Enhanced Quagent control with vision-based Object Recognition

Computer Science 242: Artificial Intelligence
April 27, 2004

Rob Van Dam and Greg Briggs

Abstract: We tested enhancements to our quagent control systems by adding vision-based mapping and object recognition. We modified the ASK RAYS command to act like a laser sensor array. In order to do this, we limited the range to which the sensors could detect anything. Beyond that distance, they return 0. Also, we added a 7% dropout rate to simulate noise that might otherwise exist within a normal sensor array. All of these points that the quagent detects are combined into walls that are formed in such a way as to assure that there were little or no holes in the map. With these walls we can make the quagent maneuver itself (whether through automated controls or natural language commands) fairly efficiently within the quake world.

The ASK RAYS sensor array is not sufficient however for the quagent to maneuver the entire quake world. Since the resultant points are all at a specific height based on the quagent's position (it cannot look up or down), world objects like stairs and slopes appear as walls that cannot be crossed. This confuses the quagent's automatic controls because it cannot recognize that it can in fact walk through these false walls to reach other parts of the map.

Ramps are very difficult to recognize but stairs can more easily be detected by checking for horizontal edges. In order to do this, we first take the image the quagent receives as its view within the quake world and pass it through a median value filter. We then pass the filtered image through a filter using a horizontal edge detecting mask. We tested different median filter sizes and different types of horizontal edge masks to find the combination that appears to be most effective in detecting the horizontal edges within the image.

It is easiest to detect the horizontal edges when we used a quake variable to draw the world with flat colored textures rather than the standard quake textures. However, this is not representative of a real world situation so we attempted to optimize the stair recognition for textured stairs. In particular, it is important to be able to distinguish the horizontal lines of stairs from the lines that might appear on the floor or on the wall due to one of the quake textures. This is best done by recognizing that the horizontal lines of the stairs when viewed in 2 dimensions get closer as the stairs get farther away where as more regular patterns due to quake textures will be evenly spaced.

Background and Motivation

Vision is an important aspect of robotics and can sometimes be one of the only ways to make a robot fully capable of maneuvering any situation. The quagent is very limited in its sensing abilities. The sensor array input allows for basic wall and object detection assuming that the objects are at least as tall as the scan height of the ASK RAYS command. However, objects that

slant towards or away from the quagent are seen only as walls. Stairs then are an instance in which vision is necessary to allow the quagent to freely move anywhere within the quake world.

Methods

Sensor Processing

Raw point data

The "ASK RAYS" result is a list of points, giving their type and relative coordinates. Any points that would indicate an object beyond a certain distance are set to 0 to simulate a sensor array with a limited range. Also, about 7% of the sensor results are deleted to simulate noise in a real world situation. All of these bad results are ignored by the rest of the code.

The current absolute position is assessed before and after the sensor data is taken, using the GPS-like "DO GETWHERE" sensor. By combining the relative position of the points and the absolute position of the bot, the absolute positions of the points can be determined.

If the bot position changed during the scan, then the point data is thrown out, because the absolute coordinates of the information are not clear. This could be a significant disadvantage to a robot where time was very critical, since it could not move and scan at the same time. Also, explosions can move the bot during a scan, as can attempting to scan while standing on a ledge. A more advanced algorithm might perform some sort of interpolation to decide on the positions of these points, if necessary.

These points are stored in an simple array. This storage system, which is O(n) for most retrieval purposes., does not become too cumbersome. This is because n, the number of points, grows slowly during exploration, since the points are soon abstracted into line segments and removed from the array.

Abstracting Line Segments from Points

The second level of abstraction is a line segment representation. The points are analyzed, and where possible, series of points are converted to lines. This significantly reduces the information complexity. A Quagent visiting the start of the initial world ("Outer Base") will easily create over 5000 points while exploring the first room. For comparison, only about 200 line segments are needed to represent the same information. Since it takes only two points to represent a line, the memory space savings is about 12 times.

The line creation algorithm has several components. The primary one works by examining individual points to see if they are already represented by a preexisting line segment. For the remaining points, all point triplets are considered to see if they form a line. This algorithm is O(n^3) where n is the number of points that are not part of line segments. Since points are continually being absorbed into lines, this number n does not grow quickly.

Additionally, since the Quake world contains many walls that are along coordinate axises, the algorithm for handling these triplets studies these computationally-simpler cases first.

Whenever the sensor points form line segments that have endpoints within a short distance, those points are extended to meet so that all "holes" within the map are "sealed." This makes it easier for the quagent to recognize where it can and can't walk. It also makes a much more complete looking map of the quake world.

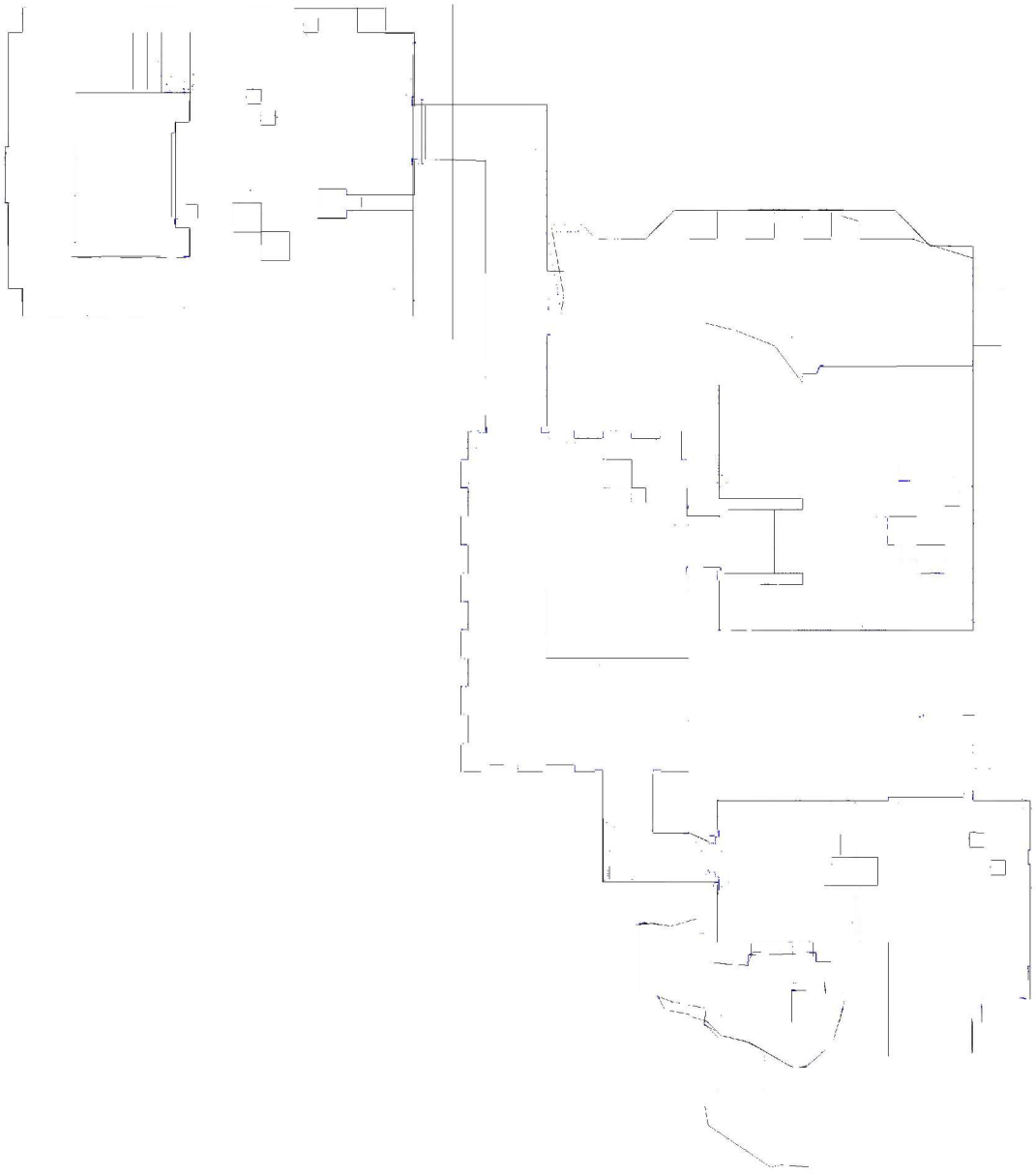


Image Processing

Raw image data

The “LOOK” result is a long string of RGBA values corresponding to a very simple bitmap representation of the current view in quake. This is normally the player’s view but can be set to be the quagent’s view by executing the “CAMERAON” command. By setting the view to the quagent’s perspective, the image data can be used to analyze what the quagent is facing and analyze that image for predefined objects such as stairs that can’t be found using the “ASK RAYS” or “ASK RADIUS” commands.

A screenshot type image is captured each time that the quagent pauses to scan its current surroundings. Unfortunately, the captured image also contains the text output normally found at the top of the window due to the commands being sent by the quagent bot. This could be eliminated by forcing the quagent to wait long enough for the text to scroll away but this would slow the quagent down dramatically.

Sample image with textures

The quake rendering can be altered so that instead of drawing the world with the usual textures, the world can be drawn with non-textured plain colored planes. This has the effect of making the edges on objects, such as at the top of a box or at each step of a stair have a very high contrast that is pretty even across the edge, rather that changing depending on how the textures meet. This is not very representative of the real world but makes the overall task much easier to complete because the later horizontal edge detector will be much more effective and so the stair recognizer will be also.

Sample raw image with out textures

Image Pre-processing

Median Filters

Since the primary focus of the image processing was to discovery stairs, which require recognizing only the horizontal edges, it can be very useful to pre-process an image in such a way as to eliminate a lot of the unnecessary information from the image. One filter that accomplishes the desired task of eliminating unnecessary data is the median filter. This filter evaluates all the points within a small box and determines the median value of all the points within that square and setting the middle point (or one of the corners) of the box to that value. This helps to normalize the image and eliminate a lot of the noise that would otherwise interfere with the later edge detection. Although there are many types of filters that help to eliminate noise in an image, the median filter is known to maintain edges and is therefore very useful for this situation.

The size of the box that is evaluated in order to determine the median value for a given pixel can be altered. Larger boxes blur the image more heavily but require a little more processing time. Larger and larger boxes will eventually distort the image and be counter-productive to edge detection.

The following images give an idea of how the image is altered by median filters of 0, 3, 5, and 9 square pixels. It was found that a median filter of 9 square pixels was very effective in eliminating noise in the image and the resulting horizontal edge detectors were more effective when run on images preprocessed with that median filter. It was, however, found that applying a median filter to an image of a non-textured world actually lowered the effectiveness of the horizontal edge detector. A non-textured image with no median filter is by far the most effective image for detecting the horizontal edges.

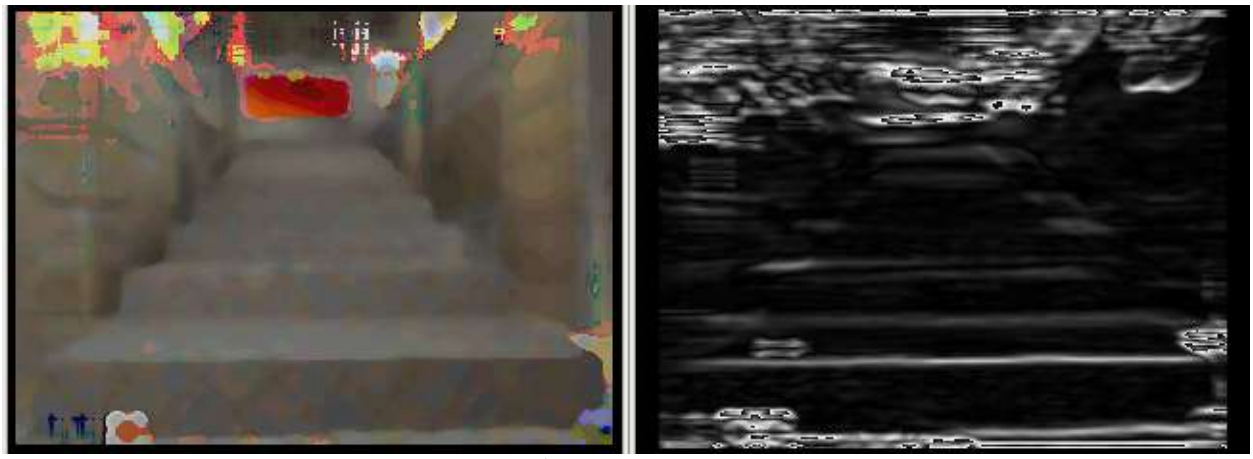
Median size 3

Median size 5

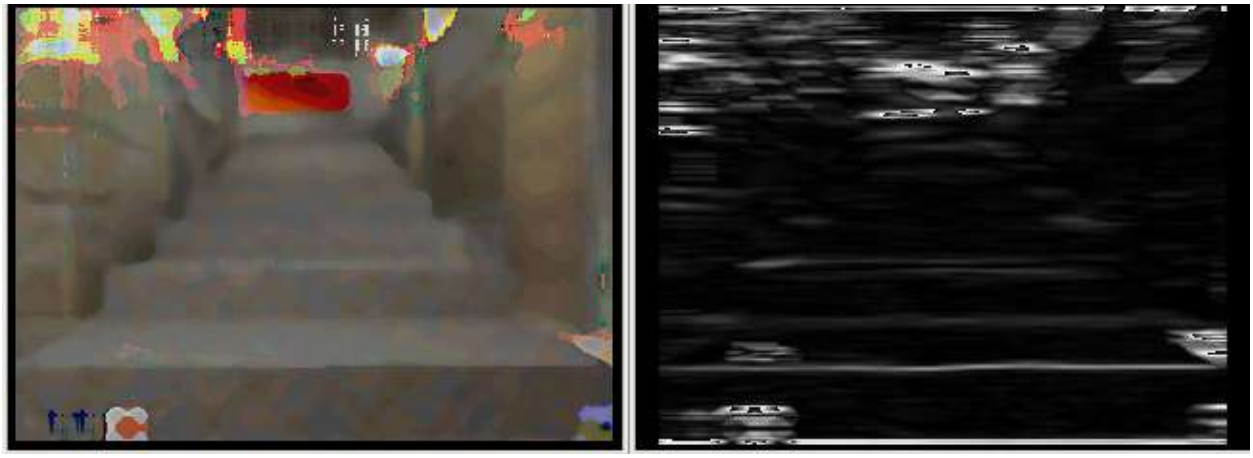
Median size 9

Horizontal Edge masks

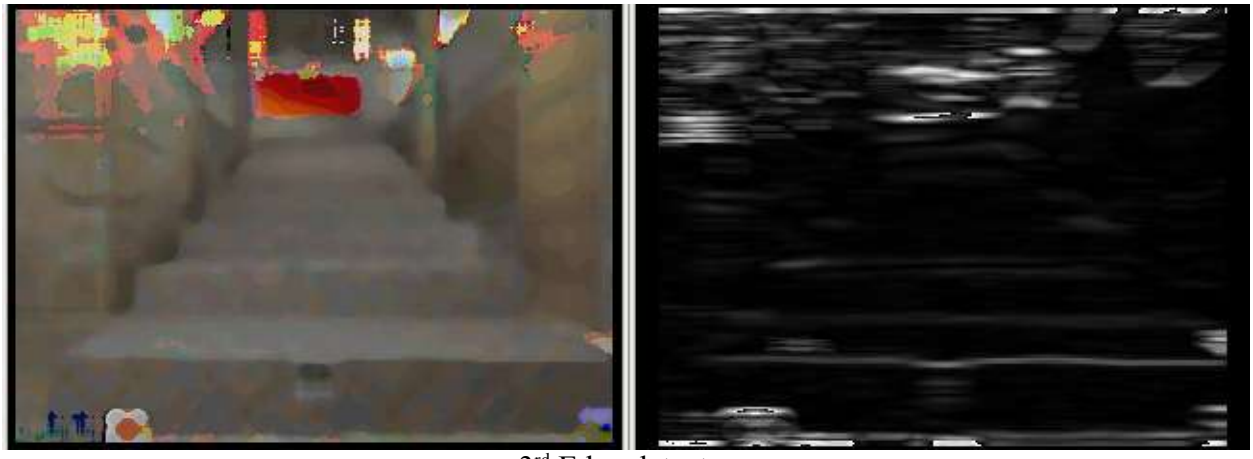
In order to detect the horizontal lines in these preprocessed images, different types of masks were used to convolute the image to emphasize the horizontal lines in the image. Convolution is the process by which a type of cross product of the mask with an equal sized portion of the original image is calculated to obtain a greyscale image. Different types of masks produce very different types of convoluted images. The right kind of mask can be very effective in emphasizing horizontal lines while another mask might emphasize vertical lines. The general idea of these masks is that if there is a strong contrast above and below a given pixel then the cross product will be large and if there is little or no contrast, the cross product will be low. This will then create an image of a line if there is a horizontal edge in the image.



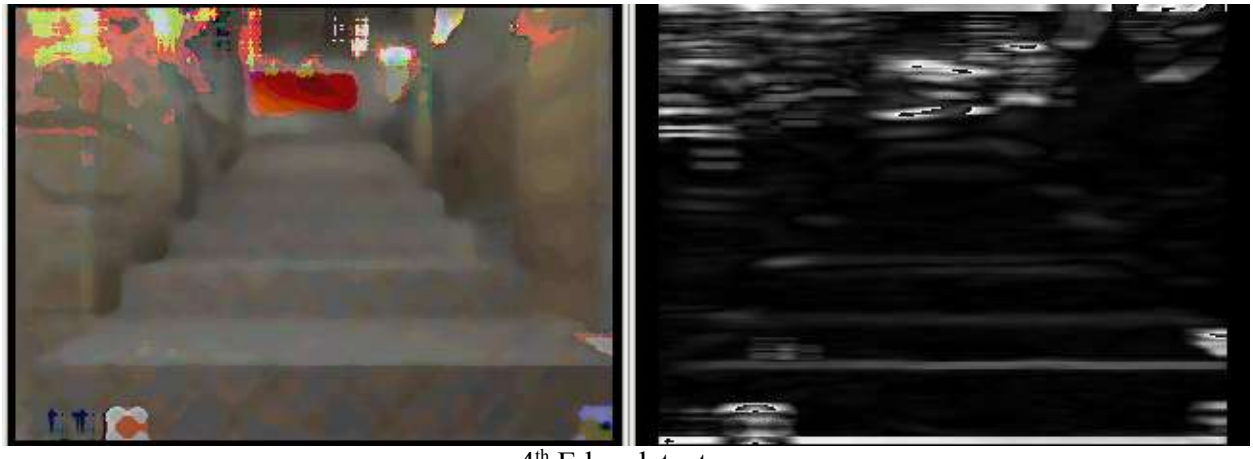
1st Edge detector



2nd Edge detector



3rd Edge detector



4th Edge detector

Stair Recognition [Greg]

One thing that that a sonar ring (or "GET RAYS" in this case) cannot distinguish between is a upward set of stairs from a wall. To address this issue, we desired to use the video camera to attempt to recognize stairs.

The first step of recognition is identifying the horizontal lines in the image, as already explained. The second step is to imagine what the set of stairs might look like. The third step is to confirm or deny one's imagination.

In order for the quagent to hallucinate what a set of stairs might look like, it was programmed with perspective-transformation formulas, which calculate how a 3D world is translated into a 2D image.

The hallucination assumes a typical step size. The hallocination also uses the result from "GET RAYS" to estimate the distance to the base of the stairs. Also, the contrast between steps decreases as the step height approaches eye level. Therefore, only the bottom 2 steps are considered in making the evaluations.

In order to calcululate the perspective projection, it was necessary to determine the focal length of the quagent's camera. By trial and error, we found it to be about the vertical resolution of the camera divided by 1.9.

Since one is not likely to imagine the precise size and location of the steps on a single try, the quagent instead generates 180 different possibilities and considers each one.

One the stairs have been hallucined, the hallucination is compared to the actual horizontal edge map. The root-mean-squares deviation between the hallocinated stairs and the corresponding subsection of the actual edge map is computed. This value gives a confidence level for the presence of stairs, with lower numbers indicating that the presence of stairs is more likely.
One can set a threshold below which to proclaim the presence of stairs.

In testing, we discovered that for when considering stairs far in the distance, there tended to be a high rate of false positives, because the hallucinated stairs essentially wound be a small lined box, which easily matched other objects. To compensate for this, the quagent will refuse to make a judgement on stairs farther away than a threshold distance.

The initial hallucination used 1-pixel thick horizontal lines of length 50 pixels, where the stair
corners were expected. Using this model, with the complete textures and lighting of the first Quake level, approximately 2/3 accuracy was attained (using a selected threshold). It only hallucinated the first 2 steps, since the rest had edges that

were too difficult to see.
The improved hallucination was given the additional knowledge that edges are brighter near the bottom of the stairs, and also used 2-pixel thick lines. With this knowledge, recognition in quake world improved to 80%. See the diagram below for an example of the mask prediction.