FAsT-Match: Fast Affine Template Matching

Simon Korman
Tel-Aviv University

Daniel Reichman
Weizmann Institute

Gilad Tsur
Weizmann Institute

Shai Avidan
Tel-Aviv University

Abstract

Fast-Match is a fast algorithm for approximate template matching under 2D affine transformations that minimizes the Sum-of-Absolute-Differences (SAD) error measure. There is a huge number of transformations to consider but we prove that they can be sampled using a density that depends on the smoothness of the image. For each potential transformation, we approximate the SAD error using a sublinear algorithm that randomly examines only a small number of pixels. We further accelerate the algorithm using a branch-and-bound scheme. As images are known to be piecewise smooth, the result is a practical affine template matching algorithm with approximation guarantees, that takes a few seconds to run on a standard machine. We perform several experiments on three different datasets, and report very good results. To the best of our knowledge, this is the first template matching algorithm which is guaranteed to handle arbitrary 2D affine transformations.

1. Introduction

Image matching is a core computer vision task and template matching is an important sub-class of it. We propose an algorithm that matches templates under arbitrary 2D affine transformations. The algorithm is fast and is guaranteed to find a solution that is within an additive error of the global optimum. We name this algorithm: FAsT-Match.

Template matching algorithms usually consider all possible translations. They differ in the way they discard irrelevant translations (see Ouyang *et al.* [15] for a comprehensive survey of the topic). Template matching under more general conditions, which include also rotation, scale or 2D affine transformation leads to an explosion in the number of potential transformations that must be evaluated.

Fast-Match deals with this explosion by properly discretizing the space of 2D affine transformations. The key observation is that the number of potential transformations that should be evaluated can be bounded based on the assumption that images are smooth. Small variations in the parameters of the transformation will result in small variations in the location of the mapping, and because of the image smoothness assumption, the Sum-of-Absolute-Difference (SAD) error measure will not change much.

Given a desired accuracy level δ we construct a net of transformations such that each transformation (outside the net) has an SAD error which differs by no more than δ from that of some transformation in the net. For each transformation within the net we approximate the SAD error using random sampling. When δ is small the net size becomes large and we apply a branch-and-bound approach. We start with a sparse net, discard all transformations in the net whose errors are not within a bound from the best error in the net and then increase the sampling rate around the remaining ones.

It is instructive to contrast Fast-Match with classical direct methods, such as Parametric Optical Flow (OF) [11]. OF methods improved considerably over the years and are the building blocks of many computer vision applications. However, at their core OF are solving a nonlinear optimization problem and as such they rely on an initial guess and might be trapped in a local minimum. Fast-Match, on the other hand, does not rely on an initial guess and is guaranteed to find an approximation to the global optimum.

To overcome the limitations of OF there is a growing focus on feature based methods, such as SIFT [10]. Such methods assume that feature points can be reliably detected and matched in both the image and the template so that there are enough potent matches to estimate the global 2D affine transformation, perhaps using RANSAC [4]. Despite the large body of work in this field, the process can fail, especially if there are not enough distinct features in the template or the image. See Figure 1 for illustrations.

OF is clearly less practical when the size of the template is considerably smaller than the size of the image because it does not have a good initial guess. In such cases we can use feature point matching to seed the initial guess of an OF algorithm. However, it is increasingly difficult to detect distinct feature points as the size of the template decreases. Fast-Match does not suffer from this problem.

Fast-Match has some disadvantages when compared to other techniques. OF techniques can reach subpixel accuracies which Fast-Match cannot. Of course, Fast-Match's solution can later be refined using OF techniques. Another limitation is when dealing with images where the important information is sparse, e.g., diagrams and text. In such cases Fast-Match treats background pixels as if they are as

1063-6919/13 $26.00 © 2013 IEEE
DOI 10.1109/CVPR.2013.302

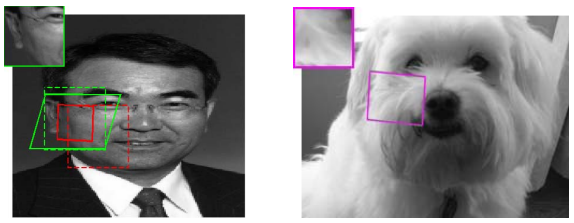


Figure 1. **Shortcomings of current methods:** Left: Direct Methods (OF) **require (good) initialization**. They find the correct template location (green parallelogram) given a close enough initialization (dashed green parallelogram), but might fail (converge to solid red parallelogram) with a less accurate initialization (dashed red parallelogram). Right: **Indirect Methods** (feature based) **require (enough) distinct features**. They typically will not detect a single matching feature in such an example. Fast-Match solves both these cases.

important as foreground pixels, potentially achieving good SAD error at the expense of good localization, in contrast to feature based techniques. Finally, Fast-Match may fail on highly textured images, as the guarantees only hold for smooth images.

While strictly speaking, Fast-Match minimizes the SAD error and our experiments validate this, we also show that minimizing SAD error serves as a proxy to finding the *location* of the template and we show results to this effect. Often, even when the size of the template is small, Fast-Match can still find the correct match, whereas feature based methods struggle to detect and match feature points between the template and the image.

We present a number of experiments to validate the proposed algorithm. We run it on a large number of images to evaluate its performance on templates of different sizes, and in the presence of different levels of degradation (JPEG artifacts, blur, and gaussian noise). We also test Fast-Match on the data sets of Mikolajczyk *et al.* [12, 13]. Finally, we report results on the ZURICH Buildings data-set [19].

2. Background

Our work grew out of the template matching literature which we review next. Since Fast-Match can be used for image matching as well we include a short reference of it. The topic of Image Matching is vast and reviewing it is beyond the scope of this paper.

Template Matching Evaluating only a subset of the possible transformations was considered in the limited context of Template Matching under 2D translation. Alexe *et al.* [1] derive an upper bound on appearance distance, given the spatial overlap of two windows in an image, and use it to bound the distances of many window pairs between two images. Pele and Werman [16] ask "How much can you slide?" and devise a new rank measure that determines if one can slide the test window by more than one pixel.

Extending Template Matching to work with more general transformations was also considered in the past. Fuh *et al.* [6] proposed an affine image model for motion estimation, between images which have undergone a mild affine deformation. They exhaustively search a range of the affine space (practically - a very limited one, with only uniform scale). Fredriksson [5] used string matching techniques to handle also rotation. Kim and Araújo [7] proposed a grayscale template matching algorithm that considers also rotation and scale. Yao and Chen [23] propose a method for the retrieval of color textures, which considers also variations in scale and rotation. Finally, Tsai and Chiang [21] developed a template matching method that considers also rotation, which is based on wavelet decompositions and ring projections. The latter three methods do not provide guarantees regarding the approximation quality of the matching.

Another related work is that of Tian and Narasimhan [20], that estimate the parameters of a dense deformation field. Unlike our method, which works in appearance space, their method minimizes the distance from the target transformation in parameter space.

Image Matching Methods Image matching algorithms are often divided into direct and feature-based methods.

In direct methods, such as Lukas-Kanade [11], a parametric Optic-Flow mapping is sought between two images so as to minimize the Sum-Of-Squared Difference between the images. See the excellent review by Baker *et al.* [2].

Alternatively, one can use feature-based methods such as SIFT [10], or its variant ASIFT [14] which is designed to be affine invariant. In this scenario, interest points are detected independently in each image and elaborate image descriptors are used to represent each such point. Given enough corresponding feature points it is possible to compute the global affine transformation between the images. This approach relies on the assumption that the same interest points can be detected in each image independently and that the image descriptors are invariant to 2D affine transformations so that they can be matched across images.

Other related work Our work is also inspired by techniques from the field of *sublinear* algorithms. The use of sublinear algorithms in image processing was advocated by Rashkodnikova [17] and followed by Tsur and Ron [18] as well as by Kleiner *et al.* [8].

3. The Main Algorithm

3.1. Preliminaries

We are given two grayscale images I_1 and I_2 of dimensions $n_1 \times n_1$ and $n_2 \times n_2$ respectively, with pixel values in the range $[0, 1]$.[1] We will refer to I_1 as the *template* and to I_2 as the *image*. The total variation of an image I, denoted

[1]The algorithm is not restricted to square images but we discuss these for simplicity throughout the article

by $\mathcal{V}(I)$, is the sum over the entire image of the maximal difference between each pixel p and any of its eight neighbors $q \in N(p)$ (we omit the dependence on I as it is always clear from the context). That is,

$$\mathcal{V} = \sum_{p \in I} \max_{q \in N(p)} |I(p) - I(q)| \ .$$

We deal with affine transformations in the plane that have scaling factors in the range $[1/c, c]$ for a fixed positive constant c. Such a transformation T can be seen as multiplying the pixel vector by a 2×2 non-singular matrix and adding a "translation" vector, then rounding down the resulting numbers. Such a transformation can be parameterized by six degrees of freedom.

Let $\Delta_T(I_1, I_2)$ be the (normalized) sum of absolute differences (SAD) distance between two images I_1, I_2 with respect to a transformation T that maps pixels $p \in I_1$ to pixels in I_2. More formally:

$$\Delta_T(I_1, I_2) = \frac{1}{{n_1}^2} \sum_{p \in I_1} |I_1(p) - I_2(T(p))| \ .$$

If p is mapped out of I_2's area then the term $|I_1(p) - I_2(T(p))|$ is taken to be 1. We wish to find a transformation T that comes close to minimizing $\Delta_T(I_1, I_2)$. The minimum over all affine transformations T of $\Delta_T(I_1, I_2)$ is denoted by $\Delta(I_1, I_2)$. A crucial component of our algorithm is the net of transformations. This net is composed of a small set of transformations, such that any affine transformation is "close" to a transformation in the net. Namely, the ℓ_∞ distance between two transformations T and T' quantifies how far the mapping of any point p in I_1 according to T may be from its mapping by T'. Mathematically,

$$\ell_\infty(T, T') = \max_{p \in I_1} \|T(p) - T'(p)\|_2 \ ,$$

where the $\| \cdot \|_2$ is the Euclidean distance in the target image plane. Note that this definition does not depend on the pixel values of the images, but only on the mappings T and T', and on the dimension n_1 of the source image I_1. The key observation is that we can bound the difference between $\Delta_T(I_1, I_2)$ and $\Delta_{T'}(I_1, I_2)$ in terms of $\ell_\infty(T, T')$ as well as the total variation $\mathcal{V}$ of I_1. This will enable us to consider only a limited set of transformations, rather than the complete set of affine transformations. We now describe the construction of such a set.

For a positive α, a net of (affine) transformations $\mathcal{T} = \{T_i\}_{i=1}^{l}$ is an α-cover if for every affine transformation T, there exists some T_j in $\mathcal{T}$, such that $\ell_\infty(T, T_j') = O(\alpha)$. The net we use for our algorithm is a δn_1-cover of the set of affine transformations, where $\delta \in (0, 1]$ is an accuracy parameter which is an input of the algorithm. The number of transformations in the net grows as a function of δ. In [9] we show how to construct such a net $\mathcal{N}_\delta$, of size $\Theta\left(\frac{1}{\delta^6} \cdot \left(\frac{n_2}{n_1}\right)^2\right)$ and prove that it is a δn_1-cover.

3.2. Algorithm Description

We describe a fast randomized algorithm that returns, with high probability, a transformation T such that $\Delta_T(I_1, I_2)$ is close to $\Delta(I_1, I_2)$. The algorithm examines the transformations in the net $\mathcal{N}_\delta$. We give provable guarantees for the quality of approximation. These guarantees are given as a function of the net's parameter δ and of the total variation $\mathcal{V}$ of I_1.

Algorithm 1 ***Approximating the Best Transformation***

Input: Grayscale images I_1, I_2 and a precision parameter δ
Output: A transformation T

- Create a net $\mathcal{N}_{\delta/2}$ that is a $\delta/2$-cover of the set of affine transformations
- For each $T \in \mathcal{N}_{\delta/2}$ approximate $\Delta_T(I_1, I_2)$ to within precision of $\delta/2$. Denote the resulting value d_T
- Return the transformation T with the minimal value d_T

In Step 1 of the algorithm we give a sublinear approximation of $\Delta_T(I_1, I_2)$, that is presented in subsection 3.3.

We proceed to bound the difference between the quality of the algorithm's result and that of the optimal transformation in terms of two parameters - $\mathcal{V}$ and δ, where δ also controls the size of the net and hence determines the running time. We first establish the following theorem which helps to bound the difference between $\Delta_{T'}(I_1, I_2)$ and $\Delta_T(I_1, I_2)$ for a general affine transformation T' and its nearest transformation T on the net.

Theorem 3.1 *Let I_1, I_2 be images with dimensions n_1 and n_2 and let δ be a constant in $(0, 1]$. For a transformation T' let T be the closest transformation to T' in a δn_1-cover. It holds that:* $|\Delta_{T'}(I_1, I_2) - \Delta_T(I_1, I_2)| \leq O\left(\delta \cdot \frac{\mathcal{V}}{n_1}\right)$.

The proof of Theorem 3.1 can be found in [9] . To get an intuition of why it holds, consider the degenerate case of vertical translations. Let T be a translation by k pixels and T' by $k+1$. Now consider the value of $|\Delta_{T'}(I_1, I_2) - \Delta_T(I_1, I_2)|$. Every pixel $p = (x, y)$ in I_1 is mapped by T to the same location that the pixel $p' = (x, y-1)$ is mapped to by T'. Thus the difference between $\Delta_{T'}(I_1, I_2)$ and $\Delta_T(I_1, I_2)$ is bounded by the total sum of differences between vertically neighboring pixels in I_1. The sum of these differences relates linearly to the total variation of I_1. Likewise, when the translations are by k pixels and by $k + \delta n_1$ pixels - the change in the SAD is bounded by the total variation multiplied by δn_1. After normalizing by the size of I_1 we get the bound stated in the theorem.

Thm. 3.1 and the use of a δn_1-cover lead directly to the theoretical bound on Alg. 1's accuracy:

Theorem 3.2 *Algorithm 1 returns a transformation T such that $|\Delta_T(I_1,I_2) - \Delta(I_1,I_2)| \leq O\left(\delta \cdot \frac{\mathcal{V}}{n_1}\right)$ holds with high probability. The total runtime (and number of queries) is $\tilde{O}\left((\frac{n_2}{n_1})^2 \cdot 1/\delta^8\right)$.* [2]

Smooth vs. General Images Many natural images have small total variation (see, e.g., [22]). We measured the total variation of 9500 random templates from the Pascal dataset [3]. Our results indicate that most images in the database sampled have a small total variation ($O(n_1)$ for $n_1 \times n_1$ templates). Some data and a discussion of these measurements can be found in [9].

This smoothness property of natural images, together with Theorem 3.2, implies that Algorithm 1 is guaranteed to provide an additive approximation of $O(\delta)$, for a given precision parameter δ.

3.3. Approximating the Distance $d_T(I_1,I_2)$

We now turn to describe the sublinear algorithm which we use in Step 1 of the algorithm to approximate $\Delta_T(I_1,I_2)$. This dramatically reduces the runtime of Algorithm 1 while having a negligible effect on the accuracy. The idea is to estimate the distance by inspecting only a small fraction of pixels from the images. The number of sampled pixels depends on an accuracy parameter ϵ and not on the image sizes.

Algorithm 2 ***Single Transformation Evaluation***

Input: Grayscale images I_1 and I_2, a precision parameter ϵ and a transformation T

Output: An estimate of the distance $\Delta_T(I_1,I_2)$

- Sample $m = \Theta(1/\epsilon^2)$ values of pixels $p_1 \dots p_m \in I_1$.
- Return $d_T = \sum_{i=1}^{m} |I_1(p_i) - I_2(T(p_i))|/m$.

Claim 3.1 *Given images I_1 and I_2 and an affine transformation T, Algorithm 2 returns a value d_T such that $|d_T - \Delta_T(I_1,I_2)| \leq \epsilon$ with probability $2/3$. It performs $\Theta(1/\epsilon^2)$ samples.*

The claim holds using an additive Chernoff bound. Note that to get the desired approximation with probability $1-\eta$ we perform $\Theta(\log(1/\eta)/\epsilon^2)$ samples.

Photometric Invariance: An adaptation of Algorithm 2 allows us to deal with linear photometric changes (adjusting brightness and contrast). We calculate the optimal change for the points sampled every time we run *Single Transformation Evaluation* by normalizing each sample by its mean and standard-deviation. This adjustment allows us to deal with real life images at the cost of little additional time.

[2] The symbol $\tilde{O}$ hides (low order) logarithmic factors

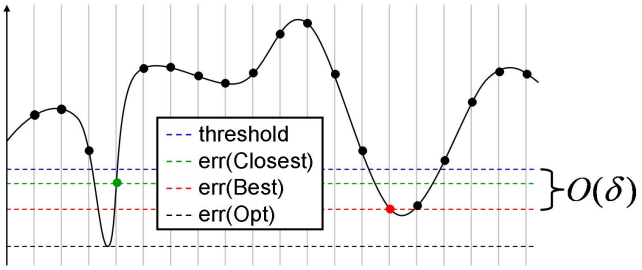


Figure 2. **Branch-and-Bound Analysis.** One stage of the branch-and-bound scheme. For simplicity the space of transformations is in 1D (x-axis) against the SAD-error (y-axis). Vertical gray lines are the sampling intervals of the net. Dots are the samples. Horizontal dotted lines are SAD errors of: Black (Optimal transformation, which is generally off the net), Red (best transformation found on the net), Green (closest-to-Optimal transformation on the net) and Blue (threshold). Only areas below the (blue) threshold are considered in the next stage. The choice of the threshold is explained in the text.

4. The Branch-and-Bound Scheme

To achieve an additive approximation of $O(\delta)$ in Algorithm 1 we must test the complete net of transformations $\mathcal{N}_\delta$, whose size is $\Theta(\frac{1}{\delta^6} \cdot (\frac{n_2}{n_1})^2)$. Achieving a satisfactory error rate would require using a net $\mathcal{N}_\delta$ where δ is small. The rapid growth of the net size with the reduction in the value of δ (linear in $1/\delta^6$) renders our algorithm impractical, despite the fact that our testing of each transformation is extremely efficient. To overcome this difficulty, we devise a branch-and-bound scheme, using nets of increasing resolution while testing small fractions of the transformations in the rapidly growing nets. This improvement is possible with virtually no loss in precision, based on our theoretical results. As a result, the number of transformations we test in order to achieve a certain precision is reduced dramatically.

We describe next the branch-and-bound scheme. The pseudo-code appears below as Algorithm 3 (Fast-Match). In each stage, Algorithm 1 is run on a subset S of the net $\mathcal{N}_\delta$. Figure 2 gives an illustration of transformations examined by the algorithm and their errors (in particular *Opt* - the optimal, *Best* - the best examined, and *Closest* - the closest on the net to *opt*). We denote by $e(Opt)$ the error of *opt* and similarly for *best* and *closest*. We wish to rule out a large portion of the transformation space before proceeding to the next finer resolution net, where the main concern is that the optimal transformation should not be ruled out. Had we known $e(Closest)$, we could have used it as a threshold, ruling out all transformations with error exceeding it. We therefore estimate $e(Closest)$ based on the relations between $e(Opt)$, $e(Best)$ and $e(Closest)$. On one hand, $e(Best) - e(Opt) = O(\delta)$ (following Theorem 3.2) and on the other hand, $e(Closest) - e(Opt) = O(\delta)$ (by the construction of the net and following Theorem 3.1). It follows that $e(Closest) - e(Best) = O(\delta)$ hence $e(Closest) < e(Best) + O(\delta)$. Using a large set of data, we estimated con-

stants α and β, such that $e(Closest) < e(Best) + \alpha \cdot \delta + \beta)$ holds for 97% of the test samples. This learned function $L(\delta) = \alpha \cdot \delta + \beta$ is used in Step 2c of Algorithm 3, for the choice of the points that are not ruled out, for each net resolution. In specific cases where the template occurs in much of the image (e.g. flat blue sky patch), we limit the size of Q_i so that the expanded S_{i+1} will fit into RAM.

Algorithm 3 ***Fast-Match: a Branch-and-Bound Algorithm***

Input: Grayscale images I_1, I_2, a precision parameter δ^*
Output: A transformation T.

1. Let S^0 be the complete set of transformations in the net $\mathcal{N}_{\delta_0}$ (for initial precision δ_0)
2. Let $i = 0$ and repeat while $\delta_i > \delta^*$
   (a) Run algorithm 1 with precision δ_i, but considering only the subset S^i of $\mathcal{N}_{\delta_i}$
   (b) Let T_i^{Best} be the best transformation found in S^i
   (c) Let $Q^i = \{q \in S^i : \Delta_q(I_1, I_2) - \Delta_{T_i^{Best}}(I_1, I_2) < L(\delta_i)\}$
   (d) Improve precision: $\delta_{i+1} = fact \cdot \delta_i$ (by some constant factor $0 < fact < 1$)
   (e) Let $S^{i+1} = \{T \in Net_{\delta_{i+1}} : \exists q \in Q_i \ s.t. \ \ell_\infty(T, q) < \delta_{i+1} \cdot n_1\}$
3. Return the transformation T_i^{Best}

5. Experiments and conclusions

We present the performance of our algorithm in three experiments. In the first experiment each template is extracted from an image and matched back to it. In the second, the template is extracted from one image and matched to another, that is related to it geometrically by a homography. In the third experiment the template is taken from one image of a scene and is mapped to an entirely different image of the same scene.[3]

Evaluating the performance of the algorithm: Fast-Match, when evaluating a transformation, estimates the SAD error. A better measure of the correct mapping is the *overlap error*, which quantifies the overlap between the 'correct' location and mapped location of the template in the image (green and magenta quadrilaterals in Figure 3). We use the overlap error to evaluate performance in the first two experiments (where 'ground truth' location is available). The overlap error is defined (following, e.g., Mikoalyciz et al [12, 13]) to be: 1 minus the ratio between the intersection and the union of the regions.

[3]the source-code is available at: www.eng.tau.ac.il/~simonk/FastMatch

Figure 3. **Example from a Fast-Match Run** Top: The template (shown enlarged for clarity), with the 152 pixels that Fast-Match samples. Bottom: Target image, with origin of the template (ground truth location) in green and a candidate area it is mapped to in magenta.

5.1. Exp. I: Affine Template Matching

In this large scale experiment, we follow the methodology used in the extensive pattern matching performance evaluation of Ouyang *et al.* [15]. We use images from the Pascal VOC 2010 data-set [3], which has been widely used for the evaluation of a variety of computer vision tasks. Each pattern matching instance involves selecting an image at random from the data-set and selecting a random affine transformation, which maps a square template into the image (the mapped square being a random parallelogram). The parallelogram within the image is then warped (by the inverse affine transformation) in order to create the square template. See Figure 3 for an example.

We test the method on different template sizes, where the square template dimensions are between 10% and 90% of the minimum image dimension. For each such size, we create 200 template matching instances, as described above. In Table 1 we report SAD and overlap errors of Fast-Match for the different template sizes. Fast-Match achieves low SAD errors, which are extremely close to those of the ground-truth mapping. The ground-truth errors are at an average of 4 graylevels (not zero), since interpolation was involved in the creation of the template. As can be seen, Fast-Match does well also in terms of overlap error. In the following experiments, we measure success only in terms of overlap error.

Template Dimension	90%	70%	50%	30%	10%
avg. Fast-Match SAD err.	5.5	4.8	4.4	4.3	4.8
avg. ground truth SAD err.	4.1	4.1	4.0	4.4	6.1
avg. Fast-Match overlap err.	3.2%	3.3%	4.2%	5.3%	13.8%

Table 1. **Fast-Match Evaluation: SAD and Overlap errors.** SAD errors are in graylevels (in [0,255]). Low SAD error rates are achieved across different template dimensions (10% to 90%). Fast-Match guarantees finding an area with similar *appearance*, and this similarity translates to a good overlap error, correctly localizing the template in the image. Fast-Match SAD error is comparable to that of the ground truth. See text for details.

Comparison to a feature based approach We examine Fast-Match's performance under 3 types of image degradations: additive white gaussian noise, image blur and JPEG distortion. We show its performance under varying template sizes at different levels of such degradations. We compare

its performance to that of ASIFT [14] - a state of the art method which is a fully affine invariant extension of SIFT, for extracting feature point correspondences between pairs of related images. Since ASIFT (without additional post-processing for transformation recovering) and Fast-Match cannot be directly compared due to their different output types[4], we define for ASIFT a success criterion which is the minimal requirement for further processing: Namely, it is required to return at least 3 correspondences, which are fairly close to being exact - the distance in the target image between the corresponded point and the true corresponding point must be less than 20% of the dimension of the template. The success criterion for Fast-Match is an overlap error of less than 20%. This is an extremely strict criterion, especially for templates mapped to small areas - See a variety of examples, below and above this criterion, in [9]. As is claimed in Mikolajczyk *et al.* [13], an overlap error of 20% is very small since regions with up to 50% overlap error can still be matched successfully using robust descriptors.

We consider 2 different template dimensions which are 50% and 20% of the minimal dimension of the image. For each such size, we repeat the template matching process described above. We consider 6 degradation levels of each type (applied to the target image), as follows: Image blurring with gaussian kernels with STD of {0,1,2,4,7,11} pixels, additive gaussian noise with STDs of {0,5,10,18,28,41} greylevels and finally - JPEG compression with quality parameter (in Matlab) set to {75,40,20,10,5,2}[5].

The comparison of the above success rates of ASIFT and Fast-Match is presented in Figure 4. This experiment validates our claim that unlike feature-based methods (e.g. ASIFT) our method can handle smaller and smaller templates (10% in each image dimension - which translate to 30x30 templates). In addition, Fast-Match is fairly robust with respect to noise and JPEG compression and even more robust to blur in comparison with ASIFT [6]. Table 2 shows the algorithm's average runtimes for several templates sizes, run on a single cpu of an Intel i7 2.7 MHz processor.

5.2. Exp. II: Varying Conditions and Scene Types

In the second experiment we examine the performance of Fast-Match under various imaging conditions and on different scene types. We test our algorithm on a dataset by Mikoalyciz et al [12, 13], originally used to evaluate the performance of interest-point detectors and descriptors. The data set is composed of 8 sequences of images, 6 images each: Blur (2), a combination of rotation and zooming (2), viewpoint change (2), JPEG compression (1) and light conditions (1). In each sequence the degradation increases, e.g., in a blur sequence, from entirely unblurred extremely blurred. Unlike the first experiment, here the template is taken from one image and searched for in a different one, related by a homography (rather than an affinity), increasing the difficulty of the task.

Template Dimension	90%	70%	50%	30%	10%
ASIFT	12.2 s.	9.9 s.	8.1 s.	7.1 s.	NA
Fast-Match	2.5 s.	2.4 s.	2.8 s.	6.4 s.	25.2 s.

Table 2. **Runtimes on different template sizes:** Average runtimes (in seconds) over 100 instances for each template dimension. Fast-Match is much faster in general. As opposed to ASIFT, Fast-Match's runtime increases with the decrease of template dimension. The reason is twofold: 1] The size of our net grows linearly in the image-area/template-area ratio. 2] Smaller templates are more common in the image and hence the Branch-And-Bound enhancement becomes less effective.

Each experiment is conducted as follows: We first choose a random axis-aligned rectangle in the first image, where the edge sizes are random values between 10% and 50% of the respective image dimensions. We then use Fast-Match to map this template to each of the other 5 images in the series. We perform 50 such experiments for which the success rates are given in Table 3. The success criterion is identical to the first experiment (i.e. overlap error $< 20\%$) [7]. The sequences of images (with an example of a single experiment for each) are shown in Figure 5.

Seq. \ Distortion Level	1	2	3	4	5
Zoom + Rotation (Bark)	100%	100%	87.5%	97.5%	87.5%
Blur (Bikes)	100%	100%	100%	100%	100%
Zoom + Rotation (Boat)	100%	100%	75%	87.5%	55%
Viewpoint change (Graffiti)	95%	95%	87.5%	90%	85%
Brightness change (Light)	97.5%	100%	100%	100%	97.5%
Blur (Trees)	100%	100%	100%	97.5%	100%
JPEG compression (UBC)	97.5%	100%	100%	100%	100%
Viewpoint change (Wall)	100%	100%	100%	5%	0%

Table 3. Percent of successful matches (overlap error$<$ 20%) per sequence and degradation level. Several examples appear in Fig. 5.

We achieve high success rates across the dataset, with the exception of the higher degradation levels of the 'Wall' and 'Boat' sequences. Note that, the smaller the template area in the target image, the more demanding the overlap error criterion becomes[8]. This is relevant especially to the zoom sequences. The 'Wall' images are uniform in appearance and this makes it difficult to translate good SAD error to correct localization. The results of Experiment II can not

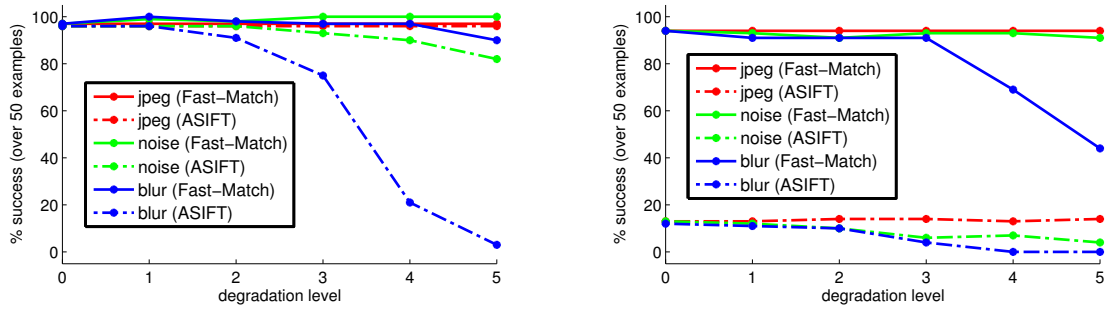
[4]Unlike our method, such feature based methods do not directly produce a geometric mapping. These can be found, based on good quality sets of matching points, using robust methods such as RANSAC [4] by assuming a known geometric model that relates the images (e.g. affine).

[5]Note that in the 3 distortion types, the lowest degradation level is equivalent to no degradation at all

[6]ASIFT is based on SIFT, which has been shown in [12] to be prominent in its resilience to image blur, with respect to other descriptors.

[7]Note that because we are approximating a projective transformation using an affine one (which means matching a general quadrilateral using a parallelogram), the optimal overlap error may be far greater than 0.

[8]This issue has been extensively discussed in [13].



(a) template dimension of 50% (b) template dimension of 20%

Figure 4. **Performance under different template sizes and image degradations.** Analysis is presented for two different template dimensions: (a) 50% and (b) 20% of image dimension. In each, the **x-axis** stands for the increasing levels of image degradation, ranging from 0 (no degradation) to 5 (highest). The **y-axis** stands for the success rates of Fast-Match and ASIFT. Fast-Match is capable of handling smaller and smaller template sizes, while the feature based method ASIFT, deteriorates significantly as template dimension decreases. Like ASIFT, Fast-Match is fairly robust to the different image degradations and is even more robust to high levels of image blur than ASIFT ($\sigma = 4/7/11$ pixels). See text for details.

be compared with those of [12] as they do not deal directly with template or image matching. In this experiment too, Fast-Match deals well with photometric changes as well as the blur and JPEG artifacts.

5.3. Exp. III: Matching in Real-World Scenes

In the third experiment, we present the algorithm's performance in matching regions across different view-points of real-world scenes. We use pairs of images from the Zurich buildings dataset [19]. As done in the second experiment, we choose a random axis-aligned rectangle in the first image, where the edge sizes are random values between 10% and 50% of the respective image dimensions. This dataset is more challenging for the performance of the algorithm, as well as for experimentation: The template typically includes several planes (which do not map to the other image under a rigid transformation), partial occlusions and changes of illumination and of viewpoint.

As there is no rigid transformation between the images, we evaluated the performance of fast match on 200 images visually. On 129 of these we found that the mapping produced by the algorithm was good, in the sense that it corresponded almost exactly to what we judged as the best mapping. In most of the remaining cases producing a good mapping from the given template was impossible: On 40 of the images, the location corresponding to the template was not present in the other image, or that the template spanned several planes which can not be mapped uniquely. In 12 of the images the location that the template was a photograph of was occluded by some outside element, such as a tree. In only 19 of the images was locating the template possible, and the algorithm failed to do so. Examples of good mappings can be found in Figure 6. Examples of cases where a good match was not found appear in Figure 7. The results on the entire dataset appear in [9].

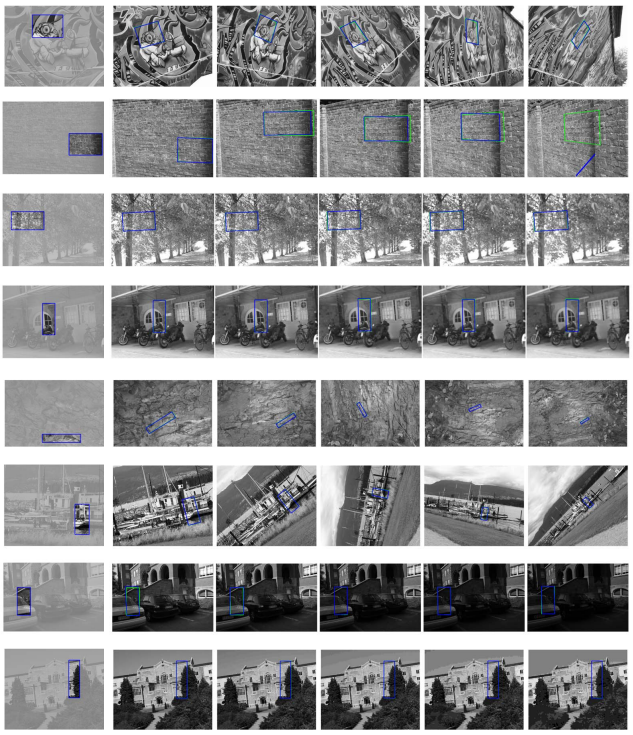


Figure 5. **A typical experiment for each of the Mikolajczyk [13] sequences**. In the leftmost image - the area marked in blue is the input given to Fast-Match. In each of the remaining images a blue parallelogram indicates the mapping produced by Fast-Match, while a green quadrilateral marks the ground truth.

Conclusions We presented a new algorithm, Fast-Match, which extends template matching to handle arbitrary 2D affine transformations. It overcomes some of the shortcomings of current, more general, image matching approaches. We give guarantees regarding the SAD error of the match (appearance related) and these are shown to translate to satisfactory overlap errors (location related). The result is an algorithm which can locate sub-images of varying sizes in other images. We tested Fast-Match on several data sets,

Figure 6. **Zurich Dataset [19] - Good Examples:** In the blue rectangle on the left of each pair of images is the template presented to Fast-Match. In the blue parallelogram on the right is the region matched by the algorithm. Note that also for some of the non-affine mappings Fast-Match gives a good result.

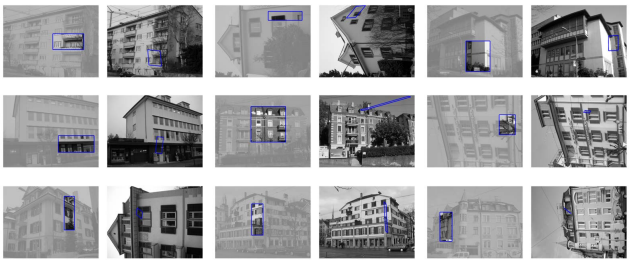


Figure 7. **Zurich Dataset [19] - the remaining:** Failures (row 1), Occlusions (row 2), Template or Target template is out of plane/image (row 3)

demonstrating that it performs well, being robust to different real-world conditions. This suggests that our algorithm can be suitable for practical applications. An interesting direction for future research is to apply similar methods to more diverse families of transformations (e.g. homographies) and in other settings, such as matching of 3D shapes.

Acknowledgements This work was supported by the Israel Science Foundation (grant No. 873/08, in part) and the Ministry of Science and Technology.

References

[1] B. Alexe, V. Petrescu, and V. Ferrari. Exploiting spatial overlap to efficiently compute appearance distances between image windows. NIPS, 2011.

[2] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *IJCV*, 56(3):221–255, 2004.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results.

[4] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[5] K. Fredriksson. *Rotation Invariant Template Matching*. PhD thesis, University of Helsinki, 2001.

[6] C.S. Fuh and P. Maragos. Motion displacement estimation using an affine model for image matching. *Optical Engineering*, 30(7):881–887, 1991.

[7] H. Kim and S. de Araújo. Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast. *AIVT*, pages 100–113, 2007.

[8] I. Kleiner, D. Keren, I. Newman, and O. Ben-Zwi. Applying property testing to an image partitioning problem. *PAMI*, 33(2):256–265, 2011.

[9] S. Korman, D. Reichman, G Tsur, and S Avidan. Fast-Match webpage. www.eng.tau.ac.il/~simonk/FastMatch.

[10] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[11] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981.

[12] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 27(10):1615–1630, 2005.

[13] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L.V. Gool. A comparison of affine region detectors. *IJCV*, 65(1):43–72, 2005.

[14] J.M. Morel and G. Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[15] W. Ouyang, F. Tombari, S. Mattoccia, L. Di Stefano, and W. Cham. Performance evaluation of full search equivalent pattern matching algorithms. *PAMI*, (99):1–1, 2012.

[16] O. Pele and M. Werman. Accelerating pattern matching or how much can you slide? *ACCV*, pages 435–446, 2007.

[17] S. Raskhodnikova. Approximate testing of visual properties. In *RANDOM*, pages 370–381, 2003.

[18] D. Ron and G. Tsur. Testing properties of sparse images. In *FOCS*, pages 468–477. IEEE Computer Society, 2010.

[19] H. Shao, T. Svoboda, and L. Van Gool. Zubudzurich buildings database for image based recognition. *Tech. Report*, 2003.

[20] Y. Tian and S.G. Narasimhan. Globally optimal estimation of nonrigid image distortion. *IJCV*, 98(3):279–302, 2012.

[21] D.M. Tsai and C.H. Chiang. Rotation-invariant pattern matching using wavelet decomposition. *Pattern Recognition Letters*, 23(1):191–201, 2002.

[22] A. van der Schaaf and J.H. van Hateren. Modelling the power spectra of natural images: statistics and information. *Vision Research*, 36(17):2759–2770, 1996.

[23] C.H. Yao and S.Y. Chen. Retrieval of translated, rotated and scaled color textures. *Pattern Recognition*, 36(4):913–929, 2003.