FASTER TEMPLATE MATCHING WITHOUT FFT

Kimmo Fredriksson and Esko Ukkonen

Department of Computer Science, University of Helsinki
PO Box 26, FIN–00014 Helsinki, Finland
email: kfredrik@cs.helsinki.fi, ukkonen@cs.helsinki.fi

ABSTRACT

We consider the *template matching problem*, in which one wants to find all good enough *occurrences* of a *pattern template* P from a larger *image* I. The current standard of template matching is based on cross correlation computed in Fourier domain, using the Fast Fourier Transform (FFT). This can be extended to rotations of the template by a suitable rotational sampling of the template. An alternative approach, pursued here, is to solve the problem in the spatial domain using so–called *filtering* algorithms. Such algorithms scan the image quickly and find all promising areas for a more accurate (but slower) examination such that no good enough occurrences of the template are lost. The paper shows that the filtering approach can be orders of magnitude faster than the FFT based method. Especially in the 3D case the FFT is intolerably slow.

1. INTRODUCTION

Rotation invariant template matching has numerous important applications in image and volume processing. Traditional approach [1] to the problem is to compute the cross correlation between each image location and each rotation of the pattern template. This can be done reasonably efficiently using the FFT. This requires time proportional to $\mathcal{O}(K|I|\log|I|)$ where $|I|$ is the number of pixels in the input image, and K is the number of rotations sampled. If we use rotational sampling step of $1/m$ radians (which can be justified using Shannon's Sampling Theorem; see also [2]), where m is the width of the template, then $K = \mathcal{O}(m)$ in the two–dimensional (2D) case. In 3D the number of rotations becomes very large, $K = \mathcal{O}(m^3)$, which makes the FFT approach very slow in practice.

In many applications only the "close enough" matches of the pattern are accepted. To this end, the user may specify a parameter κ, such that only matches that have distance (similarity) at most (at least) κ should be accepted. The distance measure can be e.g. the number of mismatching pixels between the image and the pattern. (*Hamming distance*).

In [3] it was shown how one can filter a 2D image in time $\mathcal{O}(|I|\kappa^{1/2})$ to find the image locations that may have distance at most κ, for *any* orientation of the pattern. In 3D the algorithm of [4] works in time $\mathcal{O}(|V|\kappa^{2/3})$, where $|V|$ is the number of voxels in the 3D volume. The image/volume locations that pass the filter, are then inspected more carefully to find the orientation that the pattern should have the *verification phase*. For this, one may use the filtering algorithm again. This requires time $\mathcal{O}(|P|^{2/3})$ in 3D for each volume location that passed the first filtering phase. This second pass filtering gives the best orientations of the pattern, which can then be verified using e.g. brute force method. The algorithms can be generalized with small efficiency penalty to many other distance/similarity functions, including e.g. the important sum of absolute differences (*SAD*), and cross correlation (*CC*) functions.

Work supported by the Academy of Finland.

In this paper we experimentally compare the performance of the filtering approach to the traditional FFT based correlation method. A brief overview of the filtering algorithms is also given.

2. ALGORITHMS

Let $I = I[1..n, 1..n]$ and $P = P[1..m, 1..m]$ be two dimensional arrays of *point samples*, such that $m < n$. Each sample has a *color* in a finite ordered *alphabet* Σ. The size $|\Sigma|$ of Σ is denoted by σ. We call I the input *image*, and P the *pattern* template that is sought from I.

The arrays P and I are point samples of colors of some "natural" two dimensional image. There are several possibilities to define the mapping from P to I, that is needed to compare the colors of P to colors of I. Our approach to the problem is combinatorial. Assume that P has been put on top of I, in some arbitrary position. Then we will compare each color sample of P against the color of the closest sample of I. The closeness between the samples is measured by using the Euclidean distance. When P moves with respect to I, this gives only a finite number of different mappings that can be explicitly generated to find mappings giving the best match of the colors.

The Voronoi diagram for the samples is a regular array of unit squares, which we call *pixels*. Hence we may define that the array $I = I[1..n, 1..n]$ consists of n^2 unit squares (pixels) in the real plane $\mathbb{R}^2$. In the 3D case we use term *voxel* instead of pixel and V instead of I.

Each pixel has a *center* which is the geometric center point of the pixel, i.e., the center of the pixel for $I[i,j]$ is $(i - \frac{1}{2}, j - \frac{1}{2})$. The array of pixels for the pattern P is defined similarly. The *center* of P is the center of the pixel in the middle of P. Precisely, assuming for simplicity that m is odd, the center of P is the center of pixel $P[\frac{m+1}{2}, \frac{m+1}{2}]$. The colors are associated with the center points of the pixels, that is, the centers are the sampling points.

Assume now that P has been moved on top of I using a rigid motion (translation and rotation), such that the center of P coincides exactly with the center of some pixel of I. The location of P with respect to I can be uniquely given as $((i - \frac{1}{2}, j - \frac{1}{2}), \alpha)$, where $(i - \frac{1}{2}, j - \frac{1}{2})$ is the location of the center of P in I, and α is the angle between I and P. The occurrence (or more generally, distance) between I and P at some location, is defined by comparing the colors of the pixels of I and P that overlap. We will use the centers of the pixels of P for selecting the comparison points from I. That is, for P at location $((i-\frac{1}{2}, j-\frac{1}{2}), \alpha)$, we look which pixels of I cover the centers of the pixels of P, and compare the colors of those pixels. As P rotates, the centers of the pixels of P move from one pixel of I to another. In [5] it is shown that this happens $\mathcal{O}(m^3)$ times, so there are $\mathcal{O}(m^3)$ relevant orientations of P to be checked.

More precisely, let us define the *matching function* M when P is at location $((u, v), \alpha)$ as follows.

Definition 1 *For each pixel $P[r,s]$ of P, let $I[r',s']$ be the pixel of I such that the center of $P[r,s]$ belongs to the area covered by $I[r',s']$. Then $M(P[r,s]) = I[r',s']$.*

Note that M is many–to–one mapping. This fact must be reflected in our algorithms.

We use distance (similarity) function $d(p)$ to compute the distance between the colors of pixels p and $M(p)$. Our algorithms take the distance function as a "black box", and assume only that it can be computed in a unit time per pixel.

The distance between P and I is defined as follows.

Definition 2 *The distance (similarity) D between I and P located at position $((u,v),\alpha)$ in I is $D = D(P, I, ((u,v),\alpha)) = \sum_{r,s} d(P[r,s])$.*

The distance function $d(p)$ can be, e.g.,

- $d(p) = 0$, if $\mathit{color}(p) = \mathit{color}(M(p))$, and 1 otherwise; the corresponding distance D is called the *Hamming distance*.
- $d(p) = |\mathit{color}(p) - \mathit{color}(M(p))|$; the corresponding distance D is the *SAD* distance (sum of absolute differences).

where $\mathit{color}(p)$ denotes the color of pixel p. Normalized cross correlation (CC) is a similarity function defined as:

$$D = \frac{\sum_{p \in P} \mathit{color}(p)\, \mathit{color}(M(p))}{\sqrt{\sum_{p \in P} \mathit{color}(M(p))^2}}.$$

We concentrate on the following *filtration* approach: given a parameter κ, decide quickly if $D \leq \kappa$ (for similarity, $D \geq \kappa$), for any angle α. In many applications one wants to find only reasonably good matches of the pattern template. Therefore, we may use some suitable threshold value κ to discard uninteresting parts of I quickly, and thus solve the original problem efficiently. The remaining parts of I may be verified using e.g. a brute force approach.

Our algorithms have the following property. If they state that the distance $D \leq \kappa$ (or $D \geq \kappa$ for similarity), they may be wrong, so this must be verified using some other algorithm, but if they state that the distance $D > \kappa$ ($D < \kappa$ for similarity), it is always correct. Hence the algorithms do not miss any good enough positions, that is, there are *no false negatives* in the filtration.

2.1. Hamming distance

The filtering algorithm is based on the ideas of [6] and [7]. Consider each pixel p of P such that the distance from the center of p to the center of P is at most $R \leq (m-1)/2$ (assuming odd m). These pixels fall inside a discrete circle whose radius is R. Now calculate a *histogram* of the color distribution of those pixels. This histogram is (almost) rotation invariant, and together with corresponding histograms from I it can effectively filter unpromising positions of I. The histogram $\mathcal{H}$ has one bin for each color in Σ. Let $\mathit{color}(p)$ denote the color of pixel p. The histogram for position (i,j) of I is defined as follows.

Definition 3 $\mathcal{H}_{I[i,j]}(c) = \sum_{i',j'} \{1 \mid \mathit{color}(I[i+i', j+j']) = c;\ \sqrt{i'^2 + j'^2} \leq R + \sqrt{2}/2\}$.

Note that it is easy to obtain $\mathcal{H}_{I[i+1,j]}$ from $\mathcal{H}_{I[i,j]}$ incrementally in time $\mathcal{O}(R)$. To see this, note that when the $\mathcal{H}_I$ is translated by one pixel along a coordinate axis, about $2R$ new pixels of I will come inside the circle, and exactly the same number of pixels of I will go out the circle.

We need also a histogram $\mathcal{H}_P$ for P. This is slightly more complicated to obtain than $\mathcal{H}_I$, and it depends somewhat on the distance/similarity function used. For details, see [3].

Our filter compares $\mathcal{H}_{I[i,j]}$ and $\mathcal{H}_P$ at each position (i,j) of I. $\mathcal{H}_P$ tells for each color the minimum number of pixels of that color that is at least required for a complete match. The difference $b(i,j)$ between $\mathcal{H}_{I[i,j]}$ and $\mathcal{H}_P$ gives a lower bound of the number of mismatches, i.e. the Hamming distance at position (i,j) of I. This is computed as the number of colors missing from $\mathcal{H}_P$:

$$b(i,j) = \sum_{c \in \Sigma} \max(\,0, \mathcal{H}_P(c) - \mathcal{H}_{I[i,j]}(c)\,).$$

If $b \leq \kappa$, then there may be an approximate occurrence of P in I with the Hamming distance $\leq \kappa$. Note that b can be computed incrementally also, in the same way as histograms $\mathcal{H}_{I[i,j]}$, in time $\mathcal{O}(R)$.

To get a good filtration capacity, the radius R must be chosen such that the expected $b(i,j)$ is greater than κ. At minimum, the number of pixels inside the circle must be greater than κ. This gives approximately $\pi R^2 > \kappa \Rightarrow R > \sqrt{\kappa/\pi}$. If $R = \mathcal{O}(\kappa^{1/2})$, the filtering takes time $\mathcal{O}(|I|\kappa^{1/2})$. For exact analysis of the expected value of b in 2D case, see [3].

Theorem 1 *For small κ the κ–threshold Hamming distance problem can be solved in expected time $\mathcal{O}(|I|\kappa^{1/2})$.* $\square$

It remains open how to choose a suitable κ. If just the best match is desired, then the following approach might be taken. First, filter I for $\kappa = \infty$ (that is, choose $R = m/2$), and then verify the image/volume locations by increasing distance (decreasing similarity), and decrease κ as better and better matches are found.

2.2. Histogram filters for CC and SAD

It is possible to generalize the histogram filter for many other distance functions, see [3]. The SAD filter is reported to run in time $\mathcal{O}(|I|(\sigma + R^2))$, although this can be improved to $\mathcal{O}(|I|(\sigma \log \sigma + R))$, by reducing the problem to the transportation problem, instead of to the bipartite graph matching. In 3D this requires $\mathcal{O}(|V|(\sigma \log \sigma + R^2))$ time. Our implementation uses the slower method (see the experimental results).

The CC filter requires $\mathcal{O}(|I|(\sigma + |P|^{1/2}))$ time in 2D, and $\mathcal{O}(|V|(\sigma + |P|^{2/3}))$ time in 3D. It should be noted that the CC filter does not compute the upper bound of the correlation for the whole template, but only for the maximal circle (sphere in 3D) shaped subpattern of the template. Fortunately, this is not a major problem in most applications.

2.3. Rotational filtering

Our filtering methods do not give the orientation of the template. The algorithms are, however, easy to generalize for orientation filtering also. The idea, in e.g. 3D, is to take two more histograms from P, so that the centers of the three histograms do not lie on the same line. These three histograms are searched parallelly, and the corresponding distance bounds are recorded. If the normal filter indicates a possible match, then we proceed with the two other histograms to find the possible orientation for P. This can be implemented in time $\mathcal{O}(|P|^{2/3})$, and only after this filtration we verify the match for the (hopefully few) remaining possible orientations. We have not yet implemented this method.

2.4. Cross Correlation by FFT

Cross correlation without the normalization can be easily computed by FFT. The computation is based on the well known convolution theorem. Due to the limited space, we refer the reader to e.g. [1] for discussion and the references therein for more information of the subject. For the FFT, we have used the FFTW library [8]. The normalization factors can be computed incrementally in spatial domain in time $\mathcal{O}(|I|)$ for all locations of I. For the rotational sampling, a step size of $1/w$ radians was used, where w is the width of the template. This means that the FFT method requires time $\mathcal{O}(|P|^{1/2}|I|\log|I|)$ in 2D, and time $\mathcal{O}(|P||V|\log|V|)$ in 3D.

3. COMPARISON

We have implemented the algorithms in C, compiled them with gcc 2.91.66, running under Linux 2.2.17, on Pentium III 700MHz processor.

We mainly investigated the filtering capability of SAD, and compared its running time for different κ against the FFT based method. SAD was chosen, because it is easier to cut of the verification when the distance grows past the limit κ. Also, it is attributed to give better results than CC, in practice.

Fig. 1 shows our input image for the 2D experiments. The template which is one of the planes, was extracted from that image. The "correlation" images for some distance measures are shown in Fig. 2. The pixel values show how similar the corresponding image position and the template are, (according to the histogram filter); the brighter the color, the higher the similarity. For the FFT based correlation image, the pixel (correlation) value is maximized over all rotations. The (histogram) correlation images may be useful us such, as the highest peak seems to be in correct position, so verification may be ignored. This may not be true for all inputs.

Figures 3, 4 and 5 show the performance of the algorithms. The filtering capability is reasonably high. Most of the filtering algorithms are also fast. Using the SAD function requires a lot of time compared to the other methods, but this could be improved, see Sec. 2.2. The SAD matching code is much more complex than the code for the other algorithms, also in terms of constant factors.

For the verification phase of the filtering algorithms, an almost brute force method was applied (that is, the method of [4], without the incrementalization), using the same rotational sampling as was used in the FFT based method (sampling step $1/w$ radians, where w is the width of the template).

For the 3D experiments the test data was a reconstruction of *sus1* mutant of the *PRD1–bacteriophage* [9]. The size of the density (voxel) map was $119 \times 119 \times 119$ voxels. A pattern of size $11\times11\times11$ was extracted from the shell of the virus. The FFT CC method requires approximately 111 hours of computation time. The performance of the histogram method is reported in Figs. 6 and 7. It should be noted that the times could be substantially improved by filtering also for the rotation, as presented in Sec. 2.3.

Especially in 3D, our algorithms are superior to FFT based CC method for moderate κ. Note that the relative time gap between the methods should increase as $|V|$ and $|P|$ increase. Besides time, FFT also uses a lot of memory compared to our methods.

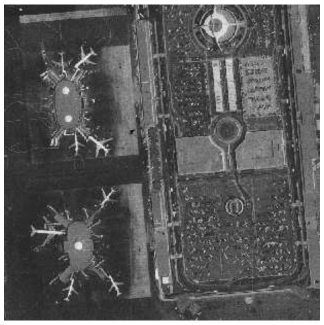

Fig. 1. Input image of size 768×768 pixels for the experiments. The pattern is one of the planes, and is of size 51×51 pixels.

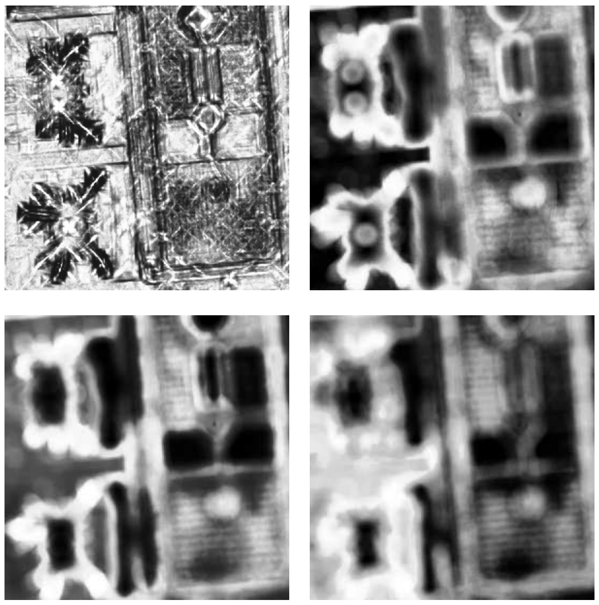

Fig. 2. Correlation images. From left to right, top to bottom: CC (FFT), CC (histogram), SAD, and Hamming distance. The histogram filter images were run using $r = 25$.

4. CONCLUSIONS

We have shown that the histogram filtering algorithms can be very effective in rotation invariant template matching, when one is interested in only "good enough" matches. This is especially true in 3D, where the FFT spends over 111 hours, whereas our method solves the problem in about an hour for $\kappa = 2000$, and in mere

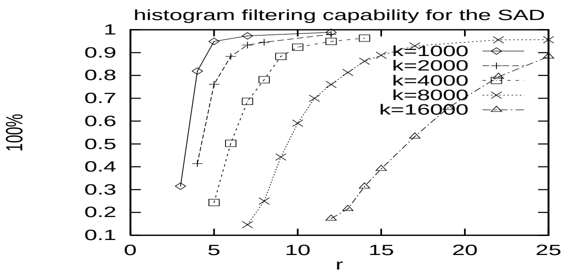

Fig. 3. Filtering capability for SAD in 2D. The x–axis is r, and the y–axis is the filtering capability. Different curves are for different κ.

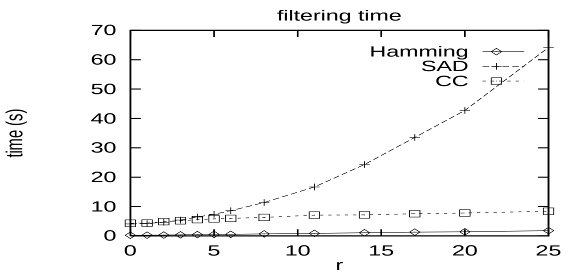

Fig. 4. Filtering times in 2D. The x–axis is r, and the y–axis is the running time. Different curves are for different distance measures.

minutes in small error levels. Unfortunately the effectiveness of the method drops fast when the parameter κ grows past a certain limit.

However, there is another way to look at our algorithms. Instead of filtering out unpromising positions of I or V, we may assume that our algorithms compute the approximate distance/similarity of P for each position of I or V. That is, we might define that P occurs at the position where the filter gives the highest peak of similarity. It is easy to find degenerate cases where this would give very bad results. However, despite of the flaws of this definition, in many applications this would give the desired result, and in that context the algorithms are very fast indeed. Our algorithms can be still improved, as explained in the text, but even now we believe that in many cases the old FFT based template matching should be substituted by our methods.

5. REFERENCES

[1] L. G. Brown, "A survey of image registration techniques," *ACM Comp. Surv.*, vol. 24, no. 4, pp. 325–376, Dec. 1992.

[2] T. M. Caelli and Z. Q. Liu, "On the minimum number of templates required for shift, rotation and size invariant pattern recognition," *Pattern Recognition*, vol. 21, pp. 205–216, 1988.

[3] K. Fredriksson, "Rotation invariant histogram filters for similarity and distance measures between digital images," in *SPIRE'2000*. 2000, pp. 105–115, IEEE CS Press.

[4] K. Fredriksson and E. Ukkonen, "Combinatorial methods for approximate pattern matching under rotations and translations in 3d arrays," in *SPIRE'2000*. 2000, pp. 96–104, IEEE CS Press.

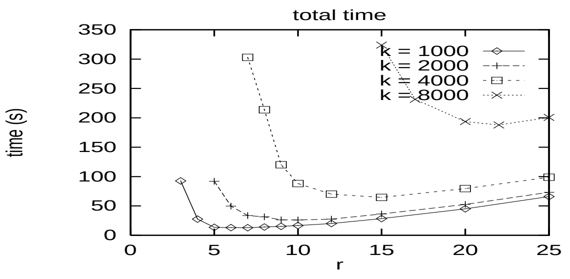

Fig. 5. Total running times for SAD in 2D. The x–axis is r, and the y–axis is the running time. Different curves are for different κ. FFT cross correlation takes 258.45 seconds.

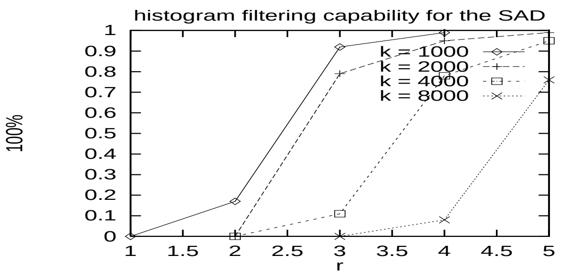

Fig. 6. Filtering capability for SAD in 3D. The x–axis is r, and the y–axis is the filtering capability. Different curves are for different κ.

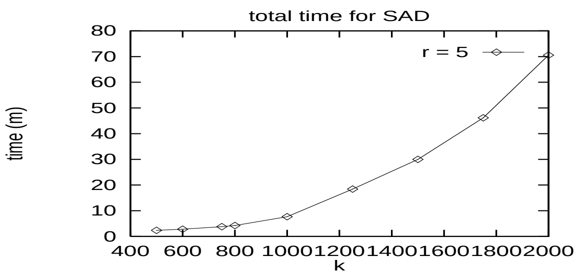

Fig. 7. Total running times in 3D, for SAD with $r = 5$. The x–axis is κ, and the y–axis is time in minutes. FFT cross correlation takes 111 *hours* for $r = 5$.

[5] K. Fredriksson and E. Ukkonen, "A rotation invariant filter for two–dimensional string matching," in *CPM'98*. 1998, vol. 1448 of *LNCS*, pp. 118–125, Springer-Verlag, Berlin.

[6] M. Werman, S. Peleg, and A. Rosenfeld, "A distance metric for multidimensional histograms," Tech. Rep. CAR-TR-90, University of Maryland, Center for Automation Research, 1984.

[7] R. Grossi and F. Luccio, "Simple and efficient string matching with k mismatches," *Information Processing Letters*, vol. 33, no. 3, pp. 113–120, 1989.

[8] M. Frigo and S. G. Johnson, "FFTW: An adaptive software architecture for the FFT," in *ICASSP'98*, 1998, vol. 3, pp. 1381–1384.

[9] S. J. Butcher, D. H. Bamford, and S. D. Fuller, "DNA packaging orders the membrane of bacteriophage PRD1," *EMBO Journal*, vol. 14, pp. 6078–6086, 1995.