Increasing Pose Estimation Performance using Multi-cue Integration

Fredrik Vikstén*, Robert Söderberg*, Klas Nordberg* and Christian Perwass†
*Computer Vision Laboratory
Department of Electrical Engineering
Linköping University, 581 83 Linköping, SWEDEN
Email: {viksten,soderberg,klas}@isy.liu.se
†Cognitive Systems Group
Computer Science Department
Christian-Albrechts-University Kiel, 24118 Kiel, GERMANY
Email: chp@ks.informatik.uni-kiel.de

***Abstract*— We have developed a system which integrates the information output from several pose estimation algorithms and from several views of the scene. It is tested in a real setup with a robotic manipulator. It is shown that integrating pose estimates from several algorithms increases the overall performance of the pose estimation accuracy as well as the robustness as compared to using only a single algorithm. It is shown that increased robustness can be achieved by using pose estimation algorithms based on complementary features, so called algorithmic multi-cue integration (AMC). Furthermore it is also shown that increased accuracy can be achieved by integrating pose estimation results from different views of the scene, so-called temporal multi-cue integration (TMC). Temporal multi-cue integration is the most interesting aspect of this paper.**

I. Introduction

One problem that computer vision research has been trying to solve for a long time is to estimate the physical state of an object in an image. Applications that would benefit from a robust pose estimation system include bin picking and augmented reality systems as well as navigation or warning systems in cars or autonomous fork lifts etc. Estimation of position and orientation in space is called pose estimation. For a pose estimation algorithm to be robust for varying types of objects and lighting conditions, it needs to be based on descriptive and robust local features [1], [2], [3], [4]. It is hard or even impossible to do this with only one pose estimation algorithm that uses only a single type of local feature. This is due to the fact that a typical local image feature is only able to robustly describe a subset of possible objects and images thereof. If several algorithms are used, there is a higher probability that at least one succeeds in estimating the object state parameters under the current conditions. However if several algorithms are running, we have to solve the problem of how to integrate the cues from the different algorithms. Best performance of a multi-cue integration system is reached if the local features produced by the different algorithms are complementary, e.g. different features such as lines, corners, ellipses or model free patches. In section III a framework for multi-cue integration for increasing the robustness is discussed.

Most of the multi-cue integration systems found in the literature [5], [6], [7], including the one described in section III, perform integration of cues coming from a single image. This type of system seems to improve the robustness rather than increasing the accuracy. Several visual servoing systems use an eye-in-hand setup [8], [9], where the camera is mounted on the manipulator. If the camera is moved such that the object is seen from different views and the movements are recorded with high accuracy, it is possible to perform a multi-cue integration over time to improve the accuracy of the object state estimate. In section IV, a system for improving the accuracy is discussed, which is based on the framework used in section III.

II. Pose estimation algorithms

Any algorithm that outputs object state parameters together with a confidence value can be used in this cue integration framework. A higher dimensionality of the voting space gives a higher chance of object state estimates not interfering with each other so it is preferable to use algorithms that output as full a 3D object state description as possible. Algorithms that are suitable to use in this framework include [1], [2], [3], [4], where the last two are the ones used in the experiments.

III. Multi-cue integration for robustness

Several systems for multi-cue integration are based on the integration of different local features as cues for getting a more robust system. The integration presented here is different since the cues are results from a number of pose estimation algorithms instead of local features. The algorithmic multi-cue integration (AMC) is illustrated in Fig. 1, where votes on object state parameters from each pose estimation algorithm are put into a voting space with corresponding confidence value and the integration is implemented by using a clustering algorithm. The center of each cluster forms a new result with a new confidence value. This confidence value is based on the confidence values of the votes in the cluster and how dense the cluster is. Observe that each pose estimation algorithm should generate several object state estimates, where the additional estimates could come from several objects or from noise.

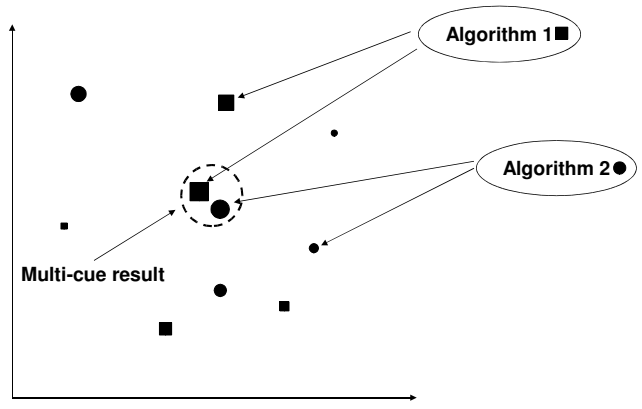

Fig. 1. Ordinary case for multi-cue integration between algorithms.

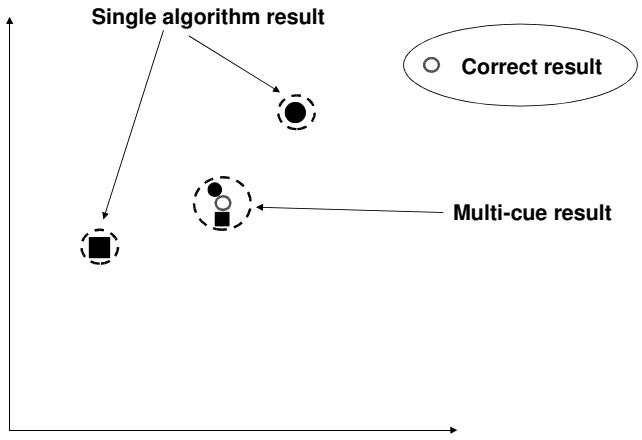

Fig. 2. Multi-cue integration increasing robustness.

The clustering algorithm used in this type of integration has to be efficient on multi-dimensional vectors and has to take the corresponding confidence value into account. One such algorithm is the mean-shift clustering algorithm [10], [11], [12], which is used here.

It is important that the resulting object state vectors from the pose estimation algorithms are unambiguous in the sense of comparing different object states. For example the Euler-angle representation is ambiguous in that sense, but rotation matrices and quaternions are not. We have chosen to work with quaternions because they are a more compact representation and will therefore reduce the computation time of the clustering.

When AMC is implemented as a clustering of pose estimate results it can behave in a number of different ways,

- Single algorithm result.
- Robust mean between two or more algorithms.
- Algorithmic voting.

The *single algorithm result* will occur when all algorithms but one have failed to estimate the object state parameters. Algorithms that fail output mostly noise. It is unlikely that such output will form clusters in the voting space. Therefore, the result from the algorithmic multi-cue integration will be the algorithm with the highest confidence. The confidence values should be relative to the quality of the object state estimate and the successful algorithm will therefore have the highest confidence. Of course, the robustness is highly dependent on the confidence measurements for this type of situation.

In the case where two or more algorithms succeed in estimating the object state parameters, the votes from these algorithms will normally cluster in the voting space. For this type of situation the result from AMC will be a *robust mean* of the votes contained in the cluster.

In case of occluded objects or when only a very small part of the object is visible in the image we may have the case that one of the correct pose estimates is ranked only as second or third choice by the confidence of the algorithms. One goal of AMC is to make the system more robust to this type of problem. Therefore the system is designed such that two or more votes that do not necessarily have the highest confidence for their algorithms, but which are similar should be chosen before a single vote that by itself has a higher confidence. We call this *algorithmic voting*. The mean-shift clustering implementation uses the confidence of each guess as a weight and therefore gives us exactly this kind of behavior. This case is illustrated in Fig. 2 where the size of the shape signifies confidence.

IV. Multi-cue integration for accuracy

Although the cameras of today have good image quality, there is still some noise in the images. The amount of noise is of course dependent on the camera quality, but also on the camera settings such as gain and exposure time. To analyze how this noise influences the quality of the two pose estimation algorithms, 40 images of an object were captured from the same camera position. The orientation part of the pose estimates from these images were then compared with the ground truth. The angular error for the 40 images is plotted in Fig. 3, where the error is the angle β between the quaternions that represent the ground truth $\mathbf{q}_g$ and the pose estimate $\mathbf{q}_p$. The error is given by

$$\beta = 2 \arccos \left(\frac{\mathbf{q}_g \mathbf{q}_p}{|\mathbf{q}_g| \, |\mathbf{q}_p|} \right). \tag{1}$$

Since neither the camera nor object were moved during the experiment, the variations in the pose estimates must be due to the image noise. The algorithms were evaluated on low-pass filtered 8-bit gray images with pixel-noise having a variance of just below 1 pixel-level[1].

It is clear that this noise is approximately unbiased, and it is therefore possible to decrease the noise by calculating a mean of a subset of the pose estimates. In Fig. 4 the pose estimate result from multi-cue integration between the algorithms is plotted together with the result from using both multi-cue integration between algorithms and with previous estimates. The dashed curve is the result when the mean is calculated

[1]Images in range 0-255 with usual values for black around 30 and the lighter areas of objects around 240.

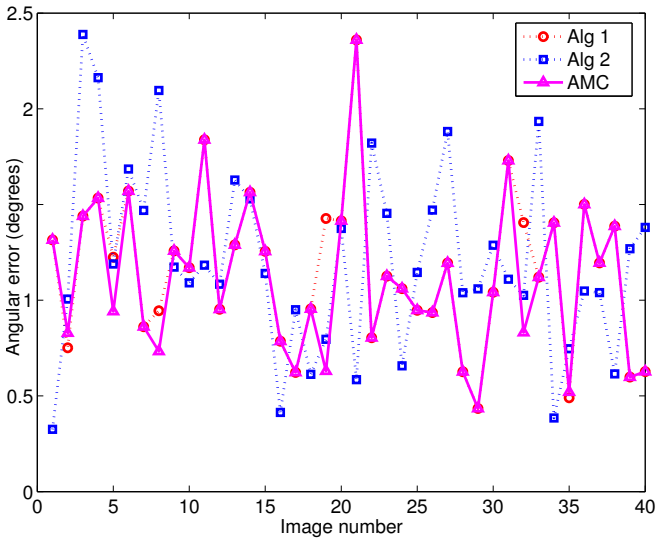

Fig. 3. Variation in degrees in pose estimate due to pixel noise.

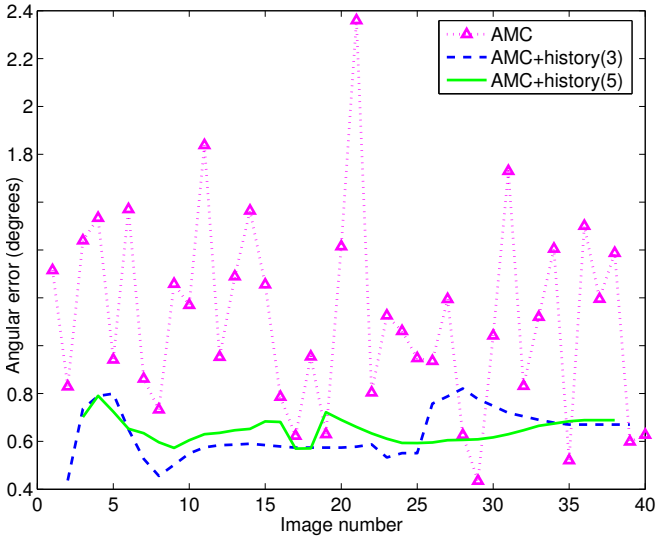

Fig. 4. Variation in pose estimate when previous estimates are integrated.

over three pose estimates and the solid curve is the result when the mean is calculated over five pose estimates. It is clear that both the mean and the standard deviation is decreased compared to only using multi-cue integration between algorithms. For example, the mean error has decreased by 42 percent and the standard deviation has decreased by 87 percent when averaging is made over five pose estimates (the solid curve) as compared to only using multi-cue integration between algorithms (dotted curve).

In a robotic system you usually don't have a number of images from the same camera position, but during movement a number of images can be captured on the way toward the target view. The problem is that the camera will have different positions for the different images and the pose estimates will therefore be different. In Fig. 5 two camera positions are shown; one where the object is close to the border of the

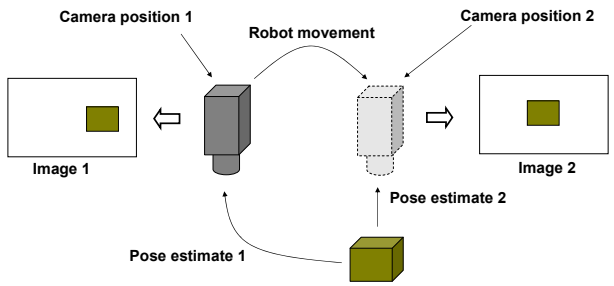

Fig. 5. The setup for multi-cue integration between several views.

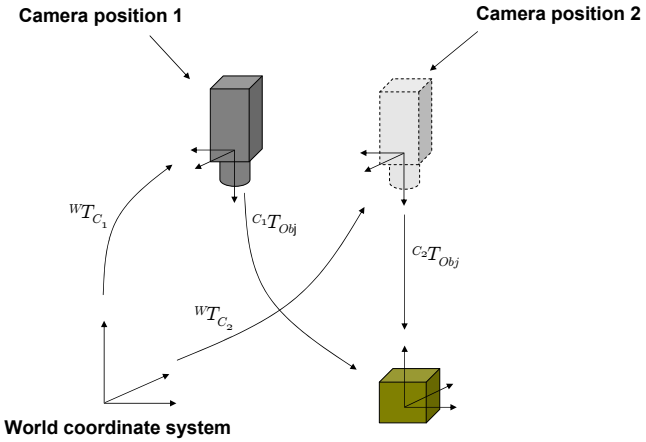

Fig. 6. The different coordinate systems for the setup used in multi-cue integration between several views.

camera image and one where the camera is centered over the object. In this case the position estimate of the object will be completely different and it is not possible to calculate a mean between the two different pose estimates. This problem is of course also present in the more general case when the movement is not constrained to translation in a plane. To work around this problem we need to transform the pose estimates to the same coordinate system before we can calculate a mean of the pose estimates.

In Fig. 6 an illustration of the different coordinate systems is seen; the world coordinate system, the camera coordinate system for two different camera positions and the object coordinate system. In Fig. 6 we have used the notation $^{W}T_{C_1}$, where W stands for the world coordinate system, C_1 stands for camera position number one and the full expression stands for camera position number one relative to the world coordinate system.

From the robot controller and the hand-eye calibration we get the transformations $^{W}T_{C_1}$ and $^{W}T_{C_2}$. The pose estimate $^{C_1}T_{Obj}$ can then be transformed to camera position two by

$$^{C_2}T'_{Obj} = \left[^{W}T_{C_2}\right]^{-1} {}^{W}T_{C_1}\, {}^{C_1}T_{Obj}. \quad (2)$$

If the pose estimate is stored together with the current camera

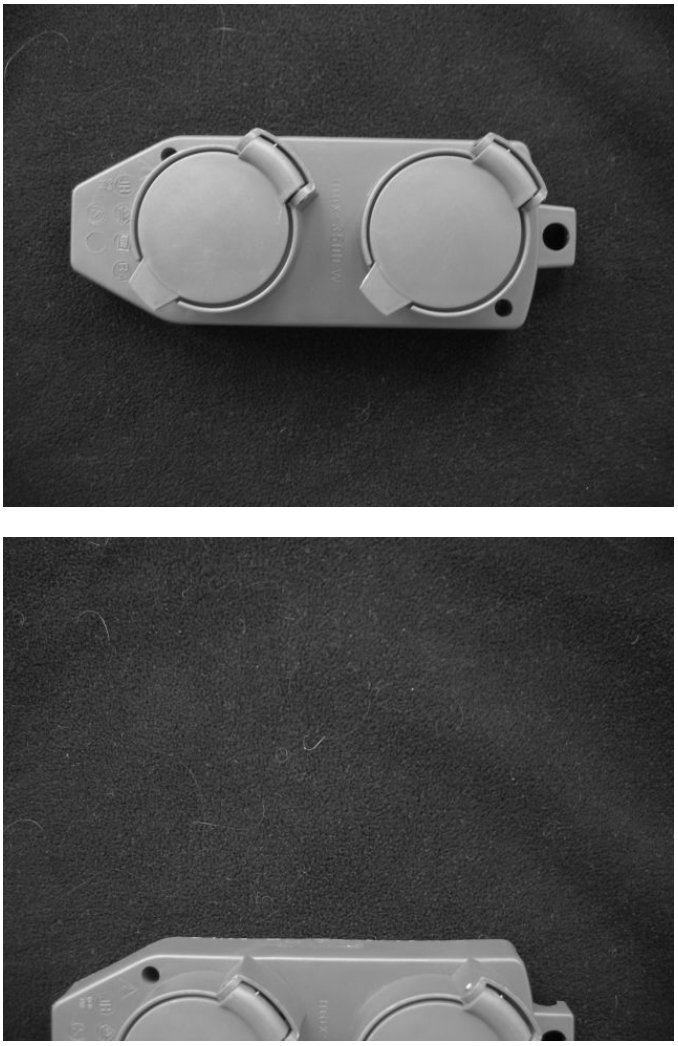

Fig. 7. Differences in seen pose due to relative camera-object position.

position for each step during movement toward the target view it is possible to perform multi-cue integration with previous estimates since we can transform them into the new coordinate system. We call this kind of integration temporal multi-cue integration (TMC).

Ordinary coordinate transformations are not enough to describe a system with a moving camera the resulting changes in estimated pose. This is due to the perspective projection and an example of this effect can be seen in Fig. 7. In Fig. 7 the camera is translated relative to the object but it appears as if the object is also rotated. The algorithms will detect this rotation and before we can do anything else we have to correct the pose estimates for this effect. The correction in θ and ϕ angles is visualized in Fig. 8 and is calculated as

$$\theta_{corr} = \arg(x_{mm} - iy_{mm}) \quad \text{and} \tag{3}$$

$$\phi_{corr} = \arg(z_{std} + i\sqrt{x_{mm}^2 + y_{mm}^2}), \tag{4}$$

where x_{mm} and y_{mm} are the position estimates relative to the camera described in millimeters and z_{std} is the distance in millimeters to the plane through the object and parallel to

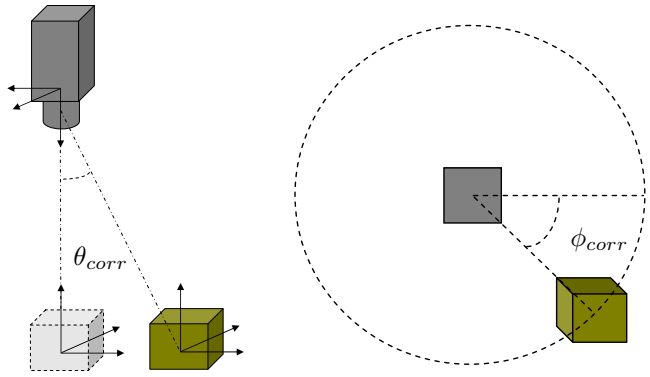

Fig. 8. Errors in pose estimates due to relative camera-object position. In the right part of the figure the camera should be thought of as being in the center of the circle, looking down on the object.

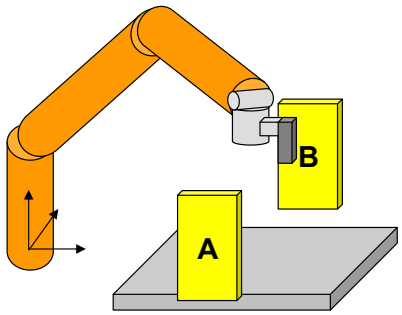

Fig. 9. Schematic of experimental setup.

the image plane. The pose estimates from each algorithm are corrected with these values by a transformation matrix.

The process of TMC can then be divided into the following steps. For a certain camera position all previous pose estimates are first corrected using (3) and (4) and finally transformed using (2). The confidence measurement for these pose estimates are then weighted with the weight w_i

$$w_i = f^{(n-i)}, \tag{5}$$

where i is the step number for the previous pose estimate, n is the current step number and f is the *forget factor*. All previous pose estimates are then used as cues in the same manner as described in section III.

V. Experiments

A. Evaluation platform

A schematic of the experimental setup can be seen in Fig. 9. The lights marked A and B in Fig. 9 were covered with a plastic cover for making the light more diffuse. The lights were placed very low to give a *low-angle lighting*. This combined with having a number of small spotlights in the ceiling, not visible in Fig. 9, produce a somewhat harsh lighting environment since it casts shadows etc. The object was placed on the table between the lights and then the robot

hand and camera was positioned at a random position 450 mm above the table looking down. The robot was made to move in such a way to first center on the object and then move on a sphere to reach a reference position. The number of steps for each part of the movement was set to 7 for movement in a plane and to 6 for the movement on the half-sphere. This enables us to evaluate if the system converges or if it oscillates. In a real system a termination condition based on the predicted level of accuracy should be used instead.

B. Variables

The properties of the test setup that have been varied during the testing phase include the object, lighting, background, number and type of other objects present, level of occlusion of the object to be detected as well as active pose estimation algorithm and state of multi-cue integration.

The objects that have been tested for are a plastic power socket with two outlets, referred to as *socket*, and a cast metal piece of an electrical motor, referred to as *bug*. It should be noted that the two objects are different in size as well as different in the type of internal structure. Even more important is perhaps the differences in surface properties such as reflectiveness and color. This suggest that the results will generalize to a large set of object types.

The lighting conditions used during the tests includes the situation where both light A and B are turned on, referred to as *light 1*, and only light A turned on, referred to as *light 2*. Light 1 is used during training. It should be emphasized that due to the low-angle property of the light, the second light condition is a significant change in lighting.

During the tests we have used a black background, but for several experiments a large part of the background was covered with other objects and we believe this is comparable to a real situation. The reason for not changing the background is that the objects can not be moved if we want to keep ground-truth.

The combinations of pose estimation algorithms and multi-cue integration used during the test were: each algorithm by itself, both of the algorithms with cue integration between them active as well as both of the algorithms with temporal multi-cue integration active.

It should be emphasized that not all combinations of the above mentioned properties have been varied during the tests. A few examples of typical images can be seen in Fig. 10.

C. Parameters and training

The pose estimation algorithms were trained by using 94 images for the socket and 177 for the bug.

All algorithm-specific parameters were kept at the same values throughout all experiments. The settings for the multi-cue integration were also kept constant throughout the experiments. These two statements are very important since they indicate how stable the algorithms really are. It is always easier to get things to work well on a subset of the problem if we are allowed to tune the parameters of the algorithms and this was something we wanted to avoid doing.

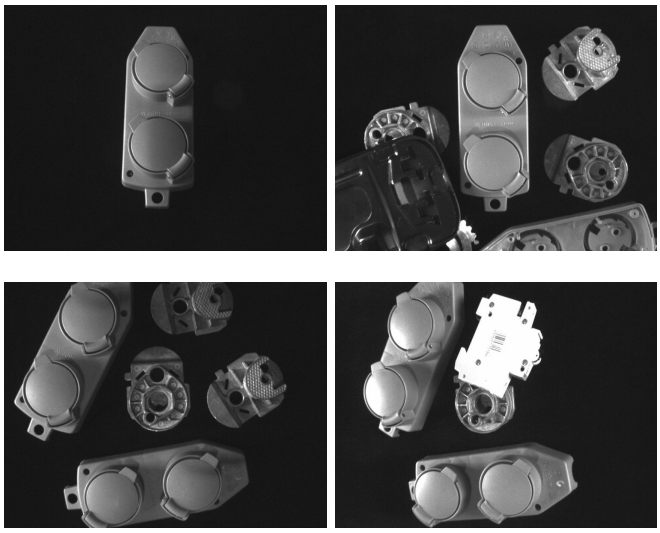

Fig. 10. Example images of changes in scene, lighting and occlusion. The images in the top row are from experiments with the socket. The images in the bottom row are from experiments with the bug.

In the forth-coming presentation of results we will use the following error measures as for the Cartesian position

$$\epsilon_{pos} = \sqrt[3]{(x_t - x_c)^2 + (y_t - y_c)^2 + (z_t - z_c)^2}, \qquad (6)$$

and for angular position

$$\epsilon_{ang} = 2\cos^{-1}(\mathbf{q}_t^T \mathbf{q}_c), \qquad (7)$$

where we used subscript t for training, which is also the reference, position, subscript c for current position and $\mathbf{q}$ is a quaternion of the rotation in column vector format.

VI. Results and Conclusions

A. Conclusions for temporal multi-cue integration (TMC)

As stated in section IV, a system should be able to generate more accurate pose estimates if cues from several views of the observed object are used. In Fig. 11 a testrun is illustrated, where the cartesian and angular distance between robot position and the reference position are plotted for each iteration. We have plotted the result for AMC between two algorithms together with the result for AMC between the same algorithms while also using TMC. Both the mean accuracy and the standard deviation is better with TMC than without TMC for this situation.

For a more complete test of the performace of TMC we made 67 runs for the socket without using TMC and 22 runs using TMC. During the tests the lighting as well as the presence of other objects were varied. The mean error and variance can be seen in table I, where TMC significantly improves the performance. These values are comparable to those found in section IV.

One of the main points of TMC is that it stabilizes the results and reduces the variations in the taken trajectories. How much of a low-pass character TMC has depends on the forget factor γ, which is a temporal weight that adjust the influence

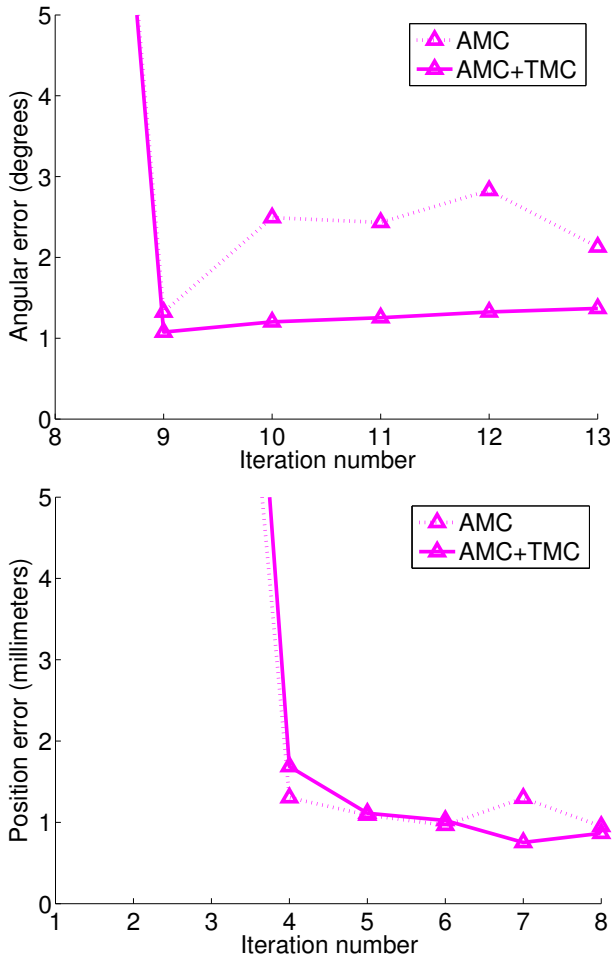

Fig. 11. A comparison between using and not using TMC with AMC (i.e. Algorithm 1+Algorithm 2). The result is for the socket with lighting 2 and no other objects in the scene.

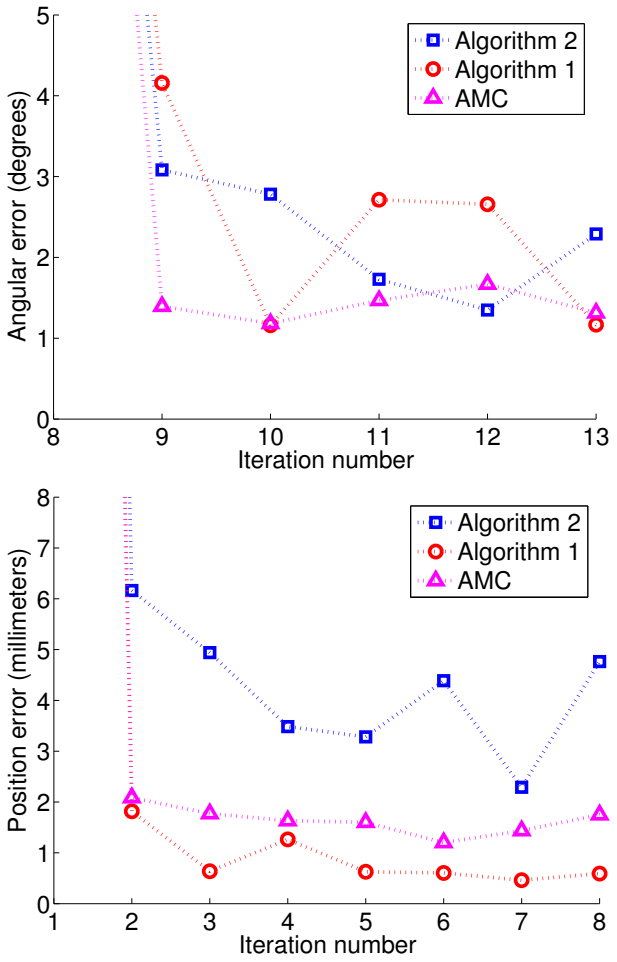

Fig. 12. The result for the socket with lighting 1 and other objects present in the scene. Algorithm 1 has slightly better result than AMC.

	No TMC	TMC
Position mean error	1.28	0.85
Position error variance	0.93	0.17
Angular mean error	1.84	0.98
Angular error variance	1.47	0.10

TABLE I

MEASURED ERRORS AND VARIANCE.

of older pose estimates. In initial tests we varied the forget factor and found $\gamma = 0.7$ to be a quite good choice.

Although the results for TMC is very good it is important to understand that the performance of this algorithm is highly dependent on the 3D position estimate of the object. In these tests, the distance to the object (z-coordinate) is fixed and known by the system. There is therefore no error in the z-coordinate, which definitely improves the result.

One point that could be a problem is if one pose estimate during the purposeful movement toward the target view has large errors. When not using TMC this is either no problem or a very severe one, i.e. the system will either have a second chance in the next iteration assuming that the object is still in the image or it will fail completely if the robot moves such that the object is no longer in the next image. When using TMC the situation is somewhat different. The large pose estimation error will remain in memory and affect intermediate results and possibly also the end result. The effect of this error is reduced by the forget factor, since the influence of older pose estimates is reduced. However, most of the time such an incorrect estimate will not be part of the most prominent cluster once a few correct pose estimates have been obtained.

The results presented above are very typical and convincing, we therefore draw the conclusion that using votes from several views will both increase the accuracy of the system and improve the repeatability of the system, i.e. give a lower standard deviation for the different test runs.

B. Conclusions for algorithmic multi-cue integration (AMC)

In section III it is stated that a system using algorithmic multi-cue integration should be more stable to changes in light, background and target object, since cues from several algorithms are used and integrated. One thing that we took

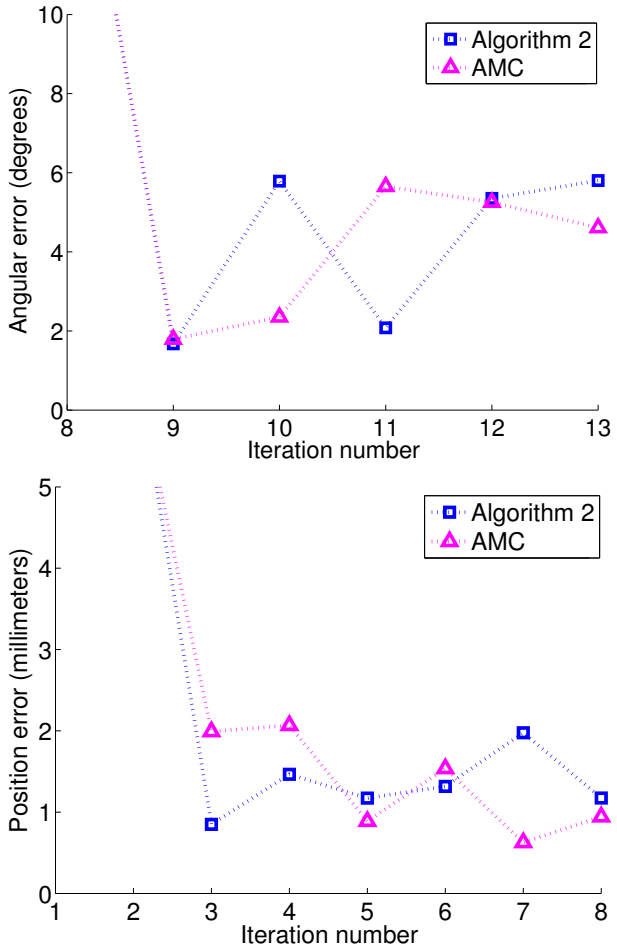

Fig. 13. The result for the socket with lighting 2 and other objects present in the scene. For this situation AMC still works even though algorithm 1 has failed, which is why it is missing.

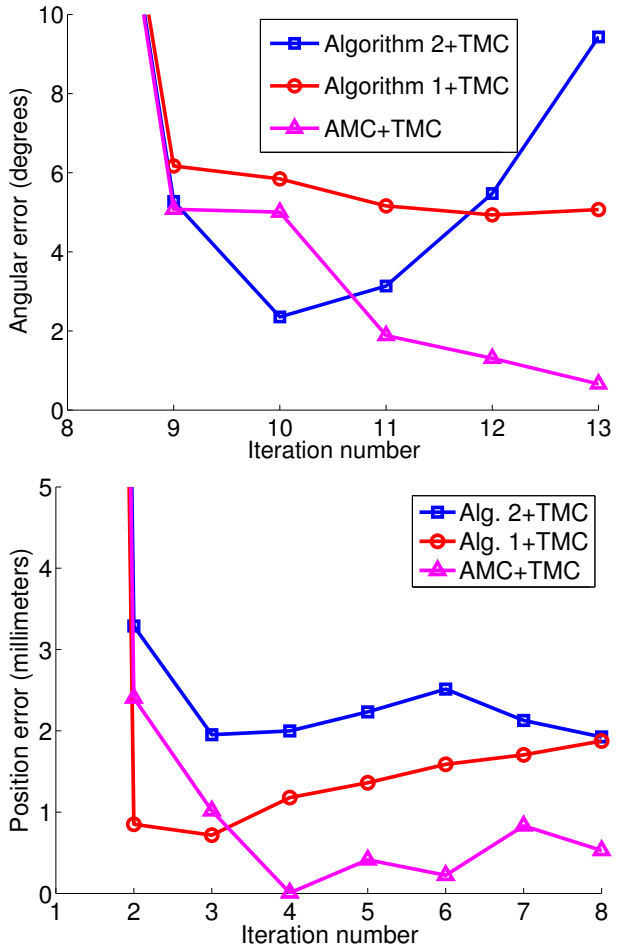

Fig. 14. An illustation where both single algorithms fails and the AMC between the algorithms significantly improves the result. The result is for the bug with lighting 2 and other objects present in the scene.

extra care to monitor during the experiments was that the confidence measures for the different algorithms had the same scale. This is important since it can make all assumptions on the properties of AMC totally wrong due to an imbalance between the algorithms.

We have performed tests for several different situations for the purpose of investigating the benefit of AMC. The following observations have been made:

The first observation is for the socket with other objects in the scene. The lighting condition is varied and the result is illustrated in Fig. 12 and Fig. 13. For those two situations we can see that AMC introduces robustness to the system. For instance, when algorithm 2 has a quite bad Cartesian error for light 1 the AMC result is still very good. Moreover, algorithm 1 fails for light 2 but still the AMC result is stable. Thus we conclude that AMC makes the system more robust to failures by single algorithms, as stated in section III.

We have also observed situations where AMC significantly improves both the angular error and the Cartesian error compared to the single algorithm result, Fig. 14. This could be the case, when the pose vote with the second highest confidence for each algorithm is the "correct" one and for the case of AMC these will cluster and be the end result. This behavior of AMC is referred to algorithmic voting in section III.

Several other tests have been performed to investigate the performance of AMC, but these tests could not be included in this paper due to lack of space. The interested reader is referred to [13], [14]. The conclusion from all these tests is that AMC makes the system more robust to different objects, lighting conditions and other objects in the scene. We have cases where one algorithm fails and AMC still works. There are situations where both algorithms work, but still it seems like AMC improves the result or at least is close to the "best" algorithm.

ACKNOWLEDGMENT

The authors would like to thank the European Commission for sponsoring the VISATEC project, [15], within which all research presented in this paper has been made.

REFERENCES

[1] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. ICCV'99*, 1999.

[2] B. Johansson and A. Moe, "Patch-duplets for object recognition and pose estimation," Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, Tech. Rep. LiTH-ISY-R-2553, November 2003.

[3] F. Vikstén and A. Moe, "Local single-patch features for pose estimation using the log-polar transform," in *2nd Iberian Conference on Pattern Recognition and Image Analysis*. Estoril, Portugal: IAPR, June 2005.

[4] R. Söderberg, K. Nordberg, and G. Granlund, "An invariant and compact representation for unrestricted pose estimation," in *2nd Iberian Conference on Pattern Recognition and Image Analysis*. Estoril, Portugal: IAPR, June 2005.

[5] C. Bräutigam, J.-O. Eklundh, and H. Christensen, "A model-free voting approach for integrating multiple cues," in *Computer Vision - ECCV'98: 5th European Conference on Computer Vision*. Freiburg, Germany: ECCV, June 1998, p. 734.

[6] D. Kragic and H. Christensen, "Cue integration for visual servoing," in *IEEE Transactions on Robotics and Automation*. IEEE, 2001.

[7] F. Furesjö, "Multiple cue object recognition," CVAP, NADA, Royal Institute of Technology, SE-100 44 Stockholm, Sweden, Lic. Thesis ISSN 0348–2952, July 2005, iSBN 91–7283–972–4.

[8] W. Jang and Z. Bien, "Feature-based visual servoing of an eye-in-hand robot with improved tracking performance," in *Robotics and Automation*. Sacramento, CA, USA: IEEE, April 1991, pp. 2254–2260 vol.3.

[9] P. I. Corke, *Visual Control of Robots: High Performance Visual Servoing*. Research Studies Press, 1996, iSBN: 0-86380-207-9.

[10] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, 1975.

[11] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, August 1995.

[12] P.-E. Forssén, "Low and medium level vision using channel representations," Ph.D. dissertation, Linköping University, Sweden, SE-581 83 Linköping, Sweden, March 2004, dissertation No. 858, ISBN 91-7373-876-X.

[13] F. Vikstén, "Methods for vision-based robotic automation," Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, Lic. Thesis LiU-Tek-Lic-2005:16, April 2005, thesis No. 1161, ISBN 91–85299–37–5.

[14] R. Söderberg, "Compact representations and multi-cue integration for robotics," Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, Lic. Thesis LiU-Tek-Lic-2005:15, April 2005, thesis No. 1160, ISBN 91–85299–36–7.

[15] URL: http://www.visatec.info.