Rotation-Discriminating Template Matching Based on Fourier Coefficients of Radial Projections with Robustness to Scaling and Partial Occlusion

Hae Yong Kim

Escola Politécnica, Universidade de São Paulo

Av. Prof. Luciano Gualberto, tr. 3, 135, São Paulo, SP, 05508-010, Brazil.

Tel: (55-11)3091-5605. Fax: (55-11)3091-5718.

hae@lps.usp.br

http://www.lps.usp.br/~hae

Abstract: We consider brightness/contrast-invariant and rotation-discriminating template matching that searches an image to analyze *A* for a query image *Q*. We propose to use the complex coefficients of the discrete Fourier transform of the radial projections to compute new rotation-invariant local features. These coefficients can be efficiently obtained via FFT. We classify templates in "stable" and "unstable" ones and argue that any local feature-based template matching may fail to find unstable templates. We extract several stable sub-templates of *Q* and find them in *A* by comparing the features. The matchings of the sub-templates are combined using the Hough transform. As the features of *A* are computed only once, the algorithm can find quickly many different sub-templates in *A*, and it is suitable for: finding many query images in *A*; multi-scale searching and partial occlusion-robust template matching.

Keywords: Template-matching; radial projection; rotation-invariant feature; brightness/contrast-invariance; scale invariance; partial occlusion.

1 Introduction

1.1 The Problem

In this paper, we consider the rotation-discriminating and brightness/contrast-invariant template matching problem, where the algorithm must search a grayscale image to analyze *A* for a query image *Q*. A template matching is rotation-invariant if it can find rotated instances of *Q* in *A* and is rotation-discriminating if it determines the rotation angle of *Q* for each matching. We define that two images $\boldsymbol{x}$ and $\boldsymbol{y}$ are equivalent under brightness/contrast variation if there are contrast correction factor $\beta>0$ and brightness correction factor γ such that $\boldsymbol{y} = \beta\boldsymbol{x} + \gamma\boldsymbol{1}$, where $\boldsymbol{1}$ is the matrix of 1's.

In the literature, there are many techniques to solve this problem. The most obvious solution is the "brute-force" algorithm. It makes a series of conventional brightness/contrast-invariant template matchings between *Q* rotated by many different angles and *A*. Conventional brightness/contrast-invariant template matching usually uses the normalized cross-correlation (NCC). Computation of NCC can be accelerated using fast Fourier transform (FFT) and integral images [1] or bounded partial correlation [2]. However, even using fast NCC, the brute-force rotation-invariant algorithm is slow because fast NCC must be applied repeatedly for many angles.

Some techniques solve rotation-invariant template matching using previous segmentation/binarization, for example [3, 4]. Given a grayscale image to analyze *A*, they first convert it into a binary image using some segmentation/thresholding algorithm. Then, they separate each connected component from the background. Once the shape is segmented, they obtain scale-invariance by normalizing the shape's size. Next, they compute some rotation-invariant features for each component. These features are compared with the template's features. The most commonly used rotation-invariant features include Hu's seven moments [5] and Zernike moments [6]. Unfortunately, in many practical cases, images *Q* and *A* cannot be converted into binary images and thus this method cannot be applied.

Ullah and Kaneko use local gradient orientation histogram to obtain rotation-discriminating template matching [7]. Marimon and Ebrahimi present a much faster technique that also uses gradient orientation histogram [8]. The speedup is mainly due to the use of integral histograms. The gradient orientation histograms are not intrinsically rotation-invariant and a "circular shifting" is necessary to find the best matchings.

The recently developed matching algorithms based on scale and rotation-invariant key-points, like SIFT [9] and GLOH [10], present very spectacular computer performance together with true scale-invariance. These algorithms are particularly fit for finding query images with rich textures. However, they can fail to find some simple shapes with little grayscale variations.

Many other techniques use circular projections (also called ring projections) for the rotation-invariant template matching, for example [11, 12, 13]. The underlying idea of these techniques is that features computed over circular or annular regions are intrinsically rotation-invariant. The circular projection can be followed by radial projection to obtain scale and rotation-discriminating template matching [14]. This technique can be drastically accelerated using dedicated hardware, like FPGA (Field Programmable Gate Array) [15]. However, these techniques are slow in conventional computers, because circular and radial projections are time-consuming processes in non-parallel computers.

1.2 The Outline of the Algorithm

Choi and Kim [16] have proposed an interesting approach to accelerate circular projection-based rotation-invariant template matching. Their method computes circular projections for each pixel (x,y) in A (that is, the mean grayscales of the circular rings centered at (x,y)) forming a one-dimensional vector $C(A(x,y))$. The circular projection reduces the 2-D information of the neighborhood of $A(x,y)$ into a one-dimensional rotation-invariant vector. To reduce even more the data, the method computes the first low-frequency complex Fourier coefficients c_1, c_2, ... of $C(A(x,y))$, and uses them as rotation-invariant features. Actually, this technique computes the Fourier coefficients directly, without explicitly computing the circular projections, by

convolving A with appropriate kernels via FFT. The features of Q are compared with the features of each pixel $A(x,y)$ to select the pixels candidate for the matching. A secondary filter based on the rotation-invariant Zernike moments is used to further test the candidate pixels. Rotation-dependent features cannot be used in this test because circular projections do not discriminate the rotation angle.

We propose to improve Choi and Kim's algorithm by using new rotation-invariant and rotation-discriminating features derived from radial projection, together with the circular features. In order to provide a short name to the new algorithm, we will call it "Forapro" template matching (Fourier coefficients of Radial Projections). We compute, for each pixel (x,y) in A, the mean grayscales of the radial lines centered at (x,y), forming a one-dimensional vector of radial projections $R(A(x,y))$. Then, we compute the first low-frequency complex inverse Fourier coefficients r_1, r_2, ... of $R(A(x,y))$. Actually, we do not compute the radial projections, but the Fourier coefficients directly. Convolutions in the frequency domain are used to compute quickly the Fourier coefficients, employing appropriate kernels and FFT. Using special instruction sets, like MMX (multi-media extensions) or SSE (streaming SIMD extensions), available in most of the nowadays processors, FFT can be computed 5-20 times faster than good conventional software implementations. Differently from the circular case, the radial coefficients are not intrinsically rotation-invariant. However, it is possible to derive many rotation-invariant features and one rotation-discriminating feature from the radial coefficients.

We show experimentally that the rotation-invariant radial features are more adequate for finding templates than circular ones. However, the maximal accuracy is obtained by using both the radial and circular features. We classify query images in "stable" and "unstable" ones and show that any local feature-based template matching can fail when searching for unstable query images. Thus, we extract one or more stable circular sub-templates T_1, ..., $T_N \subset Q$, find them in A, and test for false positive errors using Hough transform or NCC. This secondary test is essential, because any feature-based template matching reduces the original 2-D information into a set of features, and consequently many non-equivalent templates may be mapped into the same features, producing false positive errors.

Template matchings based in pre-computed rotation- and brightness/contrast-invariant features are advantageous principally when the algorithm must search an image A for a large number of templates. In this case, the vector of rotation-invariant features $v_f(A(x,y))$ is computed only once for each pixel $A(x,y)$. Then, each template T_i can be found quickly in A by computing the vector of features $v_f(T_i(x_o,y_o))$ at the central pixel (x_o,y_o) of T_i and comparing it with $v_f(A(x,y))$. If the distance is below some threshold, then the neighborhood of $A(x,y)$ is "similar" (in rotation- and brightness/contrast-invariant sense) to the template image T_i and (x,y) is considered a candidate for the matching. This property makes our algorithm suitable for: finding many different query images in A; multi-scale searching and partial occlusion-robust template matching.

Our compiled programs and some test images are available at www.lps.usp.br/~hae/software/forapro. Note that these programs are intended only for testing the ideas developed in this paper, and thus many parameters were purposely left to be set by hand.

The remainder of the paper is organized as follows: section 2 presents the new features and the concept "stability"; section 3 presents the new template matching algorithms; section 4 presents experimental results; and section 5 presents our conclusions.

2 New Features

2.1 Radial IDFT Coefficients

Given a grayscale image A, let us define the radial projection $R_\alpha^\lambda(A(x,y))$ as the average grayscale of the pixels of A located on the radial line with one vertex at (x,y), length λ and inclination α:

$$R_\alpha^\lambda(A(x,y)) = \frac{1}{\lambda}\int_0^\lambda A\big(x + t\cos(\alpha), y + t\sin(\alpha)\big)dt\,. \qquad (1)$$

In practice, a sum must replace the integral, because digital images are spatially discrete. The vector of M discrete radial projections at pixel $A(x,y)$ with radius λ can be obtained by varying the angle α:

$$R^{\lambda}_{A(x,y)}[m] = R^{\lambda}_{2\pi m/M}(A(x,y)), \quad 0 \le m < M \ . \tag{2}$$

Figures 1(b) and 2(a) depict M=36 radial projections at the central pixel of figure 1(a).

Vector of radial projections $R^{\lambda}_{A(x,y)}[m]$ characterizes the neighborhood of $A(x,y)$ of radius λ. If A rotates, then this vector shifts circularly. This property is illustrated in figure 2, the vector of radial projections of figure 1(a). The k-th Fourier coefficient of a vector of radial projections R is (we omit indices $A(x,y)$ and λ):

$$r[k] = \sum_{m=0}^{M-1} R[m] \exp(-j2\pi km/M), \quad 0 \le k < M \ . \tag{3}$$

The Fourier coefficients of a vector of radial projections can be computed directly convolving A with an appropriate kernel K, without explicitly calculating the radial projections. Figure 3(a) depicts the "sparse DFT kernel" K (with M=8 angles) such that the convolution $A * \breve{K}$ yields the first Fourier coefficient of the radial projections (where $\breve{K}(x,y) = K(-x,-y)$ is the double reflection of K):

$$(A * \breve{K})(x,y) = \sum_p \sum_q A(p,q)\breve{K}(x-p, y-q) = \sum_p \sum_q A(p,q)K(p-x, q-y) \ . \tag{4}$$

It is well known that the convolution $A * \breve{K}$ can be computed by multiplications in the frequency domain:

$$A * \breve{K} \Leftrightarrow \boldsymbol{A}\breve{\boldsymbol{K}} \ , \tag{5}$$

where $\boldsymbol{A}$ and $\breve{\boldsymbol{K}}$ are respectively the discrete Fourier transforms of A and $\breve{K}$.

The sparse kernel in figure 3(a) does not take into account most of the outer pixels and consequently it does not yield accurate features. To overcome this problem, the "dense DFT kernel" depicted in figure 3(b) can be used instead. It fills all empty kernel elements, except the very central pixel. The non-zero elements of this kernel are defined as:

$$\Re_k[x,y] = \exp\big(-jk\angle(x+yj)\big), \tag{6}$$

where k is the order of the Fourier coefficient and $\angle(.)$ is the angle of the complex number.

The linear filter using this kernel has no intuitive meaning. Using the inverse DFT, the result of the convolution acquires a meaning: it becomes analogous to the gradient. Figure 3(c) depicts the kernel obtained using IDFT and k=1. In order to make the kernel more "stable," that is, to make the result of the convolution less sensitive to small perturbations like sub-pixel translation or rotation, we assign less weight to the pixels in the outer and central regions, resulting the weighted kernel depicted in figure 3(d). We tested empirically a number of weight distributions and chose the most stable one. We explain in subsection 2.2.3 how these tests were carried out. The resulting radial kernel is:

$$\Re_k[x, y] = \sqrt{r(\lambda - r)}\, \exp\big(jk \angle(x + yj)\big), \tag{7}$$

where $r = \sqrt{x^2 + y^2}$ and λ is the radius of the kernel. The weights of the five central pixels are zeroes. In order to make the weights sum to one, all weights may be divided by the sum of the weights. The kernels used to obtain the inverse Fourier coefficients with k=2 and k=3 are depicted respectively in figures 3(e) and 3(f).

We will call the convolution of $A(x,y)$ with the double reflection of the k-th radial kernel "k-th radial coefficient" and denote it $r_k(A(x, y))$ or simply r_k. We will call $\angle r_k$ and $|r_k|$ respectively "k-th radial angle" and "k-th radial magnitude."

2.2 Rotation-Discriminating "Canonical Orientation"

2.2.1 The First Radial IDFT Coefficient

The rotation-discriminating feature, or the "canonical orientation," is the first radial angle $\angle r_1$. The canonical orientation $\angle r_1(A(x, y))$ indicates the local direction of $A(x,y)$ within a neighborhood of radius λ. If A rotates θ radians, then the vector of radial projections $R^{\lambda}_{A(x,y)}[m]$ shifts circularly θ radians and, by the time shift property of IDFT, $r_1(A(x, y))$ is multiplied by $\exp(j\theta)$. In other words, if A rotates, the first radial angle $\angle r_1(A(x, y))$ rotates by the same angle. Moreover, a brightness/contrast change does not alter the canonical orientation.

2.2.2 Stability Analysis

The canonical orientation cannot be computed if the first radial magnitude $|r_1|$ is too small, like inside regions with constant grayscale or at the very center of a symmetrical shape such as 'I', 'H' and 'O'. However, in the latter case, the orientation can be computed shifting slightly the center of the shape. This idea can be generalized using the concept "stability." Figure 4(a) depict the coefficients r_1 (using the kernel with λ=7 pixels) as arrows. Black circles exemplify "unstable" regions, where the orientations change abruptly. On the other hand, the orientations inside white circles are "stable," that is, they do not change much in the neighborhood. In unstable regions, the canonical orientation can change completely if the image is translated or rotated even by a sub-pixel distance. Consequently, the orientations in these regions are not reliable and cannot be used. However, this is not a practical problem because it is possible to extracts stable circular sub-templates $T_i \subset Q$, and use T_i to find Q in A (figure 4(b)). Seemingly, any local feature-based template matching can fail when searching for unstable templates.

We define that the canonical orientation at a pixel $A(x,y)$ is (t_a, t_m)-stable, that is, stable with angle threshold t_a and magnitude threshold t_m, if the following conditions are satisfied:

$$\begin{cases} \underset{A(x',y')\in N(A(x,y))}{\text{MAX}} \left(\Phi\left(\angle r_1(A(x,y)), \angle r_1(A(x',y'))\right)\right) < t_a \\ \left|r_1(A(x,y))\right| \geq t_m \end{cases} \tag{8}$$

where $N(A(x,y))$ is the set of neighbor pixels of $A(x,y)$ (we use 8 neighborhood) and Φ is the difference between two angles, defined as:

$$\Phi(a,b) = \min\left(\text{mod}(a-b, 2\pi), 2\pi - \text{mod}(a-b, 2\pi)\right), \tag{9}$$

where mod(x,y) calculates x modulo y, that is, the remainder f where $x = (ay + f)$ for some integer a and $0 \leq f < y$. The difference Φ between two angles is limited to the interval [0,π[.

To determine the reliability of canonical orientation, we did the following experiment. We took some images, computed the first radial coefficients using λ=15 pixels, and chose (t_a=10^o, t_m=0.01)-stable pixels. We use gray level ranging from 0 to 1. So, t_m=0.01 would become t_m=2.55 if the grayscale ranged from 0 to 255. Roughly the

half of the pixels inside the circular template was stable. Then, we distorted the images and compared the orientations of the stable pixels with the orientations of the distorted images. The distortions were:

- Contrast increasing by 35%. This operation did not saturate our images, because they were originally low-contrast ones.
- Rotation of the images by 30 degrees using bilinear interpolation. We computed the canonical orientations of the rotated images obtaining the complex orientation images, rotated the orientation images back -30^o and multiplied all complex pixels by $\exp(-j \times 30^o)$. If the canonical orientation is reliable, the resulting orientations will be similar to the original orientations.
- Translation of the images by (0.5, 0.5) pixels. We rescaled the images by factor 4 using bilinear interpolation, shifted 2 rows and 2 columns, and rescaled them back by factor 0.25.
- Addition of Gaussian noise with zero mean and standard deviation 0.005.

Table 1 (columns 2 and 3) shows the average and maximum differences between the orientations in the original and distorted images. The average differences are low. But, most importantly, the largest observed difference was 9.49^o, that is, slightly less than $t_a=10^o$. We repeated the experiments using $\lambda=30$ pixels, $t_a=5^o$ and $t_m=0.01$ (table 1, columns 4 and 5). Again, roughly the half of the pixels inside the circulate template were stable and the largest change of orientation was 3.37^o, less than $t_a=5^o$. We conclude experimentally that the canonical orientations of stable pixels with angle threshold t_a likely changes less than t_a when brightness/contrast, rotation, sub-pixel shifting or minor noise contamination distorts the image. This is not a mathematical theorem, but as the chosen threshold limited the largest difference, probably this inequality holds in almost all practical situations. This is not a surprising property, because a stable pixel (x,y) is surrounded only by pixels whose orientations differ less than t_a from the orientation of (x,y). We repeated the experiments including unstable pixels, that is, setting $t_a = \infty$ (table 1, columns 6 and 7). Although the average errors remained low, the maximum angle difference was 178^o, almost the maximal difference between two angles. This demonstrates that the canonical orientation is not reliable in unstable regions.

2.2.3 Weights of the Kernel

In subsection 2.1, we did not explain how we chose the weights of the radial kernel. The rationale of assigning low weights to the pixels in the central and outer regions is to avoid that a small translation or rotation may cause a large change in the radial coefficients. We did many experiments similar to those that tested the reliability of the canonical orientations using various weight distributions. We chose the most reliable one, that is, the weight distribution that minimizes the average and the maximal orientation errors.

2.3 Rotation-Invariant "Vector of Radial Magnitudes"

Radial magnitudes are invariant to rotation because if A rotates, then the vector of radial projections $R^{\lambda}_{A(x,y)}[m]$ shifts circularly, and a circular shifting does not change the magnitudes of the IDFT coefficients (it only changes their angles). Radial magnitudes $|r_k|$, $k\geq 1$, are also invariant to brightness because a brightness alteration only affects the DC coefficient r_0. Finally, the ratios between radial magnitudes are invariant to contrast (besides being rotation- and brightness-invariant), because a contrast alteration multiplies by the same factor all the radial coefficients. For example, let r_1 and r_k be the first and the k-th radial coefficients ($k\geq 2$). Then, the ratio of their magnitudes $|r_k|/|r_1|$ is invariant to rotation and brightness/contrast. In our implementation, instead of ratio, we use the following vector of radial magnitudes v_{rm} that takes into account the magnitudes of all radial coefficients up to degree K:

$$v_{rm} = \upsilon[\,|r_1|,|r_2|,\ldots,|r_K|\,], \tag{10}$$

where υ means L^1-versor and consists on dividing each element of the vector by its L^1-length $|r_1|+|r_2|+\ldots+|r_K|$. We use L^1-distance instead of Euclidean L^2-distance because the computation of L^1-norm is faster than L^2-norm and we noted no difference in accuracy. As we use versor, only K-1 elements of v_{rm} are "independent variables" because, given K-1 elements, it is possible to calculate the remaining element. We define the distance function Λ between two v_{rm}'s as:

$$\Lambda\big(v_{rm}(A(x,y)), v_{rm}(T(x_o,y_o))\big) = \frac{1}{2}\big\|v_{rm}(A(x,y)) - v_{rm}(T(x_o,y_o))\big\|_1 . \tag{11}$$

This distance is limited to interval [0,1].

The concept "stability", developed in subsection 2.2.2, may be also applied here. We define that $v_{rm}(A(x,y))$ is (t_d, t_m)-stable (that is, with distance thresholds t_d and magnitude threshold t_m) if:

$$\begin{cases} \underset{A(x',y')\in N(A(x,y))}{\text{MAX}} \Lambda\left(v_{rm}(A(x,y)), v_{rm}(A(x',y'))\right) < t_d \\ |r_1| + |r_2| + \ldots + |r_K| \geq t_m \end{cases} \tag{12}$$

where $N(A(x,y))$ is the set of neighbor pixels of $A(x,y)$ and r_1, ..., r_K are the radial coefficients at pixel $A(x,y)$.

To test the reliability of v_{rm}, we took some images, computed the first four magnitudes of the radial coefficients using λ=30 pixels, and chose stable pixels with t_d = 0.05 and t_m = 0.005. Then, contrast change, rotation, sub-pixel shifting and Gaussian noise distorted the images. In any stable pixel, the distance between the vectors of distorted and original images was less than 0.05. Then, we repeated the experiments in unstable pixels. The maximal observed distance between the original and distorted vectors at unstable pixels was slightly less than 1, indicating that v_{rm} is not a reliable feature in unstable regions.

2.4 Rotation-Invariant "Vector of Radial Angles"

If A rotates θ radians, then the vector of radial projections $R^{\lambda}_{A(x,y)}[m]$ shifts circularly θ radians and, by the time shift property of IDFT, the k-th radial coefficient $r_k(A(x,y))$ is multiplied by $\exp(jk\theta)$. Moreover, a brightness or contrast change does not alter $\angle r_k$. Thus, the difference between $\angle r_k$ and $k\angle r_1$ is rotation- and brightness/contrast-invariant. We call this feature "difference of radial angles" k and 1:

$$\text{dra}_k = \text{mod}(\angle r_k - k\angle r_1, 2\pi), \quad k \geq 2\,. \tag{13}$$

This feature is computed modulo 2π, because it is an angle. As before, we did the "stability analysis" and concluded that this feature also is not reliable in unstable regions. In our implementation, we packed the dra's up to order K in a structure that we named vector of radial angles v_{ra}:

$$v_{ra} = [\text{dra}_2, \text{dra}_3, \ldots, \text{dra}_K]. \tag{14}$$

We define the distance Λ between two v_{ra}'s as the weighted average of the angle differences:

$$\begin{aligned}\Lambda\left[v_{ra}(A(x,y)), v_{ra}(T(x_o,y_o))\right] = \\ (w_2/(\pi w_t))\,\Phi\left[\mathrm{dra}_2(A(x,y)), \mathrm{dra}_2(T(x_o,y_o))\right] + \\ (w_3/(\pi w_t))\,\Phi\left[\mathrm{dra}_3(A(x,y)), \mathrm{dra}_3(T(x_o,y_o))\right] + \\ \cdots + \\ (w_K/(\pi w_t))\,\Phi\left[\mathrm{dra}_K(A(x,y)), \mathrm{dra}_K(T(x_o,y_o))\right]\end{aligned} \tag{15}$$

where $w_k = 1/k$ $(2 \le k \le K)$, $w_t = w_2 + w_3 + \ldots + w_K$ and Φ is the difference of angles defined in subsection 2.2.2. This distance function is limited to interval [0,1]. We assign smaller weights to high order dra's, because they are more easily affected by image distortions. Let us consider an image distortion that rotates $\angle r_1$ by a small angle of -θ radians, while keeping constant $\angle r_k$, for all $k \ge 2$. This distortion rotates dra_2 by 2θ radians, dra_3 by 3θ radians, etc. That is, the same distortion causes a change proportional to the order k in dra's. So, we assign weights $1/k$ to equalize the contributions of different dra's.

2.5 Rotation-Invariant "Vector of Circular Features"

Choi and Kim have used the DFT coefficients of the circular projections as the rotation-invariant features. We will continue using these features, together with the newly developed radial features. However, we introduce two small alterations in the circular kernels. First, to be consistent with the radial features, we use IDFT instead of DFT. Second, instead of using the original truncated integer radius $r = (\mathrm{int})\left\lfloor\sqrt{x^2+y^2}\right\rfloor$, we use the floating-point radius $r = \sqrt{x^2+y^2}$ because experiments indicate that the latter (figure 5(b)) is more stable than the former (figure 5(a)). The resulting circular kernel is:

$$\Im_l[x,y] = \begin{cases} \dfrac{1}{2\pi r}\exp\left(\dfrac{jlr}{\lambda}\right), & \text{if } r > 0 \\ 0.73, & \text{if } r = 0 \end{cases} \tag{16}$$

where $r = \sqrt{x^2+y^2}$ and λ is the radius of the kernel. The weight $1/2\pi r$ is the inverse of the perimeter of the circle where the pixel lays. The weight for $r = 0$ was set at

0.73 to distribute evenly the angles of the complex image resulting from the convolution. In order to make the weights sum to one, all weights must be divided by the sum of the weights. Figure 5(c) depicts the kernel used to obtain the second circular coefficients.

We will call the convolution of $A(x,y)$ with the double reflection of the l-th circular kernel "l-th circular coefficient" and denote it $c_l(A(x,y))$ or simply c_l. In our implementation, we use the following "vector of circular features" that takes into account the real and imaginary components of all circular coefficients up to degree L:

$$\text{vcf} = \upsilon\left[\text{re}(c_1), \text{im}(c_1), \text{re}(c_2), \text{im}(c_2), \ldots, \text{re}(c_L), \text{im}(c_L)\right], \tag{17}$$

where υ means L^1-versor, and "re" and "im" are respectively the real and imaginary parts of the complex number. Only $2L$-1 elements of this vector are "independent variables" because, given $2L$-1 elements of this vector, it is possible to calculate the remaining element. The distance Λ between two v_{cf}'s is based on L^1-distance:

$$\Lambda\left(v_{cf}(A(x,y)), v_{cf}(T(x_o,y_o))\right) = \frac{1}{2}\left\| v_{cf}(A(x,y)) - v_{cf}(T(x_o,y_o)) \right\|_1 . \tag{18}$$

This distance is limited to interval [0,1]. As before, we did the "stability analysis" and concluded that this feature also is not reliable in unstable regions.

2.6 Combining the Rotation-Invariant Features

In previous subsections, we obtained three rotation-invariant classes of features, using up to K radial and L circular coefficients and packed them in three vectors: v_{rm}, v_{ra}, and v_{cf}. We will group these three vectors into another structure named "vector of features":

$$v_f = \left(v_{rm}, v_{ra}, v_{cf}\right). \tag{19}$$

We define the distance function Λ between two vectors of features as the weighted average of the distances of the three constituent vectors of features:

$$\begin{aligned}\Lambda\left(v_f(A(x,y)), v_f(T(x_o,y_o))\right) = {} & (w_m / w_t)\,\Lambda\left(v_{rm}(A(x,y)), v_{rm}(T(x_o,y_o))\right) \\ & + (w_a / w_t)\,\Lambda\left(v_{ra}(A(x,y)), v_{ra}(T(x_o,y_o))\right) \\ & + (w_c / w_t)\,\Lambda\left(v_{cf}(A(x,y)), v_{cf}(T(x_o,y_o))\right)\end{aligned} \tag{20}$$

where $w_m = w_a = K$-1, $w_c = 2L$-1 and $w_t = w_m + w_a + w_c$. These three weights are proportional to the number of "independent variables" in each constituent vector. This dis-

tance is limited to interval [0,1]. We define that the rotation-invariant vector of features at a pixel $A(x,y)$ is (t_d, t_m)-stable if:

$$\begin{cases} \underset{A(x',y')\in N(A(x,y))}{\mathrm{MAX}} \left(\Lambda\left(A(x,y), A(x',y')\right)\right) < t_d \\ |r_k| \geq t_m, 1 \leq k < K \end{cases}. \tag{21}$$

3 New Template Matching Algorithms

3.1 Searching for the Matching Candidates

As the central pixel of Q may not be stable, we extract a set of one or more circular templates $\{T_1, ..., T_N\}$, $T_i \subset Q$, such that the vector of features $v_f(T_i(x_o, y_o))$ at the central pixel (x_o, y_o) of T_i is (t_d, t_m)-stable and the canonical orientation $\angle r_1(T_i(x_o, y_o))$ is (t_a, t_m)-stable, for $1 \leq i \leq N$. To abridge, we will say that the templates T_i are (t_a, t_d, t_m)-stable. Figure 6 depicts the extraction of 8 stable templates from a query image. Whenever possible, we choose stable pixels that are located far from any unstable pixels. We also assure that a certain distance separates the chosen templates. This way, we assure the stability at the centers of the templates. However, the stability of A is not analyzed at all, because likely a stable template T_i will not match unstable regions of A. Then, for all pixels in A, we compute the radial and circular coefficients r_k and c_l, extract the canonical orientations and compute the vectors of features. We calculate the "matching distance images" D_i, defined as:

$$D_i(x,y) = \Lambda\left(v_f(T_i(x_o, y_o)), v_f(A(x,y))\right). \tag{22}$$

Then, we find n_c candidate pixels with the smallest values in D_i and store them in the vector of candidate pixels C_i. This algorithm can be summarized by the following pseudo-code:

function subtemplates_features_candidates(image A, image Q, int N)

{ Compute radial and circular coefficients for all pixels of Q;

Compute canonical orientation $\angle r_1$ and vector of features v_f for all pixels of Q;

Choose N stable sub-templates T_1, ..., $T_N \subset Q$;

Compute radial and circular coefficients for all pixels of A;

Compute canonical orientation $\angle r_1$ and vector of features v_f for all pixels of A;

For i = 1 to N {

For every pixel (x,y) of A

$$D_i(x, y) = \Lambda\left(v_f(T_i(x_o, y_o)), v_f(A(x, y))\right);$$

Find n_c pixels with the smallest values in D_i and store them in vector C_i;

}

Return [{T_i}, {$\angle r_1(T_i(x_o,y_o))$}, {$v_f(T_i(x_o,y_o))$}, $\angle r_1(A)$, $v_f(A)$, {D_i}, {C_i}];

}

This algorithm has two bottlenecks: (1) The fifth line: "Compute radial and circular coefficients of A." (2) The central double loop. Our implementation accelerates the first bottleneck using optimized FFT/IFFT functions provided by the OpenCV library. As a future work, we think that the second bottleneck can also be accelerated: (1) Using a special data structure, like kd-tree. Kd-tree is an algorithm used to find the nearest neighbors [17]. The original kd-tree algorithm is slow, but it can be accelerated if we allow it to find an approximate solution. (2) Writing the critical code directly in machine language using MMX/SSE instructions. (3) Using special hardware like GPU or FPGA.

3.2 Normalized Cross-Correlation

Using radial and circular features with the "stability" concept, we can detect the matching candidates. Yet, a matching candidate is not necessarily a true template matching, because many non-equivalent templates can generate the same rotation-invariant features. We took the necessary care to avoid false negative errors, but false

positive errors can still occur. So, a matching candidate must pass through other tests to decide whether it is a true or false matching.

The first of such tests is the normalized cross-correlation (NCC), also called correlation coefficient. The correlation coefficient between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined:

$$r_{xy} = \frac{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{y}}}{\|\tilde{\boldsymbol{x}}\|\|\tilde{\boldsymbol{y}}\|} \tag{23}$$

where $\|\cdot\|$ is Euclidean L^2-norm, $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \bar{\boldsymbol{x}}$ is the mean-corrected vector and $\bar{\boldsymbol{x}}$ is the mean of $\boldsymbol{x}$. Similar definitions are applicable to $\boldsymbol{y}$. NCC is limited to the range [-1, 1] and is invariant to brightness/contrast changes. Some care must be taken to avoid divisions by zero, especially in regions with constant gray-level (see [14] for more details).

The template matching using NCC becomes: Given a query image Q, extract one (t_a, t_d, t_m)-stable template sub-image $T \subset Q$. Find the set of matching candidates C by finding n_c pixels of A with the smallest matching distances. For each matching candidate pixel (x, y), compute the difference α between the canonical orientations of $A(x, y)$ and $T(x_o, y_o)$. As the consequence of the stability, the true difference of orientations is situated somewhere in the range $[\alpha\text{-}t_a, \alpha\text{+}t_a]$. So, for a discrete set of angles α_i in $[\alpha\text{-}t_a, \alpha\text{+}t_a]$, rotate T by α_i, translate it to (x, y), and compute pixel-by-pixel NCC between $A(x, y)$ and rotated and translated T. If one of the angles α_i yields sufficient high correlation, T matches $A(x,y)$ with rotation angle α_i. This algorithm can be summarized as below:

```
function Forapro_NCC(image A, image Q)
{  [T, ∠r1(T(xo,yo)), vf(T(xo,yo)), ∠r1(A), vf(A), D, C]
       = subtemplates_features_candidates(A,Q,1);
   For each matching candidate pixel (x,y) in C {
       α = ∠r1(A(x, y)) - ∠r1(T(xo, yo));
       For αi = α-ta to α+ta in discrete steps {
          If NCC(rotation(T,αi), A(x,y)) > threshold
             then T matches A at (x,y) with angle αi;
       }
   }
}
```

3.3 Hough Transform

The second test to separate true and false matchings is an algorithm inspired by the generalized Hough transform [18]. The underlying idea is, given a query image Q, to localize N sub-templates T_1, ..., $T_N \subset Q$ in A. It is possible that a single sub-template may generate a false positive error. However, if many sub-templates agree to point a pixel as the matching point, the probability of error is minimized. Our feature-based algorithm is appropriate for this approach, because finding N templates in the same image to analyze A is much faster than finding N templates in different images.

The algorithm using the generalized Hough transform becomes: Given a query image Q, extract N stable sub-templates. For each template T_i, generate the matching distance image D_i and find the set C_i of matching candidate pixels by selecting n_c pixels with the smallest matching distances. Let a matching candidate of sub-template T_i be located at (x', y'). The center of T_i matches $A(x', y')$. However, in order to compute the Hough transform, it is necessary to compute the pixel $A(x,y)$ that matches the center of Q (instead of T_i). Hence, we compute the rotation angle $\alpha = \angle r_1(A(x', y')) - \angle r_1(T_i(x_o, y_o))$. This angle, together with the position of T_i inside Q, allows us to compute the pixel $A(x,y)$. Increment the Hough transform accumula-

tor array $H(x, y)$ by $1 - D_i(x', y')$. Instead of incrementing the accumulator by 1, we increment it based on the matching distance, to provide a tie-breaking criterion. The matchings are the pixels of *H* with the largest values. The pseudo-code below summarizes this algorithm:

```
function Forapro_Hough1(image A, image Q)
{  [{T_i}, {∠r_1(T_i(x_o,y_o))}, {v_f(T_i(x_o,y_o))}, ∠r_1(A), v_f(A), {D_i}, {C_i}]
       = subtemplates_features_candidates(A,Q,N);
   Fill H with zeroes;
   for i = 1 to N {
      for j = 1 to n_c {
         Let j-th candidate pixel in C_i be located at (x', y');
         Compute the pixel (x,y) that matches Q;
         H(x, y) += 1 - D_i(x', y');
      }
   }
   The largest values in H indicate the matching positions;
}
```

We can take advantage of the small number of matching candidate pixels to accelerate the Hough transform, substituting the accumulator array *H* (of the same size as the image *A*) by a small matrix *V*. We will name this algorithm matrix Hough transform. Let us suppose that we use *N* sub-templates T_1, ..., T_N and n_c matching candidates for each template. Then, we construct a matrix *V* with *N* rows and n_c columns. Row *i* will be filled with the n_c matching candidates of sub-template T_i, sorted by the matching distances (that is, the best candidate occupies the leftmost column).

Let the *j*-th matching candidate of sub-template T_i be located at $A(x', y')$. As before, compute the pixel $A(x,y)$ that matches the center of *Q* and the rotation angle $\alpha = \angle r_1(A(x', y')) - \angle r_1(T_i(x_o, y_o))$. Each element $V[i, j]$ has four entries $[x, y, w, \alpha]$, where (x,y) is the pixel of *A* that matches the center of *Q*, $w = 1 - D_i(x', y')$ is the

weight of the matching (the larger the weight, the better the matching), and α is the rotation angle.

To find the best matching, we construct the set S of elements of V extracting at most one element of V per row, such that all elements of S are spatially near one another and the sum of their weights is maximal. The numerical example below clarifies these ideas:

$$V = \left[\begin{array}{l|l|l} 123,\ 137,\ 0.95,\ 0.18 & 125,\ 137,\ 0.89,\ 0.25 & 165,\ \ 4,\ 0.83,\ -2.45 \\ 228,\ \ 36,\ 0.90,\ 1.11 & 123,\ 135,\ 0.88,\ 0.04 & \ \ 1,\ \ 22,\ 0.84,\ 2.92 \end{array}\right]$$

In this example, we use N=2 sub-templates and n_c=3 matching candidates per sub-template. There are three elements that are spatially near one another: V[1,1], V[1,2], V[2,2]. As we can take at most one element per row, we can choose either S = {V[1,1], V[2,2]} or S = {V[1,2], V[2,2]}. The best matching is S = {V[1,1], V[2,2]}, because the weight 0.95 of V[1,1] is heavier than the weight 0.89 of V[1,2]. The algorithm that finds the best matching can be written using four nested loops. The spatial position of the matching is defined as the weighted average of the positions (x, y) in S, in the example:

x=(0.95×123+0.89×123)/(0.95+0.89) and y=(0.95×137+0.89×135)/(0.95+0.89).

The rotation angle of the matching is the weighted average of the angles α in S. Average of angles cannot be computed using the conventional formula. We transform each angle α with weight w in a complex number $w\cos(\alpha) + jw\sin(\alpha)$, add them up, and compute the angle of the resulting sum. In the example, the rotation angle is 0.113 radians. The function below summarizes this algorithm:

```
function Forapro_Hough2(image A, image Q)
{  [{T_i}, {∠r_1(T_i(x_o,y_o))}, {v_f(T_i(x_o,y_o))}, ∠r_1(A), v_f(A), {D_i}, {C_i}]
       = subtemplates_features_candidates(A,Q,N);
   for i = 1 to N {
       for j = 1 to n_c {
           Fill V[i,j] with data [x,y,w,α] of the j-th candidate pixels in C_i;
       }
   }
   Find the best matching in V as described in the text;
}
```

Using the Hough transform, Forapro can become robust to partial occlusions up to a certain degree, because the algorithm can locate the query image even if some of its sub-templates cannot be located.

3.4 Robustness to Scaling

The proposed algorithm can become robust to scale changes, that is, it can find the query image even if it appears in another scale in A (within a predefined range of scale factors). The matching is tested using the query image rescaled by a set of scale factors, making it a "brute force" approach. However, as the features of A are computed only once, it is not too slow. Robustness to scaling can be obtained using either NCC or Hough transform. The algorithm robust to scaling is: Given a query image Q, rescale it by a set of predefined scales, obtaining S rescaled query images Q_1, ..., Q_S. Find them all in A. Take the best matchings as the true matchings. As a future work, we think that the true scale invariance can be obtained by organizing image A in a pyramidal structure.

4 Experimental Results

4.1 Stability, Circular and Radial Features

In this subsection, we will demonstrate experimentally that:

1. The stability of template is essential to achieve a successful matching;
2. The combination of circular and radial features yields the best accuracy.

With this purpose, we took 24 memory game cards with 12 different figures. We scanned these cards 8 times, so that each one of the 12 figures appears only once in each scanned image (figure 7(a)). We placed magazines behind the cards to obtain non-blank backgrounds. These 8 images (each one with 471×347 pixels) will be our images to analyze A_1, ... A_8. From one of these images, we extracted the 12 query images Q_1, ..., Q_{12}, each one with 71×71 pixels (figure 7(c)). We selected (t_d=0.05, t_m=0.0025)-stable and unstable sub-templates from each query image (figures 7(d) and 7(e)). In this experiment, it is not necessary to select t_a, because we will not use the canonical orientation.

Then, we constructed the matching distance images $D(x,y)$ varying many parameters: using stable and unstable templates; varying the maximum order of the Fourier coefficients; and using circular/radial, only circular or only radial features. In each case, we took the pixel with the smallest value in $D(x,y)$ as the true matching pixel. We stress that there is no guarantee that this strategy will always work. Table 2 shows the number of observed errors using this strategy. We conclude that:

1. The minimal errors are obtained using stable templates and a combination of circular/radial features (column 2).
2. Using stable templates, radial features alone (column 4) yield better matching accuracy than circular features alone (column 3).
3. Even using only stable templates and high maximal order L, many errors are obtained when only circular features are used (column 3).

Our implementation takes 0.9 second in a Pentium 2GHz to find the 12 templates in an image to analyze A, subdivided in: 0.20s to compute the radial and circular coefficients of A, 0.55s to derive the features from the coefficients, and 0.15 seconds to compute the 12 distance images and find the 12 pixels with the smallest distance.

Our implementation uses optimized FFT/IFFT functions of the OpenCV library. However, the rest of our program is written in plain C++ and is not optimized.

4.2 The Size of Templates and the Maximum Order

In this subsection, we will analyze how the size $n_T \times n_T$ of the templates and the maximum orders K and L of the radial and circular coefficients affect the matching precision. We use only stable templates and a combination of circular/radial features (K=L) because we have already demonstrated that these choices lead to the most accurate results. Table 3 shows the obtained experimental data. We conclude that:

1. High accuracy is obtained using templates with 31×31 or more pixels. Too small templates cannot be localized with accuracy.
2. High accuracy is obtained using at least three radial and circular coefficients.

4.3 Forapro-NCC

In the last two subsections, we took the pixel in A with the smallest matching distance as the true matching pixel. This strategy may produce false positive errors, because occasionally even at a non-matching position the matching distance can be small. This fact is illustrated by non-zero error rates observed in the last two subsections, even using the most appropriate parameters.

We tested using NCC as a secondary filter, using ($t_a = 9^o$, $t_d = 0.05$, $t_m = 0.0025$)-stable 31×31 sub-templates. We varied the number of matching candidates n_c from 1 to 20 and chose the best matching pixel by computing NCC. For each matching candidate, NCC was computed 3 times with the template rotated at angles $\alpha_i \in \{-9, 0, 9\}$, because the canonical orientation may have imprecision of 9^o. Table 4 shows that no error was observed using 5 or more matching candidates. Our implementation takes 1.06 seconds to find the 12 query images in A, with K=L=3 and 5 matching candidates.

4.4 Forapro-Hough

Another possible secondary filtering is the matrix Hough transform, described in subsection 3.3. We tested this strategy using ($t_a = 9^o$, $t_d = 0.05$, $t_m = 0.0025$)-stable

sub-templates. We used K=L=4 and varied the size $n_T \times n_T$ of the sub-templates and the number N of templates. We always took 10 best matching pixels of each template to compute the Hough transform. Table 5 shows that using 4 or more sub-templates or using sub-templates with at least 31×31 pixels, no error occurs. Our implementation takes 1.90s to find the 12 query images in A, with n_T=31 and N=4.

4.5 Partial Occlusion

To test the robustness against partial occlusions, we copied 19×19 square blocks from 20 randomly chosen positions to other 20 random positions in each image to analyze. Figure 8 depicts part of such an image. Then, we tested the matching using the Hough transform, L=K=4, n_c=10 candidate pixels per sub-template, and varying the number N and the size $n_T \times n_T$ of the sub-templates. The results are depicted in table 6. Using N=8 or more templates, no error was observed. Large sub-templates intersect the occluded blocks with higher probability than small sub-templates. Hence, small sub-templates yielded fewer errors. Our implementation takes 3.61s to find the 12 query images in A, for n_T = 21 and N=8.

4.6 Robustness to Scaling Using NCC

To test robustness to scaling, we took 9 query images, resized them by scale factors chosen randomly in the range [0.7, 1.4], rotated them randomly and pasted them in random non-overlapping locations over a background image, resulting eight 400×400 images to analyze (figure 9(a)). Table 7 depicts the number of observed errors using the NCC test, varying the number of candidate pixels n_c and the number of scales n_s. The size of the sub-templates was 45×45 and K=L=4. Using 6 or more scales and 20 or more matching candidates, no error was observed. Our implementation took 3.20s to find all 9 query images in an image to analyze.

4.7 Robustness to Scaling Using Hough Transform

We repeated the test of the last subsection using the Hough transform. We kept constant the size of the sub-templates at 27×27, K=L=4, and the number of the matching candidates n_c=10 per sub-template. We varied the number of scales n_s and

the number of sub-templates N used in each Hough transform. The observed number of errors is depicted in table 8. Using N=4 or more sub-templates, no error was observed, even using the number of scales n_s as small as 3. Our implementation takes 3.86s to find the 9 query images in an image to analyze.

4.8 NCC versus Hough

According to the theoretical and experimental data gathered so far, we conclude that Forapro-Hough is superior to Forapro-NCC because:

1. Only Forapro-Hough is robust to partial occlusions.
2. Forapro-Hough becomes robust to scaling using less discrete scales than Forapro-NCC. The former needed only 3 scales, while the latter needed 6 scales to not make errors.

Consequently, from now on we will test only Forapro-Hough.

4.9 Comparison with SIFT

SIFT (Scale Invariant Feature Transform) [9] is a very popular, accurate and efficient image matching algorithm. It extracts some scale-invariant key-points and computes their associated local features (scale, rotation and local textures). Then, to find a query image Q in A, it is necessary only to match the key-points. We will compare our technique with SIFT. First, we evaluated the SIFT implemented by Lowe[1] using the three sets of images used so far (subsections 4.1, 4.5 and 4.6). SIFT made no error, like our algorithm. The processing times also were of similar magnitudes (typically SIFT takes 1.5 seconds to compute the key-points of an image to analyze).

Figure 10 presents an example created especially to explore SIFT's weakness. As SIFT is based on key-points and local textures, it may fail to find simple shapes with constant grayscales. We tested finding 13 simple query images in 4 different images to analyze. Our algorithm successfully found all the query images. Meanwhile, SIFT made 6 errors (out of 52). To find 13 query images in each image to analyze, both

[1] http://www.cs.ubc.ca/~lowe/keypoints/

algorithms took approximately 3 seconds. However, note that our algorithm is making one-scale searching, while SIFT is making scale-invariant matching.

To further compare Forapro with SIFT, we used the 8 sets of images provided by K. Mikolajczyk[2]. Each set consists of 6 images, totalizing 48 images. Some of them are depicted in figure 11. This database is adequate to test the image searching algorithm's robustness to focus blur, viewpoint changes (perspective), camera aperture, JPEG compression, zoom and rotation. For each set, we extracted 50 query images from the first image and searched for them in the 6 images. Thus, each experiment consisted of 300 searchings, whose results can be correct (the algorithm correctly localizes the query image) or erroneous (the algorithm points a incorrect location or fails to locate the query image).

Table 9 depicts the number of errors in each experiment and the principal parameters. We extracted large query images (n_Q = 211×211 pixels) in the sets that have great scale variations (Bark and Boat) because they become quite small in some of the images (figures 11(a) and 11(b)). In the remaining sets, we extracted relatively small query images (n_Q = 91×91 pixels).

SIFT is completely scale-invariant. However, Forapro needs a pre-defined scale range. We defined adequate Forapro scale ranges for the sets Bark, Boat, Graf and Wall. For the remaining sets, one-scale searching is the most appropriate choice. However, we tested both one-scale and multi-scale searchings because we deemed not fair to compare a multi-scale searching algorithm versus one-scale technique.

In average, Forapro made fewer errors than SIFT. Remarkably, Forapro made substantially fewer errors than SIFT in sets Bikes (6 versus 111 errors) and Leuven (5 versus 52 errors), depicted in figures 11(c)-11(f). Probably, focus blur and camera aperture destroy the local texture, what hinders SIFT from localizing the query images. This is the same weakness already illustrated in figure 10. We conclude that Forapro is especially suited for applications where the local texture cannot be easily identified and the scale variation is limited within some range. Some additional observations:

[2] http://www.robots.ox.ac.uk/~vgg/research/affine/

- Rarely, no SIFT key-point was found inside a query image. However, even in many of these cases, Forapro found some stable sub-templates (figure 12) and succeeded to localize the query images.
- In set Bark, both Forapro and SIFT localized correctly the 300 queries.
- Forapro and SIFT are not affine-robust neither perspective-robust. Consequently, both algorithms made many mistakes in sets Graf and Wall.
- In set Trees, the tree leaves are waving in the wind (beside the focus blur that distorts the images). So, both algorithms made many errors.
- Typically, SIFT takes 6s to make a searching and Forapro takes 8s.

We also tested robustness to illumination variation using a subset of ALOI database[3], with the resolution reduced 4 times. We took the 300 images of the first 100 objects of the database under illumination conditions 6, 7 and 8, taken by camera 1, as the query images. We used the 100 images under illumination condition 8, clustered in 5 large images as the images to analyze. The observed errors are depicted in table 9, indicating that both Forapro and SIFT are only moderately robust to illumination changes.

5 Conclusions

This paper has proposed a new feature-based, brightness/contrast-invariant and rotation-discriminating template matching. We use the first complex Fourier coefficients of the local radial projections to compute the new features. These coefficients can be computed efficiently and directly using the appropriate kernels and FFT, without explicitly computing the radial projections. From the radial Fourier coefficients, we have derived three classes of features: the canonical orientation that indicates the local orientation; rotation-invariant features derived from the magnitudes of the radial Fourier coefficients; and those derived from the angles of the coefficients. We have used these features, together with the features derived from the circular projections, to obtain a new template matching. We have classified the templates in "stable" and "unstable" and argued that any local feature-based template matching may fail to find an unstable template. So, we have suggested extracting stable sub-

[3] http://staff.science.uva.nl/~aloi/

templates from the query images and finding them in the image to analyze. As the canonical orientation computes the rotation angle, it is easy to filter the false positive matchings, using either the normalized cross correlation or the Hough transform. Using the Hough transform, we have obtained a template matching robust to scaling and partial occlusions. Many experiments have shown that the proposed algorithm is robust and efficient.

6 References

[1] J.P. Lewis, Fast template matching, Vision Interface (1995) 120-123.

[2] L.D. Stefano, S. Mattoccia, F. Tombari, ZNCC-based template matching using bounded partial correlation, Pattern Recognition Letters (26) 2005 2129-2134.

[3] W.Y. Kim, P. Yuan, A practical pattern recognition system for translation, scale and rotation invariance, Proc. Int. Conf. Comput. Vis. Pattern Recognit. (1994) 391-396.

[4] L.A. Torres-Méndez, J.C. Ruiz-Suárez, L.E. Sucar, G. Gómez, Translation, rotation and scale-invariant object recognition, IEEE Trans. Systems, Man, and Cybernetics - part C: App. and Reviews 30(1) (2000) 125-130.

[5] M.K. Hu, Visual pattern recognition by moment invariants. IRE Trans. Inform. Theory 1(8) (1962) 179-187.

[6] C.H. Teh, R.T. Chin, On image analysis by the methods of moments. IEEE Trans. Pattern Analysis Machine Intelligence 10(4) (1988) 496-513.

[7] F. Ullah, S. Kaneko, Using orientation codes for rotation-invariant template matching. Pattern Recognition 37 (2004) 201-209.

[8] D. Marimon, T. Ebrahimi, Efficient rotation-discriminative template matching, 12th Iberoamerican Congress on Pattern Recognition, Lecture Notes in Computer Science 4756 (2007) 221-230.

[9] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Computer Vision 60(2) (2004) 91-110.

[10] K. Mikolajczyk, A performance evaluation of local descriptors, IEEE. T. Patt. Analysis Machine Intelligence 27(10) (2005) 1615-1630.

[11] Y. Tao, Y.Y. Tang, The feature extraction of chinese character based on contour information. Int. Conf. Document Analysis Recognition (1999) 637-640.
[12] D.M. Tsai, Y.H. Tsai, Rotation-invariant pattern matching with color ring-projection, Pattern Recognition 35 (2002) 131-141.
[13] Y.H. Lin, C.H. Chen, Template matching using the parametric template vector with translation, rotation and scale invariance, Pattern Recognition 41(7) (2008) 2413-2421.
[14] H.Y. Kim, S.A. Araújo, Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast, IEEE Pacific-Rim Symp. Image and Video Tech., Lecture Notes in Computer Science 4872 (2007) 100-113.
[15] H.P.A. Nobre and H.Y. Kim, "Automatic VHDL Generation for Solving Rotation and Scale-Invariant Template Matching in FPGA," V Southern Programmable Logic Conference, pp. 21-26, 2009.
[16] M.S. Choi, W.Y. Kim, A novel two stage template matching method for rotation and illumination invariance, Pattern Recognition 35(1) (2002) 119-129.
[17] J.H. Friedman, J.L. Bentley, R.A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Trans. Mathematical Software 3(3) (1977) 209–226.
[18] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognition 13(2) (1981) 111-122.

Tab. 1: Variations of canonical orientations of stable pixels when the images are distorted (columns 2-5); and variations of both stable and unstable pixels (columns 6-7).

	λ=15, t_a=10°. (stable pixels only)		λ=30, t_a=5°. (stable pixels only)		λ=30, t_a=∞. (stable and unstable pixels)	
	average	maximum	average	maximum	average	maximum
contrast 35%	0.06°	3.89°	0.04°	1.05°	0.10°	38°
rotation 30°	0.29°	**9.49°**	0.04°	0.86°	0.11°	119°
sub-pixel shifting	1.78°	6.69°	0.92°	**3.37°**	2.99°	**178°**
noise	0.20°	2.13°	0.10°	2.01°	0.25°	67°

Tab. 2: Observed errors using templates with 31×31 pixels, without the secondary filtering. Using stable templates, radial features yield considerably fewer errors than circular features (columns 3 and 4).

Errors (maximum=96)	Stable templates			Unstable templates		
	circ/rad K=L	circular K=0	radial L=0	circ/rad K=L	circular K=0	radial L=0
K and/or L = 3	1	60	47	33	55	80
K and/or L = 4	1	48	18	23	52	61
K and/or L = 5	1	40	8	15	22	46
K and/or L = 6	1	37	1	7	25	41

Tab. 3: Observed errors using only stable templates and both radial and circular features (K=L), without the secondary filtering.

Errors (maximum=96)	K=L=2	K=L=3	K=L=4	K=L=5	K=L=6	K=L=7
n_T = 21	58	15	11	0	2	0
n_T = 31	29	1	1	1	1	0
n_T = 41	25	3	0	0	1	6
n_T = 51	13	0	2	2	0	1
n_T = 61	12	1	1	0	0	1

Tab. 4: Observed errors using Forapro-NCC. We used sub-templates of size n_T×n_T = 31×31.

Errors (maximum=96)	K=L=3	K=L=4	K=L=5
n_c = 1 candidate	1	1	1
n_c = 2 candidates	1	1	1
n_c = 4 candidates	1	0	0
n_c = 5 candidates	0	0	0
n_c = 10 candidates	0	0	0
n_c = 20 candidates	0	0	0

Tab. 5: Observed errors using Forapro-Hough, with n_c=10 candidate pixels per sub-template.

Errors (maximum=96)	N=2	N=4	N=6
n_T = 21	2	0	0
n_T = 31	0	0	0
n_T = 41	0	0	0
n_T = 51	0	0	0

Tab. 6: Robustness to partial occlusion using Forapro-Hough.

Errors (maximum=96)	N=4	N=6	N=8	N=10	N=12
n_T = 15	9	0	0	0	0
n_T = 21	2	0	0	0	0
n_T = 25	5	1	0	0	0

Tab. 7: Robustness to scaling using Forapro-NCC.

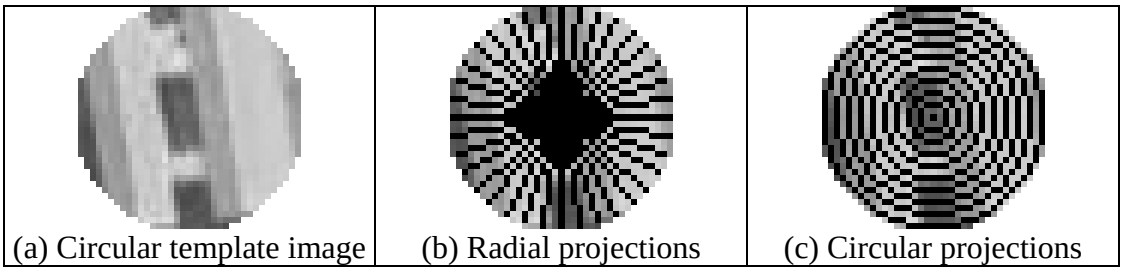
Errors (maximum=72)	n_c = 10	n_c = 20	n_c = 45
n_s = 4	8	10	4
n_s = 5	3	0	1
n_s = 6	1	0	0
n_s = 8	1	0	0
n_s = 12	0	0	0

Tab. 8: Robustness to scaling using Forapro-Hough.

Errors (maximum=72)	N = 3	N = 4	N = 5
n_s = 3	1	0	0
n_s = 4	0	0	0
n_s = 5	1	0	0
n_s = 6	0	0	0

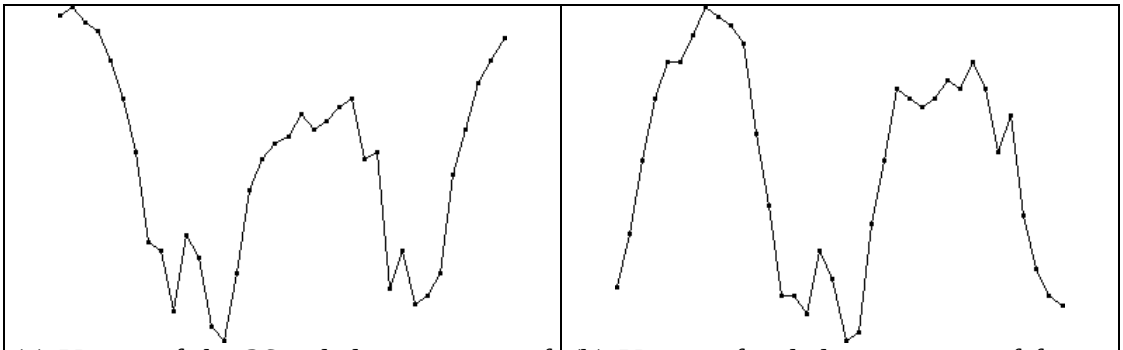
Tab. 9: Comparison between Forapro and SIFT. In each experiment, 50 query images were searched for in 6 images to analyze, resulting in 300 searchings.

Maximum number of errors = 300	Query images size n_Q	One-scale Forapro errors	Forapro's scale range	Multiscale Forapro errors	SIFT errors
Bark (zoom and rotation)	211×211	-	0.28-1.0	0	0
Bikes (focus blur)	91×91	1	0.5-2.0	6	111
Boat (zoom and rotation)	211×211	-	0.36-1.0	9	6
Graf (viewpoint)	91×91	-	0.5-1.0	109	110
Leuven (camera aperture)	91×91	3	0.5-2.0	5	52
Trees (focus blur)	91×91	80	0.5-2.0	76	134
UBC (JPEG compression)	91×91	35	0.5-2.0	35	52
Wall (viewpoint)	91×91	-	0.7-1.0	90	70
ALOI (illumination)	144×192	9	0.5-2.0	46	52
Average				41.8	65.2



(a) Circular template image (b) Radial projections (c) Circular projections

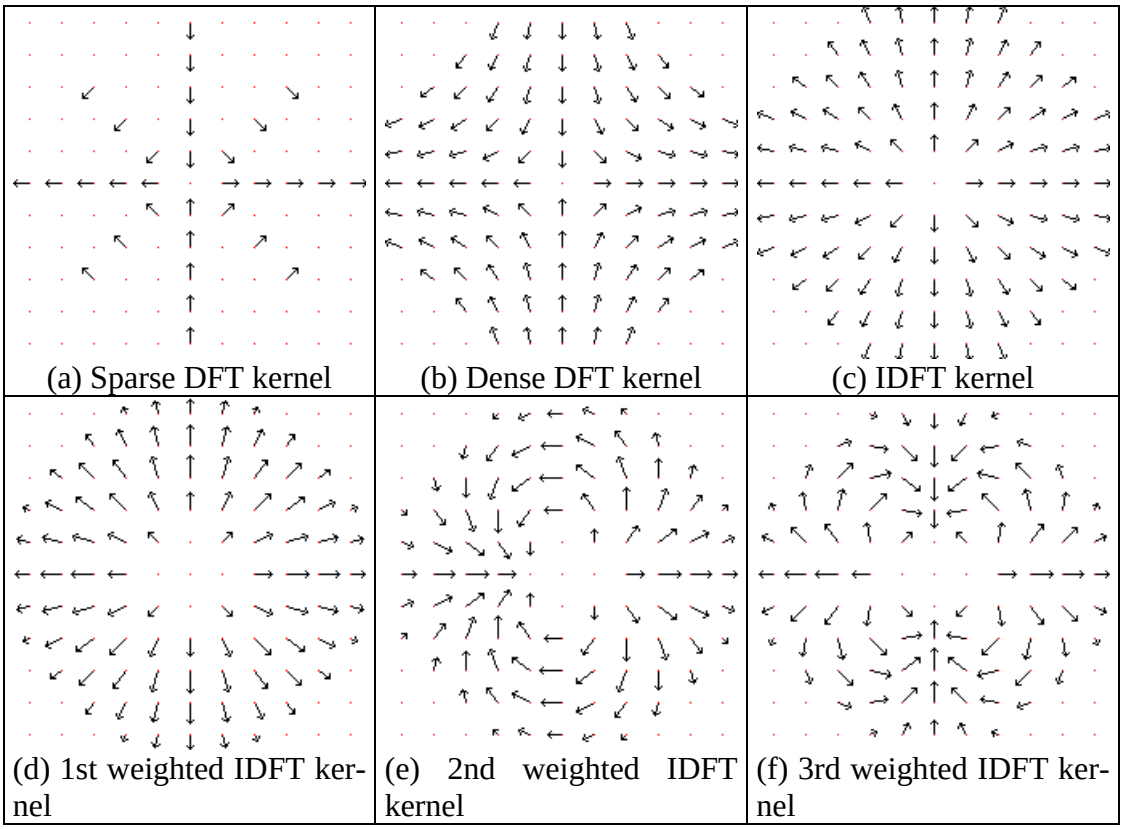
Fig. 1: A radial (circular) projection is the mean grayscale on a radial line (circular ring).



(a) Vector of the 36 radial projections of figure 1(a).

(b) Vector of radial projections of figure 1(a) rotated 60^o

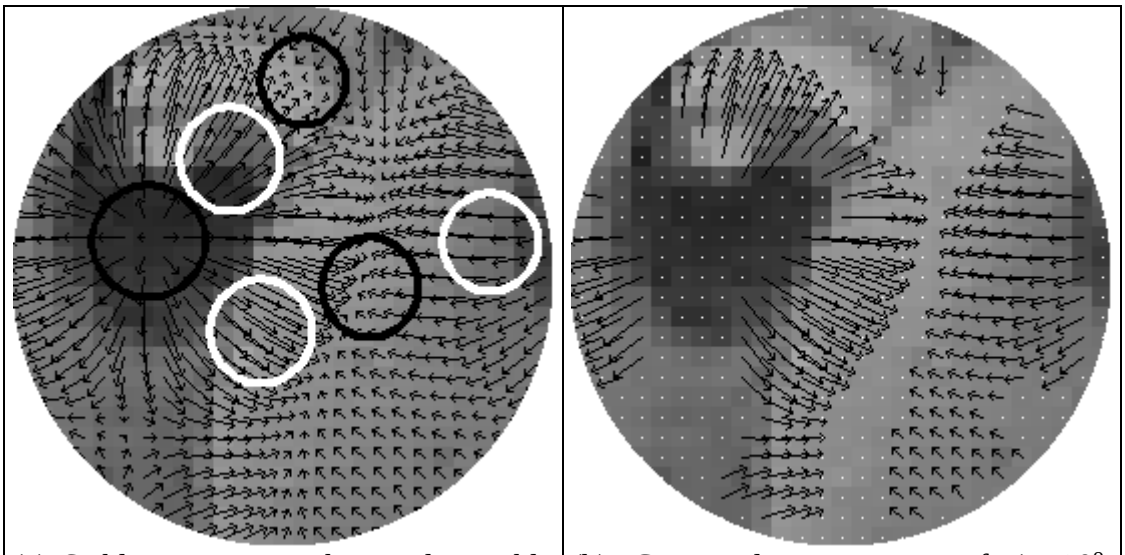
Fig. 2: Rotation of the image causes circular shifting in the vector of radial projections.



(a) Sparse DFT kernel (b) Dense DFT kernel (c) IDFT kernel

(d) 1st weighted IDFT kernel (e) 2nd weighted IDFT kernel (f) 3rd weighted IDFT kernel

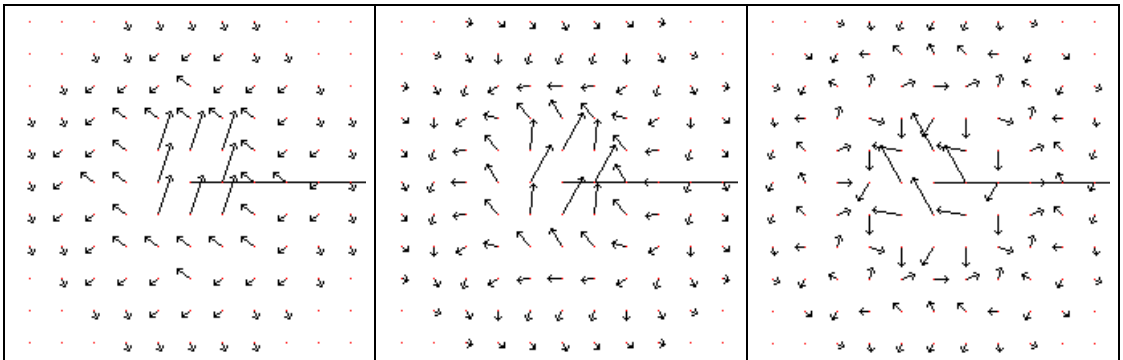
Fig. 3: Radial kernels



(a) Stable regions in white and unstable regions in black.

(b) Canonical orientations of (t_a=20^o, t_m=0.01)-stable pixels.

Fig. 4: Stability of the canonical orientation.



(a) Truncated integer circular kernel.

(b) Floating-point circular kernel.

(c) Second-order circular kernel.

Fig. 5: Circular kernels.

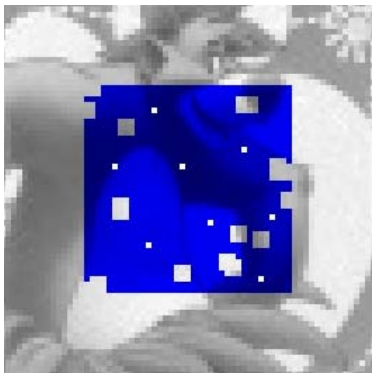
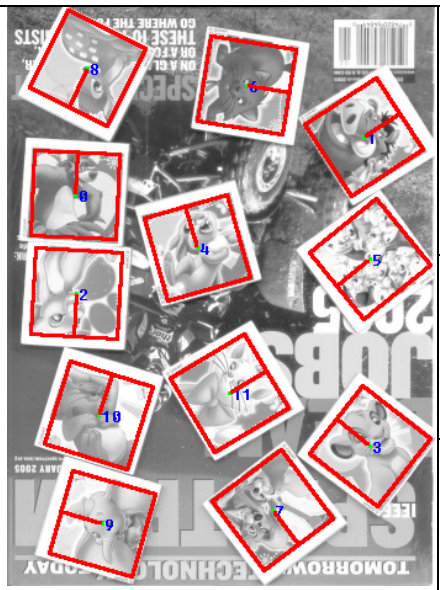


Fig. 6: Extraction of eight stable templates from a query image. The stable pixels are depicted in blue and the centers of the eight chosen templates are depicted in white.

(a) One of the eight images to be analyzed.



(b) The output of our algothm.

(c) A query image.

(d) A stable sub-template.

(e) An unstable sub-template.

Fig. 7: One of the images used to test Forapro, with the respective output.

Fig. 8: Part of an image used to test the robustness to partial occlusions.

(a) An image to be analyzed. (b) The output of our algorithm.

Fig. 9: A test to verify the robustness to scaling.

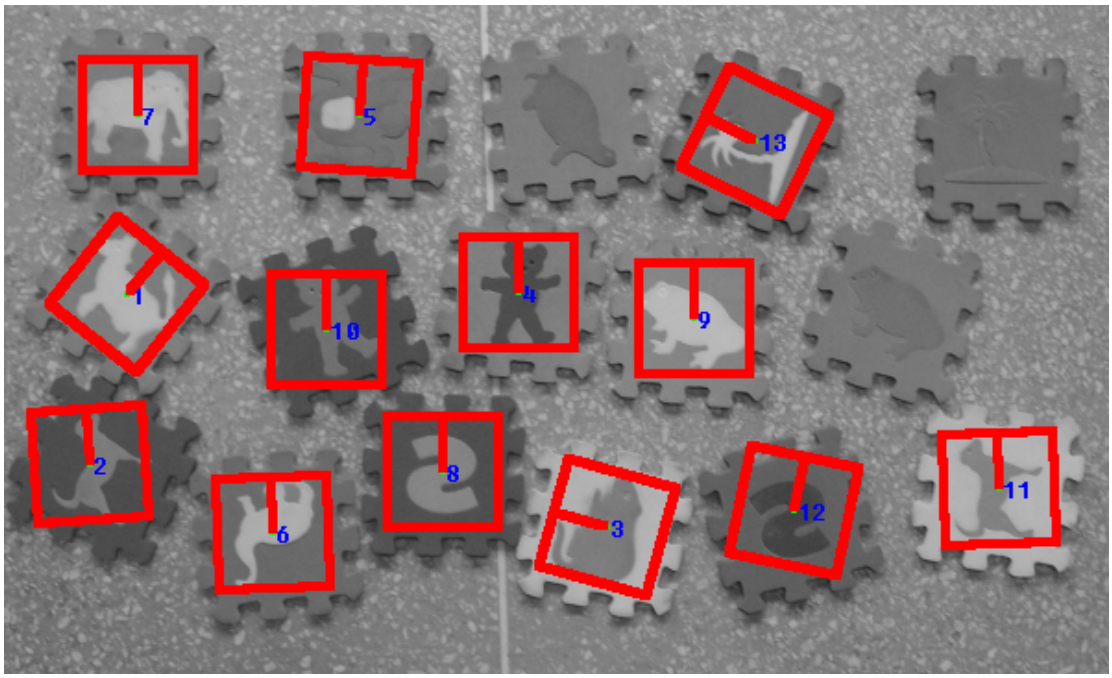
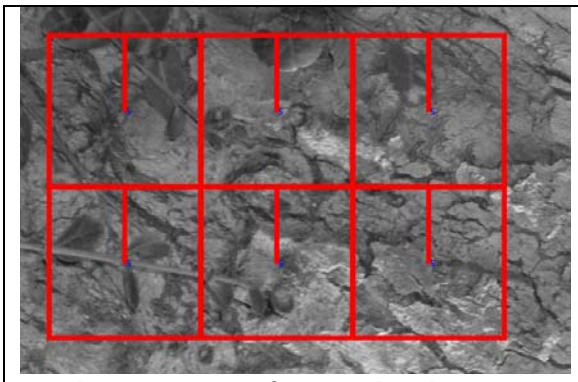


Fig. 10: SIFT may fail to find simple shapes with constant grayscale, like the ones in this figure. SIFT made 6 errors (out of 52) while Forapro made no error.



(a) The 1st image of set Bark. This set tests robustness to rotation and zoom.

(b) The 6th image of the set.

(c) The 1st image of set Bikes. This set tests robustness to focus blur.

(d) The 6th image of the set.

(e) The 1st image of set Leuven. This set tests robustness to camera aperture.

(f) The 6th image of the set.

Fig. 11: Some of the images used to compare SIFT with Forapro. Both SIFT and Forapro localized correctly 300 queries in set Bark. SIFT made substantially more errors than Forapro when the local textures could not be easily extracted, like in sets Bikes and Leuven.

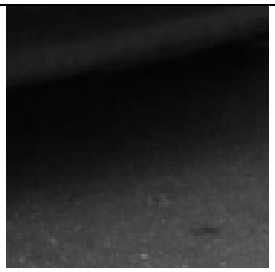
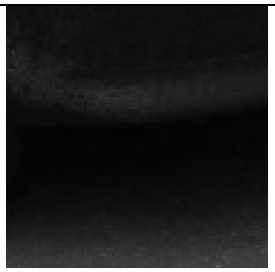
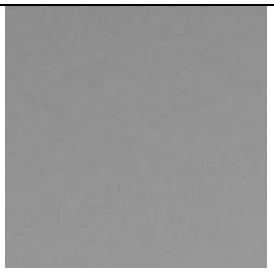
 (a) Forapro made 4 errors (out of 6).	 (b) Forapro made no error.	 (c) Forapro found no stable sub-template and made 6 errors.

Fig. 12: SIFT could find no key-point in these three query images and made 6 errors in each. Forapro found some stable sub-templates and made fewer errors.