Chapter 14

Lecture 14: Image features

Monday, March 28, 2011 MIT EECS course 6.869, Bill Freeman and Antonio Torralba
Slide numbers refer to the file "14Features2011.key"

slide 2 Suppose we want to recognize this object from a labelled dataset. We could store some representation of the entire object, but we wouldn't expect that to match anything we had in our database– the odds that the entire object would match is too small. But we might expect that a local feature could match.

That motivates a local-features approach to object recognition. Here in this image of the dog, we might guess that we could use the image appearance near the cross to describe a feature that was already found in an image in our database.

There would actually be many other uses for such an image feature. In order to do stereo matching, or to make a mosaic, we would want to find the same feature over different poses, sizes, lighting conditions, or viewing conditions of the object.

There are two parts that we'd have to get right: (a) localize the regions of the image where we'll describe the features, and (b) describe the local region. We'd like for both steps to be invariant to many of the afore mentioned common image transformations.

So here's the same object under similar lighting conditions, but viewed from a different viewpoint. we'd like for our magic feature point finder (sometimes called a keypoint) to find the same position in the second image, and to come up with a similar description vector for it.

In this image, we've changed both the lighting conditions, and the object's pose (mostly orientation, here). We'd still like to come up with a similar image description for that region.

Finally, we can change the object entirely, but stay within the object class. We would love it if our feature detector were again to fire in the (corresponding) same part of the image, and to come up with a similar description of it.

These feature positions and image areas were marked by hand for this example of what an ideal feature detector would do. Let's see how close to this ideal behavior current feature localizers and descriptors work. The topic of today's lecture is to create such a feature.

slide 3 If we had something like I've described here, we could use it for all sorts of tasks. And, indeed, modern feature detectors and descriptors are used for: object recognition, 3d reconstruction, stereo reconstruction, building panoramas, image indexing and database retrieval, location recognition and robot navigation.

slides 4-10 So let's take a look at one of those applications: building an image panorama. Here are two images from a set that we want to fit together into a panaroma. What do we want to do? First, we need to operate on each image independently and find a collection of distinctive points. These are called feature points, or keypoints.

Then we need to describe the local image around each keypoint, so that we can map corresponding keypoints from one image to the other. Finally, with the mapped pairs, we can compute the transformation that warps one image into the other.

Note that for this example, we want our feature point detector to find precisely corresponding points in each image, in order to be able to line up one image on top of the other. For other applications of feature point detection, like object description, we need for the feature points to line up only to the extent that they then give similar descriptions of the local image region, not for metrically accurate registration.

slide 10 Slide 10 shows what you can build from this approach to building panaromas. Here's a very wide-angle panarama, constructed from many photographs, each warped and seamlessly blended together.

In this lecture, well seek to derive detectors and descriptors that work perfectly But in reality, they will never be 100% perfect. Next time, well see how we can use the large number of feature points we can extract to reject outliers

slide 11, 12

To reiterate the plan: we detect points independently in each image. We describe the local image appearance (in a way that will be independent of small distortions, or changes in lighting). The we find the best matches, in a robust way (we'll introduce a commonly used algorithm called RANSAC).

slide 13

What makes a good feature point? It should be distinctive and localizable. It shouldn't change appearance much from one image to the other.

Let's examine this cartoon picture of an isolated contour to see what regions might satisfy these idealized conditions. We'll derive what is called the Harris corner detector, which is widely used. We'll describe a second feature point localizer when we introduce scale invariance.

slides 14, 15 What type of local image structure will allow us to reliably localize a position? A flat region certainly offers no chance of that–under a local window, all flat regions look the same. Being on an edge helps us to tell one location from another, but only in one dimension, and in the other dimension, along the contour, everything still looks the same. But if we're at a corner, then any change in the position of the analysis box changes the local information significantly, and in these regions it should be possible for us to repeatably localize ourselves to some canonical position and define a feature location. We'll find a unique location within such a region by taking the local maximum of a scalar value that tells us "corner-ness".

slides 16-18 So let's instantiate that mathematically. We define a local window function, $w(x, y)$, which can be a square box, or a small Gaussian window. We want to examine how the image intensities within that local window change as we translate by a small amount, (u, v). We sum that squared change in image intensity over the local window to find the squared change, $E(u, v)$, as a function of the translation vector, (u, v):

$$E(u, v) = \sum_{x,y} w(x, y)[I(x + u, y + v) - I(x, y)]^2 \tag{14.1}$$

For small translations, (u, v), we can approximate the translated image intensities in a Taylor series expansion to first order in u and v. That describes the squared change in image intensity, or the "error" $E(u, v)$, by the quadratic form,

$$\begin{aligned} E(u,v) &\approx \sum_{x,y} w(x,y)[I(x,y) + uI_x + vI_y - I(x,y)]^2 && (14.2) \\ &= \sum_{x,y} w(x,y)[uI_x + vI_y]^2 && (14.3) \\ &= (u \;\; v) \sum_{x,y} w(x,y) \begin{pmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}, && (14.4) \end{aligned}$$

where $I_x = \frac{\partial I}{\partial x}$ and $I_y = \frac{\partial I}{\partial y}$.

slides 19 The matrix

$$M = \sum_{x,y} w(x,y) \begin{pmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{pmatrix} \qquad (14.5)$$

is often called the *structure tensor*, and it reveals properties of the local image intensities. It is a weighted average (with positive coefficients) of positive, semi-definite matries, so the structure tensor is positive, semi-definite. For a differential step in the direction of a unit vector $\begin{pmatrix} u \\ v \end{pmatrix}$, the error function $E(u, v)$ tells how much the squared image intensities change. The largest (smallest) eigenvector of M reveals in what direction the image intensities change most (least) quickly. In this approximation, considering only image first derivatives, the iso-contours of squared image patch differences form an ellipse in the translation offsets of the patch to the reference image. The long axis of that ellipse shows in what direction the image changes most slowly; the short axis shows where the image changes most quickly.

slides 20 Let's see how that all works out for real image data. Here's an image, and a little local patch of the image data. If we take that patch, translate it locally over the image, and plot the squared error in the image intensities, summed over the patch, we get the error surface shown below, as a function of the patch offsets as we move over the image.

Here, we're in a very textured region of the image. There's some assymmetry to it, so the error surface climbs more quickly as we move the patch vertically (to the right on the error surface plot) than horizontally. But here both eigenvalues of the image structure matrix, M, are large, and the summed squared image differences change quickly as you wiggle the patch. So this would be a good candidate patch for localizing an image feature, because we can tell one part of the image from another. (We still have to decide where, within this region of localizable intensities, we'll mark our feature point. We'll get to that in a bit.)

slide 21 To understand our image structure tensor, let's see how this error surface behaves for a different part of the image. Up here on the roof, the image looks 1-dimensional, and, indeed our error surface tells us that. The structure tensor, our quadratic approximation to this error surface, would be long and skinny. The largest eigenvector of M points perpendicular to this contour, revealing where the error surface rises most steeply. That eigenvalue will be much larger than the eigenvalue of the perpendicular eigenvector.

slide 22 Finally, let's look at the image in a flat region. The error surface rises very slowly as you move in any direction in offset u or v. The eigenvalues will be small (note the exaggerated vertical scale on this plot relative to the scale of the previous two plots).

slide 23 So we can look at the eigenvalues of the structure tensor, M, to characterize the local image region. When both eigenvalues are small, we're in a flat region. When one is small and the other large, we're in an edge-like region. When both eigenvalues are large, then we're in a textured, or corner-like region, such that the squared error surface rises steeply in all directions.

slide 24 Let's convert these observations into a measure of "cornerness",R , which should be large where both eigenvalues are large, and otherwise small. A very natural measure, to enforce just that, was proposed by Shi and Tomasi: let the cornerness R be the minimum eigenvalue of M.

For historical reasons, a different scalar derived from the structure tensor M is often used:

$$R = \text{det}M - k(\text{trace}M) \tag{14.6}$$

where det $= \lambda_1\lambda_2$ and trace$M = \lambda_1 + \lambda_2$, where λ_1 and λ_2 are the eigenvalues of M, and k is set to a value between 0.04 and 0.06.

slides 25, 26 Now we can use R to quantify our intuitive impressions of cornerness. Values of R above some threshold (thus, not corners or flat regions) are candidates. Then to put down our marker to say the feature goes here, we can put the feature at a local maximum of the cornerness measure R.

slide 27 Slide 27 summarizes the Harris corner detection algorithm.

slide 28 Let's see how this all looks in images. Here are two photos of the same refrigerator magnet ("cow"). The photos differ in both viewpoint and in lighting. Our goal in this processing: by independently examing each image, to identify a set of feature point locations in each image, where some or many of the found feature point locations correspond to the same physical position, as viewed separately by each image.

slide 29 Here is a pseudo-color illustration of the corner response, R (sorry that there's no scale indicating magnitudes. But I assume that redder is higher; the uniform cyan regions must be be zero, and blue and dark blue, which corresponds to edge regions, must be negative.)

slide 30, 31, 32 Taking points for $R > T$, where T is some threshold, gives this set of candidate feature positions. Then taking the local maximum of those R values gives the points shown in 31. 32 shows those detected feature points overlaid on the original image.

Let's look at which feature points are at the right spot in both images. Note that many of them are, despite the change in 3d position, and lighting, between the two images.

slide 33 Some comments on the Harris detector.

1. Note that we haven't verified that the local-max-of-R position is reliably repeatable between one image and another. The position depends on the R quantity, which is only indirectly related to the image

2. Note that if we reverse the image contrast, we find exactly the same local-max-of-R feature position. Is that the behaviour we want?

3. Note that, despite having a good corner detector, sometimes we mark locations that are unreliable features, anyway. For example, note the spuriously formed good feature shown here in red. Really, it's a bad feature, since it doesn't correspond to a physical quantity in the true image. At some point, we'll need to reject this otherwise good feature for image registration. We can do that either by using some high-level grouping process to alert us that this corner region is really a background

contour running into an occluding contour, and thus an unreliable feature. Or (what we'll do next lecture) by using a robust feature matching process, such as RANSAC.

slides 34-37 Let's evaluate how this algorithm does under some of the criteria we have set-up for ourselves: invariance to various geometrical and photometric image changes.

How about image rotation?

Yes, the detector is invariant to that. ie, it will detect the same points in a rotated copy of the image, modulo effects due to spatial discretization that might give a slightly different position for the spatial local maxima. The eigen analysis of the M matrix allows us to use a coordinate-invariant representation for R.

slides 38, 39 Cordelia Schmid and collaborators did some nice studies of feature properties. Here is an analysis of the repeatability of the Harris corner detector under rotations. There are two versions plotted: one is the ordinary Harris corner detector, and the second is an improved version of the algorithm, which uses Gaussian derivatives, less sensitive to sampling grid structures than the [-2 -1 0 1 2] filters they used in their Harris implementation.

slides 40, 41

Is the corner-ness measure invariant to over-all image intensity changes, for example $I(x, y) \rightarrow aI(x, y) + b$?.

Partially. Everything works fine, except that there is the threshold, T, in the algorithm. If we scale-up the image intensities, some new points will peek above the threshold value that weren't there before, changing the feature point detections.

slides 42-45 What about invariance to image scale?

No, depending on the scale of analysis, a curve could look like a straight line, or like a corner. The figure in slide 45 shows the results of an emperical study analyzing the repeatability rate for detecting a feature as a function of the scale variation of the image. Note the very steep fall-off in performance as you change the image scale by a factor of two.

slide 46 Let's see what needs to happen to achieve scale invariant detection. Here's a curve at two different scales, and the green circle at the right shows the characteristic size of some spatial function that operates on the image and returns a number. For example, that operator could be a band-pass filter.

We want to expand the scale of that operator, on the left, until it is scaled, relative to the right hand operator, as the left hand curve is scaled to the left hand operator. Then, if the operators are constructed to give an area-normalized response, the output of the left-hand filter will be the same as that of the right hand filter.

slide 47 But we need more than that. We don't have access to the right hand curve and filter when we are filtering the left hand curve. So we need to identify a characteristic scale that will yield the properly scaled operators when applied to each curve independently. How do we pick a characteristic scale?

slide 48 When localizing something in space, we applied some operator (for example, the Harris corner-ness operator) and looked for a local maximum over space. We can do the same thing over scale.

A natural filter for analyzing structure over scale is the Gaussian filter. Under the requirement that new structures never be created as one transitions from finer to coarser scales, one can show that the Gaussian is a canonical choice for the scale space operator.

`See, for example, http://www.csc.kth.se/~tony/abstracts/Lin08-EncCompSci.html`

Differences, or derivatives, of Gaussians is a natural class of filters to use to detect characteristic points in scale. An intuition for this is as follows: the difference of two different scales of a Gaussian tells what features appear as you transition over scale space. Finding the scale at which the most new image energy appears indicates a characteristic scale.

slide 49 Two functional forms are commonly applied to find such local maxima over scale. One is a difference of Gaussians,

$$\mathrm{DoG} = G(x, y, k\sigma) - G(x, y, \sigma) \tag{14.7}$$

where

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2+y^2}{2\sigma^2}} \tag{14.8}$$

Another commonly used operator to find local maxima over scale is a scale-normalized Laplacian of a Gaussian,

$$L = \sigma^2 (G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma)) \tag{14.9}$$

These operators are similar; the figure plots one on top of the other.

slides 50-52 So here are two scale invariant detectors, used by two different groups: apply a Harris corner detector in space, and look for a local maximum there. Then apply a Laplacian of a Gaussian over scale, and look for a maximum there. You can see how, in the photograph of slide 51, the scale-normalized Laplacian of Gaussian operator produces a maximal response at the same relative scale for two different sizes of the same photograph of a white building. The difference of Gaussians operator behaves similarly, and is used by David Lowe in the SIFT feature keypoint detector.

slide 53-58 Slides 53-58 show, in a 1-d example, the difference of Gaussians filter applied to bumps of 3 different sizes. First slide 53 shows the 1-d bumps as they appear in a "scale space", where we blur the images with Gaussians of increasing width. To look at changes over scale space, we look at differences of Gaussians, over scale, in slides 54 and 55. (This is a continuous version of the Laplacian pyramid, without the spatial subsampling of the pyramid).

Slides 55-58 show visually how the difference of Gaussian filter, applied over scale, does indeed pull-out a characteristic scale for each of the bumps, capturing the size at which the characteristic feature appears. Slides 57 and 58 verify that the scale of the peak responses are indeed proportional to the width.

slide 59 The photograph in slide 51, and the experiments above, both have a single, high-contrast blob, and each of the scale invariant filters behaves in close to the idealized way. For more general images, the the behavior is less perfectly scale-invariant. This plot, from the same ICCV paper by Mikolajczyk and Schmid as the slide 51 figure, shows the performance drop-off for other images. But nonetheless, the performance is much better than without this attempt to find a scale-invariant feature.

slide 60 Of course, we must make a choice of how many scales to sample over. Here is a plot showing repeatability performance as a function of the number of scales sampled per octave. The performance is evaluated for two different measures: finding the matching location and scale in a second image, and finding the correct scale such that you can then correctly finding the nearest descriptor in a database. The performance for each hits a maximum when 3 scales per octave are sampled.

slide 61 For some applications, for example, forming an image panorama, it is important to localize the keypoints to sub-pixel and sub-scale accuracy. That can be done by fitting the difference of Gaussian function to a 2nd order polynomial using a Taylor series expansion.

slides 62, 63

Figure 63 shows example keypoint detections, with significant changes in both scale and orientation. Yet, for this example, using the Harris detector over space and Laplacian of Gaussian over scale, the corresponding keypoints are reliably found in the two different images.

slides 64-66 Now let's move to the task of image description. We want a local image descriptor that is insensitive to changes in the image caused by lighting and viewpoint changes, or small changes in shape. (In other words, we want an image descriptor that is insensitive to aspects of the image that we are insensitive to). That requirement motivates the choices for the most popular image descriptors.

slides 67, 68

We want a representation which is somewhat invariant to lighting changes. Here are two pictures of a hand, taken under different lighting conditions. In a pixel intensity representation, the two images look quite different. But if we display them, instead, as orientation maps, showing the dominant orientation at each position, then the two images look rather similar.

Indeed, an isolated cylinder, under many different lighting conditions, will have the same image orientation structure, while it may have very different image intensities.

Also note that, in some circumstances, a histogram of the image orientations is a sufficiently rich description to uniquely specify the underlying image. Here are images from a hand-gesture recognition system that I made, based on image histograms. We found the dominant orientation at each pixel location (second row), calculated the histogram over all orientations (shown in polar coordinates in the third row), and used those as feature vectors to distinguish one hand signal from another.

slides 69-72 One of the most influential papers in computer vision put these various ideas together to make a practical feature descriptor, called SIFT, for scale invariant feature transform. Here are the steps:

1. Find the keypoint scale, position, and orientation.
2. Use that to compute a scaled and rotated version of the local image.
3. Compute a 4x4 array of gradient orientation histograms. Not really a histogram–weigh them by their gradient magnitude. Weigh them in space by a Gaussian.
4. Make a 4x4 array of the gradient histograms. Eight orientations per histogram, over a 4x4 array, gives a 128 dimensional image descriptor. This allows for some sensitivity to the spatial position of image details, but not too much.
5. Normalize the 128-dim vector to have a length of 1. Then clamp those normalized feature vector values so that no entry is larger than 0.2. Then re-normalize the 128-dim vector to again have a length of 1. This will make the gradient histogram be more like an orientation histogram.

slides 73-78 So that's how the features are designed. Let's evaluate how they perform. In the IJCV, 2004 paper, "Distinctive image features from scale-invariant keypoints", David Lowe evaluated the effectiveness of the features under different design parameters and for various image distortion characteristics. The next several slides study these properties of the feature descriptors:

- Performance as a function of the number of histogram orientations and spatial subregions. These plots show that a 4x4 grid of gradient histograms, of 16 orientations per histogram, gives the best results.

- The resulting detector is relatively robust against noise (this evaluates both the keypoint detector as well as the descriptor, since both have to perform well to correctly match a databse feature).
- These features weren't explicitly designed to invariant to affine distortions, but in practise they seem to be. In slide 77, Lowe describes how you could get even better robustness to affine distortion by storing additional feature descriptors from affine distorted versions of the image region. The figure below the text shows still another way: by pre-warping each feature to have a circular measure of local image structure. Then you'd need to use a rotationally invariant descriptor, which SIFT is, if the image patch is rotated to the local dominant orientation.
- The 128 dimensional features are distinctive enough that at least up to 100,000 feature vectors can be used in the database without a significant loss of correct match performance.

slides 79-81 The resulting features do a spectacular job of "instance matching"–finding the same object in a database through identifying characteristic textures seen on the object.

slide 82 This work has had a huge impact on the field. This paper has single-handedly given IJCV a very high "impact factor".

slide 83 Next lecture, we'll examine one use of sift features: to find matching points in a sequence of photographs in order to stitch them together to form a panorama.