

Robotic Navigation Using Landmarks

Philip Levis

November 14, 2000

Abstract

In this paper we present a view-based model for robotic learning of navigational paths. The learning algorithm is simple and computationally unintensive, and robust to small perturbations in the environment. The robot is trained along a path, during which it records important choice points. When it then retraces the path, it matches what it sees to recorded choice points in order to decide what action to take. Examples from experimental results are presented.

1 Introduction

The goal of this project was to teach a robot how to follow a path that it has seen before. Many methods of robotic navigation depend on constructing detailed maps of the environment. We sought to use as minimal representations as possible: the robot should only learn the features of the environment that are important to navigation.

There has been a significant amount of research in cognitive science studying exactly how people learn to navigate novel environments, and how different factors affect their ability to do so. Issues that have been addressed include physically interacting with the environment, navigation by landmarks, navigation by an internal spatial ‘map’ of the environment, and instruction.[13][18]

Taking this research into account, we decided to address the problem of navigation using the idea of landmarks (‘choice points’) as the important features of an environment for navigation. Instead of constructing a detailed map of the environment, the robot learns landmarks and what it should do when it reaches them. For example, instead of learning a detailed topographic map of all the hallways on an office floor, the robot learns that to get from the bathroom to the stairs, it should go straight past a turn to the right, turn left when the hallway ends, and then take the first left turn.[16]

The activity of the robot can be divided into two separate tasks: **learning**, and **navigation**. A robot **learns** a path by having a person lead it along the path: the person controls the robot’s movements. As the robot moves through the environment, it records important choice points for later use. A robot **navigates** the path by using its stored knowledge of choice points to decide what to do.

The robot’s learning can be summarized into two tasks:

- determining important choice points, and
- determining what to do at each choice point.

Similarly, the robot's navigation can be summarized into two tasks:

- determining important choice points, and
- matching a training choice point to a previously learned choice point.

The goal, then, is to provide a simple, general method by which these things can be done. The rest of this paper details such a method, and supplies experimental results on its use.

2 The Basis of the Model

In order to have a general method for choice point navigation, it must be independent of the forms of sensory input the robot has available, as well as the nature of the environment being navigated. Additionally, as it is view-based, (a 'view' is not necessarily an image, but rather what the robot can sense at the moment) we chose to use no odometric information in learning or navigation.

In our model, the central abstraction made is that of a **navigational behavior**. With the exception of choice point matching during navigation, behaviors are the component of the model indirectly responsible for all of the decision making.

A navigational behavior is simply a mapping of sensor input to actions with some additional capabilities, such as whether it is appropriate to use a behavior given certain sensory inputs. These additional capabilities are used to determine when choice points should be recorded. When a choice point is recorded, it is matched to the behavior representing the best approximation of what the robot is doing at that choice point. This means that if the robot is led down a path where it does things which none of its behaviors are capable of (spinning around three times clockwise), it will not be able to learn to replicate it. However, if the path can be well represented in terms of the robot's behaviors, then it will be much more successful in retracing the path.

The relationship between choice points and behaviors can be made clearer with an example of someone giving directions. Generally, directions can be comprised of a set of landmarks, and what to do at those landmarks. For example:

- 1) Walk to the door of the AI Lab and open it.
- 2) Walk straight down the hallway until it ends in a T-intersection. Turn right.
- 3) Walk down the hall until it turns left, turn left.
- 4) Walk down the hall and take the first left.
- 5) Open the first door on your left, marked 'Stairs'.

The choice point is the landmark; the behavior is what you do at that landmark. Note, however, that there can exist many more choice points than merely those that signify a change in the behavior being used. For example, the second instruction in the above directions could easily be expanded to:

- 1) Walk straight down the hallway until there is a door on your right.
- 2) Continue to walk straight down the hallway until there is a light directly overhead.
- 3) Continue to walk straight until the hallway ends in a T-intersection. Turn right.

There are actually two different kinds of choice point: a choice point that represents a change in the behavior being used (from walking straight down the hall to turning right), and a choice point that represents a *possible* change in the behavior being used (pass the left turn and keep walking straight). Although navigation treats these types of choice point identically, they are determined differently when learning.

What a robot can do is constrained by the behaviors available to it, and so are the choice points. If a robot has only one behavior, the idea of a choice point is meaningless; it can only do one thing. In contrast, if a robot has a large selection of possible behaviors, there may be many opportunities to behave differently: for example, at every intersection.

The following three sections describe the important parts of the model in depth: navigational behaviors, learning, and navigation. First an overview of the component is described, then detailed in depth. Each section is concluded with an experimental example.

3 Navigational Behaviors

3.1 Overview

The robot's navigational behaviors are a critical part of the learning model. These behaviors can be simple, or complex. A behavior has two methods of operation: passive and active. Passive operation is used during learning. Active operation is used when the robot is navigating. Both methods of operation detect choice points, but in slightly different ways. In passive mode, when a choice point is detected, it is stored. In active mode, when a choice point is detected, it is compared to stored choice points (learned landmarks) in order to determine what to do.

3.2 Layout

A navigational behavior is capable of three things:

- navigation,
- determining whether navigation with this behavior is possible, given sensor input, and
- determining how close the actions a robot is performing are to what the behavior would do, given sensor input.

Navigation and possibility are used in active navigation. Possibility and closeness are used in passive navigation (learning).

3.3 Is Navigation Appropriate?

We have established a simple method for a robot to navigate through an environment using only optical flow information. An important consideration for the robot is whether, given an optical flow field, a given behavior is appropriate.

How this is determined is individual to each behavior. Let us take the previous example of a 'follow the right wall' behavior. Under what circumstances it is possible to use this behavior? One reasonable requirement is that there be a right wall to follow. If there is no right wall, there will be less flow in the right hemifield than in the left hemifield. There will still be some from the floor, but very little. Therefore, a possible characterization for whether the 'follow the right wall' behavior is possible is whether the horizontal component of the flow vectors in the right hemifield is above a given threshold.

3.4 So Close, But Yet So Far

The last component of a navigational behavior is its capability to gauge, given input, how closely what the robot is doing resembles what the behavior would do. For normalization's sake, this is represented as a real number between 0 and 1; 0 means that the actions do not resemble the behavior at all, while a 1 means that the actions match the behavior exactly. Each behavior determines this value individually.

3.5 Example

The robot had no sensor input except for a view of its environment. Therefore, navigation was performed on these images and the information that can be extracted from them. Optical flow was calculated using a fast, patch-matching algorithm [1]. Using only optical flow, it is possible to construct a very simple and minimalist navigational behavior that is both robust and effective.

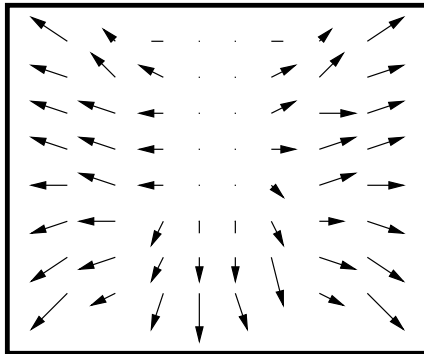


Figure 1: An example optical flow field

The navigational behaviors operated by trying to achieve some balance in the optical flow vector field. Navigating down the center of a hallway can be achieved by moving to balance the optical flow vectors to the right and left so that they are equal. By changing the balance, the robot can navigate closer to the right or left wall. This form of navigation, called the *Balance Strategy*, segments the optical flow field into left and right hemifields.[6] The robot should turn an amount

$$\Delta\Theta = k \left(\frac{\sum \|\vec{w}_L\| - \sum \|\vec{w}_R\|}{\sum \|\vec{w}_L\| + \sum \|\vec{w}_R\|} \right),$$

where k is a constant, and $\sum \|\vec{w}\|$ is the sum of the magnitudes of optical flow in a given hemifield. This form of the strategy balances the right and left hemifields equally. A slight modification allows for unequal balance (a ratio), leading to the robot staying closer to one side than another:

$$\Delta\Theta = k \left(\frac{\sum \|\vec{w}_L\| - \sum \|\vec{w}_R\|}{\sum \|\vec{w}_L\| + \sum \|\vec{w}_R\|} + r \right),$$

where r is a constant that determines the desired ratio.

3.5.1 An Optimization

This form of the algorithm works well in enclosed environments (such as hallways), but can be disrupted by some artifacts of navigating in a relatively open environment. As the robot approaches a tall, narrow object in an open environment, there is a significant amount of optical flow upwards and downwards as the object expands in the field of view. There is also flow to the left and right, but it can be very small in comparison. The left-right imbalance of the flow is overwhelmed in respect to the sum of the magnitudes of the flow vectors. If the robot approaches an obstacle almost head-on, not enough left-right imbalance is measured to avoid the object.

It is important to note that even if the robot approaches an object directly head-on, the left and right hemifields are almost never equal. Slight perturbations in the view field and the flow algorithm itself induce slight differences, that the robot adjusts for, so that it no longer approaches the object directly head-on. If following an equal-balance strategy, a robot that approaches a wall will begin to turn to one side or the other. After it turns, there will be a greater difference in flow (the wall will occupy one hemifield more than the other), that will eventually cause the robot to follow alongside the wall.

In order to deal with the issue of slight left-right imbalance being overwhelmed by the rest of the flow, we used a minor variation of the balance strategy, that uses the horizontal component, instead of the magnitude, of the flow vectors. We then obtain the final equation,

$$\Delta\theta = k \left(\frac{\sum \vec{h}_L - \sum \vec{h}_R}{\sum \vec{h}_L + \sum \vec{h}_R} + r \right), \quad (1)$$

where h_L and h_R are the horizontal components of the flow vectors in the respective hemifields.

3.5.2 Dealing With Special Cases

These methods of navigation work well when there is accurate flow computation. Problems can occur, however, when there is very little optical flow due to a very homogenous scene with little texture in it. Similarly, although usually the navigational strategy will lead to the avoidance of an obstacle, sometimes it does not, because the difference in the flow vectors in each hemifield is very slight. Using only the horizontal components of the vectors greatly diminishes, but does not remove, this problem.

Two problematic kinds of flow field can occur: in the first, there is a tremendous amount of flow, indicative of being extremely close to an obstacle; in the second, there is so little flow that no valid conclusions can be made, probably caused by a lack of features to be used in flow detection. These must be dealt with in order to prevent collisions with objects. These two cases can be handled by incorporating a pair of threshold values. If the sum of the magnitude of the flow vectors is too large, then the robot turns sharply, stops, and starts navigating again. This is to avoid a possible obstruction. If the sum of the magnitude of the flow vectors is too small, then the exact same series of actions can be performed (turn sharply, stop, start moving) in order to try to find a direction that can provide enough optical flow.[6] In practice, the case of too much flow is far more common than the case of too little.

Additionally, solely using flow for navigation is not without complications. Specifically, the flow vectors calculated while the robot is turning are very different than when not turning, and factoring out the rotation can be very difficult. Therefore, while the robot turns, it is generally unable to use its major form of information for navigation. For this reason, it is preferable that the robot turn very quickly, to minimize this sensor-deprived time. While the robot turns, it ignores optical flow and issues no new motor commands.

3.5.3 The Final Navigation Algorithm

Incorporating these cases into the navigation algorithm, we obtain:

```

BALANCE-NAVIGATE
1  if (NOT TURNING) then
2    S ← SUM-FLOW-MAGNITUDES()
3    if (S < MIN-THRESHOLD) then
4      STOP-AND-TURN()
5    else if (S > MAX-THRESHOLD) then
6      STOP-AND-TURN()
7    else
8      NAVIGATE-BY-BALANCING-FLOW()

```

3.5.4 Possibility

Determining whether a behavior is possible or appropriate can be accomplished by specifying the minimum sensory requirements for the behavior to execute in a reasonable fashion. Take a ‘go down the center of a hallway’ behavior, for example. In order to achieve its goal, there must be two walls, one on each side, that can be used to calculate flow. To generalize, there must be flow in both hemifields of view, over a certain threshold that could be merely noise. Additionally, there must be some balance in the flow vectors; if all of them point to the right, then there is no assurance that it will be possible to balance the flow. As when navigating, flow measurements while the robot is turning are ignored.

Using these two requirements, we can construct the CENTER-POSSIBLE algorithm:

```

CENTER-POSSIBLE
1  if (NOT TURNING) then
2    S ← SUM-FLOW-MAGNITUDES()
3    B ← LEFT-RIGHT-BALANCE()
4    if (S < MIN-THRESHOLD) then
5      return FALSE
6    ▷ All of the flow is in one direction
7    else if B = -1 or B = 1 then
8      return FALSE
9    else
10   return TRUE

```

where LEFT-RIGHT-BALANCE is the value of equation 1 with an r value of 0.

Whether the possibility function is effective can be easily determined empirically: place the robot in a variety of environments, and program it to navigate with a given behavior if and only if it is possible.

3.5.5 Closeness

In order to determine the closeness factor of a behavior, there must be some measure of what the robot is doing. Without odometric information, this can be very difficult: in order to obtain very accurate information about the possible motor commands being issued, some history of the sensor inputs must be maintained.

Experimentally, closeness was determined by comparing the perceived flow field with the desired flow field. For example, the optimal flow field for the previously mentioned ‘center’ behavior would be a perfectly balanced flow field, where the horizontal components of the flow vectors in the two hemifields sum to zero. The closeness was calculated as $1 - |b|$, where b is the LEFT-RIGHT-BALANCE.

4 Training

4.1 Overview

The robot is provided with a set of possible navigational behaviors, such as ‘stay to the right (taking right turns)’, or ‘go straight’. The robot is trained by leading it along the path to be learned. As it traverses the path, it determines important choice points and stores them as snapshots of the robot’s state. These choice points are determined using information from the different behaviors the robot has available. A simple method is used to smooth out the small stutters in robot input that can lead to excessively large choice point sets.

4.2 Interface

The robot is trained by a person controlling the robot to make it navigate the desired path through the environment. We controlled the robot in the virtual environment using a simple keyboard interface. With a real robot, it could be done a joystick or other manual control device. As the robot moves along the path prescribed, it computes optical flow vectors, and at appropriate points, takes snapshots of the environment to represent a choice point.

4.3 Snapshots

A snapshot is taken when a choice point in the path is reached. The snapshot contains all the information the robot has at its disposal when the choice point is reached. It must contain some form of information that will allow for reasonable matching with other choice points: sonar data, visual data, IR data, etc. Not all of the information stored at a choice point will necessarily be used for matching. It is all included in order to separate the learning and navigation processes; what is stored when learning is independent of how the robot will use that information.

4.4 Determining Choice Points

A choice point is a navigationally significant point on the path. It can be navigationally significant because it is noticeable (there is a large blue bug to the right), because it represents a point of new navigation possibilities (a hallway branching off to the right), or a change in method of navigation (after following the hallway straight for a while, turning right). Exactly what constitutes a choice point is unimportant; what is important is that the choice points contain the information necessary for the robot to reconstruct and renavigate the path shown to it.

Given the capabilities of navigational behaviors in this model, there are two important events that can occur during training, with respect to navigational behaviors:

- the set of possible behaviors changes, or
- the behavior that matches most closely to what the robot is doing changes.

The former can be thought of as an *opportunity* for a new method of navigation, or the loss of an opportunity. For example, approaching a hall branching off to the right, the set of possible behaviors may change, as suddenly the ‘turn right’ behavior becomes possible, and when the robot passes the turn, the set of possible behaviors will change again, no longer containing the ‘turn right’ behavior.

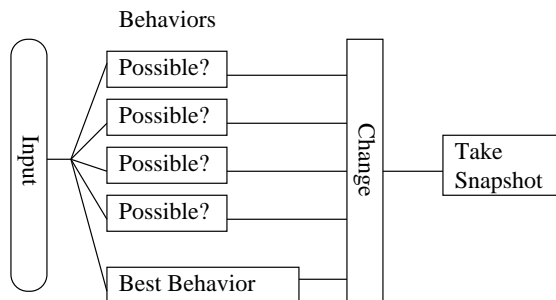


Figure 2: Taking a Snapshot

The other event that marks a choice point is if the robot believes that the behavior being used has changed. This occurs when the closest behavior to what the robot is doing changes. Using the above example, a choice point would be chosen when the ability to turn right is noticed, and if the robot makes the right turn, another choice point would be chosen to mark that the behavior changed from ‘go down the middle’ to ‘turn right’.

These two cases can be characterized as the point at which it is possible to use a new strategy, and the point at which a new strategy is used. The set of possible behaviors can merely be stored as a bit field, while the best matching behavior is the index in the bit field that represents the behavior.

4.5 Problems

This method for determining choice points has several limitations. First, the number of snapshots taken is very dependent upon the behaviors used. A large number of behaviors will lead to many more snapshots being taken. Similarly, if a behavior has a possibility condition that fluctuates a great deal, then a large number of snapshots will be taken. This means that not only should the behaviors be simple, but a robot’s behaviors should be well suited to the environment it is navigating. If a robot that navigates rectangular hallways is provided with three hundred different behaviors, each with fickle possibility functions, then the robot will not be very effective at determining the important points of the path. The representation of navigation provided to the robot must be well suited to the environment in which it is navigating. Similarly, if a robot is provided with wall-following behaviors and then trained in an open field, it will probably not perform very well.

Even with a minimalist set of behaviors, the system described is prone to ‘stuttering’, the rapid flip-flop of conditions that occurs just as the robot reaches a choice point. For example, if the conditions are almost exactly the minimum that is necessary for a behavior to be possible, then small perturbations will cause the behavior to flip back and forth from possible to not possible. This causes multiple snapshots to be taken for a single choice point.

To smooth out these stutters, a brief history is kept of the last ten sets of possible behaviors. A snapshot is taken if and only if the set of possible behaviors is different than all of the previous ten. Therefore, if there is a transition from one set of possible behaviors to another, a snapshot will be taken, but if immediately following there is another transition back to the original set, another snapshot will **not** be taken.

CHECK-POSSIBLE

```

1  S ← SET-OF-POSSIBLE()
2  R ← TRUE
3  for i ← 1 to 10 do
4      if S = P[i] then
5          R ← FALSE
  
```

```
6 P[C] ← S
7 C ← (C + 1) MOD 10
8 return R
```

5 Navigation

5.1 Overview

Given a set of choice points from a training session, the robot must be able to retrace the path taken. The robot determines when it is at a possible choice point, and matches it to the set of choice points it obtained during training. It then begins to execute the behavior that was being used when the choice point snapshot was taken. Matching is performed by comparison of the current sensor input and the stored sensor input of a choice point. Each choice point is considered equally: no information about the ordering of the choice points on the stored path is used.

5.2 Choice Points Revisited

The robot navigates by detecting choice points, and then matching the current choice point to a learned choice point. In essence, the robot detects a landmark, remembers what actions were performed at that landmark, and acts accordingly. This works best if the landmarks detected are identical or similar to those learned. Therefore, the method the robot uses to detect choice points while navigating should be as similar as possible to the method used while training.

While actively navigating, the closeness metric cannot be used, as the closest behavior will (hopefully) always be the one that is actively executing. However, the possibility values can be used. A **detected choice point** is defined as when the set of possible behavior changes. When a detected choice point is reached, the robot matches the current sensor input to the learned choice points. No weightings are applied to the learned choice points: each one is considered equally. This means that the order in that the learned choice points were reached when training is not considered during navigation.

The method of choosing a behavior is summarized in the following algorithm:

```
NAVIGATE
1 if CHECK-POSSIBLE() then
2   C ← BEHAVIOR(MATCH-CPOINT())
```

5.3 Choosing a Training Point

Once a detected choice point has been reached, the robot must match it to a learned choice point: this is performed in NAVIGATE by the MATCH-CPOINT algorithm. How choice point matching is performed is entirely implementation dependent: matching sonar images will require very different computation than visual pictures.

If there are many learned choice points that are nearly indistinguishable but have different behaviors associated with them, then the robot will probably not navigate very effectively. Although a problem, it's an understandable one: it's not hard to get lost when every turn looks the same.

5.4 Example

5.4.1 Training Point Matching

In our implementation, choice point matching was performed by comparing the image of the learned choice point with that of the detected choice point. Greyscale histograms were computed for the pair of images, and distances were computed using the Kullback-Leibler distance metric, that is defined as:

$$D(P|Q) = \sum_x P(x)[\log(P(x)) - \log(Q(x))] \quad (2)$$

$$KL = D(P|Q) + D(Q|P) \quad (3)$$

Simple euclidean and cosine distance were tried, and found to work terribly. The KL metric was used because of the nature of the environment that the robot was navigating in: the different areas of the virtual environment were distinct by the textures on the walls, floor and ceiling. The brightness of the textures varied widely.

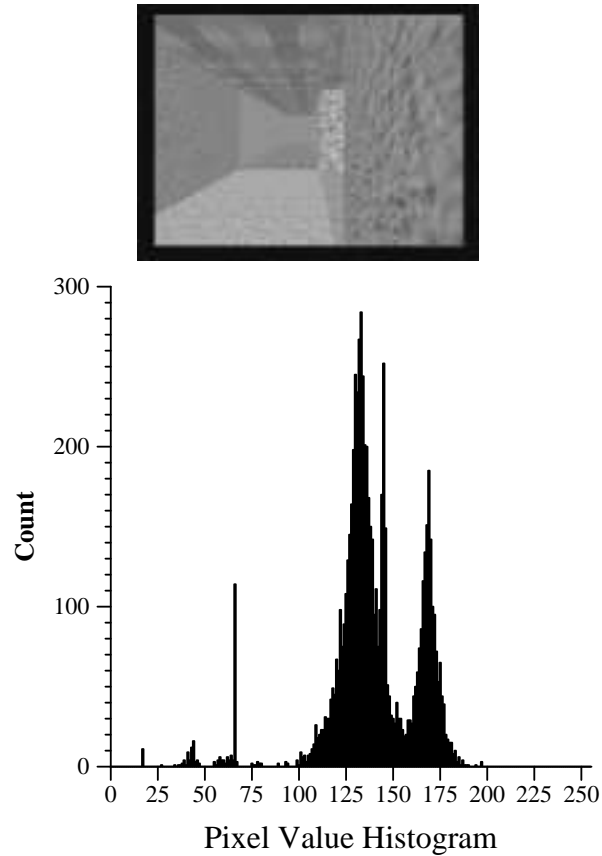


Figure 3: An Image and Corresponding Histogram

6 Experimental Results

6.1 Overview

The method of view-based navigation presented was implemented in a simulated robotic environment. The simulation was constructed by interfacing BeeSoft’s robotic simulation environment with a virtual reality system. The nature of the system and the equipment used led to several issues to be solved. The predominant problem was correctly associating behaviors with choice points while learning.

6.2 Closeness is Difficult

The examples provided in this paper describe many of the methods used in our experiments. Navigation using solely optical flow has been successfully implemented before, and was found to be very effective. Matching detected choice points to learned choice points was successful at a high rate once histograms were used. Determining the possibility of a behavior was successfully done as well; all that was needed was to state the constraints under which the behavior could execute properly.

However, correctly computing the ‘closeness’ values of behaviors given only visual input proved to be difficult. Without odometric information, determining what the robot is doing can be difficult itself; extrapolating given past history what a behavior would be doing merely compounds the problem.

Consider this example: the robot is slowly turning left. The optical flow field illustrates this by LEFT-RIGHT-BALANCE returning a value of 1. This information is insufficient to determine which behavior best models what the robot is doing: the robot could be turning left because the flow balance was too high for a ‘follow the left wall’ behavior, or because the flow balance was too low for a ‘follow the right wall’ behavior. The only way to determine is to inspect what the flow vector field was like before turning, and determine *why* the robot is turning left.

One possible solution is to not detect choice points when the robot is turning. However, then it is difficult to determine the choice point that might be associated with the turn, as the change in the flow field that marks a turn is no longer considered.

6.3 Flow at Corners

As a robot approaches an intersection, the walls to the right and left end, leaving empty space.

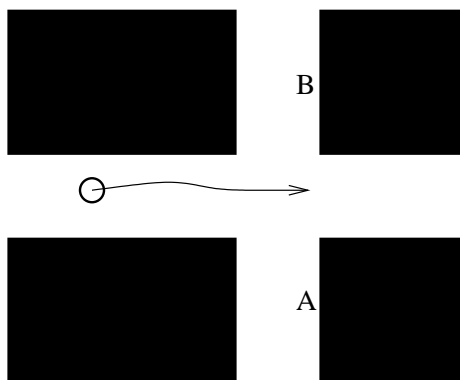


Figure 4: An Intersection

One might expect that as the robot approaches an intersection, the flow on the side of a hallway would decrease; the wall that used to be next to the robot is no longer there. In real world environments this is

often the case. However, due to the nature of the virtual environment we used, this turns out to not be true. Instead, the flow actually increases slightly, and then decreases as the robot passes the turn.

This slight increase in flow is from the facing wall of the intersection (labelled A and B in the figure). The texture on the walls parallel to the direction of the robot is blurry, as a complicated texture is viewed in a small area. The texture on the facing walls, however, is perpendicular to the direction of the robot, and so the texture is very crisp. The flow generated by the texture on the facing wall is slightly greater than the flow generated by the parallel wall.

This makes detecting turns difficult. What the robot has to do is try to detect the slight increase and then corresponding decrease, and turn enough to enter the turn. Our implementation was reasonably successful at this, but still make frequent errors.

7 Future Improvements

7.1 Solving Closeness

Constructing a simple method of determining closeness in the absence of odometric data is difficult, at best. Therefore, one of two approaches can be taken. First, the robot can have access to odometric information in order to determine what motor commands are being given. From this, a comparison of the motor commands that a behavior would give and the present motor commands can be made. Second, the task of determining closeness can be removed altogether. Instead of leading a robot along a path, a path can be specified as a set of directions, a set of choice point-behavior pairs. Some preliminary work on this latter option has had good results.

7.2 View Matching

Pixel value histograms proved to be effective when the choice points have very different textures associated with them. In a more homogenous environment, however, they are not as effective. A more complex method of view matching could take advantage of distinctive features of the environment, such as corners, hallways, etc., through segmentation or edge detection. Given these distinctive features, one would have to be able to compare them for similarity in order to find the best match.

7.3 Exploration

Instead of learning a path, the robot could explore a scene and construct a graph of the choice points of the environment, with behaviors labelling edges. Once a graph has been constructed, a starting point and finish point could be supplied, and the robot would then construct its own path given the graph.

7.4 Choice Point Selection

Currently, a brief history of possible behaviors is kept in order to determine choice points. The method used allows for a transient error to indicate a choice point. For example, let there be a set of possible behaviors A and B, where $A \neq B$. Then, the sequence

AAAAAAAAABAAAAAAAAA

will lead to a choice point being detected. Roughly thirty frames are computed a second: this single instance of B lasts for one thirtieth of a second, and can therefore be safely discarded. What we really want to detect are changes that persist for some time, while eliminating multiple choice points from stuttering. For example, the sequence

AAABAABBABBBBBBBBB

should detect a single choice point.

8 Appendix A: Biological Navigational Systems

8.1 Overview

Research on applying principles of navigation from cognitive science to robotics has become increasingly popular in the past few years. This effort has been partially due to interest in the principles of ‘ecological robotics’ proposed by Gibson. A broad overview of a few well understood animal navigation processes is made, then focusing on honey bees. The navigational strategies of honey bees bear remarkable resemblance to those used by our robot, and other robots developed at the Max-Planck Institute in Germany.

8.2 Spatial Cognition and Navigation

Navigation in an environment is often characterized in terms of three levels of cognition:

- Route following,
- Topographic maps, and
- Metric maps.

Each successive level represents a greater knowledge of the surrounding environment.

Route following is the ability for an organism to correctly navigate a learned route from one place to another. However, as the organism has no knowledge of the relationships between places, creating novel paths is very difficult. For example, honey bees may know the route from the hive to feeding site A, and from the hive to feeding site B. This knowledge, however, does not mean that a bee can easily travel directly from site A to site B, without using the hive as a waypoint. Another example of this limitation is a person’s knowledge of roads when travelling by car. Just because someone knows how to drive from Boston to Chicago and Chicago to New Orleans doesn’t necessarily mean that they know how to drive directly from Boston to New Orleans: very different routes must be taken.

In contrast, a cognitive topographic map allows for this kind of novel route construction. The organism is capable of extrapolating relationships between important points. If displaced, an organism can determine where it is (from its displacement), and act accordingly. A common distinction that is made between these two forms of navigation is the presence of landmarks. In several studies, the ability for an organism to effectively navigate to its destination in the absence of known landmarks on the route it takes to that destination is demonstrated as proof of a cognitive map of the environment.[9] [10] [2]

One example of this distinction being used experimentally is an study by Fred C. Dyer. A bee hive had two nearby feeding stations (small cups of sucrose solution). One feeding station was placed in an open, wooded area (site A). The other was placed in a deep quarry (site B). Bees which were trained to feed from site A were displaced to site B, and vice versa. The bees that were displaced to site A flew directly to site B. The bees that were displaced to site B mostly flew in the compass direction they were originally travelling, although some returned to the hive.

The important difference between the two sites was the visibility of the route to the other. Within the deep quarry, the path to site A was not visible. However, from site A, the path to the quarry was visible. More importantly than the path itself, which may have been slightly obscured, the possible landmarks which a bee might use to navigate along the path were visible.

The presence or lack of this ability, being able to construct a new path in the absence of the landmarks used to reach the destination, is the distinction made between having an internal ‘cognitive map’ of the environment, and only knowing routes between points in the environment.[9]

The most complete knowledge of an environment is obtained through having a metric map. Not only does the organism know the general spatial relationships between important points: the organism can construct paths based on internal representations of distances. A well studied example of this ability is the Saharan desert ant (*Cataglyphis* spp). These ants are capable of travelling back to their nest on a very direct path, even after wandering in a circuitous route while searching for food. Experiments have shown that the ants keep a precise metric representation of their distance from the nest, which is stable for at least several hours. They can retrace a path back to their nest quite precisely even when direct routes are not available; one experiment forced them to take a right angle turn (so that the direct path represents the hypotenuse of a right triangle). The ants began searching for the nest within one percent of the net distance they had travelled. [21]

8.3 Robot Cognition and Honeybees

Considering these three levels of cognitive representation, it is fairly clear which our robot uses: route following. Our robot is not capable of travelling novel paths to reach a destination. An inspection of the methods used by animals to travel routes can therefore lend important insights into better methods of robotic navigation.

One route following organism that has been extensively studied is the honeybee. It has been established that honeybees use learned routes to navigate to and from food resources.[9] This conclusion still leaves many details unresolved: for example, how bees locate their goal, how bees navigate towards landmarks, and how bees resolve conflicting landmarks. We will review the results which suggests answers to these questions, in order to provide an overview of the honeybee’s navigational system. It is important to note that a ‘landmark’ to a bee may be very different than a ‘landmark’ to a person. In terms of navigation, a landmark represents a discernable feature of the environment which the honeybee uses to navigate. It could be a barn, a tree, a lake, or a patch of vegetation which they bee can easily distinguish.

8.3.1 How do honeybees detect their goal?

Bees show evidence of what can be described as a heirarchical method for finding their goal (experimentally, this is usually an abundant food source of sugar solution). In the presence of a single landmark, a bee determines the distance the goal is from the landmark by its size. This has been demonstrated experimentally. Honeybees were trained to obtain food from a feeding station in a featureless white room, with a small black cylinder a small distance away from the food. The room was constructed as to supply as few navigational cues as possible, with the exception of the obvious landmark. When the food source was removed, the bees searched in the place it used to be. When the landmark was made larger, the bees searched further away from it, in an area where its size on the retina would be the same as when they obtained food. When the landmark was made smaller, the bees searched closer.

When presented with multiple landmarks, the bee relies on an angular representation. The bee tries to match a point at which the multiple landmarks are at identical compass points. This was demonstrated by scaling the distances between all of the landmarks (expanding the landmark array). When this was done, the bee looked in a place where the compass points of the landmarks were identical to where she remembered them. In the case of multiple landmarks, the size of the landmarks seemed to have little effect on where the bee searches.[3]

The angular method of detecting a goal is not without problems. In some landmark arrangements, there is not a unique place the goal could be. Similarly, in highly symmetrical arrangements, the actual goal is not

uniquely discernable by directions alone. The way that a bee solves this problem is by biasing the direction it faces when it looks for a goal. By always facing a single direction, the bee can translate relative compass points into global compass points. When presented with a landmark, a bee always searches for the goal facing south, except when the landmark is north of the goal, at which point it faces north: if the landmark is north, the bee cannot see it when it faces south at the goal. It has been determined that the bee uses the magnetic field of the earth to detect south, and not other sensory cues.[5]

8.3.2 How do honeybees navigate towards a goal using landmarks?

Unlike vertebrates, honeybees are not capable of perceiving depth solely from their retinal images: they do not have stereoscopic or binocular vision. Instead, they obtain distance information about their environment from motion and size. Objects which move more quickly or are larger are gauged to be closer than objects which move more slowly or are smaller.[17] They also use methods of optical flow balance similar to that used in our robot; when a bee navigates down a narrow hallway, it stays close to the center. If one side of the hallway is made to move parallel to the bee’s motion (up or down the hallway), the corresponding change in optical flow causes the bee to shift to the expected side of the hallway as if it were trying to balance the flow vectors.[19]

Several theoretical models on how honeybees travel to a goal have been discussed. The most promising is a model presented by Cartwright and Collett. In this model, a honeybee compares the perceived compass angles of a set of landmarks with the desired compass angles, and moves to make the two equivalent. This is accomplished by the angle between landmarks being weighted by their difference with the desired angle, and motor commands being calculated by these weights. The model does not behave exactly as a bee does; specifically, it differs in behavior when the landmarks are suddenly changed. [3]

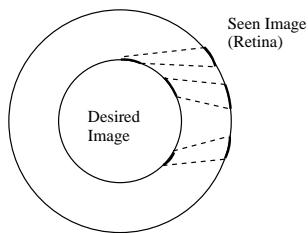


Figure 6: A Honeybee’s Recorded Snapshot and View

Bees communicate paths, not landmarks. It has been determined that bees have a ritualized dance they perform to indicate the direction and distance of a food source to other bees. The direction is defined in terms of the polarization of sunlight caused by the atmosphere. When it is overcast (and there is no significant polarization), bees are still capable of navigating the path to the food source, but will inform other bees of its location by the direction they had travelled on sunny days. This was determined experimentally by moving a bee hive on an overcast day from a training to a test site which was identical except for the compass direction of the feeding site. Bees still travelled to the food source without difficulty, but informed other bees that its direction was the direction in the training site. Once bees have found the food source, they can remember landmarks along the way and retrace the path. [8]

8.3.3 How do honeybees resolve conflicting landmarks?

Well fitting models of how honeybees match and detect landmarks have been presented and supported by experimental evidence. All of these studies, however, dealt with an unambiguous situation: there was a set of landmarks which, if the bee moved into the correct position, matched the set of landmarks remembered

from previously visiting food. These studies do not address the question of how honeybees determine which set of landmarks to match, if presented with conflicting information.

Experimental evidence has shown that bees weight landmarks according to their distance from the goal. If presented with a set of conflicting landmarks, the bee will navigate by the ones which are closer to its goal. This was established experimentally by placing different arrays of landmarks, some close, and some far, and then altering the arrangement. [4]

8.3.4 Honeybees → Robots

Robots have been developed which utilize navigational systems very similar to that of the honeybee. One robot in particular has remarkable similarities, which are not coincidental. The robot's sole input was a 360°, one dimensional image of greyscale values of the horizon.

It can be mathematically shown that in order to reach a goal position, the direction to travel is towards the image regions with maximal contraction with respect to the landmark snapshot. In a small arena 118 by 102 cm, a 10 cm tall robot was able to home in to goal positions correctly at distances of 45 cm, and navigate correctly 95% of the time at distances less than 15 cm.[12]

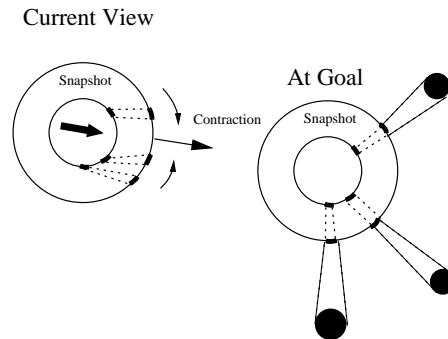


Fig 7: The Navigational Method of the Robot (from Franz et al.)

This robot was not only capable of navigating by utilizing image snapshots, but was also able to explore the environment and construct a view graph. However, in this case, the idea of a 'view graph' is somewhat misleading: since the robot had a 360° field of view which was direction independent (didn't change as the robot turned), then a single place in the environment could be represented by only a single view. Although the robot was using its vision to separate and distinguish places in the graph, the equivalence between views and places effectively made the corresponding graphs synonymous.

The method of choosing views which comprise the graph consisted of three requirements: there should be a small number of views, views should be distinguishable, and views shouldn't be so far apart as to make navigation between them difficult. The robot was programmed to home in on a previously visited vertex of the graph if it thought it was close, thereby adding edges (new routes) to a previously visited vertex. With these three criteria, complicated graphs of the environment could be constructed. So, although principles of honeybee navigation were used, cognitive maps were constructed.

8.4 Applying Cognitive Principles to our Model

An important difference between a bee's choice point selection and our model's is the preference for choice points. In our model, each choice point is weighted equally, even though they were recorded in order. One possible modification which could lead to better choice point matching would be to weight choice points that are closer in time to the last choice point visited over more distant ones. This could lead to temporal

differentiation of otherwise visually identical choice points, which would allow for intersecting paths to be traversed with a more certainty of correctness.

8.5 Application of our Model to the Honeybee Methods

The methodology applied to honeybee navigation has many parallels with our navigational methods. For example, honeybees navigate through taking ‘snapshots’ of their goals, then navigating by determining their direction of motion from their current view, in order to reach the ‘snapshot’ view. Of course, there are significant differences: it is believed that honeybees extrapolate the direction of travel from what they see, instead of corresponding behaviors to given snapshots.

However, unlike Franz’s work, which constructs cognitive maps of an environment, our navigational method learns and follows routes, as honeybees do. The methods of learning are similar: the bee (or robot) is given instructions on how to reach a goal, and learns landmarks to navigate by as it travels to that goal, then uses those landmarks on subsequent trips. However, what comprises a ‘landmark’ is very different; for bees, a landmark is a visually distinguishable object, such as a dark cylinder on a light background. For our robot, a landmark is defined in terms of the behaviors the robot has, and can be visually indistinguishable from another landmark.

Seen from another angle, however, these two models of navigation can be made equivalent. Doing so necessitates the use behaviors which are far more complicated than those presented as examples that our robot used. A bee’s method of navigation can be abstracted into two behaviors:

- Travel to a choice point and
- Look for a choice point.

Travelling to a choice point has a given choice point associated with each instance of it in the navigational sequence. At a certain choice point, the ‘travel to choice point’ behavior is associated with the next choice point on the route, so that the bee moves to the next choice point. This is necessary due to the proposed navigational method of the bee, which matches the viewed image to a recorded image.

When the current location does not match any given choice point (the bee is lost, or has been displaced), it searches for possible choice points nearby, and travels towards them, preferentially towards the choice point associated with home. When the choice point is reached, the travel behavior is no longer possible. This will lead to selecting a new choice point, which will match to where the robot is.

Of course, these two behaviors are much more complicated than those presented previously. However, this demonstrates robustness of the model; it can be applied to varying degrees of navigational complexity. Similarly, the bee’s behavior could be reduced to many more specialized behaviors.

Learning under our model, however, proves problematic on a single test run: the bee must correlate a destination point (the next choice point) with each learned choice point. Unless it binds a behavior to a previous choice point when a new one is learned, the robot must be able to see into the future in order to determine where it should go. However, if the robot can bind a choice point to the ‘travel to choice point’ behavior *after* when it selects the choice point which would be destination, then the model can be applied.

8.6 Conclusion

Animals use a wide variety of navigational method. One organism, the honeybee, has influenced robotics research and led to at least one proposed model of robotic navigation. Experimental analysis of the methods by which a honeybee navigates has shown the underlying mechanisms to be a simple hierarchical route-based navigational model. The bee’s methods of navigation can be translated into our navigational model. Learning under our model would require a minor change, due to the nature of the navigational behaviors involved. Considering the methods by which honeybees navigate suggest possible improvements to our model.

9 Appendix B: Experimental System

9.1 Overview

The robotic agent uses solely visual input during training and navigation. Both visual (image) and optical flow are used. Images are used for navigation choice point matching. Flow is used for navigation and navigational choice point selection. BeeSoft's robot simulator simulated a physical environment for a robot. WorldToolkit, provided the virtual environment for the robot to see. The two were interfaced so that BeeSoft would provide position information to WorldToolkit.

9.2 Hardware / Software

The robot and simulator were run on a Pentium II system running Linux. It used a Matrox Meteor frame grabber for image acquisition, and Teleos Research's AVP software for optical flow calculation. Image and flow vectors were obtained at a rate of just under 30 Hz. View images were color and greyscale 128 by 96 pixel images. Flow information was stored as an array of 20 by 12 flow vectors. WorldToolkit was run on a Silicon Graphics Onyx 2 with a single Reality Engine graphics card.

9.2.1 Simulator

The BeeSoft Simulator is a software package that allows the programming and testing of robot programs without using an actual robot. The robot moves through a simulated 2-dimensional environment, that provides data to the robot sensors according to its surroundings. Walls, doors, and other features can be added to the environment. As only visual input was used, the sole function the simulator used was to compute the robot's position based on its motor commands.

9.2.2 WorldToolkit

WorldToolkit is a high level virtual environment toolkit built on top of OpenGL. It allows for the creation of virtual visual environments. It is used in the Cognitive Science Department as a VR environment for experiments in human navigation.

9.2.3 Interface

The robot and BeeSoft simulator were interfaced with WorldToolkit to coordinate the two environments. This was achieved by having the BeeSoft simulator transmit the current position of the robot to WorldToolkit. A polling system was used, so that WorldToolkit polled the BeeSoft simulator for the robot's position. The BeeSoft simulator updated the robot's position one hundred times a second. WorldToolkit produced frames of the environment at a rate of 60 Hz.

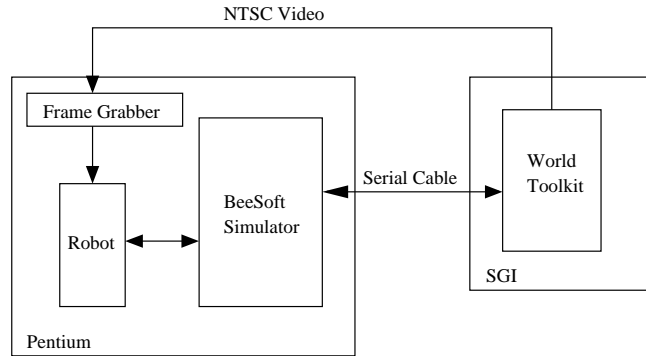


Figure 8: Experimental System

9.3 Robot

The robot API is an asynchronous pre-emptive motor control model. When a call is made to make the robot move, it returns immediately, and the robot begins to service the request. If another movement request is made while the first is being executed, it pre-empts what is already being done. For example, if the robot is told to move ten meters forward, but halfway there, is told to stop, it will stop immediately.

This asynchronous model can complicate optical flow calculations. For example, while turning, the optical flow the robot sees is very different than when moving forward. Therefore, while the robot is turning, it must take this difference into account when determining what action to take.

References

- [1] **Camus, T.** Real-time optical flow. Doctoral dissertation and technical report CS-94-36. Providence: Brown University.
- [2] **Cartwright, B.A. and Collett, T.S.** (1982) How honey bees use landmarks to guide their return to a food source. *Nature*, Vol 295, 18 February 1982: 560-4.
- [3] **Cartwright, B.A. and Collett, T.S.** (1983) Landmark Learning in Bees. *Journal of Comparative Physiology*, 151:521-43.
- [4] **Cheng, K., Collett, T.S., et al.** (1987) The use of visual landmarks by honeybees: Bees weight landmarks according to their distance from the goal. *Journal of Comparative Physiology A* 161:469-75.
- [5] **Collett, T.S. and Baron, J.** (1993) Biological compasses and the coordinate frame of landmark memories in honeybees. *Nature*, Vol. 368, 10 March 1994: 137-40.
- [6] **Duchon, A. P., Warren, W. H., and Kaelbling, L. P.** (1998) Ecological Robotics. *Adaptive Behavior*, Volume 6, Number 3/4.
- [7] **Dyer, Fred C.** *Spacial Cognition and Navigation in Insects.*
- [8] **Dyer, Fred C.** (1987) Memory and sun compensation by honey bees. *Journal of Comparative Physiology A*, 160:621-33.

- [9] **Dyer, Fred C.** (1991) Bees acquire route-based memories but not cognitive maps in a familiar landscape. *Animal Behavior*, 41: 239-46.
- [10] **Dyer, Fred C., Berry, Nancy A., and Richard, Amy S.** (1993) Honey bee spatial memory: use of route-based memories after displacement. *Animal Behavior*, 45: 1028-1030.
- [11] **Dyer, Fred C.** (1993b) How honey bees find familiar feeding sites after changing nesting sites with a swarm. *Animal Behavior*, 46: 813-6.
- [12] **Franz, Matthias O., Schölkopf, Bernhard, Mallot, Hanspeter, and Bühlhoff, Heinrich.** Learning View Graphs for Robot Navigation.
- [13] **Gillner, S. and Mallot, H.** (1998) Navigation and Acquisition of Spatial Knowledge in a Virtual Maze. *Journal of Cognitive Neuroscience*, 10:4, 445-463.
- [14] **Gould, J. L.** Honey Bees Store Learned Flower-landing Behavior According to Time of Day. *Animal Behavior*, 35: 1579-81.
- [15] **Gould, J. L.** The Locale Map of Honey Bees: Do Insects Have Cognitive Maps? *Science*, 16 May 1986: 861-3.
- [16] **Kuipers, B. and Byun, Y.** (1991) A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations. *Robotics & Autonomous Systems* 8, 47-63.
- [17] **Lehrer, M., Srinivasan, M. V., Zhang, S. W., and Horridge, G. A.** (1988) Motion cues provide the bee's visual world with a third dimension. *Nature*, Vol 332, 24 March 1988: 356-7.
- [18] **Schölkopf, B. and Mallot, H.** (1995) View Based Cognitive Mapping and Path Planning. *Adaptive Behavior*, Vol. 3, No. 3, 311-348.
- [19] **Srinivasan, et al.** (1991) Range perception through apparent speed in freely flying honeybees. *Visual Neuroscience*, 6, 519-35.
- [20] **Srinivasan, M. V., Zhang, S. W., and Rolfe, B.** (1993) Is pattern vision mediated by 'cortical' processing? *Nature*, Vol. 362, 8 April 1993: 539-40.
- [21] **Wehner, R. and Wehner, S.** Path integration in desert ants: Approaching a long-standing puzzle in insect navigation. *Monitor Zoologico Italiano (NS)*, 20:309-31.