

# Tracking a Rat in Three-Dimensional Space Using Stereo Cameras

Student Authors: Nathaniel Meierpolys, Michael Krahulec, and Daniel Wiebe

Secondary Authors: Olaf Hall-Holt and Reid Price

## Abstract

We consider the problem of determining the position and head orientation of a rat in three-dimensional space based off of two colored LED lights attached to the rat's head. In order to track the rat using only standard cameras (instead of range-finder technology), at least 2 cameras are required for triangulation. We split this problem into two separate parts: identifying the location of the LEDs in each frame of video data and using triangulation to determine the position of the rat in three-dimensional space.

The first step to our approach to finding the LED lights in the two-dimensional frames is to segment the image based on color and gradient data. Next, we use the color of the LED lights and their positions relative to each other to pinpoint the centers of the LEDs on the two-dimensional frames. One of the difficulties in identifying the LEDs occurs when the light from the LEDs reflects off other surfaces or is briefly obscured from view. It is also difficult to distinguish between the LEDs due to problems with overexposure. Our use of a combination of shape, position, and color data allows us to pinpoint the LEDs successfully.

For the triangulation problem, we first have to determine the position of the cameras in three-dimensional space with reference to an arbitrary origin. Our approach to this calibration step is to have the cameras take a picture of a stationary grid of black dots on a white surface with a known distance between the dots. Then, starting with a guess as to the position of the camera, we generate what the grid would look like, given that position. The disparity between the virtual image and the real image is then minimized, giving us the position of the camera in three-dimensional space. Based off this information, and the position of the LED in matching frames from the two cameras, we use triangulation to pinpoint the position of the LED in three-dimensional space. Thus, for each frame, we store the position of the rat, a vector indicating the direction its head is pointing, and a time-stamp for that data.

The results of this process will be used in conjunction with brain wave activity data as part of an interdisciplinary study of the neural basis for spatial navigation in a rat.

## Introduction

The problem of automatic detection and tracking of objects is a familiar one in the field of computer vision and has been analyzed in various contexts. The difficulty with this problem lies twofold in the large amount of video information to be processed and the inability of computers to recognize coherent objects and shapes the same way humans do. Basic objects that appear clear and concrete to the human eye are actually rather complex. Objects contain textures and shadows, potential for rotation and

changing orientation, and often blend into backgrounds, making them hard to identify.

Using data from a pair of high-quality video cameras, we locate two LED lights mounted to the head of a rat. The front and back LEDs have a different color, and of the position of each colored LED allows us to measure the orientation of the rat at any given time. This information can be combined with measurements of individual neurons in the brain to study the neural basis for spatial navigation in the rat.

The most difficult aspect of our case is the nature of the object being tracked. When tracking a live animal, we encounter unpredictable difficulties with occlusion from objects in the maze. We also encounter moments when the physical LEDs reflect off walls to create virtual ones in the center of the reflection, confusing our system. If these complications weren't enough, the movement of the rat changes the orientation of the LEDs at varying speeds, which makes tracking the rat's movements very difficult.

Our approach relies on image data from two cameras, without need for range-finding equipment. Processing data is performed by a software pipeline involving four distinct components. Cameras are calibrated using a calibration target within view of both cameras. Images of the rat from two cameras are segmented in parallel with the help of a powerful Beowulf cluster environment. Images are then processed to determine the location of LED lights using a doubly-linked half-edge data structure. Finally, identified points are triangulated into three-dimensional space from pairs of stereo images. The process is structured to allow the processing of many frames automatically. We sacrifice real-time speed for highly precise measurement of LED light position.

## **Calibration**

In order to utilize images from cameras to triangulate the positions of objects in three-dimensional space, we must develop a concept of what a "camera" is, and how we can translate the images it gives into meaningful data. We consider a pixel on an image as color and position information about a particular ray of light that passes through the focal point of the camera and a specific place on the light receptor of the camera (See Figure 3). As seen in the diagram, the actual image taken by the camera can be placed in space such that the ray of light passes through the pixel on the image. If we can determine the three-dimensional position of the camera's focal point, and the size and appropriate position of where the image can be placed in space, then we know everything about the camera needed for triangulation. Throughout this next section, we will call the correct position and orientation of the image the camera's screen.

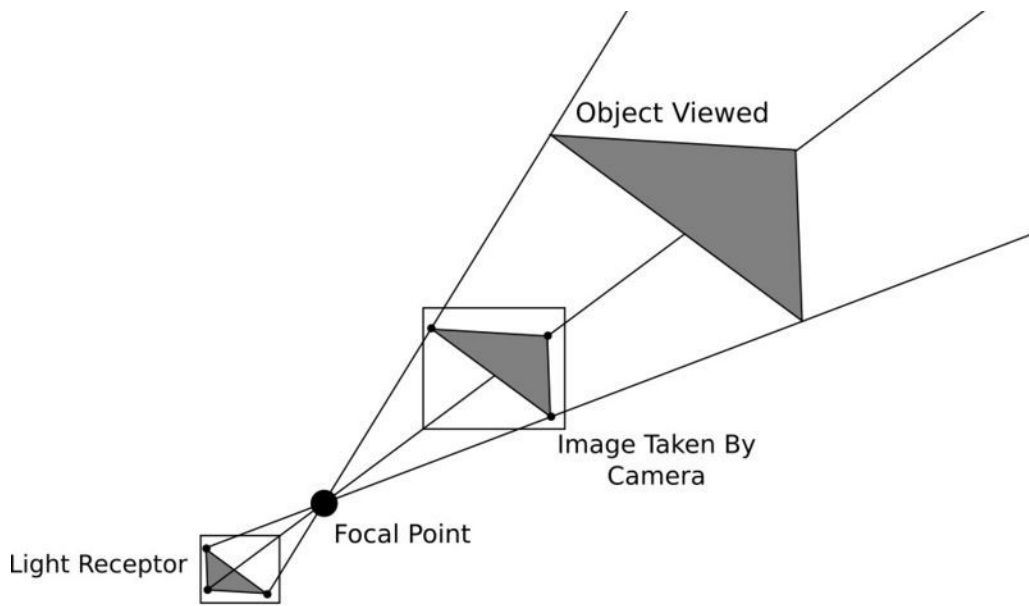


Figure 1: A Basic Camera

Calibration is the process of generating a three-dimensional coordinate system and determining the positions of the screen and the focal point of a camera in that coordinate system. In our method, we set up the cameras in the position they will be throughout the whole experiment. We then place a calibration target within clear sight of both cameras. This target is a grid of black dots on a white sheet of paper, with an odd number of rows and columns (Fig.1). The distance between the grid points is known, giving us a clear measure of length in the image. To help simplify our algorithm, the cameras must be set up such that pictures of the calibration target show the rows of dots to be parallel to the top and bottom edges of the image. The correct placement of cameras is displayed in (Fig. 2).

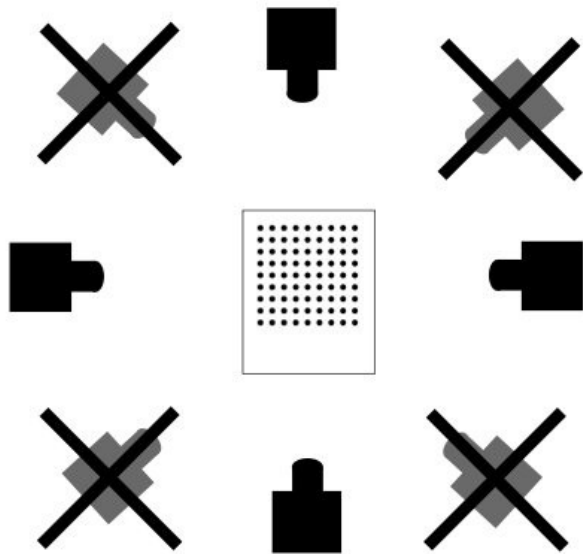


Figure 2: Example camera layout

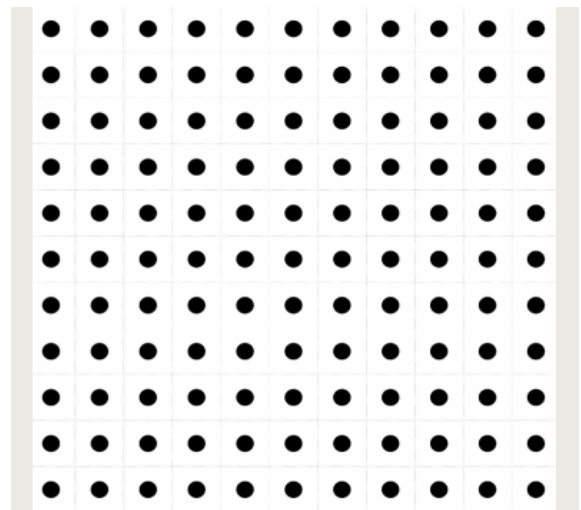


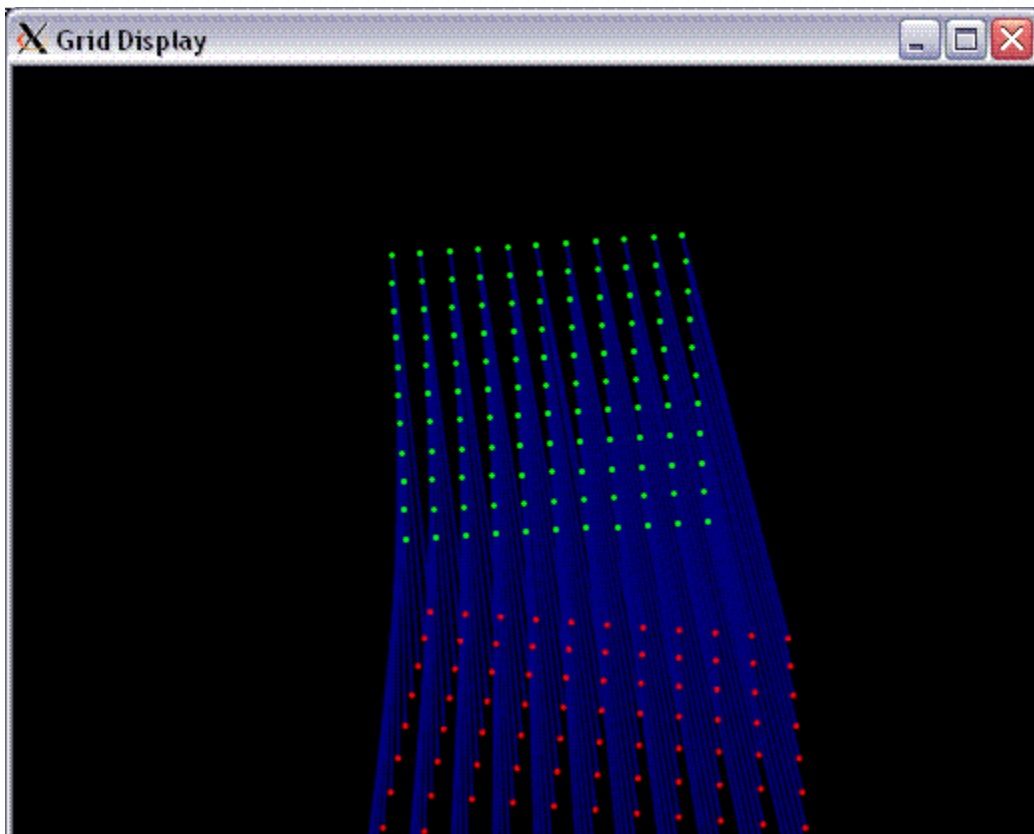
Figure 3: Calibration Grid

Once we've taken an image of the calibration target with both cameras, we continue with the experiment to its conclusion, for the calibration process can occur later, when we are actually processing the data.

Our calibration algorithm takes the images containing the calibration targets and produces all the necessary position data about the cameras. The first step in this algorithm is identifying the dots in the picture and putting them in grid-order. In order to do this, we generate a mask of a standard black dot with a square white background of a size inputted by the user. Next, we loop through all the pixels, starting at the upper left-hand corner of the image, placing the mask over the image, centered at the current pixel. We then count up the differences in color of all the pixels in the mask compared to the image pixels. This value is normalized and stored as color data in a new image at the location of the original pixel. The result of this process is also known as a disparity map. In order to isolate the grid points, we filter this disparity map, blacking out any pixels that have a disparity higher than a given tolerance level. We then find the pixel with smallest disparity in each dot-sized region of interest to determine the positions of the dots in the image. As long as the dots in the image were well-lit, this algorithm clearly identifies the positions of the dots.

The central dot is assumed to be the origin of our three-dimensional coordinate system. In order to determine the camera parameters with reference to the grid, we first guess at a series of camera parameters. Then, based on these parameters, and the distance between the grid points, we generate a virtual grid of points, and display what would appear on this cameras screen(See Figure 6). We then continuously vary the camera parameters slightly until we minimize the difference between the virtual model and physical reality.

Figure 4: Virtual grid vs. Physical Grid



## Capturing Video Data

### Camera Settings

We use two Sony BRC-300 cameras which allow manual control of shutter speed, iris, gain and zoom. The task of picking out bright LED lights can be greatly simplified using these manual settings. A high shutter speed means that the image sensor is exposed to light for a shorter period of time, causing LED lights to stand out against less bright elements of the background. Shorter length of exposure also reduces the risk of motion blur. Maintaining a low aperture setting opens the iris wider. This increases the depth of field, keeping more components of the image in focus. Both settings contribute to the reliability of our procedure for identifying LEDs.

### Scanline Issues

The cameras we use record interlaced video to effectively double the perceived frame rate of the video. The technique of creating interlaced video involves capturing all odd scanlines in one instant then all even scanlines in the next and finally piecing the two fields together to produce a single frame. This process improves the smoothness of video while reducing the amount of information to be broadcast or stored. In the context of our image-processing needs, however, the jagged edges make our analysis more difficult and less accurate. To deal with this problem and deinterlace image content, we first separate odd and even fields and then double each line to produce two distinct images from each frame recorded. The doubling ensures the maintenance of a consistent aspect ratio but unfortunately comes at the cost of losing vertical pixel precision since we cannot know exactly how closely our copied scanline matches the actual light entering the camera at that point.

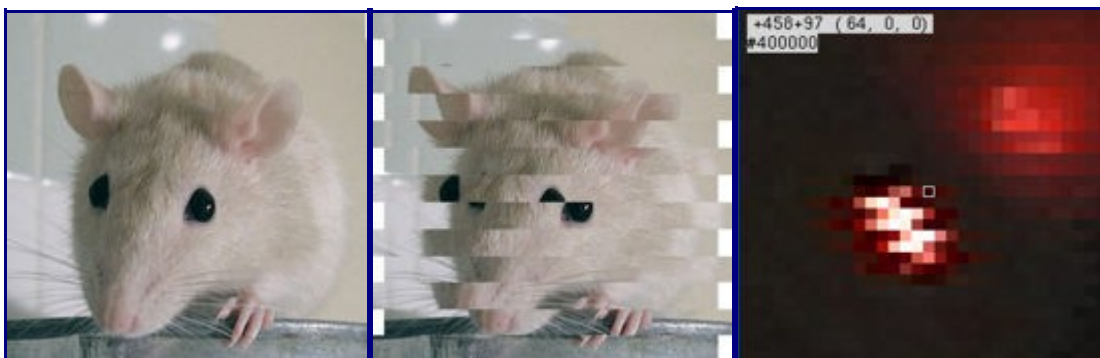


Figure 5: From left to right: Normal, deinterlaced image. The same image, with interlaced fields during movement. An actual interlaced LED image

Our relatively simple solution depends on knowledge of whether a line was initially copied up or down. For odd fields where all scanlines are copied down, we are essentially skewing the location of area within the face downwards. The same is true in the opposite direction for even fields where scanlines are copied up. To compensate for this, we adjust the vertical position of an LED's calculated center up by .5 pixels for odd scanline images and down .5 pixels for even scanline images. The extra

information gained by processing both odd and even fields allows us to more easily overcome cases in which the LED light is occluded in one scanline and visible in the other as is generally the case.

## Segmentation

In order to get any meaningful information out of the images we collect, it is necessary to analyze the image data for coherent shapes and color regions. We use a program developed by our co-researchers to segment our images. The program, called Eriol, breaks an image into shapes called faces and stores the information in a half-edge data structure. The Halfedge is a edge-centered data structure that is great for representing a polygon mesh, the exact structure of segmentations. The Halfedge Data structure received its name because instead of storing full edges of the mesh, we only store halfedges, where two halfedges together form a edge pair, and are referred to as "twins". This way each halfedge only belongs to a single incident face. Below is an illustration of a small portion of a halfedge structure.

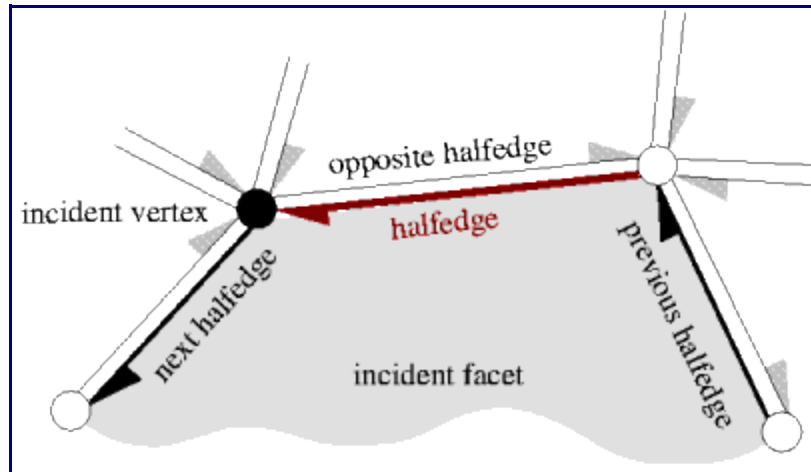


Figure 6: Halfedge example

The Halfedge data structure uses pointers to connect each face to its "outercomponent"(outer edge). Then, each halfedge has pointers to both the next edge, it's twin, and the incident face. This structure allows us to quickly move around the data structure and retrieve data in constant time per each piece of information gathered. We use the data stored in the faces of the structure to identify LED location and information more easily than sifting through raw image data.

The segmentation process itself is driven by the minimization of a cost function including four terms. These terms are *variance* (the change in color within faces), *boundary length* of faces, the *gradient* (change in color along edges), and the *number of segments* needed to describe the segmentation. The process starts with a grid containing the maximum number of pixel size squares and removes segments until the cost function reaches a minima.



Figure 7: A segmented image in Eriol. The upper-right bright spot is the actual orange LED, whereas the bottom left one is its reflection.

The very precise nature of this segmentation involves significant processing time. We employ the use of a Beowulf cluster of up to 34 machines, facilitated by the use of MPI message passing software. Together the cluster machines can segment many frames in parallel, increasing the speed of the process. Whereas a single machine processing a frame requires approximately 15 minutes to complete segmentation, many machines working in parallel reduce this task by a significant factor.

## Finding LEDs

### Picking out faces

Using the Halfedge data structure of the image segmentations, we can easily query the whole image to find the brightest faces in our segmentations. The LEDs are so bright that they superimpose themselves on the image creating a bright white spot where the center area of the LED is located. Because we have used the increased shutter speed to decrease the amount of light entering the camera, the two LEDs and possibly a reflection of one of the LEDs on the wall of the maze are the only bright spots that we find. Based on the area of the bright faces, we select the two largest bright spots as our LEDs, because the white center of the reflection of the LED on the wall is almost always much smaller than source LED itself.

## Identifying Color

Once we have identified where the LEDs are located in the image, we need to find color information about them to distinguish between the two LEDs, red or green. To find the color of the LED, since the center face is only a bright white color, we must circle around the image to find information about the color of surrounding faces. Luckily, with the Halfedge data structure, we can easily find this information in constant time. We take the mean color of all of the faces around the LED, and then, because there are only red or green LEDs, we can look at which value, red or green, is greater in the mean RGB value of the faces, and based on this we know the color of the LED.

## Finding centroids

Once we have obtained the color of the LEDs, we need to know the exact location of the centroid of our LEDs on each image to enable us to find the correct three-dimensional position using triangulation. To do this, we use a function that uses the vertices of the face to find an extremely accurate position of the centroid. The function is as follows, where  $c.x$  is the x-coordinate of the centroid and  $y.x$  is the y-coordinate:

$$c_x = \frac{1}{6A} \sum_{i=0}^{N-1} (x_i + x_{i+1}) (x_i y_{i+1} - x_{i+1} y_i)$$
$$c_y = \frac{1}{6A} \sum_{i=0}^{N-1} (y_i + y_{i+1}) (x_i y_{i+1} - x_{i+1} y_i)$$

Figure 8: Centroid function

This function is great for use in our segmentation because it does not weigh the location of the centroid based on the distribution of vertices. The coordinates of the centroid on both the right and left images are then fed into our triangulation function to find the exact three-dimensional location in space.

## Triangulation

Once the two cameras have been calibrated, we are able to use pairs of camera images to locate the points in three dimensional space. Given the position of an object in each camera's image plane, we can generate the ray that passes through the focal point of the camera and the position on the camera "screen." The object observed necessarily lies somewhere along this ray in space. Now if we have two cameras, and they both observe the same object, the object must lie at the intersection of the two rays. A visualizing of this process is shown in the figure below(Figure 8).

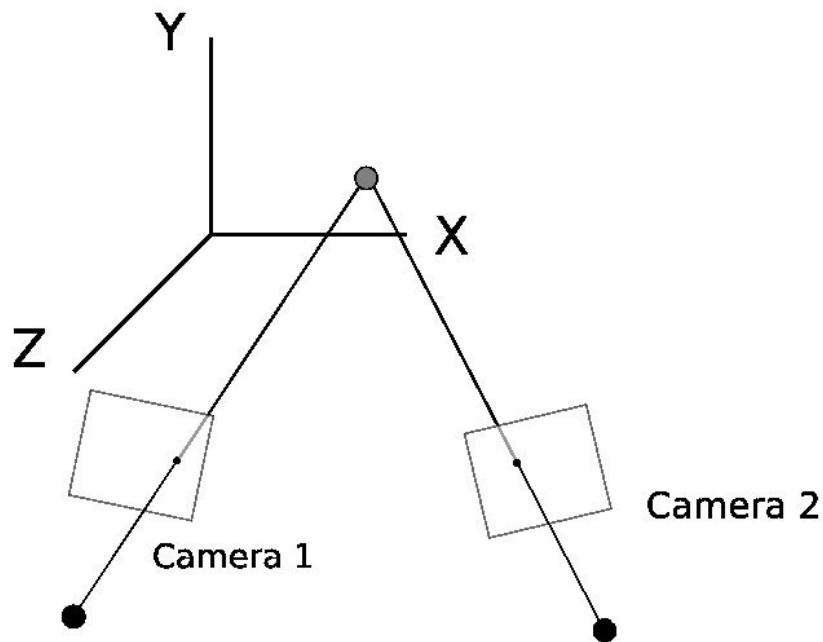


Figure 9: Triangulation Image

## Conclusion and Future Direction

Although our methods are reliable, there are some problems that can occur with tracking the LEDs that our research has not dealt with yet. First, although the size difference between the real LED and the reflection is generally a reliable distinguishing factor, we believe there could be instances when this characteristic may not be enough. Another possible method would be to look at the color spread around the two bright spots, as the color spread around a reflection is always significantly larger than the color spread around the real LED. It is also possible to devise a method to check accuracy and correctness based on the distance between the two LEDs found. Since the LEDs will always be the same distance apart from each other in three-dimensional space, information about the two potential LED positions makes it possible to check the distance between them. If they are not the correct distance apart, the algorithm must search further to obtain a new LED.

Occlusion of the LEDs is another problem that can occur in searching for the LEDs in our experiments. There is a wire that runs down from the ceiling to the head of the rat, and this cord can sometimes block the view of the LED for an instance, making it difficult to track. We are already looking at both of the alternating scanlines to help with this, but another possible resource could be the position of the LED in the image before and after the occlusion occurred. Considering information from these two sources would allow a very accurate estimate of the position of the LED, even in difficult cases.

These problems are of the foremost importance in the future direction of our research. Other explorations of the problem could involve moving the cameras to track the LEDs dynamically rather than our static setup where the cameras remain still. This would require constant calibration of the

cameras and precise control of the camera movement. Once this is accomplished, the final step in our research would be to make all of these steps occur in real-time, rather than waiting for segmentations to be developed after the video has been taken and stored.