CENTER FOR
MACHINE PERCEPTION

CZECH TECHNICAL
UNIVERSITY

RESEARCH REPORT

ISSN 1213-2365

Matching by Normalized Cross-Correlation—Reimplementation, Comparison to Invariant Features

Tomáš Petříček, Tomáš Svoboda

qqpetricek@cmp.felk.cvut.cz, svoboda@cmp.felk.cvut.cz

CTU–CMP–2010–09

September 9, 2010

The author was supported by EC project FP7-ICT-247870 NIFTi and by the Czech Science Foundation under project P103/10/1585. Any opinions expressed in this paper do not necessarily reflect the views of the European Community. The Community is not liable for any use that may be made of the information contained herein.

Research Reports of CMP, Czech Technical University in Prague, No. 9, 2010

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Matching by Normalized Cross-Correlation—Reimplementation, Comparison to Invariant Features

Tomáš Petříček, Tomáš Svoboda

September 9, 2010

Abstract

The normalized cross-correlation is one of the most popular methods for image matching. While fast implementations of the algorithm are available in standard mathematical toolboxes, there still are ways to get significant speed-up for many practical applications. This work investigates the following possibilities: reusing image sums for matching multiple templates, using maximum expected disparity to bound search regions, and using downscaling factor to reduce size of computation.

Based on our experiments we conclude that both downscaling images and bounding disparity field yields significant speed-up. Downscaling images also yields higher repeatability rate, which remains reasonably high for downscaling factors up to 5. For images related by translation, matching by normalized cross-correlation gives higher repatability rate and matching score than invariant features with SIFT decriptors.

Contents

1 Introduction **2**

2 Possible improvements and function design **2**

2.1 Normalized cross-correlation of multiple templates with single image 2

2.2 Template matching with downsampling factor 4

3 Evaluation and results **6**
3.1 Data sets and image preprocessing 6
3.2 Reusing image sums 6
3.3 Downscaling factor and search regions 9
3.3.1 Blurred images—data set 'bikes' 10
3.3.2 Blurred images—data set 'trees' 10
3.3.3 JPEG compression—data set 'ubc' 12
3.3.4 Light change—data set 'leuven' 13
3.3.5 Repeatability and accuracy discussion 15
3.3.6 Processing time 18
3.4 Comparison to invariant features 19

4 Conclusions **21**

A User guide **23**

1 Introduction

The normalized cross-correlation is one of the most popular mothods for image matching. While fast implementations of the algorithm itself are available in standard mathematical toolboxes, such as Matlab, there still are ways to get significant speed-up for many applications. We investigate some of these possibilities and provide two Matlab functions for practical use.

The Matlab implementation normxcorr2 [1] basically follows the very efficient algorithm [2] in using precomputed image sums for normalizing cross-correlation computed in spatial or transform domain, choosing the method based on image and template size to optimize number of computations. Similar approach is taken in the implementation provided, with some simplification—only 'valid' parts of convolution are computed (i.e. whole templates are matched), and exclusively in frequency domain.

2 Possible improvements and function design

2.1 Normalized cross-correlation of multiple templates with single image

When several templates are matched with a single image, precomputed image sums might be reused for all templates. This is not possible with the Matlab-provided implementation which does not allow multiple templates as argument. Despite that computing image sums usually takes only a fraction

of overall time, it still can count for seconds for large images (i.e. larger than 1000x1000 pixels). Therefore, reusing the precomputed sums is very reasonable, especially if large number of templates is being matched.

Another means to reduce computation time is to match templates only with subimages instead of the whole image. This may be true for some object-tracking applications where some constraints on object speed in image space can be assumed. Template matching can then be limited to a particular subimage which may reduce computation size significantly.

To be able to utilize the possibilities mentioned above, slightly modified version of Matlab function normxcorr2 was designed.

```
corrCoefMaps = normxcorr2ext(templates, image, ranges)
```

The function computes normalized 2-D cross-correlation of the templates and the image, reusing the images sums for all templates. Ranges may be provided to limit the computation to particular subimages for each template. The image sums are used for the normalization, i.e. for calculating the denominator of correlation coefficient. Its nominator is calculated as the 'valid' part of FFT-based convolution[1]

Input:

`templates`—templates as a cell array, should be double and grayscale. Intesity range may be arbitrary, e.g. 0-1 or 0-255.
`image`—image matrix, should be double and grayscale. Intensity range may be arbitrary, e.g. 0-1 or 0-255.
`ranges`—ranges to bound the computation to as a cell array of cell arrays of ranges, e.g. 33:132, 1:100 to limit computation for the first template to subimage containing rows 33 to 132 and columns 1 to 100. Defaults to full range, if not provided. Size of all ranges must be equal or greater than their associated templates.

Output:

`corrCoefMaps`—matrices with correlation coefficients as a cell array.

[1]There is no built-in Matlab function for FFT-based 2-D convolution. We used Luigi Rosa's implementation because it has a convenient interface and there already was a positive feedback from its users. Alternatives might be used too, of course. The function is available from http://www.mathworks.com/matlabcentral/fileexchange/4334.

2.2 Template matching with downsampling factor

As computation size increases quickly with larger image and template sizes, a possibility of reducing the size of images and templates should generally be considered if it is allowed by the particular application. In many applications this can be done without actually changing quality of the results. For example, in an object-tracking application which does not need pixel-precise accuracy, downscaling both the image and templates by a factor of 2 may reduce computation time, by a factor of 16 (i.e. 2^4), approximately, yielding same results in most cases.

For problems where the most likely positions of templates are to be found in a single image, the templatePositions function is provided. It is basically a wrapper around the normxcorr2ext function, utilizing all its advantages, but moreover allowing to use a downscaling factor for the image and templates:

```
[positions rows cols corr] = templatePositions(templates, ...
   image, minCorrCoef, downscalingFactor, initialPositions, ...
   maxDisparity)

[positions rows cols corr] = templatePositions(templates, ...
   image, minCorrCoef, downscalingFactor, searchRegions)
```

The function returns most likely positions of the templates in the image, considering minimum correlation coefficient from which a template is considered matched. The positions returned are scaled to the original image size.

There are two ways of how to bound search regions (i.e. how to call the function): either to provide the search regions directly, or to specify initial positions with maximum expected disparity of the initial positions and those to be found. The parameters and related concepts are visualized in Figure 1.

Input:

`templates`—template matrices as a cell array.
`image`—the image matrix.
`minCorrCoef`—minimum correlation coefficient for a template to be considered found (maximum is still being used to get the most likely positions). Defaults to 0.
`downscalingFactor`—downscaling factor to be used for correlation, must be a positive integer. Scaled images are constructed using the 'lanczos3' interpolation kernel.
`searchRegions`—regions of the image to search templates in, defined as a

cell array of pairs of row vectors, first of which determines the top-left corner and second the bottom right corner of the region. The region defaults to the whole image. Each template must fit into its associated region. Regions themselves are trimmed to fit the image size. Either `searchRegions`, or `initialPositions` with `maxDisparity` may be provided.
`initialPositions`—cell array of initial positions of the templates, e.g. from a previous video frame. Each position is given as a vector of row and column coordinates.
`maxDisparity`—maximum expected disparity in any dimension. It may be either a scalar value common for all templates and both dimensions, a vector with maximum disparity for y and x dimension, or a cell array of one above, with possibly different disparities for individual templates.

Output:

`positions`—cell array of most likely positions of the templates, `[NaN, NaN]` is returned for templates not found using given `minCorrCoef`.
`rows`—cell array of column vectors containing all row coordinates of positions satisfying the minimum correlation coefficient.
`cols`—cell array of column vectors containing all column coords of positions satisfying the minimum correlation coefficient.
`corr`—cell array of column vectors containing coefficients for coordinates given by rows and cols.

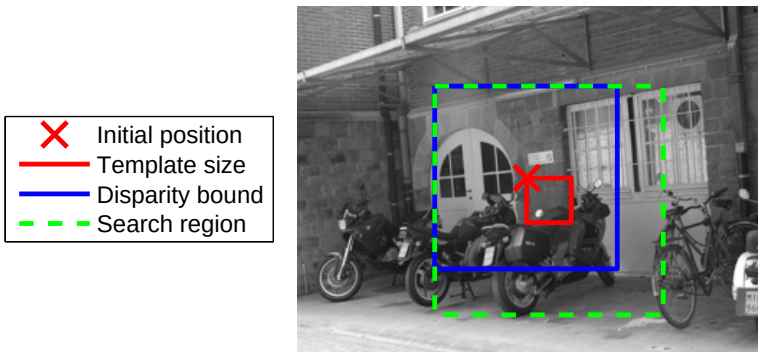

Figure 1: Concepts related to templatePositions function: initial position, template size, disparity bound, and search region.

When a minimum correlation coefficient is supplied, there are three other output parameters available: `rows` and `cols` give coordinates of all template

positions which fulfil the minimum constraint, with `corr` containing actual correlation coefficients.

3 Evaluation and results

We perform three types of tests. In the first test we investigate performance gain resulting from reusing precomputed image sums for matching multiple templates. In the second test we evaluate performance of the *templatePositions* function—we measure repeatability rate and speed-up factors for various combinations of downscaling factor and maximum expected disparity. Last, we compare our results for normalized cross-correlation to repeatability and matching score of invariant features with the SIFT descriptor presented in [3].

For all measurements regarding computation time the build-in Matlab function *cputime* is used.

3.1 Data sets and image preprocessing

For evaluation purposes we use a part of the data sets presented in [3][2]. Our tests are limited to test images with following types of distortion: blur, JPEG compression, changing light conditions. Data sets with viewpoint changes, scaling or rotation are generally not used because cross-correlation cannot generally cope with this class of transforms. To verify that invariant features perform better in such problems, two data sets with viewpoint change are included for comparison in section 3.4.

There is one reference image (img1.ppm) and 5 test images (img2.ppm - img6.ppm) in the original datasets. For testing downscaling factor and search regions in section 3.3, we use only 3 test images, namely img2.ppm, img4.ppm, and img6.ppm. For comparison to invariant features in section 3.4 all images are used. Images were converted to grayscale before further processing. The data sets used are shown in Figure 2.

3.2 Reusing image sums

For this test case any image might be used since our concern is only to measure relative performance of matching multiple templates at once reusing precomputed image sums, compared to matching templates one by one when the sums are computed from scratch for each template. In this test we

[2]Data sets are available from http://www.robots.ox.ac.uk/~vgg/research/affine/

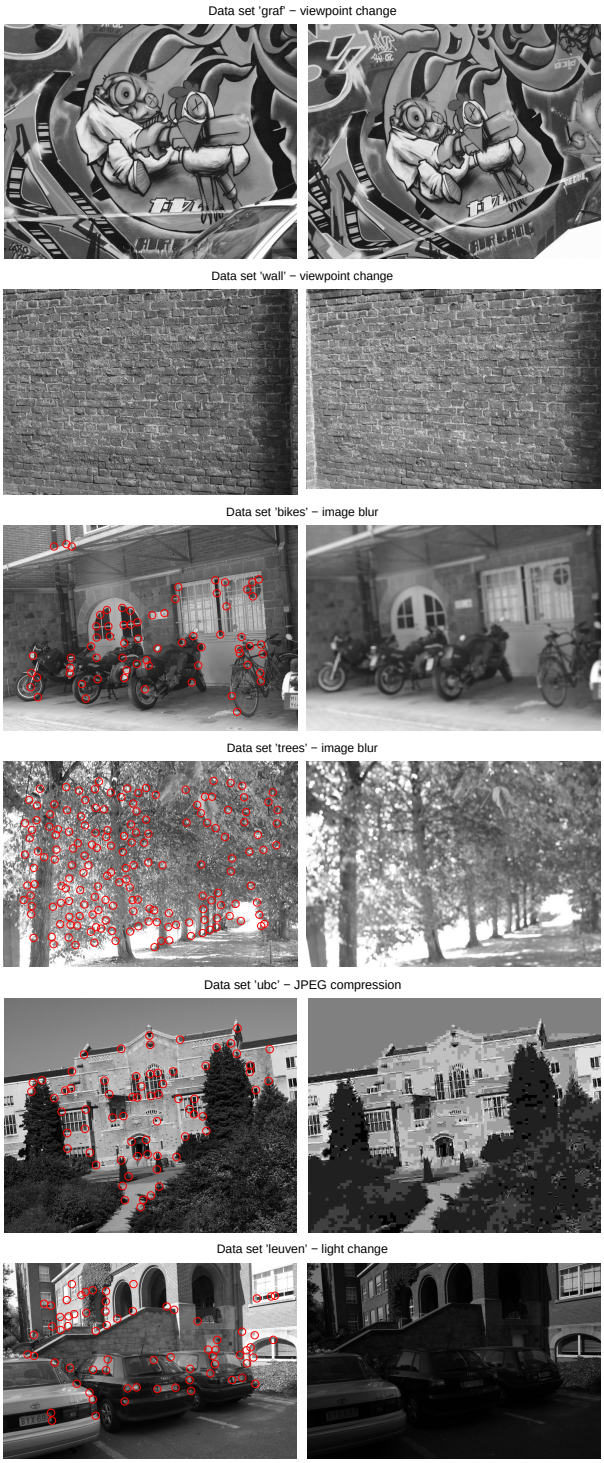

Figure 2: Data sets. The reference image and a test image for each data set. Data sets used in 3.3 are displayed with Harris corners detected.

assume that the algorithm works correctly because the template is taken as a subimage from the image it is matched with.

The first image from the data set 'bikes' in [3] serves both as an image to be searched and as a basis for creating subimages, i.e. templates. Size of the image is 1000x700 pixels. Parameters of the test are template edge size and a number of templates matched at once. Template edge size takes values 16 and 32, the number of templates takes one of following values: 1, 2, 4, 8, 16, and 32. The test is run for each combination of the template edge size and the number of templates.

Results are summarized in Figure 3 and Table 1. The 4th column in the table describes speed-up one can get from reusing image sums for multiple templates instead of recalculating them again and again for every template. The maximum speed gain is above 17% for 16x16 templates and 11% for 32x32 templates. Obviously the relative speed gain is higher for smaller templates as computing image sums takes more time relatively to whole computition and therefore relatively more time can be saved when reusing them. Matching 32 templates, the time saved is around 0.15 s per template in both cases.

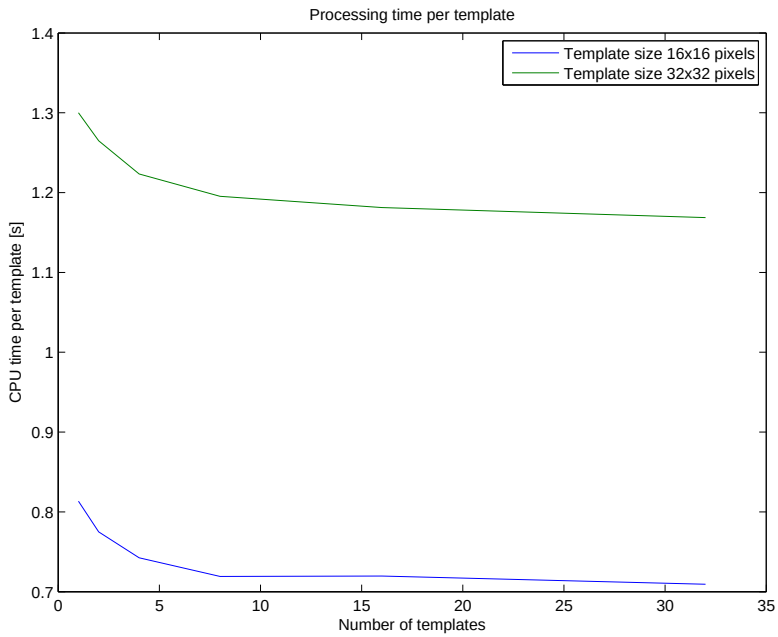

Figure 3: Processing time per template for a varying number of templates reusing image sums(image size 1000x700 pixels)

Table 1: Processing time per template for a varying number of templates reusing image sums (image size 1000x700 pixels)

Template size [px]	Number of templates	CPU time per template [s]	Speed-up for single template
16	1	0.81	0.00%
16	2	0.77	4.71%
16	4	0.74	8.71%
16	8	0.72	11.58%
16	16	0.72	11.53%
16	32	0.71	12.78%
32	1	1.30	0.00%
32	2	1.27	2.69%
32	4	1.22	5.90%
32	8	1.20	8.04%
32	16	1.18	9.13%
32	32	1.17	10.10%

3.3 Downscaling factor and search regions

In this test case the effect of the downscaling factor and search regions on speed and accuracy is investigated. The downscaling factor varies from 1 to 8. Search regions are expressed in terms of maximum expected disparity in any dimension; values 64, 128, and 256 are used in our tests. Accuracy is described in terms of repeatability, and overlap error for matched templates.

For this purposes four data sets from [3] are used: 'bikes' and 'trees' with image blur, 'ubc' with JPEG compression, and 'leuven' with changing light conditions (see Figure 2). Distortion rate increases with every test image in row.

Same tests with various values of maximum expected disparity and downscaling factor are run for every data set:

1. Regions with significant features are found in the reference image using Harris corner detector [4], namely the Matlab function *harris* from [5]. Only Harris corners far enough from image edges are used, to be sure that same regions exist in all test images. The Harris corners found are used as template centers. Template edge size is 32 pixels for all these tests; smaller templates would not be suitable for testing higher values of downscaling factor.

2. For every test image and a combination of input arguments best matches

are found using the *templatePositions* function, bounding search regions with the maximum expected disparity.

3. Positions found are then compared with positions derived from the reference positions using the homography provided for that test image. This comparison basically follows the repeatability measure mentioned in [3] but instead of ellipses rectangular regions are used. Similarly as in the paper mentioned the regions are rescaled prior to computing overlap error and repeatability to have area equal to that of a circle with radius 30. The same overlap error threshold is used too, i.e. 40%.

Using the 40% overlap error threshold is in fact rather strict measure because it says that a template must cover at least $\frac{3}{4}$ area of the region to be considered matched. Only templates which pass the overlap error test count for the average overlap error, therefore this overlap error can never be higher than the threshold used for measuring repeatability.

Feature detection and matching is not separated when normalized cross-correlation is used. Therefore our repeatability results are directly comparable both to repeatability and matching score from [3], with following caveats. First, several templates can have have the same best-match position in a test image. Second, the number of regions detected in the reference image always counts as the basis for matching score calculation because no feature detection is carried on in test images.

3.3.1 Blurred images—data set 'bikes'

A set of 66 templates was matched with images of size 1000x700 pixels.

For this data set we got the best repeatability results, with overall average of 92.5%. As can be seen from Figure 4, for downscaling factors up to 6 the repeatability does not decrease below 90% for any test image.

Regarding accuracy, such results could be expected—the overlap error rises slightly with increasing downscaling factor and image distortion.

3.3.2 Blurred images—data set 'trees'

A set of 165 templates was matched with images of size 1000x700 pixels.

Although these are another blurred images, the results differ significantly from those of 'bikes'. This is mainly caused by the texture-like scene type, where there are many similar indistinct features detected by Harris corner detector. Because of that repeatability decreases with larger search windows as more similar regions can be found in larger areas of the same texture.

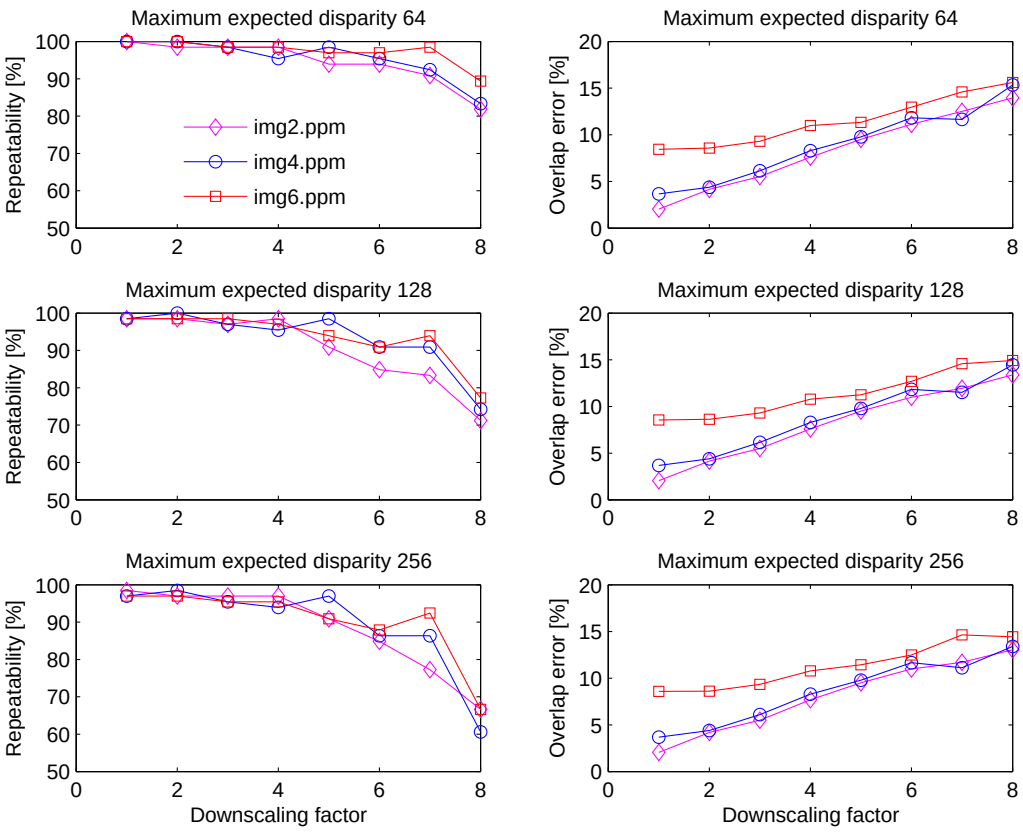

Figure 4: Repeatability and overlap error for data set 'bikes'

This is especially true for heavily blurred test images, i.e. img4.ppm and img6.ppm. See Figure 5 for more details.

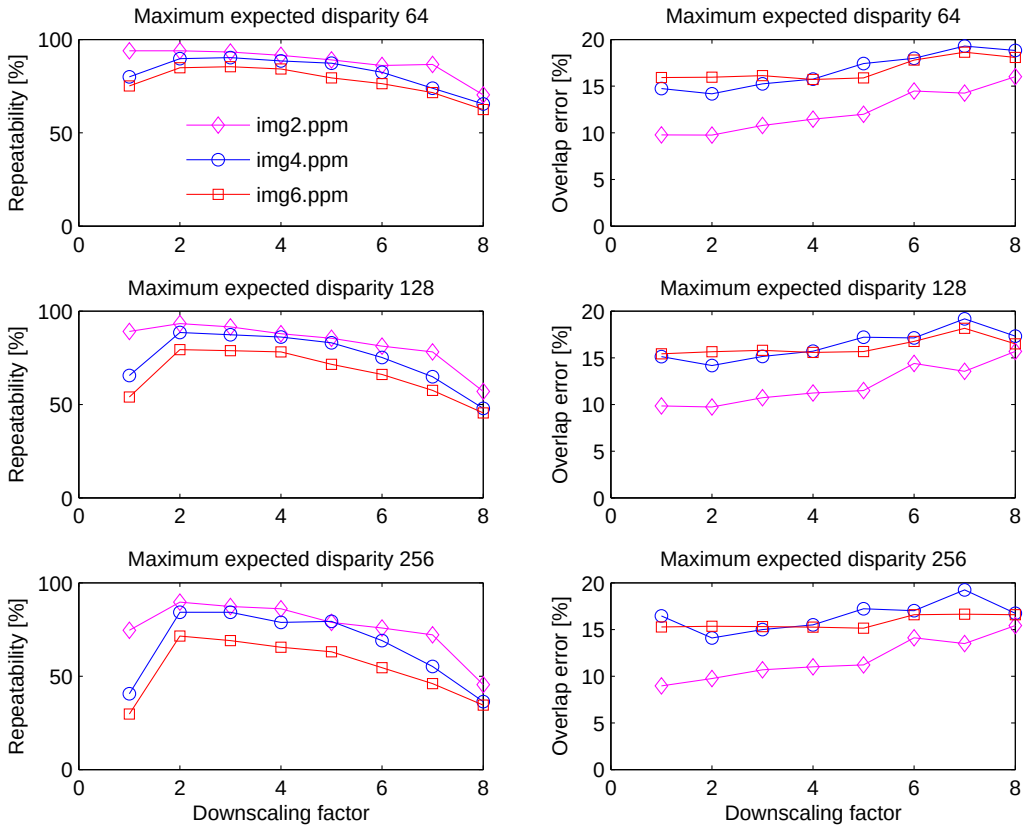

Figure 5: Repeatability and overlap error for data set 'trees'

3.3.3 JPEG compression—data set 'ubc'

A set of 66 templates was matched with images of size 800x640 pixels.

Results for JPEG-compressed images are shown in Figure 6. As can be seen from the plots, JPEG artifacts in heavily-compressed images complicate feature matching a lot, at least when cross-correlation is used. For example, having maximum disparity 256 and downscaling factor of 1, increasing compression from 60% to 90%, and then to 98%, causes the repeatability to drop from 100% to approximately 30%, and to 0%, respectively. For more compressed images repeatability can be improved by using higher downscaling factors—for this particular data set downscaling factor 5 gives even better results than 2.

Similarly as in data set 'trees' the maximum expected disparity seriously affects the repeatability rate. There are probably two main reasons for this: large amount of noise and many indistinctive features in test images, both generated by JPEG artifacts and the rectangular grid they create.

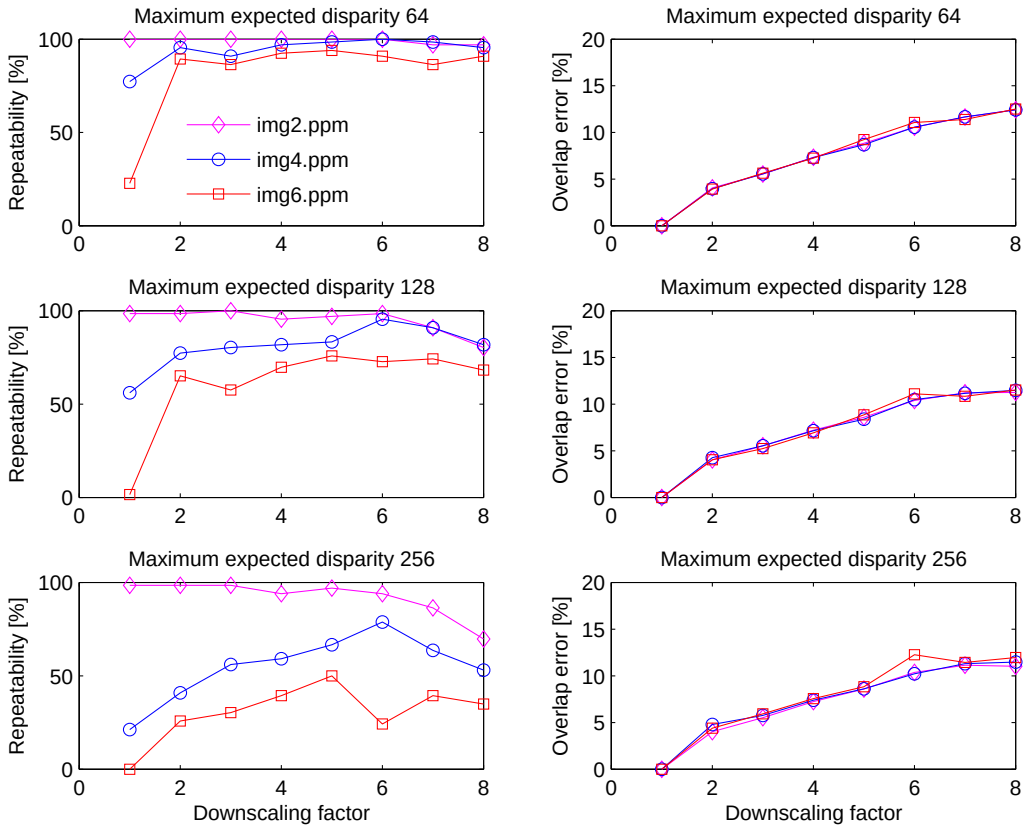

Figure 6: Repeatability and overlap error for data set 'ubc'

3.3.4 Light change—data set 'leuven'

A set of 69 templates was matched with images of size 900x600 pixels.

Average repeatability of 84.7% was the second best result in our tests, even high downscaling factors up to 6 produce reasonable results. Repeatability descreases slightly with higher expected disparity. See Figure 7 for more details.

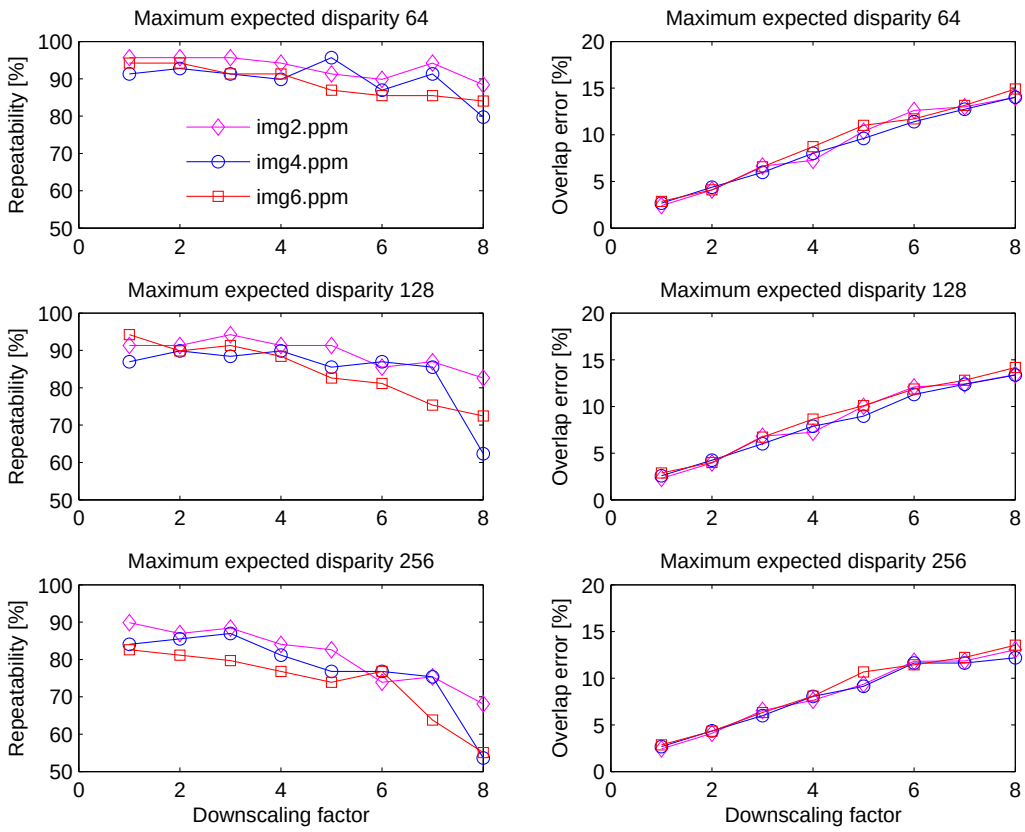

Figure 7: Repeatability and overlap error for data set 'leuven'

3.3.5 Repeatability and accuracy discussion

Accuracy summary for each data set is provided in Table 2. When comparing results for individual data sets it is apparent that both scene type and type of image distortion affects accuracy and repeatability of template matching. Same type of distortion—image blur—yields different results for two different types of scene. For data set 'bikes' we got repeatability average of 92.5%, for data set 'trees' we got 74.3%. Lack of distinctive features in texture-like scenes constitues a problem for other feature detectors and matching scenarios as well. Similar results are presented in [3] where all tested detectors performed worse for data set 'trees' than for 'bikes'.

Table 2: Repeatability and overlap error summary for data sets

Data set	Average repeatability	Average overlap error
bikes	92.5%	9.4%
trees	74.3%	14.9%
ubc	76.5%	7.4%
leuven	84.7%	8.6%

An overall conclusion can be made regarding downscaling factor. In average, configurations with downscaled images had higher repeatability rates than those with original-sized images. From our tests the best downscaling factor seems to be 2 which gives an average repeatability rate 87.8%; downscaling factor 3 gives 87.5%. With templates of size 32x32 pixels, all downscaling factors up to to 5 (86.0%) seem as a reasonable choice. All averages are summarized in Figure 8 and Table 3.

This might change if some filtering of original-sized images were made but that would prevent us from taking an advantage of speed-up related to using downscaled images. Considering related implications to computation size and performance, applications should generally consider using downscaled images for normalized cross-correlation.

For templates considered matched by the repeatability measure, overlap error increases slightly for downscaled images, as can be seen from Figure 9 and Table 4. Besides effects due to loss of information there is another limitation related to the fact that integer positions are found in downscaled images and then upscaled for the original-sized images using an integer downscaling factor. Therefore without any sub-pixel corrections used in matching, the positions found will ideally be pixel-accurate only in $100/{s_d}^2$ per cent of cases, with s_d being the downscaling factor.

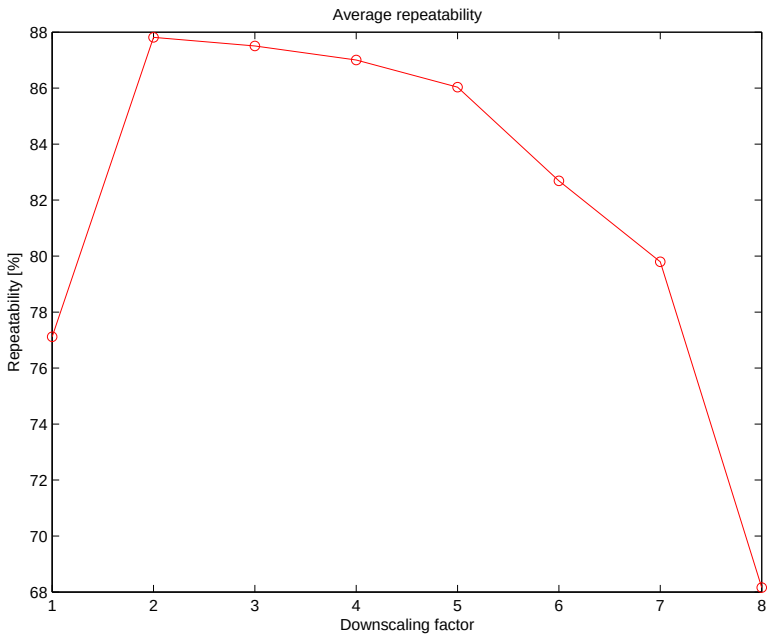

Figure 8: Average repeatability rate vs. downscaling factor

Table 3: Average repeatability rate vs. downscaling factor

Downscaling factor	Average repeatability
1	77.1%
2	87.8%
3	87.5%
4	87.0%
5	86.0%
6	82.7%
7	79.8%
8	68.2%

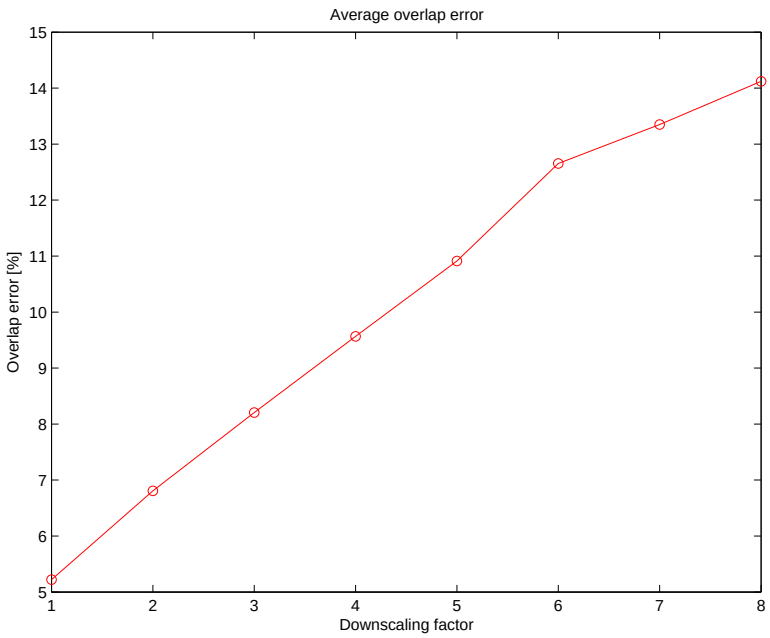

Figure 9: Average overlap error vs. downscaling factor

Table 4: Average overlap error vs. downscaling factor

Downscaling factor	Average overlap error
1	5.2%
2	6.8%
3	8.2%
4	9.6%
5	10.9%
6	12.7%
7	13.3%
8	14.1%

3.3.6 Processing time

Effects of downscaling factor and bounded disparity field on speed should be same regardless of what data set is used, provided that the number of templates is fixed. The results below are averages from all data sets, with the number of templates varying from 66 to 165. Table 5 lists processing time per template, Table 6 lists average speed-up factors for combinations of maximum expected disparity and downscaling factor, compared to the case with maximum disparity 256 and no downscaling (i.e. downscaling factor 1).

Table 5: Average processing time per template [ms] for combinations of maximum expected disparity $d_{\max}$ and downscaling factor s_d

$d_{\max} \backslash s_d$	1	2	3	4	5	6	7	8
256	242.2	45.4	26.5	13.6	10.2	8.0	6.1	5.5
128	81.0	14.5	12.1	5.5	4.5	4.0	4.0	4.1
64	42.1	8.1	4.8	3.9	3.5	3.5	3.4	3.1

Table 6: Average speed-up factor for combinations of maximum expected disparity $d_{\max}$ and downscaling factor s_d, relatively to processing time for maximum disparity 256 and downscaling factor 1

$d_{\max} \backslash s_d$	1	2	3	4	5	6	7	8
256	1.0	5.3	9.2	17.8	23.8	30.3	39.5	44.2
128	3.0	16.7	20.0	44.3	53.5	60.1	60.7	58.6
64	5.8	30.1	51.0	61.4	68.3	69.1	71.7	78.2

As can be seen from the results, both limiting maximum expected disparity and using downscaled images saves a significant amount of time. Given a maximum disparity, the computation time is reduced approximately by factor of 5.4 only by increasing downscaling factor from 1 to 2. For downscaling factor 2, which gives the highest repeatability rate, lowering maximum expected disparity from 256 to 64 gives the speed-up factor of 5.6. For parameter values considered in our tests one can get the maximum speed-up factor of 68.3 while still keeping the repeatability rate reasonably high. Note that the reference time itself is computed with bounded search regions, therefore speed-up factors would be even greater if templates were matched to whole images.

3.4 Comparison to invariant features

In the last test we compare normalized cross-correlation to invariant features with SIFT descriptors presented in [3]. Our goal was to make the test settings as close as possible so that the repeatability rate and matching score can be compared directly.

When comparing our results to [3], however, note that the test settings differ slightly and some of the original data sets were intentionally excluded because of transform class.

The same test was conducted for each of the four data sets:

1. The best-performing affine detector for the data set is used to detect features in the reference image. Only features present both in the reference image and in a particular test image are used. We limit the maximum number of features to 500—if more than 500 features are detected, 500 of them are chosen randomly.

2. Templates are constructed around centers of the features; all templates are of size 32x32 pixels.

3. These templates are then matched with test images by the *templatePositions* function, using maximum expected disparity 425 and downscaling factor 4.

Maximum expected disparity of 425 was used to encompass very high disparities in the two data sets where viewpoint changes, i.e. in data sets 'graf' and 'wall'.

Compared to our previous tests the number of templates is much higher and depends on the particular affine detector used. Number of invariant features varies from 423 for data set 'graf' with the MSER detector up to the maximum, i.e. 500.

The results, summarized in Figure 10, are very similar to those obtained from the previous test for downscaling factor 4 and maximum expected disparity 256.[3].

As assumed, invariant features out-perform normalized cross-correlation in data sets where images are related by general affine transform. This is verified on the two data sets with viewpoint changes. In data set 'graf' normalized cross-correlation gives very poor results with repeatability not exceeding 20 per cent. In data set 'wall' it gives higher repeatability and matching score than invariant features for viewpoint changes up to 30 degrees and at least comparable matching score up to 50 degrees.

[3]See the first plot in the third row in Figures 4, 5, 6, and 7.

There are probably several reasons for why these results differ so much. First, these are two types of scene—'graf' is structured while 'wall' is more texture-like. Second, there are some additional occlusions caused by a car in images of 'graf'. Third, viewpoint change includes rotation for some test images of 'graf'.

In data sets where images are related by translation-based homographies, i.e. excluding 'graf' and 'wall', normalized cross-correlation gives both higher repeatability and matching score. This is true for all data sets except 'ubc' where normalized cross-correlation provides comparable results up to JPEG compression of 80 per cent. However its accuracy drops rapidly when compression exceeds 80 percent and then better results can be obtained with invariant features.

Matching score of normalized cross-correlation is significantly higher for both data sets affected by image blur: 2–3 times higher for data set 'bikes' and 3–15 times higher for data set 'trees'.

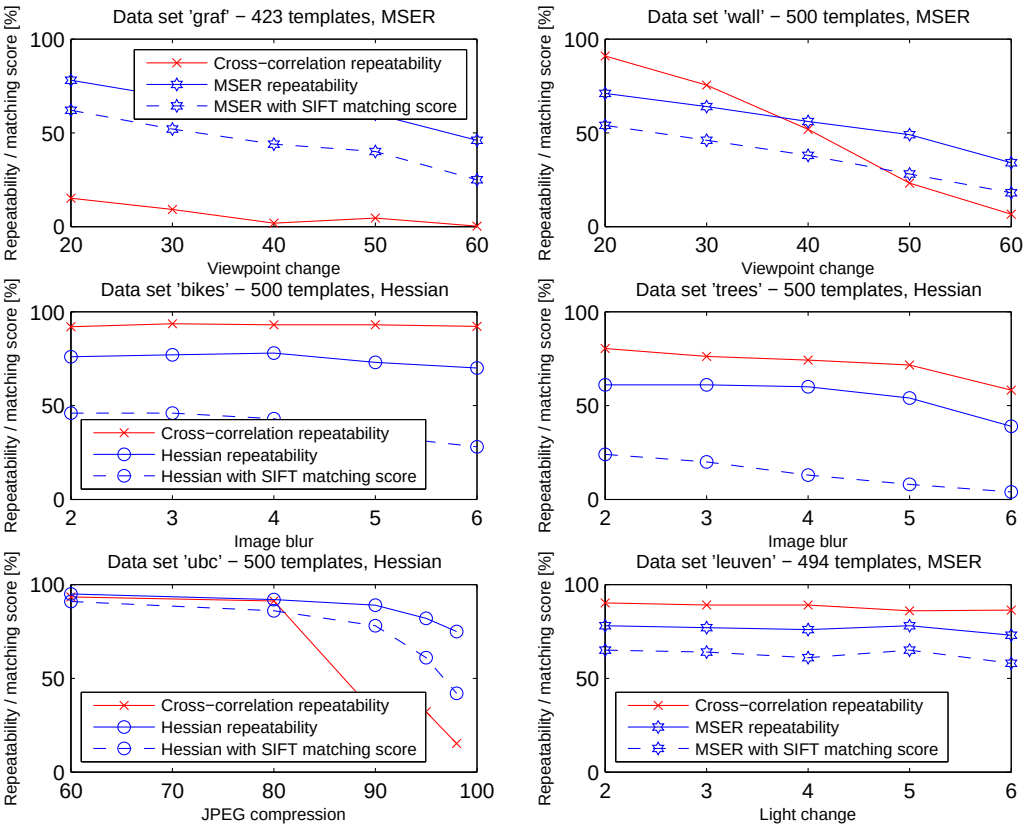

Figure 10: Comparison of normalized cross-correlation and invariant features—repeatability and matching score.

4 Conclusions

In this paper we have described some limitations of Matlab function *normxcorr2* for computing normalized cross-correlation and suggested possible improvements for practical applications: reusing image sums when matching multiple templates, using bounded search regions for individual templates, and downscaling images before computation. To exploit these possibilites two Matlab functions have been designed and implemented—*normxcorr2ext* and *templatePositions*. The latter has an interface convenient for object tracking and wraps the whole functionality presented in the paper.

Three tests have been carried out, using a standardized data set. First, we measured possible speed-up from reusing image sums for matching multiple templates with a single image. Second, we evaluated accuracy and performance of matching in various settings. Last, we compared normalized cross-correlation to invariant features presented in [3].

The accuracy is measured in terms of repeatability and overlap error, as defined in [3]. Our experiments have shown that accuracy is affected both by type of the scene and algorithm parameters. Relatively poor results were obtained for data sets 'trees' and 'ubc'. For data set 'trees' it was due to large texture-like regions in the scene containing many indistinctive features, combined with image blur which further lowers the distinctiveness. In case of 'ubc' poor results were caused mainly by JPEG compression and strong artifacts in heavily-compressed test images.

Better results in terms of repeatability were generally obtained for downscaled images. Downscaling factor of 2 seems to be the best choice providing the highest repeatability rates in most cases along with significant performance gain. It reduces the processing time by factor of 5, compared to the case with original-sized images. Repeatability is reasonably high up to downscaling factor 5. As the results suggest, applications should in general consider to use downscaled images, although the most appropriate value of downscaling factor is application-dependent. The downscaling factor could probably be higher in applications with higher image resolution and less distortion.

Using maximum expected disparity and thus bounding search regions for individual templates also yields considerable speed-up. Lowering maximum disparity from 256 to 64 gives an approximate speed-up factor of 5. General conclusion about the value of maximum expected disparity cannot be made without a particular application in mind—one should choose the lowest value allowed by the application to reduce size of computation as much as possible.

Last, we compared normalized cross-correlation to invariant features with SIFT descriptors. Our results have shown that for images related by trans-

lation normalized cross-correlation gives both higher repeatability rate and matching score than invariant features presented in [3].

The standardized data set provides repeatable evaluation environment but can be seen as too static for some object-tracking applications. Another evalutation in real-world scenario would probably provide deeper insight into how to tune the parameters for real applications.

A User guide

A brief user guide is provided in this appendix in form of a commented Matlab script. The output is shown in Figure 11.

```
%% Initialize search path for third-party functions, e.g. harris, conv2fft.
% We assume that current working directory points to the directory with
% templatePositions.m etc.
initPath

%% Load reference and test images.
% We will match templates from the reference image with the test image. We
% can load two images from the dataset ‘bikes’ used in our tests.
refImPath = '../img/bikes/img1.ppm';
testImPath = '../img/bikes/img4.ppm';
refImage = double(rgb2gray(imread(refImPath)));
testImage = double(rgb2gray(imread(testImPath)));

%% Find some feature points in the reference image.
% These will be used as template centers.
[refCenterRows refCenterCols] = featurePoints(refImPath);

%% Create templates using the feature points.
% Choosing appropriate templates for matching depends on a particular
% application...
TEMPLATE_SIZE = 32;
refCenterPos = vec2cellpos(refCenterRows, refCenterCols);
refTopleftPos = center2topleft(refCenterPos, TEMPLATE_SIZE);
[refTopleftRows refTopleftCols] = cell2vecpos(refTopleftPos);

templates = cell(1, length(refTopleftRows));
for iTpl = 1:length(refTopleftRows)
  templates{iTpl} = refImage(refTopleftRows(iTpl):refTopleftRows(iTpl) ...
                             + TEMPLATE_SIZE - 1, ...
                             refTopleftCols(iTpl):refTopleftCols(iTpl) ...
                             + TEMPLATE_SIZE - 1);
end

%% Find the most likely template positions in the test image.
MIN_CORR_COEF = 0.9; MAX_DISPARITY = 64; DOWNSCALING_FACTOR = 2;
testTopleftPos = templatePositions(templates, testImage, MIN_CORR_COEF, ...
  DOWNSCALING_FACTOR, refTopleftPos, MAX_DISPARITY);
testCenterPos = topleft2center(testTopleftPos, TEMPLATE_SIZE);
[testCenterRows testCenterCols] = cell2vecpos(testCenterPos);

%% Show the correspondences found.
concatImage = [refImage testImage];
demo2Fig = figure; imshow(concatImage, [0 255]); hold on;

% Plot points and matching pairs.
for iPos = 1:length(refCenterRows)
  % Highlight points not matched.
  if isnan(testCenterRows(iPos)) || isnan(testCenterCols(iPos))
    plot(refCenterCols(iPos), refCenterRows(iPos), 'r+', 'MarkerSize', 10);
    continue;
  end
  % Connect points from reference and test images.
  plot([refCenterCols(iPos); testCenterCols(iPos) + size(refImage, 2)], ...
       [refCenterRows(iPos); testCenterRows(iPos)], '--g+');
end
```

Figure 11: Matching results. Templates taken as subimages from the reference image on the left were matched with the test image on the right. Correspondences are shown as dashed green lines. Templates not found using the minimum correlation coefficient are highlighted red.

References

[1] *Matlab documentation*. normxcorr2. Available from (June 9, 2010): http://www.mathworks.com/access/helpdesk/help/toolbox/images/normxcorr2.html.

[2] Lewis, J. P. *Fast Normalized Cross-Correlation*. Industrial Light & Magic. Available from (June 9, 2010): http://www.idiom.com/~zilla/Work/nvisionInterface/nip.html.

[3] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Van Gool, L. *A Comparison of Affine Region Detectors*. International Journal of Computer Vision. Springer Science + Business Media, Inc. Available from (June 9, 2010): http://www.robots.ox.ac.uk/~vgg/research/affine/det_eval_files/vibes_ijcv2004.pdf.

[4] Harris, C. G., Stephens, M. J. *A combined corner and edge detector*. Proceedings Fourth Alvey Vision Conference, Manchester. pp 147-151, 1988.

[5] Svoboda, T., Kybic, J., Hlavac, V. *Image Processing, Analysis, and Machine Vision: A MATLAB Companion*. Thomson Learning 2008. ISBN: 0495295957.