Building and navigating maps of road scenes using an active sensor [1]

Martial Hebert

The Robotics Institute Carnegie Mellon University Pittsburgh, PA 15213

Abstract

This paper presents algorithms for building maps of road scenes using an active range and reflectance sensor and for using the maps to traverse a portion of the world already explored. Using an active sensor has some attractive advantages: It is independent of the illumination conditions, it does not require complex calibration in order to transform observed features to the vehicle's reference frame, and it provides 3-D terrain models as well as road models. Using the map built from sensor data facilitates navigation in two respects: The vehicle may navigate faster since less perception processing is necessary, and the vehicle may follow a more accurate path since the navigation system does not rely entirely upon inaccurate visual data. We present a complete system that includes road following, map building, and map-based navigation using the ERIM laser range finder. We report on experimentations of the system both on the CMU NAVLAB and the Martin Marietta ALV.

1 Introduction

Autonomous road following using visual information is an important application of mobile robots. In addition to navigating on roads, the visual information can be used to build maps of the observed environment. An area of research that has not been explored is to close the loop by using the map built from previous observations to guide the navigation on a portion of the world already explored. Such a capability of map based navigation would enable us to improve the performances of the vehicle in three directions:

- Faster navigation: Perception is typically the bottleneck in autonomous mobile systems because images have to be processed as often as possible to compensate for the lack of knowledge about the world. If apriori knowledge of the environment is available from previous observations, perception is needed only to periodically check that the vehicle stays on the path prescribed by the map. The perception bottleneck is therefore reduced, thus leading to faster navigation.
- More reliable navigation: Autonomous navigation is unreliable because of the uncertainty associated with any sensor data and processing. Relying more on a map means relying less on sensor data acquired during the execution of a navigation plan. Map based navigation should therefore provide more accurate navigation.
- Simpler perception: A map can provide the expected appearance of the environment at any location. That includes the expected location of objects, and the expected position and appearance of the road. This additional knowledge allows for simpler perception processing.

Athough map based navigation algorithms could be used with a man made map (*e.g.* from surveying), using a map built from sensor information does not make any assumptions on the amount of knowledge available to the system, thus leading to a fully autonomous system. This is also important since it is difficult to obtain the resolution of a map built from sensor data by using surveying alone.

Most of the existing road following systems are based on intensity or color image processing [12, 16, 13]. In this paper, we investigate the use of active sensing, namely laser range finding, for both road following and map building. Using such a sensing modality has some attractive features such as its stability with respect to illumination conditions and the direct conversion to world coordinates without calibration. Our goal is therefore to build a complete system from road following to map building using active sensing, whereas previous research on active sensing for autonomous vehicles focused on 3-D map building or obstacle detection [2, 3, 6, 4].

The images used in the experiments reported in this paper are range and reflectance images from a laser range finder, the ERIM scanner [15]. The images are 64 rows by 256 columns 8-bit images. The maximum range is 64 feet corresponding to a pixel value of 255. The vertical (resp. horizontal) field of view is 30° (resp. 80°).

Even though the road following programs were demonstrated on the Martin Marietta vehicle (the ALV), all results presented in this paper were obtained using the Carnegie-Mellon vehicle (the NAVLAB) [9].

2 Following roads using active reflectance images

Early work on road following from active sensing focused on the use of range data to find the edges of the road [10, 1]. The drawback of this approach is that it assumes that the road is limited by edges that correspond to discontinuities of the terrain surface. This assumption limits severely applicability of the algorithms. An alternative approach is to use the active reflectance images for road following. Active reflectance images have two

[1] This research was sponsored by the Defense Advanced Research Projects Agency, DoD, through ARPA Order 5351, monitored by the US Army Engineer Topographic Laboratories under contract DACA76-85-C-003

CH2750-8/89/0000/1136$01.00 © 1989 IEEE

characteristics that make them attractive for road-following applications: First, they are insensitive to outside illumination, that is no shadows are cast by objects in reflectance images and the influence of the level of ambiant light on the image is minimal (in fact, any program using reflectance images would work as well under night conditions). Second, each pixel in the reflectance image is also a range pixel whose position in space can be derived from the geometry of the scanner. This allows us to compute the position of the edges of the road found in a reflectance image in the vehicle's 3-D world without any of the calibration procedures that are typical of the video-based road following algorithms [5].

Edge detection would be the natural way of finding road edges in grey level images. The nature of the reflectance data, however, suggests the use of a region-based technique for two reasons: First, the dynamic range of the image is low, many spurious edges that are of similar strength as the road edges will be found. Second, the intensity of the road in reflectance images is very stable because it is insensitive to shadows and changes in illumination. This is to be compared with video images in which the appearance of the road region varies significantly, thus requiring the use of multiple classes of road and non-road regions [12]. Instead of extracting the road edges directly, a road region extractor identifies the pixels that are part of the road based on the road location and appearance predicted from a previous image.

The principle of the road region extractor is to keep the mean m_i and variance σ_i of the reflectance values inside the road region for each group i of four scanlines in the image. The statistics are computed on groups of scanlines instead of the entire image in order to account for the intensity attenuation at long range and for the presence of small markings on the road that would have an effect on a few scanlines only instead of propagating to the entire road region. The statistics are computed on the first image by selecting the road region interactively. The road region is extracted from the reflectance image by thresholding the pixels in swath i that are between $m_i + \sigma_i$ and $m_i - 2 * \sigma_i$, where (m_i, σ_i) are the values computed on the previous image. The resulting binary image is then processed to remove small isolated regions. The road region is extracted from the set of remaining regions by using three criteria: the shape of the boundary that is the value of the elongation, the size of the region knowing the average with of the road, and the position of the region in the image as predicted from the previous image. Once the road region is extracted, the values m_i and σ_i are computed for each swath in order to process the next image. This algorithm is similar to [12, 13], except that is uses only one road class, that is only one set of statistics, and that it computes and predicts the road appearance over small swaths instead of the entire image.

The final output of the road finder program is the direction of the center line of the road. The line is computed by fitting two parallel lines to the left and right edges of the road polygon. If the two lines are parametrized by a direction $\vec{v}$, which is the direction of the road common to both edges, and the signed distances of the lines to the origin d_l and d_r (Figure 1), the best fit is computed by minimizing:

$$\sum_{leftedge} (p_l.\vec{v} + d_l)^2 + \sum_{rightedge} (p_r.\vec{v} + d_r)^2 \qquad (1)$$

The center line of the road is the middle line of $(\vec{v}, d_l)$ and $(\vec{v}, d_r)$, the width of the road is $w = |d_l - d_r|$. Figure 2 shows the result of the road following program on a typical reflectance image. The top part of Figure 2 shows the reflectance image, the bottom part shows the road edges and the center line of the road projected on the ground plane. The scale on the left side on the image is in meters; the road is found between five and fifteen meters in this case.

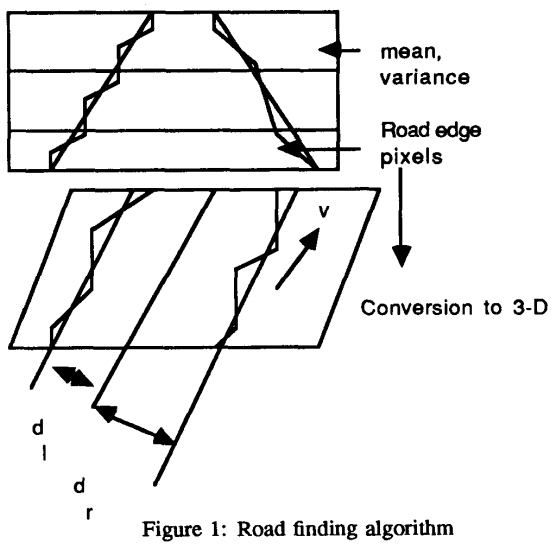

Figure 1: Road finding algorithm

In order to drive the vehicle, two points on the center line are sent to a local path planner. The path planner generates a sequence of circular arcs using a "pure pursuit" algorithm derived from [14]. The road following program drove successfully two vehicles, the CMU NAVLAB [8] and the Martin Marietta ALV, over several hundred meters at a speed of 40cm/s. In both cases, the road following is implemented on a Sun3 workstation. The average computation time is 3 seconds per reflectance image which allows for enough overlap between consecutive images.

3 Building maps from range and reflectance images

The main problem in building a map from a sequence of consecutive images is to compute the relative positions of features observed from different vantage points in order to merge them in a consistent map expressed in a single coordinate system. Two types of information may be used to compute the relative positions: The matching of geometric features from image to image, and the best estimate of the current position of the vehicle as given by the dead reckoning. The position estimate from the

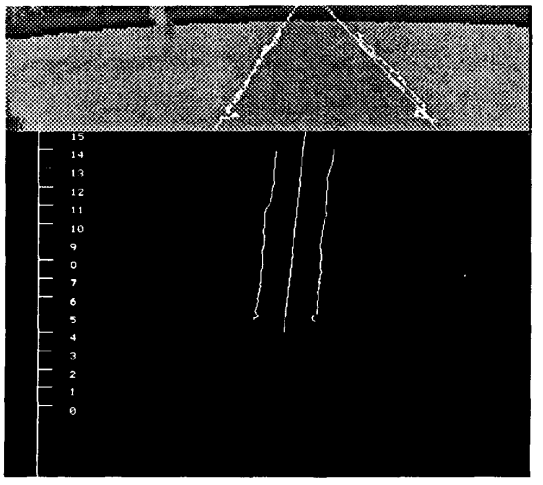

Figure 2: Road following on a reflectance image

motion of the vehicle cannot be used alone unless a sophisticated navigation system is used as in [3] since positional errors do accumulate in time, thus leading to unacceptable errors in the position estimate. The final position estimates should be a combination in which the estimate from the dead reckoning is used to predict matches between features, and a set of consistent matches is used to estimate the resulting displacement between images. In general, if F_i^1 and F_j^2 are two sets of features extracted from two images, I_1 and I_2, we want to find a transformation $\hat{T}$ and a set of pairs $C_k = (F_{i_k}^1, F_{j_k}^2)$ such that $F_{j_k}^2 \approx \hat{T}(F_{i_k}^1)$, where $T(F)$ denotes the transformed by T of a feature F. We first investigate the feature matching algorithm independently of any particular feature type so that we can then apply it to any level of terrain representation.

For each feature F_i^1, we can first compute the set of features F_{ij}^2 that could correspond to F_i^1 given an initial estimate T_0 of the displacement. The F_{ij}^2's should lie in a prediction region centered at $T_0(F_i^1)$. The size of the prediction region depends on the confidence we have in T_0 and in the feature extractors. For example, the centers of the polygonal obstacles are not known accurately The confidence on the displacement T is represented by the maximum distance δ between a point in image 1 and the transformed of its homologue in image 2, $\|Tp^2 - p^1\|$, and by the maximum angle ϵ, between a vector in image 2 and the transformed of its homologue in image 1 by the rotation part of T. The prediction is then defined as the set of features that are at a Cartesian distance lower than δ, and at an angular distance lower than ϵ from $T_0(F_i^2)$. The parameters used to determine if a feature belongs to a prediction region depend on the type of that feature. For example, we use the direction of a line for the test on the angular distance, while the center of an obstacle is used for the test on the Cartesian distance. Some features may be tested only for orientation, such as lines, or only for position, such as point features. The features in each prediction region are sorted according to some feature distance $d(F_i^1, T_0(F_{ij}^2))$ that reflects how well the features are matched. The feature distance depends also on the type of the feature: for points we use the usual distance, for lines we use the angles between the directions, and for polygonal patches (obstacles or terrain patches) we use a linear combination of the distance between the centers, the difference between the areas, the angle between the surface orientations, and the number of neighboring patches. The features in image 1 are also sorted according to an "importance" measure that reflects how important the features are for the matching. Such importance measures include the length of the lines, the strength of the point features (*i.e.* the curvature value), and the size of the patches. The importance measure also includes the type of the features because some features such as obstacles are more reliably detected than others, such as point features.

Once we have built the prediction regions, we can search for matches between the two images. The search proceeds by matching the features F_i^1 to the features F_{ij}^2 that are in their prediction region starting at the most important feature. We have to control the search in order to avoid a combinatorial explosion by taking advantage of the fact that each time a new match is added both the displacement and the future matches are further constrained. The displacement is constrained by combining the current estimate T with the displacement computed from a new match (F_i^1, F_{ij}^2). Even though the displacement is described by six components, the number of components of the displacement that can be computed from one single match depends on the type of features involved: point matches provide only three components, line matches provide four components (two rotations and two translations), and region matches provide three components. We therefore combine the components of T with those components of the new match that can be computed. A given match prunes the search by constraining the future potential matches in two ways: if connectivity relations between features are available, as in the case of terrain patches, then a match (F_i^1, F_{ij}^2) constrains the possible matches for the neighbors of (F_i^1) in that they have to be adjacent to F_{ij}^2. In the case of points or patches, an additional constraint is induced by the conservation of the relative placement of the features in the scene: if F_1 is to the left of F_2 in one image, then the corresponding features F_1' and F_2' must be ordered in the same way in the next image, provided that the images a close enough and that the features are far enough from each other.

The result of the search is a set of possible matchings, each of which is a set of pairs $S = (F_{i_k}^1, F_{j_k}^2)_k$ between the two sets of features. Since we evaluated T simply by combining components in the course of the search, we have to evaluate T for each S in order to get an accurate estimate. T is estimated by minimizing an error function of the form:

$$E = \sum_k d(F_{i_k}^1 - T(F_{i_k}^2)) \qquad (2)$$

The distance $d(.)$ used in Equation (2) depends on the type of the features involved: For point features, it is the usual distance between two points; for lines it is the weighted sum of the angle between the two lines and the distance between the distance vectors of the two lines; for regions it is the weighted sum of the distance between the unit direction vectors and the distance between the two direction vectors. All the components of T

can be estimated in general by minimizing E. We have to carefully identify, however, the cases in which insufficient features are present in the scene to fully constrain the transformation. The matching S that realizes the minimum E is reported as the final match between the two maps while the corresponding displacement $\hat{T}$ is reported as the best estimate of the displacement between the two maps. The error $E(\hat{T})$ can then be used to represent the uncertainty in T.

This approach to feature based matching is quite general so that we can apply it to many different types of features, provided that we can define the distance $d(.)$ in Equation (2), the importance measure, and the feature measure. The approach is also fairly efficient as long as δ and ϵ do not become too large, in which case the search space itself becomes large.

In addition to the road edges, the features that we consider for map building are polygons that describe the surface of the terrain and the discrete obstacles. The algorithms for extracting the polygonal description are reported in [7] and [6]. To summarize, the features used in the matching are:

- The polygons describing the terrain parametrized by their areas, the equation of the underlying surface, and the center of the region.
- The polygons describing the trace of the major obstacles detected (if any).
- The road edges found in the reflectance images if the road detection is reliable enough. The reliability is measured by how much a pair of road edges deviates from the pair found in the previous image.

The obstacle polygons have a higher weight in the search itself because their detection is more reliable than the terrain segmentation, while the terrain regions and the road edges contribute more to the final estimate of the displacement since their localization is better. Once a set of matches and a displacement T are computed, the obstacles and terrain patches that are common between the current map and a new image are combined into new polygons, and the new features are added to the map while updating the connectivity between features.

In the current implementation, the initial estimates of the displacement T_0 are taken from the central database that keeps track of the vehicle's position using dead reckoning. The size of prediction region is fixed with δ = one meter, and $\epsilon = 20°$. This implementation of the feature matching has performed successfully over the course of runs of several hundred meters. The final product of the matching is a map that combines all the observations made during the run, and a list of updated obstacle descriptions that are sent to a map module at regular intervals. Since errors in determining position tend to accumulate during such long runs, we always keep the map centered around the current vehicle position. As a result, the map representation is always accurate close to the current vehicle position. As an example, Figure 4 shows the result of the matching on five consecutive images separated by about one meter. The scene in this case is a road bordered by a few trees. Figure 3 shows the range images (right) and the reflectance images (left) used in the matching. Figure 4 is a rendition of the combined maps using the displacement and matches computed from the feature matching algorithm. This display is a view of the map rotated by 45° about the x axis and shaded by the values from the reflectance image.

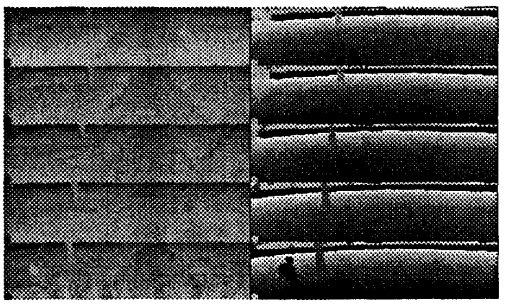

Figure 3: A sequence of range and reflectance images

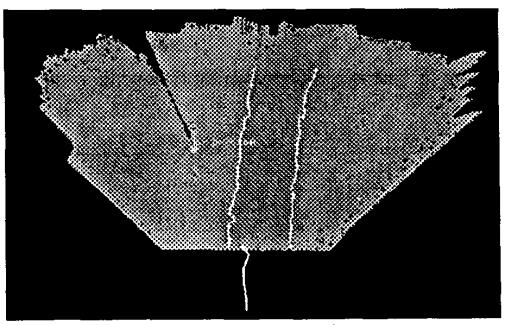

Figure 4: Perspective view of the combined map

Figure 6 shows a map built from twenty images over sixty meters. In this display, only the road edges, the center line of the road, and the discrete obstacles are shown. To obtain this result, the vehicle was driven by the road following algorithm of Section 2 at a continuous speed of 20 cm/s. The road following and map building modules are separated because the map building module requires an average of fifteen seconds of computation time per image which would prevent stable continuous motion. The overall structure of the map building/road following system that was used in this experiment is shown in Figure 5. The map building and road following modules are executed on two separate processors (Sun3's). They both access the ERIM scanner through a network interface. The road following module sends a new path that is a sequence of arcs to a separate helm module running on a third processor. The helm also provides initial estimates of the vehicle position to the map building module. The communications between the helm and the two other modules are handled through the CODGER system [9].

4 Map-based road following

In this Section, we investigate the last part of the system, that is the use of the map built from road following to traverse the same portion of the world.

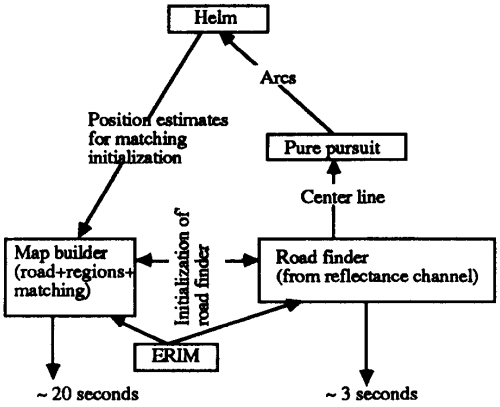

Figure 5: The map building/road following system

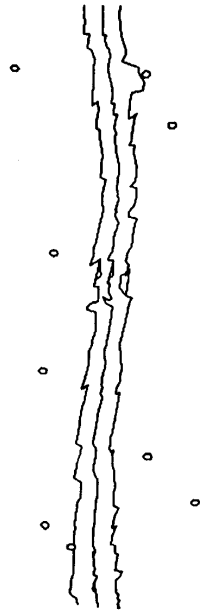

Figure 6: Complete map of a road scene

The map-based road following must proceed in three steps: computation of the starting position, path planning in the map, path execution and correction. The first step is needed to avoid constraining the starting position and heading of the vehicle at the beginning of the traversal of the map to those used to initiate the map building stage. The position and heading of the vehicle with respect to the map are computed by matching the features, road edges and objects, observed in an image taken at the starting position with the features of the map that are predicted to be visible given a rough initial guess of starting position. The matching algorithm is basically the same as the one used for the map building except that in the current implementation, only road edges and discrete obstacles are used. For example, Figure 7 shows the initial guess of the starting position (marked by a cross) and the portion of the road and the obstacles that are used for the matching. The map features are predicted by intersecting the sensor field of view with the map.

Figure 7: Estimation of the starting position and heading

Given the starting position, the second step is to compute a path that follows the road using the map. This step is the most straightforward in that any path planner that provides for smooth paths can be used. For example, Figure 8 shows a path composed of a sequence of circular arcs. The path is computed by dividing the center curve of the road into small segments over which the pure pursuit path planning algorithm of Section 2 is applied.

Once a path is computed, the vehicle is ready to follow the road based on the map. Ideally, the vehicle should be able to correctly execute the path without any perception at all. In practice, however, the vehicle will drift away from the ideal path due to wheel slippage, and the accumulation of small controller errors and numerical errors. Therefore, the position and heading of the vehicle with respect to the map must be recomputed periodically by comparing the features that are actually observed while executing the path and the features that are predicted from the map given the current estimate of the vehicle's position. The question now is how often should we make a position correction, that is take an image, extract road edges and objects, and match them with the map, in order to stay within reasonable bounds of the original path. This problem is the key to map-based navigation: If the corrections are performed too often we are back to the original road following approach and we loose the benefit of

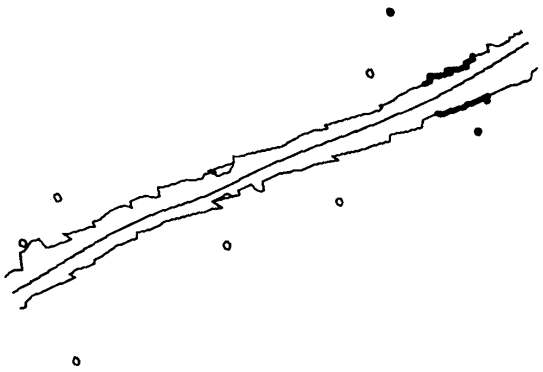

Figure 8: Path planned using a map

having a map. If, on the other hand, we do not perform enough corrections along the path, we may drift significantly far from the nominal path and eventually run off-road. Furthermore, the corrections should be meaningful in the sense that enough features should be present at the time of the correction to ensure that the newly computed position is indeed closer to the truth than the currently available estimate. Several strategies are possible to choose the locations at which corrections should be performed. An attractive strategy is to estimate the uncertainty on the position and heading as the vehicle moves, a new correction is requested whenever the uncertainty reaches a threshold that indicates that the vehicle is too far from its nominal path [11]. This approach guarantees that the distance between the vehicle's path and the nominal path always lies within preset bounds. It does not, however, guarantee that the images taken at the time at which a correction is needed contain enough features of interest. Another possible approach is to make a correction whenever the map predicts that features of interest may be observed from the current position. In our case, it is important to guarantee that the corrections are performed when objects are visible, since otherwise the correction would be computed on the basis of the road edges only and would therefore be ambiguous. A correction is therefore computed whenever at least one object is predicted to be visible from a position along the path. Matching the predicted objects and road edges from the map with the observed road and objects provides an unambiguous new estimate of the vehicle's position and heading. Figure 9 shows the locations at which new images are taken for computing the corrections along the path of Figure 8. The road edges and objects that are matched with the corresponding observed features are shown as bold segments of the road edges and dark circles respectively. The crosses along the path indicate the successive positions of the vehicle at regular intervals of one second (at a speed of 20 cm/s). The position is not displayed if an image is being processed, therefore the gaps in the stream of positions in this display illustrate the time spent in processing images while executing the initial path (The percentage is in reality a bit lower than what appears on this display because the map, range image processing, and helm modules normally run on different processors whereas this display was produced with all the modules executing on one Sun).

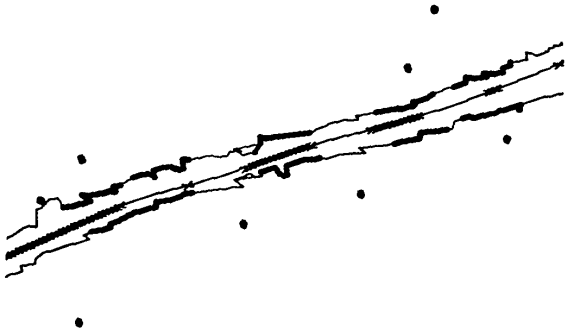

Figure 9: Locations at which images are taken along the path

Computing a correction gives an offset $\Delta = (\Delta x, \Delta y, \Delta\theta)$ between the nominal position and heading and the actual values at the time the image is taken. This offset must be used to correct the current course of the vehicle. This is achieved by shifting the path that has been executed while the image was being processed by Δ, by replanning from the current position as given by the shifted, and by replacing the pending set of motion commands by this new path. Figure 10 illustrates such a sequence of events: As the vehicle comes into view of the first objects, an image is taken and matched against the map, the new position is shown as a cross on the left of the initial path, a new path is planned that takes the vehicle back to its original course.

These results show that it is possible to use a map to efficiently guide the navigation of an autonomous vehicle. The main benefit is that considerably fewer images have to be processed while retraversing the map. For example, The map of Figure 9 requires seven images to be processed. Following the same road at the same speed without the support of a map would require at least 25 images for a displacement of two meters between consecutive images. The reason for the discrepancy is that even if the position of the road were computed perfectly from each individual image, the path planner would not have information far enough in front the vehicle to plan a stable path that is guaranteed to remain on the road. Although the same results could be obtained by using a map that is entered manually, it is important to note that the combination of map building from sensor data and map-based navigation results in a fully autonomous system that can learn its environment and use its new knowledge to navigate it.

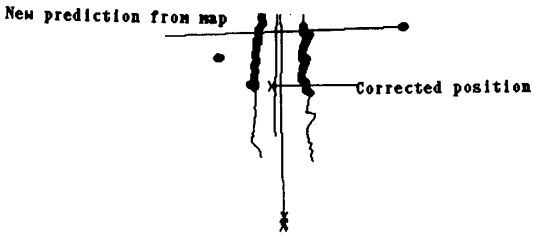

Figure 10: Corrected path

5 Conclusion

The road following and map building system shows that road environments can be efficiently navigated and mapped using an active sensor such as a laser range finder. The map based navigation system shows that the information gathered during a initial traversal of the road can be used to improve the navigation over a portion of the stretch of road already explored. Specifically, using the map provides an initial path to follow, and a list of optimal locations at which visual data should be processed in order to correct the vehicle position that drifts over time. The combination of those three components provide a basis for autonomous navigation of roads including 3-D terrain modeling and knowledge gathering and utilization through map building and map based navigation.

We are currently extending the ideas used in those systems to the case of cross-country navigation and combined on road/off road navigation in which the map contains a representation of terrain regions in addition to the road model and the discrete obstacles. This type of information is currently extracted but it is not used for the map based navigation. The system presented here uses a simple path planner based on the pure pursuit control scheme. Our plan is to use the path planner described in [11] to take into account vehicle model and uncertainty, and to be able to apply this approach to cross-country navigation.

Acknowledgements

Mike Blackwell, James Frazier, and David Simon made the NAVLAB experiments possible. Chuck Thorpe provided the path planner used in this system. Keith Gremban ported and demonstrated the road following program on the Martin Marietta ALV.

References

[1] J. Beyer, C. Jacobus, and F. Pont. Autonomous Vehicle Guidance using Laser Range Imagery. In *SPIE Vol. 852, Mobile Robots II*, Cambridge, 1987.

[2] M. J. Daily, J. G. Harris, and K. Reiser. Detecting Obstacles in Range Imagery. In *Image Understanding Workshop*, Los Angeles, 1987.

[3] M.J. Daily, J.G. Harris, and K. Reiser. An Operational Perception System for Cross-Country Navigation. In *Proc. Image Understanding Workshop*, Cambridge, 1988.

[4] R. T. Dunlay and D. G. Morgenthaler. Obstacle Detection and Avoidance from Range Data. In *Proc. SPIE Mobile Robots Conference*, Cambridge, MA, 1986.

[5] K.D. Gremban, C.E. Thorpe, and T. Kanade. Geometric Camera Calibration Using Systems of Linear Equations. In *Proc. Image Understanding Workshop*, Cambridge, 1988.

[6] M. Hebert and T. Kanade. 3-D Vision for Outdoor Navigation by an Autonomous Vehicle. In *Proc. Image Understanding Workshop*, Cambridge, 1988.

[7] M. Hebert, I. Kweon, and T. Kanade. *3-D Vision Techniques for Autonomous Vehicles*. Technical Report CMU-RI-TR-88-12, The Robotics Institute, Carnegie-Mellon University, 1988.

[8] S. Shafer and W. Whittaker. *June 1987 Annual Report: Development of an Integrated Mobile Robot System at Carnegie Mellon*. Technical Report CMU-RI-TR-88-10, The Robotics Institute, Carnegie-Mellon University, 1988.

[9] S. A. Shafer, A. Stentz, and C. E. Thorpe. *An Architecture for Sensor Fusion in a Mobile Robot*. Technical Report CMU-RI-TR-86-9, Carnegie-Mellon University, the Robotics Institute, 1986.

[10] U.K. Sharma and L.S. Davis. Road Following by an Autonomous Vehicle using Range Data. In *SPIE Vol. 727, Mobile Robots II*, Cambridge, 1986.

[11] T. Stentz. *The NAVLAB System for Mobile Robot Navigation*. PhD thesis, Carnegie-Mellon University, Fall 1988.

[12] C.E. Thorpe, M. Hebert, T. Kanade, and S.A. Shafer. Vision and Navigation for the Carnegie-Mellon Navlab. *PAMI*, 10(3), 1988.

[13] M.A. Turk, D.G. Morgenthaler, K.D. Gremban, and M. Marra. VITS- A Vision System for Autonomous Land Vehicle Navigation. *PAMI*, 10(3), may 1988.

[14] R. Wallace, K. Matsuzaki, Y. Goto, J. Crisman, J. Webb, and T. Kanade. Progress in robot road-following. In *IEEE International Conference on Robotics and Automation*, 1986.

[15] R. Watts, F. Pont, and D. Zuk. *Characterization of the ERIM/ALV Sensor - Range and Reflectance*. Technical Report , Environmental Research Institute of Michigan, Ann Arbor, MI, 1987.

[16] A.M. Waxman, J.J. LeMoigne, L.S. Davis, B. Srinivasan, T.R. Kushner, E. Liang, and T. Siddalingaiah. A Visual Navigation System for Autonomous Land Vehicles. *Journal of Robotics and Automation, Vol. 3*, 1987.