Say Cheese!: Experiences with a Robot Photographer

Zachary Byers and **Michael Dixon** and **William D. Smart** and **Cindy M. Grimm**
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130
United States
{zcb1,msd2,wds,cmg}@cse.wustl.edu

Abstract

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 paper, we outline the overall architecture of the system and describe how the various components inter-relate. We also describe our experiences of deploying the robot photographer at a number of real-world events.

Introduction

In this paper, we describe our experiences with an autonomous photography system mounted on a mobile robot. The robot navigates around social events, such as wedding receptions 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 paper gives an overview of the entire robot photographer system, and details of the architecture underlying the implementation. We discuss our experiences with deploying the system in several environments, including a scientific conference and an actual wedding, and how it performed. We also attempt to evaluate the quality of the photographs taken, and discuss opportunities for improvement.

The system is implemented with two digital cameras (one still and one video), mounted on an iRobot B21r mobile robot platform. The robot stands slightly over four feet tall, and is a bright red cylinder approximately two feet in diameter. The cameras are mounted on top of the robot on a Directed Perception pan/tilt unit. All computation is done on-board, on a Pentium-III 800MHz system. The only sensors used for this project are the cameras and a laser range-finder, which gives 180 radial distance measurements over the front 180° 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.

At a high level, the system works as follows. The robot navigates around the room, continually looking for "good" photograph 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 a "perfect" 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 the paper, we discuss our motivation for undertaking this project and describe the various aspects of the system. We then describe some of the major deployments of the system, and show examples of the photographs that it 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. The robot photographer project started as a framework within which we could do that research. It was also designed to be appealing to undergraduates, and to encourage them to get involved in research. Automated photography is a good choice of application, since 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 (how do you automate the skill of photograph composition).

Because the concept of a robot photographer 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 very 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, and to catch pictures of people interacting with each other, rather than standard "mug-shot"

Copyright © 2003, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

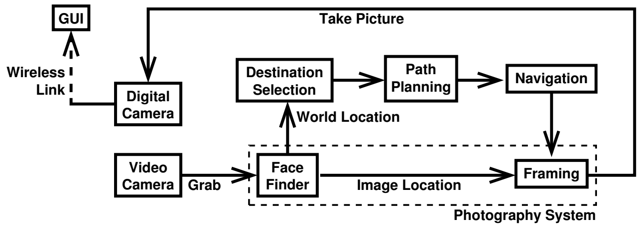

Figure 1: An overview of the photography system architecture.

types of photos.

We feel that we should note that there is a lot of room for improvement in the current system. Many of the algorithms are quite basic, and the performance of the system would be improved if they were improved or replaced. 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 costs power, and can lead to significantly shorter battery lifetimes. We are, therefore, interested in the simplest algorithm that we can get away with, even if performance is not quite as good.

Now that the basic system is in place we are finding that it is 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, *etc.*

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 are summarized in figure 1.

Locating Potential Subjects

In order 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, we are able to define a look-up 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 non-skin. 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. This 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 associates with faces, then this 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 then used to calculate the location of the best photographic opportunity. We discretize the floor plane into a grid with squares 20cm 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. This objective function is calculated using knowledge about good picture composition.

- The best pictures are taken between 4 and 7 feet from the subject.
- One subject should not occlude another.
- Photographs should not be taken from the perpendicular bisector of two subjects.
- Positions that are closer to the current robot position should be preferred.
- If the direct path to a position is obstructed, that position should be less desirable.

These rules are encoded as parts of the objective function. For example, the first rule could be implemented by calculating the distance, d_i, from the position under consideration to each person in the environment. This rule would then contribute a value, v_1, to the objective function, where $v_1 = \sum_i \exp\left(-(d-5.5)^2\right)$. There will be one such term, v_j, for each of the rules above, and the total value, v, is just the sum of them, $v = \sum_{j=1}^{5} v_j$. This is illustrated in figure 2. Once we calculate values for all cells within a set distance of the robot, we select the one with the highest value as the next destination.

Navigation

Given 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 that or the closest clear heading until the desired position is achieved. After a specified number of deviations from the ideal heading, the robot will give up on that photograph, preventing it from endlessly trying to reach an impossible position.

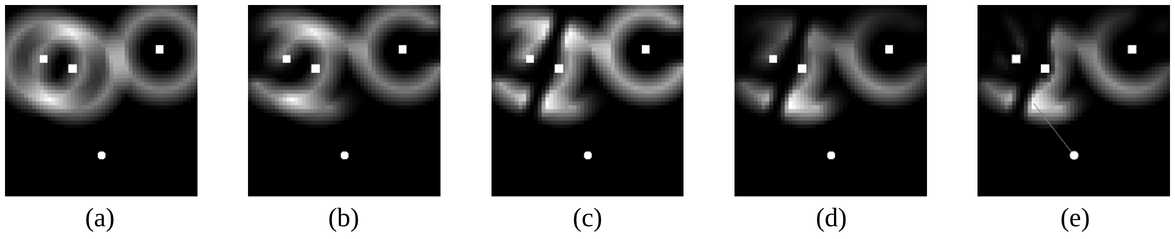

Figure 2: Constructing the objective function to take into account (a) distance, (b) occlusion, (c) bisection, (d) movement, and (e) reachability. Lighter shades represent larger values of the objective function. The lowest white dot represents the robot position. The other dots are detected people.

The system also has a random navigation mode, where it randomly wanders through the environment, opportunistically taking photographs. We found that this actually works better in very crowded environments. In these cases, the robot spends so much time avoiding people, that it hardly ever gets to its goal in time. Also, since there are so many people about, most positions are reasonable for taking photographs.

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. The specific composition rules used to calculate this ideal framing is beyond the scope of this discussion, but our design allows us to easily vary the framing algorithm based on the level of complexity required.

This 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 adjusts its camera orientation 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 for displaying 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 email them to someone. The number of photographs printed or emailed is one of our evaluation metrics. We reason that if the robot is taking better photographs, more of them will be printed or emailed. We discuss this in more detail later in the paper.

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 to allow us to restrict the robot's motion more easily. Both layers deal with the sensor abstraction, rather than directly with the sensors themselves.

The main 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 will allow us to more easily deploy other applications without significantly rewriting the basic routines.

We should also note that we 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 makes debugging of the system significantly easier, since we know exactly what each sensor reading is at every point in the computation. this would not be the case if we were reading from the sensors every time a reading is used in a calculation. This model also allows us to inject modified sensor readings into the system, as described below.

The Control Layer

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

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

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

Path Planning The path planning module is responsible for movement to non-local 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.

Localization 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 is notoriously prone to cumulative errors due to wheel slippage.

We have a simple localization strategy which involves finding two or more visual landmarks, and using triangulation to calculate the robot position. We currently localize only when needed, trusting localization for short periods of time (about 5 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 this layer were discussed in the previous section.

The Sensor Abstraction

We have introduced a sensor abstraction layer in order to separate the task layer from concerns about physical sensing devices. We process the sensor information (from the laser range-finder in this application) into distance measurements from the center of the robot. This allows consideration of sensor error models and performance characteristics to be encapsulated, and easily re-used across applications.

This encapsulation, and the serial computation model, allows us to alter the sensor values before the task and control layers ever see them. We have found that this is 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 there was a wall in the position of the fence, 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. We cover the amount of control we had over the environment, the configuration used, and perceived successes and failures. At the time of writing, the three most significant deployments of the robot photographer system are at a major computer graphics conference, at a science journalist meeting, and at a wedding reception.

SIGGRAPH 2002 The first major deployment of the system was at the Emerging Technologies exhibit at SIGGRAPH 2002, in San Antonio, TX. The robot ran for a total of more than 40 hours over a period of five days during the conference, interacted with over 5,000 people, and took 3,008 pictures. Of these 3,008 pictures, 1,053 (35%) were either printed out or emailed to someone.

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 deployment, 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. Due to time constraints, we ended up fielding a bare-minimum system, which turned out to be surprisingly effective.

We used a landmark (a glowing orange lamp) 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 re-oriented itself or was avoiding objects.

CASW 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, MO. 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 emailed by the reception guests, although several more were printed and displayed in a small gallery.

We spent three evenings calibrating the system in the hotel. Primarily, this was to calibrate the face-finding software to the lighting in the room and determine 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 at a "good" place 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 a lot of time navigating to places (while avoiding people) only to discover that the people had moved. At this point it would have been ideal to swap out the current navigation module and return to the simpler one.

An Actual Wedding The system was deployed at the wedding reception of one of the support staff in our department. At this event, it ran for slightly over two hours and took 82 pictures, of which only 2 (2%) were printed or emailed. 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 the calibration was largely done on-site an hour before the reception. The robot ran a system that was nearly identical to the one used in the CASW meeting.

The robot performed well while people were standing in the buffet line, but after this the lights were lowered and we had to re-calibrate the system again. At this point, most people were sitting so there were few potential shots. Then the lighting was lowered again for dancing, and the face-finding system was unable to function at those lighting levels.

Successes

The modules that are least susceptible to environment changes are the low-level people-avoidance routines, camera control, image-capture communication, and the random navigation. Framing shots 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 to use the types of lights they could provide us, although we could position them where we wanted to.

The other area where we had variable 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 this case it would have been very useful to be able to automatically swap navigation strategies depending on the situation.

Evaluation

A system like the robot photographer 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 performance at SIGGRAPH, approximately one third of the pictures that the robot takes are at least good enough to qualify as souvenirs. This agrees with some recent evaluations we have begun. People were asked to classify randomly-selected photographs from the robot's portfolio 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 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 see 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 14 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 & Yuta 2000). Another notable long-term deployment involves a robot that provides assistance for elderly persons (Montemerlo *et al.* 2002), which included several day-long deployments.

Although each of these robot systems has proven very successful, they all share something in common. They are all designed for a single environment, or for a very similar set of environments. This allows them to be optimized for that particular task. We believe that our experiences in a range of widely different indoor environments adds a dimension that this previous work does not address: the beginnings of general design principles for a robot system that must be deployed across several different 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 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. More details of the system, and example photographs, are available on the project web site at `http://www.cse.wustl.edu/~lewis`.

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 our SIGGRAPH deployment, can have hidden complexities. These are almost impossible to predict with accuracy ahead of time. This argues for a software architecture that can be altered easily at the site of the deployment. Since we really do not want to be writing and compiling code on-site, we would like that 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 elements that are present, the less there is to go wrong.

Highly Modular Framework On-site customization is much easier if the system is designed to be highly modular. It also allows it to be more readily expandable, as new sensors and algorithms become available. More importantly, however, it 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 be able to evaluate it against all of the current strategies, in the context of a whole application.

Being highly modular also suggests an incremental design strategy. As new problems crop up due to new environmen-

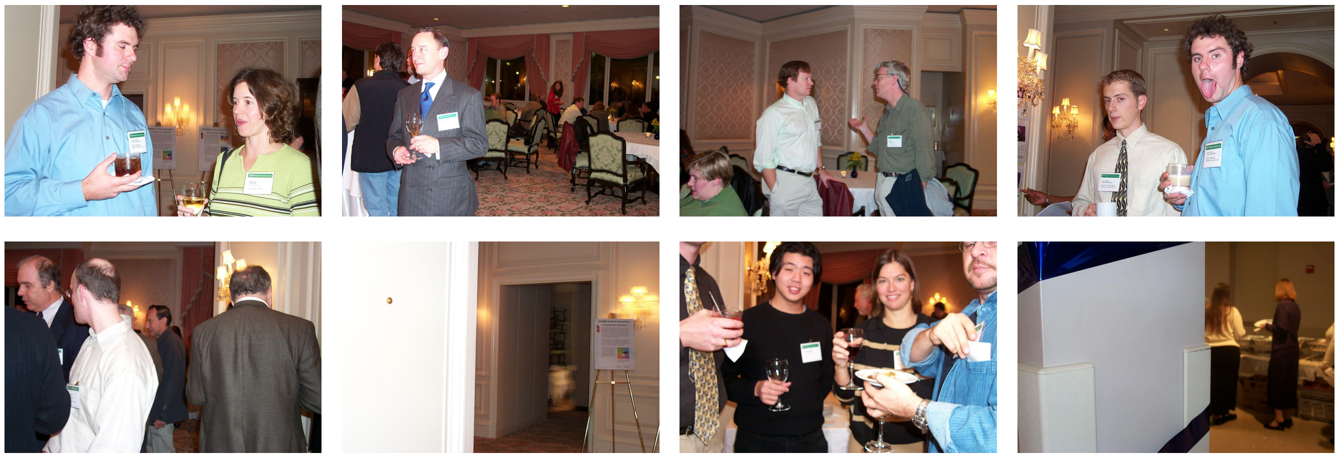

Figure 3: Some well-composed examples (top row), and some less well-composed ones (bottom row).

tal complexities, we might be able to write a new module to deal with them. The provides us with 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 complexity of the system, as noted above). The second benefit is that it stops us from engineering solutions to problems that do not exist, at least to some extent. If we follow a demand-driven approach to software design, it forces us to concentrate on fixing problems that actually matter. If in doing so we discover a generally applicable improvement, it can be incorporated 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 first-hand 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, then commands are sent to the motors. This ensures that the sensor values are constant throughout the computation, which makes debugging of code *much* easier. These 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, this is an invaluable debugging tool. This has proven to be the single design decision that has saved the most development time overall. It should be noted that only the actual control of the robot follows this 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: error from selecting the wrong modules, and 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.

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(1-2):3–55.

Byers, Z.; Dixon, M.; Goodier, K.; Grimm, C. M.; and Smart, W. D. 2003. An autonomous robot photographer. Under review. Available from the authors on request.

Hada, Y., and Yuta, S. 2000. A first-stage experiment of long term activity of autonomous mobile robot — result of repetitive base-docking over a week. In *Proceedings of the 7th International Symposium on Experimental Robotics (ISER 2000)*, 235–244.

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 AAAI National Conference on Artificial Intelligence*. Edmonton, Canada: AAAI.