Robotic Eye-in-hand Calibration in an Uncalibrated Environment

Sebastian van Delden and Frank Hardy
Division of Mathematics and Computer Science
University of South Carolina Upstate
Spartanburg, SC 29303
{svandelden, flhardy}@uscupstate.edu

Abstract — *The optical flow of high interest points in images of an uncalibrated scene is used to recover the camera orientation of an eye-in-hand robotic manipulator. The system is completely automated, iteratively performing a sequence of rotations and translations until the camera frame is aligned with the manipulator's world frame. The manipulator must be able to translate and rotate its end-effector with respect to its world frame. The system is implemented and being tested on a Stäubli RX60 manipulator using an off-the-shelf Logitech USB camera.*

Keywords *— Camera Calibration, Eye-in-hand manipulator, Robotics, Computer Vision.*

I. INTRODUCTION

Visual input sensors are incorporated into robotic automation tasks to enhance an application's flexibility, precision, and/or functionality. Key features are extracted from input image sequences and used to guide the manipulator's end-effector to a desired pose where a specific automation task is then accomplished. Some typical automation tasks include picking/placing, spot welding, spray painting, drilling holes, and product assembly. [1] is an excellent article that describes several vision-based object handling industrial applications currently in use.

There has been much work in the area of closed-loop visually guided robotics ([2]-[7] are just a few of the papers in the literature that present good overviews of visually guided robotics research with the focus on closed loop systems, or visual servoing). However, many current industrial applications employ also calibrated systems – [8]-[10].

In a visual servoing system, visual feedback is used to minimize the image plane error of the manipulator's actual and desired positions. The vision system looks at the current pose of the manipulator and estimates how its joints should be moved so that the manipulator draws closer to the desired pose. Typical tasks like tracking and positioning are performed by reducing the image distance error between a set of current and desired image features in the image plane.

In a calibrated system, the camera and robot kinematics are calibrated relative to a fixed 3D frame. The classical approach is to move the end-effector and observe/perceive the movement of the eye: or AX =XB, where A is the robot end-effector motion ${}_{t1}T^{t2}$, B the induced camera motion ${}_{c1}T^{c2}$,and X is the hand-eye transformation ${}_{t}T^{c}$ to be determined.

In this paper we present an approach that overcomes two traditional problems that calibrated systems face: having to manually re-calibrate over time, and relying on a calibrated input scene.

Over time the precision of the robot/camera coordinate system calibration degrades due to movement, vibrations and other forces [11], and so the system must be periodically re-calibrated. This is a potentially cumbersome task if a human operator is responsible for performing this constant re-calibration procedure. The approach presented here automatically recovers the orientation of the camera frame with respect to (w.r.t) the robot world frame. The algorithms are designed for a monocular eye-in-hand system where the robot controller is capable of rotating and translating the tool frame w.r.t. the world frame. Since this method is completely automated, there is no need for a human operator to re-calibrate the system. The re-calibration procedure could be run periodically and automatically by the system to ensure that precise calibration is maintained.

A calibrated input image is usually required in order to recover camera orientation. For example, [12] uses a bimodal thresholding algorithm [13] and the movements of a single *blob* on a solid colored background to recover camera orientation, [13] uses several circles on a cube shaped surface, and [14] uses the traditional checkerboard pattern. The approach in this paper brings enhanced flexibility to the input scene by using the optical flow of high interest points in an uncalibrated scene to recover camera orientation. Although a precisely calibrated image is not required, there are some restrictions placed on the input scene which are outlined in the next section.

In a nutshell, the approach employs three iterative processes, each of which can be described as follows. An image is captured, the end-effector is translated in a specific direction, and a second image is then captured. The correspondences of high interest points (corner points) from the first image is found in the second image. Based on the *actual* movements of the points and the known *desired* movements, the end-effector is incrementally rotated about a specific axis. This process is repeated until the actual movement equals the desired movement.

The remainder of this paper is organized as follows. In the following section we list assumptions about the robot's work

cell environment and also describe the initializations that are required for the algorithms to function properly. The control and vision algorithms are presented in sections III and IV, respectively. In section V, we present some experimental results and section VI offers some concluding remarks and future work.

II. Initial Setup

In order for the algorithms in the following sections to properly converge, the following assumptions and initializations are required:

- Although a precisely calibrated image is not required, there cannot be moving objects in the scene and there must be high interest points (we used corner points) somewhere in the input scene. For example, an input scene of a solid color wall, floor or table will not contain high interest points. Figure 1 depicts an example input scene that is used in our implementation. The camera is looking at an un-calibrated scene of randomly placed shapes and objects.
- The manipulator must be able to translate and rotate its tool frame {T} w.r.t. its world frame {W}. As the location of the tool flange changes in time, the transformation ($_{T}T^{W}$) between points in {T} and {W} is automatically maintained internally by the robot controller.
- The camera must be mounted to the end effector. The pose of the camera frame {C} is not known w.r.t. {T} or {W}.
- A rough alignment of the {C} w.r.t. {W} must be initially determined. The closest axis X_W, Y_W, and Z_W in {W} (within 45^o or -45^o) must be mapped to X_C, Y_C, and Z_C in {C}. For example, $+X_C$ is within 45^o or -45^o to $-Y_W$, $+Y_C$ is within 45^o or -45^o to $-X_W$, and $+Z_C$ is within 45^o or -45^o to $-Z_W$. This could be done manually, but section II.A describes an algorithm that automates this initialization process.

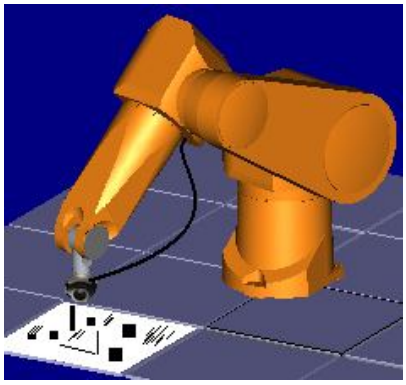

Fig. 1. An example configuration of the robot and camera. The camera is looking at an uncalibrated scene of random shapes and objects.

A Initial Rough Alignment

An initial rough alignment of the robot world {W} and camera frame {C} axes must be determined so that an approximate correlation between movements in the robot world frame and the interest points can be established. Figure 2 depicts an initial rough alignment that was used in one of our experiments.

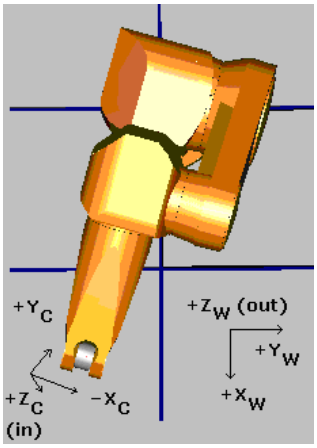

Fig 2. An example initial alignment of camera and robot frames. The unknown angle differences are recovered by the algorithms in the following sections.

The initial axis correlations are recovered by moving the end-effector (and thus the camera) along each of the robot's world axes and observing the greatest interest point change in the camera coordinate system. For example, in Figure 2, a translation of the end-effector in $+Y_W$ resulted in maximum interest point movement along $+X_C$, which would imply that $+Y_W$ is most aligned with $-X_C$.

The initial alignment is determined as follows:

- Translate some distance in $+X_{W.}$ $+Y_{W.}$, and $+Z_{W.}$
  - The distance is arbitrary, but the high interest points should not move out of the FOV of the camera.
- Compute high interest point correspondence movement in X_C and Y_C after each translation.
  - Each translations results in X_C and Y_C high interest point movements (six values in total).
- The top two high interest point movements in X_C and Y_C indicate the alignment of two of the robot axes, and the third alignment can then be automatically determined.

We will refer to this initial alignment later by means of a MAPPING( ) function, where, for example, MAPPING(X_C) corresponds to the robot world axis that is most closely aligned with the camera's X axis.

III. MANIPULATOR CONTROL ALGORITHMS

The vision algorithms communicate with the manipulator control algorithms by sending a three tuple of information that indicates what type of incremental movement should be made by the end-effector: ({rotation, translation}, {X_W, Y_W, Z_W axis, {positive or negative decimal number}). The positive or negative decimal number indicates the direction of the translation or rotation, and also how many mm or degrees should be moved. Only incremental movements are made until the camera and robot world frames are aligned, avoiding the need to determine a mm per pixel relationship which would be work cell specific.

The algorithm is summarized below in a V+ type syntax which is used by Stäubli RX series manipulators.

```
WHILE (NOT ALIGNED) DO

(TYPE,AXIS,VALUE) ← THREE TUPLE

CUR_POS ← CURRENT END-EFFECTOR POSITION

(CUR_X,CUR_Y,CUR_Z,
CUR_YAW,CUR_PITCH,CUR_ROLL) ← DECOMPOSE(CUR_POS)

IF (TYPE == TRANSLATE) THEN
  IF (AXIS == X) THEN
      MOVE TRANS(VALUE,0,0,0,0,0):CUR_POS
  END
  ELSE IF (AXIS == Y) THEN
      MOVE TRANS(0,VALUE,0,0,0,0):CUR_POS
  END
  ELSE (IF AXIS == Z) THEN
      MOVE TRANS(0,0,VALUE,0,0,0):CUR_POS
  END
END

IF (TYPE == ROTATE) THEN
  IF (AXIS == X) THEN
     MOVE TRANS(CUR_X,CUR_Y,CUR_Z):RX(VALUE):
          TRANS(0,0,0,CUR_YAW,CUR_PITCH,CUR_ROLL)
  END
  IF (AXIS == Y) THEN
     MOVE TRANS(CUR_X,CUR_Y,CUR_Z):RY(VALUE):
          TRANS(0,0,0,CUR_YAW,CUR_PITCH,CUR_ROLL)
  END
  IF (AXIS == Z) THEN
     MOVE TRANS(CUR_X,CUR_Y,CUR_Z):RZ(VALUE):
          TRANS(0,0,0,CUR_YAW,CUR_PITCH,CUR_ROLL)
  END
END

END
```

The control algorithm receives the three tuple of information and makes the movement relative to the current location of its end-effector. Each time a movement is made, the location of the end-effector must be updated. Communication and movement must be synchronized so that the robot completes its current motion before the vision algorithms compute the next motion.

The DECOMPOSE function recovers the X, Y, Z, Yaw, Pitch, and Roll values of CUR_POS, the current location of the end-effector. The TRANS(X, Y, Z, Yaw, Pitch, Roll) function returns a transformation created from its parameters. The RX(p), RY(p),and RZ(p) functions returns pure rotation transformations of p degrees around the world X, Y, and Z axes, respectively. A colon denotes transformation matrix multiplication.

Notice for the rotations portion of the algorithm, that a pure translation transformation is first created from the end-effector's X, Y, and Z values. The yaw, pitch and roll values of this transformation are equal to the world frame. This transformation is then multiplied by a pure rotation transformation around the desired world axis. Finally, the result is multiplied by a pure rotation transformation created from the original Yaw, Pitch and Roll from CUR_POS. This ordering is essential for rotating the end-effector around {W} and not {T}. Using the X, Y, and Z components from CUR_POS ensures that the rotation is made from a point close to the camera which will prevent a large end-effector movement that would move most high interest points identified in the first image out of the camera's FOV.

IV. VISION ALGORITHMS

A High Interest Points and Correspondences

We implemented a traditional approach to detecting high interest points, the Moravec operator [16]. This operator detects corner points as high interest points by observing the intensity variation in every direction - horizontal, vertical, left diagonal and right diagonal. The variance is calculated over a nxn search window where n is a positive odd integer. The sums of squares of differences of pixels adjacent to each other in each of the four directions is calculated and the minimum value is returned.

The non-maximal suppression technique that we used to identify local maxima is to choose high interest points as those image locations with normalized values above 0.9. Figure 3 shows a test image in which high interest points have been identified using this process. The image features an uncalibrated, oddly shaped object that would deliver good high interest points.

We used a 7x7 matching window in our experiments since the Moravec operator is more sensitive to noisy data in very small windows. Also, because the Moravec operator is sensitive to noise, we first smooth the input images using a median filter. A median filter replaces the center pixel of the window with the median value in the window and is implemented using a quicksort approach which usually finds the median value without having to sort all values in the window [13].

Fig 3. An example test image where the Moravec operator has identified high interest corner points.

Each high interest point in the first image is searched for in the second image which has also been smoothed. The location of the best match of each interest point is determined by using the sum of squared differences. Figure 4 shows a second test image in which the camera has moved and the correspondences and movements of the interest points are overlaid.

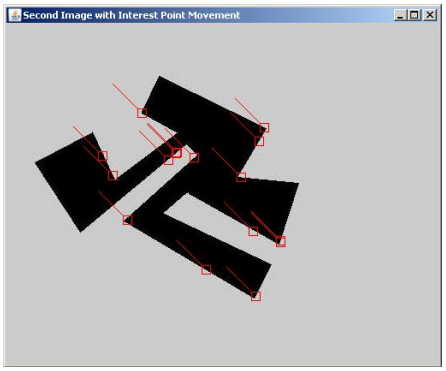

Fig 4. A second example test image in which the camera has moved and the correspondences and movements of the interest points are drawn on top it.

Because of lens distortion issues and the possibility of interest points moving out of the field of view of the camera, we do not consider high interest points close to outer edges of the image (outer 20% of pixels).

B Orientation Recovery

We use a sequence of three iterative processes to recover the camera's orientation. In each process:

- An image is taken
- The camera is translated along one of the robot's world axes
- A second image is taken
- High interest points are found in the first image and their correspondences found in the second image.
- Based on the actual movement of the interest points, small rotations around the robot world axes are made until the actual movements equal the desired movements.

The below ordering of the translations and rotations is not necessarily required as long as a sequence of three Euler angle rotations is used to recover the three angles.

Desired interest point movement is very intuitive. If the camera and robot axes are perfectly aligned, then:

- A translation along MAPPING (X_C) would result in perfect horizontal interest point movements in the image.
- A translation along MAPPING (Y_C) would result in perfect vertical interest point movements in the image.
- A translation along MAPPING (Z_C) would result in interest point movements whose corresponding lines extend through the center of the image.

In our system, the sequence of three iterative steps is arranged as follows:

- **First**, translate back and forth along MAPPING(X_C), note the movement of the interest points, and then incrementally rotate around MAPPING(Z_C) until the difference of the row values of the correspondences is minimized.
  - This aligns X_C to the *plane* created by MAPPING(X_C) and MAPPING(Z_C) axes.
- **Second,** translate back and forth along MAPPING(Y_C) direction, note the movement of the interest points, and then incrementally rotate around MAPPING(X_C) until the difference in the column values of the correspondences is minimized.
  - This aligns Y_C perfectly with MAPPING(Y_C).
- **Third**, translate back and forth along MAPPING(Z_C), note the movement of the blob, and then incrementally rotate around MAPPING(Y_C) until the difference between the line that extends through the correspondences and the center of the image is minimized.
  - This results in all three camera axes being aligned with their corresponding world axes.

The distance to be translated in each step is an arbitrary length in millimeters that must be small enough to prevent the high interest points around the center of the image to move outside of the camera's FOV. The incremental rotation value should be small. In our experiments it was set at 1^o,

but using a much smaller rotational amount would not have an adverse affect of system. A positive or negative rotation is made if the location of the actual column/row correspondence value is greater than or less than the desired value, respectively.

V. Experimental Results

The system has been implemented on a Stäubli RX60 robotic manipulator which is controlled by a CS7B controller running the V+ operating system and programming language. The vision algorithms are written in Java and use the Java Media Framework (JMF) API to communicate with an off-the-shelf Logitech USB camera. We have been choosing random starting configurations and then executing the algorithms. Initial results show that the algorithms converge for the test cases as long as the scene contained high interest points whose correspondences were correctly found in the image pairs. The rotation algorithms iteratively recover the unknown angles in a linear fashion, so convergence speed of the algorithm is linear and varied based on the size of the angles. A typical example execution of the first alignment step is carefully outlined below. The second and third steps are similar and will not be further presented here.

Figure 5 shows an example input scene taken initially before the orientation recovery algorithms are executed. Notice that the uncalibrated scene contains several arbitrary objects lying in random. This is the second image taken after the robot has translated in MAPPING(X_C). The movement of the strongest high interest points (normalized value > 0.9) from the previous image are overlaid on this image.

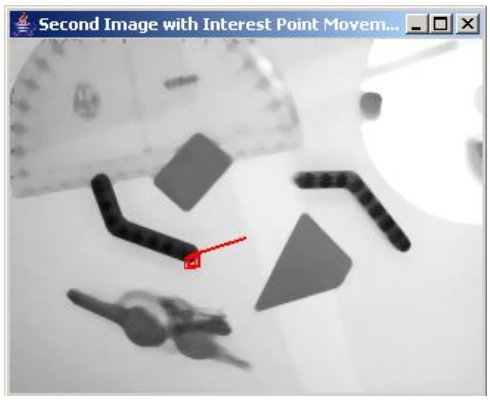

Fig. 5 Example initial input scene of random objects with the movements of strong high interest points overlaid.

From this image, it is obvious that MAPPING(X_C) is not aligned with X_C, so a small rotation is made around MAPPING(X_Z) and the process is repeated until the high interest points make a perfectly horizontal movement.

Figure 6 show the same scene after at the eleventh iteration of this process. This screen shot was chosen because a completely incorrect correspondence of a high interest point was calculated which resulted in an incorrect rotational value. The algorithm converges as long as correspondences for the high interest points are correctly determine for the majority of input image pairs. An occasional incorrect match as shown in Figure 6 can sometimes result in a rotational value opposite to the desired rotational value (which happened in iteration eleven), reducing convergence speed of the algorithm. If the majority of correspondences are incorrect, the algorithm will not converge.

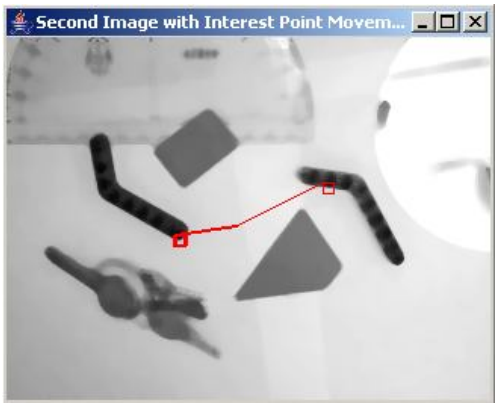

Fig. 6 Example screenshot at the eleventh iteration of the algorithm which contains an incorrect interest point correspondence.

Finally, Figure 7 shows movement of the interest points after 23 iterations. The algorithm has converged and movement of high interest points are perfectly horizontal which indicates X_C is aligned with the plane created by MAPPING(X_C) and MAPPING(X_Z).

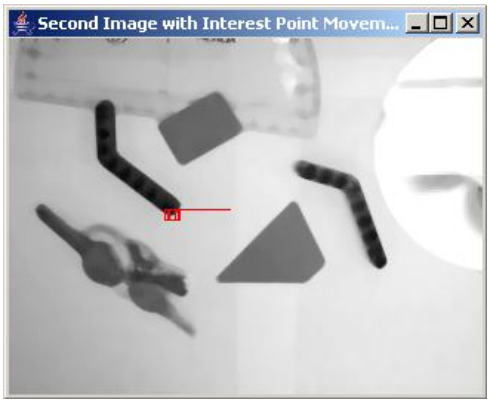

Fig. 7 Example screenshot at the twenty third iteration of the algorithm which shows perfect horizontal movement of the interest points which causes the algorithm to terminate.

Figure 8 plots the row error of the correspondences during the execution of the algorithm. The somewhat jagged nature of the plot is expected due to slightly incorrect correspondence matches. At iterations 11 and 13, incorrect correspondences causes the error distance to flip signs which delayed the convergence of the algorithm.

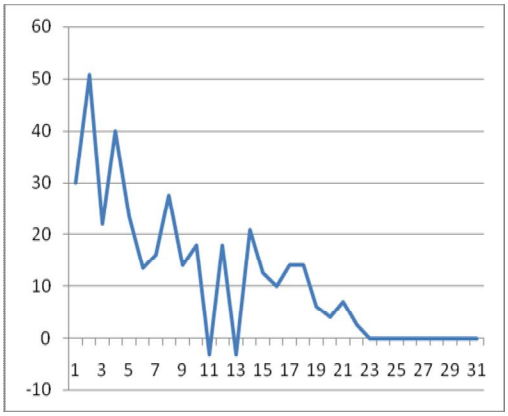

Fig. 8 Plot of row error distances from interest point correspondences.

A 19^{o} angle offset between MAPPING(X_C) and X_C was correctly recovered by the algorithm after 23 iterations. The algorithm converges linearly, but needed an additional four iterations: two incorrect rotations were made which then required an additional two correct rotations to compensate for these erroneous movements.

VI. Conclusions and Future Work

The optical flow of high interest points in an uncalibrated input scene can be used to iteratively recover the camera orientation of an eye-in-hand robotic manipulator. The algorithms could be periodically executed by the manipulator to maintain precise calibration over time.

To recover translation offsets of the camera's coordinate system, we are implementing the depth extraction algorithms in [15] and will also try to improve on the accuracy of that work. We are also implementing various corner point detection approaches [18] to determine which technique yields the best results for our application.

Acknowledgment

The Stäubli Corporation generously donated six RX60 manipulators and a RS20 manipulator to our institution, and also provided the Stäubli Robotics Studio software package to us which was used to create the 3D figures in this paper. We would like to express our sincere thanks to the Stäubli Corporation for their continued support of undergraduate research at our institution.

References

[1] P. Sanz, A. Requena, J. Inesta, and A. Del Pobil. "Grasping the not-so-obvious: vision-based object handling for industrial applications," in *IEEE Robotics & Automation Magazine*, vol. 12(3), pp. 44-52, 2005.

[2] D. Kragic. "Visual servoing for manipulation: robustness and integration issues," *Ph.D. Thesis*, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden, 2001.

[3] D. Kragic and H. Christensen. "Robust visual servoing," *The International Journal of Robotics Research*, vol. 22(10-11), pp. 923-939, 2003.

[4] K. Hashimoto. "A review on vision-based control of robot manipulators," in *Advanced Robotics*, vol. 17(10), pp. 969-991, 2003.

[5] J. A. Piepmeier, and H. Lipkin. "Uncalibrated Eye-in-Hand Visual Servoing," in the *International Journal of Robotics* Research, vol. 22(10-11), pp. 805-819, 2003.

[6] S. Hutchinson, G. Hager and P. Corke. "A tutorial on visual servo control," *in IEEE Transaction on Robotics and Automation*, vol. 12(5), pp. 651-670, 1996.

[7] P. Corke. "Visual control of robot manipulators – a review," *in Visual Servoing, vol. 7 of Robotics and Automated Systems*, pp. 1-31, World Scientific, 1993.

[8] K. H. Strobl and G. Hirzinger. "Optimal hand-eye calibration," in *Proc. of the IEEE/RSJ International Conference of Intelligent Robots and Systems*, Bejing China, 2006.

[9] H. Malm and A. Heyden. "Extensions of plane-based calibration to the case of translational motion in robot vision sensing," *IEEE Transaction on Robotics*, vol. 22(2), 2006

[10] S. Remy, M. Dhome, J. M. Lavest, and N. Daucher. "Hand-eye calibration," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Grenoble, France, pp. 1057-1065, 1997.

[11] M. Salinger, "Point-and-click camera-space manipulation, mobile camera-space manipulation, and some fundamental issues regarding the control of robots using vision," *Ph.D. Dissertation*. University of Notre Dame, 1999.

[12] S. van Delden, R. Farr, and S. Hensley. "An Automated Camera Orientation Recovery Algorithm for an Eye-in-Hand Robotic Manipulator," in *Proc. of the 5th IEEE International Conference on Robotics and Sensor Environments*, Ottawa, Canada, pp. 1-6, 2007.

[13] L. G. Shapiro and G. C. Stockman. "Computer vision," *Prentice Hall*, 2001.

[14] J. Heikkilä, "Geometric Camera Calibration Using Circular Control Points", *in the IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22(10), pp. 1066-1077, 2000.

[15] R. Tsai, "A Versatile Camera Calibration Technique for a High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *in the IEEE Journal of Robotics and Automation*, vol. RA-3(4), 1987.

[16] H. Moravec. "Visual mapping by a robot rover," *In Proc. of the 6th International Joint Conference on Artificial Intelligence*, pp. 598-600, 1979.

[17] D. Perrin, C. Smith, and N. Papanikolopoulos. "Depth extraction for contours by monocular eye-in-hand systems," in *Proc. of the 8th IEEE Mediterranean Conference on Control and Automation*, Rio, Greece, 2000.

[18] C. Schmid, R. Mohr, and C. Bauckhage. "Evaluation of Interest Point Detectors," *in the International Journal of Computer Vision*, Springer Netherlands, vol. 37(2), pp. 151-172, 2000.