AI Magazine Volume 17 Number 1 (1996) (© AAAI)

LOLA

Object Manipulation in an Unstructured Environment

Richard LeGrand and Ren C. Luo

■ LOLA won the Office Cleanup event at the 1995 Robot Competition and Exhibition, held as part of the Fourteenth International Conference on Artificial Intelligence. The event called for a robot to pick up trash in an unstructured environment and sort it such that the recyclable trash winded up in the recycle bin and the regular trash in the trash bin. The only allowable information lola was given beforehand were model-based descriptions of the trash and recyclables, which it located using color vision. Much of LOLA's success can be attributed to the simple, fast algorithms and methods that also model sensor uncertainty. The ideas and design philosophy that went into LOLA borrow heavily from those of previous competitors—to which we are greatly indebted. These methods and ideas are discussed here.

My teammate, Ricardo Gutierrez-Osuna, had just finished event 1 and was moving LOLA out of the competition ring using the joystick. I had been at the workstation in case LOLA needed a restart, but I could tell by the look on Ricardo's face that LOLA had performed well. He gave me a thumbs up, and it was confirmed. I returned with an enthusiastic two thumbs.

There had been considerable talk about "demo-itis" and Murphy's Law while we prepared for the competition. "It might work well now but wait until a hundred or so people are watching. Then you'll know what I'm talking about..." or so we had been told by past competitors. But the first event was over, and Murphy was nowhere to be found. My head began to fill with gushy thoughts about the joys of robotics. Yep, today was Murphy's day off.

Ricardo handed me the joystick. I looked at my watch and saw that we had three hours until we were up for event 2—plenty of time for a quick test of things and then maybe a bite to eat. I look at LOLA's console and see an unfamiliar message. What's this? Bad hard-drive sector?

A week ago I had asked someone after I had made hard-drive backups, "Do people crash hard drives these days?" He hadn't heard of anyone recently but told me a story about a guy who had dropped his drive on the kitchen floor—but that was years ago.

I was thinking to myself as LOLA rebooted, "Heh heh, what are the chances of our hard drive crashing on the day of the competition finals?" After a few more tests, it was confirmed: LOLA's hard drive had decided to crash at the worst possible time. The irony of the situation was overwhelming. "I hate robotics," I said aloud. I really meant it too.

Of course, we were happy that LOLA went on to win event 2 (it was awfully close). We'll talk more about the problems we encountered, the solutions we imposed, the people we'd like to thank, and that mysterious phenomenon called demo-itis—because after the competition, there was no other logical explanation. In all fairness, though, LOLA did a pretty good job of picking up trash—we'll talk about that too.

Hardware

LOLA is a NOMAD 200 with a standard sonar ring and tactile bumper sensors. Figure 1 shows LOLA holding a Coke can with its forklift-like arm and gripper from Nomadic Technologies. Because the arm is in the rear, it doesn't obstruct any frontal sensors (for example, sonar, vision, bumpers) and is safe from collision during forward motion. We took advantage of the Nomad's modularity

Figure 1. LOLA (NOMAD 200).

and moved three of the rear sonar sensors to the front, just above the bumper. This arrangement provided LOLA with the ability to sense chairs and other potential obstacles that might otherwise be unseen.

LOLA's main processor is a 486DX2-66 running LINUX (UNIX). The combination of LINUX and radio Ethernet makes code development on LOLA a real joy. That is, from any workstation on the network (including the Internet), we can telnet to LOLA and export the display for developing, debugging, and executing code while we monitor LOLA's status during operation, which beats having to wheel around a terminal and an extension cord. The idea is to do all the development without leaving your chair.

The vision hardware consists of an on-board image processor and a single RGB (red, green, blue) camera mounted on a pan-tilt unit. LOLA's image processor was purchased from Traquair Data Systems and is blessed with two C40 digital signal processors running in parallel. Performing all computation on board has several advantages: The video data are not corrupted by radio-transmission noise, commands are not lost, and there's no communication lag that might result in LOLA crashing into things. These findings are consistent with those of previous competitors (Nourbakhsh, Powers, and Birchfield 1995). On the down side, the on-board image processor contributes significantly to the battery drain, which is partly the result of its intended desktop use. Still, we are able to get about two hours of operation to each charge. Nomadic Technologies is currently making efforts to offer a version that is better suited for mobile robot use.

Software Architecture

To keep things simple, we reasoned that the operations for performing trash pickup can be described by a state machine. Unlike the first event, state estimation is not critical; if the robot gets confused, it can always abandon what it is doing currently and resort to looking for another piece of trash. Figure 2 conveys the basic idea of LOLA's state machine:

Locate locates either trash or receptacles in the image.

Turn turns LOLA toward the object once the trash or receptacle is located.

Pursue heads LOLA straight toward the trash or receptacle.

Pickup turns LOLA around and grasps the trash.

Deposit turns LOLA around and deposits the trash.

Avoid causes LOLA to attempt to avoid the obstacle and continue with Pursue after the obstacle is successfully avoided if an obstacle is detected by sonar during Pursue.

Wander is used if LOLA cannot find trash or receptacles.

The idea is that if a robot can accomplish these basic tasks well, it can perform trash pickup in an unstructured environment. LOLA's state of mind never extends beyond a single piece of trash at a time; so, there is no provision for path planning around trash that is in the way of other trash. To compensate for the trash that inevitably gets knocked over as a result, LOLA relies on the ability to pick up trash on its side. Thus, LOLA doesn't

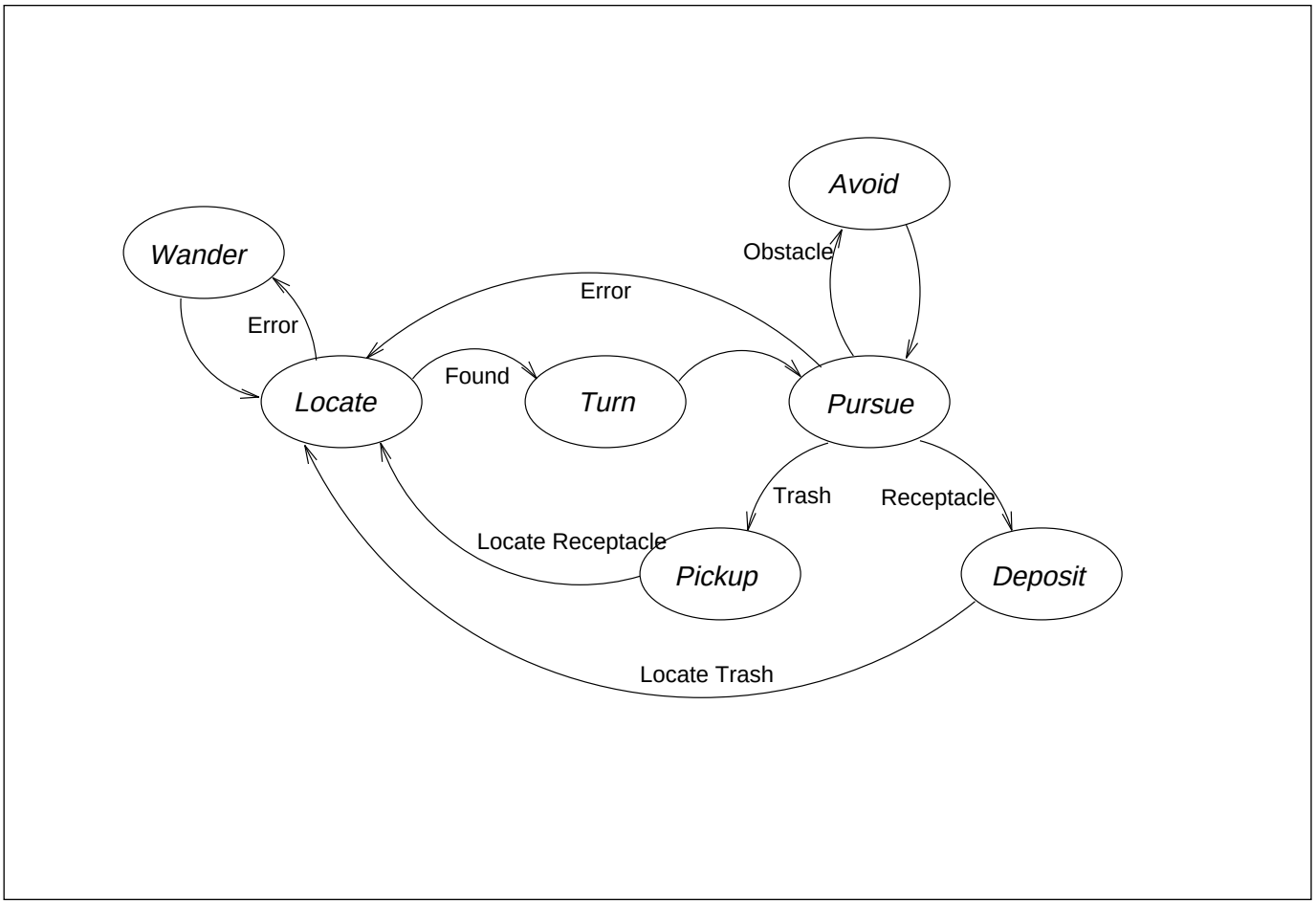

Figure 2. LOLA's State Diagram.

need to retain much information about the environment. Only the relative locations of recently visited receptacles are retained to help locate them more quickly.

In addition to this framework are (1) a vision algorithm that locates trash and receptacles and (2) an estimation algorithm that provides the position estimate of these objects. We reason that if LOLA can construct an accurate position estimate of the trash and receptacles, the *manipulation task* (picking up and depositing) becomes easy. That is, the position estimate tells LOLA when to stop during Pursue and exactly where to place the gripper for Pickup and Deposit. Additionally, during Avoid, the position estimate is used to perform a goal-directed avoidance behavior. Once the obstacle is successfully avoided, the trash or receptacle can be reacquired in the image by mapping the position estimate to the image plane. Thus, obstacles that occlude the goal won't confuse LOLA once they are avoided. However, during the competition, no obstacles were encountered, and Avoid wasn't used. This ability was demonstrated during the exhibition.

LOLA uses color-histogram back projection (Swain and Ballard 1991) to locate trash and receptacles.[1] The algorithm is inexpensive and does a good job of locating targets in an image, simply given the target's color histogram. More specifically, it provides the centroid location in the image of the *best candidate* (the candidate that forms the best model-based match). For the competition, we used three types of trash: (1) orange soda cans and (2) lime soda cans for recyclable trash and (3) paper Coke cups for regular trash. Colored markers were placed around the bottom of the receptacles: Yellow indicated recyclable trash, and pink indicated regular trash. During Locate, LOLA refers to a database of color-histogram models, containing all trash and receptacle objects, to locate specified objects.

Various methods are used to speed up Locate, which tends to be the most time consuming and boring to watch. To locate trash, for example, the trash histograms are merged

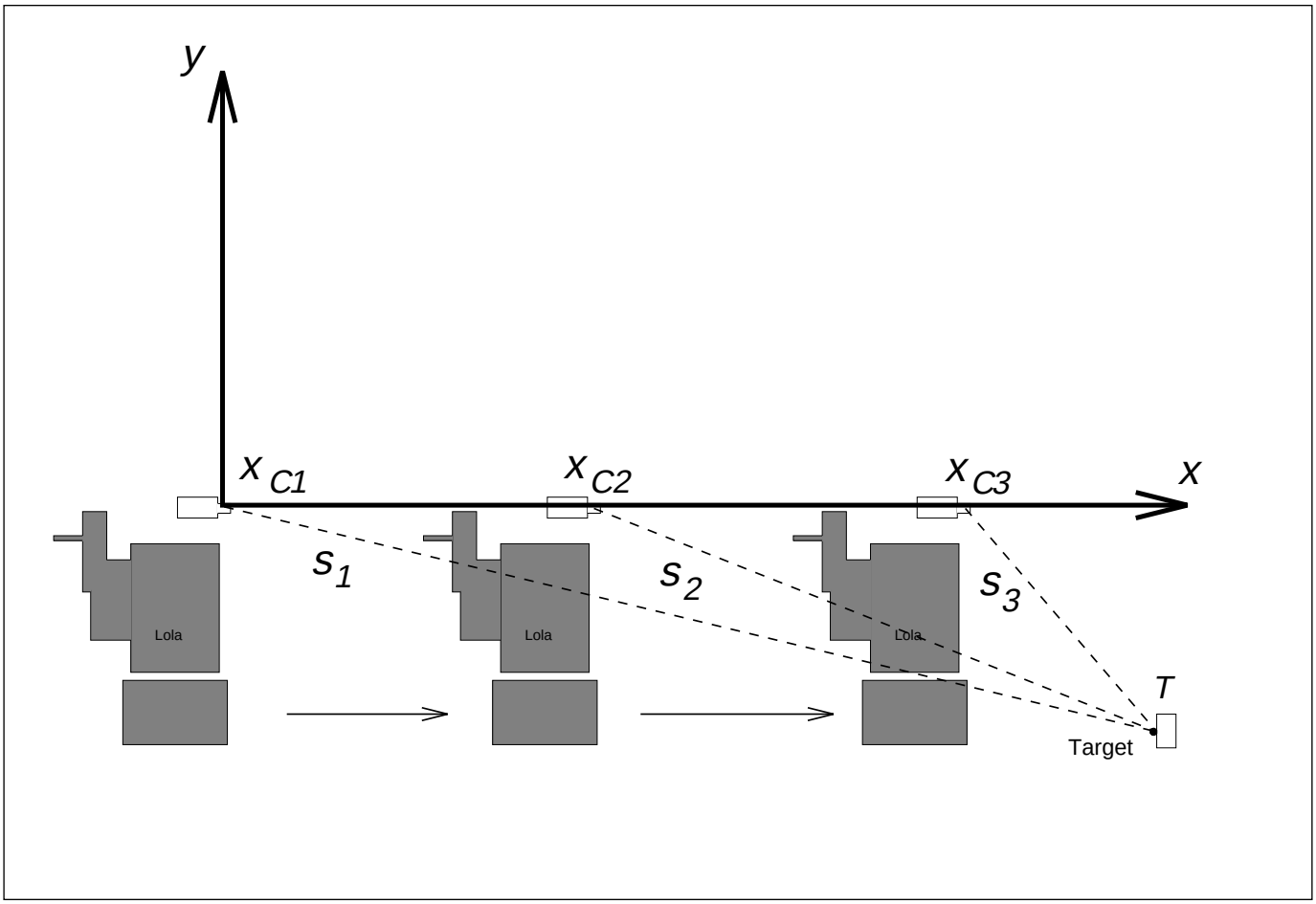

Figure 3. Pursue While Estimating Target Position.

into a single histogram by summing them up binwise. The merged histogram is used to quickly find generic trash candidates (as opposed to single types of trash) in the image. A voting algorithm then identifies the trash by choosing the trash model with the most votes. Determining the type of trash (regular or recyclable) is then a simple lookup.

Once the trash is identified, a smaller square region of interest (ROI) is formed in the image closely surrounding the trash candidate. Back projection is then applied to the ROI using the model histogram with the most votes. For trash targets, the reduced ROI is typically 100 by 100 pixels, which greatly boosts the computation speed. For example, LOLA's image-processing hardware is able to perform back projection on an image this size at about 15 frames a second. At this rate, we can reasonably assume that the target will not move far in the image between successive frames while the robot is moving. This assumption allows us to perform simple target tracking by centering the ROI where the centroid of the target was in the previous frame. The resulting algorithm controls the placement of the ROI in the image such that it tracks the target regardless of the camera motion (or target motion, for that matter). Of course, the target's image velocity must be kept below a maximum, but in practice, this technique works well for stationary objects and camera motion induced by normal robot movement. This technique borrows some basic ideas from the active vision community (Ballard 1991; Aloimonos 1990).

The combination of these simple, fast algorithms makes visual serving possible. For example, during Pursue, some method must be used to actively control the robot heading to the target. LOLA simply uses a proportional-derivative compensator to compensate for the difference between the target centroid and the center pixel column on the image plane. To give an idea of how simple this process is, implementing this in C takes less than 15 lines of code.

The Estimation Algorithm

In addition to actively controlling the heading during Pursue, LOLA simultaneously estimates the position of the target by using the centroid measurements and the camera motion inferred from the wheel encoders.

Figure 3 illustrates this idea. That is, we are only interested in the position of the target in the plane normal to the ground that contains the target, the camera focal point, and the line of pursuit. It is assumed, for example, that an accurate position estimate of a trash target within this plane is all that is needed to fully specify the placement of the gripper for physical pickup. Of course, the assumption requires that the heading error be small immediately before LOLA turns around to grab the trash. In practice, the heading error never exceeds two degrees, which was apparent in the competition when LOLA would pick up trash oriented on its side. In this case, there was only one inch of tolerance on each side of the forklike gripper before it closed.

Derivation

The algorithm assumes pinhole camera dynamics with focal length f. Because we are only estimating within a plane, we only need to consider the centroid measurement in one dimension on the image plane, calling this dimension v:

$$v_T = f\frac{y_T}{x_T} \quad . \tag{1}$$

Here, v_T is the noiseless projection of the target centroid onto the image plane, and x_T and y_T are the x and y coordinates of the target in the estimation plane. Next, we assume that the centroid measurements (v_c) from the back-projection algorithm are corrupted by Gaussian noise, which implies the conditional distribution:

$$f_{v_c|v_T} = \alpha \exp\left(\frac{-(v_c - v_T)^2}{2\sigma^2}\right) \quad . \tag{2}$$

For simplicity, we divide equation 1 through by f and introduce s_c, which is the slope of the measured centroid line. We now rewrite equation 2 with the pinhole assumption and introduce a new term, x_c, which is the position of the camera along the x axis:

$$f_{s_c|x_T,y_T} = \alpha \exp\left(\psi\left[s_c - \frac{y_T}{x_T - x_C}\right]^2\right) . \tag{3}$$

Thus, we have expressed each centroid measurement as a conditional distribution with conditioning set $\{x_T, y_T\}$. That is, given any target location, equation 3 describes the probability density function (PDF) of s_c, and evaluating the PDF at a measurement value gives us the probability of measuring the value under the target position hypothesis. However, what we really want is a bivariate distribution of the target in the estimation plane given all the previous centroid measurements.

Applying Bayes's rule and assuming conditional independence between slope measurements, we get the product form

$$f_{x_T,y_T|S_k} = \beta_k \prod_{j=1}^{k} f_{s_j|x_T,y_T} \quad , \tag{4}$$

where

$$S_k = \{s_1, s_2 \ldots s_k\} .$$

It is assumed that β_k is constant as x_T and y_T are varied. Combining equations 3 and 4, we get the bivariate distribution of the target within the estimation plane given all previous measurements:

$$f_{x_T,y_T|S_k} = \gamma_k \exp\left(\psi \sum_{j=1}^{k}\left[s_j - \frac{y_T}{x_T - x_{Cj}}\right]^2\right) \quad . \tag{5}$$

Here, s_j and x_{Cj} are the slope and camera position measurements at time j, respectively. For convenience, we create a cost function, $C_k(x_T, y_T)$, as follows:

$$C_k(x_T, y_T) = \sum_{j=1}^{k}\left(s_j - \frac{y_T}{x_T - x_{Cj}}\right)^2 \quad . \tag{6}$$

Because $\psi < 0$, if we minimize $C_k(x_T, y_T)$, we maximize equation 5. Thus, min $C_k(x_T, y_T)$ provides $\hat{x}_T$ and $\hat{y}_T$, which are the *maximum likelihood estimate* of the target position. Notice that equation 5 describes the shape of the bivariate probability distribution around the estimate scaled with the constant γ_k.

Additionally, it is simple to calibrate a camera for use of this estimation technique. Recognizing that each measurement is a slope value in the estimation plane, all that is required for calibration is a mapping from each pixel in the v dimension on the image plane to a corresponding slope value in the estimation plane. In practice, this mapping can be created by placing markers in front of the camera, measuring each marker's y_M / x_M value (M denotes marker) and its projection onto the image plane, and creating a lookup table that describes the mapping. By placing 80 pieces of tape on the floor in front of the camera and linearly interpolating between their projections on the image plane, our camera was calibrated in an afternoon (LeGrand 1995). Thus, none of the intrinsic parameters of the camera have to be recov-

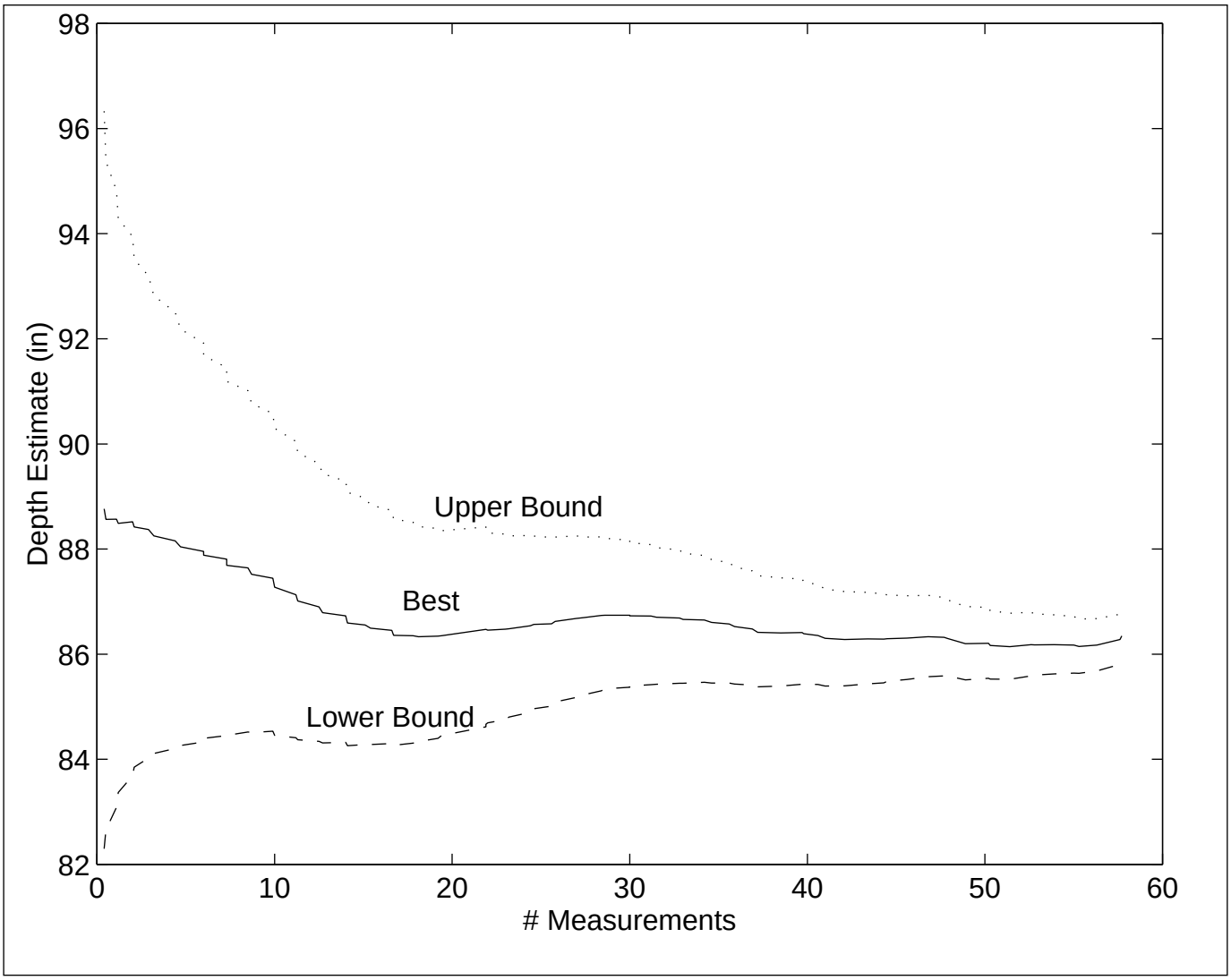

Figure 4. Position Estimate Convergence.

ered, and deterministic biases are kept to a minimum. Although this calibration method cannot claim to be accurate to much less than one pixel, higher accuracy is not needed because the back-projection centroid measurements typically have many more pixels of noise (five pixels of noise are not unreasonable).

Applying the Estimator

For the competition, we took advantage of the fact that all the trash was on the ground and applied a ground-plane assumption. This assumption meant fixing $y_T = y_{\text{ground}}$ in equation 6 and minimizing equation 6 with respect to x_T by using a golden-sectioning optimization algorithm to give the most likely estimate, $\hat{x}_T$. In general, however, the ground-plane assumption is not necessary, and LeGrand (1995) describes quick methods for recovering both x_T and y_T estimates simultaneously.

Recovering the probability information in equation 5 requires a little trick because we don't know the normalizing constant, γ_k, but we've presumably found $\hat{x}_T$, which is the maximum of equation 5 within the ground plane. By evaluating equation 5 at $x_T = \hat{x}_T$, and $y_T = y_{ground}$, we obtain the relative–maximum-likelihood value we call P_{max}. We can then proceed to evaluate equation 5 for several values by moving along $y_T = y_{ground}$ in each direction from $\hat{x}_T$ and noting where the relative-probability value drops below a certain percentage of P_{max} (say, 10 percent). This evaluation gives a good idea of the upper and lower bounds of the estimate. Figure 4 demonstrates this idea by showing the upper

and lower bounds of the maximum- likelihood estimate converging as more measurements are taken. This plot was generated using measurements collected from LOLA while it pursued a Coke can.

The rationale for all this mess is that a position estimate by itself is not useful, but a position estimate combined with probabilistic information (in this case, an upper- and a lower-bound metric) can be used by a robot to make intelligent decisions. For trash pickup, the probabilistic information can be used to determine whether the estimate is good enough to perform a successful grasp. The robot might then decide to take more measurements, for example, or give up and look for another piece of trash. Unfortunately (or fortunately), at the competition, the estimates were all good enough, and LOLA never used the probabilistic information in this way. However, picking up Coke cans on our wooden lab floor is a different story because the back-projection algorithm occasionally confuses parts of the reddish wood for Coke cans. However, this confusion isn't a problem because the misidentified floor tends to have much noisier centroid measurements, which result in poorly converged position estimates. Thus, the problem can be caught before LOLA tries to pick up a "phantom" Coke can.

Many of these ideas were inspired by last year's winning team from Stanford University, which approached the problem of robot navigation by modeling the sensor uncertainty and retaining the relative uncertainty of the robot's many possible states (Nourbakhsh, Powers, and Birchfield 1995). By assuming that it's in the most likely state and making decisions based on this assumption, the robot was able to make intelligent decisions on how to proceed. The previous estimation algorithm borrows these same ideas. The key to this technique is modeling the uncertainty of the robot's sensors because they are, after all, uncertain devices.

"Demo-itis"

Unfortunately, LOLA had more in store for us after we settled the hard-drive problem. LOLA had performed well during the preliminaries, with the exception of a single-gripper failure. This failure happened, as Murphy's Law dictates, at the worst possible time—when everyone was watching. LOLA was about to pick up the first piece of trash when the gripper simply locked up. I was forced to move LOLA, using the joystick, out of the competition ring in shame while it was frozen in a ridiculous about-to-pick-up-trash position. Afterwards, I couldn't reproduce the error. I changed some stuff that I thought might be causing it and kept my fingers crossed going into the final, but as one might expect, the same problem occurred three times during the final. Fortunately, I had prepared a plan of action in case it did happen again and was able to recover LOLA after each occurrence and scrape by with enough points for the win.

We ran debugging runs off and on for two weeks after the competition and couldn't reproduce a single reoccurrence of the error. It wasn't until a photographer came out to take some pictures of LOLA that we reproduced the error and pinpointed its cause.

The gripper has infrared sensors that indicate when the gripper has opened or closed beyond its limits of travel. To prevent possible damage, the controller shuts down any axis that registers a limit sense. During the competition, photoflashes from cameras and infrared emitted from auto-focus mechanisms contained enough intensity to confuse the gripper controller into thinking that the gripper had reached a limit and should shut down.[2] The software didn't account for this situation and waited patiently for the gripper to finish its business, which it never did.

Unfortunately for LOLA, the most prime photo and video opportunities were when LOLA was picking up and depositing trash—which involves opening and closing the gripper. It's funny to think that the gripper was being bombarded with infrared exactly when and where it was most vulnerable. This problem has since been remedied in the latest revision of the NOMAD arm by using mechanical limit switches.

Acknowledgments

We express sincere thanks to the Kansas State team, which recognized our dire situation and loaned us its hard drive after LOLA's crashed. Luckily, their team had a NOMAD 200 with LINUX installed, which made recovery possible within the small amount of time available before the final. We also want to express sincere thanks to Keith Mason of Nomadic Technologies, whose expertise rescued LOLA's hard-drive information and got us back in the competition with time to spare. Of course, we would not have been able to compete without every bit of help we received in the last hours before the final. Thanks, guys!

Notes

1. Ironically, the author of this widely used algo-

rithm was on the University of Chicago team.

2. Video footage from the competition confirmed this theory.

References

Aloimonos, J. 1990. Purposive and Qualitative Active Vision. In *Proceedings of the First European Conference on Computer Vision,* 816–828. New York: Springer-Verlag.

Ballard, D. 1991. Animate Vision. *Artificial Intelligence* 48:57–86.

LeGrand, R. 1995. Position Estimation of Selected Targets. Master's thesis, Center for Robotics and Intelligent Machines, North Carolina State University.

Nourbakhsh, I.; Powers, R.; and Birchfield, S. 1995. DERVISH: An Office-Navigating Robot. *AI Magazine* 16(2): 53–60.

Swain, M., and Ballard, D. 1991. Color Indexing. *International Journal of Computer Vision* 7(1): 11–32.

Richard LeGrand received a bachelor's degree in both electrical engineering and computer science from Rice University in 1993 and is currently completing his Master's at North Carolina State University. As an undergraduate, he worked with robotics groups at Lawrence Livermore National Laboratory investigating toxic-waste cleanup, and he recently worked with the Intelligent Mechanisms Group at NASA Ames. His research interests include computer vision, controls, and uncertainty modeling.

Ren C. Luo is currently a professor in the Department of Electrical and Computer Engineering and the director of the Center for Robotics and Intelligent Machines at North Carolina State University in Raleigh. From 1992 to 1993, he was Toshiba Endowed Chair Professor in the Institute of Industrial Science at the University of Tokyo. He received his Diplom-Ing. and Ph.D. from Technische Universitaet Berlin in 1979 and 1982, respectively. Luo is a fellow of the Institute of Electrical and Electronics Engineers.