Template matching for large transformations

Julian Eggert, Chen Zhang, Edgar Körner

2007

Template matching for large transformations

Julian Eggert[1], Chen Zhang[2] and Edgar Körner[1]

[1] Honda Research Institute Europe GmbH
Carl-Legien-Strasse 30, 63073 Offenbach/Main, Germany
[2] Darmstadt University of Technology, Institute of Automatic Control, Control Theory and Robotics Lab, 64283 Darmstadt, Germany

Abstract. Finding a template image in another larger image is a problem that has applications in many vision research areas such as models for object detection and tracking. The main problem here is that under real-world conditions the searched image usually is a deformed version of the template, so that these deformations have to be taken into account by the matching procedure. A common way to do this is by minimizing the difference between the template and patches of the search image assuming that the template can undergo 2D affine transformations. A popular differential algorithm for achieving this has been proposed by Lucas and Kanade [1], with the disadvantage that it works only for small transformations. Here we investigate the transformation properties of a differential template matching approach by using resolution pyramids in combination with transformation pyramids, and show how we can do template matching under large-scale transformations, with simulation results indicating that the scale and rotation ranges can be doubled using a 3 stage pyramid.

1 Introduction

Image registration using template matching, either directly on a pixel image or on an array of images that result from an appropriate preprocessing step on an image, is a fundamental step that serves as basis for many vision algorithms. The most straightforward way is to take the patch containing the template, overlay it onto the search image at all desired transformations (e.g. positions, rotations, etc.), and calculate a matching score that indicates how well the transformed template matches with the search image for each particular transformation.

In visual object classification and detection, this is the case for connectionist models that use nonlinearities alternated with correlation-based patch template matching with feature-sets in a weight-sharing architecture [2–4]. The weight sharing activity calculation basically corresponds to a feature search at all positions of the input image. More complex transformations of the templates (features) are usually not considered or included explicitly by building all transformation variants of a basis feature.

In template-based tracking, the picture is similar, with the difference that we can restrict the search to those templates and transformations that are likely to occur according to the tracker state predictions. A third field of research where template matching is important is motion processing (see e.g. [5]). Here, the task is to find patch-to-patch correspondences between two images from consecutive timesteps, in order to extract a displacement field that indicates how the different patches move from one timestep to the next.

The main problem for template matching is the number of transformations that have to be checked in order for the procedure to cope with deformations. The appearance of real-world objects undergo severe changes as the objects e.g. translate, rotate, come closer or rotate in 3D. Some of these transformations can (and have to) by covered by the template matching procedure, while others like true 3D appearance changes can only be captured as approximations and only if the transformations remain sufficiently small. This is the case e.g. for rotations in depth and approximately planar objects, whose transformation can then be approximated by a projective transformation.

A popular approach is to introduce 2D transformations into the matching process, in particular 2D affine transformations covering rotation, scaling and shearing in addition to translation. That is, we then search the best match between a search image and a template subject to its tansformations. In order to achieve this, an extensively large number of transformed templates has to be compared with the search image, corresponding to all possible combination of transformations.

Three things can be done to alleviate the costs of matching under transformations. Firstly, not all transformations are equally important and interdependent, so that we can sometimes search for separate transformation dimensions independently. As a second point, we can assume small differences between template and search image (introduced by small transformations), linearize and try to calculate the template match for small transformations computationally more effectively. And third, we can introduce search strategies for the transformation parameters, e.g. by sampling the transformation parameter space first coarsely to get a hint on the transformation range and then refining the search.

In this paper, we combine points 1, 2 and 3 by introducing a resolution pyramid in combination with a transformation pyramid that allows to estimate affine transformations for the image matching problem over a broad range of parameters, calculating first the coarse transformations and refining them in the successive stages of the pyramid. Although pyramidal approaches have been proposed already a number of times in the vision systems community, here we address explicitly the question of transformation pyramids for template matching, analysing the potential of such methods for large scale transformations in combination with Lucas-Kanade type 2D affine matching methods.

In the next section, we sketch the architecture of the approach. To this end, we first summarize a popular differential approach for 2D affine template matching and then show how we utilize it in a transformation pyramid. In the third section, we show in simulations how the approach performs for large-scale transformations. We show exemplar results of valid transformation ranges (since these depend also on intrinsic object characteristics and generally cannot be formulated for arbitrary objects).

2 Approach

2.1 Template matching with 2D affine transformations

A popular approach for template search under small transformations that works well for the affine case has been introduced by Lucas and Kanade [1] and used for many extensions like tracking of objects by means of point features and the appearance transformations of the image patches around these points [6].

To compare two images we start with image points $\mathbf{x} = (x, y)^T$ a template $T(\mathbf{x})$ and its "mask" resp. window of validity $M(\mathbf{x})$ (either binary or continuous, but zero outside of the region of validity of the template) on one hand, and the search image $I(\mathbf{x})$ with its warping transformation $\mathbf{W}(\mathbf{x})$ on the other hand. For this paper, we restrict $\mathbf{W}(\mathbf{x})$ to be a linear transformation composed of an affine transformation matrix $\mathbf{A}$ and a translation vector $\mathbf{d}$, so that an image position $\mathbf{x}$ transforms according to

$$\mathbf{x} \rightarrow \mathbf{W}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d} \ . \tag{1}$$

The target of the template match is to find the parameters $\mathbf{A}$ and $\mathbf{d}$ which minimize a functional

$$F = \int [I(\mathbf{A}\mathbf{x} + \mathbf{d}) - T(\mathbf{x})]^2 M(\mathbf{x}) d\mathbf{x} \ , \tag{2}$$

that is, which leads to the best Euclidean match between template and image under consideration of the geometrical transformation eq. 1 of the search image $I(\mathbf{x})$.

The reason that we transform the search image $I(\mathbf{x})$ and not the template $T(\mathbf{x})$ is that we consider the window $M(\mathbf{x})$ to be attached to the template. If we then transform the template, we would have to transform the window as well, which makes the derivation more complicated (nevertheless, this is a matter of interpretation of eq. 2 since template and search image are exchangeable). For tracking applications, or if we are interested in keeping the template fixed, the inverse transformation from the template to the search image can be easily calculated according to $\mathbf{A}^{-1}\mathbf{x} - \mathbf{A}^{-1}\mathbf{d}$ (e.g., if we want to say: "The pattern in the search image corresponds to the template rotated by ... degrees").

For small affine transformations it makes sense to write $\mathbf{A} = \mathbf{1} + \mathbf{D}$ (small deviation $\mathbf{D}$ from the unity matrix $\mathbf{1}$), with the deformation matrix

$$\mathbf{D} = \begin{pmatrix} d_1 & d_3 \\ d_2 & d_4 \end{pmatrix} \tag{3}$$

now completing the transformation parameters together with the displacement vector

$$\mathbf{d} = \begin{pmatrix} d_5 \\ d_6 \end{pmatrix} . \tag{4}$$

The 6 transformation parameters (4 for the deformation matrix $\mathbf{D}$ and 2 for the displacement vector $\mathbf{d}$) can be collected in a vector

$$\mathbf{z} = (d_1, d_2, d_3, d_4, d_5, d_6)^T \ . \tag{5}$$

A (local) minimization of the functional $F(\mathbf{z})$ can then be achieved by using gradient-descent. Nevertheless, since the warping of the image is the expensive step, it is desirable to avoid too many iterations during the gradient descent. To do few iterations, here we use the Newton method, setting $\nabla_{\mathbf{z}} F(\mathbf{z}) = 0$, with

$$\nabla_{\mathbf{z}} F(\mathbf{z}) = \int 2\left[I(\mathbf{Ax}+\mathbf{d}) - T(\mathbf{x})\right] \nabla_{\mathbf{z}} I(\mathbf{Ax}+\mathbf{d}) M(\mathbf{x}) d\mathbf{x} \tag{6}$$

and linearizing and solving the equation system repetitively for $\mathbf{z}$, as indicated in [7].

We assume smooth changes in the search image. Linearization with respect to $\mathbf{x}$ then yields

$$I(\mathbf{Ax}+\mathbf{d}) \approx I(\mathbf{x}) + \left[\nabla_{\mathbf{x}} I(\mathbf{x})\right]^T (\mathbf{Dx}+\mathbf{d}) \tag{7}$$

which we use to get

$$\nabla_{\mathbf{z}} I(\mathbf{Ax}+\mathbf{d}) \approx \left[\nabla_{\mathbf{x}} I(\mathbf{x})\right]^T \left[\nabla_{\mathbf{z}} (\mathbf{Dx}+\mathbf{d})\right] \tag{8}$$

with the $2*6$-matrix (for a 2-dimensional vector $\mathbf{v}$)

$$\nabla_{\mathbf{z}} \mathbf{v} = \begin{bmatrix} (\nabla_{\mathbf{z}} v_1)^T \\ (\nabla_{\mathbf{z}} v_2)^T \end{bmatrix} . \tag{9}$$

It is straightforward to calculate

$$\begin{aligned} \mathbf{g}(\mathbf{x}) &:= \left[\nabla_{\mathbf{x}} I(\mathbf{x})\right]^T \left[\nabla_{\mathbf{z}} (\mathbf{Dx}+\mathbf{d})\right] \\ &= \left[x\frac{\partial I(\mathbf{x})}{\partial x}, x\frac{\partial I(\mathbf{x})}{\partial y}, y\frac{\partial I(\mathbf{x})}{\partial x}, y\frac{\partial I(\mathbf{x})}{\partial y}, \frac{\partial I(\mathbf{x})}{\partial x}, \frac{\partial I(\mathbf{x})}{\partial y}\right]^T \end{aligned} \tag{10}$$

and setting $\nabla_{\mathbf{z}} F(\mathbf{z}) = 0$ and extracting $\mathbf{z}$, we arrive at the 6-dimensional linear equation system

$$\mathbf{Tz} = \mathbf{a} \tag{11}$$

with

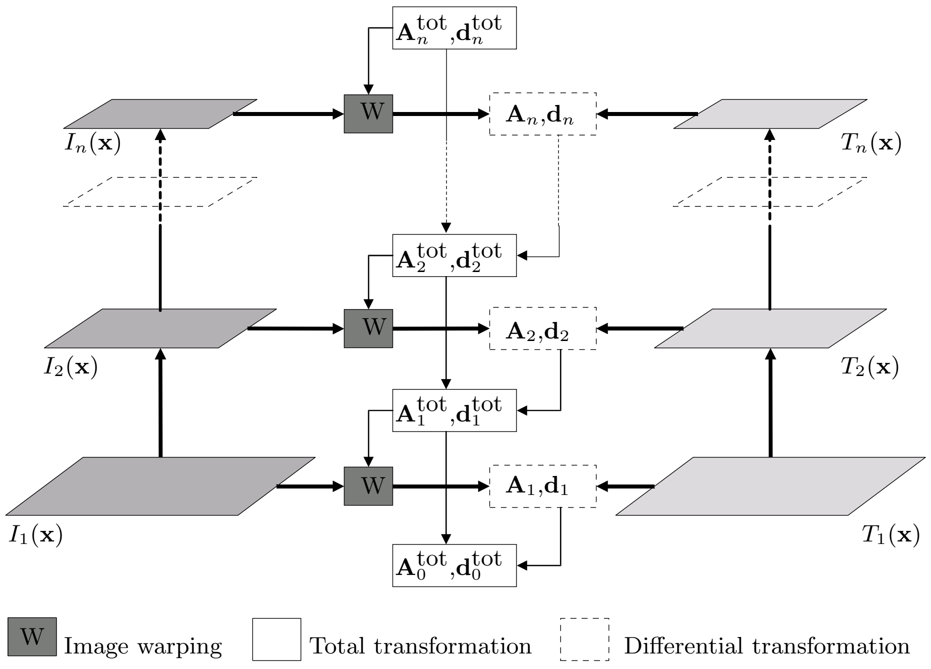
$$\mathbf{T} = \int M(\mathbf{x}) \left\{\mathbf{g}(\mathbf{x})[\mathbf{g}(\mathbf{x})]^T\right\} d\mathbf{x} \quad \text{and} \tag{12}$$

$$\mathbf{a} = \int M(\mathbf{x}) \left[T(\mathbf{x}) - I(\mathbf{x})\right] \mathbf{g}(\mathbf{x}) d\mathbf{x} \tag{13}$$

that can be solved $\mathbf{z} = \mathbf{T}^{-1}\mathbf{a}$ by inverting $\mathbf{T}$.

2.2 Template matching in resolution and transformation pyramids

The problem of the method from section 2.1 is that it works well as long as the gradients introduced by eqs. 7 and 8 and incorporated into the method by eq. 10 provide sufficient information. For large transformation parameters, this may not be the case any more. A remedy then is to use coarser resolutions



$\mathbf{A}_n^{\text{tot}}$,$\mathbf{d}_n^{\text{tot}}$
W
$\mathbf{A}_n$,$\mathbf{d}_n$
$I_n(\mathbf{x})$
$T_n(\mathbf{x})$
$\mathbf{A}_2^{\text{tot}}$,$\mathbf{d}_2^{\text{tot}}$
W
$\mathbf{A}_2$,$\mathbf{d}_2$
$I_2(\mathbf{x})$
$T_2(\mathbf{x})$
$\mathbf{A}_1^{\text{tot}}$,$\mathbf{d}_1^{\text{tot}}$
W
$\mathbf{A}_1$,$\mathbf{d}_1$
$I_1(\mathbf{x})$
$T_1(\mathbf{x})$
$\mathbf{A}_0^{\text{tot}}$,$\mathbf{d}_0^{\text{tot}}$
W Image warping
Total transformation
Differential transformation

Fig. 1. Transformation and resolution pyramid for the estimation of large-scale transformations. On top of the pyramid coarse transformation estimates are calculated on coarse resolutions of the search and template images $I_n(\mathbf{x})$ and $T_n(\mathbf{x})$, which are used to warp the search image and refine the transformation estimate in the successively lower stages of the pyramid. $\mathbf{A}_n^{\text{tot}}$,$\mathbf{d}_n^{\text{tot}}$ is an initial transformation based on prior knowledge, $\mathbf{A}_0^{\text{tot}}$,$\mathbf{d}_0^{\text{tot}}$ is the overall transformation gained from the entire transformation pyramid.

which smoothen the image, like in a Gaussian resolution pyramid, and combine them with a transformation pyramid that allows to calculate the total transformation as a concatenation of single differential transformations for each stage. (Transformation pyramids have been proposed repetitively in different contexts but mostly in combination with translational transformations, see e.g. [8] for an early proposal and [9] for an application of the same principle in a motion estimation system. Here we show in simulations to what extent they can be applied to the Lucas and Kanade type image matching procedures.)

The idea is to use a pyramid with n levels (1: original, n: coarsest resolution, images and templates at different resolutions $I_i(\mathbf{x})$, $T_i(\mathbf{x})$), and apply the procedure on the coarsest level n with $I_n(\mathbf{x})$, $T_n(\mathbf{x})$ to get a first, rough estimation of the transformation parameters $\mathbf{A}_n$, $\mathbf{d}_n$, from which we get the first total transformation estimate $\mathbf{A}_{n-1}^{\text{tot}}$, $\mathbf{d}_{n-1}^{\text{tot}}$ (the indices are chosen indicating that this transformation is the currently best estimate for level $n-1$). Afterwards, we use the transformation parameters to warp the search image $I_{n-1}(\mathbf{x})$ in order to compare it with the template $T_{n-1}(\mathbf{x})$ for a refinement of the transformation parameters. As a result we then get $\mathbf{A}_{n-1}$, $\mathbf{d}_{n-1}$, which has to be composed with $\mathbf{A}^{\text{tot}}_{n-1}$, $\mathbf{d}^{\text{tot}}_{n-1}$ to get the improved estimate of the transformation parameters $\mathbf{A}^{\text{tot}}_{n-2}$, $\mathbf{d}^{\text{tot}}_{n-2}$, with

$$\mathbf{A}^{\text{tot}}_{i-1} := \mathbf{A}_i \, \mathbf{A}^{\text{tot}}_i \tag{14}$$

and

$$\mathbf{d}^{\text{tot}}_{i-1} := \mathbf{d}_i + \mathbf{d}^{\text{tot}}_i \; . \tag{15}$$

This has to be repeated until we arrive at the lower end of the pyramid which works on the images at original resolution. The expectation is that, since each stage of the pyramid already receives a search image that was moved closer to the template image (in terms of Euclidean match), the matching procedure from sec. 2.1 can be applied further to improve the transformation estimation, allowing to find image matches over a much broader range of parameters.

Figure 1 shows a schema of the resolution and transformation pyramid with the mentioned warping, transformation estimation and transformation concatenation steps. If we are e.g. in a tracking application, we usually do already have some initial estimate (from previous steps) of the overall transformation to start with, which we can include as $\mathbf{A}^{\text{tot}}_n$, $\mathbf{d}^{\text{tot}}_n$ to warp the search image $I_n(\mathbf{x})$ at the top of the pyramid. On the lower end of the pyramid, we get our total transformation estimate $\mathbf{A}^{\text{tot}}_0$, $\mathbf{d}^{\text{tot}}_0$. In the following examples, we used a Gaussian resolution pyramid with 3 levels, applied on images of $128x128$ pixel size, so that at each level of the pyramid the resolution halfened. The change in resolution has to be taken into accout in the calculation of the total translation since in eq. 15, $\mathbf{d}^{\text{tot}}_{i-1}$ and $\mathbf{d}^{\text{tot}}_i$ were assumed to operate on the same spatial scale. For different spatial scales, the translation vectors $\mathbf{d}$ have to be normalized, so that for our case of the Gaussian pyramid, we used $\mathbf{d}^{\text{tot}}_{i-1} := \mathbf{d}_i + 2\mathbf{d}^{\text{tot}}_i$.

3 Results for large-scale transformations

The size of transformations that can be estimated with our method depend on the particular structure of the template and search images. Two types of properties are beneficial for the estimation of large transformations: 1. There has to be sufficient structure (so that there are pronounced minima in the functional eq. 2) and 2. the structure has to be sufficiently smooth so that gradients can drive the search towards a solution. Point 2 also implies that the minima are not too narrow, which is the case e.g. for templates with no pronounced autocorrelation lengths (in the simple case of a random pattern template created using white noise, there is basically no gradient information that can be used to find the minimum, if search image and template are more than one pixel offset).

To quantify the results of simulations with the presented algorithm, we took arbitrary single objects from the COIL [3] database. We chose not to average the results over a large image database (e.g., the entire COIL or more complex image databases) because of the dependency on the intrinsic object properties explained

[3] Columbia Object Image Library

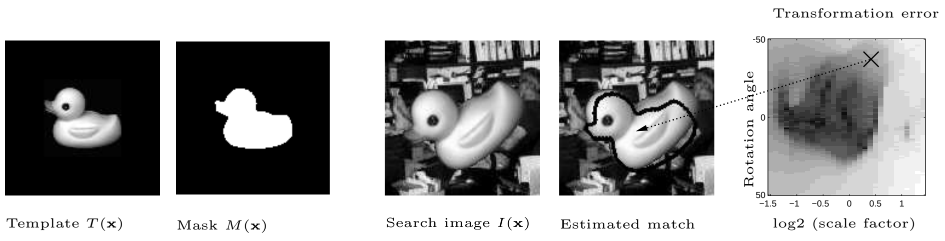


Fig. 2. Typical template, mask, (cluttered) search image and match result. The task is to find the transformation between the template and the rotated and scaled duck. On the 2 rightmost images, the match on the search image (indicated by a black contour) and the affine transformation error for different scaling factors and rotation angles are shown. The cross indicates the current target transformation (-40 degree rotation and scale factor 1.5) that was used to generate the search image. At the right, a logarithmic scale (of basis 10: e.g. -2 are errors in the 10^{-2}-range) indicates the transformation error (Euclidean distance between the gained and the true affine transformation matrix entries). Darker region correspond to good transformation matches, for the cross position the transformation estimation error is already considerable, so that the match is not perfect.

above. Nevertheless, even from single objects the benefits of the method for large scale transformations can be evaluated.

We used 2 different paradigms: In the clean condition, we searched for a match between the original and the transformed version of the object and in the clutter condition, we searched a transformed version of the object in an image with clutter.

In the following two figures, we evaluated the affine transformation matrix error. We generated search images that were a rotated and scaled version of the template and then run the method with a transformation pyramid of $1-3$ layers. The resulting transformation matrix at each of the layers was then compared to the "real" transformation matrix, calculating the Euclidean distance of all its entries, shown in fig. 3. For easier visualization, we restricted the plots to affine transformations that are a combination of homogeneous scaling and rotation. It can be seen that the transformation pyramid increases the validity regions substantially.

To get a better quantitative idea of the estimation errors and to see how the error distributes among scaling and rotation parameters, we extracted from the affine transformation results a scaling parameter λ and a rotation angle α that approximate $\mathbf{A}$ according to

$$\mathbf{A} = \begin{pmatrix} a_1 \; a_3 \\ a_2 \; a_4 \end{pmatrix} \approx \begin{pmatrix} \lambda \; 0 \\ 0 \; \lambda \end{pmatrix} \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \; . \tag{16}$$

Afterwards, we calculated the absolute difference between the real scale and orientation (used to generate the transformed objects for the search image) and the results after applying the transformation pyramid. Figs. 4 and 5 show the results for scaling and rotation separately.

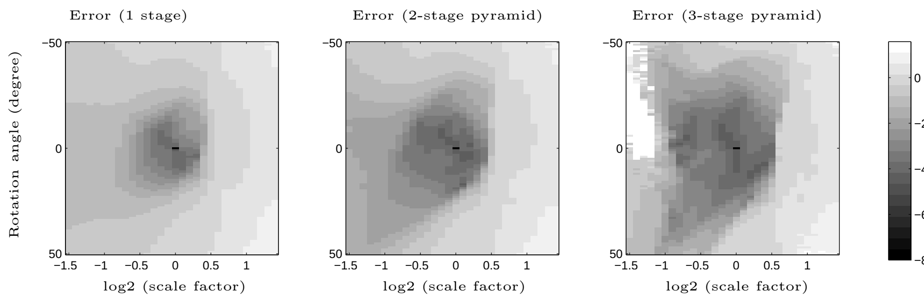


Fig. 3. Clean condition, Euclidean distance between real and computed affine transformation matrix for template matching pyramids consisting of 1, 2 and 3 stages, applied on the duck picture. The x and y-axes show the scaling factor (log 2: -1 meaning half size, $+1$ meaning double size) and the rotation angle of the template object in the search image. Darker regions of the graphs denote more accurate estimations of the affine transformation. Notice the increase in size of the dark region for an increasing number of stages, indicating that the pyramidal matching procedure is able to cope with scaling factors of nearly $0.5 - 2.0$ and rotation angles of about ± 20 degrees.

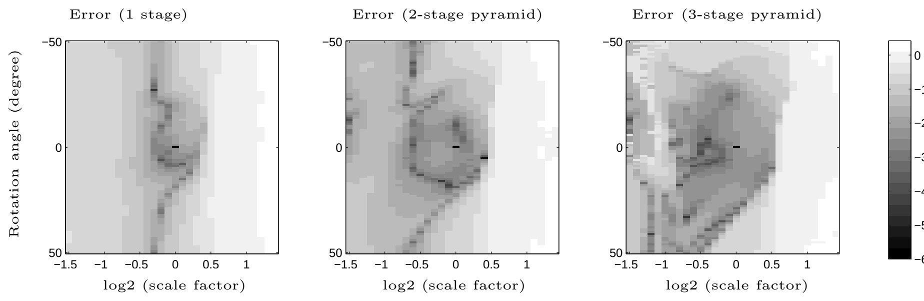


Fig. 4. Clean condition, scaling error for a template matching pyramid consisting of 1, 2 and 3 stages. The x and y-axes again show the scaling factor and the rotation angle of the template objects in the search image.

Figure 6 shows the results for a search image that contains a transformed template object on a cluttered background. It can be seen that the gain (in terms of a larger valid region where transformation parameters are accurately estimated) is even larger than in the clean condition from fig. 3.

4 Conclusion

The translational parameters play a special role in the transformation estimation process. In many cases, the estimation of the full system eq. 11 leads to spurious or false minima, specially if the initial translational mismatch between template and object in the search image is large [7]. Then it is beneficial to separate the estimation of the translational parameters **d** from the estimation of the

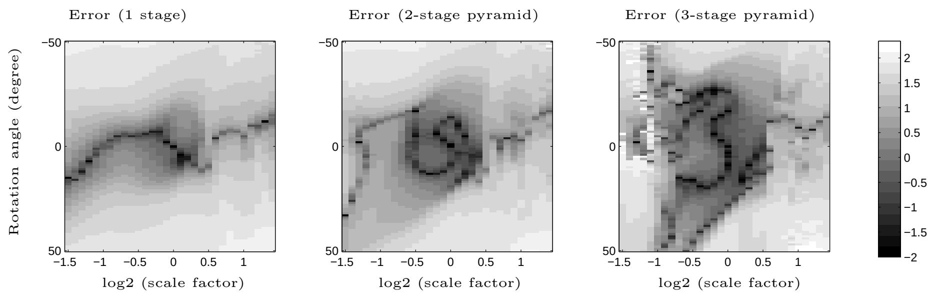


Fig. 5. Clean condition, rotation error for a template matching pyramid consisting of 1, 2 and 3 stages. The x and y-axes again show the scaling factor and the rotation angle of the template objects in the search image.

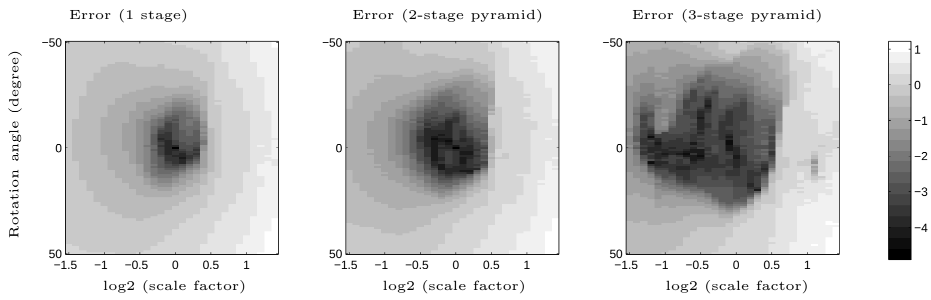


Fig. 6. Clutter condition, affine transformation error for a template matching pyramid consisting of 1, 2 and 3 stages. The x and y-axes again show the scaling factor and the rotation angle of the template objects in the search image. Although the absolute error is larger than for the clean case (fig. 6), the increase of the region of valid transformation estimations is even larger when going from 1 level to a 3-level pyramid.

affine transformation parameters $\mathbf{A}$. This is something that we observed for the pyramid method at all levels. We therefore estimated first the translation at each level, and afterwards the affine transformation.

In the simulations we found that for the affine parameter estimation using a pyramid with 3 levels, a single iteration of the Newton method from section 2.1 at each level already suffices to produce good matching results. The costly part of the method, the warping of the search images at each pyramid level (see fig. 1), therefore occurs only 3 times (and only once for the full resolution image).

Since the Euclidean match in the functional eq. 2 is sensitive to differences over the entire region of the mask $\mathbf{M}(\mathbf{x})$, it is important that the mask matches covers only the relevant parts of the template. In our case, we used a binary mask calculated from the object itself by thresholding against a zero background (see fig. 2). As soon as the mask and the template mismatch, the transformation estimations degrade. Therefore, in tasks which require a template update, like e.g. when a real object is being tracked which changes its appearance beyong 2D affine transformations, care has to be taken that the mask is updated consistently with the template.

The number of pyramid levels depends on the size and the structure/texture of the template, since fine details are lost with increasingly coarse resolution. In our case, the 3-level pyramid provided to be a good choice; with more levels the results degraded since the highest level then did not have sufficient clues to estimate its transformation correctly. One way to estimate the number of needed pyramid levels without a fully extensive check is to regard the surface-averaged reconstruction error of the highest hypothetical pyramid level only (the level with the coarsest resolution), starting from the lowest possible resolution, and increasing step-by-step the number of pyramid levels, searching for a minimum.

All shown plots were gained using the same image (see fig. 2), which provided a good example with sufficiently detailed form but not too much resolution, so that it still presented a challenge for the template matching process. For other objects of the COIL database (which all have a comparable size), the gained results were very similar in terms of gain of applicable transformation range.

Summarizing, we have shown that transformation pyramids considerably extend the range of transformations that can be covered by template-matching procedures of the Lucas and Kanade type. Transformation ranges for objects increased from approx. $[0.65 - 1.4]$ to $[0.5 - 1.75]$ for the scaling factor and from $[-15, 15]$ to nearly $[-30, 30]$ degrees for the rotation angle, providing good results for situations with cluttered background.

References

1. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: International Joint Conference on Artificial Intelligence. (1981) 674–679
2. Wersing, H., Körner, E.: Learning optimized features for hierarchical models of invariant recognition. Neural Computation **15**(7) (2003) 1559–1588
3. Riesenhuber, M., Poggio, T.: Hierarchical models of object recognition in cortex. Nature Neuroscience **2**(11) (1999) 1019–1025
4. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics **39** (1980) 139–202
5. Willert, V., Eggert, J., Adamy, J., Körner, E.: Non-gaussian velocity distributions integrated over space, time and scales. IEEE Transactions on Systems, Man and Cybernetics B (2006)
6. Tomasi, C., Kanade, T.: Detection and tracking of point features. Technical report, Carnegie Mellon University (1991)
7. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), Seattle (1994)
8. Bergen, J.R., Anandan, P., Hanna, K.J., Hingorani, R.: Hierarchical model-based motion estimation. In: ECCV '92: Proceedings of the Second European Conference on Computer Vision, London, UK, Springer-Verlag (1992) 237–252
9. Eggert, J., Willert, V., Körner, E.: Building a motion resolution pyramid by combining velocity distributions. In: 26th Pattern Recognition Symposium DAGM. (2004) 310–317