

# Mabel: building a robot designed for human interaction

**Jonathan Schmid, Thomas Kollar, Eric Meisner, Victoria Sweetser, David Feil-Seifer, Chris Brown**  
with Ben Atwood, Jenine Turner, Diana Calarese, Scott Cragg, Habib Chaudhary, Mike Isman

University of Rochester  
Computer Science Department  
PO Box 270226  
Rochester, NY 14627  
brown@cs.rochester.edu, tk010j@mail.rochester.edu

## Abstract

Mabel (the Mobile Table) is a robotic system that can perform waypoint and vision guided navigation, speech generation, speech recognition, natural language understanding, person finding, face finding, and face following. Mabel can interact intelligently with humans in two different settings: food and information serving. The robot's architecture is flexible and easily adaptable to other tasks such as search and rescue. Mabel was the winner of the 2002 robot host event at AAAI's 2002 conference in Edmonton, Alberta.

## Introduction

The American Association for Artificial Intelligence (AAAI) holds robotic competitions each year at its annual conference. At the 2002 conference, the events included food serving, information serving, robot rescue, and robot challenge. The first two events can be grouped into a single competition called robot host.

The food serving event involved serving people refreshments in the break area as quickly and elegantly as possible. This event also required the robots to return to a "refill station" where the robot would retrieve more food and serve it to the patrons. The information serving event involved giving people information about the conference schedule. The robot rescue event involved a robot going into a mock disaster scene, finding human victims, mapping the victims' location, and returning safely out of the scene. Finally, the robot challenge event required a robot to be dropped off at the conference center, navigate its way to the registration desk, register for the conference, navigate to the auditorium, and give a talk on itself.

Mabel (the Mobile Table) is the robotic system developed exclusively by undergraduates at the University of Rochester to compete in the first three events. Mabel serves desserts and information about the conference by performing waypoint and vision guided navigation, speech generation, speech recognition, natural language understanding, person finding, face finding, and face following. Moreover, with some slight adaptation, Mabel can also find victims in a mock disaster scene. A picture of Mabel as used in the robot host events can be seen in Figure 1.

Copyright © 2002, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

The overall design philosophy for Mabel was to integrate human interaction with robotic control. We achieved this goal by using an interactive speech system, a pan/tilt/zoom camera that actively follows faces and a tray that turns to the correct dessert. In this paper, we present the methods used to make Mabel compete in these three events and we will discuss what worked and what we can improve upon in the future. Moreover, we will discuss the methods that we used to integrate the hardware and software into a usable system (Kortnkamp, Bonasso and Murphy, ch.5,9).

## Hardware and Software

Mabel's hardware largely defined the ways that we attacked the problems for the food serving and information serving events. The devices available included: an ActivMedia Pioneer 2 DX mobile robot, an EVI-D30 Sony Pan Tilt Zoom camera, a Logitech webcam, a Pontech board with servos and infrared detectors, a directional microphone, and a 600 MHz PIII IBM laptop.

Moreover, many sensors and effectors were used to attain information from the environment and to perform actions in the environment. The effectors of the robot include a tray that can be turned to one of three positions, a pan/tilt/zoom camera, and the wheel motors on the robot. The hardware used to turn the tray was adapted from a CMU palm pilot robot kit. The sensors of the robot include infrared and sonar range detectors, wheel counters, a pan/tilt/zoom camera, a webcam, and a microphone.

Also, the body design of the robot dictated many of the algorithms that we were able to employ. For example, if the body swayed too much when looking for people, then image differencing techniques would have been less helpful when finding motion in a scene. Figure 1 shows the body configuration used in the competition. It is designed to reduce sway and increase the visibility of the camera mounted atop the robot.

The body consists of the following crucial parts: the pole, the sway retardant bars, a triangular storage area, and the tray holder. The pole holds the camera, the tray, and attaches to the base of the robot. The sway retardant bars connect just under the tray and at the front of the robot. The triangular storage area is connected to the sway bars and the pole. The tray holder connects to the pole and holds infrared sensors, a speaker, the Pontech, servos, and the tray.



Figure 1: Body Design

The libraries and software packages were a crucial part of the rapid development of this project. We could not have developed our own speech recognition, text to speech, or low level image processing routines and still completed the project on time. The libraries and software packages that were used include: the Intel Image Processing Library, the Intel OpenCV library, CMU's Sphinx speech recognition system, IBM's ViaVoice text to speech system, and ActivMedia's ARIA robot control language (IBM Viavoice, web) (Intel OpenCV, web) (ARIA, web).

### Capabilities and Architecture

There are four major applications that will be discussed in this paper: the LearnServer, navigational control, conversational speech interaction, and vision. Figure 2 indicates the structure of the system. The light colored boxes and circles are programs. Arrows between programs means that they are communicating over TCP. Any darker colored diamonds are devices and any lighter colored diamond is a subdevice of another device. An arrow going into a device from a program means that the program controls that device's effectors. An arrow coming from a device means that the program is getting information from the device. The conversational speech and vision applications communicate with the navigational control over TCP.

### LearnServer

We implemented a Java program called the LearnServer for real time parameter adjustment and sensor visualization. It was typically run on a desktop in the lab, with a wireless connection to the mobile robot. The wireless connection allowed for immediate feedback in the behavior of the system.

Robotic behaviors in need of parameter adjustment services acted as clients. Each maintained a tcp/ip connec-

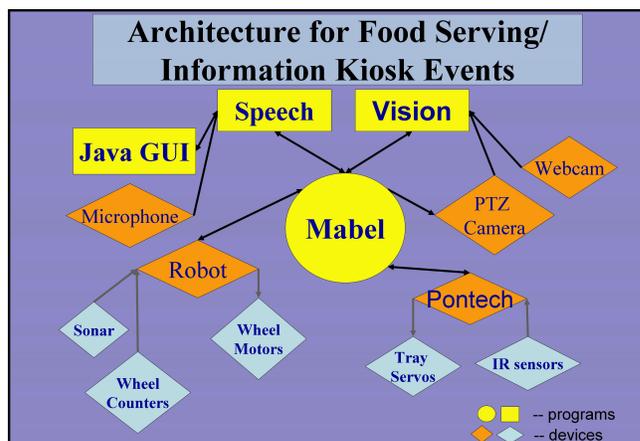


Figure 2: System structure

tion to the server. Parameters were controlled in real time through graphical slider bars. Some behaviors on the robot made use of datagrams to visualize their sensor input off-board in real time.

Some of the most important parameters that were set included the time to wait between visual searches and between checks for food presence on the tray. Another parameter set the number of estimates used to median filter person position. Color bounds of the hue involved in flag finding were also set in this way. We could save all these parameters to a file after adjusting them. The LearnServer is shown in Figure 3.

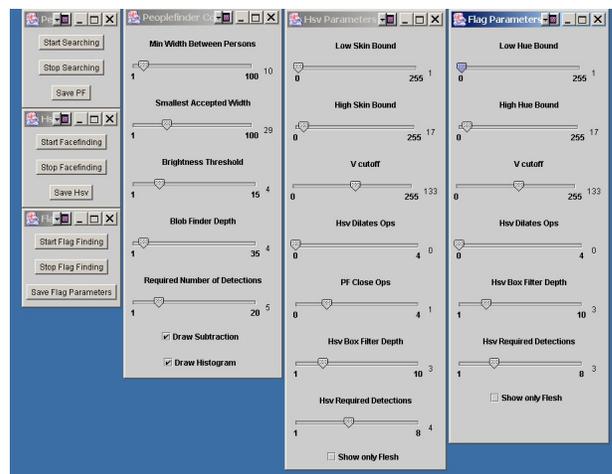


Figure 3: Parameter Adjustment GUI

### Navigational Control

The navigational control is the core application. It produces the behavior of the robot, controls the effectors, takes input from the sensors, and uses the vision and conversational speech applications as subroutines. The navigational control is designed to facilitate rapid food serving over a large area

and to demonstrate intelligent behavior. More generally, the navigational control uses a behavior based paradigm for controlling the movement of the robot (ARIA, web).

There are four distinct states to the navigational control: *refill*, *wandering*, *approaching*, and *servicing*. At the highest level, the *refill* acquires food when the robot's supply is exhausted, the *wandering* accomplishes the ability to acquire new patrons, the *approaching* encounters new patrons, and the *servicing* speaks with patrons, maintains 'eye' contact, and turns the tray (Maxwell et. al., p.2-4).

**Refill** The navigational control application periodically uses vision application to determine the existence of food on the tray of the robot. If the vision application determines that there is enough food, then the robot continues with its *wandering*. If there is not food, then the robot stops *wandering* and tries to go back to the refill station. The refill station is a waypoint (an x and y coordinate) set by a human supervisor of the system at initialization.

A reactive navigation procedure employs dead reckoning to enter the general vicinity of the station. Using the vision application, a visual search for the UR flag is then performed by panning the Sony camera in 20 degree increments from the far left to the far right. If the flag is not found over the entire 180 degree sweep, then the robot rotates 180 degrees and begins the visual search again. If the flag is found, the robot begins moving toward the flag until an obstacle is found (presumably the refill station). The robot remains stationary until it is refilled.

**Wandering** There are two major parts to *wandering*: waypoints and a random walk. The waypoint system keeps the robot in the vicinity of known locations, so that it will not go into areas that might be dangerous for the robot or its clients. The human controller of the robot can set waypoints using the robot's dead reckoning system.

The waypoints are used to define the center of circles with three meter radii. When *wandering*, the robot performs a random walk inside a waypoint's circle. Should the robot go outside the current circle, it will turn around to move back inside the circle to continue its random walk. The robot moves between waypoints at a predetermined interval.

**Approaching** *Approaching* involves two parts: searching and moving toward a subject. At a predetermined time interval, the robot will stop and use the vision application to search for structured motion in a scene. We call this searching. It will search three times: once to the left, once at center, and once to its right. If the robot finds very little or no structured motion then the robot will go back to *wandering*. However, if it finds this structured motion in the scene, then the robot will move toward the direction of that motion. This structured motion that we use here is described in the Person Finder Subsection of Vision.

While moving, the robot will check its sonar and infrared values to see if it is nearing an obstacle (presumably a human). If the robot does not find an obstacle it will continue *wandering*. If it finds an obstacle within a predetermined amount of time and within a predetermined distance, then it presumes that it has found a person. At such a time, the robot

uses a vision application to search for the person's face. The camera then pans and tilts to the person's face in the scene, face tracking (built into the camera) is turned on, the speech program begins, and the robot begins *servicing*.

**Servicing** *Servicing* involves very little from the navigational control. The robot remains stationary and turns the tray when asked by the speech application. When the speech application deems that there is no longer a person nearby or the conversation has ended, the robot resets the camera's position to center and returns to *wandering*.

## Conversational Speech

The conversational speech application is at the core of *servicing*. It interacts with patrons by speaking, recognizing speech, and turning the tray. The conversational speech also decides when there is no longer a patron in the vicinity and parses "*first contact*", a term used to describe the case where a patron wanders up to the robot and begins speaking to it.

There are three major parts to the conversational speech engine: speech generation, *first contact* parsing, and speech recognition. The speech generation uses the IBM ViaVoice Text to Speech system (TTS) (IBM Viavoice, web). Since ViaVoice is a commercial product, we merely configured ViaVoice to give us a *female* voice. Aside from this, TTS was as simple as linking to the right libraries.

*First contact* parsing first takes a base sound level. Then, if anything in the room creates a level above this base level, then the robot presumes a person is waiting to be served. Thus, the robot can go from any state into *servicing* when a person is found by *first contact*.

SPHINX is the speaker independent speech recognition system used in this project (CMU Sphinx, web). SPHINX does well with multiple speakers because it limits itself to particular domain of words and sentences (given in a corpus), not a particular speaker. This sentence corpus contains all possible words and the general sentence structure of the expected input. In the Mabel system it contains almost 400 sample sentences typical of the food serving domain. Moreover, we found SPHINX to provide impressive recognition for speakers of both genders with various accents.

Thus, the speech recognition uses a directional microphone mounted atop the pan/tilt/zoom camera to pick up the audio signal. Since the microphone is directional, much of the background noise is reduced. The audio signal is then amplified by a small guitar amp. The SPHINX Speech Recognition system then parses this audio signal by fitting a phoneme string to the sentence in a pre-determined example corpus. The sentence is then handed off to a natural language understanding subcomponent of the speech application.

Finally, when the natural language understanding comes up with a tag of *< want - strawberry >*, *< want - cracker >*, or *< want - cookie >* (Table 1), the speech application tells the navigational control application to turn the tray the position of the wanted desert. If the natural language understanding comes up with a tag of *< want - gen >*, then the tray will do a full spin to show our selection of food to the patron. For any tag, the speech program uses the TTS

to give a predetermined randomized response. If there has been no input to Sphinx in a given amount of time, then the conversational speech will reset itself and the robot will go back to *wandering*.

Moreover, since natural language understanding is the most complex and the most difficult problem solved by the speech application we discuss it in depth.

**Natural Language Understanding** For the hors d'oeuvre domain, the sentence given by Sphinx is interpreted using Naive Bayesian sentence classification. The training corpus for this classification consists of a list of tagged sentences representative of the domain. The tags represent the sentence category – the basic statement or command intended by the statement. Mabel can recognize a total of nine tags in the hors d'oeuvre domain, including a request for each of three different snacks. These ten tags can be seen in Table 1.

Tag	Intention	Example Sentence
< want – cookie >	request for a cookie	"Can I have a cookie"
< want – cracker >	request for a cracker	"Give me a cracker"
< want – strawberry >	request for a strawberry	"I'll have a strawberry"
< want – gen >	unspecified request	"Sure" "Yes please"
< greet >	greeting	"Hello" "Hi Mabel"
< no >	no	"No thanks" "Nope"
< thank >	thank-you	"Thanks Mabel"
< end >	end conversation	"Goodbye" "Bye Mabel"
< want – list >	request for menu list	"What do you have"

Table 1: Tags used in Speech Recognition

When processing a new sentence, we wish to determine which of the possible tags it is most similar to. There are three basic factors we will consider: the probability of each tag occurring, the probability of each tag given all of the single words (unigrams), and the probability of each tag given all of the word pairs (bigrams). Each of these factors can be estimated from the training corpus, and the probability of a tag, given a new sentence, is determined by the product of all three factors. The probability of each tag,  $P(tag)$  is straightforward:  $\frac{count(tag)}{N}$  where  $N$  is the total number of tagged sentences in the training corpus. The  $n$ -gram factors are slightly more complicated.

Because calculating the probability of each tag for a new sentence involves multiplication, we wish to avoid any factors of 0. For instance, if a user utters the sentence "Hi, I'd like a strawberry," we would like this sentence to be tagged with < want – strawberry >. If we look at Table 1, the unigram "hi" never appears in the context of a < want – strawberry > tag, and thus has a probability of 0. So even though the rest of the sentence will strongly indicate the desired tag, the 0 factor clobbers the probability, forcing an outcome of 0.

The solution to this is data smoothing. The simplest way of calculating  $P(tag|word)$  is  $\frac{count(wordintag)}{count(wordincorpus)}$ . To avoid 0 probabilities, such as  $P(< want – strawberry > | "hi")$  a small constant ( $\lambda$ ) is added to every possible factor. In order to ensure that we have a valid probability distribution function, we add the vocabulary size (number of unique words

Mabel: Hello, my name is Mabel. Would you like an hors d'oeuvre ?
User: Yes, I'll have a strawberry
Mabel: (tags sentence as < want – strawberry > and rotates tray to strawberry section) Here is a strawberry. Please enjoy it. Can I offer you anything else?
User: No thanks.
Mabel: (tags sentence as < no >) Well, think of me later if you change your mind. Goodbye.

Figure 4: Sample Conversation 1

in corpus) times  $\lambda$  to the denominator, giving:

$$P(tag|word) = \frac{count(wordintag) + \lambda}{count(wordincorpus) + (vocab * \lambda)}$$

The unigram probability of a tag given a word string  $W_{1,N}$  is the product of each unigram probability, or,  $\prod_{i=1,N}(tag|word_i)$ . Likewise, the bigram probability of a tag given the same word string is  $\prod_{i=1,N}(tag|word_{i-1}, word_i)$ . Ultimately, the probability for each tag for a new sentence is calculated by

$$P(tag) * \prod_{i=1,N}(tag|word_i) * \prod_{i=1,N}(tag|word_{i-1}, word_i)$$

The greater the vocabulary of a particular domain, the smaller the probabilities are for individual words. In human speech, however, a single object (snack) or idea (want) can be expressed with many different words. For instance, a chocolate chip cookie might be referred to as "cookie," "chocolate chip cookie," "chips aho cookie," etc. We can generate stronger probabilities if these are all condensed into a single probability for "cookie." The simplest way of producing this was implementing a simple substitution filter as the sentences were read in, both during training and during the processing of new sentences.

The tags were chosen to be as broad as possible, representing the smallest number of meaningful distinctions possible in the user's intention. While there are clearly many more categories of requests and statements in a possible conversation, most can be either considered a subcategory of an existing tag, or disregarded. This approach proved beneficial when handling incorrectly tagged sentences. Mabel was more likely to do the right thing (or something similar) if there were fewer options to choose from. Consider the sample dialogues in Figure 4 and Figure 5. In Figure 4, the interaction ran perfectly. All of the user's utterances were correctly tagged, allowing Mabel to respond appropriately. However, in Figure 5, the sentence started with an incorrectly tagged utterance. However, this did not create much of a problem. Most of the time, guests seemed to interpret this as a "suggestion" on Mabel's part, giving Mabel an accidental semblance of personality.

## Vision

The vision program uses the Intel Image Processing library and the Intel Open Computer Vision library to create four major features: a food counter, a person finder, a face finder,

Mabel: Hello, my name is Mabel. Would you like an hors d'oeuvre ?
User: Sure
Mabel: (Incorrectly tags sentence as <i>&lt; want - cracker &gt;</i> and rotates tray to cracker section) OK, here is a cracker. Please enjoy it. Would you like anything else?

Figure 5: Sample Conversation 2

and a flag finder (Maxwell et. al., p.6-11). Each feature is implemented as a reactive visual behavior, written as a C++ class. Each of these classes executes its own image preprocessing, and gathers symbolic information about the scene over a series of frames.

At the core of the face finder, the person finder, and the flag finder is a formalism that we call a Fuzzy Spike. Segmentation and labeling of blobs is achieved through this formalism. We first define a Spike to be a continuous one-dimensional region of non-zero histogram values from a one-dimensional histogram. A Fuzzy Spike is also a group of histogram values, but intermittent regions of zeros are allowed. When labeling the clusters of non-zero values, we set a limit on the number of continuous zeros allowed between patches of signal within a single spike. This limit is set in the person finder to a value which strikes a balance between the possible over-segmentation of a single person into more than one blob hypothesis, and the misclassification of many people as one.

**Food Counter** Determining whether food remains on the tray is accomplished visually. The black cloth covering of the tray fills the field of view of a webcam. A brightness threshold serves to segment the regions of the image containing food. The number of food pixels visible to the camera is summed and compared to a minimum value corresponding to one remaining dark hued cookie. When the time-filtered white pixel count falls below this threshold for several consecutive frames, the robot will return to the base station to refill.

**Person Finder** The Person-Finder behavior is designed to detect an approach vector to the closest person in the scene. It takes as input each image of a live stream captured from the EVI-D30 Sony Pan Tilt Zoom camera. Image differencing techniques yield a motion field that is used in finding the closest person. A bounding box represents the current hypothesis for the location of the closest person in the scene, as in Figure 6. The center of that rectangle is used in estimating a robot-egocentric approach vector to the person.

The behavior employs a two frame pre-roll period. A person detection signal is only available after the third frame in the stream has been processed. During the pre-roll period, the behavior is storing grayscale copies of each image from the stream. When three grayscale images are available, the behavior enters a search mode.

In the search mode, the behavior uses grayscale copies of the current frame and the last two frames stored from the stream. It subtracts all pair-wise permutations of the

three frames into a set of six difference images. Since the images were unsigned, all six permutations of subtraction were required because negative results of pixel subtraction underflowed to 0. This avoided converting the format of the three scene images. The difference images are processed with a single iteration of a standard erode operation and then summed together. The resulting sum image is thresholded into a binary image.

Fuzzy Spike detection is applied to the x-axis of the binary motion image. The extents of the largest blob in the x-direction are used for the sides of the rectangle. A vertical search for the extents of the person is conducted only between the known x-bounds. When the vertical and horizontal extent of the person has been found at a particular frame, the extents are median filtered with those of previous frames. The number of past estimates that are used is adjustable. A good estimate of true person location is usually available after about 5 search frames.

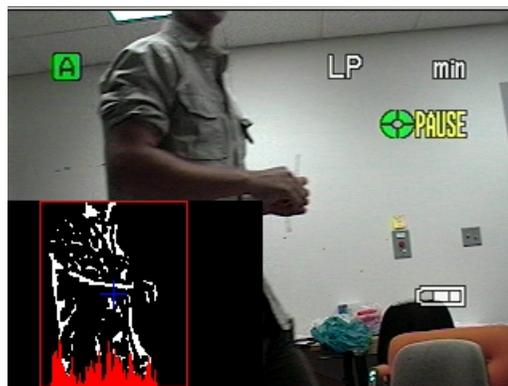


Figure 6: Person Finder

When the control system requests a vector, the x coordinate of the center of the current bounding box is used to compute an approach vector. In our system, this vector is in the range:  $(-22, 22)$  degrees. This angular range depends on the field of view of the camera. A visualization of the person-finding algorithm can be seen in Figure 6.

**Face Finder** After the robot has approached a patron, we use a facefinding behavior to find a face in the scene. Thus, we presume that we are near a person before running the face finder. This procedure takes advantage of the camera's proximity to the patron's face by looking for the largest flesh colored blob that displays motion in the scene.

To do this, it uses a binary map of flesh colored pixels produced by determining if each pixel lies within a precalibrated hue range. This flesh tone map is then combined using a logical image operation of AND with a binary map of motion pixels found by frame differencing. Blob finding via fuzzy spikes is then run on the combined flesh and motion image map. If the resulting largest detected blob is over a size threshold, then it is presumed that there is a face in the scene. Thus, the center of the face in the image is computed and these coordinates are used to pan and tilt the camera so that the camera is centered on the patron's face.

After the camera moves to the patron's face, the camera's onboard face tracking can be initialized through a serial interface. This results in the camera metering the patron's face and following it with a saccade chase mode. Camera motion occurs quickly enough after locating the face to ensure that the subject will still remain centered in the frame.

**Flag Finder** A University of Rochester blue and yellow flag designated the food refilling station. The first step in localizing the flag was to find blobs of blue hue. Thresholds for the minimum required width and height of the blob were set according to the apparent size of the flag in the camera's field of view when the flag was at an estimated maximum detectable distance from the robot. These thresholds were used to eliminate from consideration any smaller blobs.

The largest continuous blue area that met the requirements of the width and height thresholds was then used as the boundary of a 2d region. The pattern of text and graphics on the flag appear in a uniform yellow hue. Therefore, to distinguish the UR flag from other large blue areas in the environment, we counted the yellow hued pixels within each candidate flag blob, see Figure 7. The yellow pixels in an image of the UR flag make up a significant percentage of its area. Therefore we set a threshold for the minimum percentage of yellow pixels required within the 2d blob candidate area. If this ratio lay above the threshold, then the detector would yield a positive result. An absolute heading to the flag could then be used to approach it and wait for refilling.

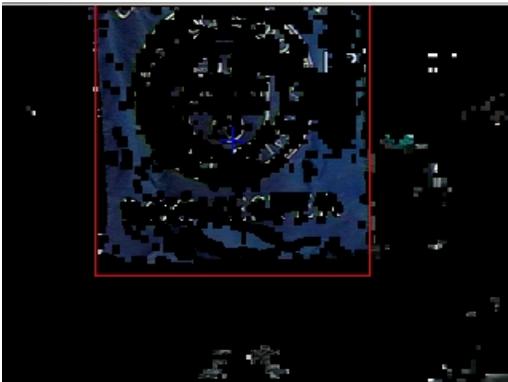


Figure 7: Flag Finder: the Rochester banner is a blue square with a circular yellow and blue college seal and yellow lettering underneath. Blue pixels appear gray in this image, all other colors are lighter or darker.

The blue region was often sufficient for finding the flag in a room that lacked blue walls or posters. We found the most frequent non-flag source of blue to be from people's shirts. We ignored blue shirts by ANDing all non-moving pixels with blue pixels before detecting the blobs. Very few false negatives were encountered, but the detector was unreasonably fragile to the presence of non-moving blue in the environment. With the addition of the yellow pixel threshold we achieved almost no false positive detections, but occasionally missed the flag when it appeared not wide enough or

lacked enough apparent yellow pixels. The problems were due to a lack of hue-invariance to the changing light conditions, since color information yielded from inexpensive USB frame grabbers is highly compressed. We implemented a text-reading system to read the word ROCHESTER at the bottom of the flag, but did not use it in the vision system due to its higher rate of false negatives.

### Adapting Mabel for Information Kiosk

Almost all of the components used in the food serving event were ported over to the Information Kiosk event very easily. In fact, the navigational control and vision parts of the system were unchanged.

The conversational speech, however, had to be drastically modified for the larger domain of words that people could use. Ideally, the robot would have taken speech input from the microphone and tried to give an intelligent response based on this input. When that failed, patrons would have used the GUI as a reliable backup mode of communication. However, due to time constraints, the interface was limited to a Graphical User Interface written in JAVA.

The JAVA GUI allows patrons to type natural language queries about conference events. In response to a query, the GUI displays a list of conference events that match the patron's query. It also displays conversational and visual responses to their query.

However, understanding these natural language queries provides a new set of difficulties. In the hors d'oeuvre domain, conversation was handled entirely by sentence classification. An input sentence would be tagged as the most probable of the nine possible tags, and appropriate behavior generated based on the output tag. The most obvious difference in the information kiosk domain is the dramatic increase in vocabulary. This new vocabulary includes speaker names, locations, topical keywords, event titles, times, and dates. This sheer increase alone deals an immediate blow to the accuracy of natural language understanding.

More problematically, the Bayes Net approach is not equipped to deal with the amount of precise data presented in this type of query. In the sentence "When is Robin Murphy's speech?" it is clear that the speaker is asking a question relating to Robin Murphy. The best a Bayes Net can do is add some weight to the probability that the sentence concerns Robin Murphy. With the number of speakers, locations, times, and events possible to reference, sentence classification is clearly insufficient.

In order to remedy this, we began by filtering out any known keywords from a sentence and replacing them with their category. For instance, the above sentence would become "When is [speaker] speech?" Any such replacements are kept track of. This is done using a battery of regular expressions, with patterns generated from a formatted schedule of conference events. Next, the sentence classifier is tagged. For this, we used the same sentence classification scripts, trained on a corpus of tagged, filtered sentences. With the tag, we attempted to identify what piece of information the user was requesting; for example:

< <i>time</i> >	"When does [speaker] talk"
< <i>loc</i> >	"Where is the workshop on [event-topic]"
< <i>speaker</i> >	"Who is leading the panel discussion in [loc] on [day]"

Combining the tag with any found keywords, we constructed a string of the form [*< time >*, *speaker*="Robin Murphy"]. The complexity can range from a standalone tag with no keyword values, to multiple values for several different keywords. For instance, the sentence "Where is the meaning negotiation workshop on Sunday with Kendall Lister and Paolo Bouquet?" would be processed into [*< loc >*, *speaker*="kendal lister+paolo bouquet", *topic*="meaning negotiation", *date*="jul 28"]. This string was then used in a database lookup to find all matching events. The events that were returned would then be passed to the GUI and displayed graphically.

### Adapting Mabel for Robot Rescue

Mabel was developed primarily to compete in AAI's robot host competition. However Mabel's architecture was flexible enough to allow her to compete in the robot rescue event with only a few modifications. In robot rescue, Mabel was partly teleoperated, but could also operate autonomously. The robot rescue system was developed in parallel and shares many components with the robot host navigation system. It consists of three core components: a navigational control component, a person finding vision component and a LearnServer interface.

**LearnServer** The physical state of the robot is sent over TCP to a Graphical User Interface (GUI) called the LearnServer. The LearnServer provides the user with a visual model of the current robot state. The user can zoom in and out on the robot and see a graph of new and old sonar values. A user of the system can, when zoomed out, see a global view of the sonar values. However, when zoomed in, a user may see more detail about the current state of the robot. The user can also see the location and angle of the robot relative to its initial state. An example of the robot plot can be seen in Figure 8. The LearnServer also constructs a map of the disaster scene from this sensor data. An example of this map can be seen in Figure 9.

**Vision** Vision allows the teleoperator to make decisions for the robot based on visual feedback. Thus, the frames are sent to the teleoperator's computer by way of a remote desktop manager. We rely on wireless ethernet technology for this capability.

The vision component uses the same person finding utility (above) used in the robot host system. This person finder gives us a way to search for motion in the scene. Presumably motion can be correlated to a victim when the disaster scene is fairly static.

**Navigational Control** The purpose of the navigational control component is to allow a single user to control the speed of the robot, the direction of the robot, and the pan, tilt and zoom of the camera. It also allows the user to switch from teleoperation to autonomous navigation. The autonomous navigation or wander behavior allows the robot

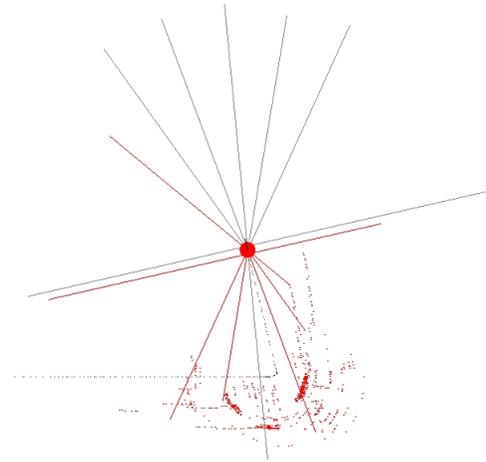


Figure 8: Robot plot, showing recent sonar values



Figure 9: LearnServer Map made from sonar values

to explore space without user control. This behavior causes the robot to move forward until it discovers that its path is blocked based on the data from its sonar array, at which time it changes its heading accordingly.

The *go home* behavior causes the robot to return to its point of origin. The robot's point of origin is calculated based on the data stored from its wheel counters. The *go home* behavior is invoked either by key stroke, or automatically if the user fails to issue a command to the robot in a period of two minutes. This feature is designed to be a fail safe for the (likely) event that the operator loses contact with the robot. In this case, the robot returns to its point of origin so that it can be reset or communication can be re-established.

The navigational control also has an interface to the vision application's person finder. The person finding utility may be started and stopped with keystrokes to the navigational control. When the vision application finds a person, the navigational control calculates a vector along which the victim may be located, based on the pan tilt and zoom of the camera, the reading from relevant sonar, and the location and orientation of the robot. This data is passed on to the LearnServer and displayed graphically. The location of a victim can also be plotted manually without the use of the person finding component.

## Discussion

During the summer of 2002, there were six people working on the project full time. Thomas Kollar focused on the navigational control of the robot for the host events. Jon Schmid focused on the vision components of the robot. Three people (David Feil-Seifer, Victoria Sweetser and Jenine Turner) worked exclusively on the conversational speech component and the interactive kiosk engine. Eric Meisner worked on map visualization and modifying components for the robot rescue competition.

All of the vision and navigational control was developed in a windows lab and with a common CVS tree, while the speech and information kiosk engines were developed in a separate unix lab. CVS proved to be an exceptionally valuable tool, and allowed for easier integration of the vision and navigational control. Before using CVS, we could not achieve a successful vision integration, despite many attempts.

Several times during the project, large parts of the system were rewritten. The navigational control, as of the beginning of the summer, existed in a C robot control language named Ayllu. Because of the structure of Ayllu, keyboard input, TCP connections, and system core existed as three separate programs. Message passing between these components made the system increasingly difficult to understand and maintain. Thus, the decision was made to rewrite the navigational control as a single program in the brand new cross-platform C++ robot control language named Aria. Its object oriented nature made the code more readable and maintainable.

Aria's cross-platform abstraction of TCP allowed for the creation of a shared network library. This is important because integration symbols could be shared between commu-

nicating programs. This, and the use of CVS, had the effect of substantially reducing integration time.

Parts of the vision system were rewritten during development as well. A memory offset error in the person finder program was traced to complications in the standard template library's vector class. The component was rewritten using standard template library's list class. Henceforth, the component remained stable and showed better performance due to faster sorting algorithms which could be applied to the list from the STL.

## Future Work

In the Fall of 2002, the Robot Research Team class is continuing at the University of Rochester. Many new and exciting features are being added to the robotic system this semester.

Schmoozing is one of the features currently in the planning stage. Schmoozing has as its goal to interact with a person while trying to guess what information that person might be interested in knowing. This will build off of the Bayes Net and the keyword replacement used in the robot host events. One of the subgoals of schmoozing is to remember the state of the conversation as well.

The ability to acquire a patron's name from a nametag is another feature that we propose to add to the Mabel system. The most ambitious version of this goal includes reading a general nametag with only a few examples of what the nametag might look like. This problem creates many difficulties. First, we must find the nametag, and visually segment the name from this area. Then, we must segment each character and perform some pattern matching on these letters to identify them. Much work has already been done towards this goal by Jon Schmid and Thomas Kollar. With this feature we can attain the names of patrons and look up relevant information about the patrons via Google or some other large store of information. Such information could be incredibly useful for schmoozing.

Stereo Vision is also another feature that is currently being implemented for possible use in the Mabel system. Possible applications of this could be path planning, obstacle avoidance and gesture recognition.

## Acknowledgements

Many thanks to George Feguson, James Allen, Marty Guenther, and the Computer Science Department at the University of Rochester for their endless support and wisdom. Thanks to Dr. Paul Chou and his colleagues at the IBM Watson Research Center for the loan of a laptop. This project was funded in part by a grant from the National Science Foundation: EIA-0080124 and REU supplement.

## References

- ActivMedia ARIA robot control language, <http://robots.activmedia.com/aria/>
- CMU Sphinx, <http://fife.speech.cs.cmu.edu/sphinx/>
- IBM ViaVoice SDK for Windows, <http://www-3.ibm.com/software/speech/dev/>

Intel Open Source Computer Vision Library,  
<http://www.intel.com/research/mrl/research/opencv/>

Kortenkamp, D., Bonasso, R., and Murphy, R. eds.  
1998. *AI and Mobile Robots*. Menlo Park, Calif.: AAAI  
press/MIT press.

Maxwell, B., Meeden, L., Addo, N., Brown, L., Dickson,  
P., Ng, J., Olshfski, S., Silk, E., and Wales, J. 1999. Al-  
fred: The Robot Waiter Who Remembers You. Proceedings  
AAAI Workshop on Robotics. Menlo Park, Calif.: AAAI  
Press.