Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

Real Time Object Tracking using Reflectional Symmetry and Motion

Wai Ho Li and Lindsay Kleeman
Intelligent Robotics Research Centre
Department of Electrical and Computer Systems Engineering
Monash University, Clayton, Victoria 3800, Australia
{ Wai.Li, Lindsay.Kleeman } @eng.monash.edu.au

***Abstract*— Many objects found in domestic environments are reflectionally symmetric. In this paper, we present a system that can visually track moving objects by their reflectional symmetry in real time. Apart from the assumption of symmetry, the tracking system does not require any prior object models of the target, such as its colour and shape. The system is robust to shadows and specular reflections. It can also deal with transparent objects. Block motion detection is used in conjunction with symmetry for object tracking. A Kalman filter is used to estimate the object state. Predictions from the Kalman filter is used to improve the efficiency of the symmetry detector. The tracker provides a real time segmentation of an object by searching for motion that is symmetric about the object's mirror line. The tracking system also generates a rotated bounding box, aligned with the object's symmetry line, which can be used as a window for other image processing operations. The final system can track single objects in 640x480 videos at over 40 frames per second using a standard notebook PC.**

For videos of tracking results, please visit:
`www.ecse.monash.edu.au/centres/irrc/li_iro2006.php`

I. Introduction

Visual tracking systems employed in robotics generally require *a priori* models. These object models range in complexity from colour histograms to three dimensional mesh grids consisting of thousands of polygons. This prior knowledge allows for robust tracking, especially when several models are used in a synergetic manner. However, object models usually require significant human effort in their construction, which may include the collection of large, hand-labelled, training data sets, or removing an object from its environment for 3D scanning. For example, Boosting algorithms [1] can provide robust real time tracking, but require a large set of positive and negative training images, as well as significant offline training time. Systems that make use of multiple feature modalities, such as [2], can also perform robustly under real time requirements but again, require significant prior knowledge.

Constructing prior models for every object that can appear in a robot's environment is neither practical nor cost effective in many cases. In many environments, unmodelled objects may appear without warning. For example, a domestic robot can expect to encounter new toys and kitchen utensils during its cleaning operations. For increased adaptability, a robot should possess some means of tracking novel objects, in real time, without an *a priori* model. This will allow the robot to obtain information about its surroundings more autonomously. Ideally, the robot can obtain enough information during tracking to construct simple models for most new objects. This will shift the work of the user from building complete models, to the augmentation of models built by the robot. The latter should require less human effort. Segmentation of the tracked object is another valuable tool for model building. For example, colour models can be built by looking for colour consistency in the segmented object across multiple video frames.

To robustly track an object without its prior model, features that are robust to affine transformation and illumination changes are needed. Descriptive and robust feature detection schemes, such as SIFT [3], cannot be applied to real time tracking because of their computational cost. In order to perform tracking in real time, model-free methods usually make use of features that are computationally inexpensive to extract and match. Huang et al [4] used region templates of similar intensity. Satoh et al [5] utilized colour histograms. Both approaches can tolerate occlusions, but are unable to handle shadows and colour changes caused by variations in lighting. To track objects under different illumination conditions require features that do not directly rely on colour or intensity information.

Gestalt suggested that symmetry is one of several salient features humans use to visually identify objects. Indeed, many man-made objects are reflectionally symmetric. Apart from aesthetics, many objects are intentionally designed to be reflectionally symmetric for practical reasons. For example, headphones are symmetric so that sound can be delivered to both ears. Most drinking utensils, such as bottles and cups, are cylindrical to allow for easy manipulation, which makes them reflectionally symmetric when viewed from the side. As such, reflectional symmetry appears to be a valid salient feature for model-free tracking.

The detection of symmetry in digital images has been an area of heavy research. The Generalized Symmetry Transform [6] can detect bilateral and radial symmetry at different scales. Yip's [7] symmetry detector can detect skew symmetry, but at high computational cost. Ogawa suggested a Hough transform method to find symmetry in edge segments [8]. Other approaches include the use of ribbons [9] and modified versions of the Generalized Symmetry Transform. While radial

1-4244-0259-X/06/$20.00 ©2006 IEEE

symmetry has been used in real time applications [10], reflectional symmetry detectors, due to their high computational costs, are generally used in offline applications.

The research presented in this paper uses reflectional symmetry as a feature for the real time tracking of moving objects. The use of symmetry and motion as a segmentation tool is also examined. Apart from the assumption of reflectional symmetry, no prior object models are used. Note that the object being tracked does not have to be completely symmetric. *Partially* symmetric objects, such as a coffee mug with a handle, can also be tracked. The tracker, as it only relies on symmetry, can track multi-colour objects as well as transparent objects. The system can also tolerate occlusions and illumination changes.

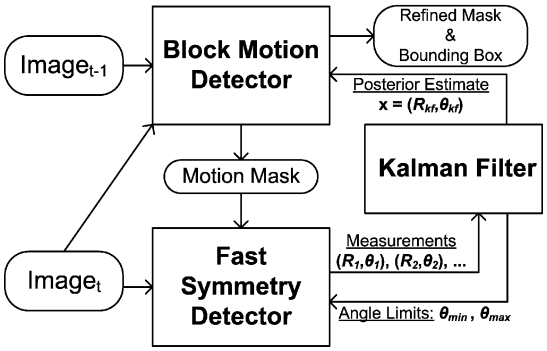


Fig. 1. System Diagram of Object Tracker

The system diagram in Figure 1 gives an overview of the tracking process. Motion detection results are used to limit symmetry detection to areas with movement. The Kalman filter prediction, before the measurement update, is used to speed up symmetry detection by limiting the detection angle. The detection results are then passed to the Kalman filter as measurement. Using the symmetry line estimate produced by the Kalman filter, the motion detection results are refined. This produces a near-symmetric segmentation of the object. A rotated bounding box is then computed based on the segmentation.

This paper is partitioned in the following manner. Our implementation of the *Fast Symmetry* detection algorithm is covered in brief by Section II. The block motion masking algorithm, and the refinement of the motion mask using symmetry, is discussed in Section III. Section IV covers the Kalman filter configuration and data association. A technique to automatically initialize the Kalman filter is also discussed. Tracking results are shown in Section V. Video frames of tracking result sequences can be found at the end of paper.

II. Fast Reflectional Symmetry Detection

An improved implementation of the *Fast Symmetry* detection algorithm [11] provides the tracker with symmetry line measurements. This symmetry detection algorithm has also been used to perform static object segmentation. Details concerning the implementation and its computational complexity can be found in the segmentation paper [12]. A brief summary of the detection method is provided below.

A. Algorithm Description

Symmetry detection is performed using the edge pixels of an image. By doing this, detection indirectly benefits from the noise rejection, edge linking and weak edge retention properties of edge filters. The Canny edge detector is used for edge detection, with the same aperture and threshold values employed across all experiments.

Standard Hough parameterization is used for the symmetry lines. Edge pixels are grouped into pairs and each pair votes for a single symmetry line in parameter space. Unlike traditional Hough Transform [13], which requires multiple votes per edge pixel, our approach only requires a single vote per edge pixel *pair*. This *convergent* voting scheme is similar to the approach used in Randomized Hough Transform [14].

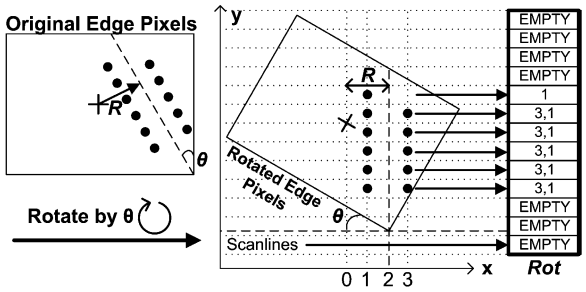


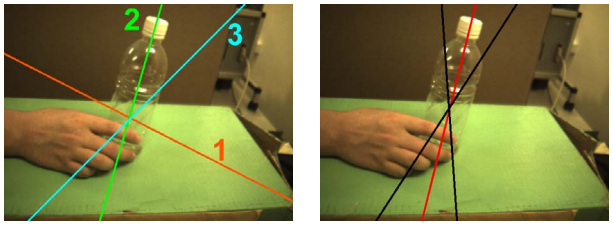
Fig. 2. Fast Symmetry Detection. Edge pixels (•) are rotated by θ about the image center. Then, the rotated pixels are inserted into a 2D array, Rot. Edge pixels from the same scanline are placed into the same row. Edge pixels in the same row are paired up and votes for symmetry lines in R, θ Hough space. The [3,1] rows will vote for the dashed symmetry line ($R = 2$)

For each Hough angle bin θ_{index}, the edge pixels are rotated about the center of the image by a corresponding angle θ. The rotated edge pixels are then quantized into a 2D array Rot. Edge pixels are placed into rows based on their *scanline* after rotation, as shown in Figure 2. Notice that any pair of rotated pixels from the same scanline can only vote for vertical lines of symmetry. This corresponds to the dashed symmetry line with angle θ prior to the rotation operation. The algorithm only allows edge pixels on the same scanline (same row in Rot) to pair up during Hough voting. This guarantees that all votes will be made for the same angle, shown as θ in Figure 2. The line radius R can be found by taking the average of the x coordinates of an edge pixel pair. After voting, symmetry line are found by looking for peaks in the Hough accumulator. A non-maxima suppression algorithm [15] was used for peak finding. The Kalman filter is provided with (R, θ) parameters of the symmetry lines with the highest number of votes.

B. Improving Symmetry Detection for use in Object Tracking

The raw Hough transform results cannot be used directly as measurements for tracking. Inter-object symmetry as well as symmetric portions of the background, like table corners, can overshadow the symmetry of the object being tracked. Figure 3(a) is an example where background symmetry lines may cause problems in tracking. The bottle's symmetry line is

weaker, in terms of its Hough vote total, than the orange symmetry line (line 1). As such, *non-object* symmetry should be rejected before symmetry detection, to improve the robustness of tracking. This is achieved by only allowing edges in the moving portions of an image to cast votes. A motion mask, generated using the algorithm detailed in Section III, is used to suppress background edge pixels. By doing this, the majority of votes will be cast by edge pixel pairs belonging to the moving object.



(a) Top three symmetry lines (b) Angle limits

Fig. 3. Symmetry Detection for use in Object Tracking.
Left: Top three symmetry lines returned by our detector. Lines are numbered according to the quantity of votes they received, with line 1 having received the most votes. Notice that the object's symmetry line is not the strongest one in the image.
Right: Angle limits (black) imposed on symmetry detection. The angle limits are generated using the Kalman filter prediction and the prediction covariance

The state prediction of the Kalman filter is used to improved the computational efficiency of symmetry detection. Recall that the symmetry detector iteratively rotates the edge pixels to find symmetry lines at different angles. Using the Kalman filter θ prediction and the prediction covariance, the range of rotation angles can be limited. Figure 3(b) is an example of such angle limits provided by the Kalman filter. By limiting the Hough voting angle, the total number of votes cast is reduced. This provides a large improvement to the execution time of the symmetry detection algorithm. In our experiments, three standard deviations either side of the symmetry line was used as angle limits.

III. Block Motion Detection

As seen in Figure 3(a), the amount of background symmetry needs to be reduced before applying the symmetry detector. In order to do this, a binary motion mask is used to eliminate static portions of video frames. Background modelling approaches are inappropriate for our application due to their assumption of a near-static background and consistent illumination conditions. Also, background modelling is not suitable for the detection and tracking of transparent and reflective objects. Instead, a fast block-based frame differencing approach is employed to generate the motion masks. The classic two-frame difference, first suggested by Nagel [16], is used.

A. Algorithm Description

The colour video frames are first converted to grayscale images. The absolute difference between time-adjacent images is calculated. The resulting *difference image* is then converted into a block image by spatially grouping pixels into 8x8 *blocks*. The choice of block size is arbitrary, and should be determined based on the smallest scale of motion to be considered by the tracker. The sum of pixel values in the difference image is calculated for each 8x8 block. Each block's sum is compared against the average value across all blocks. Blocks with a sum higher than a threshold value are classified as moving parts of a video frame. The average block sum multiplied by a constant factor is used as the motion threshold. The constant factor was determined experimentally, by starting at a value of 1, and increasing it until camera noise and small movements were successfully ignored. In all our experiments, a factor of 1.5 was used.

Algorithm 1: Block Motion Detection

Input: $I0$, $I1$ – Video frames at time t, $t+1$
Output: $mask$ – Motion Mask
Parameters: mf –Motion threshold

$diff \leftarrow |I1 - I0|$
res, sum are images $\frac{1}{blocksize}$ the size of $diff$
$sum[][] \leftarrow 0$
$i \leftarrow 0$
for $ii \leftarrow 0$ *to height of* res **do**
  $m \leftarrow i$
  $i \leftarrow i + blocksize$
  for *increment* m *until* $m == i$ **do**
    $j \leftarrow 0$
    for $jj \leftarrow 0$ *to width of* res **do**
      $n \leftarrow j$
      $j \leftarrow j + blocksize$
      for *increment* n *until* $n == j$ **do**
        $sum[ii][jj] \leftarrow sum[ii][jj] + diff[m][n]$
$res \leftarrow$ THRESHOLD(sum, AVERAGE(sum) $\times$ mf)
Median filter res then Dilate res
$mask \leftarrow res$ resized by a factor of $blocksize$

Algorithm 1 details the procedure used to generate the motion mask. The AVERAGE function returns an average of input elements. The THRESHOLD(A,b) function returns a binary image, consisting of 0's and 1's. An output element is set to 1 if the corresponding element in A is above the threshold value b. Otherwise, it is set to 0. Median filtering on the block level is used to remove spurious motion blocks caused by small movements and camera noise. The result, res, is then dilated to ensure that the edge pixels belonging to the moving object's contour are included in the motion mask. The mask is then produced by resizing the res image by a factor of $blocksize$.

B. Motion Mask Refinement

In Figure 4(a) and 4(c), motion masks have been overlayed on to video frames for illustrative purposes. In actual operation, the mask is used to suppress static edge pixels before passing the edge image to the symmetry detector. Images on

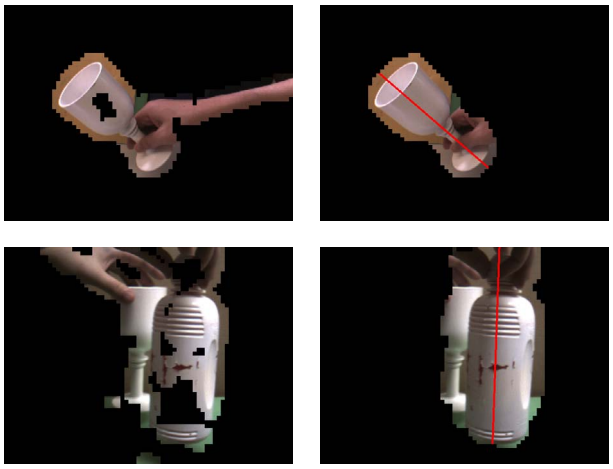


Fig. 4. Block Motion Masking
Left column: Images masked by the unrefined block motion mask.
Right column: Images masked using the refined block motion mask. The symmetry line estimate from the tracker is shown in red. The refined mask is generated based on this symmetry line estimate

the right column of the same figure is produced by applying a refined motion mask to the source image. The refined mask is produced through a two step process. Firstly, the location of each block with motion, b, is reflected across the symmetry line. The reflected location is searched using a local window. If none of the blocks in the window is classified as moving, the original block b is re-classified as static. This first step removes motion that are not symmetric about the object's symmetry line, which may have been caused by the end effector, and other moving objects. After the generation of a near-symmetric mask, the second step attempts to remove holes and gaps in the mask. This is achieved by looking for blocks that are surrounded by multiple neighbours that contain motion. These two steps are very efficient as they only operate on res, which has fewer pixels than the source image. The refinement process is a single pass operation.

IV. Kalman Filter

A Kalman filter, as described in [17], is used to estimate symmetry line parameters. Hough (R, θ) index values are used directly as measurements. A linear acceleration motion model is used. The filter plant and measurement matrices are shown below.

$$A = \begin{bmatrix} 1\,0\,1\,0\,\frac{1}{2}\,0 \\ 0\,1\,0\,1\,0\,\frac{1}{2} \\ 0\,0\,1\,0\,1\,0 \\ 0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$x = \begin{bmatrix} R\ \theta\ \frac{dR}{dt}\ \frac{d\theta}{dt}\ \frac{d^2R}{dt^2}\ \frac{d^2\theta}{dt^2} \end{bmatrix}^T$$

Process and measurement noise were chosen empirically. Measurement and process noise variables are assumed to be independent. The noise values used for all experiments are as follows. The R measurement variance is 9 $pixels^2$ and the θ variance is 9 $degrees^2$. The diagonal elements of the process covariance matrix are (1, 0.1, 10, 1, 10, 1). The odd elements are the position, velocity and acceleration covariance of R, the even elements are the θ covariances.

Data association and validation is performed using a validation gate. The top symmetry lines, in terms of its Hough votes, is given to the Kalman filter's validation gate. Symmetry line parameters that generate an error above 9.21 (2-DOF Chi-square, $P = 0.01$) are discarded by the gate. If no symmetry line passes through the gate without exceeding the Chi-square error threshold, the next state will be estimated using the state model alone.

To use the tracker in situations where new objects are being discovered by a robot, it must have an automatic initialization scheme. The initial state must be set to a value close to the moving object's symmetry line to ensure convergence. A novel initialization method is used to find the object's initial state. The number of moving blocks returned by the motion detector is continuously monitored. By looking for a sharp jump in the detected motion, frames where objects begin to move can be found. Symmetry lines detected from the three time-consecutive frames *after* an object begins to move are used to initialize the Kalman filter. Firstly, all possible data associations across the three frames are generated. In our experiments, the top three symmetry lines were used as measurements for each frame. This produced 3^3 *permutations*. Each generated data association permutation is used as Kalman filter measurement sets. The Kalman filter is initialized using the first measurement in the permutation, and updated using the second and third. The validation gate error for the updates are accumulated and logged. After iterating through all 27 permutations, the permutations are ranked according to their errors. The best permutation, that is, the data association sequence with minimum error, is used to initialize the Kalman filter. This automatic initialization procedure is used to start the tracker for all video sequences used in our experiments, without any manual intervention.

V. Object Tracking Results

The entire tracking system was implemented using C++ under Microsoft Windows XP, with no platform specific optimizations. The host computer was a notebook PC, with a 1.73GHz Pentium M processor and 2GB of 533MHz RAM. The video frames are 640x480 pixels in size, and were recorded at 25 frames per second. All experiments were performed using the same tracker parameter values. The Canny edge filter thresholds were set to 30 and 60, with an aperture size of 3 pixels. The block motion detector used a motion factor of 1.5. Borrowing from Randomized Hough Transform [14], a sampling ratio of 0.6 was used to obtain a random subset of the edge pixels.

Table I contains the execution times of the tracking code. Each sequence, numbered 1 to 10, contains up to 400 video frames. The code responsible for symmetry detection, block motion masking, mask refinement and Kalman filtering were timed independently. The average run time of these code segments can be found under the "Average Time" heading of

the table. The column labelled "Init" contains the time taken to perform automatic initialization as discussed at the end of Section IV. The average frame rates obtained are shown in the column labelled as "FPS". Note that the tracker was able to perform at above 40 frames per second for many sequences.

TABLE I
OBJECT TRACKER EXECUTION TIMES AND FRAME RATES

#	Average Time (ms)				Init	FPS
	Sym	Motion	Refine	Kalman	(ms)	(Hz)
1	37.87	4.84	0.86	0.09	10.41	22.91
2	16.76	4.76	0.75	0.06	9.74	44.77
3	17.95	4.85	0.85	0.04	10.69	42.22
4	18.31	4.74	0.75	0.04	11.90	41.96
5	33.69	4.87	0.87	0.05	11.38	25.33
6	20.84	4.94	0.85	0.04	13.18	37.50
7	35.29	5.01	0.87	0.13	11.32	24.22
8	34.48	4.94	0.79	0.14	11.14	24.79
9	18.19	4.91	0.79	0.06	11.83	41.75
10	27.01	4.89	0.82	0.06	12.50	30.51

The tracking system generates a rotated bounding box around the object being tracked. The bounding box is oriented such that two of its edges are parallel with the object's symmetry line. The size of the box is determined by the refined motion mask. Figure 5 shows two example bounding boxes, and the motion masks from which they were generated.

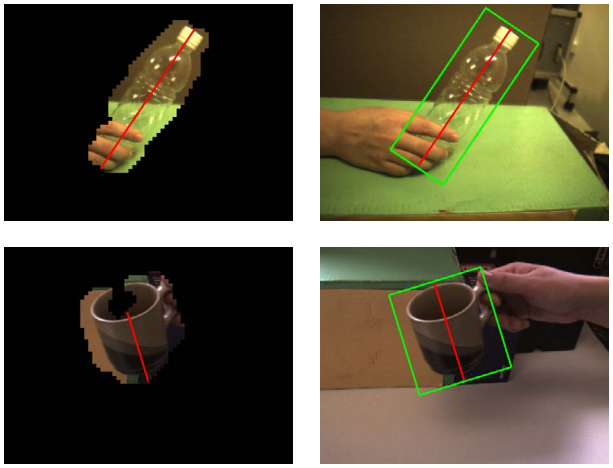


Fig. 5. Generation of rotated bounding boxes from refined motion masks
Left column: Symmetry-refined motion masks
Right column: Bounding boxes in green, symmetry lines in red

Frame sequences of the tracking results can be found at the end of this paper. Videos of tracking results can be downloaded from:

`www.ecse.monash.edu.au/centres/irrc/li_iro2006.php`

VI. CONCLUSION

The novel combination of using motion and reflectional symmetry for real time object tracking has been demonstrated. Tracking predictions were used successfully to improve the efficiency of symmetry detection. In addition, the use of symmetry to refine motion-based segmentation has been shown. The tracker can also generate a rotated bounding box, which can be used as a processing window for other vision algorithms to improve their efficiency. The bounding box may also be useful as a means of generating positive learning examples for training classifiers. The final system can track objects under difficult illumination conditions. The results in Figure 6 show that partially occluded objects can be successfully tracked, as well as multi-colour and transparent objects. The timing results show that the tracker can be operated in real time. Objects in 640x480 video sequences were successfully tracked at above 40 frames per second.

ACKNOWLEDGEMENTS

The authors would like to thank the ARC Centre for Perceptive and Intelligent Machines in Complex Environments (pimce.edu.au) for their financial support.

REFERENCES

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, no. 1. IEEE Computer Society, 2001, pp. 511–518.
[2] S. Khan and M. Shah, "Object based segmentation of video using color, motion and spatial information." in *CVPR (2)*, 2001, pp. 746–751.
[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, p. 91, November 2004.
[4] Y. Huang, T. S. Huang, and H. Niemann, "A region-based method for model-free object tracking." in *ICPR (1)*, 2002, pp. 592–595.
[5] Y. Satoh, T. Okatani, and K. Deguchi, "A color-based tracking by kalman particle filter." in *ICPR (3)*, 2004, pp. 502–505.
[6] D. Reisfeld, H. Wolfson, and Y. Yeshurun, "Context-free attentional operators: the generalized symmetry transform," *Int. J. Comput. Vision*, vol. 14, no. 2, pp. 119–130, 1995.
[7] R. K. K. Yip, "A hough transform technique for the detection of reflectional symmetry and skew-symmetry." *Pattern Recognition Letters*, vol. 21, no. 2, pp. 117–130, 2000.
[8] H. Ogawa, "Symmetry analysis of line drawings using the hough transform." *Pattern Recognition Letters*, vol. 12, no. 1, pp. 9–12, 1991.
[9] J. Ponce, "On characterizing ribbons and finding skewed symmetries," *Comput. Vision Graph. Image Process.*, vol. 52, no. 3, pp. 328–340, 1990.
[10] G. Loy and A. Zelinsky, "Fast radial symmetry for detecting points of interest," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 959–973, 2003.
[11] W. H. Li, A. Zhang, and L. Kleeman, "Fast global reflectional symmetry detection for robotic grasping and visual tracking," in *Proceedings of Australasian Conference on Robotics and Automation*, M. M. Matthews, Ed., December 2005.
[12] W. H. Li, A. M. Zhang, and L. Kleeman, "Real time detection and segmentation of reflectionally symmetric objects in digital images," Accepted for publication at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Beijing, China, October 2006.
[13] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.
[14] L. Xu and E. Oja, "Randomized hough transform (rht): basic mechanisms, algorithms, and computational complexities," *CVGIP: Image Underst.*, vol. 57, no. 2, pp. 131–154, 1993.
[15] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using MATLAB*. Prentice Hall, 2004.
[16] H. H. Nagel, "Formation of an object concept by analysis of systematic time variations in the optically perceptible environment," *j-CGIP*, vol. 7, no. 2, pp. 149–194, Apr. 1978.
[17] Y. Bar-Shalom, T. Kirubarajan, and X.-R. Li, *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2002.

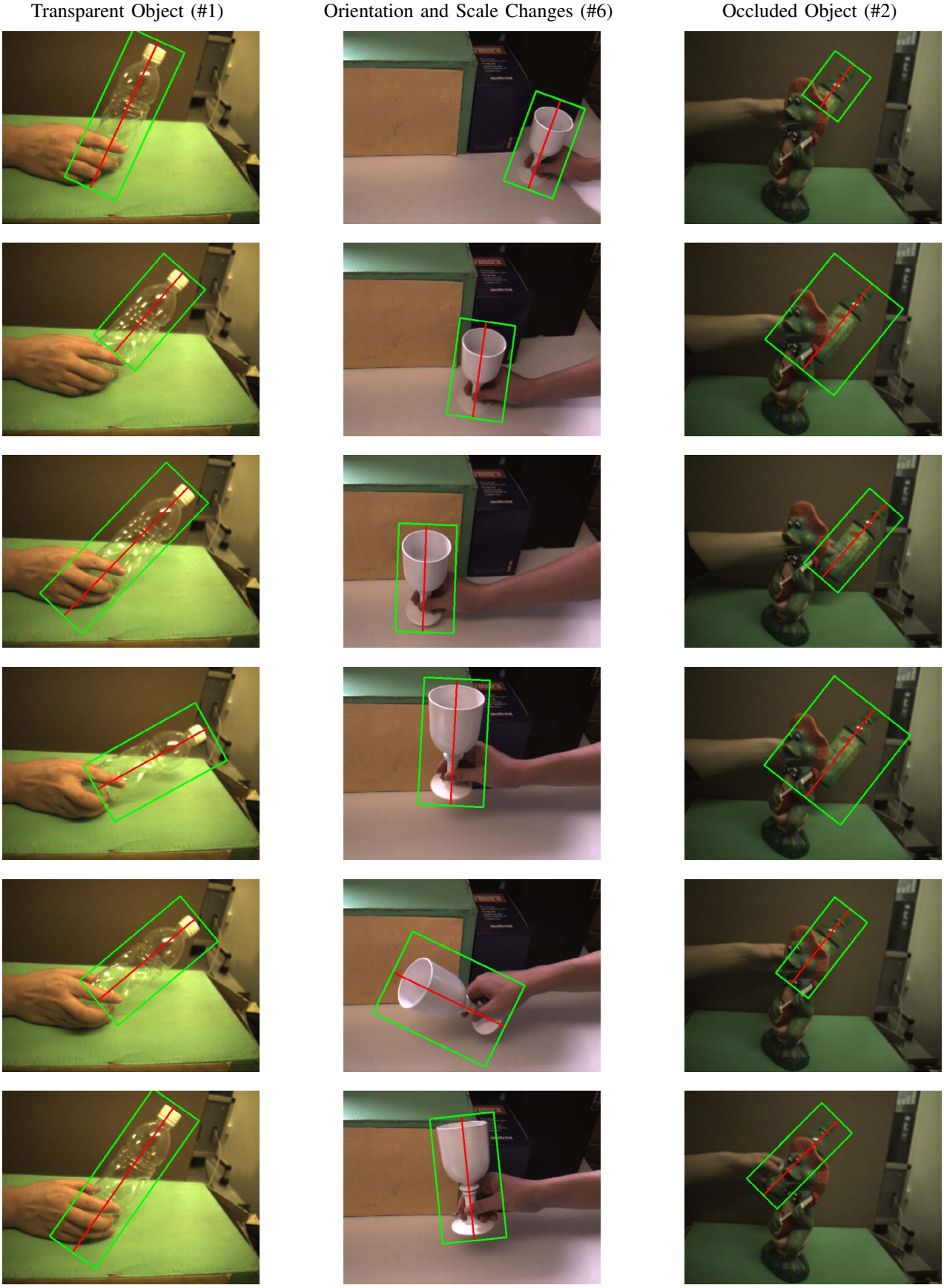


Fig. 6. Tracking Results. The sequence number shown in brackets next to the title is the same as that used in Table I. The symmetry line posterior estimate produced by the Kalman filter is shown in red. Rotated bounding boxes are shown in green around the moving object.
Left column: Transparent object tracking. Every 15th frame shown
Middle column: Object tracked across scale and orientation changes. Every 30th frame shown
Right column: Tracking an object through occlusion. Every 10th frame shown