Image Features

Image Features

- Image features are useful descriptions of local or global image properties designed (or learned!) to accomplish a certain task
- You may want to choose different features for different tasks
- Depending on the problem we need to typically answer three questions:
  - **Where** to extract image features?
  - **What** to extract (what's the content of the feature)?
  - How to use them for your task, e.g., **how to match** them?

Image Features

- Let's watch a video clip

Image Features

- Where is the movie taking place?

Image Features

- Where is the movie taking place?

Image Features

- Where is the movie taking place?

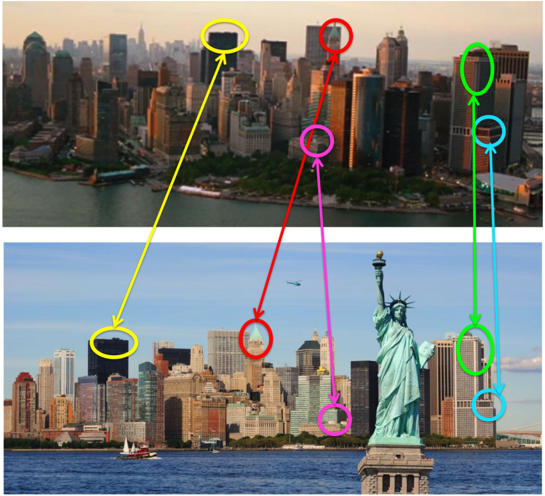


Image Features

- Where is the movie taking place?

We matched in:

- Distinctive locations: **keypoints**
- Distinctive features: **descriptors**

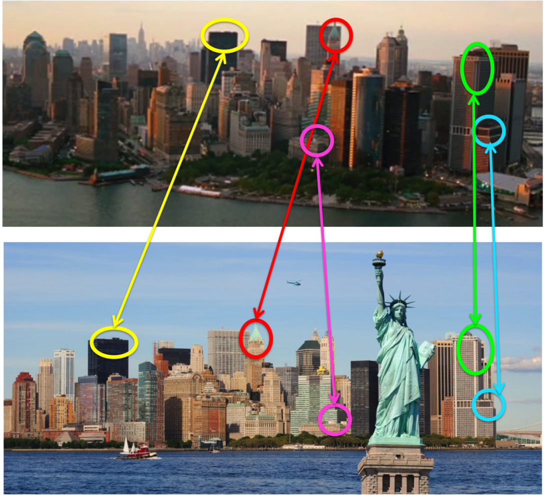


Image Features

- **Tracking**: Where to did the scene/actors move?

Where did it each point originate from the previous frame?

Image Features

- **Tracking**: Where to did the scene/actors move?

We matched:

- Quite distinctive locations
- Quite distinctive features

Where did it each point originate from the previous frame?

Image Features

- A shot in a movie is a clip with a coherent camera (no sudden viewpoint changes)

Image Features

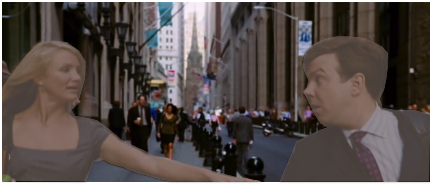
- A shot in a movie is a clip with a coherent camera (no sudden viewpoint changes)

We matched:

- **Globally** – one descriptor for full image
- Descriptor can be simple, e.g. **color**

Image Features

- How could we tell which type of scene it is?



What kind of scene is behind the actors?
Kitchen? Bedroom? Street? Bar?

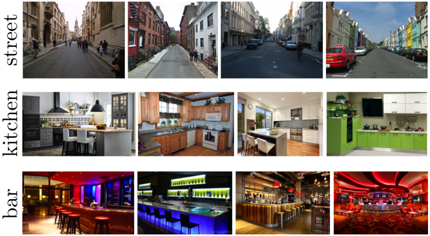
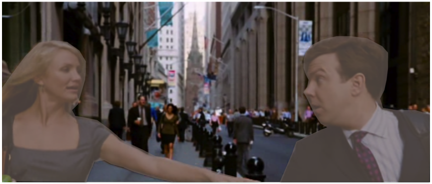


Image Features

- How could we tell which type of scene it is?

We matched:

- **Globally** – one descriptor for full image (?)
- More complex descriptor: color, gradients, "deep" features (learned), etc



What kind of scene is behind the actors?
Kitchen? Bedroom? Street? Bar?

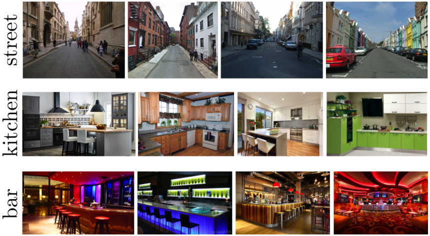


Image Features

- How would we solve this?

Are these two cups of the same type?

- How would we solve this?

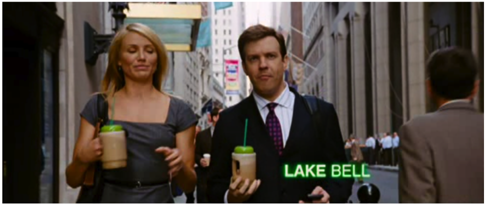
We matched:

- One descriptor for full **patch**
- Descriptor can be simple, e.g. **color**

Are these two cups of the same type?

Image Features

- How would we solve this?



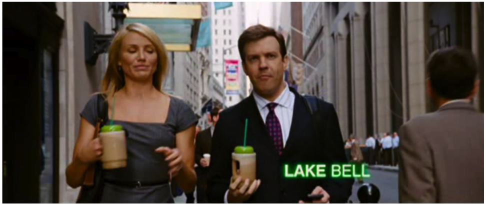
Where can I find this pattern? 

Image Features

- How would we solve this?

We matched:

- At each location
- Compared pixel values



Where can I find this pattern? 

Image Features

- How would we solve this?

Where can I find this pattern? ⇢

Image Features

- How would we solve this?

We matched:

- Distinctive locations
- Distinctive features
- Affine invariant

Where can I find this pattern?

Image Features

- How would we solve this?

Image Features

- **Detection**: **Where** to extract image features?
  - "Interesting" locations (keypoints, interesting regions)
  - In each location (densely)
- **Description**: **What** to extract?
  - What's the spatial scope of the feature?
  - What's the content of the feature?
- **Matching: How to match** them?

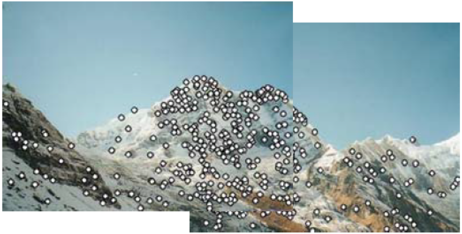
Image Features

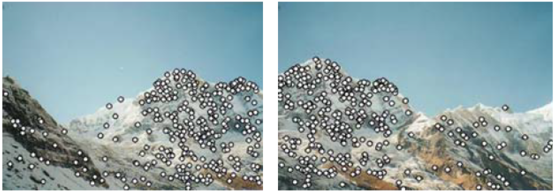
- **Detection**: **Where** to extract image features?
  - "Interesting" locations (keypoints) **TODAY**
  - In each location (densely)
- **Description**: **What** to extract?
  - What's the spatial scope of the feature?
  - What's the content of the feature?
- **Matching: How to match** them?

Image Features:

Interest Point (Keypoint) Detection

Application Example: Image Stitching

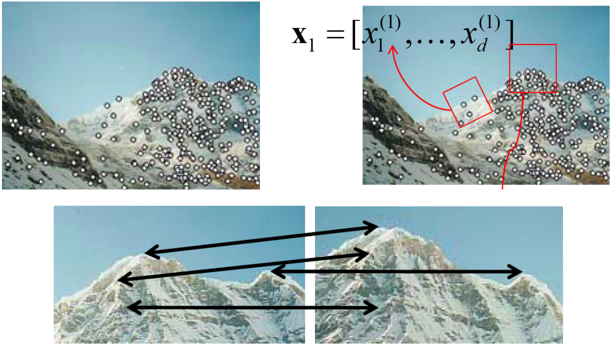




[Source: K. Grauman]

Local Features

- **Detection**: Identify the interest points.
- **Description**: Extract **feature vector** descriptor around each interest point.
- **Matching**: Determine correspondence between descriptors in two views.



[Source: K. Grauman]

Goal: Repeatability of the Interest Point Operator

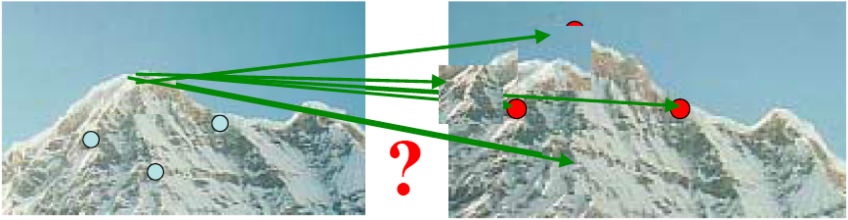
- Our goal is to detect (at least some of) the same points in both images
- We have to be able to run the detection procedure independently per image
- We need to generate enough points to increase our chances of detecting matching points
- We shouldn't generate too many or our matching algorithm will be too slow

Figure: Too few keypoints → little chance to find the true matches

[Source: K. Grauman, slide credit: R. Urtasun]

Goal: Distinctiveness of the Keypoints

- We want to be able to **reliably** determine which point goes with which.

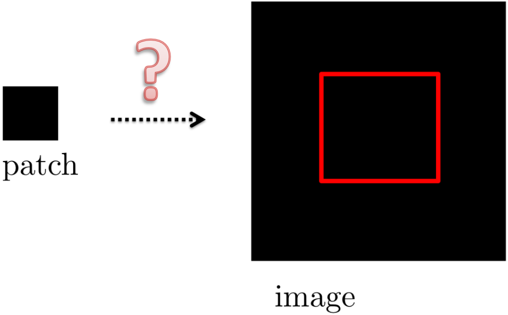


[Source: K. Grauman, slide credit: R. Urtasun]

What Points to Choose?

[Source: K. Grauman]

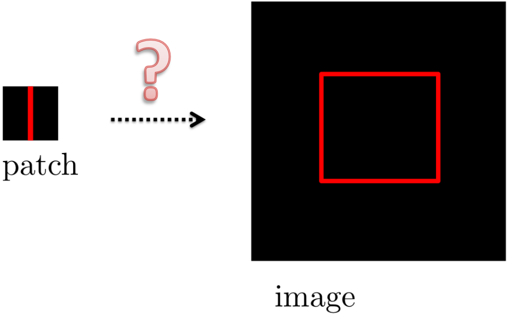
What Points to Choose?



- Textureless patches are nearly impossible to localize.

[Adopted from: R. Urtasun]

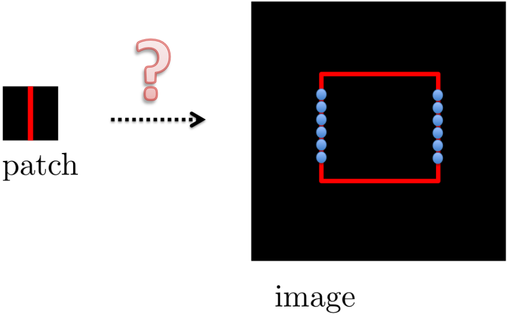
What Points to Choose?



- Textureless patches are nearly impossible to localize.
- Patches with large contrast changes (gradients) are easier to localize.

[Adopted from: R. Urtasun]

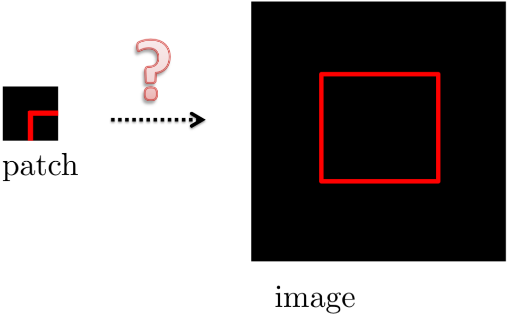
What Points to Choose?



- Textureless patches are nearly impossible to localize.
- Patches with large contrast changes (gradients) are easier to localize.
- But straight line segments cannot be localized on lines segments with the same orientation (aperture problem)

[Adopted from: R. Urtasun]

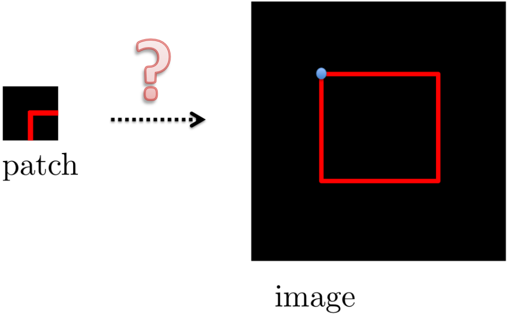
What Points to Choose?



- Textureless patches are nearly impossible to localize.
- Patches with large contrast changes (gradients) are easier to localize.
- But straight line segments cannot be localized on lines segments with the same orientation (aperture problem)
- Gradients in at least two different orientations are easiest, e.g., **corners**!

[Adopted from: R. Urtasun]

What Points to Choose?



- Textureless patches are nearly impossible to localize.
- Patches with large contrast changes (gradients) are easier to localize.
- But straight line segments cannot be localized on lines segments with the same orientation (aperture problem)
- Gradients in at least two different orientations are easiest, e.g., **corners**!

[Adopted from: R. Urtasun]

Interest Points: Corners

- How can we find corners in an image?

Interest Points: Corners

- We should easily recognize the point by looking through a small window.
- Shifting a window in any direction should give a large change in intensity.

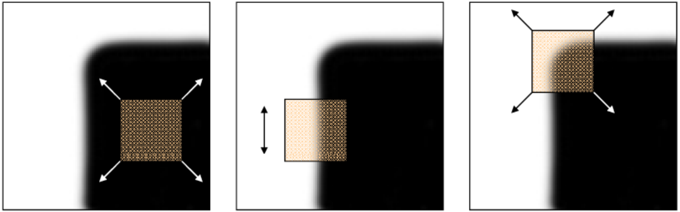


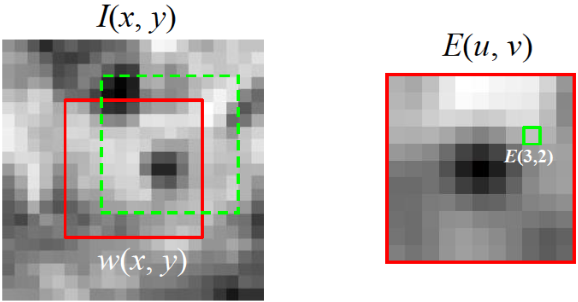
Figure: (left) flat region: no change in all directions, (center) edge: no change along the edge direction, (right) corner: significant change in all directions

[Source: Alyosha Efros, Darya Frolova, Denis Simakov]

Interest Points: Corners

- Compare two image patches using (weighted) summed square difference
- Measures change in appearance of window $w(x, y)$ for the shift

$$E_{\mathrm{WSSD}}(u, v) = \sum_x \sum_y \underbrace{w(x, y)}_{\text{window function}} [\underbrace{I(x + u, y + v)}_{\text{shifted intensity}} - \underbrace{I(x, y)}_{\text{intensity}}]^2$$



[Source: J. Hays]

Interest Points: Corners

- Compare two image patches using (weighted) summed square difference
- Measures change in appearance of window $w(x, y)$ for the shift

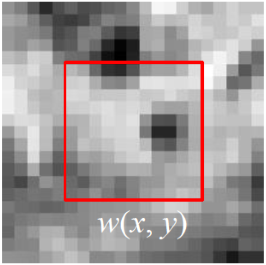
$$E_{\text{WSSD}}(u, v) = \sum_x \sum_y w(x, y)[I(x+u, y+v) - I(x, y)]^2$$

window function

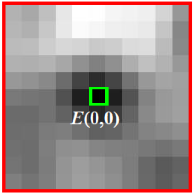
shifted intensity

intensity

$I(x, y)$



$E(u, v)$



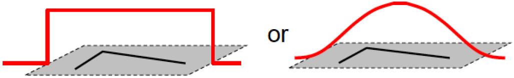
[Source: J. Hays]

Interest Points: Corners

- Compare two image patches using (weighted) summed square difference
- Measures change in appearance of window $w(x, y)$ for the shift

$$E_{\text{WSSD}}(u, v) = \sum_x \sum_y \underbrace{w(x, y)}_{\text{window function}}[\underbrace{I(x + u, y + v)}_{\text{shifted intensity}} - \underbrace{I(x, y)}_{\text{intensity}}]^2$$

Window function $w(x,y)$ =



1 in window, 0 outside

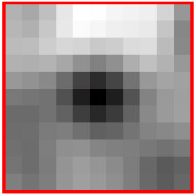
Gaussian

[Source: J. Hays]

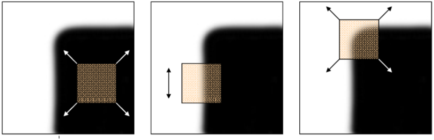
Interest Points: Corners

- Let's look at E_{WSSD}
- We want to find out how this function behaves for small shifts

$$E(u, v)$$



- Remember our goal to detect corners:



Interest Points: Corners

- Using a simple first-order Taylor Series expansion:

$$I(x+u, y+v) \approx I(x,y) + u \cdot \frac{\partial I}{\partial x}(x,y) + v \cdot \frac{\partial I}{\partial y}(x,y)$$

- And plugging it in our expression for E_{WSSD}:

$$\begin{aligned}
E_{\mathrm{WSSD}}(u,v) &= \sum_x \sum_y w(x,y)\Big(I(x+u,y+v) - I(x,y)\Big)^2 \\
&\approx \sum_x \sum_y w(x,y)\Big(I(x,y) + u \cdot I_x + v \cdot I_y - I(x,y)\Big)^2 \\
&= \sum_x \sum_y w(x,y)\Big(u^2 I_x^2 + 2u \cdot v \cdot I_x \cdot I_y + v^2\ I_y^2\Big) \\
&= \sum_x \sum_y w(x,y) \cdot \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}
\end{aligned}$$

Interest Points: Corners

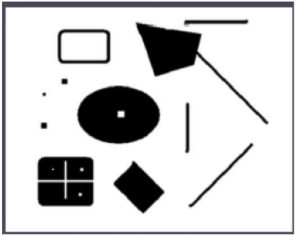
- Since (u, v) doesn't depend on (x, y) we can rewriting it slightly:

$$
\begin{aligned}
E_{\text{WSSD}}(u, v) &= \sum_x \sum_y w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\
&= \begin{bmatrix} u & v \end{bmatrix} \underbrace{\left(\sum_x \sum_y w(x, y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix} \right)}_{\text{Let's denotes this with } M} \begin{bmatrix} u \\ v \end{bmatrix} \\
&= \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}
\end{aligned}
$$

- M is a 2×2 *second moment matrix* computed from image gradients:

$$
M = \sum_x \sum_y w(x, y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}
$$

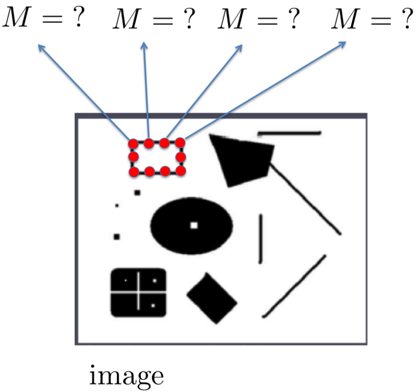
How Do I Compute M?



image

- Let's say I have this image

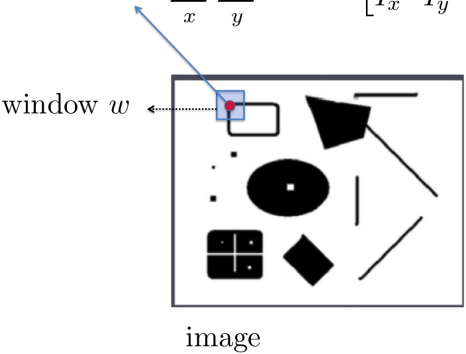
How Do I Compute M?



- Let's say I have this image
- I need to compute a 2×2 second moment matrix in each image location

How Do I Compute M?

$$M = \sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$



- Let's say I have this image
- I need to compute a 2×2 second moment matrix in each image location
- In a particular location I need to compute M as a weighted average of gradients in a window

How Do I Compute M?

$$M = \sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$

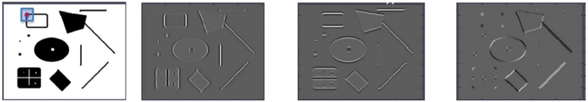


image $\quad I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y} \quad I_x \cdot I_y$

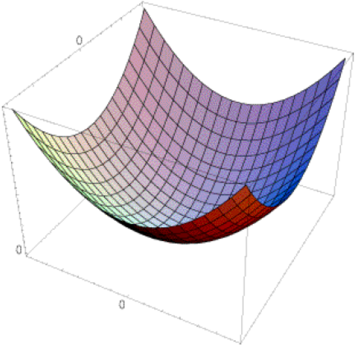
- Let's say I have this image
- I need to compute a 2×2 second moment matrix in each image location
- In a particular location I need to compute M as a weighted average of gradients in a window

I can do this efficiently by computing three matrices, I_x^2, I_y^2 and $I_x \cdot I_y$, and convolving each one with a filter, e.g. a box or Gaussian filter

- We now have M computed in each image location
- Our E_{WSSD} is a **quadratic function** where M implies its shape

$$E_{\mathrm{WSSD}}(u,v) \quad = \quad \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$



[Source: J. Hays]

Interest Points: Corners

- Let's take a horizontal "slice" of $E_{\mathrm{WSSD}}(u, v)$:

$$\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \mathrm{const}$$

- This is the equation of an ellipse

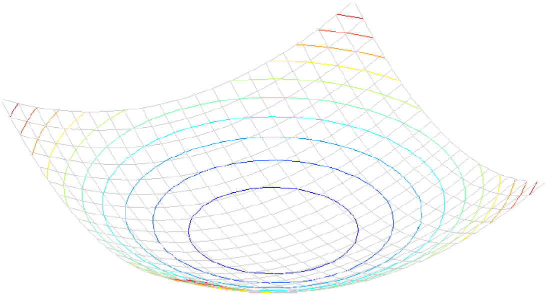


Figure: Different ellipses obtain by different horizontal "slices"

Interest Points: Corners

- Let's take a horizontal "slice" of $E_{\mathrm{WSSD}}(u, v)$:

$$\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \mathrm{const}$$

- This is the equation of an ellipse

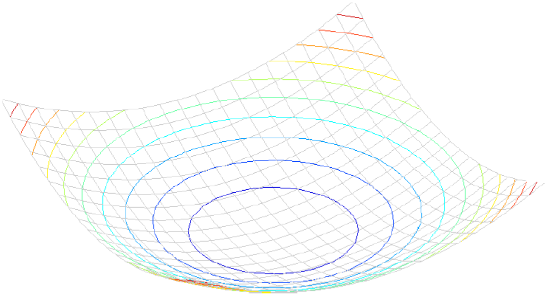


Figure: Different ellipses obtain by different horizontal "slices"

Interest Points: Corners

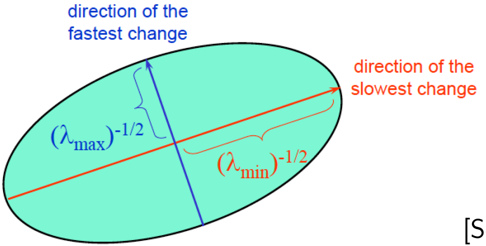
- Our matrix M is symmetric:

$$M = \sum_x \sum_y w(x,y) \begin{bmatrix} I_x^2 & I_x \cdot I_y \\ I_x \cdot I_y & I_y^2 \end{bmatrix}$$

- And thus we can diagonalize it (in Matlab: [V,D] = $\textsc{eig}$(M)):

$$M = V \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} V^{-1}$$

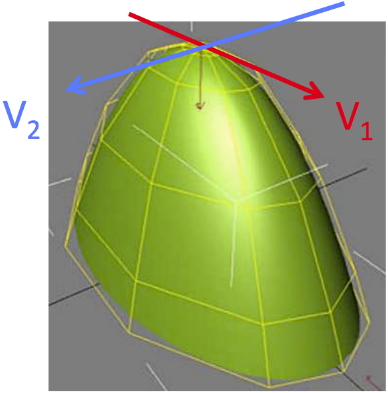
- Columns of V are major and minor axes of ellipse, $\lambda^{-1/2}$ are radius

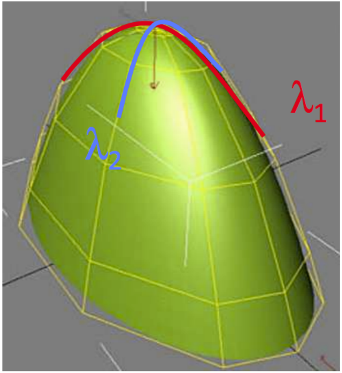


[Source: J. Hays]

Interest Points: Corners

- Columns of V are **principal directions**
- λ_1, λ_2 are **principal curvatures**

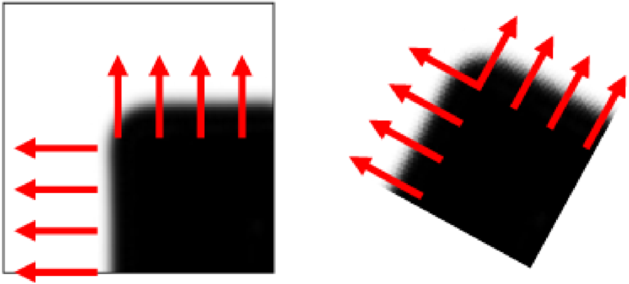




[Source: F. Flores-Mangas]

Interest Points: Corners

- The eigenvalues of M (λ_1, λ_2) reveal the amount of intensity change in the two principal orthogonal gradient directions in the window



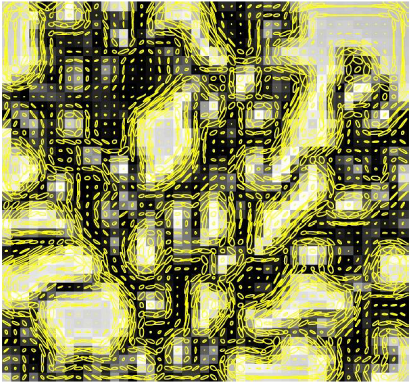
[Source: R. Szeliski, slide credit: R. Urtasun]

- How do the ellipses look like for this image?

[Source: J. Hays]

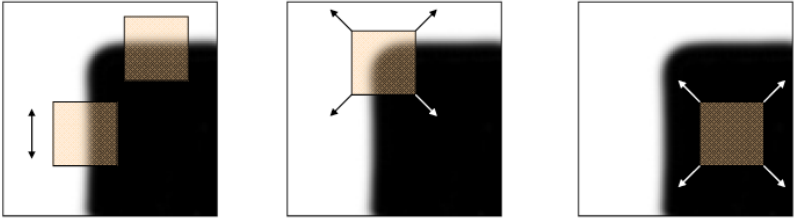
Interest Points: Corners

- How do the ellipses look like for this image?



[Source: J. Hays]

Interest Points: Corners



"edge":
$\lambda_1 >> \lambda_2$
$\lambda_2 >> \lambda_1$

"corner":
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;

"flat" region
λ_1 and λ_2 are small;

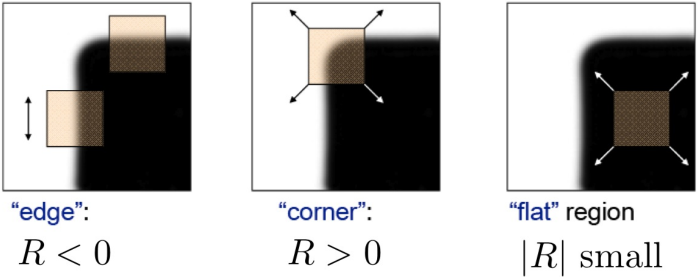
[Source: K. Grauman, slide credit: R. Urtasun]

Interest Points: Criteria to Find Corners

- Harris and Stephens, '88, is rotationally invariant and downweighs edge-like features where $\lambda_1 \gg \lambda_0$

$$R = \det(M) - \alpha \cdot \mathrm{trace}(M)^2 = \lambda_0 \lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$$

- **Why** go via det and trace and not use a criteria with λ?
- α a constant (0.04 to 0.06)



- The corresponding detector is called **Harris corner detector**

Interest Points: Criteria to Find Corners

- Harris and Stephens, 88 is rotationally invariant and downweighs edge-like features where $\lambda_1 \gg \lambda_0$

$$R = \det(M) - \alpha \text{trace}(M)^2 = \lambda_0 \lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$$

- Shi and Tomasi, 94 proposed the smallest eigenvalue of **A**, i.e., $\lambda_0^{-1/2}$.
- Triggs, 04 suggested

$$\lambda_0 - \alpha \lambda_1$$

  also reduces the response at 1D edges, where aliasing errors sometimes inflate the smaller eigenvalue

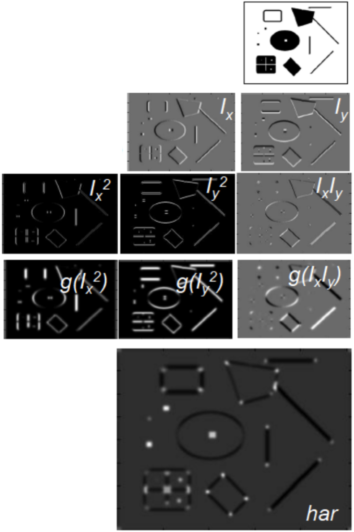
- Brown et al, 05 use the harmonic mean

$$\frac{\det(\mathbf{A})}{\text{trace}(\mathbf{A})} = \frac{\lambda_0 \lambda_1}{\lambda_0 + \lambda_1}$$

[Source R. Urtasun]

Harris Corner detector

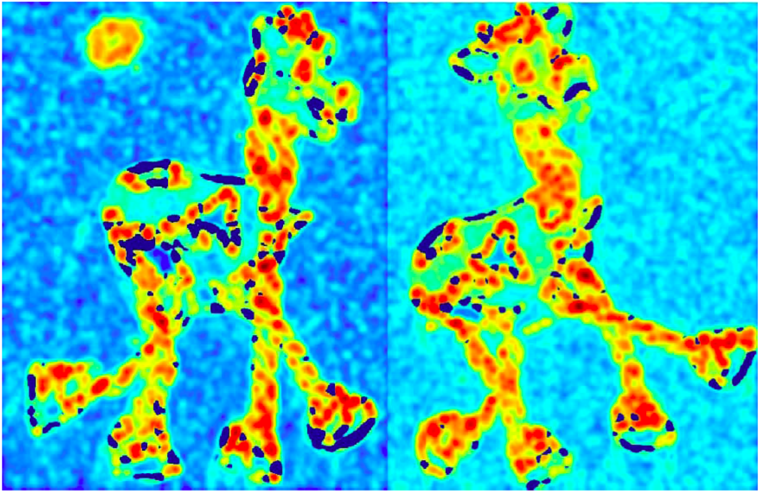
1. Compute gradients I_x and I_y
2. Compute I_x^2, I_y^2, $I_x \cdot I_y$
3. Average (Gaussian) $\rightarrow$ gives M
4. Compute $R = \det(M) - \alpha\,\mathrm{trace}(M)^2$ for each image window (*cornerness* score)
5. Find points with large R ($R >$ threshold).
6. Take only points of local maxima, i.e., perform non-maximum suppression



Example

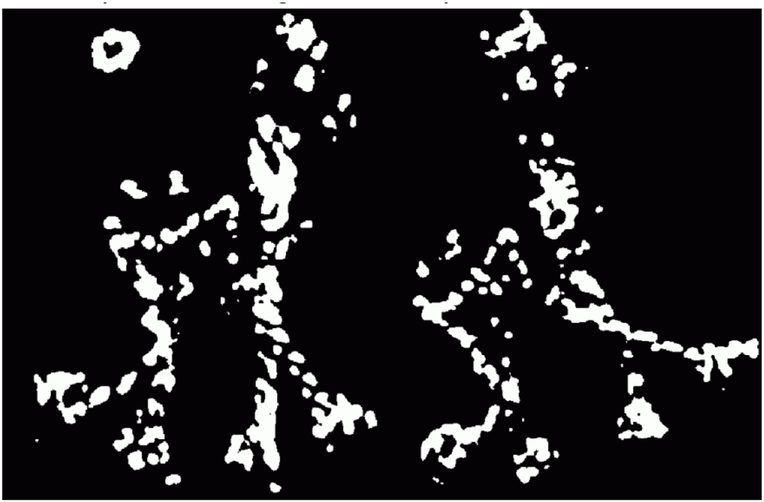
[Source: K. Grauman]

1) Compute Cornerness



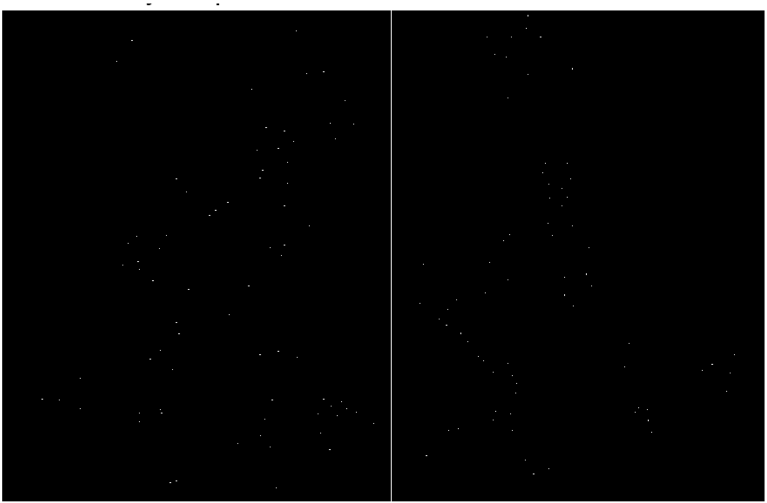
[Source: K. Grauman]

2) Find High Response



[Source: K. Grauman]

3) Non-maxima Suppresion



[Source: K. Grauman]

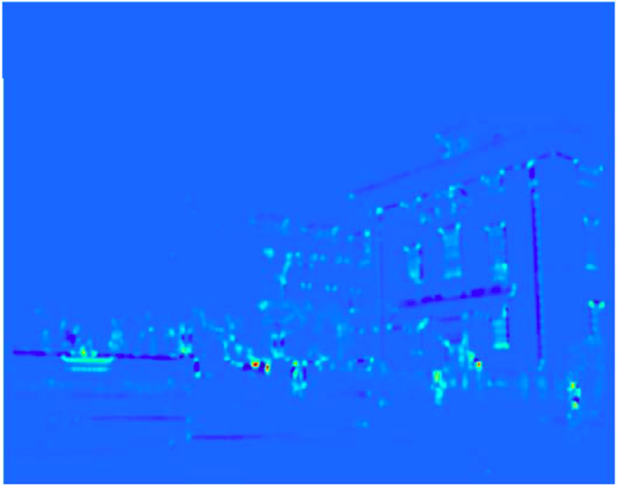
Results

[Source: K. Grauman]

Another Example

[Source: K. Grauman]

Cornerness



[Source: K. Grauman]

Interest Points

[Source: K. Grauman]

Interest Points – Ideal Properties?

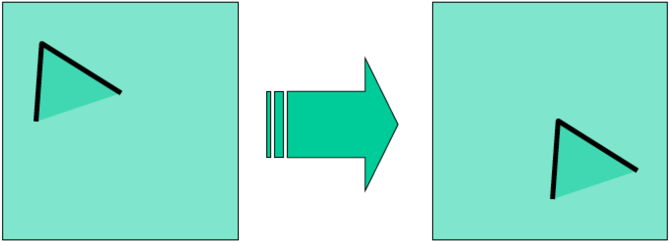
- We want corner locations to be **invariant** to photometric transformations and **covariant** to geometric transformations

Invariance : Image is transformed and corner locations do not change

Covariance : If we have two transformed versions of the same image, features should be detected in corresponding locations

Properties of Harris Corner Detector

- Shift?

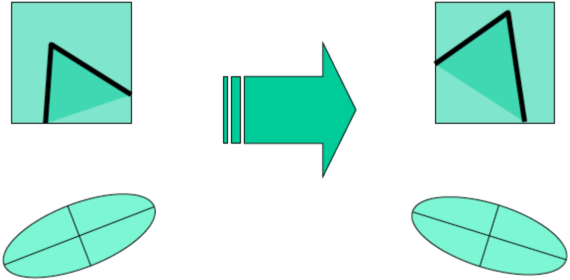


- Harris corner detector is shift-covariant (our window functions shift)

[Source: J. Hays]

Properties of Harris Corner Detector

- Rotation?

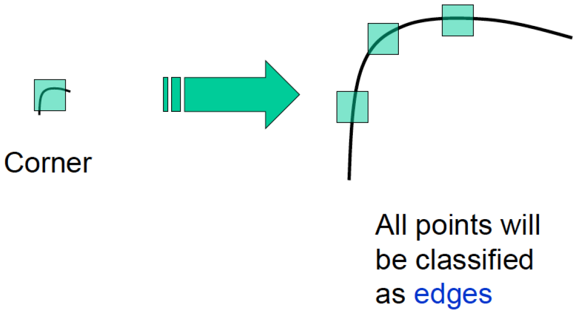


- Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same
- Harris corner detector is rotation-covariant

[Source: J. Hays]

Properties of Harris Corner Detector

- Scale?



- Corner location is **not scale invariant/covariant**!

[Source: J. Hays]

Next Time

- Can we also define keypoints that are shift, rotation and scale invariant/covariant?
- What should be our **description** around keypoint?