Fast and Accurate Vision-Based Pattern Detection and Identification

James Bruce (jbruce@cs.cmu.edu)
Manuela Veloso (mmv@cs.cmu.edu)
Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213, USA

***Abstract*— Fast pattern detection and identification is a fundamental problem for many applications of real-time vision systems. The desirable characteristics for a solution are that it takes little computation, localizes a pattern robustly and with high accuracy, and can identify a large number of unique pattern identifiers so that many patterns can be tracked within a field a view. We will present a system that can accurately track a broad class of patterns both accurately and quickly, when using a suitable low level vision system that can return calibrated coordinates of regions in a image. Both pattern design and the detection algorithm are considered together to find a solution meeting the above criteria. Along the way, assumptions are verified to make informed choices without relying on guesswork, and allowing similar systems to be designed on a solid experimental and statistical basis.**

I. Introduction

Object identification and tracking is one of the most important current applications of machine vision. While much research has been directed to the topic of object detection and tracking for general objects, such as faces, cars, and doors. While much progress has been made, many of the algorithms require substantial amounts of processing and are less accurate than can be achieved with patterns specifically designed for detection. Thus many of the current applications of object detection and tracking using machine vision use customized patterns, such as factory automation or package routing systems. Although the use of customized patterns prevents the system from being usable in every environment, in many environments specifying a pattern to be used is not a major limitation. Of course, if the pattern can be specified in order to suit the capabilities and limitations of the machine vision system and detection algorithm, in return we expect high performance from the system. Specifically, the detection for the pattern should be fast and highly accurate; especially when compared to more general object detection and tracking systems. For low level vision, we will employ the freely available[1] CMVision[2] library we have developed for earlier projects. It performs color segmentation and connected components analysis to return colored regions at real time rates without special hardware or dedicating the entire CPU to the task.

This research was sponsored by Grants Nos. DABT63-99-1-0013, IIS-9900298. The information in this publication does not necessarily reflect the position of the funding agencies and no official endorsement should be inferred.

Fig. 1. The CMDragons'02 Robots. Note the tracking and identification pattern on the top of each of the robots.

The environment in which most of this work has been done is the RoboCup[8] F180 "Small Size" League, where robots up to 18cm in diameter play soccer on a 2.8m by 2.3m carpeted soccer field. The game is played with two teams with five robots each, and uses an orange golf ball for the ball. One team must have a 40mm blue colored circle centered on the top of its robot while the other team must have a 40mm yellow patch. Teams may add extra patches and colors to the top of their robot to aid in tracking, so long as those colors are differentiable from the three standard colors (orange golf ball, yellow team patch, and blue team patch). The robots from our team, CMDragons'02[3], can be seen in Figure1. Each team can control its robots either onboard, or offboard, and cameras are allowed to be placed above the field. Thus, most teams use a single overhead camera, with an offboard PC interpreting the camera signal and sending commands to the robots via a radio link.

This environment thus poses a tracking problem for up to 11 small objects in known planes (in this case the possibly different, but known, heights above the ground plane).

Few other environments currently demand accurate, multiple pattern detection at very high speed, one such environment is Virtual or Augmented Reality. For these environments, patterns are tracked in order to localize head mounted displays and locate objects in the physical environment that are mapped into the virtual environment[5], [4]. Detection must be fast and accurate to minimize observable lag and jitter in the visualization. Due to work in these two environments, fast, accurate, multiple pattern detection has become better understood and more practical. Thus we expect many more applications for such tracking systems in the future.

Another aspect that makes the RoboCup F180 environment challenging is the high speeds of the tracked objects, since the robots move quite quickly relative to their size. Robot speeds peaking in excess of 2m/s are not uncommon, and ball speeds (via robot kicking mechanisms) can reach up to 5m/s. Thus we feel this is a good testbed for a tracking system. Two other vision tracking and identification systems for the F180 league have been described in [9] and [7]. Each describes a working system used by a team, but neither motivates the choice of pattern by a thorough analysis of the underlying feature error, or attempts to generalize detection to other similar patterns in order to compare their performance. In this paper we will outline the choices and trade-offs made in designing an identification and tracking pattern by gathering real and simulated data at each step so that informed tradeoffs can be made. We hope that this will help others to implement similar high performance tracking systems both within RoboCup and in many other environments where a similar problem exists. Such designs should not have to rely on any guesswork.

In the first section, we will motivate the type of patches chosen from which to build patterns, and examine their error distributions when viewed from a camera. In the second section, we will describe several common patterns and a broad class that includes most of the patterns. We will motivate the use of this most popular class for its simplicity and accuracy, and for which an efficient generic detection algorithm can be created. In the following section, the performance of several such patterns will be examined in simulation. Finally the best performing pattern from simulation will be evaluated on a real-time vision system.

II. Single Patches

In order to build up a detection pattern, we must have some simpler building block on which to build. We will use simple colored patches whose position can be calculated accurately. To detect orientation, multiple patches can be employed. For detection, the simplest approach to take for a single patch is to use a simple regular geometric shape of a single color. Detection in the vision system can be carried out on a binary or multiclass threshold image from which connected regions of common color class can be extracted. This approach is common, and is known to be quite efficient, so this is the approach we will use. The next variable to determine for a patch shape. We chose circles, because they guarantee rotational invariance, and analytical corrections for the projective distortions of their image centroids are known[6]. In addition, they are compact, minimizing the length of the border with other regions, where thresholding is most difficult. In experiments, other regular shapes such as squares, hexagons, and octagons, perform roughly on par with circles, but they do not offer any benefits in light of the analytical guarantees outlined above.

The more difficult problem to answer is what size of patch to use. When the dimensions of the overall pattern are known, this still leaves the question of whether it is better to have a pattern with a few large patches, or more patches where each is of a smaller size. To address this, we created a test setup where a small moving platform would carry three different sized white patches 2 meters across the field of view of a camera looking down from 3 meters. The platform moved at a slow constant speed (about 23mm/sec) allowing large amounts of data to be gathered from a variety of locations across the field. Using this setup, we gathered positional at 30 samples/sec for 40mm, 50mm, and 60mm circles. A total of 5 runs were gathered, each one having about 2570 data points. As a convention, we labelled the dimension along the primary direction of travel as x, and the dimension perpendicular to the direction of travel as y.

Although there is no ground truth from which to measure true error, the error can be estimated by smoothing the data with a large Gaussian kernel ($\sigma = 10$) and then comparing single samples with the smoothed version of the signal. The aggregate errors appear to follow a Gaussian distributions quite well, as can be seen in Figure 2. However, more outliers occurred than would be expected in a pure Gaussian distribution, and the variance seemed to change noticeably between runs, and even varied over different segments in the course of a single run. The most sup rising result, however, is that the size of the patch had very little effect. The overall standard deviations were around 0.52mm in both x and y for all sizes with only slight (although significant) variation. The cumulative distributions of absolute error in x and y are shown in Figure 3.

TABLE I

The estimated standard deviations and 95% confidence intervals by patch size.

Diameter	σ_x	σ_y
40 mm	$0.553, [0.546 - 0.561]$	$0.473, [0.467 - 0.480]$
50 mm	$0.543, [0.535 - 0.550]$	$0.504, [0.497 - 0.511]$
60 mm	$0.489, [0.482 - 0.496]$	$0.533, [0.526 - 0.541]$

The estimated standard deviations for each patch size can be found in Table I, along with 95% confidence intervals for the standard deviation. For hypothesis testing, we used the non-parametric Wilcoxon signed-rank test due to its robustness to outliers and lack of strong assumptions about the distributions being tested. The significant results (in all cases, $p < 0.0001$) were that along the direction of travel (x), the 60mm diameter circle had significantly less error than both

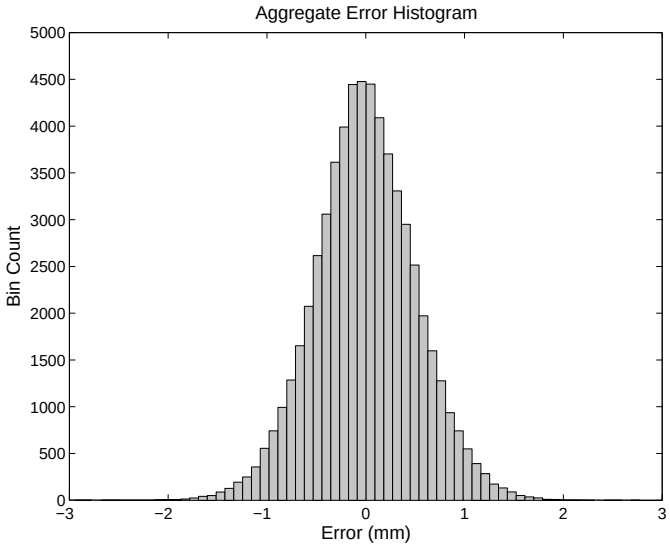

Fig. 2. Aggregate error distribution for all samples including all three patch diameters.

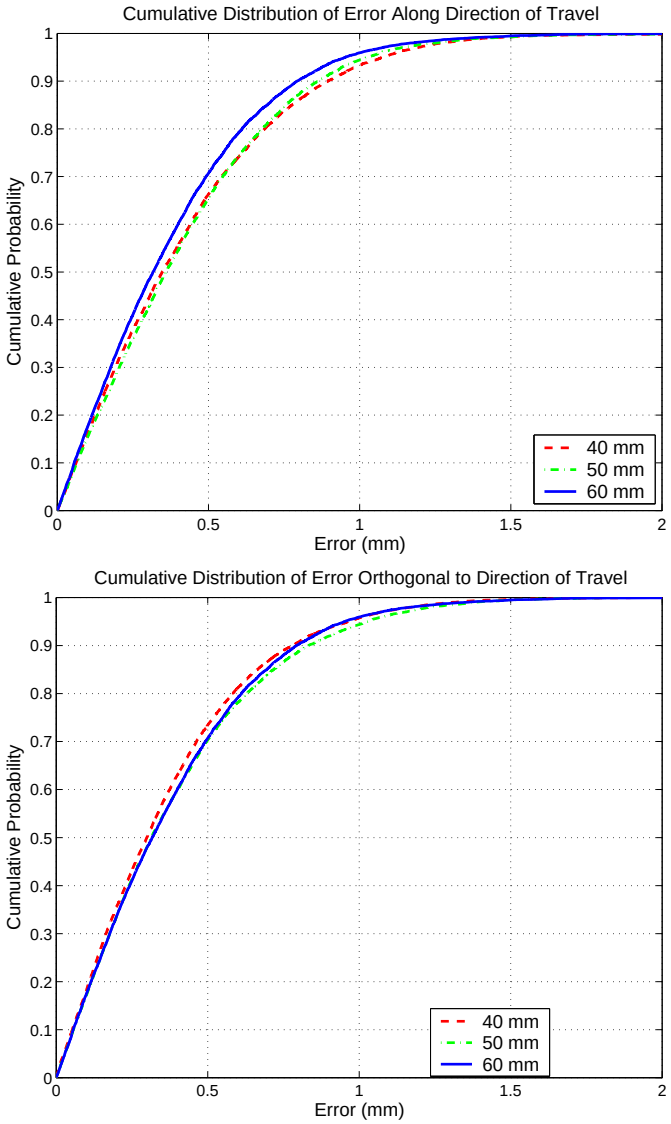

Fig. 3. Cumulative distributions of absolute error. Note that patch size does not have a large effect on error. Along the direction of travel, the largest patch size decreases error somewhat, but does worse than the smallest patch size perpendicular to the direction of travel.

the 50mm and 40mm patches. Perpendicular to travel (y) however, error *increased* with patch size, with all means being significantly different. Combined error in x and y indicated that the 50mm patch was slightly worse than the other two, with $p = 0.02$ against each of the other patches. Given the number of data points (over 10,000) however, we do not consider a difference at $p = 0.02$ ultimately conclusive. In addition, the difference in error from best to worst is less than 5%, which is much less variation than we expected since the largest patch has 2.25 times the area of the smallest patch.

Thus the conclusion we can draw are that patches should be large enough they can be detected reliably, but need not be made any larger for purposes of accuracy. This is important in that it is contrary to conventional wisdom about region detection. It seems that other factors, such as quantization due to pixels, play a larger part in determining the error than the area of the region. As we will show later, we can do somewhat better by adding more patches rather than using fewer patches and increasing their size.

III. Patterns and Detection

Now that the basis for choosing patches has been established, we can use this knowledge to evaluate tracking patterns. One source for many different ideas are the various patterns used by the over 20 teams in the RoboCup F180 League. The rules for the patterns on the top of the robots in that league has naturally led to many tracking and identification patterns being tried. Examples of some of the more popular designs can be seen in Figure 4. The approaches taken thus far generally fall into one of three broad categories. By far the most common type is like that shown above, which we call *patch based*, where in addition to the team marker patch in the center, one or more additional circular or rectangular patches are used to encode position and orientation. Patch based systems have the advantage that position and orientation detection can combine several features (patches in this case) with sub-pixel accuracy. One alternative to this is to have a key patch marking the center of the pattern, surrounded by radial "pie slices" of two or more colors. Orientation and identification can be performed by scanning at some constant radius from the central patch [7]. Unfortunately, by depending on features (color edges in this case) that are difficult to quickly detect with sub-pixel accuracy, it is difficult to get very accurate orientation using this method. Another type of tracking pattern that has been tried is to use a central patch with an nearby line feature. By finding the edge points of that feature, a least squares fit can be made to get an accurate orientation measurement. Unfortunately, both these classes of patterns do not offer a straightforward way to improve the

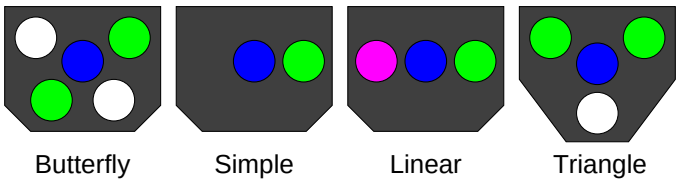

Fig. 4. Examples of common tracking patterns from the RoboCup F180 environment.

position estimate. For a given pattern size, we would like to make the most of the space. Ideally, we would like all of the patches to contribute to position, orientation, and identification detection. In this regard, patch based systems tend to do quite well, which has led to them becoming the most common class of pattern used for tracking.

If we look again at Figure 4, we can notice similarities that can aid in creating a generic detection algorithm. All are keyed by a colored patch (in the center of the pattern) indicating the presence of a pattern. Each patch occurs at some unambiguous angle radially from the key patch. Thus there is a distinct circular ordering of the non-key patches that can be calculated even in the presence of moderate noise. This means a generic detection algorithm can start by searching for the key patch, and then detecting and sorting the additional patches radially. All that needs to be done after that is to find the rotational correspondence of the additional patches with the geometric model of the pattern. In the case of the Simple pattern, the correspondence is trivial. For the Linear and Triangle patterns, the colors of patches can be used to disambiguate geometrically similar correspondences. Finally, in the case of the Butterfly pattern, geometric asymmetry can be used to find the rotational correspondence, assuming the distances between patches can be measured accurately enough. As shown in the previous section, this boils down to the difference of two Gaussians. When we consider the standard deviations determined in the previous section, distances that differ by over 10mm should be differentiated correctly with very high certainty. Using geometric asymmetry offers the benefit of freeing up the patch colors to encode only identification, rather that both identification and rotational correspondence. This gives the butterfly pattern (or any other geometrically asymmetric pattern) an identification advantage over symmetric patterns, as shown in Table II.

TABLE II

THE NUMBER OF UNIQUELY IDENTIFIABLE PATTERNS THAT CAN BE DETECTED USING A CERTAIN NUMBER OF COLORS (EXCLUDING THE KEY PATCH AND KEY PATCH COLOR)

Pattern	2 colors	3 colors	4 colors	n colors
Butterfly	16	64	256	4^n
Simple	2	3	4	n
Linear	1	3	6	$n \cdot (n-1)/2$
Triangle	2	6	12	$n \cdot (n-1)$

Once the correspondence is established, the position and orientation estimates must be made. What we would like is to get near optimal detection but without resorting to iterative methods or other time consuming operations. Here we take a simple approach that turns out to be not only fast but in practice nearly indistinguishable from optimal formulations. First, the mean location of the patches is determined. This is an optimal estimate, although for many patterns this location is offset from the actual location we want to report (so it is not an optimal estimator of that point). After this mean position has been determined, we get displacement vectors between a pre-specified set of "orientation pairs" from the patches. These pairs should be well separated (because error decreases with distance), and different pairs that share a patch should be as orthogonal as possible (to avoid correlated errors). For the butterfly pattern, we use vectors between the four non-key patches. For the Triangle pattern we Similarly take the triangle edges formed by the pattern's three external patches. For the Linear and Simple pattern, only one nearly orthogonal pair exists. For the Linear pattern we choose the longest option of the opposite patches because this will minimize the error compared to the two shorter vectors that include the central key-patch. After the separation vectors are determined, they can be rotated into a consistent frame of reference because their angle relative to forward is known from the model of the pattern. Once all the vectors are lined up by the model, they can be added to form a single vector, and the arctangent calculated to get the angle measurement.

The motivation for adding the vectors comes from the observation that the angular error of a vector is roughly proportional to the separation of the patches when the separation distance (d) is much larger than the positional standard deviation (σ), or:

$$\sigma_\theta \approx \frac{\sqrt{2\sigma^2}}{d}$$

The term $\sqrt{2\sigma^2}$ comes from the subtraction of two Gaussians (since the separation distance is large this is roughly a 1D subtraction). The division by d is the result of arctangent being linear near the origin. In practice, we've found this approximation works well when $d > 10\sigma$. Finally, once the angle estimate is made, we can use this to project the mean of the patches to the coordinates of the patch that are to be reported (normally the origin of the patch model coordinate system).

For comparison, we also derived an iterative Maximum Likelihood (ML) estimation method that co-optimizes position and angle estimates assuming Gaussian positional error for the patches. Its full derivation is omitted here for brevity. First, it is a well known fact that minimizing sum-squared-error in 1D is identical to maximizing the log likelihood (and thus likelihood) of samples from a Gaussian error distribution. Since in the 2D case variances can be added, this correspondence carries over into the 2D case. Thus by minimizing the sum-squared-error of the measured position of patches from their model positions given the estimated pattern position and orientation, we can obtain an ML estimate. Thus for a pattern with n patches, we define the current estimate of

robot position as r, marker locations as $v_i \ldots v_n$, and patch locations from the pattern model as $p_i \ldots p_n$. Then we have following derivatives for sum-squared-error E:

$$\begin{aligned} s &= \sin(r_\theta) \\ c &= \sin(r_\theta) \\ x_i &= r_{ix} + cp_{ix} - sp_{iy} - v_{ix} \\ y_i &= r_{iy} + sp_{ix} + cp_{iy} - v_{iy} \end{aligned}$$

$$\begin{aligned} \frac{\partial E}{\partial r_{ix}} &= \sum_{i=1..n} 2x_i \\ \frac{\partial E}{\partial r_{iy}} &= \sum_{i=1..n} 2y_i \\ \frac{\partial E}{\partial r_\theta} &= \sum_{i=1..n} 2x_i \cdot (-sp_{ix} - cp_{iy}) + 2y_i \cdot (cp_{ix} - sp_{iy}) \end{aligned}$$

These partial derivatives, along with the obvious implementation of the error function itself, can be used to create an iterative ML estimation method using Newton's method.

IV. Pattern Comparison

In order to evaluate the four patch based patterns introduced earlier, a small simulator was created that would generate patch positions using a Gaussian error model for a pattern at random positions and orientations. Then the detection algorithm was run on the patches, and the resulting position and orientation measurements compared to the true values used to generate the input patches. The results for the simulation are shown in Figure 5, plotted as pattern position and orientation standard deviation vs. input patch positional standard deviation. Each data point was generated from 100,000 simulated detections. One can easily see that multi-patch patterns have a distinct advantage for both position and angle measurements. The Simple pattern can fair no better than a single patch using the generic detection algorithm described in the previous section. It could perhaps benefit more from maximum likelihood detection, but this would make detecting the Simple pattern slower than detecting the more complicated patterns. The Butterfly pattern has the most accurate position estimation, followed by Triangle and Linear at somewhat decreased accuracy. For angular error, the Butterfly pattern again shows the lowest error, with Triangle close behind. With their multiple patches allowing several well separated orientation pairs to be used, they both perform much better than Linear or Simple, each of which only have a single orientation pair. Linear fares better than the Simple pattern because the separation distance for its orientation pair is twice that of the Simple pattern.

Since the Butterfly pattern worked best for both positional and angular error in simulation, we decided to make further tests to evaluate its performance using the iterative maximum likelihood detection and then measure the pattern's performance on a real vision system. To compare our generic detection algorithm with the ML estimate, we ran each of the

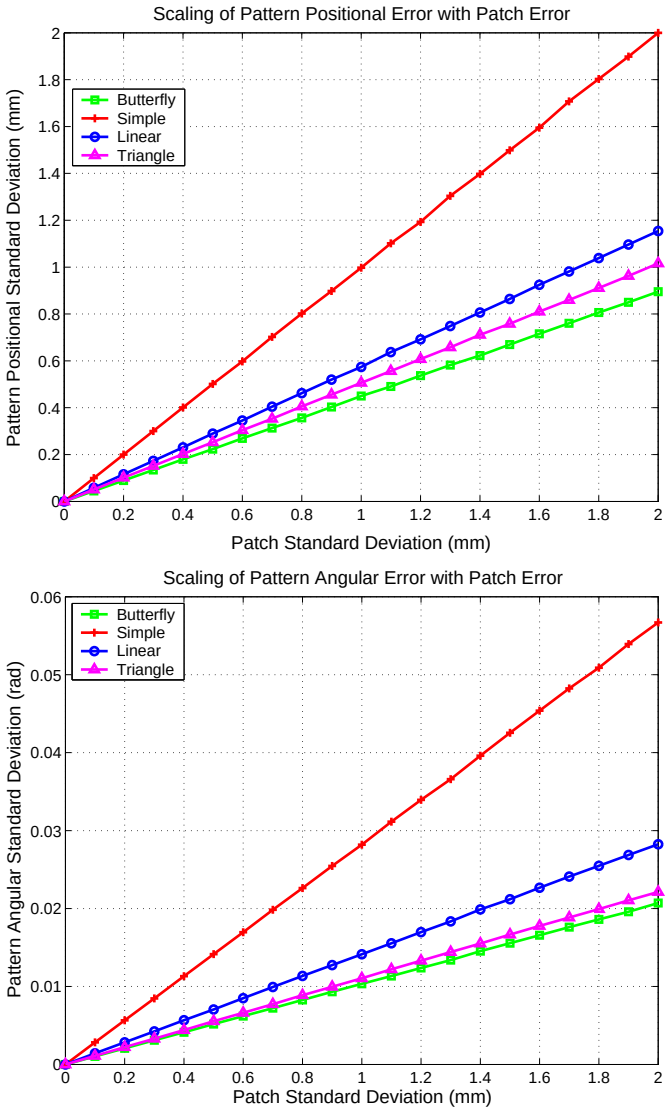

Fig. 5. Comparison of the positional and angular error of different patterns as the error of individual patches vary. For relatively small patch standard deviations, a linear relationship exists between the two, although the number of patches and layout of the pattern vary the factor.

detection methods in simulation. Even after 100,000 samples, no statistically significant differences could be detected with a Kolmogorov-Smirnov test, leading to the conclusion that at least for complicated patterns, the simple detection algorithm was indistinguishable in terms of accuracy from the ML estimation.

Finally, we evaluated the Butterfly pattern on a real vision system. We ran 5 runs similar to those for single patches but this time with a pattern being tracked rather than individual patches. The overall standard deviations were $\sigma_x = 0.3766$mm, $\sigma_y = 0.3432$mm, and $\sigma_\theta = 0.0070$rad. This was significantly better than a single patch ($p < 0.0001$), although not as low as predicted by simulation. The error was about 70% higher than predicted, which is most likely explained by some correlation in the patches' error (such as

error from camera jitter). The pattern error was more consistent with a patch standard deviation of around 0.8mm, and thus could also have been due to outliers affecting the measurements.

Another possible problem (but one that is easily measurable) is correlation of errors over time. Typically filters assume that all readings are independent measurements, however this may not be the case for some sources of error. In Figure6, we show 2D scatter plots of adjacent readings for a single patch (top) and for a full pattern (bottom). The single patch shows structure indicating that errors are likely to repeat (the center diagonal stripe) or jump up or down by a fixed amount (the upper and lower diagonal stripes). As best we could determine, this appears to be due to the binary color segmentation; As the patch moves across the camera image, pixels switch from background color to patch color (and back to background) abruptly, changing the location of the centroid by fixed amounts. The full pattern (bottom) does not display this structure (most likely because combining 5 patches made the structured error of individual patches small enough to make it unnoticeable). However the plot is still not an unbiased circular cloud, so adjacent readings are still somewhat correlated. With a few time steps separating readings, no observable correlations are present. Thus, assuming measurements are independent for patterns seems to be a reasonable simplification, although increasing the standard deviation to a more conservative estimate may be prudent. Assuming independence for patches may be more problematic, so for tracking single patches the extra complexity of modeling error correlation may be necessary.

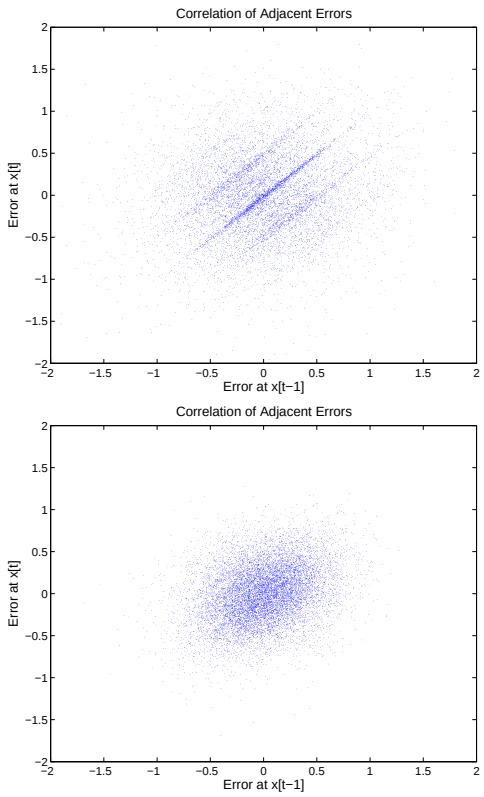

Fig. 6. 2D scatter plots of adjacent readings for single patches (top) and for a full Butterfly pattern (bottom). Uncorrelated readings would show up as a circular 2D Gaussian cloud. Note the structure in the plot for a single patch, and the unstructured but non-circular distribution for the full pattern.

V. Conclusion

We presented the derivation of an efficient and highly accurate detection algorithm along with an analysis of the performance of many different patch-based patterns. We first looked at the performance of single patch detection, noting that size, although important for robust detection, does not have a large effect on the accuracy of the positional measurement of a patch. We presented a fast patch based detection algorithm along with an iterative ML variant, which perform similarly in terms of accuracy. We compared several patterns in simulation to find out how accurate their detection scaled with the error of the patches form which they were made. We then tested a pattern on a real vision system with positive results, and examined the assumption of independence on which higher levels of an object tracking system rely.

We hope that this paper is informative for those constructing similar high performance detection and identification systems. There are many future directions in which to explore, such as a larger comparison of patterns on a real vision system, trying non-binary color classification to improve single patch location, and determining the individual sources of error for detection so that they may be reduced. We feel the data and algorithms presented here offer a good starting point for such work.

References

[1] J. Bruce. CMVision realtime color vision system. The CORAL Group's Color Machine Vision Project. `http://www.cs.cmu.edu/~jbruce/cmvision/`.

[2] J. Bruce, T. Balch, and M. Veloso. Fast color image segmentation for interactive robots. *Proceedings of IROS 2000*, 2000.

[3] J. Bruce, M. Bowling, B. Browning, and M. Veloso. Multi-robot team response to a multi-robot opponent team. *Proceedings of IROS 2002 Workshop on Cooperative Robotics*, 2002.

[4] Y. Cho, J. Lee, and U. Neumann. Multi-ring color fiducial systems and a detection method for scalable fiducial tracking augmented reality. In *Proceedings of IEEE International Workshop on Augmented Reality*, November 1998.

[5] Y. Cho, J. Park, and U. Neumann. Fast color fiducial detection and dynamic workspace extension in video see-through self-tracking augmented reality. In *Proceedings of the Fifth Pacific Conference on Computer Graphics and Applications*, pages 168–166, October 1997.

[6] J. Heikkila. Geometric camera calibration using circular control points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1066–1077, 2000.

[7] S. Hibino, Y. Kodama, Y. Nagasaka, T. Takahashi, K. Murakami, and T. Naruse. Fast image processing and flexible path generation system for robocup small size leauge. In *Proceedings of the RoboCup-2002 Symposium*, 2002.

[8] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of the IJCAI-95 Workshop on Entertainment and AI/ALife*, 1995.

[9] M. Simon, S. Behnke, and R. Rojas. Robust real time color tracking. In *Lecture Notes in Artificial Intelligence 2019, RoboCup 2000: Robot Soccer World Cup IV*, 2001.