# Say Cheese! Experiences with a Robot Photographer

*Zachary Byers, Michael Dixon, William D. Smart, and Cindy M. Grimm*

■ We have developed an autonomous robot system that takes well-composed photographs of people at social events, such as weddings and conference receptions. The robot, Lewis, navigates through the environment, opportunistically taking photographs of people. In this article, we outline the overall architecture of the system and describe how the various components interrelate. We also describe our experiences deploying the robot photographer at a number of real-world events.

In this article, we describe our experiences with an autonomous photographic system mounted on a mobile robot. The robot navigates around at social events, such as weddings and conference receptions, opportunistically taking photographs of the attendees. The system is capable of operating in unaltered environments and has been deployed at a number of real-world events.

This article gives an overview of the entire robot photographer system, and provides details of the architecture underlying the implementation. We relate our experiences deploying the system in several environments, including a scientific conference and an actual wedding, and discuss how it performed. We evaluate the quality of the photographs taken, and discuss opportunities for improvement.

Our system is implemented with two digital cameras (one still and one video), mounted on an iRobot B21r mobile robot platform. The robot, a bright red cylinder approximately two feet in diameter, stands slightly over four feet tall. The cameras are mounted on top of the robot on a Directed Perception pan/tilt unit. All computation is done onboard, on a PentiumIII 800 megahertz system. The only sensors used for this project are the cameras and a laser rangefinder, which gives 180 radial distance measurements over the front 180 degrees of the robot, in a plane approximately one foot above the floor. The robot communicates with a remote workstation, where photographs can be displayed using a wireless Ethernet link. The robot is shown in figure 1.

Generally, the system works as follows. The robot navigates around the room, continually looking for "good" photographic opportunities. A face-detection system that fuses data from a video camera and the laser range finder locates the position of faces in the scene. These faces are then analyzed by a composition system, based on a few simple rules from photography, and the "optimal" framing of the scene is determined. The camera then pans, tilts and zooms in an attempt to match this framing, and the photograph is taken.

In the remainder of this article, we will discuss our motivation for undertaking this project and describe the various aspects of the system. We then describe some of the system's major deployments, and show examples of the photographs that the system took. Finally, we offer some conclusions based on our experiences, attempt to evaluate the performance of the current system, and suggest future directions for research.

## Motivation

Why robot photography? Our primary research interests are in the areas of long-term autonomy, autonomous navigation, and robot-human interaction. Our robot photographer project started as a framework within which we could do that research. It was also designed to be appealing to undergraduates,

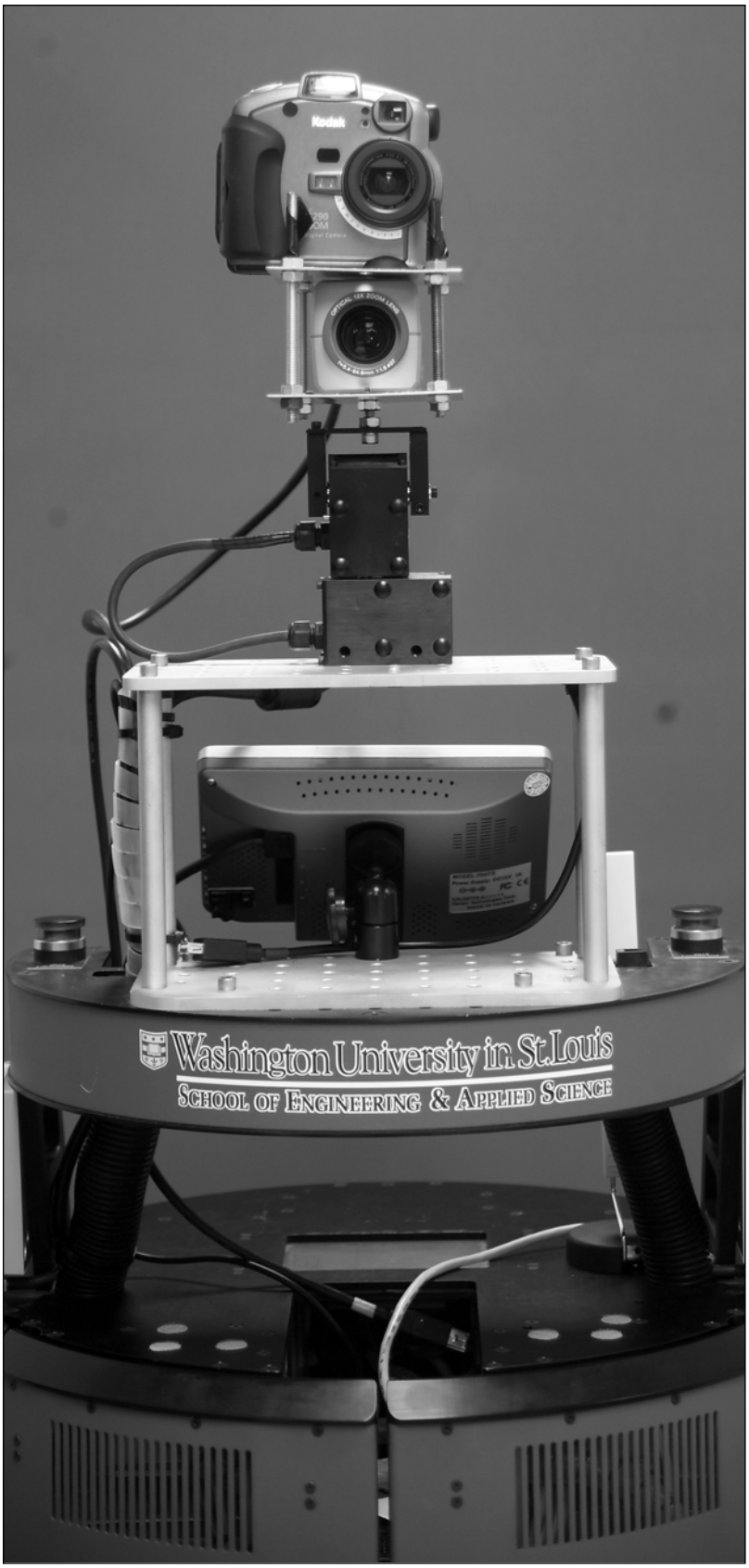


*Figure 1. Lewis, the Robot Photographer.*

thus encouraging them to get involved in research. Automated photography is a good application choice, because it incorporates all of the basic problems of mobile robotics (such as localization, navigation, path planning, etc.), is easily accessible to the general public (everyone knows what a photographer does), and has a multidisciplinary element (for example, how does one automate the skill of photograph composition?).

Because the robot photographer concept is easily understood by the public, it is an excellent umbrella under which to study human robot interaction. Members of the public who have seen the system have responded very positively to it and have been quite willing to interact with the robot. Since the application is accessible to people without technical knowledge of robotics and computer science, the interactions that people have with the system tend to be very natural.

Our original goals were to create a system that was able to autonomously navigate crowded rooms, taking candid, well-composed pictures of people. The intent was to have an automated event photographer that would take pictures of people casually interacting with one other, rather than standard "mug shot" types of photographs.

We should note that there is significant room for improvement in the current system. Many of the algorithms are quite basic, and system performance would improve if the algorithms were enhanced or replaced. However, we believe it is useful to present the system in its current state because it illustrates the overall level of performance that can be achieved with very simple components working together. When working on a mobile robot, there is also utility in using algorithms that are as computationally simple as possible. Computation takes power and contributes to significantly shorter battery lifetimes. We are, therefore, interested in the simplest algorithm that we can get away with, even if performance is not optimal.

Now that the basic system has been deployed, we have found it to be a good platform for general mobile robotics research. The system is purposefully designed to be modular, so that more advanced algorithms can be easily added and evaluated. It also provides a vehicle for research into areas not specifically tied to the photography project, such as navigation and path planning. Our efforts are currently directed at evaluating the system, and the effects that adding more sophisticated algorithms will have, in terms of overall performance, battery life, responsiveness, and so on.

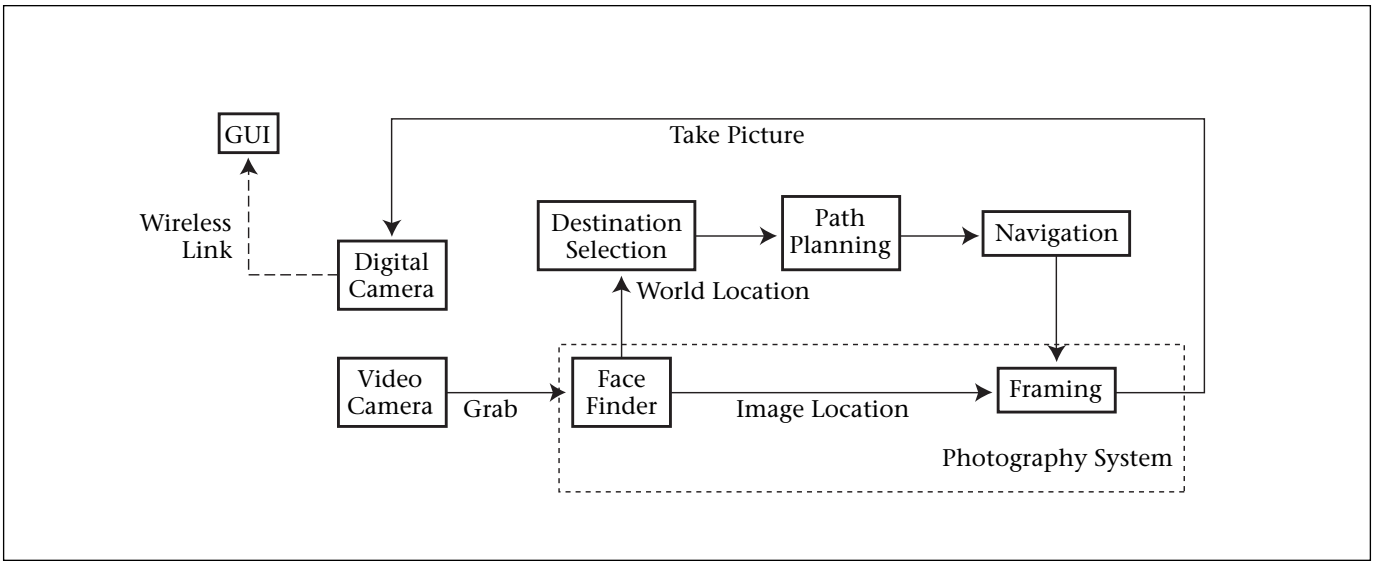


*Figure 2. An Overview of the Photography System Architecture.*

## Robot Photography

We have broken the task of photography into the following sequential steps: locating potential subjects, selecting a photographic opportunity, navigating to the opportunity, framing and taking a shot, and displaying the final photograph. These steps are summarized in figure 2.

### Locating Potential Subjects

To locate potential subjects, we search for faces in the images from the video camera. A common strategy in face detection is to use skin color to help isolate regions as potential faces. Because skin occupies an easily definable region in color space (Forsyth and Fleck 1999), we are able to define a lookup table, which maps from a color's chromaticity to its likelihood of being skin. Applying this function to each pixel of an image allows us to construct a binary image representing each pixel as either skin or nonskin. We then segment this image into contiguous regions, with each region representing a potential face.

The next step is to determine the size and relative location in space of the object associated with each skin region in the image. The pixel location of a region can be translated into a ray extending from the camera through the center of the object. The ray's projection onto the ground plane can then be associated with one of the 180 rays of laser data. If we make the assumption that all perceived objects extend to the floor (as is usually the case with the bodies associated with faces), then the laser reading will tell us the horizontal distance to the object. Knowing this distance allows us to calculate the position in space and the absolute size of each object.

All regions whose geometric and spatial properties fall within the range of expected face sizes and heights are classified as faces.

### Selecting a Photographic Opportunity

The relative positions of potential photographic subjects are used to calculate the location of the best photographic opportunity. We discretize the floor plane into a grid with squares 20 centimeters on a side. For each grid square within a given range of the robot, we calculate the value of an objective function that measures the potential quality of a photograph taken from that position. The function is initially zero everywhere, and is updated according to the distance from the subject, occlusion, bisector, movement, and reachability.

*Distance from subject* (figure 3a). The ideal operating distance of the still digital camera's zoom and flash is between four and seven feet. Therefore, the robot should be in this range for at least one of the subjects. We increase the values of the objective function in a band around each subject, with the value peaking at a distance of 5.5 feet.

*Occlusion* (figure 3b). Locations where faces appear to overlap will not yield good photographs. For each pair of subjects, we find the line that runs through them. Cells that lie on this line, but not on the line segment between the subjects, are reduced in value.

*Bisector* (figure 3c). Photos taken from along a perpendicular bisector between two subjects will result in both subjects being the same distance from the camera. If they are talking to each other, this results in two profile shots, which we would like to avoid. We calculate the

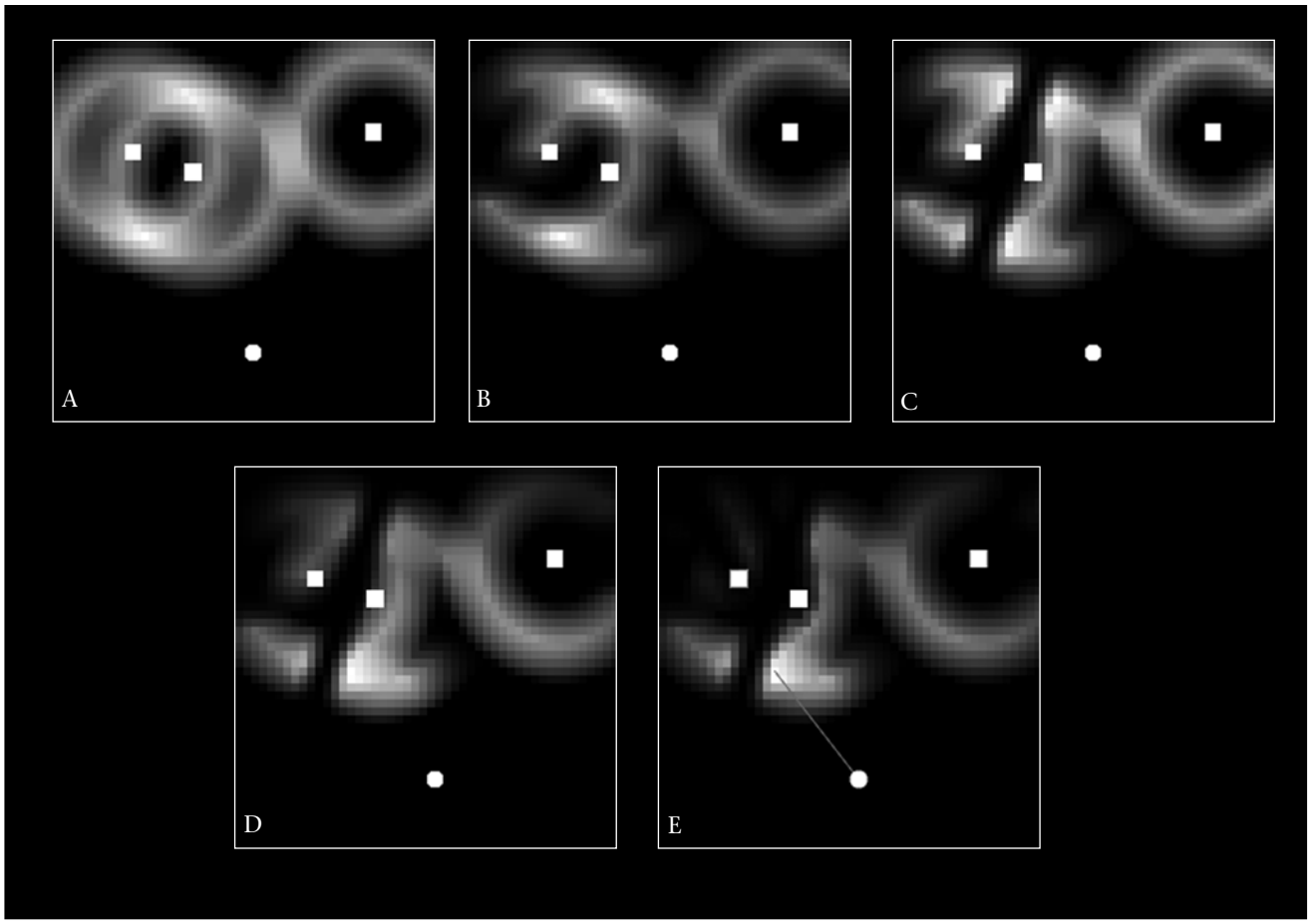


*Figure 3. Constructing the Objective Function to Take into Account*
*(a) Distance, (b) Occlusion, (c) Bisection, (d) Movement, and (e) Reachability.*

Brighter shades represent larger values of the objective function. The lowest white dot represents the robot position. The other dots are detected people.

perpendicular bisector of all subjects within five feet of each other and decrease the values of the objective function along this line.

*Movement* (figure 3d). We want to minimize the distance the robot travels to reduce the possibility that a photo opportunity will disappear. We also want to discourage the robot from taking multiple photographs at the same location. We decrease the objective function to zero for all points closer than 2 feet from the robot, or further than 20 feet. Between these extremes, we decrease the values linearly based on distance.

*Reachability* (figure 3e). To make navigation simple, we avoid destinations that would require sophisticated path planning and obstacle avoidance. A point is considered unreachable if the robot cannot drive in a straight line to it without going through something. We use the laser rangefinder information to calculate the horizon of all reachable points, and we set the objective function to zero for all points beyond this horizon.

Once the objective function is constructed, we simply look for the point with the greatest value and drive towards it. If an obstacle is encountered on the way, the destination is recalculated based on the new obstacles and the current location of faces.

## Navigation

When presented with a photographic opportunity, the system will attempt to move the robot to the given destination while avoiding obstacles. If obstacles prevent the robot from traveling along the ideal heading, a clear heading nearest to the ideal is chosen instead. The system continually reassesses the ideal heading, choosing either the ideal or the closest clear heading until the desired position is achieved. After a specified number of deviations from the ideal heading, the robot will abandon that photographic opportunity, preventing it from endlessly trying to reach an impossible position.

The system also has a random navigation mode that allows it to wander through the en-

vironment, opportunistically taking photographs. We found that this mode actually works best in crowded environments. In such conditions, the robot spends so much time avoiding people that it hardly ever gets to its goal in time. Also, because there are so many people milling about, most positions present reasonable photographic opportunities.

## Framing

When a suitable photographic opportunity has been reached, the system attempts to find a pleasing composition and take a photograph (Byers et al. 2003). Given a set of detected faces and their positions in the image, a framing algorithm calculates the image boundary of the ideal photo. We use four well-accepted rules from photography: (1) the rule of thirds, (2) the empty-space rule, (3) the no-middle rule, and (4) the edge rule (Grill and Scanlon 1990). The use of these rules is illustrated in figure 4.

*Rule of thirds:* It is best to place the faces in a photograph at, or near, the one-third and two-thirds horizontal lines in an image.

*Empty space:* The faces in an image should occupy at least the middle third of the image, either horizontally or vertically.

*No middle:* Do not place a single subject exactly on the midline of the photograph.

*Edge:* Subjects should not be placed at, or crossing, the edge of the photograph.

The amount each rule is enforced depends on the type of shot being taken (Byers et al. 2003). We have one set of rules for group shots (mostly rule of thirds and no empty space) and another for a single person shot, which requires tighter framing.

The ideal framing is then converted into the amount of pan, tilt, and zoom required to align the image boundary with the frame. The system continually calculates this framing and orients its camera until the ideal frame and current frame are sufficiently similar or until a predetermined amount of time has elapsed. Both of these values can be adjusted to adapt to different situations in order to accommodate a balance between precision and speed. When either condition is reached, a photograph is taken with the still camera.

### Displaying Photographs

We have a separate viewing station that displays the robot's results. As the robot takes photographs, they are transmitted to the viewing station. Attendees at the event can browse through the photographs and print them out, or e-mail them to someone. The number of photographs printed or e-mailed constitutes one of our evaluation metrics. We reason that if the robot is taking better photographs, more

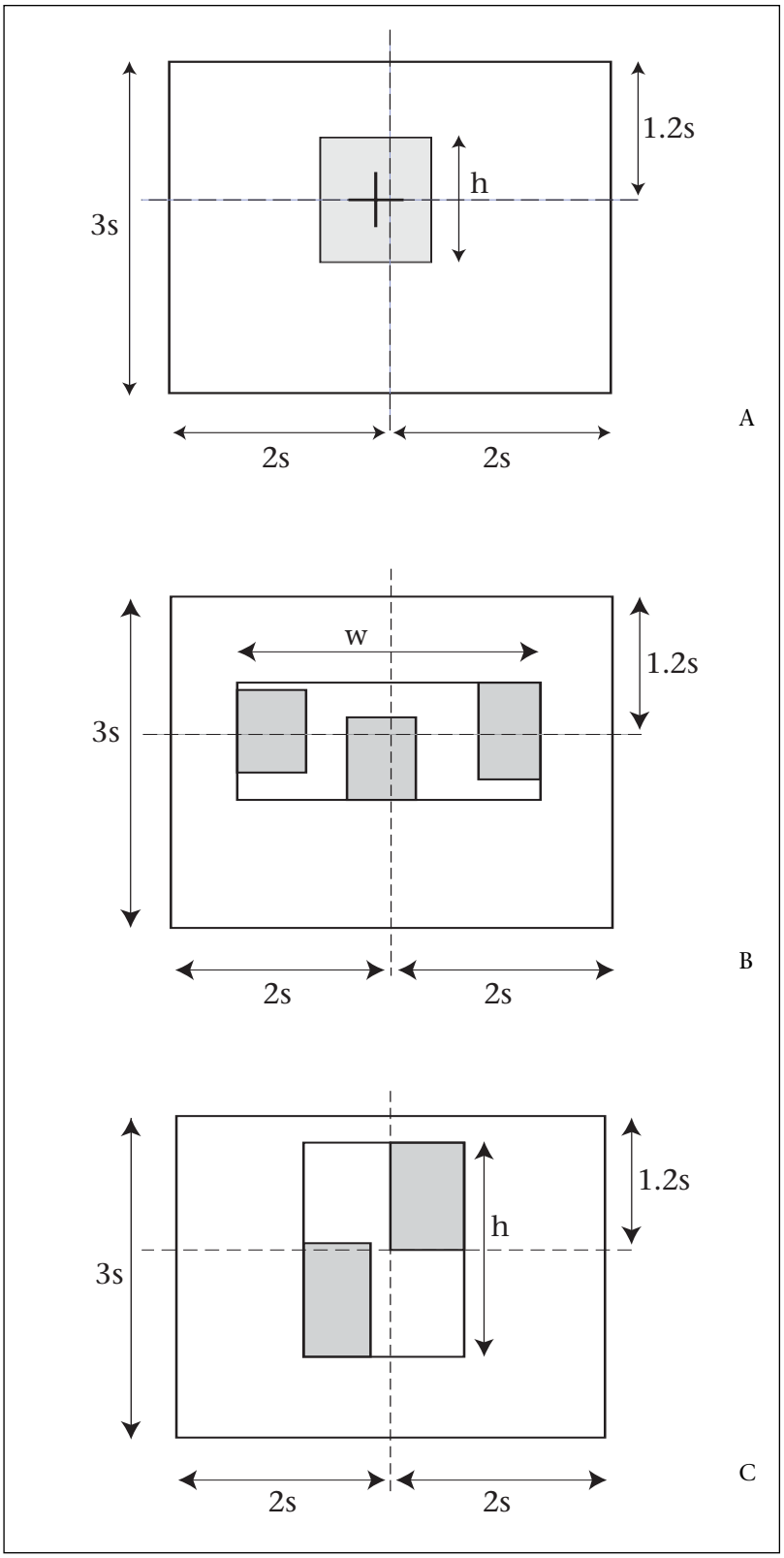


*Figure 4. Calculating the Ideal Framing for (a) A Single Face, and (b) A Wide Group of Faces, and (c) A Narrow Group of Faces.*

(a) $s = 1.5h$ (b) $s = {}^{3}/_{8}\,w$ (c) $s = {}^{1}/_{2}\,h$

of them will be printed or e-mailed. We discuss this rationale in more detail later in this article.

## System Overview

The current system consists of two layers of control and a sensor abstraction. The control layer takes care of all low-level navigation, localization, and obstacle avoidance. The task layer contains the code for the actual photography application, including the cameras and pan/tilt unit. We also include a sensor abstraction that allows us to restrict the robot's motion more easily. Both layers deal with the sensor abstraction, rather than directly with the sensors themselves.

The primary reason for arranging the system in this manner is to promote reuse of code across future applications. All of the photography specific code is contained in the task layer, while all of the general-purpose navigation systems are implemented in the control layer. This organization allows us to more easily deploy other applications without significantly rewriting basic routines.

Note that we also use a serial computation model for this system. We take a snapshot of the sensor readings, compute the next action, write that action to the motors, and then repeat the process. This model makes system debugging significantly easier, because we know exactly what each sensor reading is at every point in the computation; something that would not be the case if we were reading from the sensors every time a reading was used in a calculation. This model also allows us to inject modified sensor readings into the system, as described in the next section.

### The Control Layer

The control layer has three modules running concurrently: obstacle avoidance, relative motion, path planning, and localization.

The obstacle avoidance system is purely reactive, and attempts to keep the robot from colliding with other objects in the world. If there is an obstacle within a given range in the path of the robot, the heading is altered appropriately to avoid it. Obstacles closer to the robot tend to cause more drastic changes in course than those further away.

The relative motion module causes the robot to move towards a new position, specified relative to the current one. This module is responsible for local movement, and is superseded by the obstacle avoidance module.

The path planning module is responsible for movement to nonlocal destinations. It sequences partial paths, and uses the relative motion module to actually move the robot. Currently, this module is extremely simple. We orient the robot in the desired direction and drive towards the goal point.

The localization module is responsible for keeping track of where the robot is, and for correcting odometry errors. The robot counts the rotation of its wheels to keep track of position, but this procedure is notoriously prone to cumulative errors due to wheel slippage.

We have a simple localization strategy that involves finding two or more visual landmarks, and using triangulation to calculate the robot position. We currently localize only when necessary, trusting localization for short periods of time (about five minutes). In certain environments, for example when the robot is physically confined in a room, we have found that we do not need to localize at all.

### The Task Layer

The task layer contains all of the application specific code for the photography system. It requests robot motions from the control layer, and directly controls the camera and pan/tilt unit. The details of the composition system and picture finding were discussed in the previous section.

### The Sensor Abstraction

We introduced a sensor abstraction layer to separate the task layer from concerns about physical sensing devices. We process the sensor information (from the laser rangefinder in this application) into distance measurements from the center of the robot, thus allowing consideration of sensor error models and performance characteristics to be encapsulated, and easily reused across applications.

Sensor encapsulation and the serial computation model allow us to alter the sensor values before the task and control layers ever see them. We have found this to be a convenient mechanism for altering the behavior of the robot. For example, if we want to keep the robot within a particular area of a room, we can define an "invisible fence" by artificially shortening any sensor readings that cross it. The robot then behaves as if the invisible fence was a wall, and avoids it.

## Deployments

We have deployed the robot photographer system at a number of events. In this section, we describe the more important deployments, the amount of control we had over the environment, the configuration used, and perceived successes and failures. At the time of this writing, there had been four most signif-

icant deployments of the robot photographer system: at a major computer graphics conference (SIGGRAPH), at a science journalist meeting (CASW), at a wedding reception, and at Washington University's 150th birthday party celebration.

## SIGGRAPH 2002

The first major deployment of the system was at the Emerging Technologies exhibit at SIGGRAPH 2002, in San Antonio, Texas. The robot ran for a total of more than forty hours over a period of five days during the conference, interacted with over 5,000 people, and took 3,008 pictures, of which 1,053 (35 percent) were either printed out or e-mailed.

The robot was located in the corner of the exhibit space, in an open area of approximately 700 square feet. The area was surrounded by a tall curtain, with an entrance approximately eight feet wide. Other than a small number of technical posters and some overhead banners the space was mostly filled with grey or black curtains. Light was supplied by overhead spotlights, and three large standing spotlights in the enclosed area were added at our request to increase the overall lighting.

Deployment at SIGGRAPH took several days, in part because this was the first use, and in part because it took some time to adjust the lighting so that it illuminated faces without washing them out. We initially had plans for more advanced navigation and localization. Time constraints caused us to field a minimal system, which turned out to be surprisingly effective.

We used a glowing orange lamp as a landmark to prevent the robot from straying from the booth. Since there was only one door, it was sufficient to "tether" the robot to the lamp. Navigation was random, except when the robot reoriented itself or was avoiding objects.

## Council for the Advancement of Science Writing Meeting

The second major deployment was at a meeting of the Council for the Advancement of Science Writing (CASW), which took place in the dining room of the Ritz Carlton hotel, in St. Louis, Missouri. The robot operated in an unaltered area of about 1,500 square feet, as an evening reception took place. The robot shared the space with the usual furnishings, such as tables and chairs, in addition to approximately 150 guests, (mostly science journalists). The robot operated for two hours and took a total of 220 pictures. Only 11 (5%) of these were printed out or e-mailed by the reception guests, although several more were printed and displayed in a small gallery.

We spent three evenings calibrating the system. Primarily, this task involved calibrating the face-finding software to the lighting in the room and determining if there were any serious potential problems. At this event we added two new modules to the SIGGRAPH system: a digital camera to take better-quality photographs, and navigation software that attempted to place the robot in "good" places to take pictures. The success of this navigation module varied with the number of people present and how active they were. It performed best with a small number of people who did not move around too much.

As the room became more crowded and active the robot spent more time navigating to places (while avoiding people) only to discover that that the "good" place to take a picture had moved. Once this problem became evident, it would have been ideal to swap the more complex navigation module with the simpler one.

## An Actual Wedding

The system was deployed at the wedding reception for one of the support staff in our department. At this event, the system ran for slightly over two hours and took 82 pictures, of which only 2 (2%) were printed or e-mailed. The robot shared a space of approximately 2,000 square feet with 70 reception guests, some of whom were dancing.

We took a camera to the reception hall before the event, but calibration was largely done onsite an hour before the reception. The robot ran a system that was nearly identical to the one used at the CASW meeting.

The robot performed well while people were standing in the buffet line, but once the lights were lowered, we had to recalibrate the system. At this point, most people were sitting, so there were few potential shots. When the lighting was lowered again for dancing, the face-finding system was unable to function because of the low lighting levels.

## Birthday Party

The robot was deployed for three hours in an attic laboratory space with a handful of other exhibits. The chairs and tables were pushed to the side, but otherwise the space was unaltered. Over 3,000 people visited during the time frame, so the lab was always very crowded. The robot spent most of its time just taking pictures because the crowded conditions made navigation extremely difficult. The robot took 240 pictures (approximately one a minute) of which 79 (33%) were e-mailed out and about 120 (50%) were printed.

The only setup time was spent calibrating the skin-finding routine on the morning of the

event. The system was nearly identical to the one deployed at the wedding CASW meeting.

Although the robot was unable to demonstrate its navigational abilities at the birthday party, it was extremely popular—especially with children.

### Successes

The modules least susceptible to environment changes are the low-level people avoidance routines, camera control, image capture communication, and random navigation. The framing shots module is also fairly robust, provided the face-detection algorithm is functioning. The localization system worked well in the SIGGRAPH environment, but was not needed at the other events because of the configuration of the environment. Random navigation worked surprisingly well in crowded situations.

### Failures

The most fragile component of the system is face finding, which is highly dependent on the color and intensity of the lights and the background wall colors. In most environments we had very little control over the lighting. Even at SIGGRAPH we were constrained by the types of lights they could provide, although we could position them where we wanted. For example, we have deployed the robot at the St. Louis Science Center with mixed results because the walls are all painted with splashes of various colors that confuse the face finder.

The other area of mixed success was high-level navigation. Our two navigation strategies perform best in different environments—crowded versus sparse. At the CASW event and the wedding the number of people changed throughout the evening. In these situations, it would have been very useful to be able to automatically swap navigation strategies as conditions changed.

### Evaluation

The robot photographer system is inherently hard to evaluate. Most natural characterizations of performance are highly subjective. We also know of no similar system with which to compare ours. Based on the system's performance at SIGGRAPH, approximately one third of the pictures that the robot takes are at least good enough to qualify as souvenirs. This deduction agrees with some of our after-the-fact evaluations. People not affiliated with the project were asked to classify randomly selected photographs from the robot's portfolio, across all events, as either "very bad," "bad," "neutral," "good," or "very good." Roughly one third of the photographs were classified as "good" or "very good." While this is certainly not conclusive, we believe that it is encouraging, especially given the early development stage of the overall system.

We are currently planning more extensive evaluations. These include double-blind studies, where some human taken photographs will be randomly mixed in with the robot's to learn if people have a significant preference. We also plan evaluations by subjects who do not know a robot took the photographs, to see if there is a bias in our current results.

## Conclusions and Further Work

Several other robots have been fielded in similar real world deployments. For example, Minerva gave tours of the Smithsonian Museum of American History over a period of fourteen days (Burgard et al. 1999). This is certainly a longer deployment than we have had, with a similar level of environmental complexity. Other robots have been deployed for longer, but generally with much simpler tasks and environments (Hada and Yuta 2000). Another notable long-term deployment involves a robot that provides assistance for elderly persons (Montemerlo et al. 2002), which included several daylong deployments.

Although each of these robot systems has proven successful, they are all designed for a single environment, or for a very similar set of environments, thus allowing them to be optimized for a particular task. We believe that our experiences in a range of widely different indoor environments add a dimension that previous work does not address: the beginnings of general design principles for a robot system that must be deployed across differing environments.

Our robot photography system is still very much a work in progress. However, based on a number of real world deployments, we believe that there are a few general design rules that can be extracted from our experiences. These rules specifically apply to the design and implementation of an autonomous mobile robot system that must accomplish a complex task in an unaltered environment, while still being portable to other environments.[1]

### Adaptable to the Environment

The complexity that any successful robot system must deal with is a combination of the complexities of both the task and the environment. Even simple tasks can be hard to accomplish in complex environments. Although we have control over the task complexity, we often have little or no control over the environment.

Even simple environments, such as the one at SIGGRAPH, can have hidden complexities, and they are almost impossible to predict with

*Figure 5: Some Well-Composed Examples (Top Row and Middle Row Left), and Some Less Well-Composed Ones (Middle Row Right and Bottom Row).*

accuracy ahead of time. This constraint argues for a software architecture that can be altered easily at the deployment site. Since we really do not want to write and compile code onsite, we would like the system to be composed of relatively small modules that can be combined as necessary to get everything working.

Our experiences also argue for using as simple a system as possible to accomplish the task. Any complete robot system is, by definition, a complex collection of software that must all work at the same time. The fewer the number of elements that are present, the less the likelihood that something can go wrong.

## Highly Modular Framework

Onsite customization is much easier if the system is designed to be highly modular. Modularity also allows the system to be more readily expandable, as new sensors and algorithms become available. More importantly, however, the modular framework allows new

experimental modules to be easily added to the system and evaluated. For example, a student working on a new navigation algorithm can add it to the system and quickly evaluate it against all the current strategies, in the context of a whole application.

Being highly modular also suggests an incremental design strategy. As new problems arise due to new environmental complexities, we might be able to write a new module to deal with them. This demand-driven approach gives us two benefits. First, it means that if we do not need the new solution in a particular environment, we can easily remove it from the system (reducing the overall system complexity, as noted above). Second, it stops us from engineering solutions to problems that do not exist—at least to some extent. By following a demand driven approach to software design, we are forced to concentrate on fixing problems that actually matter. If, along the way, we discover a generally applicable improvement, we can incorporate it into an existing module.

As we pointed out previously, the only way to really be sure what the problems will be in an environment is to actually try out the system in that environment. When making changes to the system to accommodate the new location, a highly modular design allows compartmentalization of these changes, and prevents "creeping featuritis". We have observed this problem firsthand on other projects. If the code is in one monolithic system, the temptation to change some of it for a particular demo is large. Such changes often get left in the code, sometimes commented out, sometimes not. After a few such incidents, the source code for the system is likely to be a tangled mess of special cases.

## Serial Computation Model

Our main control loop follows a serial computation model. The sensors are read, computation is done on them, and then commands are sent to the motors. This sequence ensures that the sensor values are constant throughout the computation, which makes code debugging much easier. Snapshots of the robot state can also be saved for later replay and analysis. Because it is impossible to accurately recreate the state of the robot's sensors from run to run, snapshots of the robot state are an invaluable debugging tool. This single design decision has saved the most development time overall. It should be noted that only the actual control of the robot follows the serial computation model. We use multiple threads to handle communications, and other computations as needed.

## No One Size Fits All Solution

Perhaps the most important general observation that we can make is that there is currently no single best solution for our task. Even the same physical location changes from deployment to deployment, making it necessary to adapt the solution every time it is deployed. Although a completely autonomous system is our ultimate goal, at the present time we believe that it is not practical for the system to decide which modules are most appropriate on its own. By selecting and testing the modules actually used for a specific deployment, we can separate two possible sources of error: (1) error from selecting the wrong modules, and (2) errors caused by poorly designed modules.

## Acknowledgements

This work was supported in part by NSF REU award #0139576, and NSF award #0196213. The help of Michal Bryc, Jacob Cynamon, Kevin Goodier, and Patrick Vaillancourt was invaluable in the implementation, testing, and tweaking of the photographer system.

## Note

1. More details of the system, and example photographs, are available on the project web site at www.cse.wustl.edu/_lewis.

## References

Burgard, W.; Cremers, A.; Fox, D.; Hähnel, D.; Lakemeyer, G.; Schulz, D.; Steiner, W.; and Thrun, S. 1999. Experiences with an Interactive Museum Tour Guide Robot. *Artificial Intelligence* 114(12):3–55.

Byers, Z.; Dixon, M.; Goodier, K.; Grimm, C. M.; and Smart, W. D. 2003. An Autonomous Robot Photographer. Paper presented at the International conference on Robots and Systems (IROS 2003), Las Vegas, Nev., October.

Forsyth, D.; and Fleck, M. 1999. Automatic Detection of Human Nudes. *International Journal of Computer Vision* 32(1):63–77.

Grill, T.; and Scanlon, M. 1990. *Photographics Composition*. Orlando, Fla.: American Photographics.

Hada, Y.; and Yuta, S. 2000. A First-Stage Experiment of Long Term Activity of Autonomous Mobile Robot—Result of Repetitive Basedocking over a Week. Paper presented at the Seventh International Symposium on Experimental Robotics (ISER 2000). Carnegie Mellon University, Pittsburgh, Penn., 10–13 December.

Montemerlo, M.; Pineau, J.; Roy, N.; Thrun, S.; and Verma, V. 2002. Experiences with a Mobile Robotic Guide for the Elderly. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*. Menlo Park, Calif.: AAAI Press / The MIT Press.

**Zach Byers** earned his M.S. degree in computer science at Washington University in St. Louis, where he worked in the Media and Machines laboratory. His research interests cover all aspects of mobile robotics.

**Michael Dixon** (msd2@cse.wustl.edu) is a staff researcher at the Media and Machines Laboratory at Washington University in St. Louis where he received his M.S. and B.S. in computer science. His research interests include sensor fusion and computer vision.

**Bill Smart** (wds@cse.wustl.edu) is an assistant professor of computer science and engineering at Washington University in St. Louis, where he codirects the Media and Machines laboratory. His research interests lie in the areas of machine learning and mobile robotics.

**Cindy Grimm** (cmg@wustl.edu) is an assistant professor at Washington University in St. Louis. She obtained her Ph.D. in computer science from Brown University in 1996. Her research interests include art-based rendering and surface modeling.