KSU Object Recognition

Andrew King, Stephen Stair, Matthew Brubaker, Peter Samland, and Dr David A Gustafson

Computing and Information Sciences, Kansas State University
Manhattan, KS 66506
dag@cis.ksu.edu

Abstract

The Kansas State University entry into the AAAI 2005 robot scavenger hunt consists of a Pioneer P3AT robot running Windows 2000, scalable client/server software architecture, blob-based object recognition, and a Monte-Carlo localization package. As a team, we wanted to build an image recognition system that would be adequate to the tasks that are a part of the scavenger hunt competition.

Client / Server Architecture

For flexibility in development, we built a client / server model (see figure 1) to abstract the Active Media's ARIA API and allow for distributed computation. The server process(es) run on the robot(s). The server keeps track of robot state; such as the velocity of the drive motors, the data being returned from the sonar sensors, the pan/tilt angle of the camera, and the charge of the internal battery.

As the state of the robot changes (either from default behavior coded into the server, or by other clients requesting a state change) the server generates a series of messages which are broadcast to all connected clients to notify them of the state change. All state change requests from clients are served in order of the time they were received.

A client is simply any process that connects to the server. For the scavenger hunt we developed two clients: one to provide robot remote control and visual object identification, the other to perform Monte-Carlo localization. Clients can send requests to the server for information that the broadcast messages don't provide or to alter the running state of the robot. For instance clients can ask for the current image from the camera, request the motors to change velocity, request the camera to tilt and request the camera to pan. Clients have to be written to play nice with other clients; because the server has no facility to resolve any contention between requests (i.e. two clients keep requesting different drive motor velocities.)

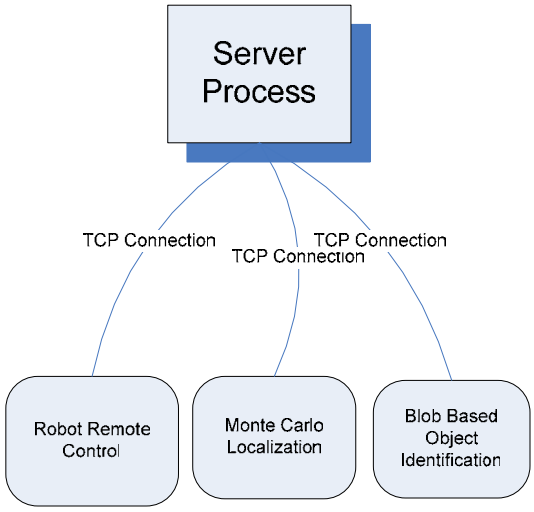

Figure 1. Server process with example remote clients performing various robot related computation

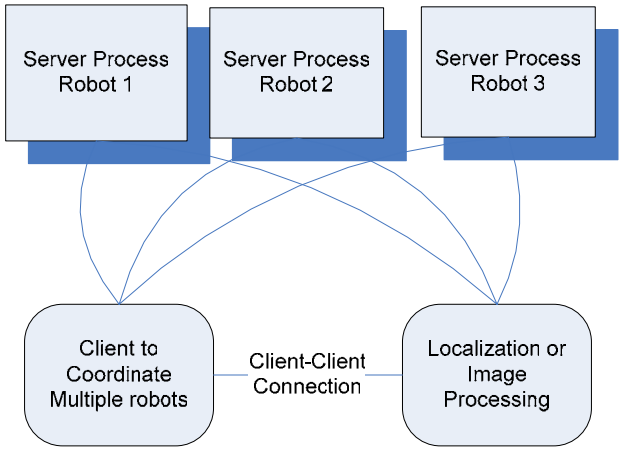

Figure 2. This architecture can scale into a multi robot distributed computing systems quite easily. This diagram shows how different clients can interact with multiple robots.

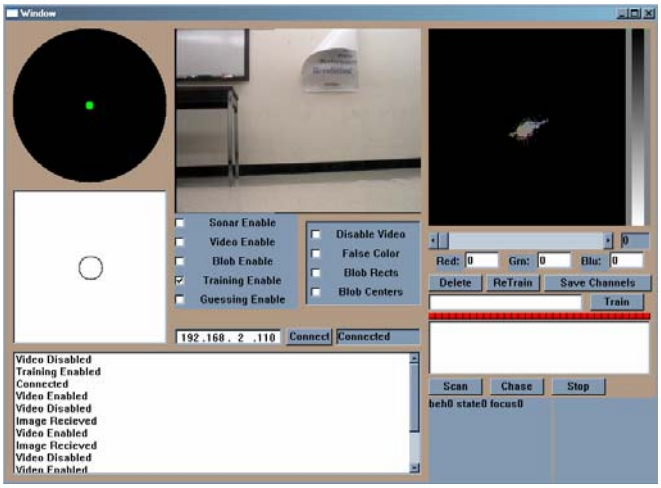

Figure 3. This is our remote control and object recognition client. It connects to the server process on the robot and allows us to perform all scavenger hunt tasks (except localization) from this interface that we run on a laptop.

Blob Based Image Recognition

Because the colors of the objects in the scavenger hunt (lots of fluorescent colors) were not likely to be found in the competition environment we developed a simple blob based object recognition system. Although we started using the ACTS software, we eventually switched and integrated the CORAL Group's Color Machine Vision Project blob software (CMVision for short) into the server process. The server process only performs the actual blobbing computation, blob information about the current camera image can then be retrieved by a client though the client server connection. We felt that blobbing should take place on the robot itself because that would allow us to process blob information on the client without having to transmit images across a potentially overload wireless frequency. The client can also set blobbing parameters (such as what color space ranges should be blobbed on a channel) over the client server connection.

The image recognition client has three purposes:
Calibration: Allows the human operator to calibrate the blobbing system with a YUV color space interface to isolate the color regions that should be blobbed (see figures 4 and 5)
Training: The human operator can present the robot with an example of an object and the software keeps track of the blob statistics associated with that image
Real-time object recognition. Provides instant feedback to the human operator concerning what the robot is tracking and how well it is tracking it.

Calibration allows the operator to choose what part of the YUV color space is to be associated with a channel. For instance, an operator may wish to assign channel 1 to blob colors that are orange with low luminosity (such as the orange cone) then the operator could take another channel to associate with orange of a different luminosity (the orange of the stuffed dinosaur seems to have a higher luminosity component than the cone.) The operator, of course, can calibrate other channels to blob other colors that are present in the objects being searched for.

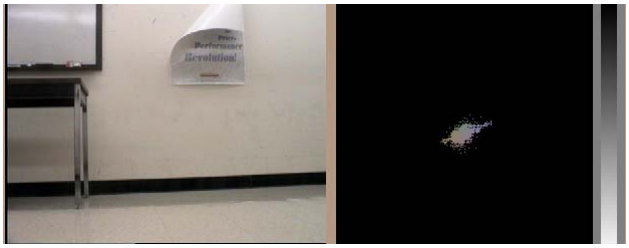

Figure 4. This is a close up of our object recognition client. On the left is the image that is being seen by the robot camera, on the right is the YUV visualization of that image.

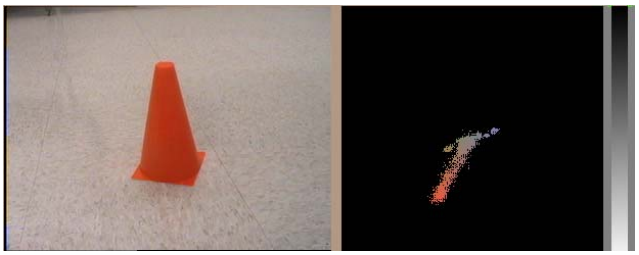

Figure 5. This image has the bright orange cone present, notice how the U and V visualization has grown to show that the particular color of orange is present in the image now.

Object training is a process that takes place after the blobbing system's channels have been calibrated to the appropriate colors (see figure 6 for a screenshot). The training system allows the operator to present the robot with a good example of the object. The blob statistics from the example image are recorded and associated with the object label that the operator provides. In essence, the operator would place a cone in front of the robot, type in the name "cone" and click the "train" button. That training data and its associated label is then stored in a list of known objects.

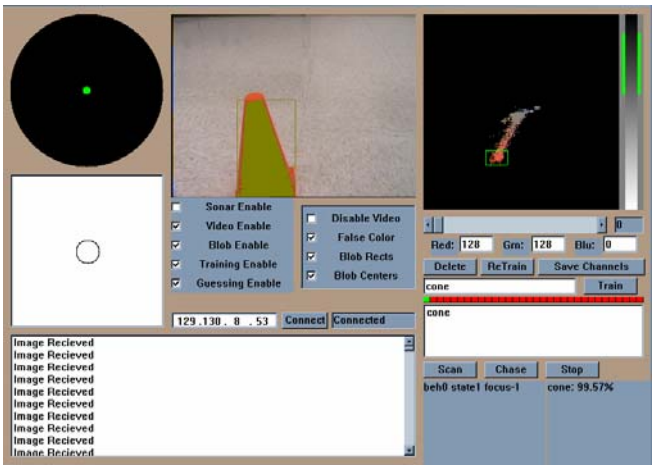

Figure 6. The necessary colors have been calibrated to their channels and the object "cone" has been trained. The system is overlaying false color onto the blobs, drawing a rectangle around the object, and displaying its confidence that the object is present.

Object recognition itself is very simple: CMVision is set up so that it has multiple channels that it is performing blobbing on. Each channel is calibrated to a different region of the YUV color space that could be relevant to the scavenger hunt objects. Each time new blob information is sent to the client the client compares the blob count on each channel against the objects that it has been trained to see according to this algorithm:

```
Function compare(channels trained, channels
current){
  Diff := 0.0
  Count := 0
  For each channel in trained
    Ratio := channel.numBlobs /
current[same_index].numBlobs
    If Ratio > 1.0 Ratio := 1 / Ratio
    Diff += Ratio
    Count++
  End for
Return (1 – (Diff / Count))
```

If the result is above a specified threshold then the object is considered to be found. The center of the object is computed by averaging the center of all the blobs that compose it, and the region of the object in the image is computed by bounding an area with the maximum X and Y edges of all the blobs in the RGB version of the image.

This software can be trained to track multiple objects simultaneously. The client iterates through a list of objects that it has been trained to see and applies the above algorithm to the current image. It will display a value for each object to communicate how confident it is that the object exists in the current image. It will also box all objects it finds on the real image, such as in figure 6.

Summary/Conclusion

Since most of our time was spent building a simple and resilient image recognition system specifically tuned to the hunt objects, the image recognition worked very well, especially in the somewhat poor lighting conditions at the conference. Unfortunately we didn't spend nearly enough time working on other features to aid the robot in its scavenger hunt.

We plan on improving our image recognition system over the coming year. We would like to make the blob "coupling" tighter by examining blob geometry in our object discovery process, develop/refine a crowd tolerant MCL system, and link the MCL and image recognition together so the robot could be "kidnapped" or disorientated and still locate the scavenger hunt objects.