

Vision-Based Urban Navigation Procedures for Verbally Instructed Robots

Theocharis Kyriacou, Guido Bugmann, Stanislaw Lauria

Robotic Intelligence Laboratory, School of Computing, University of Plymouth, Plymouth, United Kingdom,
<http://www.tech.plym.ac.uk/soc/staff/guidbugm/robolab/robolab.html>

Abstract

Humans who explain a task to a robot, or to another human, use chunks of actions that are often complex procedures for robots. An instructable robot needs to be able to map such chunks to existing pre-programmed primitives. We investigate the nature of these chunks in an urban visual navigation context and describe the implementation of one of the primitives: "take the n^{th} turn right/left". This implementation requires the use of a "short-lived" internal map updated as the robot moves along. The recognition and localisation of intersections is done using task-guided template matching. This approach takes advantage of the content of human instructions to save computation time and improve robustness.

1. Introduction.

This work is part of a project on "Instruction-Based Learning" (IBL) where robots acquire user-specific skills based on verbal instructions given by the user. One of the issues in the project is the mapping from action chunks used in natural language to actions executable by the robot. The approach used here is to provide a set of pre-programmed primitives corresponding to action chunks referred to by users. This facilitates the mapping from the semantic analysis of the spoken input to a sequence of executable robot actions.

In an earlier part of this project, subjects were invited to speak to a small robot in a miniature town (Figure 1) and to explain to it how to navigate between two landmarks. The instructions were recorded, transcribed and analysed to identify chunks of actions [1]. These chunks were grouped into about 15 "primitive" procedures that are listed and discussed in section 2.

In implementing such primitives, one can take advantage of the content of verbal instructions to minimize computational time and improve robustness, for instance by limiting visual search to those features mentioned in the instructions. The example of the navigation primitive "take the n^{th} left turn" is detailed in section 3. This primitive requires counting of landmarks and imposes the use of a "short-lived" internal map of the environment. Detection of road features, such as

intersections and turnings, is implemented by matching feature templates in the internal map.

The requirements of natural language understanding induce the internal representation of a route as a sequence of high-level task specifications (primitives). This is in principle very robust against environmental variations, provided that the primitives handle these appropriately.

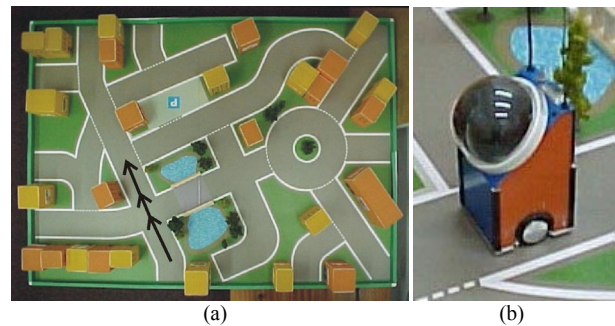


Figure 1: The miniature town (a) and robot with an 8×8 cm base (b). The marked path on the town image refers to the route followed by the robot in the example given in section 3.5 (where each arrowhead denotes a waypoint).

2. Action chunks in verbal instructions and corresponding primitives.

A corpus of route descriptions was collected from 24 subjects in the miniature town environment. Each subject was asked to give 6 route descriptions from a starting location (different for each subject) to a different destination each time. Subjects were instructed to assume the robot's point of view during their instructions. This seems to have worked since no description used a survey view.

A total of 144 route descriptions were analysed for their functional components (action chunks). These are the smallest possible units of information that compose each route instruction. Analysis was done by hand and all action chunks were then categorised into groups. The functional analysis revealed 15 functional groups (table 1). A similar approach for chunking of route descriptions was used by [2] and [3].

Table 1: Functional primitives extracted from the corpus.

	Primitive	Description
1	<code>go(description_1, landmark_1, preposition_1, description_2, landmark_2)</code>	Instructs the robot to follow a known route (with known starting point and destination).
2	<code>location_is(description_1, landmark_1, direction_1, preposition_1, description_2, landmark_2, description_3, landmark_3, ordinal_1)</code>	Specifies a location.
3	<code>destination_is(description_1, landmark_1, direction_1, , preposition_1, description_2, landmark_2, description_3, landmark_3, ordinal_1)</code>	Indicates the destination landmark.
4	<code>go_until(description_1, landmark_1, preposition_1, description_2, landmark_2)</code>	Follow known route to a landmark until a specified location in the route.
5	<code>exit_roundabout(ordinal_1, preposition_1, description_1, landmark_1)</code>	Take a specified exit from a roundabout.
6	<code>turn(ordinal_1, direction_1, preposition_1, description_1, landmark_1)</code>	Take a specified turn off a road.
7	<code>follow_road_until(preposition_1, description_1, landmark_1)</code>	Move forward until a certain location.
8	<code>rotate(direction_1, extend_1, around_1)</code>	Rotate to a certain extend.
9	<code>exit_from(description_1, landmark_1)</code>	Exit from a place, usually used for the car park.
10	<code>cross_to(description_1, landmark_1)</code>	Instructs the robot to cross the road to a landmark.
11	<code>enter_roundabout(direction_1)</code>	Enter the roundabout in a specific direction.
12	<code>park(preposition_1, description_1, landmark_1)</code>	Park on, or close, to a certain landmark.
13	<code>take_road(preposition_1, description_1, landmark_1)</code>	Take a road in view.
14	<code>goto_side(preposition_1, description_1, landmark_1)</code>	Go round a landmark to one of its sides.
15	<code>fork(direction_1)</code>	Follow a one of the two branches of a fork (Y split).

The number of functional groups found is subjective as it depends on the annotation method. Here, the annotation was done with two objectives in mind: 1. Produce parameterised primitives that generalize the description found in the corpus. For instance, the procedure designed for “turn left after the tree” should also work if “tree” is replaced by “Church”. 2. An important issue is knowledge representation. Route following is a continuous chain of actions. When, as in this case, a route is represented as a sequence of primitives, the initial state of the robot in each primitive must be consistent with the final state in the previous primitive. Therefore, all actions referred to by subjects were assumed to have an initial and a final state. Subjects however rarely specified explicitly the starting point of an action and sometimes did not define the final state in the same utterance. It was assumed that the IBL system would be able to retrieve missing information from the context. For instance, when a subject specified a non-terminated action, such as “keep going”, it was classified as “follow the road until”, assuming that a termination point would be inferred from the next specified action.

Not all functional primitives in table 1 are purely navigation tasks. For example “go” consists mainly of retrieving from memory the list of primitives

corresponding to a given route, and “location is” specifies spatial relations between landmarks. In contrast, “destination is” is found at the end of a route description to indicate the location of the goal. The robot needs then to find its way to that location.

3. Implementation example: The primitive “turn()”.

3.1. The parameter combinations for the “turn” primitive.

Four different combinations of parameters can be passed to the “turn” primitive procedure (table 2). The program implementing the primitive executes a different sequence of operations depending on the combination of parameters passed. For each of the four-parameter combinations a dedicated sub-routine is called in the primitive procedure. The next section shows the pseudo-code for the second case in the table above.

3.2. Pseudo-code for the case “take the nth turn left/right”.

In the first step of the pseudo-code in table 3, the templates selected for this case represent straight or

Table 2: Different combinations of parameters of the “turn” primitive procedure. The examples indicate some of the values that the parameters can take.

Parameter combination	Example
<code>turn(direction_1)</code>	“Take a right turn” “Turn left”
<code>turn(ordinal_1, direction_1)</code>	“Take the second left turn”
<code>turn(direction_1, preposition_1, landmark_1)</code>	“Turn left after the post-office”
<code>turn(ordinal_1, direction_1, preposition_1, description_1, landmark_1)</code>	“Take the second turning after Tesco’s” “Turn left at the library” “Turn right after the tall blue building”

curved road segments and intersections of various angles (section 3.4).

Table 3: Pseudo-code for case “take the n^{th} turn left/right”. The resulting sequence of displacements is illustrated in section 3.5 for n (ordinal_1) = 2 and $direction_1=left$.

Define set of road features (templates – see section 3.4) to look for. Loop: { <ul style="list-style-type: none"> • Capture and process road image. • Update internal map & localize robot (see section 3.3). • Find best matching template in the map. • Execute procedure (e.g. robot motion) associated with the winning template. }
--

Templates are mapped to road-like areas in the top view projection of the image captured by the camera. The template with the best match will determine the action to be performed next. For example the templates for straight or curved road will cause the robot to move further along the road. The intersection templates can have one of two actions associated with them: 1. either cause the robot to move to the centre of the intersection and rotate in the direction of turn or 2. just move ahead along the road. The first action is associated with the intersection templates when approaching the n^{th} intersection. In this case the robot takes the turn and the loop is exited so that execution is passed to the primitive associated with the next chunk in the route description. The second action is associated with the intersection templates until (but excluding) the n^{th} intersection. In these cases the robot carries on following the road. In this procedure, the robot must keep track, not only of the number of intersections passed but also of their location. When an intersection is identified, its location is compared against a record of previously found intersections and if a relatively close match is not found, it is considered to be a new intersection. Intersection locations are recorded in the egocentric reference frame of the robot. Each time the robot moves these are updated to reflect their new relation to the robot. To perform this updating, the robot must know by how much it has moved since the last image was taken. In our purely vision-based system, this is done by tracking the displacements of landmarks in the image, using a “short-lived” feature map, as described in

section 3.3

3.3. Short-lived map

A short-lived map serves several purposes during the robot’s navigation. It is used to compensate for the dead angles of the robot by recording visual information, to keep track of landmark locations (like the intersections in the example of the previous section), and for resolving spatial relationships between a landmark and the road (e.g. to define a road area “after” a building). The map is constructed progressively as the robot moves using road surface and road edge information filtered out from the top view of the scene (this is illustrated in figure 3). This view is produced by applying a perspective transform to the camera image. Road surface information is extracted from the top view image using chromaticity information. Chromaticity is an intensity-invariant two-dimensional vector describing colour. The two components of the vector are the ratios of red to blue and green to blue components of the RGB vector. A road surface likelihood image is constructed by assigning a value to each pixel location in the original image, which is proportional to the Euclidian distance between its chromaticity vector and a reference chromaticity vector. The reference vector is obtained by calculating the average chromaticity of a sample of road area in the first image along the route. To increase computation speed, a threshold is applied on the road surface likelihood image resulting in a binary image with either road-like or non road-like areas. For road edge extraction, an illumination-invariant approach similar to the one suggested in [4] is used to discriminate the white lines (road markings) along each side of the road. This is done by effectively convolving a two-dimensional low-high-low intensity mask with the original image with the high intensity span of the mask being at least equal to the width of the road markings in the top view image. Again, the resulting image is thresholded to obtain a binary image. Column B of figure 3 shows examples of road edge and road-like surface images. After the execution of each motion command, the map view is translated according to the expected motion vector of the robot, so reflecting its new expected pose in the environment. The difference between the expected and actual location of the robot is due to

motion errors. These are corrected by finding the best matching pose of the new top view in the map, then by translating the map again to reflect the actual position of the robot. The matching process uses only the road edge image rather than the road surface image because edges are robust features of the image and allow a more precise matching. To save computational time and limit the risk of matching the new view at the wrong location, not all the map is searched but only a limited area defined around the expected position of the robot in the map. To further improve speed, a crude search is performed initially using coarse steps of position and orientation. The search is then refined for a more accurate determination of the position and orientation (match vector). The resulting match vector is used to paste the road surface image of the top view on the map and to translate the map. The map is termed “short-lived” because it is only maintained for a limited area around the robot’s position (e.g. the size of images in columns C and D of figure 3). These steps are illustrated in section 3.5. The road surface likelihood map is built for the purposes of template matching which is described in the following section.

3.4. Road-feature templates

Templates are binary images of local road features drawn at the same scale as the short-lived map of road-like areas (Figure 2).

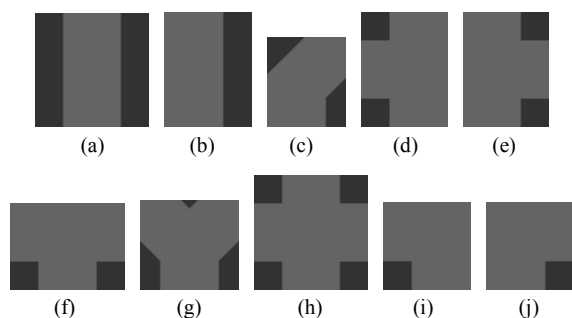


Figure 2: Examples of templates for: road following (a,b,c,i,j), for intersection detection (d,e,f,g,h). The light grey areas indicate road-like areas and the darker grey areas represent non-road areas.

For each road navigation task a subset of the available templates (corresponding to the task) is selected. For example, in the case of the action chunk: “take the second turn right” (illustrated in the following section) templates a, b and e are selected, for following the road and for detecting the intersection. The selected templates are continuously matched against the road surface map for each new image captured.

The matching process for each location and orientation of the template on the map produces a match quality measure made of the sum of two ratios: 1. the score, which is the sum of the matching road and non-road

pixels in the two images divided by the number of template pixels falling onto areas of the map where information is available from previous images and 2. the confidence factor, which is the fraction of the template area falling onto areas of the map with information. The best match of the template is the one where the sum of these two components is maximum. When the best vectors of all candidate templates are found, the “winner” template is selected as the one whose vector is associated with the best match. As with the map building, not all of the map is searched for the best match of the templates and the search is initially coarse, then refined. The position and orientation of the winning template defines the next motion command sent to the robot.

3.5. Example

Figure 3 shows the successive states of the short-lived maps and images processing results as the robot executes the instruction chunk: “take the second turning to the right”. The path followed by the robot is marked on figure 1a. Each arrowhead indicates the points where the robot finishes an action and captures a new image to determine the next action. In step 1 there is no information on the edge map and so the edge and surface top views are simply pasted (in the egocentric reference frame) on the edge and surface maps respectively. In successive steps, this initial map is progressively shifted backwards and eventually rotated. Column D of figure 3 shows the best matching template in each step. Step 5 shows the resulting map after the rotation of the robot at the second right turn.

4. Concluding comments

Two aspects of natural language instructions influence the method proposed here for navigation in our urban model environment: Their division into action chunks and their under specified nature.

Each chunk can be considered as a search-and-act loop which exits when a condition is met. Primitives were written to reflect this. Like chunks, primitives loop until a condition is met. They have an initial state and a final state and only when their final state is reached, execution passes to the next primitive. The initial state of the next primitive must be consistent with the final state of the previous primitive to ensure consistency of execution.

In natural language, task specification is very abstract. For example in: “take the second turn right”, the absolute locations of the intersections, their orientations or shapes are not given. These pieces of information must be retrieved in-situ by the robot to successfully complete the task. This is achieved here by the use of local road-feature templates that enable to recover orientation and shape information. Robustness is

achieved on one hand by defining very general template shapes and on the other hand by limiting visual search to those salient features selected by the

instructor.

To localize road features, the use of road surface information is deemed more robust than edge

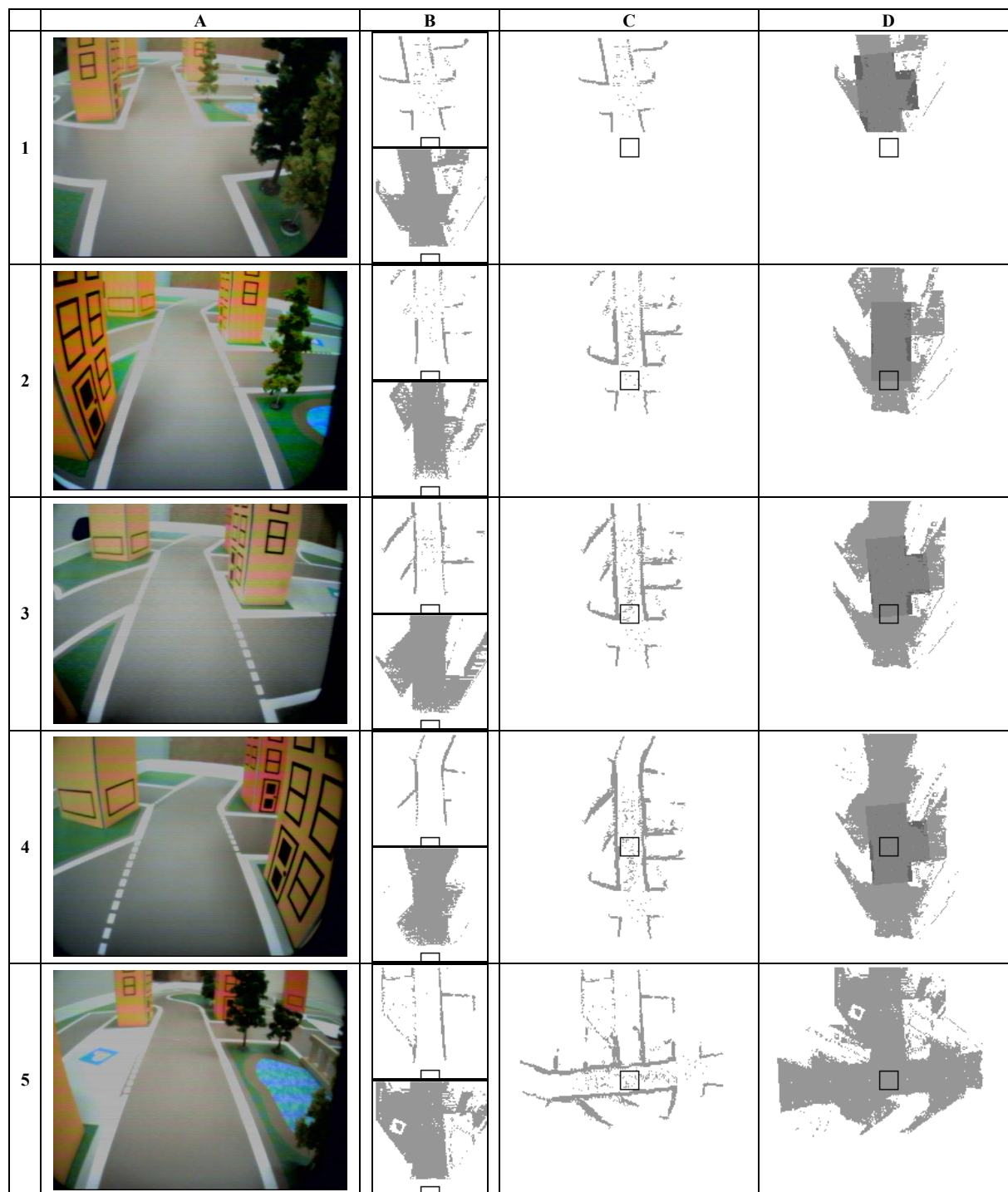


Figure 3: Step-by-step illustration of the execution of “take the second turning to the right”. The execution is completed in five steps with the corresponding images at each step shown in the rows of the figure. Column A shows the camera view, column B shows the road edge and road surface images of the top view. In column C the road edge map is displayed and in column D the road surface map is shown. The best match position and orientation of the winning template for the step is also shown superimposed on the road surface map. Note also the indication of the position of the robot (black outline) in all the top view and map images.

information. A template has a good chance to match correctly even if road areas are partially missing, e.g. due to occlusion. For instance, in figure 3, image 1D, the "right turn" template matches at the correct location although the trees prevent full recovery of the road in the filtered image.

Most of the methods suggested in the past to recover the road layout from road images deal with the case of a straight or curved road extending in front of a vehicle, but without any turns, intersections, splits, roundabouts etc. [5], [6], [7], [8], [9] and [10]. These methods require that both sides of the road are visible (though not necessarily continuous) in the image to be able to recover the road. These methods are effective in cases where a vehicle needs to stay in the middle of a road lane when following a highway for example, but they are unsuitable in more complex urban environment.

Methods to recognize intersections on the road were proposed by [11] and [12]. In [11] a previously trained neural network is used to distinguish the road. The method lacks precision because of the neural network approach used and fails to accurately determine the location and orientation of the road. Furthermore, the method suggested for modelling a road intersection required the knowledge of either the position of the intersection, to determine its precise layout, or the layout, to find its position. A priori information for an intersection is also available in our case, through the natural language, but this is not absolute as far as the intersection's location or complete as far as its layout.

In [12] dynamic model building and matching are applied on a road surface likelihood image to determine the layout of the road. This method effectively finds intersections spurring from a straight road but would fail to find an intersection on a curve or an exit from a roundabout for example. Furthermore, the suggested method attempts to reconstruct the whole intersection. A strength of our method is that only the necessary road features (for the completion of the task in hand) are sought in the map, thus saving computational time and improving on system robustness.

Other landmarks (buildings, trees, lake, bridge etc.) of our model town are mentioned in the corpus and will need to be located to enable following the instructions. Ongoing work is addressing the problem of landmark recognition, the resolution of spatial relations between landmarks, and those between landmarks and the robot as sometimes mentioned in the corpus. The solutions to these problems are not expected to modify the navigation methodology described here but will rather merge with it.

In a real urban environment the template-based method should still work. The main issue is the segmentation of the scene into navigable and non-navigable areas.

Finally, an interesting property of such a system is that it has all the perceptual components required to robustly learn a route from experience (e.g. by following a human guide) in terms of reportable action chunks

rather than in terms of odometric measurements.

Applications of this work include intelligent helper robots such as autonomous wheelchairs for indoor/outdoor applications.

Acknowledgements: This work is supported by EPSRC grants GR/M90023.

5. References

- [1] Lauria S., Bugmann G., Kyriacou T., Bos J., Klein E., **Personal Robot Training via Natural-Language Instructions**, IEEE Intelligent Systems, 16:3, 2001, pp. 38-45.
- [2] Fraczak, L., **From route descriptions to sketches: a model for a text-to-image translator**, ACL-95 Student Session, MIT, Cambridge, USA, 1995.
- [3] Denis, M., **The description of routes: A cognitive approach to the production of spatial discourse**, CPC, 16:4, 1997, pp.409-458.
- [4] Broggi, A., **Robust Real-Time Lane and Road Detection in Critical Shadow Conditions**, Proc. of the IEEE International Symposium on Computer Vision, Coral Gables, Florida, November 1995, pages 353-358.
- [5] Waxman, A. M., Lemoigne, J. J., Davis, L. S., Srinivasan, B., Kushner, T. R., Liang, E., Siddalingaiah, T., **A Visual Navigation System for Autonomous Land Vehicles**, IEEE Journal of Robotics and Automation, Volume 3, Issue 2, April 1987, pp. 124-141.
- [6] DeMenthon, D., Davis, L. S., **Reconstruction of a Road by Local Image Matches and Global 3D Optimization**, Proc. of the IEEE International Conference on Robotics and Automation, 1990, pp. 1337-1342.
- [7] Kaske, A., Wolf, D., Husson, R., **Lane Boundary Detection Using Statistical Criteria**, Proc. of International Conference on Quality by Artificial Vision, QCAV'97, Le Creusot, France, 1997, pp. 28-30.
- [8] Sayd, P., Chapuis, R., Aufrere, R., Chausse, F., **A Dynamic Vision Algorithm to Recover the 3D Shape of a Non-Structured Road**, Proc. of the 1998 IEEE International Conference on Intelligent Vehicles, 1998, pp. 80-86.
- [9] Wilson, M. B., Dickson, S., **Poppet: A Robust Boundary Detection and Tracking Algorithm**, BMVC99 (British Machine Vision Conference 1999), pp. 352-361.
- [10] Wang, Y., Shen, D., Teoh, E. K., **Lane Detection Using Spline Model**, Pattern Recognition Letters, 21(8), July 2000, pp. 677-689.
- [11] Jochem, T. M., Pomerleau, D. A., Thorpe, C. E., **Vision Based Intersection Navigation**, Proc. of the 1996 IEEE Symposium on Intelligent Vehicles, September 1996, pp. 391-396.
- [12] Crisman, J.D., Thorpe, C.E., **SCARF: A Color Vision System that Tracks Roads and Intersections**, IEEE Transactions on Robotics and Automation, Volume 1, Issue 1, February 1993, pp. 49-58.