Introduction

Navigating through the world using vision can be a complicated task. The constant changes in lighting conditions as well as various objects motions can make the analysis of patterns very challenging. In addition to, the various methods of extracting patterns from images for classification that do exist can be very complex and time consuming. This often is not sufficient for real time navigation (by the time you notice you are approaching a tree, you might hit it). Real time is therefore defined as the speed required for making optimal decisions (avoiding the tree in time). However, recognizing a tree in a given image might be a complex task, which requires a great deal of processing. Fortunately, this high order processing is not required at every single frame, since objects in our world do not change every millisecond. We could then recognize the tree in a slower rate then we need to track its position. This can be accomplished by biasing low level features detectors to shift our attention toward the tree in real time, without recognizing the tree in real time. As a result, the integration between slower processes, which are used to extract high order understanding, must interact with faster processes so that decision can be processes in real time.

One method of navigation through the world can be accomplished by using landmarks. The idea of landmark-based navigation has been in existence for many years. In its simplest form, you pick a landmark as a goal position and simply go toward that landmark to reach the goal. This was well known from the way insects navigated (Tinbergen, 1932; Anderson, 1977; Wehner and Räber, 1979; Cartwright and Collett, 1983). Once that goal position is reached, the agent can then navigate toward a new goal position and ultimately reach the “final goal”. This type of method relies on the fact that every goal position can be correlated with a landmark. If a landmark does not indicate the goal position, then landmarks around the goal position can be located by their bearings (angles) to the goal position and to one another. This type of method is known as guidance or piloting (O’Keffe & Madel, 1978; Trullier et al. 1997).

The previous methods discussed above have been used for many years to develop autonomous robots that are able to visually navigate. One of the main challenges of these navigation schemes is locating the landmarks in real time. In addition to, the notion of a landmark is not coherently defined. That is, it is difficult to select which landmarks stay the same under various conditions.

Aspiring from the insect world, it has been proposed that insect store a template or a snapshot of the goal position and simply navigate toward that goal until the image on their retina matches the stored snapshot (Wehner, 1981; Srinivasan, 1994). Other models attempted to claim that a parameters based landmarks are used (Anderson, 1977, Lambrinos et al., 2000; M¨oller, 2000). These models states that features from the image are extracted in order to find the "goal image". Moller (2001) has proposed that it is probably a combination of both, which help the insect find their home.

Despite the fact that insects can find their way home from different locations, they still can not navigate from one location to another (Collett 1993, Wehner & Mezel 1990). Therefore, for navigating a path, a cognitive map is required. Cognitive maps have been found to exist only in vertebrates throughout the animal kingdoms. These maps allow an agent to be able to determine where he is in the world and what he needs to do. Some experiments have shown that a type of recognition-triggered response exists in humans (Mallot & Gillner 2000). That is, when a person is at a particular location, particular response information is associated with it. The information can include "turn left here", or more complex types of information like "follow the yellow brick road". Other experiments have also found that the next intermediate destination (the next landmark) exists in subjects following a path (McNamara et al. 1984).

The Robotic System

This paper proposes a navigation scheme based on landmark navigation. The system integrates high order image understanding using SIFT (Lowe 2004) with lower image tracking techniques (biasing attention using template matching) to navigate a robot through the environment. For simplicity, only goal positions, which can be correlated with landmarks, are considered. Therefore, this project focuses on finding the landmarks and moving toward them. That is, the robot would drive directly toward the landmark to reach the goal. However, implementing a bearing type of navigation is simple once landmarks recognition can be accomplished. Furthermore, the concept of landmark is taken to be any place in the image within a fixed window size that can be found in the image despite various changes (view point, time of day, etc.). This process is accomplished by finding invariant key points derived from the landmark position and matching these key points with a stored set of key points to determine the landmark position using the SIFT algorithm. This method, however, heavily relies on the assumption that the landmarks would not change position and are fixed. For example, choosing

a tree as a landmark is a good choice, but choosing a dog as a landmark is not. This problem was solved by having human choose the landmark position for the robot to follow. Once the landmark is detected, a low level processing is used to track the landmark and navigate towards it. This is accomplished using a model of attention (Itti 1998), which is biased toward finding the landmark in subsequent frames. The process then allows tracking in real time.

Finding Landmarks to follow

In order to robustly find the same key points in a window under varieties of transformations (scale, luminance, location) the SIFT algorithm proposed by Lowe (2004) is used. However, two problems were encountered when trying to use the algorithm for tracking the landmark position in real time. The first problem was that the algorithm lost the landmark position very fast over a few frames. This was mainly due to small changes in the landmark due to 3D transformation as a result of the robot moving. That is, when the landmark was initially added to the database of key points, it is at a smaller scale and at a particular angle from when the robot would see it as it moves toward the landmark (scale would get bigger and the angle to the object will change). Since the SIFT algorithm works best when the landmark is added at a larger scale and then found at lower scale, it losses the landmark (See experiments 1-4).

One way to solve this was to have additional key points represent the same landmark position (to account for the small variations). As a result, the addition of these key points added more time to the algorithm, and confused the match after a few sets of key points were added. Additionally, the algorithm occupied more memory as a result of the larger database.

The second problem with the algorithm is that it is very inefficient at computing and comparing the key points in real time. Searching through a database containing about 120,000 key points took over 7 seconds. This can caused the robot to loose direction very fast. For example, if the robot found a landmark and tried to drive toward it (but was not tracking the landmark in real time), the landmark might have moved off the camera by the time the same key point is located again. This would result in the robot losing its position and the landmark.

Tracking the Landmark

To solve the problems outlined above, it was found that template-matching algorithms could be used to track the landmark position for a sufficient amount of time. That would be done until the SIFT algorithm computed the next landmark position or the current landmark position (to keep the template matching from drifting). This enabled the robot to keep its heading toward the landmark so that the landmark would not be lost.

Various algorithms used for template matching were evaluated. These included the cross-correlation, normalized cross-correlation, squared difference, normalized squared difference, correlation coefficient, normalized correlation coefficient and Fourier transform implementation of template matching (Nixon & Alberto, 2004). It was found that the simple squared difference was able track the landmark the best. Using the attention model proposed by Itti (1998) it was found that template matching could be preformed on the low level features (i ntensity, difference between red and green, and difference between blue and yellow) to achieve better results then just using template matching on the raw image pixels. Furthermore, the template was also temporally changed by the following formula proposed by (Deans, 1997).

$$T(t) = \alpha * M(t) + (1 - \alpha) * T(t-1)$$

Where α is a constant, T(t) is the template at time t, T(t-1) is the template in the previous time step, and M(t) it the extracted template from the current found location. The constant alpha determined how much of the new template would be included in the update. For this paper, alpha was set to 0.9 as recommended by (Deans, 1997). Lastly, the located location weighted from the last found landmark to resolve any ambiguities or close matches. This meant that the landmark was not allowed to move very fast.

Due to time delays between each SIFT match, the SIFT algorithm could not simply return the position of the landmark for the template to track. This is because the image that the SIFT algorithm sees could be over a second old, which means that the landmark has already changed position by then. Therefore, returning the position of the landmark could result in the robot following a different landmark. Instead, the SIFT algorithm must return a description of where the landmark is in terms of low-ordered features. For this system, the SIFT algorithm returned a new template of the found key point from the delayed image. This template then

replaced the old template that was used for tracking. As a result, the tracking now was able to follow the correct new landmark position.

Following a path

In order for the robot to follow a path, it must robustly find a landmark, drive toward the landmark until it found the next landmark. However, storing multiple landmarks position in a single database for matching proved to be difficult. Adding multiple objects to the database increased the amount of time and space (memory) required for matching as well as increased the number of mismatches. This was because we needed to find the current landmark among other stored landmarks, even though most of the landmarks would never be matched. This is because most of the landmarks in the database are not going to be reached for a long time (until the robot arrives at that particular leg). Furthermore, some of these landmarks have already past, in which case we are not interested in finding them anymore. A variety of ambiguities also resulted from when multiple landmark position would appear on the image at the same time (for instance, when the robot was taking a turn). This caused the system to not know which landmark to choose next.

So solve these issues, multiple databases where created to hold the different landmark positions. At each leg in the path, only two databases of landmarks would be loaded. The database containing the key points for the current landmark, and the database containing the next landmark to follow. This can be thought of as keeping the current landmark in memory and only priming the next expected landmark. To determine when was the appropriate time to switch databases (make the next database the current and load the next leg database), the scale of the next matched landmark was used. If that scale was equal to one, then this meant that the landmark is found where we switched landmarks during training. That is, we have arrived at the current goal location when next landmark appears at the same position from were we seen it during training. The scale of the current landmark can also be used in conjunction with the next landmark scale to help make finding the goal location more robust. This method enabled the robot to have multiple representation of the same landmark without interfering with other irrelevant landmarks. The method also conserved on memory, since not all of the landmarks had to be loaded at the same time. That gave the robot the ability to follow an arbitrarily long path, since only two small databases needed to be loaded into memory.

Training the robot

A user, who guided the robot on the desired path, trained the robot. The user was presented with a window showing the image returned from the robot's camera. The robot was then made to follow a path by clicking on the desired landmark that the robot should navigate to. To change direction and follow a new landmark, the user simply clicked on the new landmark in the window as the robot was moving. From the clicked point a 50x50 pixels window was examined and the SIFT key points returned and learned (stored in the database as the landmark to follow in the current leg). The robot would then also use a 21x21 template to track that point and drive toward it. This was done until the user clicked a different point, in which case the robot would store and follow the next point. This continued until the robot reached the desired

References

- Anderson, A. M. (1977). A model for landmark learning in the honey-bee. J. Comp. Physiol. 114, 335–355.
- Cartwright, B. A. and Collett, T. S. (1983). Landmark learning in bees: experiments and models. J. Comp. Physiol. 151, 521–543.
- Deans, M. (1997). Natural Landmark Based Navigation, < http://www.cs.cmu.edu/~deano/Landmark/tracking2.html>
- Itti L., Koch C., Niebur E., (1998) A Model of Saliency-Based Visual Attention for Rapid Scene Analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 11, pp. 1254-1259, Nov.
- Lowe, D.G. and David G. , (2004) Distinctive Image Features from Scale-Invariant

Keypoints, IJCV(60), No. 2, November 2004, pp. 91-110.

- Mallot, H. A. (2000). Computational vision: information processing in perception and visual behavior. 2nd edition, MIT Press, Cambridge, Massachusetts.
- Moller R. (2001), Do insects use templates or parameters for landmark navigation?, Journal of Theoretical Biology. May 7;210(1):33-45.
- Nixon, M., Aguado A. (2004) Feature Extraction and Image Processing, Butterworth Heineman, GB: Newnes-Oxford
- Srinivasan, M. V. (1994) Pattern recognition in the honeybee: Recent progress. Journal of Insect Physiology, 40 (3), 183–194.
- Tinbergen, N. (1932). Über die Orientierung des Bienenwolfes (Philanthus triangulum Fabr.). Z. Vergl. Physiol. 16, 305–334.
- Wehner, R. (1981) Spatial vision in arthropods. In: *Handbook of Sensory Physiology VII/6C: Comparative physiology and evolution of vision in invertebrates*, (Autrum, H., ed) pp. 287–616. Springer Berlin, Heidelberg, New York.

- Wehner, R. and Räber, F. (1979). Visual spatial memory in desert ants, Cataglyphis fortis (Hymenoptera, Formicidae). Experientia 35, 1569–1571.