$. Therefore the selected data points can now be represented as a discrete function $y_i = f(x_i)$. The sonar sensors sometimes return a spurious echo. These outliers generally degrade the performance of a least square fit. Hence a median filter is applied first on the data points given by $y_i = f(x_i)$. The filter is applied twice, using window sizes three and five cells. The two parameters of the straight line $y = mx + c$ can now be found by using standard formulae for a least square fit for n data points (x_i, y_i):

$$m = \frac{n\sum x_i y_i - (\sum x_i)(\sum y_i)}{n\sum x_i^2 - (\sum x_i)^2}$$

$$c = \frac{\sum y_i}{n} - m\frac{\sum x_i}{n} \qquad (3)$$

Also, the standard error of estimate is given by:

$$s_0 = \sqrt{\frac{\sum [y_i - (mx_i + c)]^2}{n-2}} \qquad (4)$$

The data trajectory parameters m and c are stored and updated during each system cycle (Refer to Fig. 6). The line can now be used by a path tracker to steer the vehicle (Refer to [1]). To ensure a consistent steering response, the line fit also has to be checked for validity. It may happen that the sensors do not detect any features for some driving distance. One reason in the case of parked cars for example may be a gap between cars and no reflectors on the road edge. Here the line fit will produce no valid output. The standard error s_0 is used to provide a measure of how well the data points could be fitted. If s_o is too high then again the output of the line fit is not taken to be valid. In these cases the old path parameters are used and the vehicle will continue driving in the previously calculated direction until it encounters features again. Since the old path parameters are referenced to the vehicle position they still have to be updated to compensate for vehicle movement during one system cycle. The position and orientation increment during one system cycle is known from . Hence m is transformed as follows:

$$m_{new} = \tan(\varphi - \delta\varphi), \qquad \varphi = \operatorname{atan}(m) \qquad (5)$$

In order to obtain c_{new}, point P (x_1, y_1) is selected on the path where it intersects the y-axis (for convenience) as shown in Fig. 7. Using , (1) and (2) the coordinates of P in the new coordinate system (x_{1new}, y_{1new}) can be calculated. Hence c_{new} can be calculated by

$$c_{new} = y_{1new} - m_{new}\, x_{1new} \qquad (6)$$

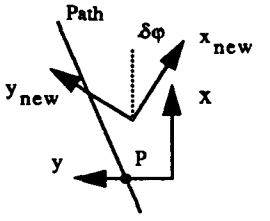


Fig. 7 Path parameter transformation

The method described above worked well in an environment that provided good reflectors and continous smooth features. However, especially in the case of parked cars, problems arise when gaps are encountered or reflectors that do not belong to the feature being tracked are near by. A typical situation is shown in Fig. 8. In the least square fit sequence 1 - 4 , only line fits 1 and 4 track the feature. Type A data does not belong to the feature being tracked, type B data is due to corner effects at a gap and type C data is a noisy range measurement. Type B data can lead to an undesired least square fit as shown by line fits 2 and 3. In order to reduce these errors, obtain a smoother steering response and make the system more robust against outliers, the values obtained from the least square fit are filtered and path parameters are updated by

merging new information with past values. The method is described in the following paragraphs:

An initial noise reduction is achieved by selecting data points only from a small window in the sonar map: As we have some knowledge about the environment the vehicle is driving in, we can predict up to a certain degree where we should look for valid data in the map. This means in practice that only data points within a certain distance from the data trajectory are taken. This procedure removes outliers of type A as shown in Fig. 8. Furthermore a point is selected only if it is within a certain distance and direction from previously selected adjacent points. This method removes outliers of type C and ensures that most points are already grouped close to a line segment.

Again a straight line is fitted using (3). For each distance δs_i travelled, we obtain a new value of gradient m_i. The change in gradient is then given by

$$\delta m_i = m_i - m_{i-1} \tag{7}$$

and $\delta\varphi_i$ is the corresponding change in vehicle orientation during that interval. Since not all type B data is removed, there still may remain a problem with sudden large changes in orientation as shown by the least square fit sequence 1 - 4 in Fig. 8.

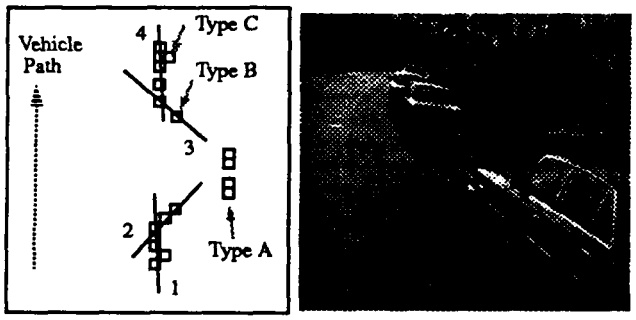


Fig. 8 Removing outliers from least square fits:
(a) Map display, (b) Typical corresponding scene

These sudden changes in orientation basically appear as median noise. They can be filtered out by putting current and past readings of δm into a buffer of N values, passing a median filter across and then averaging over N values. The buffer is implemented as an ordered set,

$$m_{Buffer} = \{\delta m_i,\ \delta m_{i-1},\ \delta m_{i-2},\ \ldots,\ \delta m_N\} \tag{8}$$

During each update sequence, all elements are shifted right by one, the last element being discarded and the current result replacing the first element. Median filtering is achieved by replacing element δm_{i-k} with a median value, where $2k+1$ is the maximum possible window size of the filter. Since values towards the right in the buffer represent increasingly earlier points in time/distance, successive applications of a median filter can be achieved by replacing elements δm_{i-k-t} with the filtered value, where t is the discrete shift in time/distance. The new value $m_{Path}^{\{new\}}$ of the path parameter is then computed, compensating also for change in vehicle orientation during the averaging interval,

$$m_{Path}^{\{new\}} = \frac{1}{N} \cdot \sum_{k=1}^{N} \tan\left(\operatorname{atan}(\delta m_k) - \sum_{i=1}^{k} \delta\varphi_i\right) \tag{9}$$

m_{Path} is then updated by merging the current and the new value for m_{Path} using a weighted average,

$$m_{Path}^{\{t+1\}} = \frac{m_{Path}^{\{new\}} + w \cdot m_{Path}^{\{t\}}}{1+w} \quad \text{, where } w \geq 0 \tag{10}$$

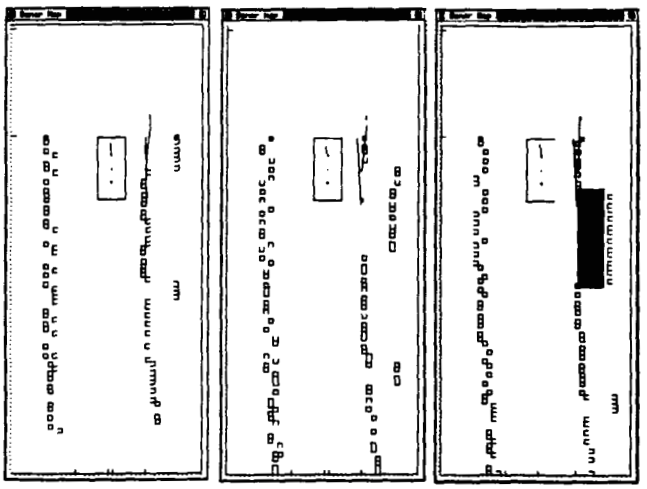
In a similar fashion as described above, the path parameter c_{Path} is updated. The weight w and the number of values N control how close the path parameters m_{Path} and c_{Path} should follow the actual data points. If a lot of noise is present in the data, the path parameters should be influenced only slightly, whereas if little noise is present, the data should be followed closely. An estimate of the noise present is given by the standard error s_0 from (4) and w and N are adjusted accordingly. As a result the steering of the vehicle is now much smoother.

A drawback of the method is its slow reaction to a relatively sudden change in road direction. This effect is also due to the short range of the ultrasonic sensors and the fact that almost no echoes are received at large angles of incidence. In the case of features being tracked on the right side, the system is able to handle curves to the right since data points slowly move away from the vehicle and the vehicle can follow with a slight delay. On the other hand a curve to the left poses a problem because by the time the vehicle recognizes that it should change direction, it has usually come already too close to the feature and has no space left to make a sharp left turn anymore. For this reason a monitor is added which monitors a particular area for objects towards the right hand side of the vehicle as indicated in Fig. 2. If objects are detected in this area, a steering radius R_m necessary to avoid the closest object is calculated. The value of R_m is given by a relation between R_m and the closest distance D_c similar to the one for velocity control as shown in Fig. 5. Velocity is here replaced by steering radius and for example for D_c between 0 and 2 m, R_m varies between -20 and -100 m. D_c is defined as the closest distance between the objects and a point at the front right edge of the vehicle. If R is now the steering radius calculated by a path tracker as described in [1] and the following condition is true:

$$R > 0 \quad \text{or } R < R_m$$

then R_m overrides R and the vehicle follows steering commands issued by the monitor. Otherwise the path tracker takes over again. Similarly a monitor can be used when tracking features on the left side. Note that steering radii are negative when turning left and positive when turning right.

In general the system performs well when tracking a wall or a feature that changes curvature smoothly. When tracking parked cars it does not perform as well in curves and fails when the road curvature becomes sharp. The reason here is that often parked cars are not very well aligned and even on straight road stretches parked at different angles to each other. The maximum range of the sensors is simply too short to detect these configurations well enough. Fig. 9 shows a preliminary result of the vehicle tracking parked cars, detecting a gap and preparing for reverse parking.



(a) (b) (c)

(d)

Fig. 9 Tracking parked cars on the right hand side and searching for parking space: (a) Approaching gap, (b) Detecting gap, (c) Preparing vehicle for parking maneuvre, (d) Typical street scene.

V. Results and Conclusions

The sonar system successfully drove the Navlab on a dirt road next to a railroad track, using the railroad to guide the vehicle. Parking or docking maneuvres are also important autonomous vehicle tasks. At present the sonar system is used to drive the vehicle parallel to parked cars in a city street. Eventually the system should be able to detect a parking space and autonomously park the vehicle. The sonar is also successfully integrated into two other systems that drive the Navlab. The first one is YARF ([3]), which drives the robot on city streets. YARF uses colour vision for road following but cannot detect if obstacles obstruct the vehicle's path and thus cannot adjust the velocity accordingly. The sonar system takes over that task and sends velocity commands to YARF via the EDDIE toolkit ([9]). The sonar is also integrated into the new architecture DARN (Distributed Architecture for Robot Navigation) of the Navlab. The outputs of several modules that can drive the vehicle are used here to decide on an optimum path for the vehicle ([6]). The sonar module sends all object positions with respect to vehicle position from the sonar map and the system selects an obstacle free path. Communication is again facilitated via the EDDIE toolkit.

The sonar system proved to work reliably in a variety of different situations. There were no major problems with false returns or sensor noise that could not be dealt with. One reason is most probably that an outdoor environment like a road is generally less cluttered than most indoor environments where autonomous vehicles are used. Outdoor objects tend to be large, having usually enough corners and projections that reflect ultrasound well. Care has to be taken in avoiding reflections from the ground. This problem can be solved in most cases by mounting the sensor high enough above the ground and pointing it slightly upward. The system can also be easily integrated with other vehicle navigation systems or adapted to other vehicles.

A drawback of using ultrasound in air is the limitation of range and data update due to high attenuation and low speed of sound. At present this fact does not matter that much since the system is used only at low vehicle speeds. However, the current system design is not limited to using only ultrasonic sensors. A microwave radar, laser scanner or stereo camera can be used as well. In the future therefore we intend to integrate some of these sensors.

Aknowledgements

This research is partly supported by contracts from DARPA, titled "Robot System Development and Testing" (monitored by TACOM) and "Perception for Outdoor Navigation" (monitored by ETL), and partly supported by a grant from NSF titled "Annotated Maps for Autonomous Underwater Vehicles". The authors would also like to thank Siemens AG for providing some of the sensors and helpful information.

References

[1] Omead Amidi. *Integrated Mobile Robot Control*. Technical Report, Robotics Institute, Carnegie Mellon University, 1990.

[2] A. Elfes. A Sonar-Based Mapping and Navigation System. In *Proc. IEEE Conference on Robotics and Automation*, 1986.

[3] Karl Kluge and Charles Thorpe. Explicit Models for Robot Road Following. *Vision and Navigation: The Carnegie Mellon Navlab*. Kluwer Academic Publishers, 1990, Chapter 3.

[4] J. Leonard and H. Durrant-Whyte. Application of Multi-Target Tracking to Sonar-based Mobile Robot Navigation. In *Proc. IEEE Conference on Decision and Control*, 1990.

[5] V. Magori, H. Walker. Ultrasonic Presence Sensors with Wide Range and High Local Resolution. In *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, Vol. UFFC-34, No. 2, 1987

[6] David W. Payton, Kenneth Rosenblatt and David M. Keirsey. Plan Guided Reaction. *IEEE Journal on Systems, Man and Cybernetics*. December 1990

[7] C. Thorpe and J. Gowdy. Annotated Maps for Autonomous Land Vehicles. In *Proceedings of DARPA Image Understanding Workshop*, Pittsburgh PA, September 1990.

[8] Charles Thorpe, Martial Hebert, Takeo Kanade and Steven Shafer. Toward Autonomous Driving: The CMU Navlab. Part I - Perception. In *IEEE Expert*, August 1991.

[9] Charles Thorpe, Martial Hebert, Takeo Kanade and Steven Shafer. Toward Autonomous Driving: The CMU Navlab. Part II - Architecture and Systems. In *IEEE Expert*, August 1991.