Camera Feature Detection

- Motivation: understanding images is really hard!
  - Lots of data
  - Human brain does bottom-up and top-down processing

- Idea: a lot of the information is concentrated in a few areas
  - Just look at those areas!
  - How do we find those areas?
Corner Detectors

- Perhaps corners can help us
  - Relatively easy to find
  - Trackable
  - Recognizable (?)

- Wait… what is a corner?
  - Not just a corner of a physical object
  - Any pixel pattern that is well localized in all directions
    - Not a uniform area
    - Not a line or edge
  - We could hard-code a bunch of patterns, but that's not appealing (especially for large image patches!)

Image Gradients

- Idea: let’s look at gradients of a patch of pixels
  - Gradient at pixel a is (b-a, c-a)

- Compute gradients for 2x2 area
  - We need 3x3 input…
Image Gradients

Is this a good corner? What are the gradients?

Is this a good corner? What are the gradients?

Is this a good corner? What are the gradients?

Is this a good corner? What are the gradients?

Is this a good corner? What are the gradients?

Is this a good corner? What are the gradients?
Good and Bad Corners

- **Good Corners**
- **Bad Corners**

**How can we identify good corners from their gradients?**

Gradient Plots

- **Good Corners**

The ellipses are the sample covariance of the points, i.e., the sum of the outer products.

\[ S = \sum_i g_i g_i^T \]
Gradient Plots

- Bad Corners
  - The magnitude of the ellipse’s axes are the _________ of the matrix S.
  - Basic method: Compute matrix S at each pixel, compute the ________, report corners whose minimum magnitude is greater than a threshold.

Harris Corner Detector

- Good Corners
- Bad Corners

- Idea: Ellipses for good corners have two strong axes.
Harris Corner Problems

- Computing eigenvalues is slow
- Harris Corners are not isotropic
- Discrete gradient suffers from nasty non-linear quantization effects (bad gradient estimates!)
- Not scale invariant

Harris Corners: Faster Eigenvalues

- Computing eigenvalues is slow
  - Okay, so let's use easy-to-compute quantities.
    \[
    \text{trace}(S) = \lambda_1 + \lambda_2 \\
    \text{det}(S) = \lambda_1 \lambda_2
    \]
  - Common heuristic:
    \[
    M = \text{det}(S) - \kappa \text{trace}(S)^2
    \]
    - (mostly use the area of the ellipse, with a penalty for ellipses with eccentricity.)
Harris Corner: Isotropicity, Poor gradients

- Two problems:
  - Not isotropic
  - Discrete gradient doesn’t always compute quite the best answer

- One solution:
  - Perform a Gaussian blur on the image
    - Greatly improves continuousness of gradients in all directions
    - Reduces effects of discrete gradient computations, which are the source of anisotropic behavior.
  - Each gradient becomes the weighted sum of those gradients around it

Harris Corners: Scale Invariance

- Corners of high-resolution images are often blurry or noisy at fine levels of detail

- Idea: run Harris corner detector on down-sampled versions of the image
  - Extract corners, blur, decimate, repeat.
Image Pyramids

- Look for features on multiple scales
  - Just repeat image processing algorithm on successively lower-resolution images
  - Must produce lower-resolution images

- Avoiding aliasing requires low-pass filters
  - Ideal low-pass filter?
  - Don’t create new features when filtering

Harris Corners: Example

- Find Harris corners at multiple scales, threshold according to strength

Low threshold

High threshold
Harris Corner: Summary

- Principled method of detecting corners
- Look for image patches whose gradients span the whole space
- With simple modifications, scale invariant, isotropic, fast.

Corners

- Now that we have corners, what can we do with them?
  - Landmarks for SLAM
  - Use them to track moving objects
  - Find registration marks on calibration targets
Improving tracking using appearance

- Suppose we identify corners from two image frames.
  - Which corners in image A correspond to those in image B?

- We could (should) use RANSAC the way we did with other point features

- Could we use the appearance of the corner itself to help matching?

Image patch matching

- Extract regions from images, compute sum of absolute/squared (SAD/SSE) differences

  - These will have a fairly small error

  - These will have a large error: may not match
Robust image patch matching

- Goal: Detect distinctive features, maximizing repeatability
  - Scale invariance
    - Robust to changes in distance
  - Rotation invariance
    - Robust to rotations of camera
  - Affine invariance
    - Robust to tilting of camera
  - Brightness invariance
    - Robust to minor changes in illumination
  - Produce small descriptors that can be compared using simple mathematical operations (SSE)

SIFT: Scale-Invariant Feature Transform

- Developed by David Lowe
  - See paper on course website
  - Very useful tool, reflexively used by vision researchers in many contexts. (Even when SIFT isn’t really a good choice.)
  - Watch out: patented, commercial use restricted.
SIFT

Algorithm outline:
- Detect interest points (aka corners)
- For each interest point
  - Determine dominant orientation
  - Build histograms of gradient directions
  - Output feature descriptor

SIFT: Interest Points

- SIFT doesn’t use Harris corners

- Instead: Difference of Gaussians
  - Compute image pyramid, subtract images subjected to different Gaussian filter sizes
  - Local maxima in DoG indicate corners and edges
  - Filter local maxima using Harris-Corner like test
Sub-Octave Image Pyramids

Difference of Gaussians
SIFT: Canonical Orientation

- We now have an interest point and want to compute a descriptor

- Begin by computing canonical orientation
  - This is where rotational invariance comes from

- Compute histogram (10 degree bins) of gradient orientations
  - Peak = canonical orientation

SIFT: Keypoint Descriptor

- "Official" SIFT uses 16x16 pixel patches, 4x4 bins, 8 histogram buckets
  - How many degrees of freedom in SIFT descriptor?
SIFT: Matching Descriptors

- Each SIFT feature:
  - $(x, y, \text{scale})$
  - descriptor[128]

- Two descriptors can be compared using Euclidean distance...
  - Small distances = similar descriptors
  - What if same/similar feature appears more than once? → nearest neighbor may not be good enough

- Common approach:
  - Suppose best match for $A_i$ is $B_j$ (with $d_{ij}$).
  - Suppose next best match for $A_i$ is $B_k$ (with $d_{ik}$).
  - Require $d_{ij} < a d_{ik}$. ($a$ typically 0.8).

SURF
SIFT: Object Recognition

SIFT: Conclusions

- SIFT very popular, often effective
- Quite slow, though some fast implementations exist (GPU versions from UNC-Chapel Hill)