Landmarks for absolute localization

Jon Howell
Keith Kotay

Technical Report TR2000-364
Department of Computer Science
Dartmouth College
Hanover, NH 03755-3510
jonh@cs.dartmouth.edu

Abstract

For certain experiments in mobile robotics, it is convenient to eliminate positional estimation error in the interest of analyzing other parts of the experiment. We designed and implemented a simple, accurate scheme for encoding and recovering absolute position information. The encoding is a two-dimensional image printed on the plane of the floor, and the absolute position information is recovered using a downward-looking video camera mounted on a mobile robot.

1 Introduction

In any scientific undertaking, it is important to hold control variables constant so that one can examine the experimental variables carefully. In our experiments with mobile robotics, we had certain experiments that would have been facilitated if the variable of positional or odometry error could have been eliminated. This inspired us to design a laboratory tool that could do just that: provide simple, inexpensive, and accurate position information over the plane of the floor, eliminating positional uncertainty from those experiments where it intrudes as an irrelevant variable. The tool might also be useful in a factory, warehouse, or any other location that admits an engineered environment.

The idea is to print a visual pattern on the floor of the robots' environment. We imagine printing a pattern on large-format electrostatic plotters and adhering the sheets to the floor of our lab. On one of our mobile robots, we mounted a low-resolution black and white video camera pointed at the floor (see Figure 1). From any position in the room, the robot should be able to capture a single video frame and recover its position and orientation.

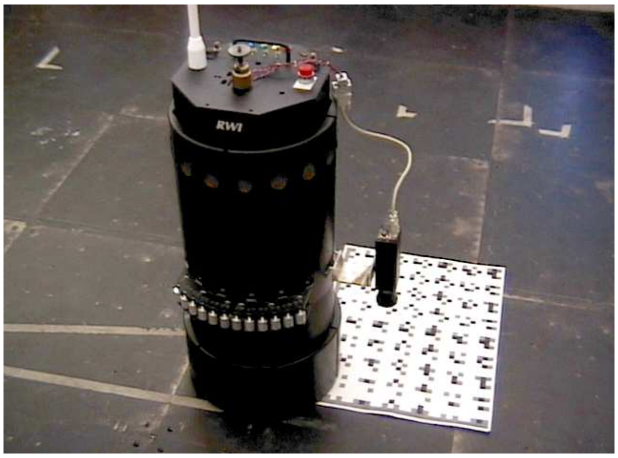

Figure 1: Our mobile robot Clyde captures an image from a sample region of the pattern.

We devised two solutions, described in the following two sections. The first solution requires an environmental image with three levels of grey; the second reduces the requirement to a binary image. We discuss future directions in Section 4.

2 The ternary pattern

In the sample in Figure 2, the ternary cells are arranged into 5x5 super-cells. Three of the cells are grey control bits, sixteen are black or white data bits, and six are unused (always white). Eight data bits convey information about the x dimension, and eight communicate the y dimension.

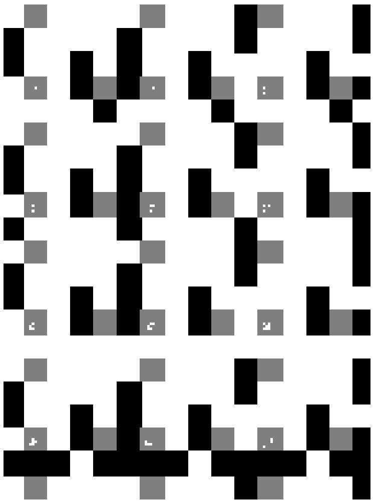

Figure 2: A sample of the 5×5 ternary pattern.

2.1 Analyzing an image

Figure 3 shows a sample frame captured from the robot's camera. In this section, we describe each of the steps required to recover the pose (position and orientation) information from the image.

First, artifacts from the capturing process are removed. The black band at the top of the image is cropped, and image is widened to correct the aspect ratio (so that squares in the pattern appear as squares in the image). Barrel distortion is a common artifact of inexpensive wide-angle lenses. It causes features at the edge of the image to appear smaller than identical features at the center [Tsa87]. This distortion is removed, producing an image that appears flat (Figure 4). The vector at the center of the image is the optical center of the camera; it is the position for which we will determine absolute pose. The robot's pose is thus a constant offset from that location determined by how the camera is mounted to the robot.

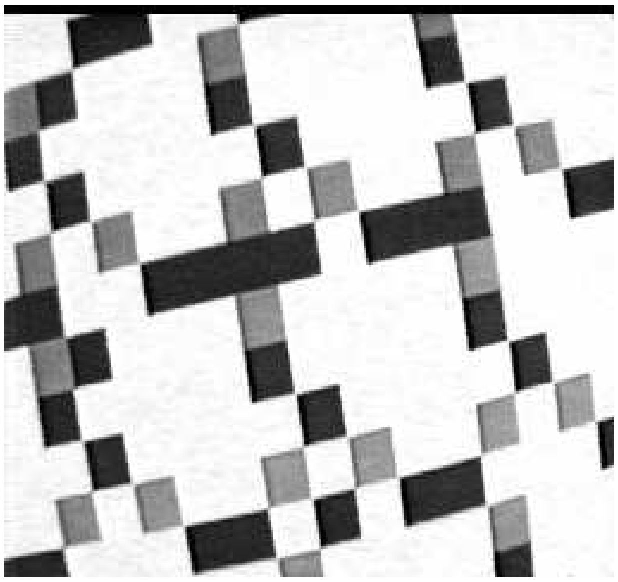

Figure 3: A raw image captured by our mobile robot Clyde with a downward-pointing video camera The image is 256×240 pixels.

Each pixel value in the image is thresholded to one of the three possible values white, grey, or black. The image is then convolved with an edge-detection filter that highlights edges of sharp contrast, as shown in Figure 5. The edge-detected image is cropped to a circle so that no orientation of line has a disproportionate contribution in the next step.

A Hough transform maps the points in the edge-detected image into a dual space in which each point represents a line in the original image [IK88]. The brightest points in the Hough map correspond to the lines in the original image that pass through the most edge-detected points.

The peaks in the Hough space represent a set of equations for lines, rather than the pixels in the raster image that appear as lines to the human eye. In Figure 6, we show the image from Figure 4 with the recovered lines drawn in. The slope of the lines tell us how far we are rotated from one of the four axes (or compass points); the line intersection nearest the origin (in robot space) is used to recover the offset of the robot's

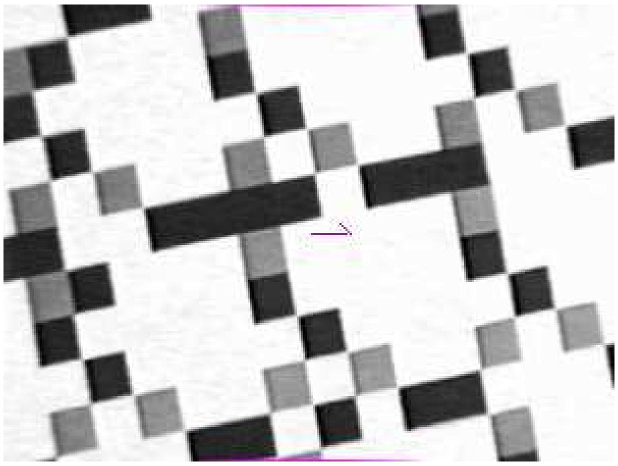

Figure 4: The image with the barrel distortion due to the camera lens removed.

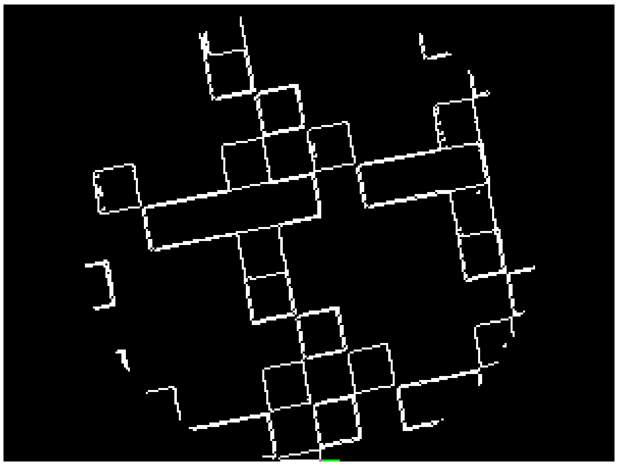

Figure 5: The image after applying an edge-detection filter.

origin from the ternary cell edges.

We apply both corrections to the image. We rotate it between 0 and 90 degrees to make the detected lines orthogonal to the robot (camera) axes, and we translate it on the order of the size one cell to put the corner of a cell at the origin (image center). The algorithm stores the applied corrections in a matrix that is used later to recover the original fine-grained position of the robot and camera.

The remaining task is to retrieve the coarse-grained information: the four-way rotational symmetry must be resolved, and we need to recover the "address" of the cell at the origin of the image. Figure 7 shows the aligned image. This particular image was rotated 80 degrees counterclockwise from Figure 4.

The center of each cell is averaged and thresholded to recover its ternary value (white, grey, or black). At this point, we have found a 5×5 supercell in the image, and we know our position accurately with respect to the supercell.

2.2 Control encoding

Each cell is black, white, or grey. Grey cells are *control* cells, used to identify the offset and orientation of a supercell in a field of cells. The black and white bits encode the address of the cell in binary.

The control cells uniquely identify the orientation of the image. In a canonically oriented supercell, control cells (**C**) are interleaved with data cells (d) in this way:

$$
\begin{array}{ccccc}
\mathrm{d} & \mathrm{d} & \mathrm{d} & \mathrm{d} & \mathrm{d} \\
\mathbf{C} & \mathrm{d} & \mathrm{d} & \mathrm{d} & \mathrm{d} \\
\mathrm{d} & \mathrm{d} & \mathrm{d} & \mathrm{d} & \mathrm{d} \\
\mathrm{d} & \mathrm{d} & \mathrm{d} & \mathrm{d} & \mathrm{d} \\
\mathbf{C} & \mathrm{d} & \mathrm{d} & \mathbf{C} & \mathrm{d}
\end{array}
$$

The lower-left control cell is called the "base."

The control bits are tiled uniformly across the floor pattern, so we can pretend that the control bits in the image wrap around the edges of the image; that is, we can do cell math modulo 5. In any orientation, the base control bit is the only one with the property that two control bits appear at cell $base + (3,0) \cdot T$ and $base + (0,3) \cdot T$, where T is one of the four 90-degree rotations. This test identifies the lower-left grey cell in Figure 7 as the control cell, and further indicates that the image is 180 degrees rotated from the canonical orientation. This rotational correction is applied (and stored for later recovery), giving the following canonically-oriented supercell:

$$
\begin{array}{ccccc}
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & \mathbf{C} & 1 & \mathbf{C} \\
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & \mathbf{C}
\end{array}
$$

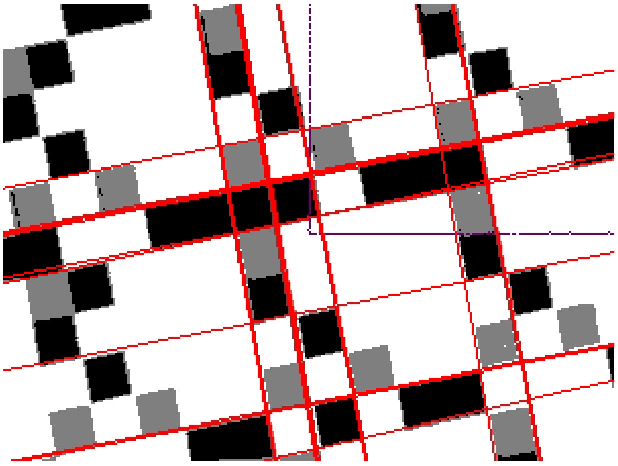

Figure 6: The image with the detected lines superimposed. (The two orthogonal segments are the axes in robot space, not spuriously detected lines.)

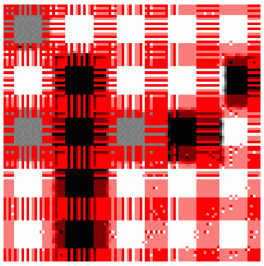

Figure 7: The image aligned to the axes of the camera. The centers of the cells are sampled for their value; hash marks indicate the intermediate regions where the pixel values are ignored.

We know that the upper-right **C**is the base control bit, so we are actually seeing the junction of four supercells:

$$
\begin{array}{cccc|c}
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & \mathbf{C} & 1 & \mathbf{C} \\
\hline
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & \mathbf{C}
\end{array}
$$

2.3 Data encoding

Now we explain how the data bits are laid out in a supercell. The x bits occupy two rows of the supercell, and the y bits occupy two columns. The most significant bits of the x value run down the center of a supercell, and the most significant bits of the y value run across the center. The least significant bits run across the top row and down the right column.

$$
\begin{array}{ccccc}
y_3 & y_2 & x_7 & y_1 & y_0 \\
\mathbf{C} & & x_6 & & x_2 \\
y_7 & y_6 & x_5 & y_5 & y_4 \\
 & & x_4 & & x_1 \\
\mathbf{C} & & x_3 & \mathbf{C} & x_0
\end{array}
$$

In any 5×5 view of the floor pattern, we will be able to see at least nine cells of some supercell. Our image is represented again below with the assignments of values to each variable according to Figure 7; the upper-left supercell is the most visible one, so we will extract its address using information from all four of the visible supercells.

$$
\begin{array}{ccccc}
y_6 = 0 & x_5 = 0 & y_5 = 0 & y_4 = 1 & y_7 = 0 \\
 & x_4 = 0 & & x_1 = 1 & \\
 & x_3 = 1 & \mathbf{C} & x_0 = 1 & \mathbf{C} \\
y_2 = 1 & x_7 = 0 & y_1 = 0 & y_0 = 1 & y_3 = 0 \\
 & x_6 = 0 & & x_2 = 0 & \mathbf{C}
\end{array}
$$

Because x values run down columns, x_7 and x_6 that we can see in the neighboring cell below must be the same as those we cannot see in our target cell because they appear above x_5. Likewise, x_2 from the cell below corresponds with x_1 and x_0. Since both columns of x bits run through our target cell, we can simply concatenate them together into an eight-bit value that gives the x-coordinate of this supercell as $00001011_2 = 11_{10}$.

The y-coordinate is a bit trickier. Again, y_7 from the neighboring supercell to the right can still be used, because that supercell has the same y-coordinate. But the least-significant y bits appear in the supercells in the row below. Therefore, we extract y_3, y_2, y_1, y_0, add one, and perform a bitwise-and with 1111_2 to figure out what the least-significant y bits in our target cell would be if they were visible. In this case, $y_3, y_2, y_1, y_0 = 0101_2 + 1 \mod 1111_2 = 0110_2$. Combined with the most significant bits, we arrive at a y-coordinate of $00010110_2 = 22_{10}$.

Cells in our implementation are 1 cm $\times$ 1 cm, so the supercell at $(11, 22)$ is $(55, 110)$ centimeters from the world-space origin. We add $(1, -2)$

centimeters to account for the offset of the base cell: the view was not centered over the supercell, but over a point 1 cell to the right and 2 cells below center.

2.4 Recovering world-space coordinates

At each step of this process, we have recovered a matrix which transforms the image from robot coordinates toward world coordinates. The individual transformations are listed here:

- A rotation orients the cells orthogonal to the axes of the image.
- A translation usually less than one cell in magnitude aligns the cell boundaries with the center of the image.
- A 0, 90, 180, or 270 degree rotation orients the control cells canonically.
- A translation of up to four cells accounts for the base control cell not appearing in the lower-left corner of the view.
- A translation by multiples of five cells (supercells) accounts for the address encoded in the measured supercell.

The resulting transformation in SE(2) describes absolutely the robot's position and orientation in world coordinates. Notice that the resolution is indeed finer than one centimeter (cell size); since the first two transformations encode pixel-level details, the resolution of the algorithm is limited only by the resolution of the camera and the success of the hough transform in accurately recovering lines in the edge-detected image. The latter has proven to be very robust, since our images are composed of features oriented at two orthogonal angles.

2.5 Characteristics and limitations

The ternary supercell encoding given here supports eight bits of address in each dimension, giving it a scope of 256×256 supercells. Used with our $1cm$ cells, it could cover a floor 12.80 meters on a side. By extending the neighboring cell recovery argument used above to recover $y_3 \ldots y_0$, the six unused cells could be stuffed with data, covering a space 102.4 meters on a side.

Clearly the supercells could be enlarged to store more bits and address a larger space. Enlarging the supercells requires a tradeoff: either the camera must have a larger view of the floor, or a higher-resolution view, or the algorithm may become less robust as the cells in the image are represented with fewer pixels.

Although we have not performed quantitative experiments with this algorithm, its performance in our ad-hoc experimentation was quite robust.

3 The binary pattern

One of the unattractive features of the algorithm above is that it requires ternary cells. Ternary cells "waste" information, because only three of the cells (the control cells) can possibly contain the third value; the others are always binary. Therefore as n grows, the information density (as a fraction of bits) limits to $log_2 3$.

The diagram in Figure 8 gives the binary pattern we designed. The strip of black bits across the top and left are a control channel to identify the boundaries of the supercells. The diagonal strip of white bits ensures that no other bit pattern will result in the accidental formation of an n-bit long sequence of black bits that could be mistaken for the black boundary control channel. The black bit at upper right and the white bit at lower right serve to disambiguate the four rotational symmetries of the pattern. Once we find the black control channels, the four control bits at the corners will always be one black and three white.

The remaining spaces can be filled with data bits. In this example, the 6×6 grid provides space for an 18-bit data channel, which can address a 512×512-supercell grid. The formula for the number of data bits in an $n \times n$ supercell is:

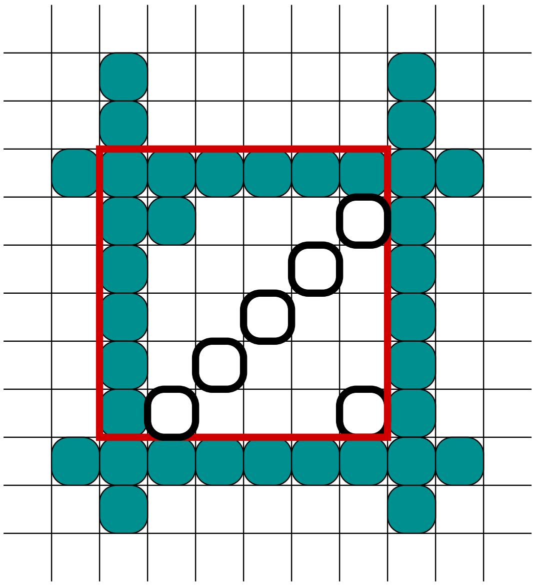

Figure 8: A supercell encoded in a binary (two-level) image. The thick border delimits the supercell.

supercell width n	data bits (total) $n^2 - 3n$
4	4
5	10
6	18
7	28
8	40

Have we been more careful with the information density? A ternary supercell of size $n \times n$ has the following information density, computed in terms of binary bits:

$$\frac{(n^2 - 3) + log_2(4n^2)}{log_2 3^{n^2}}$$

The first term in the numerator reflects the $n^2 - 3$ available binary data cells, and the second term accounts for the rotational and translational symmetries resolved by the control bits. A binary supercell of size $n \times n$, in contrast, has the following information density:

$$\frac{(n^2 - 3n) + log_2(4n^2)}{n^2}$$

Both functions are plotted in Figure 9. Cases where $n \leq 2$ are degenerate. In cases $2 < n \leq 6$, the ternary supercells are more efficient. After $n \geq 7$, however, the functions cross and approach their separate asymptotes; the binary function approaches 1 (efficient encoding), while the ternary function limits to $log_2 3$, since it wastes one of three potential values in most of its cells.

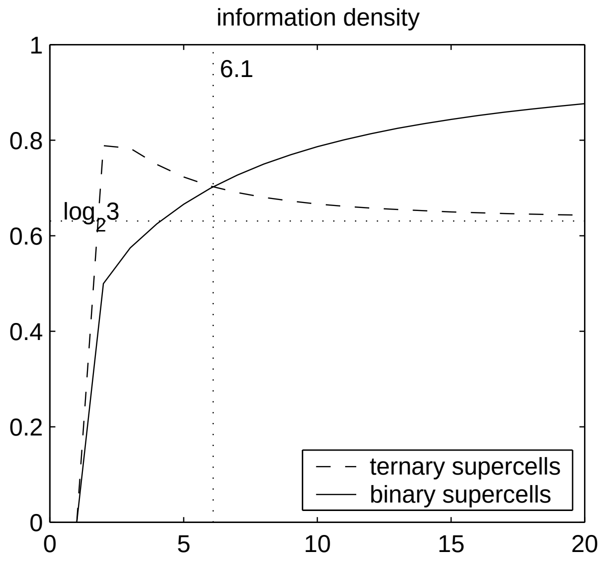

Figure 9: The control channel used in binary supercells (solid line) is less wasteful of encoded information density than that in ternary supercells.

4 Future directions

Although the photos above are genuine snapshots of the working system, we never built a whole floor using our encoding for use in the lab as a substrate for other experiments. We have, however, considered some interesting extensions.

Error-check bits. It would be useful to allocate some of the data in a supercell to encode error-checking bits, to protect against mistakes in the long chain of image transformations. Error-correcting may be less practical, since the encoding is not generally robots against errors in the control channel.

Discrete sensors. It would be nice to reduce the hardware requirements for this system.

One approach would be to use a strip of eleven reflective optical sensors spaced 1/2 cell apart, and sample them every 1/2-cell using the robot's shaft encoders to measure local forward motion. The Nyquist limit suggests that this should be enough samples to recover a usable image. However, we may need more clever tricks to recover rotational information. Perhaps we could record far more samples in the rolling dimension, increasing our resolution enough in that dimension to infer the slopes of lines the robot crosses at an angle.

Spread spectrum. We do not know much about it, but it seems like a way to achieve error checking and correction using redundancy without spatial control channels. It may enable us to do so using two level bits, and perhaps even allow us reduce the number of input sensors to one by encoding information in both horizontal and vertical frequency channels.

Imagine that (x, y)-tuples are encoded in rows in one "image", and in columns in another. A third and fourth image each encode a strictly-increasing signal along the x and y dimensions, respectively. All four signals are encoded in separate spectral bands. The floor image is the sum of all four channels. We are not sure whether this sum can be done in a two-level image, but it would seem so.

Now, a sufficiently-long line segment of any orientation will read at least one (x, y)-tuple channel, and at least one strictly-increasing signal. The tuple directly communicates the robot's position. The arctangent of the relative "lengths" of the two increasing signal channels gives the angle of the line segment's path. That line-segment could be provided by an input device as simple as a single reflective optical sensor, perhaps using the robots' wheel shaft encoders to more accurately normalize the signal to spatial distance.

Less intrusive environment modifications. The previous item dealt with how to reduce the on-robot requirements. Here we consider ways to reduce the environment requirements; that is, alternatives to printing a pattern to cover the entire floor of the laboratory.

- Tileability. One complaint is that we cannot just produce one kind of floor tile to install repeatedly. We see no obvious way around this problem, since any repeating tile necessarily contains no locally-visible, globally-unique information. One could imagine a specific arrangement of a small set of tiles, but in the limit this is simply our specific arrangement of two or three very small tiles (black, white, and perhaps grey).

- Projection. One possibility is to project the pattern onto the floor, a wall, or the ceiling using infrared light, and pick it up using a CCD camera with an infrared-pass filter. We would need to deal with shadows (the robot's own shadow could interfere with a floor projection) and occlusion (the light fixtures could interfere with a ceiling projection). With a discrete- or single-sensor encoding, we could focus the image onto a plane at some constant height above the floor, and have an upward-pointing sensor on the robot pick up the projected image as the robot moves.

- Stochastic labels. Another approach would be to supply the laboratory with a big stack of labels, each with a spread-spectrum or bar code tile. These labels would be stuck to the floor at random intervals. A robot would get an absolute fix whenever it happened to pass over a label; it would use dead-reckoning in between labels. If it became disoriented, it would use a wall-following or space-filling strategy to attempt to cover ground until it passed over a label. An initial calibration would be required to establish known positions and orientations for each of the labels. Another advantage of this approach is that damage to the floor is trivially repaired: apply a new sticker, and calibrate it.

5 History

This document describes work done in the Dartmouth Robotics Laboratory in April of 1997 and

August of 1999. It was previously "published" as a web page, but we thought it would make sense to document it more permanently.

Keith initially framed the problem around April of 1997. We first approached it by trying to generate fields of black and white cells with the property that a given section of the image would be unique with respect to any translation. We initially attempted to generate the field in a brute-force way, by generating new random bits and checking each against every existing bit pattern. Not surprisingly, brute force was too slow. Perhaps it could have been improved with more thoughtful data structures. Then we came up with the deterministic ternary layout.

In August of 1999, we revisited the problem long enough to generate the binary layout. Upon hearing about the problem, attendees at the Workshop on Algorithmic Foundations of Robotics in March 2000 gave us positive feedback, which led to our publishing this technical report.

6 Summary

We have presented optical encodings that have a specific property useful to localization: any $n \times n$ region of the encoding contains enough information to unambiguously resolve the location of the region in the three degrees of freedom of the plane. We described both our initial ternary encoding, and a revised two-value encoding that is more information-frugal. We also detailed the process used to extract the position information from a sampled image. The encoding allows us to recover positional information up to the resolution of the sensed image, not just the resolution of the encoded cells.

The encoding has practical applications in any environment where an untethered agent roams over a large, "paintable" surface, and benefits from localization with only local sensing of the painted surface. We described promising future directions for our approach to landmarking.

Acknowledgements

Thanks to Daniela Rus for providing the great infrastructure of the Dartmouth Robotics Laboratory.

References

[IK88] J. Illingworth and J. Kittler. A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87–116, October 1988.

[Tsa87] R.Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, 1987.