Learning OpenCV

Gary Bradski and Adrian Kaehler

O'REILLY®
Beijing · Cambridge · Farnham · Köln · Sebastopol · Taipei · Tokyo

CHAPTER 6

Image Transforms

Overview

In the previous chapter we covered a lot of different things you could do with an image. The majority of the operators presented thus far are used to enhance, modify, or otherwise "process" one image into a similar but new image.

In this chapter we will look at *image transforms*, which are methods for changing an image into an alternate representation of the data entirely. Perhaps the most common example of a transform would be a something like a *Fourier transform*, in which the image is converted to an alternate representation of the data in the original image. The result of this operation is still stored in an OpenCV "image" structure, but the individual "pixels" in this new image represent spectral components of the original input rather than the spatial components we are used to thinking about.

There are a number of useful transforms that arise repeatedly in computer vision. OpenCV provides complete implementations of some of the more common ones as well as building blocks to help you implement your own image transforms.

Convolution

Convolution is the basis of many of the transformations that we discuss in this chapter. In the abstract, this term means something we do to every part of an image. In this sense, many of the operations we looked at in Chapter 5 can also be understood as special cases of the more general process of convolution. What a particular convolution "does" is determined by the form of the *Convolution kernel* being used. This kernel is essentially just a fixed size array of numerical coefficients along with an *anchor point* in that array, which is typically located at the center. The size of the array* is called the *support* of the kernel.

Figure 6-1 depicts a 3-by-3 convolution kernel with the anchor located at the center of the array. The value of the convolution at a particular point is computed by first placing

* For technical purists, the support of the kernel actually consists of only the nonzero portion of the kernel array.

the kernel anchor on top of a pixel on the image with the rest of the kernel overlaying the corresponding local pixels in the image. For each kernel point, we now have a value for the kernel at that point and a value for the image at the corresponding image point. We multiply these together and sum the result; this result is then placed in the resulting image at the location corresponding to the location of the anchor in the input image. This process is repeated for every point in the image by scanning the kernel over the entire image.

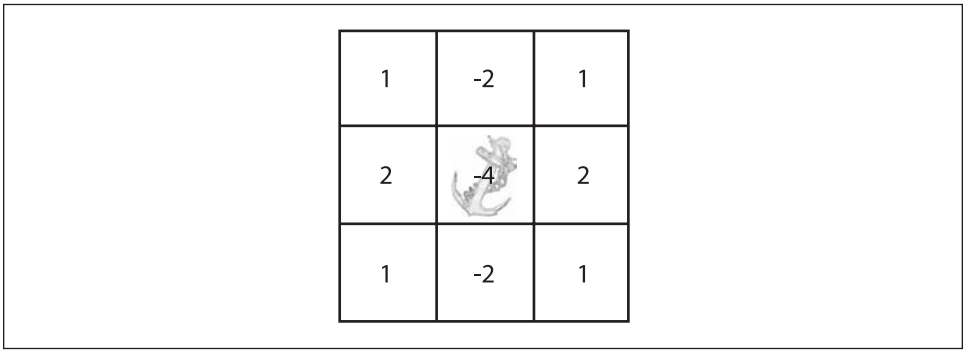

Figure 6-1. A 3-by-3 kernel for a Sobel derivative; note that the anchor point is in the center of the kernel

We can, of course, express this procedure in the form of an equation. If we define the image to be $I(x, y)$, the kernel to be $G(i, j)$ (where $0 < i < M_i - 1$ and $0 < j < M_j - 1$), and the anchor point to be located at (a_i, a_j) in the coordinates of the kernel, then the convolution $H(x, y)$ is defined by the following expression:

$$H(x,y) = \sum_{i=0}^{M_i-1} \sum_{j=0}^{M_j-1} I(x+i-a_i, y+j-a_j) G(i,j)$$

Observe that the number of operations, at least at first glance, seems to be the number of pixels in the image multiplied by the number of pixels in the kernel.* This can be a lot of computation and so is not something you want to do with some "for" loop and a lot of pointer de-referencing. In situations like this, it is better to let OpenCV do the work for you and take advantage of the optimizations already programmed into OpenCV. The OpenCV way to do this is with `cvFilter2D()`:

```
void cvFilter2D(
   const CvArr*     src,
   CvArr*           dst,
   const CvMat*     kernel,
```

* We say "at first glance" because it is also possible to perform convolutions in the frequency domain. In this case, for an N-by-N image and an M-by-M kernel with $N > M$, the computational time will be proportional to $N^2 \log(N)$ and not to the N^2M^2 that is expected for computations in the spatial domain. Because the frequency domain computation is independent of the size of the kernel, it is more efficient for large kernels. OpenCV automatically decides whether to do the convolution in the frequency domain based on the size of the kernel.

```
    CvPoint          anchor = cvPoint(-1,-1)
);
```

Here we create a matrix of the appropriate size, fill it with the coefficients, and then pass it together with the source and destination images into `cvFilter2D()`. We can also optionally pass in a `CvPoint` to indicate the location of the center of the kernel, but the default value (equal to `cvPoint(-1,-1)`) is interpreted as indicating the center of the kernel. The kernel can be of even size if its anchor point is defined; otherwise, it should be of odd size.

The `src` and `dst` images should be the same size. One might think that the `src` image should be larger than the `dst` image in order to allow for the extra width and length of the convolution kernel. But the sizes of the `src` and `dst` can be the same in OpenCV because, by default, prior to convolution OpenCV creates virtual pixels via replication past the border of the `src` image so that the border pixels in `dst` can be filled in. The replication is done as input($-dx$, y) = input(0, y), input($w + dx$, y) = input($w - 1$, y), and so forth. There are some alternatives to this default behavior; we will discuss them in the next section.

We remark that the coefficients of the convolution kernel should always be floating-point numbers. This means that you should use `CV_32FC1` when allocating that matrix.

Convolution Boundaries

One problem that naturally arises with convolutions is how to handle the boundaries. For example, when using the convolution kernel just described, what happens when the point being convolved is at the edge of the image? Most of OpenCV's built-in functions that make use of `cvFilter2D()` must handle this in one way or another. Similarly, when doing your own convolutions, you will need to know how to deal with this efficiently.

The solution comes in the form of the `cvCopyMakeBorder()` function, which copies a given image onto another slightly larger image and then automatically pads the boundary in one way or another:

```
void cvCopyMakeBorder(
    const CvArr*    src,
    CvArr*          dst,
    CvPoint         offset,
    int             bordertype,
    CvScalar        value     = cvScalarAll(0)
);
```

The `offset` argument tells `cvCopyMakeBorder()` where to place the copy of the original image within the destination image. Typically, if the kernel is N-by-N (for odd N) then you will want a boundary that is $(N - 1)/2$ wide on all sides or, equivalently, an image that is $N - 1$ wider and taller than the original. In this case you would set the offset to `cvPoint((N-1)/2,(N-1)/2)` so that the boundary would be even on all sides.*

* Of course, the case of N-by-N with N odd and the anchor located at the center is the simplest case. In general, if the kernel is N-by-M and the anchor is located at (a_x, a_y), then the destination image will have to be $N - 1$ pixels wider and $M - 1$ pixels taller than the source image. The offset will simply be (a_x, a_y).

The `bordertype` can be either `IPL_BORDER_CONSTANT` or `IPL_BORDER_REPLICATE` (see Figure 6-2). In the first case, the `value` argument will be interpreted as the value to which all pixels in the boundary should be set. In the second case, the row or column at the very edge of the original is replicated out to the edge of the larger image. Note that the border of the test pattern image is somewhat subtle (examine the upper right image in Figure 6-2); in the test pattern image, there's a one-pixel-wide dark border except where the circle patterns come near the border where it turns white. There are two other border types defined, `IPL_BORDER_REFLECT` and `IPL_BORDER_WRAP`, which are not implemented at this time in OpenCV but may be supported in the future.

Figure 6-2. Expanding the image border. The left column shows IPL_BORDER_CONSTANT where a zero value is used to fill out the borders. The right column shows IPL_BORDER_REPLICATE where the border pixels are replicated in the horizontal and vertical directions

We mentioned previously that, when you make calls to OpenCV library functions that employ convolution, those library functions call `cvCopyMakeBorder()` to get their work done. In most cases the border type called is `IPL_BORDER_REPLICATE`, but sometimes you will not want it to be done that way. This is another occasion where you might want to use `cvCopyMakeBorder()`. You can create a slightly larger image with the border you want, call whatever routine on that image, and then clip back out the part you were originally interested in. This way, OpenCV's automatic bordering will not affect the pixels you care about.

to the product. The transforms are the slowest* part of this operation; an N-by-N image takes $O(N^2 \log N)$ time and so the entire process is also completed in that time (assuming that $N > M$ for an M-by-M convolution kernel). This time is much faster than $O(N2M^2)$, the non-DFT convolution time required by the more naïve method.

Discrete Cosine Transform (DCT)

For real-valued data it is often sufficient to compute what is, in effect, only half of the discrete Fourier transform. The *discrete cosine transform* (DCT) [Ahmed74; Jain77] is defined analogously to the full DFT by the following formula:

$$c_k = \sum_{n=0}^{N-1} \sqrt{n = \begin{cases} \frac{1}{N} & \text{if } n = 0 \\ \frac{2}{N} & \text{else} \end{cases}} \cdot x_n \cdot \cos\left(-\pi \frac{(2k+1)n}{N} \right)$$

Observe that, by convention, the normalization factor is applied to both the cosine transform and its inverse. Of course, there is a similar transform for higher dimensions.

The basic ideas of the DFT apply also to the DCT, but now all the coefficients are real-valued. Astute readers might object that the cosine transform is being applied to a vector that is not a manifestly even function. However, with `cvDCT()` the algorithm simply treats the vector as if it were extended to negative indices in a mirrored manner.

The actual OpenCV call is:

```
void cvDCT(
  const CvArr* src,
  CvArr*       dst,
  int          flags
);
```

The `cvDCT()` function expects arguments like those for `cvDFT()` except that, because the results are real-valued, there is no need for any special packing of the result array (or of the input array in the case of an inverse transform). The `flags` argument can be set to `CV_DXT_FORWARD` or `CV_DXT_INVERSE`, and either may be combined with `CV_DXT_ROWS` with the same effect as with `cvDFT()`. Because of the different normalization convention, both the forward and inverse cosine transforms always contain their respective contribution to the overall normalization of the transform; hence `CV_DXT_SCALE` plays no role in `cvDCT`.

Integral Images

OpenCV allows you to calculate an integral image easily with the appropriately named `cvIntegral()` function. An *integral image* [Viola04] is a data structure that allows rapid

* By "slowest" we mean "asymptotically slowest"—in other words, that this portion of the algorithm takes the most time for very large N. This is an important distinction. In practice, as we saw in the earlier section on convolutions, it is not always optimal to pay the overhead for conversion to Fourier space. In general, when convolving with a small kernel it will not be worth the trouble to make this transformation.

summing of subregions. Such summations are useful in many applications; a notable one is the computation of *Haar wavelets*, which are used in some face recognition and similar algorithms.

```
void cvIntegral(
   const CvArr*  image,
   CvArr*        sum,
   CvArr*        sqsum      = NULL,
   CvArr*        tilted_sum = NULL
);
```

The arguments to `cvIntegral()` are the original image as well as pointers to destination images for the results. The argument `sum` is required; the others, `sqsum` and `tilted_sum`, may be provided if desired. (Actually, the arguments need not be images; they could be matrices, but in practice, they are usually images.) When the input image is 8-bit unsigned, the `sum` or `tilted_sum` may be 32-bit integer or floating-point arrays. For all other cases, the `sum` or `tilted_sum` must be floating-point valued (either 32- or 64-bit). The result "images" must always be floating-point. If the input image is of size *W*-by-*H*, then the output images must be of size (*W* + 1)-by-(*H* + 1).*

An integral image sum has the form:

$$\text{sum}(X,Y) = \sum_{x \le X} \sum_{y \le Y} \text{image}(x,y)$$

The optional `sqsum` image is the sum of squares:

$$\text{sum}(X,Y) = \sum_{x \le X} \sum_{y \le Y} (\text{image}(x,y))^2$$

and the `tilted_sum` is like the sum except that it is for the image rotated by 45 degrees:

$$\text{tilt\_sum}(X,Y) = \sum_{y \le Y} \sum_{\text{abs}(x-X) \le y} \text{image}(x,y)$$

Using these integral images, one may calculate sums, means, and standard deviations over arbitrary upright or "tilted" rectangular regions of the image. As a simple example, to sum over a simple rectangular region described by the corner points (*x*1, *y*1) and (*x*2, *y*2), where *x*2 > *x*1 and *y*2 > *y*1, we'd compute:

$$\sum_{x1 \le x \le x2} \sum_{y1 \le y \le y2} [\text{image}(x,y)]$$
$$= [\text{sum}(x2,y2) - \text{sum}(x1-1,y2) - \text{sum}(x2,y1-1) + \text{sum}(x1-1,y1-1)]$$

In this way, it is possible to do fast blurring, approximate gradients, compute means and standard deviations, and perform fast block correlations even for variable window sizes.

* This is because we need to put in a buffer of zero values along the *x*-axis and *y*-axis in order to make computation efficient.

To make this all a little more clear, consider the 7-by-5 image shown in Figure 6-18; the region is shown as a bar chart in which the height associated with the pixels represents the brightness of those pixel values. The same information is shown in Figure 6-19, numerically on the left and in integral form on the right. Integral images (I') are computed by going across rows, proceeding row by row using the previously computed integral image values together with the current raw image (I) pixel value $I(x, y)$ to calculate the next integral image value as follows:

$$I'(x,y)=I(x,y)+I'(x-1,y)+I'(x,y-1)-I'(x-1,y-1)$$

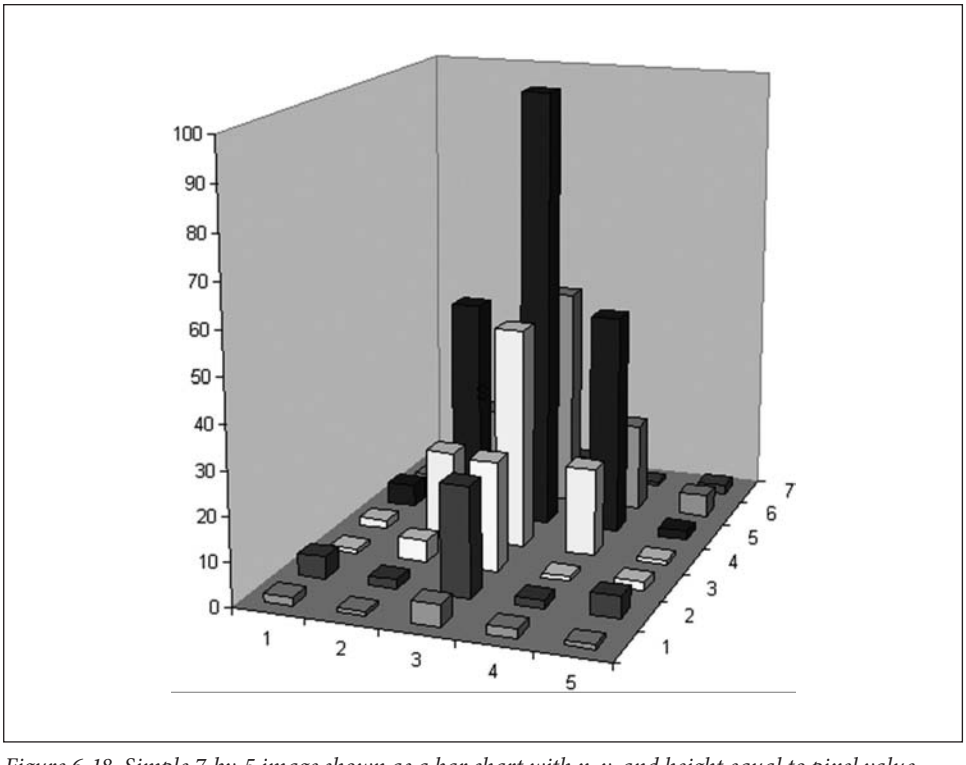

Figure 6-18. Simple 7-by-5 image shown as a bar chart with x, y, and height equal to pixel value

The last term is subtracted off because this value is double-counted when adding the second and third terms. You can verify that this works by testing some values in Figure 6-19.

When using the integral image to compute a region, we can see by Figure 6-19 that, in order to compute the central rectangular area bounded by the 20s in the original image, we'd calculate 398 – 9 – 10 + 1 = 380. Thus, a rectangle of any size can be computed using four measurements (resulting in $O(1)$ computational complexity).

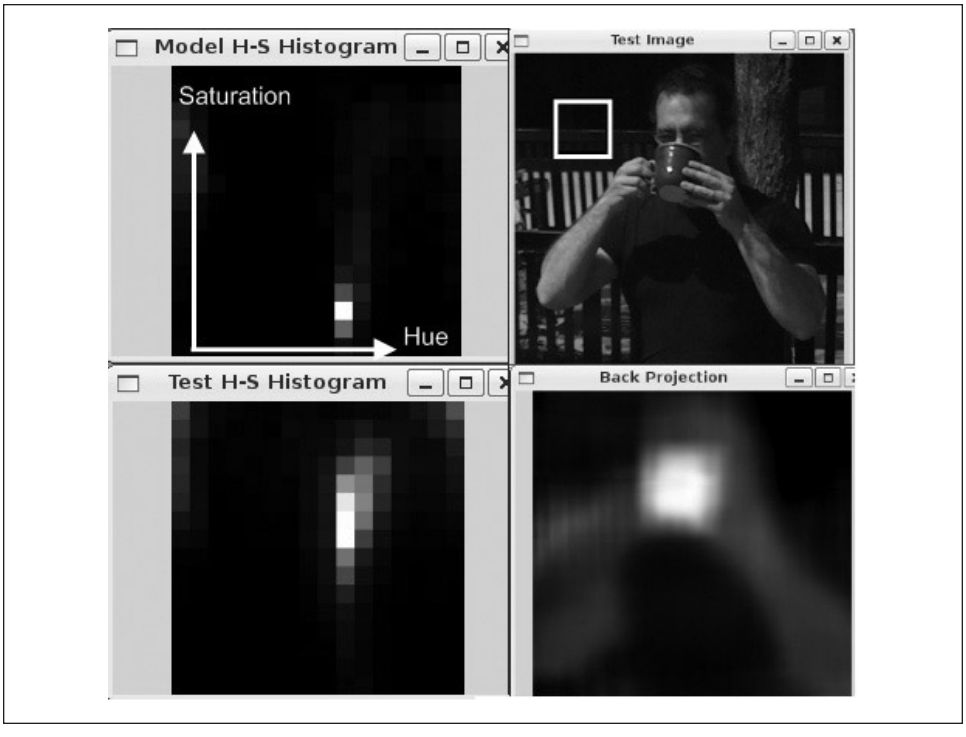

Figure 7-9. Using cvCalcBackProjectPatch() to locate an object (here, a coffee cup) whose size approximately matches the patch size (white box in upper right panel): the sought object is modeled by a hue-saturation histogram (upper left), which can be compared with an HS histogram for the image as a whole (lower left); the result of cvCalcBackProjectPatch() (lower right) is that the object is easily picked out from the scene by virtue of its color

Template Matching

Template matching via `cvMatchTemplate()` is not based on histograms; rather, the function matches an actual image patch against an input image by "sliding" the patch over the input image using one of the matching methods described in this section.

If, as in Figure 7-10, we have an image patch containing a face, then we can slide that face over an input image looking for strong matches that would indicate another face is present. The function call is similar to that of `cvCalcBackProjectPatch()`:

```
void cvMatchTemplate(
    const CvArr* image,
    const CvArr* templ,
    CvArr*       result,
    int          method
);
```

Instead of the array of input image planes that we saw in `cvCalcBackProjectPatch()`, here we have a single 8-bit or floating-point plane or color `image` as input. The matching model in `templ` is just a patch from a similar image containing the object for which

Figure 7-10. cvMatchTemplate() sweeps a template image patch across another image looking for matches

you are searching. The output object image will be put in the `result` image, which is a single-channel byte or floating-point image of size (`images->width - patch_size.x + 1`, `rimages->height - patch_size.y + 1`), as we saw previously in `cvCalcBackProjectPatch()`. The matching `method` is somewhat more complex, as we now explain. We use I to denote the input image, T the template, and R the result.

Square difference matching method (method = CV_TM_SQDIFF)

These methods match the squared difference, so a perfect match will be 0 and bad matches will be large:

$$R_{\text{sq\_diff}}(x,y) = \sum_{x',y'} [T(x',y') - I(x+x',y+y')]^2$$

Correlation matching methods (method = CV_TM_CCORR)

These methods multiplicatively match the template against the image, so a perfect match will be large and bad matches will be small or 0.

$$R_{\text{ccorr}}(x,y) = \sum_{x',y'} [T(x',y') \cdot I(x+x',y+y')]^2$$

Correlation coefficient matching methods (method = CV_TM_CCOEFF)

These methods match a template relative to its mean against the image relative to its mean, so a perfect match will be 1 and a perfect mismatch will be –1; a value of 0 simply means that there is no correlation (random alignments).

$$R_{\text{ccoeff}}(x,y)=\sum_{x',y'}[T'(x',y')\cdot I'(x+x',y+y')]^2$$

$$T'(x',y')=T(x',y')-\frac{1}{(w\cdot h)\sum_{x'',y''}T(x'',y'')}$$

$$I'(x+x',y+y')=I(x+x',y+y')-\frac{1}{(w\cdot h)\sum_{x'',y''}I(x+x'',y+y'')}$$

Normalized methods

For each of the three methods just described, there are also normalized versions first developed by Galton [Galton] as described by Rodgers [Rodgers88]. The normalized methods are useful because, as mentioned previously, they can help reduce the effects of lighting differences between the template and the image. In each case, the normalization coefficient is the same:

$$Z(x,y)=\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+x')^2}$$

The values for `method` that give the normalized computations are listed in Table 7-2.

Table 7-2. Values of the method parameter for normalized template matching

Value of method parameter	Computed result
`CV_TM_SQDIFF_NORMED`	$R_{\text{sq\_diff\_normed}}(x,y)=\frac{R_{\text{sq\_diff}}(x,y)}{Z(x,y)}$
`CV_TM_CCORR_NORMED`	$R_{\text{ccor\_normed}}(x,y)=\frac{R_{\text{ccor}}(x,y)}{Z(x,y)}$
`CV_TM_CCOEFF_NORMED`	$R_{\text{ccoeff\_normed}}(x,y)=\frac{R_{\text{ccoeff}}(x,y)}{Z(x,y)}$

As usual, we obtain more accurate matches (at the cost of more computations) as we move from simpler measures (square difference) to the more sophisticated ones (correlation coefficient). It's best to do some test trials of all these settings and then choose the one that best trades off accuracy for speed in your application.

Again, be careful when interpreting your results. The square-difference methods show best matches with a minimum, whereas the correlation and correlation-coefficient methods show best matches at maximum points.

As in the case of `cvCalcBackProjectPatch()`, once we use `cvMatchTemplate()` to obtain a matching `result` image we can then use `cvMinMaxLoc()` to find the location of the best match. Again, we want to ensure there's an area of good match around that point in order to avoid random template alignments that just happen to work well. A good match should have good matches nearby, because slight misalignments of the template shouldn't vary the results too much for real matches. Looking for the best matching "hill" can be done by slightly smoothing the `result` image before seeking the maximum (for correlation or correlation-coefficient) or minimum (for square-difference) matching methods. The morphological operators can also be helpful in this context.

Example 7-5 should give you a good idea of how the different template matching techniques behave. This program first reads in a template and image to be matched and then performs the matching via the methods we've discussed here.

Example 7-5. Template matching

```
// Template matching.
//   Usage: matchTemplate image template
//
#include <cv.h>
#include <cxcore.h>
#include <highgui.h>
#include <stdio.h>
int main( int argc, char** argv ) {
    IplImage *src, *templ,*ftmp[6]; //ftmp will hold results
    int i;
    if( argc == 3){
        //Read in the source image to be searched:
        if((src=cvLoadImage(argv[1], 1))== 0) {
            printf("Error on reading src image %s\n",argv[i]);
            return(-1);
        }
        //Read in the template to be used for matching:
        if((templ=cvLoadImage(argv[2], 1))== 0) {
            printf("Error on reading template %s\n",argv[2]);
                return(-1);
        }
        //ALLOCATE OUTPUT IMAGES:
        int iwidth = src->width - templ->width + 1;
        int iheight = src->height - templ->height + 1;
        for(i=0; i<6; ++i){
            ftmp[i] = cvCreateImage(
                            cvSize(iwidth,iheight),32,1);
        }
        //DO THE MATCHING OF THE TEMPLATE WITH THE IMAGE:
```

Example 7-5. Template matching (continued)

```
        for(i=0; i<6; ++i){
            cvMatchTemplate( src, templ, ftmp[i], i);
            cvNormalize(ftmp[i],ftmp[i],1,0,CV_MINMAX)*;
        }
        //DISPLAY
        cvNamedWindow( "Template", 0 );
        cvShowImage(   "Template", templ );
        cvNamedWindow( "Image", 0 );
        cvShowImage(   "Image", src );
        cvNamedWindow( "SQDIFF", 0 );
        cvShowImage(   "SQDIFF", ftmp[0] );
        cvNamedWindow( "SQDIFF_NORMED", 0 );
        cvShowImage(   "SQDIFF_NORMED", ftmp[1] );
        cvNamedWindow( "CCORR", 0 );
        cvShowImage(   "CCORR", ftmp[2] );
        cvNamedWindow( "CCORR_NORMED", 0 );
        cvShowImage(   "CCORR_NORMED", ftmp[3] );
        cvNamedWindow( "CCOEFF", 0 );
        cvShowImage(   "CCOEFF", ftmp[4] );
        cvNamedWindow( "CCOEFF_NORMED", 0 );
        cvShowImage(   "CCOEFF_NORMED", ftmp[5] );
        //LET USER VIEW RESULTS:
        cvWaitKey(0);
    }
    else { printf("Call should be: "
                  "matchTemplate image template \n");}
}
```

Note the use of `cvNormalize()` in this code, which allows us to display the results in a consistent way (recall that some of the matching methods can return negative-valued results. We use the `CV_MINMAX` flag when normalizing; this tells the function to shift and scale the floating-point images so that all returned values are between 0 and 1. Figure 7-11 shows the results of sweeping the face template over the source image (shown in Figure 7-10) using each of `cvMatchTemplate()`'s available matching methods. In outdoor imagery especially, it's almost always better to use one of the normalized methods. Among those, correlation coefficient gives the most clearly delineated match—but, as expected, at a greater computational cost. For a specific application, such as automatic parts inspection or tracking features in a video, you should try all the methods and find the speed and accuracy trade-off that best serves your needs.

* You can often get more pronounced match results by raising the matches to a power (e.g., `cvPow(ftmp[i], ftmp[i], 5);`). In the case of a result which is normalized between 0.0 and 1.0, then you can immediately see that a good match of 0.99 taken to the fifth power is not much reduced (0.99^5=0.95) while a poorer score of 0.20 is reduced substantially (0.50^5=0.03).

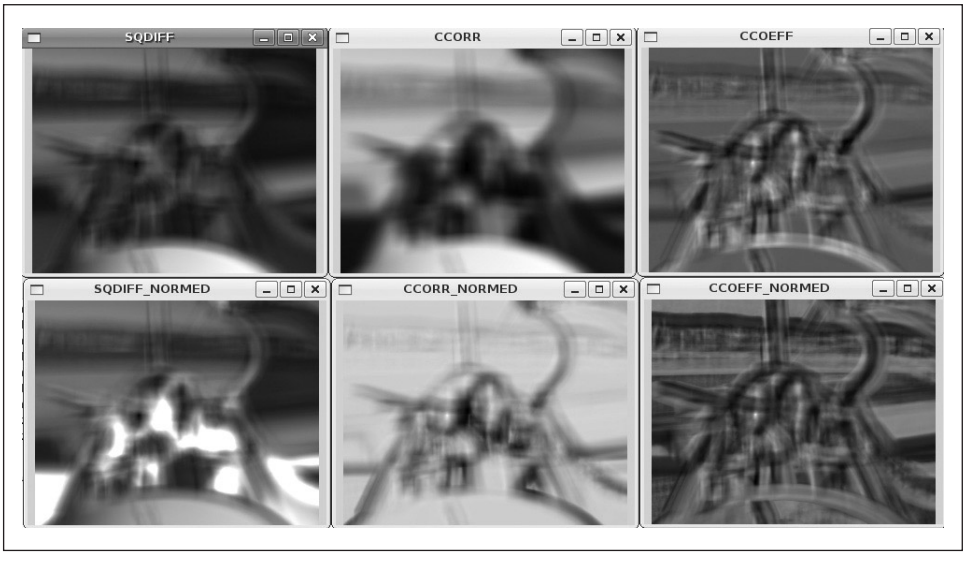

Figure 7-11. Match results of six matching methods for the template search depicted in Figure 7-10: the best match for square difference is 0 and for the other methods it's the maximum point; thus, matches are indicated by dark areas in the left column and by bright spots in the other two columns

Exercises

1. Generate 1,000 random numbers r_i between 0 and 1. Decide on a bin size and then take a histogram of $1/r_i$.
   a. Are there similar numbers of entries (i.e., within a factor of ±10) in each histogram bin?
   b. Propose a way of dealing with distributions that are highly nonlinear so that each bin has, within a factor of 10, the same amount of data.
2. Take three images of a hand in each of the three lighting conditions discussed in the text. Use `cvCalcHist()` to make an RGB histogram of the flesh color of one of the hands photographed indoors.
   a. Try using just a few large bins (e.g., 2 per dimension), a medium number of bins (16 per dimension) and many bins (256 per dimension). Then run a matching routine (using all histogram matching methods) against the other indoor lighting images of hands. Describe what you find.
   b. Now add 8 and then 32 bins per dimension and try matching across lighting conditions (train on indoor, test on outdoor). Describe the results.
3. As in exercise 2, gather RGB histograms of hand flesh color. Take one of the indoor histogram samples as your model and measure EMD (earth mover's distance) against the second indoor histogram and against the first outdoor shaded and first outdoor sunlit histograms. Use these measurements to set a distance threshold.

a. Using this EMD threshold, see how well you detect the flesh histogram of the third indoor histogram, the second outdoor shaded, and the second outdoor sunlit histograms. Report your results.

b. Take histograms of randomly chosen nonflesh background patches to see how well your EMD discriminates. Can it reject the background while matching the true flesh histograms?

4. Using your collection of hand images, design a histogram that can determine under which of the three lighting conditions a given image was captured. Toward this end, you should create features—perhaps sampling from parts of the whole scene, sampling brightness values, and/or sampling relative brightness (e.g., from top to bottom patches in the frame) or gradients from center to edges.

5. Assemble three histograms of flesh models from each of our three lighting conditions.

   a. Use the first histograms from indoor, outdoor shaded, and outdoor sunlit as your models. Test each one of these against the second images in each respective class to see how well the flesh-matching score works. Report matches.

   b. Use the "scene detector" you devised in part a, to create a "switching histogram" model. First use the scene detector to determine which histogram model to use: indoor, outdoor shaded, or outdoor sunlit. Then use the corresponding flesh model to accept or reject the second flesh patch under all three conditions. How well does this switching model work?

6. Create a flesh-region interest (or "attention") detector.

   a. Just indoors for now, use several samples of hand and face flesh to create an RGB histogram.

   b. Use `cvCalcBackProject()` to find areas of flesh.

   c. Use `cvErode()` from Chapter 5 to clean up noise and then `cvFloodFill()` (from the same chapter) to find large areas of flesh in an image. These are your "attention" regions.

7. Try some hand-gesture recognition. Photograph a hand about 2 feet from the camera, create some (nonmoving) hand gestures: thumb up, thumb left, thumb right.

   a. Using your attention detector from exercise 6, take image gradients in the area of detected flesh around the hand and create a histogram model for each of the three gestures. Also create a histogram of the face (if there's a face in the image) so that you'll have a (nongesture) model of that large flesh region. You might also take histograms of some similar but nongesture hand positions, just so they won't be confused with the actual gestures.

   b. Test for recognition using a webcam: use the flesh interest regions to find "potential hands"; take gradients in each flesh region; use histogram matching

above a threshold to detect the gesture. If two models are above threshold, take the better match as the winner.

   c. Move your hand 1–2 feet further back and see if the gradient histogram can still recognize the gestures. Report.

8. Repeat exercise 7 but with EMD for the matching. What happens to EMD as you move your hand back?

9. With the same images as before but with captured image patches instead of histograms of the flesh around the hand, use `cvMatchTemplate()` instead of histogram matching. What happens to template matching when you move your hand backwards so that its size is smaller in the image?

Bibliography

[Acharya05] T. Acharya and A. Ray, *Image Processing: Principles and Applications*, New York: Wiley, 2005.

[Adelson84] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA Engineer* 29 (1984): 33–41.

[Ahmed74] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers* 23 (1974): 90–93.

[Al-Haytham1038] I. al-Haytham, *Book of Optics*, circa 1038.

[AMI] Applied Minds, *http://www.appliedminds.com*.

[Antonisse82] H. J. Antonisse, "Image segmentation in pyramids," *Computer Graphics and Image Processing* 19 (1982): 367–383.

[Arfken85] G. Arfken, "Convolution theorem," in *Mathematical Methods for Physicists*, 3rd ed. (pp. 810–814), Orlando, FL: Academic Press, 1985.

[Bajaj97] C. L. Bajaj, V. Pascucci, and D. R. Schikore, "The contour spectrum," *Proceedings of IEEE Visualization 1997* (pp. 167–173), 1997.

[Ballard81] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition* 13 (1981): 111–122.

[Ballard82] D. Ballard and C. Brown, *Computer Vision*, Englewood Cliffs, NJ: Prentice-Hall, 1982.

[Bardyn84] J. J. Bardyn et al., "Une architecture VLSI pour un operateur de filtrage median," *Congres reconnaissance des formes et intelligence artificielle* (vol. 1, pp. 557–566), Paris, 25–27 January 1984.

[Bay06] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," *Proceedings of the Ninth European Conference on Computer Vision* (pp. 404–417), May 2006.

[Bayes1763] T. Bayes, "An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F.R.S. communicated by Mr. Price, in a letter to John

Canton, A.M.F.R.S.," *Philosophical Transactions, Giving Some Account of the Present Undertakings, Studies and Labours of the Ingenious in Many Considerable Parts of the World* 53 (1763): 370–418.

[Beauchemin95] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Computing Surveys* 27 (1995): 433–466.

[Belongie00] S. Belongie, J. Malik, and J. Puzicha, "Shape context: A new descriptor for shape matching and object recognition," NIPS 2000, Computer Vision Group, University of California, Berkeley, 2000.

[Berg01] A. C. Berg and J. Malik, "Geometric blur for template matching," *IEEE Conference on Computer Vision and Pattern Recognition* (vol. 1, pp. 607–614), Kauai, Hawaii, 2001.

[Bhattacharyya43] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by probability distributions," *Bulletin of the Calcutta Mathematical Society* 35 (1943): 99–109.

[Bishop07] C. M. Bishop, *Pattern Recognition and Machine Learning*, New York: Springer-Verlag, 2007.

[Black92] M. J. Black, "Robust incremental optical flow" (YALEU-DCS-RR-923), Ph.D. thesis, Department of Computer Science, Yale University, New Haven, CT, 1992.

[Black93] M. J. Black and P. Anandan, "A framework for the robust estimation of optical flow," *Fourth International Conference on Computer Vision* (pp. 231–236), May 1993.

[Black96] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Computer Vision and Image Understanding* 63 (1996): 75–104.

[Bobick96] A. Bobick and J. Davis, "Real-time recognition of activity using temporal templates," *IEEE Workshop on Applications of Computer Vision* (pp. 39–42), December 1996.

[Borgefors86] G. Borgefors, "Distance transformations in digital images," *Computer Vision, Graphics and Image Processing* 34 (1986): 344–371.

[Bosch07] A. Bosch, A. Zisserman, and X. Muñoz, "Image classification using random forests and ferns," *IEEE International Conference on Computer Vision*, Rio de Janeiro, October 2007.

[Bouguet] J.-Y. Bouguet, "Camera calibration toolbox for Matlab," retrieved June 2, 2008, from *http://www.vision.caltech.edu/bouguetj/calib_doc/index.html*.

[BouguetAlg] J.-Y. Bouguet, "The calibration toolbox for Matlab, example 5: Stereo rectification algorithm" (code and instructions only), *http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example5.html*.

[Bouguet04] J.-Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm," *http://robots.stanford.edu/cs223b04/algo_tracking.pdf*.

[Bracewell65] R. Bracewell, "Convolution" and "Two-dimensional convolution," in *The Fourier Transform and Its Applications* (pp. 25–50 and 243–244), New York: McGraw-Hill, 1965.

[Bradski00] G. Bradski and J. Davis, "Motion segmentation and pose recognition with motion history gradients," *IEEE Workshop on Applications of Computer Vision*, 2000.

[Bradski98a] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision*, October 1998.

[Bradski98b] G. R. Bradski, "Computer video face tracking for use in a perceptual user interface," *Intel Technology Journal* Q2 (1998): 705–740.

[Breiman01] L. Breiman, "Random forests," *Machine Learning* 45 (2001): 5–32.

[Breiman84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Monteray, CA: Wadsworth, 1984.

[Bresenham65] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal* 4 (1965): 25–30.

[Bronshtein97] I. N. Bronshtein, and K. A. Semendyayev, *Handbook of Mathematics*, 3rd ed., New York: Springer-Verlag, 1997.

[Brown66] D. C. Brown, "Decentering distortion of lenses," *Photogrammetric Engineering* 32 (1966): 444–462.

[Brown71] D. C. Brown, "Close-range camera calibration," *Photogrammetric Engineering* 37 (1971): 855–866.

[Burt81] P. J. Burt, T. H. Hong, and A. Rosenfeld, "Segmentation and estimation of image region properties through cooperative hierarchical computation," *IEEE Transactions on Systems, Man, and Cybernetics* 11 (1981): 802–809.

[Burt83] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Transactions on Communications* 31 (1983): 532–540.

[Canny86] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8 (1986): 679–714.

[Carpenter03] G. A. Carpenter and S. Grossberg, "Adaptive resonance theory," in M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*, 2nd ed. (pp. 87–90), Cambridge, MA: MIT Press, 2003.

[Carr04] H. Carr, J. Snoeyink, and M. van de Panne, "Progressive topological simplification using contour trees and local spatial measures," *15th Western Computer Graphics Symposium*, Big White, British Columbia, March 2004.

[Chen05] D. Chen and G. Zhang, "A new sub-pixel detector for x-corners in camera calibration targets," *WSCG Short Papers* (2005): 97–100.

[Chetverikov99] D. Chetverikov and Zs. Szabo, "A simple and efficient algorithm for detection of high curvature points in planar curves," *Proceedings of the 23rd Workshop of the Austrian Pattern Recognition Group* (pp. 175–184), 1999.

[Chu07] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, "Map-reduce for machine learning on multicore," *Proceedings of the Neural Information Processing Systems Conference* (vol. 19, pp. 304–310), 2007.

[Clarke98] T. A. Clarke and J. G. Fryer, "The Development of Camera Calibration Methods and Models," *Photogrammetric Record* 16 (1998): 51–66.

[Colombari07] A. Colombari, A. Fusiello, and V. Murino, "Video objects segmentation by robust background modeling," *International Conference on Image Analysis and Processing* (pp. 155–164), September 2007.

[Comaniciu99] D. Comaniciu and P. Meer, "Mean shift analysis and applications," *IEEE International Conference on Computer Vision* (vol. 2, p. 1197), 1999.

[Comaniciu03] D. Comaniciu, "Nonparametric information fusion for motion estimation," *IEEE Conference on Computer Vision and Pattern Recognition* (vol. 1, pp. 59–66), 2003.

[Conrady1919] A. Conrady, "Decentering lens systems," *Monthly Notices of the Royal Astronomical Society* 79 (1919): 384–390.

[Cooley65] J. W. Cooley and O. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation* 19 (1965): 297–301.

[Dahlkamp06] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski, "Self-supervised monocular road detection in desert terrain," *Robotics: Science and Systems*, Philadelphia, 2006.

[Dalai05] N. Dalai, and B. Triggs, "Histograms of oriented gradients for human detection," *Computer Vision and Pattern Recognition* (vol. 1, pp. 886–893), June 2005.

[Davis97] J. Davis and A. Bobick, "The representation and recognition of action using temporal templates" (Technical Report 402), MIT Media Lab, Cambridge, MA, 1997.

[Davis99] J. Davis and G. Bradski, "Real-time motion template gradients using Intel CVLib," *ICCV Workshop on Framerate Vision*, 1999.

[Delaunay34] B. Delaunay, "Sur la sphère vide," *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk* 7 (1934): 793–800.

[DeMenthon92] D. F. DeMenthon and L. S. Davis, "Model-based object pose in 25 lines of code," *Proceedings of the European Conference on Computer Vision* (pp. 335–343), 1992.

[Dempster77] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B* 39 (1977): 1–38.

[Det] "History of matrices and determinants," *http://www-history.mcs.st-and.ac.uk/history/HistTopics/Matrices_and_determinants.html*.

[Douglas73] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required for represent a digitized line or its caricature," *Canadian Cartographer* 10(1973): 112–122.

[Duda72] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the Association for Computing Machinery* 15 (1972): 11–15.

[Duda73] R. O. Duda and P. E. Hart, *Pattern Recognition and Scene Analysis*, New York: Wiley, 1973.

[Duda00] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, New York: Wiley, 2001.

[Farin04] D. Farin, P. H. N. de With, and W. Effelsberg, "Video-object segmentation using multi-sprite background subtraction," *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2004.

[Faugeras93] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*, Cambridge, MA: MIT Press, 1993.

[Fei-Fei98] L. Fei-Fei, R. Fergus, and P. Perona, "A Bayesian approach to unsupervised one-shot learning of object categories," *Proceedings of the Ninth International Conference on Computer Vision* (vol. 2, pp. 1134–1141), October 2003.

[Felzenszwalb63] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions" (Technical Report TR2004-1963), Department of Computing and Information Science, Cornell University, Ithaca, NY, 1963.

[FFmpeg] "Ffmpeg summary," *http://en.wikipedia.org/wiki/Ffmpeg*.

[Fischler81] M. A. Fischler and R. C. Bolles, "Random sample concensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the Association for Computing Machinery* 24 (1981): 381–395.

[Fitzgibbon95] A. W. Fitzgibbon and R. B. Fisher, "A buyer's guide to conic fitting," *Proceedings of the 5th British Machine Vision Conference* (pp. 513–522), Birmingham, 1995.

[Fix51] E. Fix, and J. L. Hodges, "Discriminatory analysis, nonparametric discrimination: Consistency properties" (Technical Report 4), USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[Forsyth03] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Englewood Cliffs, NJ: Prentice-Hall, 2003.

[FourCC] "FourCC summary," *http://en.wikipedia.org/wiki/Fourcc*.

[FourCC85] J. Morrison, "EA IFF 85 standard for interchange format files," *http://www.szonye.com/bradd/iff.html*.

[Fourier] "Joseph Fourier," *http://en.wikipedia.org/wiki/Joseph_Fourier*.

[Freeman67] H. Freeman, "On the classification of line-drawing data," *Models for the Perception of Speech and Visual Form* (pp. 408–412), 1967.

[Freeman95] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," *International Workshop on Automatic Face and Gesture Recognition* (pp. 296–301), June 1995.

[Freund97] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences* 55 (1997): 119–139.

[Fryer86] J. G. Fryer and D. C. Brown, "Lens distortion for close-range photogrammetry," *Photogrammetric Engineering and Remote Sensing* 52 (1986): 51–58.

[Fukunaga90] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Boston: Academic Press, 1990.

[Galton] "Francis Galton," *http://en.wikipedia.org/wiki/Francis_Galton*.

[GEMM] "Generalized matrix multiplication summary," *http://notvincenz.blogspot.com/2007/06/generalized-matrix-multiplication.html*.

[Göktürk01] S. B. Göktürk, J.-Y. Bouguet, and R. Grzeszczuk, "A data-driven model for monocular face tracking," *Proceedings of the IEEE International Conference on Computer Vision* (vol. 2, pp. 701–708), 2001.

[Göktürk02] S. B. Göktürk, J.-Y. Bouguet, C. Tomasi, and B. Girod, "Model-based face tracking for view-independent facial expression recognition," *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition* (pp. 287–293), May 2002.

[Grauman05] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," *Proceedings of the IEEE International Conference on Computer Vision*, October 2005.

[Grossberg87] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Science* 11 (1987): 23–63.

[Harris88] C. Harris and M. Stephens, "A combined corner and edge detector," *Proceedings of the 4th Alvey Vision Conference* (pp. 147–151), 1988.

[Hartley98] R. I. Hartley, "Theory and practice of projective rectification," *International Journal of Computer Vision* 35 (1998): 115–127.

[Hartley06] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge, UK: Cambridge University Press, 2006.

[Hastie01] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, New York: Springer-Verlag, 2001.

[Heckbert90] P. Heckbert, *A Seed Fill Algorithm* (Graphics Gems I), New York: Academic Press, 1990.

[Heikkila97] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition* (p. 1106), 1997.

[Hinterstoisser08] S. Hinterstoisser, S. Benhimane, V. Lepetit, P. Fua, and N. Navab, "Simultaneous recognition and homography extraction of local patches with a simple linear classifier," *British Machine Vision Conference*, Leeds, September 2008.

[Hinton06] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation* 18 (2006): 1527–1554.

[Ho95] T. K. Ho, "Random decision forest," *Proceedings of the 3rd International Conference on Document Analysis and Recognition* (pp. 278–282), August 1995.

[Homma85] K. Homma and E.-I. Takenaka, "An image processing method for feature extraction of space-occupying lesions," *Journal of Nuclear Medicine* 26 (1985): 1472–1477.

[Horn81] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence* 17 (1981): 185–203.

[Hough59] P. V. C. Hough, "Machine analysis of bubble chamber pictures," *Proceedings of the International Conference on High Energy Accelerators and Instrumentation* (pp. 554–556), 1959.

[Hu62] M. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory* 8 (1962): 179–187.

[Huang95] Y. Huang and X. H. Zhuang, "Motion-partitioned adaptive block matching for video compression," *International Conference on Image Processing* (vol. 1, p. 554), 1995.

[Iivarinen97] J. Iivarinen, M. Peura, J. Särelä, and A. Visa, "Comparison of combined shape descriptors for irregular objects," *8th British Machine Vision Conference*, 1997.

[Intel] Intel Corporation, *http://www.intel.com/*.

[Inui03] K. Inui, S. Kaneko, and S. Igarashi, "Robust line fitting using LmedS clustering," *Systems and Computers in Japan* 34 (2003): 92–100.

[IPL] Intel Image Processing Library (IPL), *www.cc.gatech.edu/dvfx/readings/iplman.pdf*.

[IPP] Intel Integrated Performance Primitives, *http://www.intel.com/cd/software/products/asmo-na/eng/219767.htm*.

[Isard98] M. Isard and A. Blake, "CONDENSATION: Conditional density propagation for visual tracking," *International Journal of Computer Vision* 29 (1998): 5–28.

[Jaehne95] B. Jaehne, *Digital Image Processing*, 3rd ed., Berlin: Springer-Verlag, 1995.

[Jaehne97] B. Jaehne, *Practical Handbook on Image Processing for Scientific Applications*, Boca Raton, FL: CRC Press, 1997.

[Jain77] A. Jain, "A fast Karhunen-Loeve transform for digital restoration of images degraded by white and colored noise," *IEEE Transactions on Computers* 26 (1997): 560–571.

[Jain86] A. Jain, *Fundamentals of Digital Image Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1986.

[Johnson84] D. H. Johnson, "Gauss and the history of the fast Fourier transform," *IEEE Acoustics, Speech, and Signal Processing Magazine* 1 (1984): 14–21.

[Kalman60] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering* 82 (1960): 35–45.

[Kim05] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-Time Imaging* 11 (2005): 167–256.

[Kimme75] C. Kimme, D. H. Ballard, and J. Sklansky, "Finding circles by an array of accumulators," *Communications of the Association for Computing Machinery* 18 (1975): 120–122.

[Kiryati91] N. Kiryati, Y. Eldar, and A. M. Bruckshtein, "A probablistic Hough transform," *Pattern Recognition* 24 (1991): 303–316.

[Konolige97] K. Konolige, "Small vision system: Hardware and implementation," *Proceedings of the International Symposium on Robotics Research* (pp. 111–116), Hayama, Japan, 1997.

[Kreveld97] M. van Kreveld, R. van Oostrum, C. L. Bajaj, V. Pascucci, and D. R. Schikore, "Contour trees and small seed sets for isosurface traversal," *Proceedings of the 13th ACM Symposium on Computational Geometry* (pp. 212–220), 1997.

[Lagrange1773] J. L. Lagrange, "Solutions analytiques de quelques problèmes sur les pyramides triangulaires," in *Oeuvres* (vol. 3), 1773.

[Laughlin81] S. B. Laughlin, "A simple coding procedure enhances a neuron's information capacity," *Zeitschrift für Naturforschung* 9/10 (1981): 910–912.

[LeCun98a] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE* 86 (1998): 2278–2324.

[LeCun98b] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient BackProp," in G. Orr and K. Muller (Eds.), *Neural Networks: Tricks of the Trade*, New York: Springer-Verlag, 1998.

[Lens] "Lens (optics)," *http://en.wikipedia.org/wiki/Lens_(optics)*.

[Liu07] Y. Z. Liu, H. X. Yao, W. Gao, X. L. Chen, and D. Zhao, "Nonparametric background generation," *Journal of Visual Communication and Image Representation* 18 (2007): 253–263.

[Lloyd57] S. Lloyd, "Least square quantization in PCM's" (Bell Telephone Laboratories Paper), 1957. ["Lloyd's algorithm" was later published in *IEEE Transactions on Information Theory* 28 (1982): 129–137.]

[Lowe04] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision* 60 (2004): 91–110.

[LTI] LTI-Lib, Vision Library, *http://ltilib.sourceforge.net/doc/homepage/index.shtml*.

[Lucas81] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proceedings of the 1981 DARPA Imaging Understanding Workshop* (pp. 121–130), 1981.

[Lucchese02] L. Lucchese and S. K. Mitra, "Using saddle points for subpixel feature detection in camera calibration targets," *Proceedings of the 2002 Asia Pacific Conference on Circuits and Systems* (pp. 191–195), December 2002.

[Mahal] "Mahalanobis summary," *http://en.wikipedia.org/wiki/Mahalanobis_distance*.

[Mahalanobis36] P. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Science* 12 (1936): 49–55.

[Manta] Manta Open Source Interactive Ray Tracer, *http://code.sci.utah.edu/Manta/index.php/Main_Page*.

[Maron61] M. E. Maron, "Automatic indexing: An experimental inquiry," *Journal of the Association for Computing Machinery* 8 (1961): 404–417.

[Marr82] D. Marr, *Vision*, San Francisco: Freeman, 1982.

[Martins99] F. C. M. Martins, B. R. Nickerson, V. Bostrom, and R. Hazra, "Implementation of a real-time foreground/background segmentation system on the Intel architecture," *IEEE International Conference on Computer Vision Frame Rate Workshop*, 1999.

[Matas00] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic Hough transform," *Computer Vision Image Understanding* 78 (2000): 119–137.

[Matas02] J. Matas, O. Chum, M. Urba, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," *Proceedings of the British Machine Vision Conference* (pp. 384–396), 2002.

[Meer91] P. Meer, D. Mintz, and A. Rosenfeld, "Robust regression methods for computer vision: A review," *International Journal of Computer Vision* 6 (1991): 59–70.

[Merwe00] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan, "The unscented particle filter," *Advances in Neural Information Processing Systems*, December 2000.

[Meyer78] F. Meyer, "Contrast feature extraction," in J.-L. Chermant (Ed.), *Quantitative Analysis of Microstructures in Material Sciences, Biology and Medicine* [Special issue of *Practical Metallography*], Stuttgart: Riederer, 1978.

[Meyer92] F. Meyer, "Color image segmentation," *Proceedings of the International Conference on Image Processing and Its Applications* (pp. 303–306), 1992.

[Mikolajczyk04] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (2004): 1615–1630.

[Minsky61] M. Minsky, "Steps toward artificial intelligence," *Proceedings of the Institute of Radio Engineers* 49 (1961): 8–30.

[Mokhtarian86] F. Mokhtarian and A. K. Mackworth, "Scale based description and recognition of planar curves and two-dimensional shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8 (1986): 34–43.

[Mokhtarian88] F. Mokhtarian, "Multi-scale description of space curves and three-dimensional objects," *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 298–303), 1988.

[Mori05] G. Mori, S. Belongie, and J. Malik, "Efficient shape matching using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (2005): 1832–1837.

[Morse53] P. M. Morse and H. Feshbach, "Fourier transforms," in *Methods of Theoretical Physics* (Part I, pp. 453–471), New York: McGraw-Hill, 1953.

[Neveu86] C. F. Neveu, C. R. Dyer, and R. T. Chin, "Two-dimensional object recognition using multiresolution models," *Computer Vision Graphics and Image Processing* 34 (1986): 52–65.

[Ng] A. Ng, "Advice for applying machine learning," *http://www.stanford.edu/class/cs229/materials/ML-advice.pdf*.

[Nistér06] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[O'Connor02] J. J. O'Connor and E. F. Robertson, "Light through the ages: Ancient Greece to Maxwell," *http://www-groups.dcs.st-and.ac.uk/~history/HistTopics/Light_1.html*.

[Oliva06] A. Oliva and A. Torralba, "Building the gist of a scene: The role of global image features in recognition visual perception," *Progress in Brain Research* 155 (2006): 23–36.

[Opelt08] A. Opelt, A. Pinz, and A. Zisserman, "Learning an alphabet of shape and appearance for multi-class object detection," *International Journal of Computer Vision* (2008).

[OpenCV] Open Source Computer Vision Library (OpenCV), *http://sourceforge.net/projects/opencvlibrary/*.

[OpenCV Wiki] Open Source Computer Vision Library Wiki, *http://opencvlibrary.sourceforge.net/*.

[OpenCV YahooGroups] OpenCV discussion group on Yahoo, *http://groups.yahoo.com/group/OpenCV*.

[Ozuysal07] M. Ozuysal, P. Fua, and V. Lepetit, "Fast keypoint recognition in ten lines of code," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[Papoulis62] A. Papoulis, *The Fourier Integral and Its Applications*, New York: McGraw-Hill, 1962.

[Pascucci02] V. Pascucci and K. Cole-McLaughlin, "Efficient computation of the topology of level sets," *Proceedings of IEEE Visualization 2002* (pp. 187–194), 2002.

[Pearson] "Karl Pearson," *http://en.wikipedia.org/wiki/Karl_Pearson*.

[Philbin07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[Pollefeys99a] M. Pollefeys, "Self-calibration and metric 3D reconstruction from uncalibrated image sequences," Ph.D. thesis, Katholieke Universiteit, Leuven, 1999.

[Pollefeys99b] M. Pollefeys, R. Koch, and L. V. Gool, "A simple and efficient rectification method for general motion," *Proceedings of the 7th IEEE Conference on Computer Vision*, 1999.

[Porter84] T. Porter and T. Duff, "Compositing digital images," *Computer Graphics* 18 (1984): 253–259.

[Prados05] E. Prados and O. Faugeras, "Shape from shading: A well-posed problem?" *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[Ranger07] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, "Evaluating mapreduce for multi-core and multiprocessor systems," *Proceedings of the 13th International Symposium on High-Performance Computer Architecture* (pp. 13–24), 2007.

[Reeb46] G. Reeb, "Sur les points singuliers d'une forme de Pfaff completement integrable ou d'une fonction numerique," *Comptes Rendus de l'Academie des Sciences de Paris* 222 (1946): 847–849.

[Rodgers88] J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *American Statistician* 42 (1988): 59–66.

[Rodrigues] "Olinde Rodrigues," *http://en.wikipedia.org/wiki/Benjamin_Olinde_Rodrigues*.

[Rosenfeld73] A. Rosenfeld and E. Johnston, "Angle detection on digital curves," *IEEE Transactions on Computers* 22 (1973): 875–878.

[Rosenfeld80] A. Rosenfeld, "Some Uses of Pyramids in Image Processing and Segmentation," *Proceedings of the DARPA Imaging Understanding Workshop* (pp. 112–120), 1980.

[Rousseeuw84] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American Statistical Association*, 79 (1984): 871–880.

[Rousseeuw87] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*, New York: Wiley, 1987.

[Rubner98a] Y. Rubner, C. Tomasi, and L. J. Guibas, "Metrics for distributions with applications to image databases," *Proceedings of the 1998 IEEE International Conference on Computer Vision* (pp. 59–66), Bombay, January 1998.

[Rubner98b] Y. Rubner and C. Tomasi, "Texture metrics," *Proceeding of the IEEE International Conference on Systems, Man, and Cybernetics* (pp. 4601–4607), San Diego, October 1998.

[Rubner00] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision* 40 (2000): 99–121.

[Rumelhart88] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in D. E. Rumelhart, J. L. McClelland, and PDP Research Group (Eds.), *Parallel Distributed Processing. Explorations in the Microstructures of Cognition* (vol. 1, pp. 318–362), Cambridge, MA: MIT Press, 1988.

[Russ02] J. C. Russ, *The Image Processing Handbook*, 4th ed. Boca Raton, FL: CRC Press, 2002.

[Scharr00] H. Scharr, "Optimal operators in digital image processing," Ph.D. thesis, Interdisciplinary Center for Scientific Computing, Ruprecht-Karls-Universität, Heidelberg, *http://www.fz-juelich.de/icg/icg-3/index.php?index=195*.

[Schiele96] B. Schiele and J. L. Crowley, "Object recognition using multidimensional receptive field histograms," *European Conference on Computer Vision* (vol. I, pp. 610–619), April 1996.

[Schmidt66] S. Schmidt, "Applications of state-space methods to navigation problems," in C. Leondes (Ed.), *Advances in Control Systems* (vol. 3, pp. 293–340), New York: Academic Press, 1966.

[Schroff08] F. Schroff, A. Criminisi, and A. Zisserman, "Object class segmentation using random forests," *Proceedings of the British Machine Vision Conference*, 2008.

[Schwartz80] E. L. Schwartz, "Computational anatomy and functional architecture of the striate cortex: A spatial mapping approach to perceptual coding," *Vision Research* 20 (1980): 645–669.

[Schwarz78] A. A. Schwarz and J. M. Soha, "Multidimensional histogram normalization contrast enhancement," *Proceedings of the Canadian Symposium on Remote Sensing* (pp. 86–93), 1978.

[Semple79] J. Semple and G. Kneebone, *Algebraic Projective Geometry*, Oxford, UK: Oxford University Press, 1979.

[Serra83] J. Serra, *Image Analysis and Mathematical Morphology*, New York: Academic Press, 1983.

[Sezgin04] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging* 13 (2004): 146–165.

[Shapiro02] L. G. Shapiro and G. C. Stockman, *Computer Vision*, Englewood Cliffs, NJ: Prentice-Hall, 2002.

[Sharon06] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt, "Hierarchy and adaptivity in segmenting visual scenes," *Nature* 442 (2006): 810–813.

[Shaw04] J. R. Shaw, "QuickFill: An efficient flood fill algorithm," *http://www.codeproject.com/gdi/QuickFill.asp*.

[Shi00] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000): 888–905.

[Shi94] J. Shi and C. Tomasi, "Good features to track," *9th IEEE Conference on Computer Vision and Pattern Recognition*, June 1994.

[Sivic08] J. Sivic, B. C. Russell, A. Zisserman, W. T. Freeman, and A. A. Efros, "Unsupervised discovery of visual object class hierarchies," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[Smith78] A. R. Smith. "Color gamut transform pairs," *Computer Graphics* 12 (1978): 12–19.

[Smith79] A. R. Smith, "Painting tutorial notes," Computer Graphics Laboratory, New York Institute of Technology, Islip, NY, 1979.

[Sobel73] I. Sobel and G. Feldman, "A 3 × 3 Isotropic Gradient Operator for Image Processing," in R. Duda and P. Hart (Eds.), *Pattern Classification and Scene Analysis* (pp. 271–272), New York: Wiley, 1973.

[Steinhaus56] H. Steinhaus, "Sur la division des corp materiels en parties," *Bulletin of the Polish Academy of Sciences and Mathematics* 4 (1956): 801–804.

[Sturm99] P. F. Sturm and S. J. Maybank, "On plane-based camera calibration: A general algorithm, singularities, applications," *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[Sun98] J. Sun and P. Perona, "Where is the sun?" *Nature Neuroscience* 1 (1998): 183–184.

[Suzuki85] S. Suzuki and K. Abe, "Topological structural analysis of digital binary images by border following," *Computer Vision, Graphics and Image Processing* 30 (1985): 32–46.

[SVD] "SVD summary," *http://en.wikipedia.org/wiki/Singular_value_decomposition*.

[Swain91] M. J. Swain and D. H. Ballard, "Color indexing," *International Journal of Computer Vision* 7 (1991): 11–32.

[Tanguay00] D. Tanguay, "Flying a Toy Plane," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (p. 2231), 2000.

[Teh89] C. H. Teh, R. T. Chin, "On the detection of dominant points on digital curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11 (1989): 859–872.

[Telea04] A. Telea, "An image inpainting technique based on the fast marching method," *Journal of Graphics Tools 9* (2004): 25–36.

[Thrun05] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics: Intelligent Robotics and Autonomus Agents*, Cambridge, MA: MIT Press, 2005.

[Thrun06] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. "Stanley, the robot that won the DARPA Grand Challenge," *Journal of Robotic Systems* 23 (2006): 661–692.

[Titchmarsh26] E. C. Titchmarsh, "The zeros of certain integral functions," *Proceedings of the London Mathematical Society* 25 (1926): 283–302.

[Tola08] E. Tola, V. Lepetit, and P. Fua, "A fast local descriptor for dense matching," *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, June 2008.

[Tomasi98] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *Sixth International Conference on Computer Vision* (pp. 839–846), New Delhi, 1998.

[Torralba07] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing visual features for multiclass and multiview object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (2007): 854–869.

[Torralba08] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large databases for recognition," *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, June 2008.

[Toyama99] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," *Proceedings of the 7th IEEE International Conference on Computer Vision* (pp. 255–261), 1999.

[Trace] "Matrix trace summary," *http://en.wikipedia.org/wiki/Trace_(linear_algebra).*

[Trucco98] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*, Englewood Cliffs, NJ: Prentice-Hall, 1998.

[Tsai87] R. Y. Tsai, "A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal of Robotics and Automation* 3 (1987): 323–344.

[Vandevenne04] L. Vandevenne, "Lode's computer graphics tutorial, flood fill," *http://student.kuleuven.be/~m0216922/CG/floodfill.html*.

[Vapnik95] V. Vapnik, *The Nature of Statistical Learning Theory*, New York: Springer-Verlag, 1995.

[Videre] Videre Design, "Stereo on a chip (STOC)," *http://www.videredesign.com/templates/stoc.htm*.

[Viola04] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision* 57 (2004): 137–154.

[VXL] VXL, Vision Library, *http://vxl.sourceforge.net/*.

[Welsh95] G. Welsh and G. Bishop, "An introduction to the Kalman filter" (Technical Report TR95-041), University of North Carolina, Chapel Hill, NC, 1995.

[Werbos74] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioural sciences," Ph.D. thesis, Economics Department, Harvard University, Cambridge, MA, 1974.

[WG] Willow Garage, *http://www.willowgarage.com*.

[Wharton71] W. Wharton and D. Howorth, *Principles of Television Reception*, London: Pitman, 1971.

[Xu96] G. Xu and Z. Zhang, *Epipolar Geometry in Stereo, Motion and Object Recognition*, Dordrecht: Kluwer, 1996

[Zhang96] Z. Zhang, "Parameter estimation techniques: A tutorial with application to conic fitting," *Image and Vision Computing* 15 (1996): 59–76.

[Zhang99] R. Zhang, P.-S. Tsi, J. E. Cryer, and M. Shah, "Shape form shading: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21 (1999): 690 –706.

[Zhang99] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," *Proceedings of the 7th International Conference on Computer Vision* (pp. 666–673), Corfu, September 1999.

[Zhang00] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000): 1330–1334.

[Zhang04] H. Zhang, "The optimality of naive Bayes," *Proceedings of the 17th International FLAIRS Conference*, 2004.