Vision-Based Urban Navigation Procedures for Verbally Instructed Robots

Theocharis Kyriacou, Guido Bugmann[1], Stanislao Lauria

Centre for Interactive Intelligent Systems,
School of Computing, Communications and Electronics, University of Plymouth
Drake Circus, Plymouth PL4 8AA, United Kingdom

Abstract

When humans explain a task to be executed by a robot they decompose it into chunks of actions. These form a chain of search-and-act sensory-motor loops that exit when a condition is met. In this paper we investigate the nature of these chunks in an urban visual navigation context, and propose a method for implementing the corresponding robot primitives such as "take the n^{th} turn right/left". These primitives make use of a "short-lived" internal map updated as the robot moves along. The recognition and localisation of intersections is done in the map using task-guided template matching. This approach takes advantage of the content of human instructions to save computation time and improve robustness.

Keywords: Urban navigation, road layout recognition, robot primitives, route instructions, template matching.

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

[1] To whom correspondence should be addressed.
http://www.tech.plym.ac.uk/soc/staff/guidbugm/ibl/index.html

facilitates the mapping from the semantic analysis of the spoken input to a sequence of executable robot actions.

In an earlier phase 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 (Lauria et al., 2001). The categorisation of these chunks evolved throughout the project and they were eventually grouped into 13 "primitive" procedures that are listed and discussed in section 2.

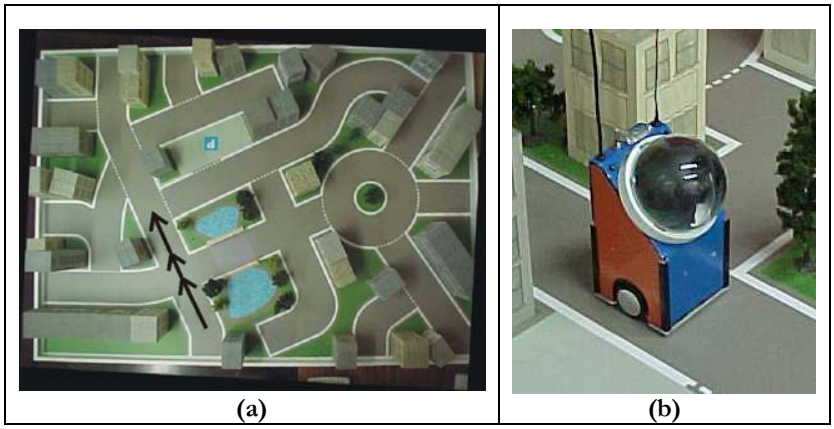

Figure 1: *(a) The miniature town. The marked path on the town image refers to the route followed by the robot in the example given in section 5 (where each arrowhead denotes a waypoint). (b) The robot with an 8x8 cm base. The robot's wireless colour video camera sends images to a PC for processing and the PC sends motion commands by radio to the robot.*

To successfully implement the primitives, the robot needs to manipulate spatial knowledge and keep track of its own position. This is achieved by using a "short-lived" internal map of the environment (section 3). Another requirement, the detection of road features such as intersections and turnings is implemented by matching local road feature templates on the internal map (section 4).

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. As an example, the execution of the navigation primitive "take the n^{th} left/right turn" is described in section 5.

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 can handle these appropriately. In this paper we focus on the detection of visual features of the road layout in a way that should be robust enough to allow traversal of complete routes. Although the environment was simplified to allow focusing on the learning components of the IBL project, there were enough sources of noise in perception (variations in illumination conditions, occlusions, etc.) and in execution (wheel slippage, shaft encoder errors, etc) to require the design of the

robust sensory-motor primitives described here. The environment was also complex and realistic enough to enable the collection of a rich corpus of unconstrained natural language route instructions. This was a key requirement for the user-centred approach developed in the project.

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. A total of 144 route descriptions were analysed for their functional components (action chunks). The functional analysis revealed 13 functional groups (table 1).

	Primitive Procedure	Description
1	**follow_road**(relation_1, ordinal_1, object_1, relation_2, object_2)	Instructs the robot to move forward on the road until a specified location.
2	**turn**(ordinal_1, relation_1, object_1, relation_2, object_2)	Instructs the robot to take a turn from the current road.
3	**location**(object_1, relation_1, ordinal_1, object_2='road', object_3, destination_1)	Specifies the location of an object. If the object is the destination the robot moves to it otherwise the robot stops as soon as it locates the object.
4	**exit_roundabout**(ordinal_1, relation_1, object_1)	Instructs the robot to take an exit off the roundabout. If the robot has not entered the roundabout then it follows the road until it meets the roundabout, enters it turning left (clockwise around the roundabout) and takes the designated exit.
5	**Go**(relation_1, object_1)	Instructs the robot to execute a previously explained route.
6	**Go_until**(object_1, relation_1, object_2)	Instructs the robot to use part of a previously explained route.
7	**enter_roundabout**(direction_1, relation_1, object_1)	Instructs the robot to enter the roundabout in a specific direction.
8	**cross**(object_1, relation_1, object_2)	Instructs the robot to cross the road to an object (usually the car park) ahead or to just cross to the opposite road at a crossroads for example.
9	**rotate**(relation_1, object_1)	Instructs the robot to rotate about itself.
10	**take_road**(relation_1, object_1)	Instructs the robot to take a road in view. Usually used when the robot is at an intersection and needs to get on an opposite road.
11	**exit_object**(object_1)	Instructs the robot to exit from a place. Usually used for exiting the car park.
12	**park**(relation_1, object_1)	Instructs the robot to park either on/by a specific location.
13	**bear**(relation_1, object_1)	Instructs the robot to take one of the two directions at a y-junction.

Table 1: *Functional primitives extracted from the corpus.*

This number 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", when used with an object other than the destination landmark, specifies spatial relations between landmarks. In contrast, when "location" refers to the destination landmark (normally at the end of a route description) the robot needs to find its way to that location.

Different combinations of parameters can be initialised for each of the primitives. Not all possible combinations may be valid though. For each of the valid parameter combinations a dedicated sub-routine is called in the primitive procedure. Table 2 shows the four different combinations of parameters that can be passed to the "turn" primitive procedure. Section 5 describes how the sub-routine for the second case in the table is implemented.

Parameter combination	Example
turn(relation_1, object_1)	"Take a right turn" "Turn left"
turn(ordinal_1, relation_1, object_1)	"Take the second left turn"
turn(relation_1, object_1, relation_2, object_2)	"Turn left after the post-office"
turn(ordinal_1, relation_1, object_1, relation_2, object_2)	"Take the second turning after Tesco's"

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

In most primitive procedures the robot needs to navigate to a visually identified target location on the road. For example, in the "turn" primitive, regardless of the combination of parameters passed, the robot eventually needs to identify a specific turn, move to it and rotate to face the new direction. Our method to discriminate components of the road layout and navigate to them is described in the following sections.

3 Short-lived maps and self-localization

A short-lived map is a map of the immediate vicinity of the robot that is updated as the robot moves in its environment. The map records previously seen visual information which go out of view as the robot moves. The robot's position is always centred on the map and facing towards the top of the map. As the robot moves the map is translated and rotated to maintain this frame of reference. In the process, elements of the map that reach its edge will disappear. Thus the term "short-lived".

The purpose of constructing such a map is to compensate for the dead angles of the robot (areas in the immediate locality of the robot which fall outside the visual field), to keep track of landmark locations and road layout features such as intersections and for resolving spatial relationships between a landmark and the road (e.g. to define a road area "after" a building).

Two versions of the short-lived map are used in this paper, the first represents the position of the road surface and the second represents the position of road edges. These maps are constructed using road surface and road edge information filtered out from the top view of the scene. This view is produced by applying a perspective transform to the camera image. This transform assumes a flat ground plane. Figures 2b and 2f show the top views of 2a and 2e respectively. This section describes how the road edge map is used to align new visual information with the ones existing in the map. The use of the road surface map for the detection of road features is described in section 4.

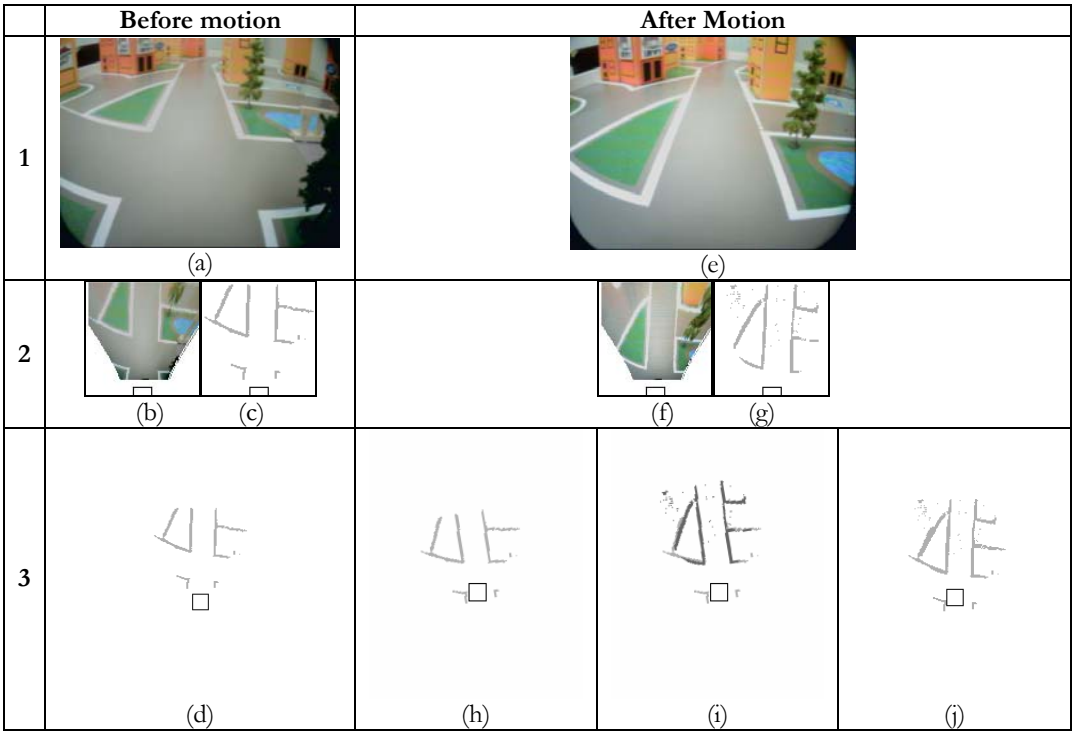

Figure 2: *Camera view (row 1), top view (row 2) and road edge map (row 3) prior to robot motion (a to d) and after robot motion, showing expected position (h), best match of new top view on map (new view shown darker) and (i) and actual position (j) after translating map.*

Road edge information is extracted from the top view image using an illumination-invariant approach similar to the one suggested in (Broggi, 1995). This approach discriminates the white lines (road markings) along each side of the road by effectively convolving a two-dimensional low-high-low intensity mask with the original image. The high intensity span of the mask is equal to the width of the road markings in the top view image. To increase computation speed in later stages, a threshold is applied on the road edge image resulting in a binary image with either road-edge or non road-edge pixels. Figures 2c and 2g are examples of thresholded road edge images of the top views in figures 2b and 2f respectively.

The top view of the scene is aligned with the map using the following steps. After the completion of each motion command, the map is translated according to the motion command sent to the robot, so reflecting its new expected pose in the environment (figure 2h). Any difference between the expected and actual location of the robot in the map results from possible motion errors. These are corrected by using a newly captured image and by finding where the road edge image of the new top view best fits on the map. The map is then translated again to reflect the actual position of the robot.

The robot's position with regards to the top view image is a point outside the field of view called here "the pivot point" (figure 3). While searching for the best match,

the top view image is displaced and rotated (by a vector $[x,y,\phi]$) so that its pivot point scans the map image.

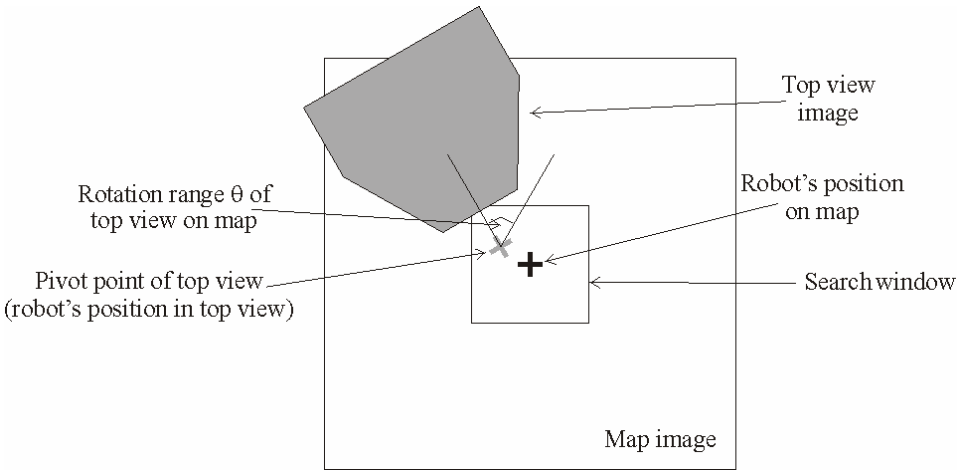

Figure 3: *Illustration of one match position of the top view image on the map.*

The match quality Q_1 describes how the edges of the two images overlap. Q_1 is made of the sum of two ratios: 1. the score, which is the number of matching road edge pixels in the intersecting area of the two images divided by the number of road edge pixels in the map image and 2. the confidence factor, which is the fraction of the top view area falling onto areas of the map containing information. The score is the standard measure of match between two overlapping images (fraction of matching pixels), while the confidence reflects the number of pixels available for the matching process. This is helpful here because a significant part of the template can fall outside of the map for a given translation and rotation. A good score in that case could be due to a fortuitous overlap of similar small-scale features.

$$Q_1(x,y,\varphi) = a\frac{\sum_{p\in N(x,y,\varphi)\cap M} AND(m_p,n_p)}{\sum_{p\in N(x,y,\varphi)\cap M} m_p} + b\frac{\sum_{p\in N(x,y,\varphi)\cap M} NOR(m'_p,n'_p)}{\sum_{p\in N} n_p} \qquad (1)$$

$$m,m',n,n' \in \{0,1\}$$

This is formally described by Equation 1, where p is a pixel location in the overlapping area of the two images. m and n are values of pixels in the road edge map image M and road edge image of the top view N respectively. Value 0 denotes no road edge, and value 1 denotes road edge. $N(x,y,\phi)$ is the road edge image of the top view translated by (x,y) and rotated by ϕ. m' and n' are the information masks of the map and road edge images where 0 denotes the presence of information (mask is off) and 1 denotes no information (mask is on). a and b are weighting constants which bias the relative importance of respectively the score and the confidence of the match. At the moment we are using a=1 and b=1. The best matching position and orientation of the road edge image of the top view is the one where Q_1 is maximum. Equation 1 ensures that, for two configurations with equal score, the one with highest confidence has the best match quality.

To save computation time and limit the risk of matching the new top view at the wrong location, the search is limited to a small window defined on the map around the expected position of the robot. The range of rotation of the top view for each match position is limited to a small angle θ. The size of the search window and angle θ reflect the precision of the motion of the robot. To further improve speed, a crude search is performed initially using coarse steps of position and rotation of the top view on the map. The search is then refined for a more accurate determination of the position and orientation (match vector). An example of matching a new top view on the map after the robot's motion is illustrated in figure 2.

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. The resulting match vector is used to paste the road surface image of the top view on the map and to translate both versions of the map.

To obtain the road surface image the colour of the road is filtered from the top view image. However, simple colour filtering is not possible because. Although the colour of the road on the miniature model town is a uniform shade of grey, it does not appear so in the camera image because of several reasons such as: (a) the automatic white balance and aperture control of robot camera, (b) casting of shadows from other objects, (c) casting of colour shadows from objects and the robot and (d) changing colour and intensity of light sources.

The apparent road colour differences were minimised by trying to maintain the illumination source constant. In order to also eliminate problems due to shadows caused by landmarks on the road an intensity-invariant measure of colour, called chromaticity, was used instead of the absolute RGB values. Chromaticity is a two-dimensional vector. 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 C_{mean}. This is a value that is initially set to [1.0, 1.0] (chromaticity of grey) but continuously changed, each time a new road filtered image is produced, to the average chromaticity value of the road pixels found in the image, thus bootstrapping it to any changes to the apparent colour of the road. Bootstrapping minimises problems due to the casting of colour shadows on the road surface and colour shifts due to changes in camera white balance.

As with the road edge image, to obtain the road surface image, a threshold is applied on the road surface likelihood image resulting in a binary image with either road-like or non road-like areas. Examples of the road surface images of the top view and map are shown in Columns B and D (respectively) of figure 6.

The road surface version of the map is used to locate local features of the road layout using templates. This is described in the following section.

4 Road-feature templates

Templates are binary images of local road surface features drawn at the same scale as the short-lived map. Some examples are shown in Figure 4.

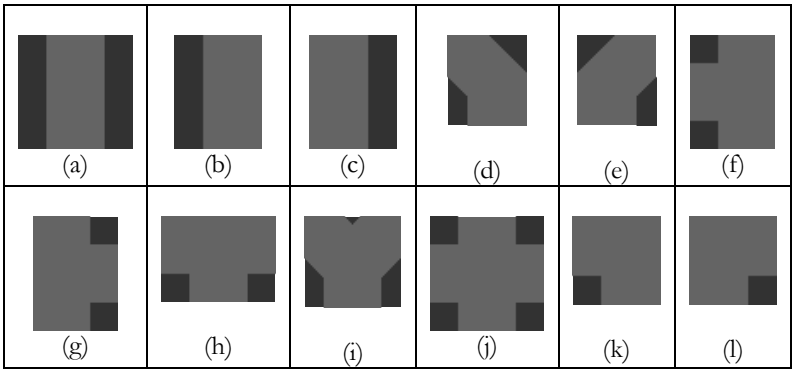

Figure 4: *Examples of templates for: road following (a,b,c, d, e), for intersection detection (f,g,h, i, j, k, l). The light grey areas indicate road-like areas and the darker grey areas represent non-road areas.*

For each road navigation task a specific subset of the available templates is selected with its associated action. For example, in the case of the task: "take the second turn right" (illustrated in the following section) templates a, c and g are selected, for following the road and for detecting the right turn. The selected templates are continuously matched against the road surface version of the map for each new scene captured and the robot navigation sequence associated with the "winner" template is executed. In this way template matching is interleaved with short motion sequences.

Like in map building, the matching process for each location and orientation (vector $[x,y,\phi]$) of the template on the map produces a match quality measure Q_2. Here the score term of Q_2 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 and the confidence factor, like the map building, is the fraction of the template area falling onto areas of the map with information.

Both, the score and confidence terms, are required to give an indication of how good a matching position is. The score gives an indication of how "well" the template matches the map and the confidence gives an indication of how much template image area (compared to the total area of the template image) was considered to obtain the score. The bigger the area considered, the more confidence is associated with the score value.

$$Q_2(x,y,\varphi) = a\frac{\sum_{p\in T(x,y,\varphi)\cap M} XOR(m_p,t_p)}{\sum_{p\in T(x,y,\varphi)\cap M} NOR(m'_p,t'_p)} + b\frac{\sum_{p\in T(x,y,\varphi)\cap M} NOR(m'_p,t'_p)}{\sum_{p\in T} t_p} \qquad (2)$$

$$m,m',t,t' \in \{0,1\}$$

This is formally described by Equation 2, where p is a pixel location in the overlapping area of the two images. m and t are values of pixels in the road surface map image M and template image T respectively. Value 0 denotes no road, and value 1 denotes road. $T(x,y,\phi)$ is the template image translated by (x,y) and rotated by ϕ. m' and t' are the information masks of the map and template images where 0 denotes the presence of information (mask is off) and 1 denotes no information (mask is on). a and b are weighting constants which bias the relative importance of respectively the score and the confidence of the match. We are using a=1 and b=1. The best matching position and orientation of the template is the one where Q_2 is maximum. Equation 2 (like equation 1) ensures that, for two configurations with equal score, the one with highest confidence is the winner.

Every template is associated with a minimum quality Q_{2min}. If the template matching quality Q_2 is less than Q_{2min} then it is assumed that the road feature associated with the template does not exist in the robot's view. The value of Q_{2min} determines how "tolerant" the template matching is to false positives caused by occlusions of the road surface by other landmarks or image noise. This value is dependent on the structure of the template image and therefore it is different for each template. Its value ranges from 0.75 to 0.85 depending on the template. This means that the template matching process is robust against 15% to 25% distortion of road surface information.

Each template is associated with a pivot point. This is denoted by a dot-centred circle in figure 5 for some of the templates.

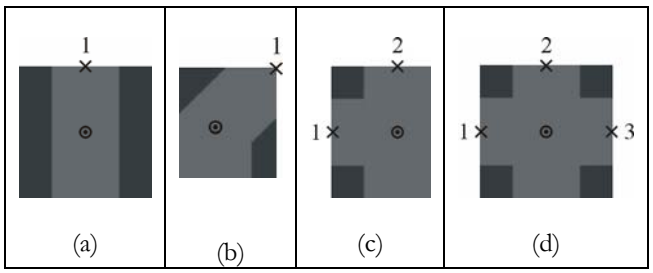

Figure 5: *Pivot point and search centres for the crossroad template.*

Translation and rotation of the template during the matching process is done with reference to this point. The pivot point of the winner template becomes a waypoint for the robot. Each template has a set of associated "search centres" (denoted by the numbered crosses in figure 5) which will define the template search window in the next image captured. The search window is defined around one of these centres (projected onto the map after the match) depending on the action associated with

the winner template. If for example the action is to turn left at the crossroad, search centre 1 of the crossroad template (figure 5d) will be used to define the search window of the next template. As with map building, the search for the best match vector is initially coarse and then refined (section 3).

5 Example: "take the nth turn left/right".

In this section the instance of the "turn" primitive is described when the ordinal and direction parameters are passed to it (second case in table 2). Table 3 shows the pseudo-code of the sub-routine of the primitive that is called in this case.

```
Define set of road features to look for.
Loop:
{
    Capture and process road image.
    Update internal map & localize robot.
    Find best matching template in the map.
    Execute procedure (e.g. robot motion) associated with the
    winning template.
}
```

Table 3: *Pseudo-code for case "take the nth turn left/right". The resulting sequence of displacements is illustrated in section 3.5 for n (ordinal_1) = 2 and direction_1=left.*

In the first step of the pseudo-code in table 3, the templates selected for this case represent straight or curved road segments and intersections of various angles. In the loop, the selected templates are matched on the road surface map. The template with the best match in each iteration of the loop 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 to cause the robot to move to the centre of the intersection and rotate in the direction of the turn or 2. just move ahead along the road (head in the centre of the intersection but without rotation at the end). The first action is associated with the intersection templates when approaching the nth intersection. In this case the robot makes the turn and the loop is exited so that execution is passed to the primitive associated with the next chunk in the route instruction. The second action is associated with the intersection templates until (but excluding) the nth 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 locations. 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 position relative 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 the short-lived feature map described in section 3.

Figure 6 shows the successive states of the short-lived maps and images processing results as the robot executes the road navigation task: "take the second turning to the right".

The path followed by the robot is marked in figure 1a, where 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 road edge map and so the road edge and road surface top views are simply pasted (in the egocentric reference frame) on the road edge and road surface maps respectively. In successive steps, this initial map is progressively shifted backwards and eventually rotated. Column D of figure 6 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.

Row 1 of figure 6 demonstrates the robustness of the template matching method against occlusions of the road surface. In image 1A of figure 6 the trees on the right of the robot occlude part of the right turning. Nevertheless, the right turning template is successfully matched in the correct place producing the correct location and orientation of the sought landmark (see image 1D of figure 6).

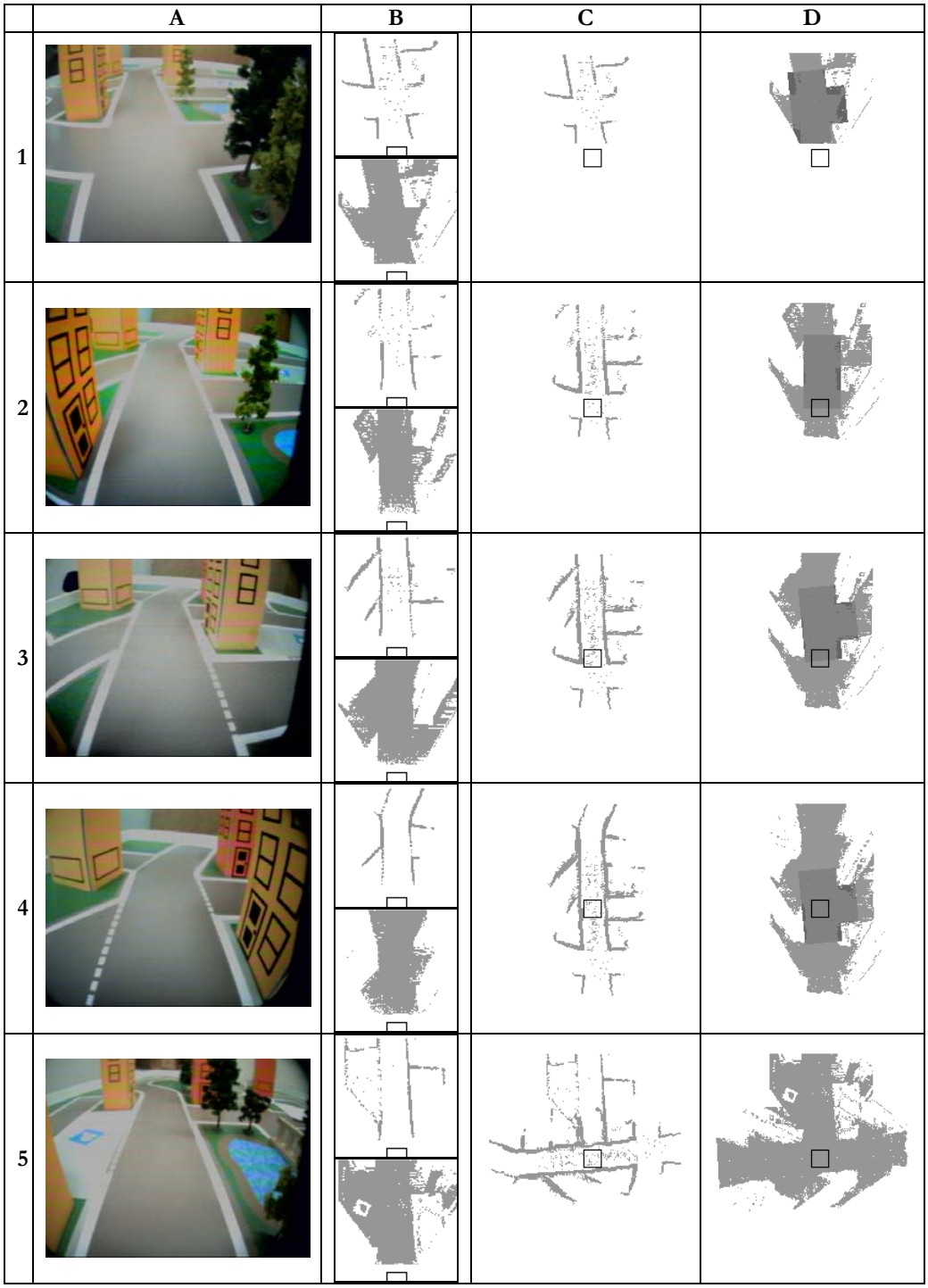

Figure 6: *Step-by-step illustration of the execution of "take the second turning to the right". The execution is completed in five steps with the corresponding images at each step shown in the rows of the figure. Column A shows the camera view, column B shows the road edge and road surface images of 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.*

6 Evaluation of the primitive procedures

The primitive procedures were tested independently of the other components of the IBL system that includes speech recognition, semantic analysis, etc. (Lauria et al., 2001). For that purpose, after the collection of the corpus and the initial specification of the primitive procedures, each route description was manually translated into a list of primitive procedure calls. An example of one such translation is given in Table 4.

(a)	"from the roundabout take the first exit on the left continue straight over the crossroads continue over the bridge continue straight over the second crossroads the post office should be on your right" (u7_GC_CX)
(b)	**exit_roundabout**(ordinal_1='first') **follow_road**(relation_1='over', object_1='crossroads') **follow_road**(relation_1='over', object_1='bridge') **follow_road**(relation_1='over', ordinal_1='second', object_1='crossroads') **location**(object_1='post_office', relation_1='right_of', object_2='self', destination_1='post_office')

Table 4: *An example of a translation of a route description to its corresponding primitive calls. Row (a) shows the transcribed version of the route description u7_GC_CX[2]. User 7 explains the route from Boots to the Post-office. Row (b) shows the corresponding manual translation of the description to its primitive procedure calls.*

The translated route descriptions were then split at random into to equal sets. One was used for the development of the primitive procedures (development set) and the other for their evaluation (evaluation set). During development, the robot was placed at the start of each route in the development set, facing roughly in the right direction, and was given the full translation of the route to execute. Changes were made to the specification and program code of the primitive procedures until as many as possible primitive procedure calls in the development set could be executed successfully. In several cases, the robot could not execute the full sequence of primitives in a route because of errors or ambiguities introduced by subjects in their instructions (see (Kyriacou, 2003)). These missed primitives were not counted as error. In three other cases however, the narrow angle of view of the vision system prevented the detection of a landmark, e.g. placed on the inside of a tight curve. This was counted as an error (see table 5). Note that visual recognition of non-road landmarks such as (buildings, trees, lake, bridge etc.) of our model town was not implemented fully. Instead, coloured markers were placed on the roadside at the foot of such landmarks and were recognized visually.

[2] This code refers to an entry in the corpus available from:
http://www.tech.plym.ac.uk/soc/staff/guidbugm/ibl/

	Development set	Evaluation set
Total primitive calls	336	344
Executed primitive calls	227	218
Successful	224 (98.7%)	214 (98.2%)
Unsuccessful	3 (1.3%)	4 (1.8%)

Table 5: *Primitive call success results during the evaluation phase of the primitive procedures. Note that the percentage values indicate the proportions of the executed primitive calls and not the total primitive calls.*

For the testing phase each translated route description in the evaluation set was given to the robot for execution. During this phase no changes were made to the specifications of the primitive procedures. Success results were recorded as above per primitive procedure call. Table 5 shows that only 218 primitives out of the 344 contained in the evaluation set were actually called. This reflects essentially instruction errors. Two primitive procedure failures were caused by the occasional inability to detect a marker due to the finite field of view of the camera, e.g. in a curved section of the road. This is a minor problem. Two of the four unsuccessful primitive calls were due to two primitive procedures only found in the evaluation set. This was expected since previous study of the collected corpus indicated that the functional vocabulary of the corpus was not closed, i.e. new primitive procedures were likely to appear in instructions after the completion of the system (see (Bugmann et. al., 2001)).

Nevertheless, the high success rate of primitive procedures indicates that the proposed implementation of navigation primitives is robust, supporting the idea that robots can be programmed using high-level natural language instructions.

7 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 terminating condition is met. Primitives were written to reflect this. Primitives are like Lego bricks that the user can in principle combine in any sequence through his verbal instructions. It is indeed necessary to verify that the terminating condition of one primitive is an initial condition that the next primitive in the sequence can handle (Lauria et al., 2002).

In natural language, task specifications are 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 the 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 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 6 (column D, row 1) 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. (Waxman et al., 1987), (DeMenthon and Davis, 1990), (Kaske et al., 1997), (Sayd et al., 1998), (Wilson et al., 1999) and (Wang et al., 2000). 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 (Jochem et al., 1996) and (Crisman and Thorpe, 1993). In (Jochem et al., 1996) a previously trained neural network is used to distinguish the road. The method lacks precision because of the neural network approach used and therefore 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. Some a-priori information on the intersection is also available in our case, through the natural language instruction.

In (Crisman and Thorpe, 1993) 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.

Although the template matching method is applied here in a simplified model of the world, the same navigation principle could be applied for instructable vehicles navigating in the real world. The simplicity of the robot's environment in this project helps to set aside the problems of extracting road/non-road information from complex real-world scenes and focuses on determining the road layout once this information is found. Other information, absent in our model but present in the real world could give clues which could contribute towards a statistical measure suggesting the road layout ahead of the instructed vehicle. Such clues could be for example the direction of motion of other road vehicles, the alignment of buildings, road signs, etc.

Finally, an interesting property of the proposed approach to primitive design 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 or sensory scene snapshots.

Acknowledgements

This work is supported by EPSRC grant GR/M90023. We a grateful for the comments of the reviewers.

References

Broggi, A. (1995) Robust Real-Time Lane and Road Detection in Critical Shadow Conditions, Proceedings of the IEEE International Symposium on Computer Vision, Coral Gables, Florida, November 1995, pages 353-358.

Bugmann, G., Lauria, S., Kyriacou, T., Klein, E., Bos, J., Coventry, K., Using Verbal Instructions for Route Learning: Instruction Analysis, Proceedings of TIMR 2001 (Towards Intelligent Mobile Robots), Manchester, 2001. Technical Report Series, Department of Computer Science, Manchester University, ISSN 1361-6161. Report number UMC-01-4-1.

Crisman, J.D., Thorpe, C.E. (1993) SCARF: A Color Vision System that Tracks Roads and Intersections, IEEE Transactions on Robotics and Automation, Volume 1, Issue 1, pp. 49-58.

DeMenthon, D., Davis, L. S. (1990) Reconstruction of a Road by Local Image Matches and Global 3D Optimization, Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1337-1342.

Jochem, T. M., Pomerleau, D. A., Thorpe, C. E. (1996) Vision Based Intersection Navigation, Proceedings of the 1996 IEEE Symposium on Intelligent Vehicles, September 1996, pp. 391-396.

Kaske, A., Wolf, D., Husson, R. (1997) Lane Boundary Detection Using Statistical Criteria, Proceedings of International Conference on Quality by Artificial Vision, QCAV'97, Le Creusot, France, 1997, pp. 28-30.

Kyriacou, T., (2003) Vision-Based Urban Navigation Procedures for Verbally Instructed Robots, PhD Thesis, University of Plymouth, December 2003.

Lauria S., Bugmann G., Kyriacou T., Bos J., Klein E. (2001) Personal Robot Training via Natural-Language Instructions, IEEE Intelligent Systems, 16:3, pp. 38-45.

Lauria S., Bugmann G., Kyriacou T., Klein E. (2002) Mobile Robot Programming Using Natural Language. *Robotics and Autonomous Systems, 38 (3-4): 171-181*

Sayd, P., Chapuis, R., Aufrere, R., Chausse, F. (1998) A Dynamic Vision Algorithm to Recover the 3D Shape of a Non-Structured Road, Proceedings of the 1998 IEEE International Conference on Intelligent Vehicles, 1998, pp. 80-86.

Wang, Y., Shen, D., Teoh, E. K. (2000) Lane Detection Using Spline Model, Pattern Recognition Letters, 21(8), pp. 677-689.

Wilson, M. B., Dickson, S., Poppet (1999) A Robust Boundary Detection and Tracking Algorithm, BMVC99 (British Machine Vision Conference 1999), pp. 352-361.

Waxman, A. M., Lemoigne, J. J., Davis, L. S., Srinivasan, B., Kushner, T. R., Liang, E., Siddalingaiah, T. (1987) A Visual Navigation System for Autonomous Land Vehicles, IEEE Journal of Robotics and Automation, Volume 3, Issue 2, pp. 124-141.

	Theocharis Kyriacou is currently a research associate at the university of Essex (UK) after successfully defending his Ph.D. thesis in "Vision-Based Urban Navigation Procedures in Verbally Instructed Robots" at the university of Plymouth (Plymouth, UK). He earned his B.Eng. (Honours) degree in Electronic Engineering Systems from the University of Sheffield in 2000. Prior to studying for his undergraduate degree he obtained a Higher Engineering Diploma in Electrical Engineering from the Higher Technical Institute (H.T.I.) of Cyprus.
	Guido Bugmann is a reader in the University of Plymouth's School of Computing, Communications and Electronics where he develops vision-based navigation systems for robots, human-robot dialogue systems and investigates biological planning and spatial memory. He previously worked at the Swiss Federal Institute of Technology in Lausanne, NEC's Fundamental Research Laboratories in Japan and King's College London. He has three patents and more than 100 publications. Bugmann studied physics at the University of Geneva and received his PhD in physics at the Swiss Federal Institute of Technology in Lausanne. He is a member of the Swiss Physical Society, the Neuroscience Society, the British Machine Vision Association and AISB. Contact him at the Centre for Interactive Intelligent Systems, SoCCE – PSQ – A309, University of Plymouth, Drake Circus, Plymouth PL4 8AA, UK; gbugmann@plymouth.ac.uk.
	Stanislao Lauria is currently a lecturer at Brunel University (UK). He has been working on the IBL project as a research fellow at the University of Plymouth (Plymouth, UK) between 2000 and 2003. He received a Laurea in Physics from the Universita' di Napoli and a PhD degree in Cybernetics from the University of Reading. He has been research fellow at the University of Reading. His research interests are in the area of Neural Networks, Artificial Intelligence and robot vehicles. He can be contacted at Stasha.Lauria@brunel.ac.uk