EECS 4422/5323 Computer Vision

Feature Detection Lecture 2

Calden Wloka

23 September, 2019

Announcments

- White papers are due today
- Reminder: When you submit, follow instructions. PDF filenames must start with your last name.
- Assignment 1 is out
  - You can download the assignment PDF from the link in the course schedule
  - Files for the programming components can be found at the bottom of the schedule page under "Assignment Downloads"
- This weeks' labs are dedicated time for you to work on the assignment and get help or clarifications from the TA
  - Please don't ask "Is this the right answer?" - this is not what the time is for
  - Use this time to make sure you understand what is being asked, and get any help you might need with software requirements or syntax

Computing Resources - The GPUs in the Room

A number of you are looking into projects which will be easier to complete with GPU resources, and so I wanted to mention available options.

- There are ten machines in the lab with GPUs
- There are options for cloud-based computing resources (an example link here), but I can't vouch for them

Outline

- Historical Patterns
- Semantic Features Overview
- Patches
- Edges

Research Often Follows Patterns

There have been a number of cycles to computer vision. For example, the type of data (*natural* vs. *synthetic*) which is most prevalent has switched several times.

Research Often Follows Patterns

There have been a number of cycles to computer vision. For example, the type of data (*natural* vs. *synthetic*) which is most prevalent has switched several times.

Synthetic data advantages:

- Full control over the appearance and properties of the data

Research Often Follows Patterns

There have been a number of cycles to computer vision. For example, the type of data (*natural* vs. *synthetic*) which is most prevalent has switched several times.

Synthetic data advantages:

- Full control over the appearance and properties of the data
- Easy to generate as much data as needed

Research Often Follows Patterns

There have been a number of cycles to computer vision. For example, the type of data (*natural* vs. *synthetic*) which is most prevalent has switched several times.

Synthetic data advantages:

- Full control over the appearance and properties of the data
- Easy to generate as much data as needed
- Usually ground-truth labels can be accurately generated im tandem with the data

Research Often Follows Patterns

There have been a number of cycles to computer vision. For example, the type of data (*natural* vs. *synthetic*) which is most prevalent has switched several times.

Synthetic data advantages:

- Full control over the appearance and properties of the data
- Easy to generate as much data as needed
- Usually ground-truth labels can be accurately generated im tandem with the data

Synthetic data disadvantages:

- Data can be "too clean"

Research Often Follows Patterns

There have been a number of cycles to computer vision. For example, the type of data (*natural* vs. *synthetic*) which is most prevalent has switched several times.

Synthetic data advantages:

- Full control over the appearance and properties of the data
- Easy to generate as much data as needed
- Usually ground-truth labels can be accurately generated im tandem with the data

Synthetic data disadvantages:

- Data can be "too clean"
  - Important sources of natural variation might not be captured

Research Often Follows Patterns

There have been a number of cycles to computer vision. For example, the type of data (*natural* vs. *synthetic*) which is most prevalent has switched several times.

Synthetic data advantages:

- Full control over the appearance and properties of the data
- Easy to generate as much data as needed
- Usually ground-truth labels can be accurately generated im tandem with the data

Synthetic data disadvantages:

- Data can be "too clean"
  - Important sources of natural variation might not be captured
  - Lack of realistic noise might lead to overfitting of features

Research Often Follows Patterns

Natural data advantages:

Research Often Follows Patterns

Natural data advantages:

- More likely to capture a wide variety of data variations

Research Often Follows Patterns

Natural data advantages:

- More likely to capture a wide variety of data variations
  - Realistic noise patterns

Research Often Follows Patterns

Natural data advantages:

- More likely to capture a wide variety of data variations
  - Realistic noise patterns
  - No need to explicitly encode all possible scenarios or data dimensions

Research Often Follows Patterns

Natural data advantages:

- More likely to capture a wide variety of data variations
  - Realistic noise patterns
  - No need to explicitly encode all possible scenarios or data dimensions

Natural data disadvantages:

- Costly and tedious to gather and annotate

Research Often Follows Patterns

Natural data advantages:

- More likely to capture a wide variety of data variations
  - Realistic noise patterns
  - No need to explicitly encode all possible scenarios or data dimensions

Natural data disadvantages:

- Costly and tedious to gather and annotate
- Can still overfit, particularly as performance begins to saturate

Research Often Follows Patterns

Natural data advantages:

- More likely to capture a wide variety of data variations
  - Realistic noise patterns
  - No need to explicitly encode all possible scenarios or data dimensions

Natural data disadvantages:

- Costly and tedious to gather and annotate
- Can still overfit, particularly as performance begins to saturate
- Can lead to false confidence in the degree of data representation (*e.g.* see the Training Humans project)

Patterns in Features

Just like with the style and design of computer vision data, there are patterns in the design and approach of computer vision features.

Patterns in Features

Just like with the style and design of computer vision data, there are patterns in the design and approach of computer vision features.

First, some terminology:

Patterns in Features

Just like with the style and design of computer vision data, there are patterns in the design and approach of computer vision features.

First, some terminology:

- We say that a feature or process is *semantic* if we can ascribe interpretable meaning to the specific activity along its dimension
  - Semantic features are typically carefully designed
  - Examples include edges and corners

Patterns in Features

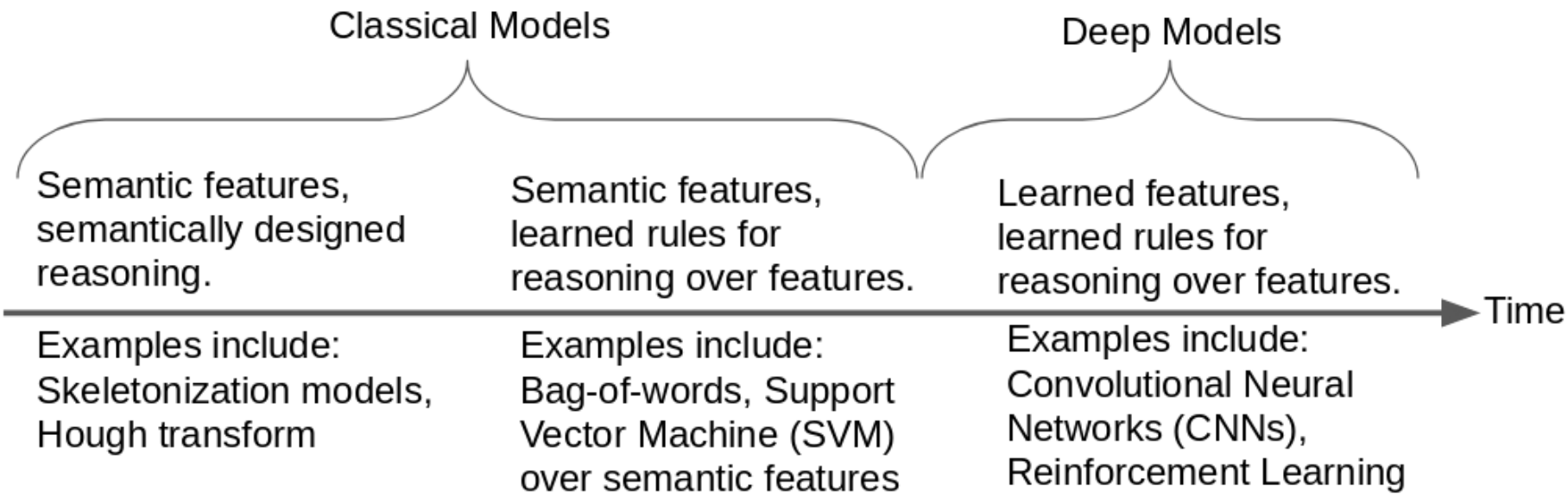
Just like with the style and design of computer vision data, there are patterns in the design and approach of computer vision features.

First, some terminology:

- We say that a feature or process is *semantic* if we can ascribe interpretable meaning to the specific activity along its dimension
  - Semantic features are typically carefully designed
  - Examples include edges and corners
- We say that a feature or process is *learned* if it has been set based on some pattern in a set of training data
  - Learned features often are difficult to interpret on their own
  - Examples include features based on PCA and the features found in deep networks

The Learning Creep

A very prevalent pattern in computer vision is in the balance between semantic and learned design.



So Why Bother With Semantic Features?

If it's possible to learn features and then learn the rules to reason over those features, why bother studying semantic feature design?

- Deep learning is data hungry and computationally intensive

So Why Bother With Semantic Features?

If it's possible to learn features and then learn the rules to reason over those features, why bother studying semantic feature design?

- Deep learning is data hungry and computationally intensive
- While learning usually gives better overall performance, it also tends to give more unpredictable behaviour

So Why Bother With Semantic Features?

If it's possible to learn features and then learn the rules to reason over those features, why bother studying semantic feature design?

- Deep learning is data hungry and computationally intensive
- While learning usually gives better overall performance, it also tends to give more unpredictable behaviour
- Some applications require explanatory traceback

So Why Bother With Semantic Features?

If it's possible to learn features and then learn the rules to reason over those features, why bother studying semantic feature design?

- Deep learning is data hungry and computationally intensive
- While learning usually gives better overall performance, it also tends to give more unpredictable behaviour
- Some applications require explanatory traceback
- Understanding how to handle features, whether learned or semantic, is important for many fields of artificial intelligence, and semantic features are in many ways easier to visualize and intuit

Image Patch Features

Image patch feature detectors are sometimes also known as *keypoint detectors*, and the the primary goal for these methods is to find local patches of an image which are distinctive and which can be matched to corresponding patches in other images.

Applications of Keypoints

Image stitching for panoramas.

Image source: Wloka Farms fruitstand

Applications of Keypoints

Object detection and localization.

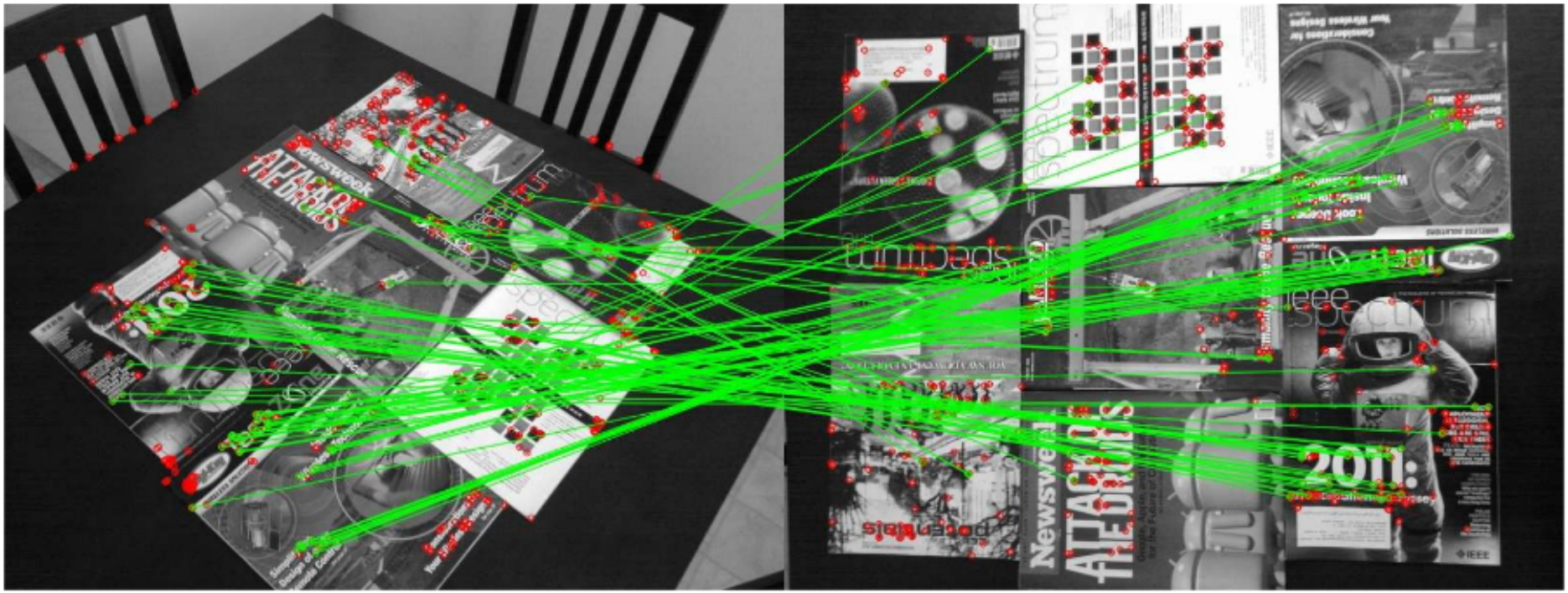


Image source: Rublee *et al.*, 2011

Applications of Keypoints

As a basis for stereo or temporal correspondence, or any other application in which correspondence between frames is desired.

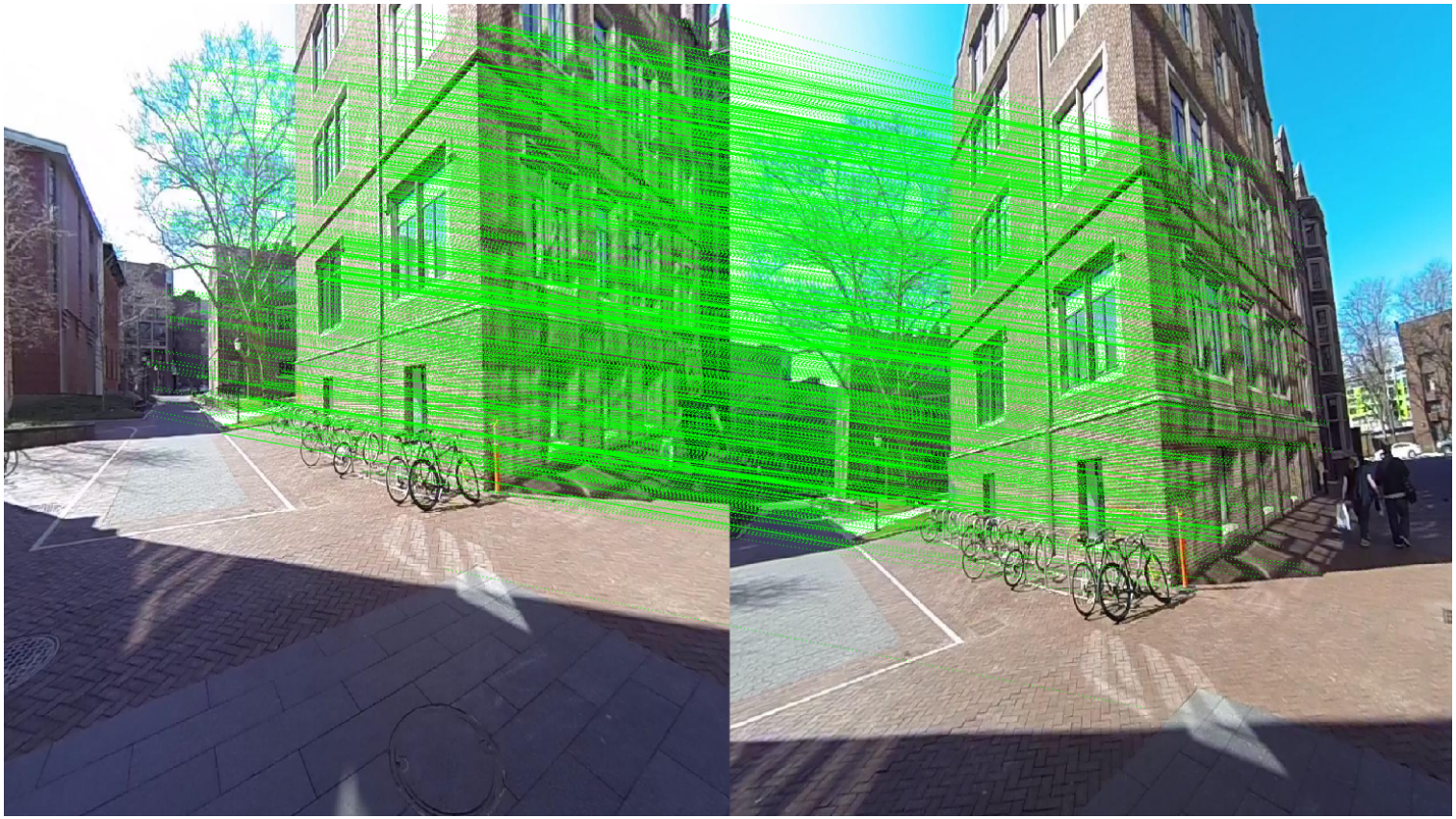


Image source: Singh, Structure from Motion course material

What Makes a Good Patch?

Patch features tend to be sparsely assigned over an image, because not all patches are equivalently good for matching.

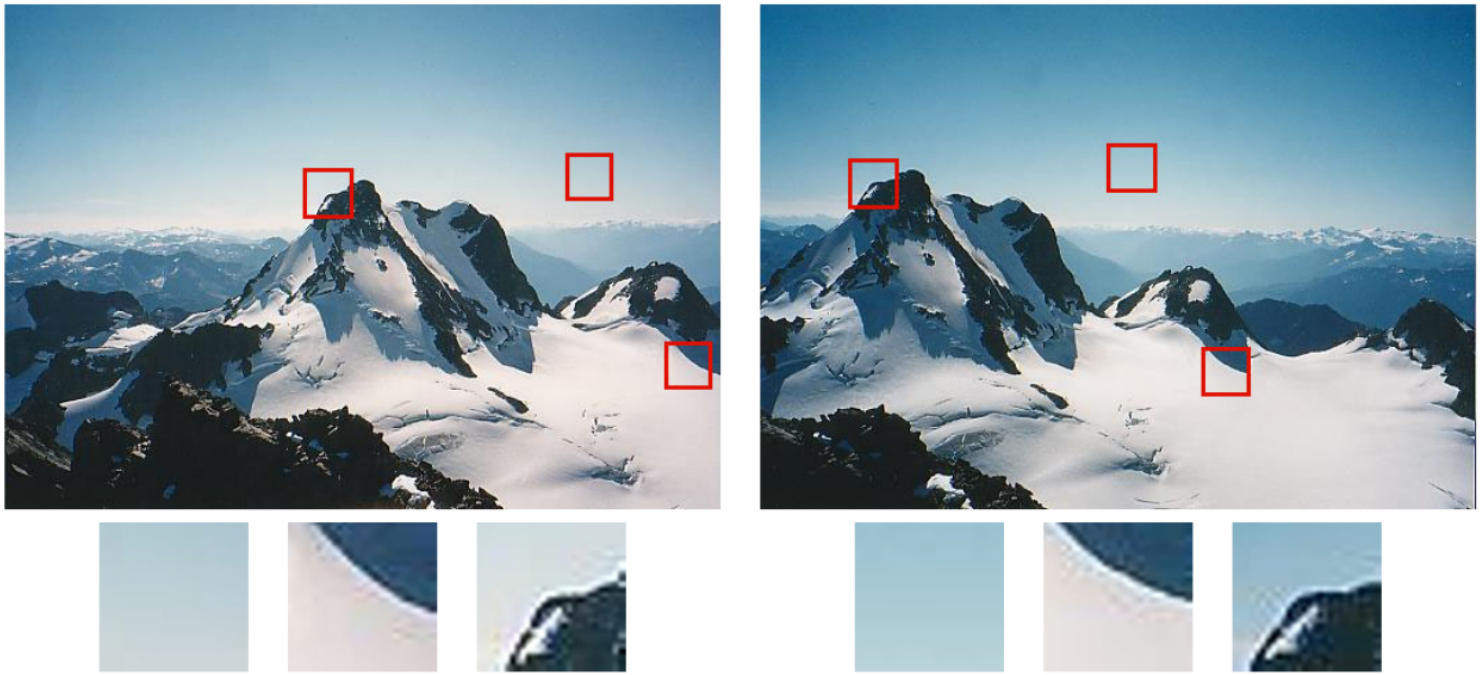


Image source: Szeliski, 2011

What Makes a Good Patch?

A schematic overview of patch suitability.

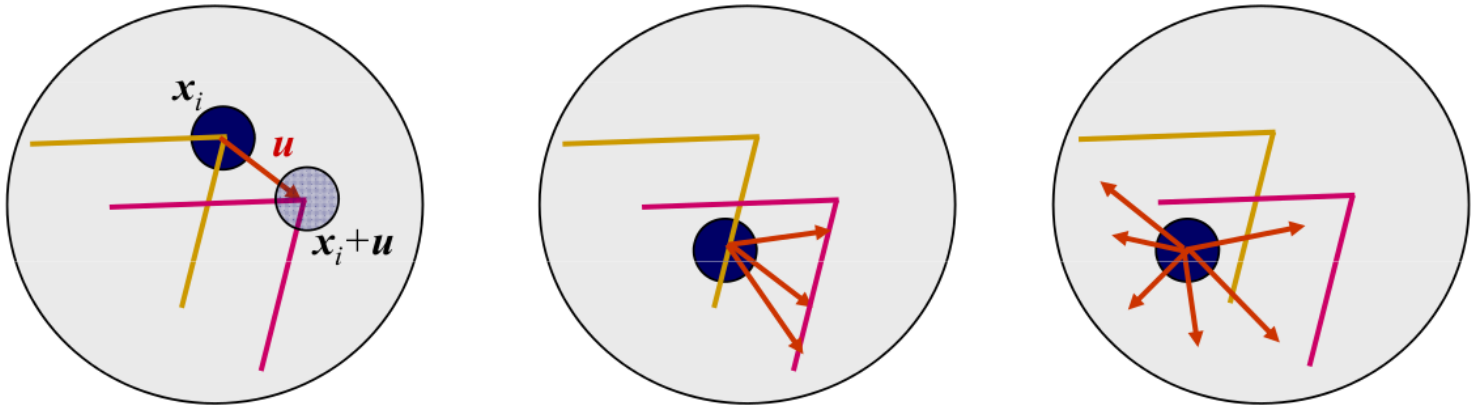


Image source: Szeliski, 2011

a) The dark patch contains two uniquely angled segments and their intersection, so may be matched to the light patch

What Makes a Good Patch?

A schematic overview of patch suitability.

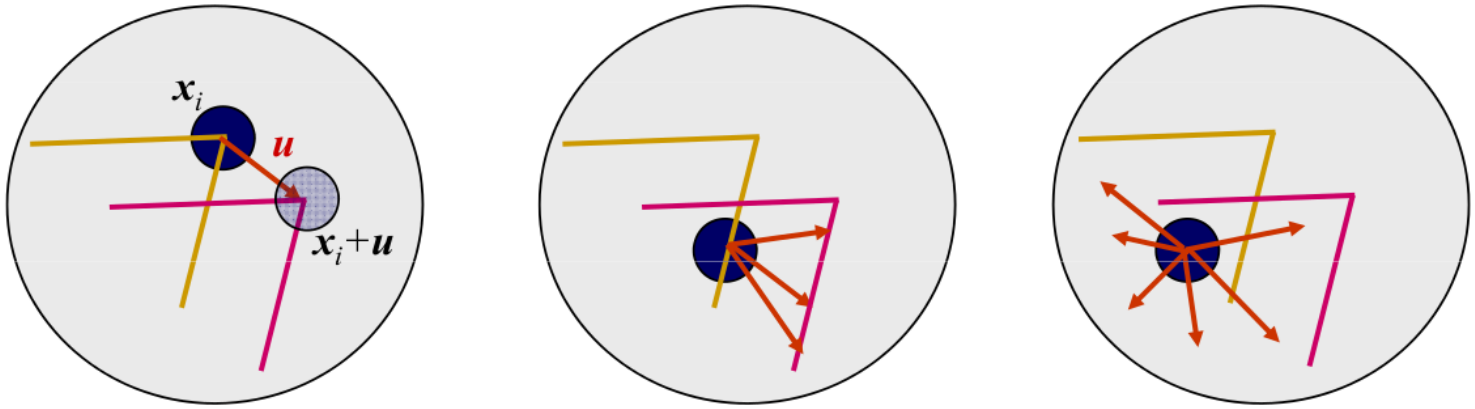


Image source: Szeliski, 2011

a) The dark patch contains two uniquely angled segments and their intersection, so may be matched to the light patch
b) The dark patch only has enough information to be matched somewhere along the corresponding line segment (aperture problem or barber pole illusion)

What Makes a Good Patch?

A schematic overview of patch suitability.

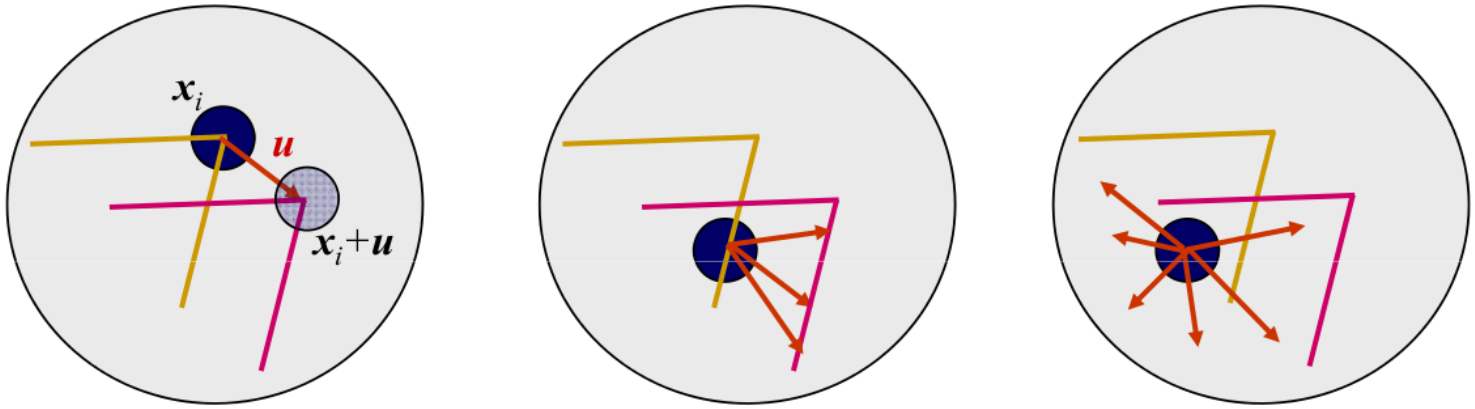


Image source: Szeliski, 2011

a) The dark patch contains two uniquely angled segments and their intersection, so may be matched to the light patch
b) The dark patch only has enough information to be matched somewhere along the corresponding line segment (aperture problem or barber pole illusion)
c) The dark patch has no useful information, and could be conceivably matched to any background patch

What Makes a Good Patch?

An example in a real image.

- Top image: input
- Middle row: patches extracted
- Bottom row: autocorrelation surfaces

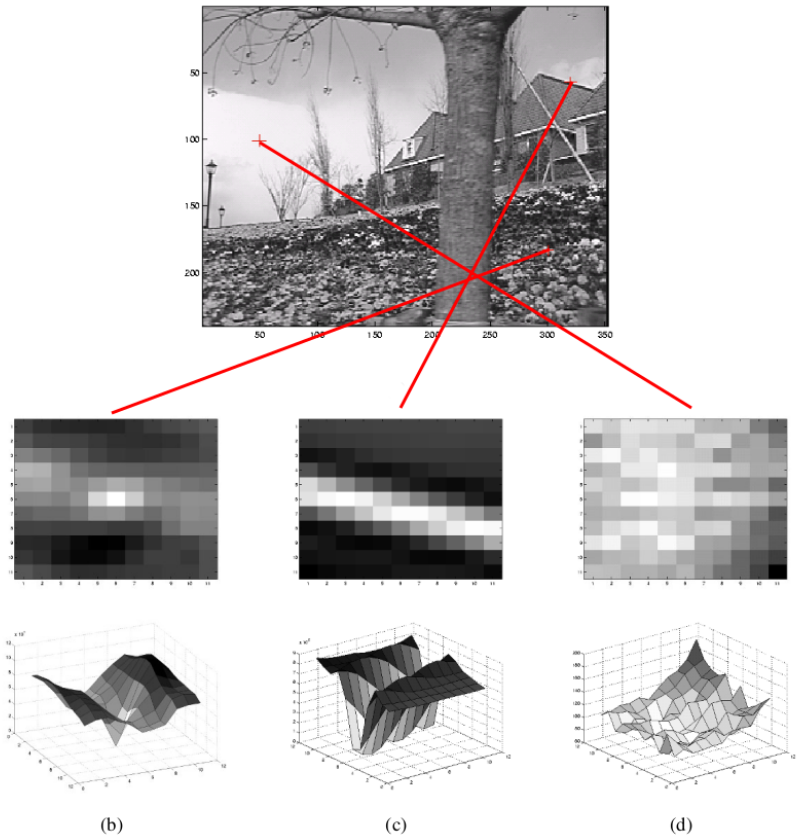


Image source: Modified from Szeliski, 2011

Patch cross-correlation is brittle

For some applications (such as panoramic image stitching), we can assume minimal changes to spatial scale and object orientation while performing patch matching.

However, as we saw in lab, template matching under even very small changes in size or orientation often fails. We therefore would like to find approaches to patch matching which are more robust.

Feature Matching Method Steps

Most feature matching algorithms include two distinct steps:

1. Candidate keypoint detection
2. Keypoint descriptor calculation

Keypoint Detection

There are a number of different ways to do this, including:

- Maxima over patch autocorrelation

Keypoint Detection

There are a number of different ways to do this, including:

- Maxima over patch autocorrelation
- Difference of Gaussians (DoG)

Keypoint Detection

There are a number of different ways to do this, including:

- Maxima over patch autocorrelation
- Difference of Gaussians (DoG)
- Harris corner detection (Harris and Stephens, 1988)

Keypoint Detection

There are a number of different ways to do this, including:

- Maxima over patch autocorrelation
- Difference of Gaussians (DoG)
- Harris corner detection (Harris and Stephens, 1988)
- FAST (Rosten and Drummond, 2006)

Keypoint Selection

Once the keypoint detector has run, we need to decide which keypoints to keep. Common approaches include:

- Cut-off thresholding

Keypoint Selection

Once the keypoint detector has run, we need to decide which keypoints to keep. Common approaches include:

- Cut-off thresholding
- Adaptive non-maximal suppression (ANMS - Brown, Szeliski, and Winder, 2005)

Keypoint Selection

Once the keypoint detector has run, we need to decide which keypoints to keep. Common approaches include:

- Cut-off thresholding
- Adaptive non-maximal suppression (ANMS - Brown, Szeliski, and Winder, 2005)
- Threshold or soft-weighting from an additional value (*e.g.* saliency)

Keypoint Description

Once keypoints are found, they can be made more effective by computing a *descriptor*. This is typically a high-demonsional feature vector computed at each keypoint. The goal of a keypoint descriptor is typically to be robust to an *affine* transformation (*i.e.*, scale, rotation, reflection, and shear), and as such includes the following aims:

- High discriminability between keypoints

Keypoint Description

Once keypoints are found, they can be made more effective by computing a *descriptor*. This is typically a high-demonsional feature vector computed at each keypoint. The goal of a keypoint descriptor is typically to be robust to an *affine* transformation (*i.e.*, scale, rotation, reflection, and shear), and as such includes the following aims:

- High discriminability between keypoints
- Rotational invariance (often done by computing a canonical orientation for a given keypoint)

Keypoint Description

Once keypoints are found, they can be made more effective by computing a *descriptor*. This is typically a high-demonsional feature vector computed at each keypoint. The goal of a keypoint descriptor is typically to be robust to an *affine* transformation (*i.e.*, scale, rotation, reflection, and shear), and as such includes the following aims:

- High discriminability between keypoints
- Rotational invariance (often done by computing a canonical orientation for a given keypoint)
- Scale invariance (often done by computing descriptor features over an image pyramid)

Comparing Keypoints

Once our keypoint descriptors have been computed we need to determine a method for computing the distnace between two descriptors (a *metric*). This can often be based directly on Euclidean distance, but it is sometimes more effective to use PCA or whitening.

Comparing Keypoints

Once our keypoint descriptors have been computed we need to determine a method for computing the distnace between two descriptors (a *metric*). This can often be based directly on Euclidean distance, but it is sometimes more effective to use PCA or whitening.

This metric can then be used to perform clustering over descriptor feature space, compute nearest neighbours during matching, and other reasoning tasks.

Comparing Keypoints

Once our keypoint descriptors have been computed we need to determine a method for computing the distnace between two descriptors (a *metric*). This can often be based directly on Euclidean distance, but it is sometimes more effective to use PCA or whitening.

This metric can then be used to perform clustering over descriptor feature space, compute nearest neighbours during matching, and other reasoning tasks.

For large numbers of keypoints, it is often beneficial to use data representations like a k-d tree to more efficient calculations.

Keypoint Models

For many years keypoint detectors dominated a broad range of applications in computer vision, and still prove useful for a number of specific roles and areas today. Examples of keypoint detection methods which see high use include:

- Scale Invariant Feature Transform (SIFT - Lowe, 1999), and related methods:

Keypoint Models

For many years keypoint detectors dominated a broad range of applications in computer vision, and still prove useful for a number of specific roles and areas today. Examples of keypoint detection methods which see high use include:

- Scale Invariant Feature Transform (SIFT - Lowe, 1999), and related methods:
  - SURF (Bay *et al.*, 2006)

Keypoint Models

For many years keypoint detectors dominated a broad range of applications in computer vision, and still prove useful for a number of specific roles and areas today. Examples of keypoint detection methods which see high use include:

- Scale Invariant Feature Transform (SIFT - Lowe, 1999), and related methods:
  - SURF (Bay *et al.*, 2006)
  - PCA-SIFT (Ke and Sukthankar, 2004)

Keypoint Models

For many years keypoint detectors dominated a broad range of applications in computer vision, and still prove useful for a number of specific roles and areas today. Examples of keypoint detection methods which see high use include:

- Scale Invariant Feature Transform (SIFT - Lowe, 1999), and related methods:
  - SURF (Bay *et al.*, 2006)
  - PCA-SIFT (Ke and Sukthankar, 2004)
- Gradient Location-Orientation Histogram (GLOH - Mikolajczyk and Schmid, 2005)

Keypoint Models

For many years keypoint detectors dominated a broad range of applications in computer vision, and still prove useful for a number of specific roles and areas today. Examples of keypoint detection methods which see high use include:

- Scale Invariant Feature Transform (SIFT - Lowe, 1999), and related methods:
  - SURF (Bay *et al.*, 2006)
  - PCA-SIFT (Ke and Sukthankar, 2004)
- Gradient Location-Orientation Histogram (GLOH - Mikolajczyk and Schmid, 2005)
- Oriented FAST and Rotated BRIEF (ORB - Rublee *et al.*, 2011)

Edges

We have already talked about gradients and edge detection filters, including steerable Gaussian derivative kernels. This section will expand upon this topic, particularly with respect to the interpretation and characterization of edges.

Why Edges?

We spend a lot of time in this course talking about edges. This is because edges often carry a lot of useful information.

- Object boundary

Why Edges?

We spend a lot of time in this course talking about edges. This is because edges often carry a lot of useful information.

- Object boundary
- Internal structure or components

Why Edges?

We spend a lot of time in this course talking about edges. This is because edges often carry a lot of useful information.

- Object boundary
- Internal structure or components
- Symbolic meaning (*e.g.* text)

Why Edges?

We spend a lot of time in this course talking about edges. This is because edges often carry a lot of useful information.

- Object boundary
- Internal structure or components
- Symbolic meaning (*e.g.* text)
- Distinctive texture or patterns for recognition

Semantic Edges

Some edges are more meaningful than others.

Image source: Shi *et al.*, 2013

Illusory Contours

Sometimes an "edge" is missing local contrast.

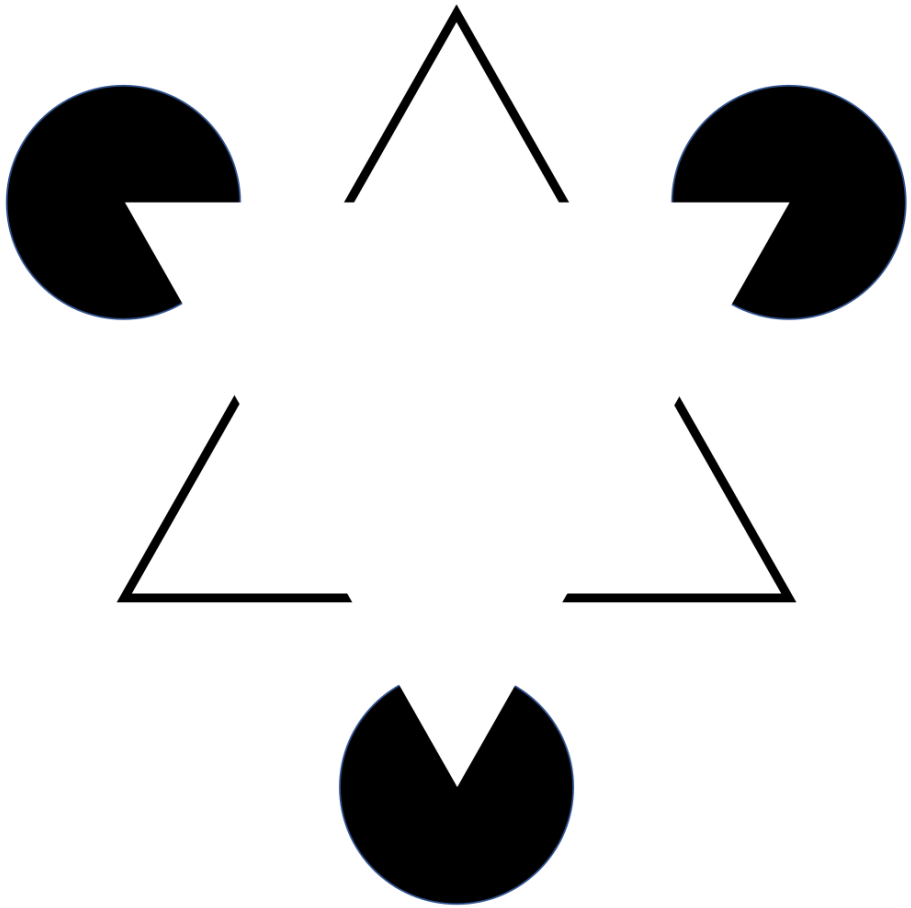


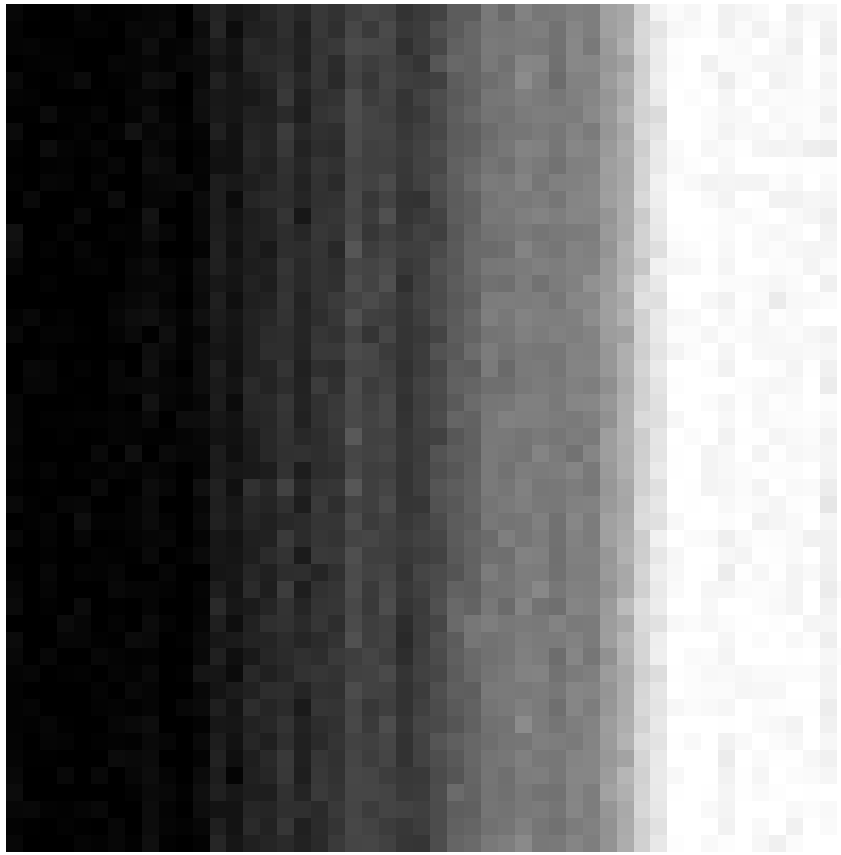
Image source: Kanizsa's Triangle

The Challenge of Shadows

Image source: Steinmetz, National Geographic, 2005

Edge vs. Gradient

Where is the edge in this picture?



Localizing an Edge

As we saw in the Image Representation 2 lecture, we can detect edges with the spatial derivative of the image surface, and can define the direction of that edge by the gradient:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial x}\right]$$

Localizing an Edge

As we saw in the Image Representation 2 lecture, we can detect edges with the spatial derivative of the image surface, and can define the direction of that edge by the gradient:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial x}\right]$$

To localize the edge explicitly, we can calculate the extrema of the derivative, which we can find by taking the second derivative:

$$\nabla \cdot \nabla f = \nabla^2 f$$

This operation is known as the *Laplacian*.

Laplacian of Gaussians

As we saw previously, it is helpful to combine a Gaussian kernel with the derivative to better incorporate local context and reduce the impact of noise on edge detection. This approach remains valid for the Laplacian, and taking the Laplacian of a Gaussian is sometimes referred to as a LoG filter.

As before, this can be equivalently computed using separable filters.

$$\nabla^2 G_\sigma(x,y) = \frac{1}{\sigma^3}\left(1-\frac{x^2}{2\sigma^2}\right)G_\sigma(x)G_\sigma(y) + \frac{1}{\sigma^3}\left(1-\frac{y^2}{2\sigma^2}\right)G_\sigma(x)G_\sigma(y)$$

Localized Edges and Further Reasoning

Once we have localized edge information, we can begin trying to link those edge elements (sometimes referred to as *edgels*) into more complete contours.

This will be explored more next class with a look at the subtopic *Lines*.