Lecture 7: Correspondence Matching

Reading: T&V Section 7.2

Recall: Derivative of Gaussian Filter

\[
\begin{align*}
&I_x = \frac{dI(x,y)}{dx} \\
&I_y = \frac{dI(x,y)}{dy}
\end{align*}
\]

Convolve

Observe and Generalize

Derivative of Gaussian

Looks like vertical and horizontal step edges

Key idea: Convolution (and cross correlation) with a filter can be viewed as comparing a little “picture” of what you want to find against all local regions in the image.

Observe and Generalize

Key idea: Cross correlation with a filter can be viewed as comparing a little “picture” of what you want to find against all local regions in the image.

Maximum response: vertical edge; lighter on right

Minimum response: vertical edge; lighter on left

Template Matching

What if we cut little pictures out from an image, then tried convolve them with the same or other images?
Confession

I’ve cheated a little bit.

I subtracted the mean greyvalue from both the image and the template before doing cross correlation.

Why? (we will discuss later, but think about it first)

The Correspondence Problem

• Basic assumptions:
  – Most scene points are visible in both images
  – Corresponding image regions are similar
• These assumptions hold if:
  – The distance of points from the cameras is much larger than the distance between cameras

• Is a “search” problem:
  – Given an element in the left image, search for the corresponding element in the right image.
  – We will typically need geometric constraints to reduce the size of the search space
• We must choose:
  – Elements to match
  – A similarity measure to compare elements
Correspondence Problem

- Two classes of algorithms:
  - Correlation-based algorithms
    - Produce a DENSE set of correspondences
  - Feature-based algorithms
    - Produce a SPARSE set of correspondences

Correlation-based Algorithms

Elements to be matched are image patches of fixed size

Task: what is the corresponding patch in a second image?

Correlation-based Algorithms

Task: what is the corresponding patch in a second image?

1) Need an appearance similarity function.

2) Need a search strategy to find location with highest similarity. Simplest (but least efficient) approach is exhaustive search.

Comparing Windows:

Some possible measures:

\[
\begin{align*}
SSD &= \sum_{[i,j] \in R} (f(i,j) - g(i,j))^2 \\
C_{fg} &= \sum_{[i,j] \in R} f(i,j)g(i,j)
\end{align*}
\]

Most popular

Correlation \[ C_{fg} \]

\[
C_{fg} = \sum_{[i,j] \in R} f(i,j)g(i,j)
\]

If we are doing exhaustive search over all image patches in the second image, this becomes cross-correlation of a template with an image. We have seen this before – imfilter(im,template,’corr’).

Example

Note: this is a stereo pair from the NASA mars rover. The rover is exploring the “El Capitan” formation.
Example: Cross-correlation

Note that the score image looks a lot like a blurry version of image 2.

This clues us in to the problem with straight correlation with an image template.

Problem with Correlation of Raw Image Templates

Consider correlation of template with an image of constant grey value:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>g</td>
<td>h</td>
<td>i</td>
</tr>
</tbody>
</table>

Result: \( v \times (a+b+c+d+e+f+g+h+i) \)

Problem with Correlation of Raw Image Templates

Now consider correlation with a constant image that is twice as bright:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>g</td>
<td>h</td>
<td>i</td>
</tr>
</tbody>
</table>

Result: \( 2\times v \times (a+b+c+d+e+f+g+h+i) \) > \( v \times (a+b+c+d+e+f+g+h+i) \)

Larger score, regardless of what the template is!

Solution

Subtract off the mean value of the template.

In this way, the correlation score is higher only when darker parts of the template overlap darker parts of the image, and brighter parts of the template overlap brighter parts of the image.
Correlation, zero-mean template

Better! But highest score is still not the correct match. Note: highest score IS best within local neighborhood of correct match.

“SSD” or “block matching” (Sum of Squared Differences)

$$\sum_{[i,j] \in R} (f(i,j) - g(i,j))^2$$

1) The most popular matching score.
2) We used it when deriving Harris corners
3) T&V claim it works better than cross-correlation

Relation between SSD and Correlation

$$SSD = \sum_{[i,j] \in R} (f - g)^2$$

$$= \sum_{[i,j] \in R} f^2 + \sum_{[i,j] \in R} g^2 - 2 \left( \sum_{[i,j] \in R} fg \right)$$

$$C_{fg} = \sum_{[i,j] \in R} f(i,j)g(i,j)$$

Correlation!

Handling Intensity Changes

Intensity Changes:
- the camera taking the second image might have different intensity response characteristics than the camera taking the first image
- illumination in the scene could change
- The camera might have auto-gain control set, so that it’s response changes as it moves through the scene.

Intensity Normalization

- When a scene is imaged by different sensors, or under different illumination intensities, both the SSD and the $C_{fg}$ can be large for windows representing the same area in the scene!
- A solution is to normalize the pixels in the windows before comparing them by subtracting the mean of the patch intensities and dividing by the std.dev.

$$f' = \frac{f - \bar{f}}{\sqrt{\sum(f-f)^2}}$$

$$g' = \frac{g - \bar{g}}{\sqrt{\sum(g-g)^2}}$$
Normalized Cross Correlation

\[ \hat{f} = \frac{f - \bar{f}}{\sqrt{\Sigma(f - \bar{f})^2}} \quad \hat{g} = \frac{g - \bar{g}}{\sqrt{\Sigma(g - \bar{g})^2}} \]

\[ \text{NCC}(f, g) = C_{fg} = \sum_{(i,j)\in R} \hat{f}(i,j)\hat{g}(i,j) \]

Highest score also coincides with correct match. Also, looks like less chances of getting a wrong match.

Important point about NCC:
- Score values range from 1 (perfect match) to -1 (completely anti-correlated)

Intuition: treating the normalized patches as vectors, we see they are unit vectors. Therefore, correlation becomes dot product of unit vectors, and thus must range between -1 and 1.