Simple features

Morten R. Hannemose, mohan@dtu.dk

March 11, 2022

02504 Computer vision course lectures,
DTU Compute, Kgs. Lyngby 2800, Denmark

Learning objectives

After this lecture you should be able to:

- explain the image correspondence problem
- explain the use of image features (key points, interest points)
- understand and implement image filtering
- implement Harris corner detection
- run Canny edge detection

Presentation topics

- Image correspondence problem
  - Features and feature descriptors
- Simple features
  - Filtering and convolution
  - Harris corners
  - Canny edges

Image correspondence problem

Image correspondence

The problem of matching two parts of different images:

Image correspondence

More images of the same scene

- Correspondence exists between parts that are visible in each image
- Not all corresponding points have a unique pattern
- Idea: Only choose points that can be identified uniquely

What are good points?

What are good points?

We are able to identify the same corners in multiple images

Problems

Movement of camera

- Scale – distance of camera to object
- Rotation – objects and camera is rotated between frames
- Translation – movement of camera from one place to another

Result: Appearance change dependent on distance to camera

Problems

Movement of camera

Problems

Other issues

- Occlusion – objects can be in front of other objects
- Lighting change – darker/lighter, color of light (change in spectrum), direction of light
- Clutter – many objects in a scene

Solution

Invariance to imaging problems

- Focus on points and a small area around each point
- Two aspects
  - identify key points (focus of today)
  - characterize pattern around point (week 8)

Features and feature descriptors

First some terminology:

Key points, interest points, and feature points are the anem – namely points in an image with a coordinate position (also between pixels).

Descriptors, key point descriptors, interest point descriptors, and feature descriptors are the same and used for characterizing the pattern around a point. Usually a vector that can be matched between images.

The image patch feature descriptor

One easy way of describing a feature is to use the local pixel information around the feature.

The image patch matching

Stretch the image patch into a vector $\boldsymbol{f}$, and that is your descriptor.

A simple comparison operator $d(\boldsymbol{f}_1, \boldsymbol{f}_2)$ just uses the scalar product as the distance between features:

$$d(\boldsymbol{f}_1, \boldsymbol{f}_2) = \boldsymbol{f}_1^\top \boldsymbol{f}_2$$

Good with only small transformations, low noise, and unchanged environment.

Cross correlation

A better comparison $X(\boldsymbol{f}_1, \boldsymbol{f}_2)$ uses the cross correlation as the distance:

$$X(\boldsymbol{f}_1, \boldsymbol{f}_2) = \frac{\mathrm{cov}(\boldsymbol{f}_1, \boldsymbol{f}_2)}{\sqrt{\mathrm{var}(\boldsymbol{f}_1) + \mathrm{var}(\boldsymbol{f}_2)}}, \text{ where}$$

$$\mathrm{cov}(\boldsymbol{p}, \boldsymbol{q}) = \frac{1}{N-1} \sum_i^N (\boldsymbol{p}_i - \overline{\boldsymbol{p}})^\mathsf{T} (\boldsymbol{q}_i - \overline{\boldsymbol{q}}), \text{ and}$$

$$\mathrm{var}(\boldsymbol{p}) = \frac{1}{N-1} \sum_i^N ||\boldsymbol{p}_i - \overline{\boldsymbol{p}}||^2.$$

Cross correlation

- The cross correlation picks up local variation, and is therefore more robust against environment changes than squared distance
- Only useful when images are extremely similar.

More complicated feature descriptors will follow in two weeks.

Simple features

Filtering and convolution

$$I(x;t) = \int_{\xi \in \mathbb{R}^2} I(x-\xi) g(\xi;t) d\xi \approx \sum_{i=-k}^{k} \sum_{j=-k}^{k} I(x-i, y-j) g(i,j)$$

2D convolution

We use filtering and convolution for the same operation and use the symbol $*$ for convolution. Convolution is commutative

$$I_g = g * I = I * g$$

2D Gaussian $g : \mathbb{R}^2 \times \mathbb{R}_+ \to \mathbb{R}$

$$g(x, y; \sigma^2) = \frac{1}{2\pi\sigma^2} \exp\left(\frac{-(x^2 + y^2)}{2\sigma^2}\right). \tag{1}$$

Convolution – separability

Useful property: The (isotropic) Gaussian is separable

$$g(x, y; \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-x^2}{2\sigma^2}\right) \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-y^2}{2\sigma^2}\right)$$

This separability means that

$$I_g = \left(\mathbf{g} * \mathbf{g}^\mathsf{T}\right) * I = \mathbf{g} * \left(\mathbf{g}^\mathsf{T} * I\right)$$

where $\mathbf{g}$ is a vector of the Gaussian.

Size of Gaussian filter

- Ideally: Infinitely big – but this is impractical
- Empiric rule: 3σ, 4σ or 5σ rule – example:
  - if $\sigma = 2$, filter size of Gaussian is $2 \cdot 5 \cdot 2 + 1 = 21$
  - if $\sigma = 20$, filter size of Gaussian is $2 \cdot 5 \cdot 20 + 1 = 201$

Demo – Gaussian filtering

Derivatives of an image

How can we find the derivative of an image in e.g. the x-direction?

Derivatives of an image using Gaussians

Let $\mathbf{g}$ be column vector of a Gaussian. Consider the blurred image

$$I_b = \mathbf{g} * \mathbf{g}^\top I.$$

Derivatives of an image using Gaussians

Let $\mathbf{g}$ be column vector of a Gaussian. Consider the blurred image

$$I_b = \mathbf{g} * \mathbf{g}^\top I.$$

Then the derivative of I_b in the x-direction is

$$\begin{aligned}\frac{\partial}{\partial x} I_b &= \frac{\partial}{\partial x}\left(\mathbf{g} * \mathbf{g}^\top I\right) \\ &= \mathbf{g} * \left(\frac{\partial}{\partial x}\mathbf{g}^\top\right) I \\ &= \mathbf{g} * \mathbf{g}_d^\top I,\end{aligned}$$

where $\mathbf{g}_d$ is the derivative of $\boldsymbol{g}$. We can compute $\frac{\partial}{\partial y} I_b$ similarly.

Derivative of the Gaussian

Recall the one-dimensional Gaussian

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-x^2}{2\sigma^2}\right).$$

We can then compute the derivative

$$g_d(x) = \frac{d}{dx} g(x) = \frac{-x}{\sigma^2} g(x).$$

This is straightforward to implement on a computer.

Harris corners

Corners and blobs are great features because they are easy to describe and detect.

Harris corners are detected through as points with locally maximum change from a small shift

$$\Delta I(x, y) = I(x, y) - I(x + \Delta x, y + \Delta y)$$

Harris corner measure

We define a corner as having a large $\Delta I(x, y)^2$, and we average the value over a small area to ensure robustness against noise.

The corner measure is

$$\begin{aligned} c(x, y) &= g * \Delta I(x, y)^2 \\ &= g * \big(I(x, y) - I(x + \Delta x, y + \Delta y)\big)^2, \end{aligned}$$

where $g * \ldots$ is the convolution with the Gaussian.

Replace with Taylor approximation

$$I(x + \Delta x, y + \Delta y) \approx I(x, y) - \frac{\partial I}{\partial x}(x, y)\,\Delta_x - \frac{\partial I}{\partial y}(x, y)\,\Delta_y$$

Replace with Taylor approximation

$$I(x+\Delta x, y+\Delta y) \approx I(x,y) - \underbrace{\frac{\partial I}{\partial x}(x,y)}_{I_x}\,\Delta_x - \underbrace{\frac{\partial I}{\partial y}(x,y)}_{I_y}\,\Delta_y$$

Replace with Taylor approximation

$$I(x+\Delta x, y+\Delta y) \approx I(x,y) - \underbrace{\frac{\partial I}{\partial x}(x,y)}_{I_x}\Delta_x - \underbrace{\frac{\partial I}{\partial y}(x,y)}_{I_y}\Delta_y$$

$$= I(x,y) - \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}$$

I_x and I_y are also the values at (x, y), but we omit it for readability.

Harris corner measure derivation

The corner measure is then

$$
\begin{aligned}
c(x,y) &= g * \big(I(x,y) - I(x+\Delta x, y+\Delta y)\big)^2 \\
&\approx g * \left(\begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}\right)^2
\end{aligned}
$$

Harris corner measure derivation

The corner measure is then

$$
\begin{aligned}
c(x,y) &= g * \big(I(x,y) - I(x+\Delta x, y+\Delta y)\big)^2 \\
&\approx g * \left(\begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}\right)^2 \\
&= g * \left(\begin{bmatrix} \Delta_x & \Delta_y \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}\right) \\
&= g * \left(\begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}\right)
\end{aligned}
$$

Harris corner measure derivation

Finally, we take the convolution with the Gaussian inside

$$
\begin{aligned}
c(x,y) &\approx g * \left(\begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right) \\
&= \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \underbrace{\begin{bmatrix} g * (I_x^2) & g * (I_x I_y) \\ g * (I_x I_y) & g * (I_y^2) \end{bmatrix}}_{\boldsymbol{C}(x,y)} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}.
\end{aligned}
$$

The averaged Hessian

We end up with an averaged Hessian matrix $\boldsymbol{C}(x, y)$.

Corners have a large $c(x, y)$ regardless of the direction of $\begin{bmatrix}\Delta_x & \Delta_y\end{bmatrix}$.

$$c(x, y) = \begin{bmatrix}\Delta x & \Delta y\end{bmatrix} \begin{bmatrix} a & c \\ c & b \end{bmatrix} \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$$

When is this the case?

The averaged Hessian

We end up with an averaged Hessian matrix $\boldsymbol{C}(x, y)$.

Corners have a large $c(x, y)$ regardless of the direction of $\begin{bmatrix}\Delta_x & \Delta_y\end{bmatrix}$.

$$c(x, y) = \begin{bmatrix}\Delta x & \Delta y\end{bmatrix} \begin{bmatrix} a & c \\ c & b \end{bmatrix} \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$$

When is this the case?

When both eigenvalues of $\boldsymbol{C}(x, y)$ are large.

The Harris corner metric continued

Let λ_1 and λ_2 be the eigenvalues of $\boldsymbol{C}(x, y)$.
We then have the following

$$\det\big(\boldsymbol{C}(x,y)\big) = \lambda_1\lambda_2,$$
$$\operatorname{trace}\big(\boldsymbol{C}(x,y)\big) = \lambda_1 + \lambda_2, \text{ let us then define}$$
$$r(x,y) = \det\big(\boldsymbol{C}(x,y)\big) - k \ \operatorname{trace}\big(\boldsymbol{C}(x,y)\big)$$

The Harris corner metric

Now we have the Harris corner metric

$$r(x, y) = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2.$$

k is a free parameter, typically $k = 0.06$.

Notice that $r(x, y)$ is:

- negative for one eigenvalue much greater than the other
- large and positive for large eigenvalues
- small and positive for small eigenvalues

The Harris corner detector

We can now detect corners by finding points where $r(x, y)$ is greater than some threshold τ.

Typically, you can choose τ to be

$$0.1 \cdot \max\big(r(x, y)\big) < \tau < 0.8 \cdot \max\big(r(x, y)\big).$$

Non-maximum suppression

Find the local maximum of one pixel compared to neighbours

$$(I(x,y) - I(x',y')) > 0 \;\; \forall \;\; x' \in \mathcal{N}(x,y)$$

where $\mathcal{N}(x,y)$ is a neighbourhood around the point (x,y)

Suggested procedure

- Initialize an array of size of image with ones
- Compare entire image with one neighbour (e.g to the right)
- Set pixels to zero where it is not larger than neighbour
  - Repeat for all neighbours
- Find coordinate of pixels with value one

Canny edges

Often, we might also consider detecting lines.

To detect (non-straight) lines, we can use the Canny edge detector.

Canny edges

The metric in the canny edge detector is simply the gradient magnitude

$$m(x, y) = \sqrt{I_x^2(x, y) + I_y^2(x, y)}.$$

And the edges are detected by thresholding the magnitude $m(x, y)$, however, in two stages: seed and grow.

Canny edges

- Seed: label edges by a threshold where $m(x, y) > \tau_1$
- Grow: with a second threshold where $m(x, y) > \tau_2$, label iif the new points are next to previously labelled edges.
- The threshold values are chosen such that $\tau_1 > \tau_2$

Exercise

During the exercise, you will

- implement the Harris corner detector
- try the Canny edge detector

Suggestion: After finishing your implementation – try on your own images and perhaps draw some features and see how it works

Learning objectives

After this lecture you should be able to:

- explain the image correspondence problem
- explain the use of image features (key points, interest points)
- understand and implement image filtering
- implement Harris corner detection
- run Canny edge detection

Exercise time!