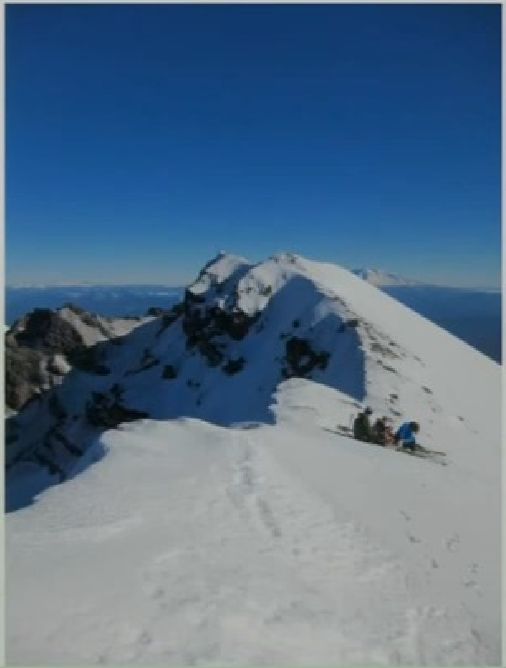
How close are two patches?

- Sum squared difference
- Images I, J
- $\Sigma_{x,y} (I(x,y) - J(x,y))^2$

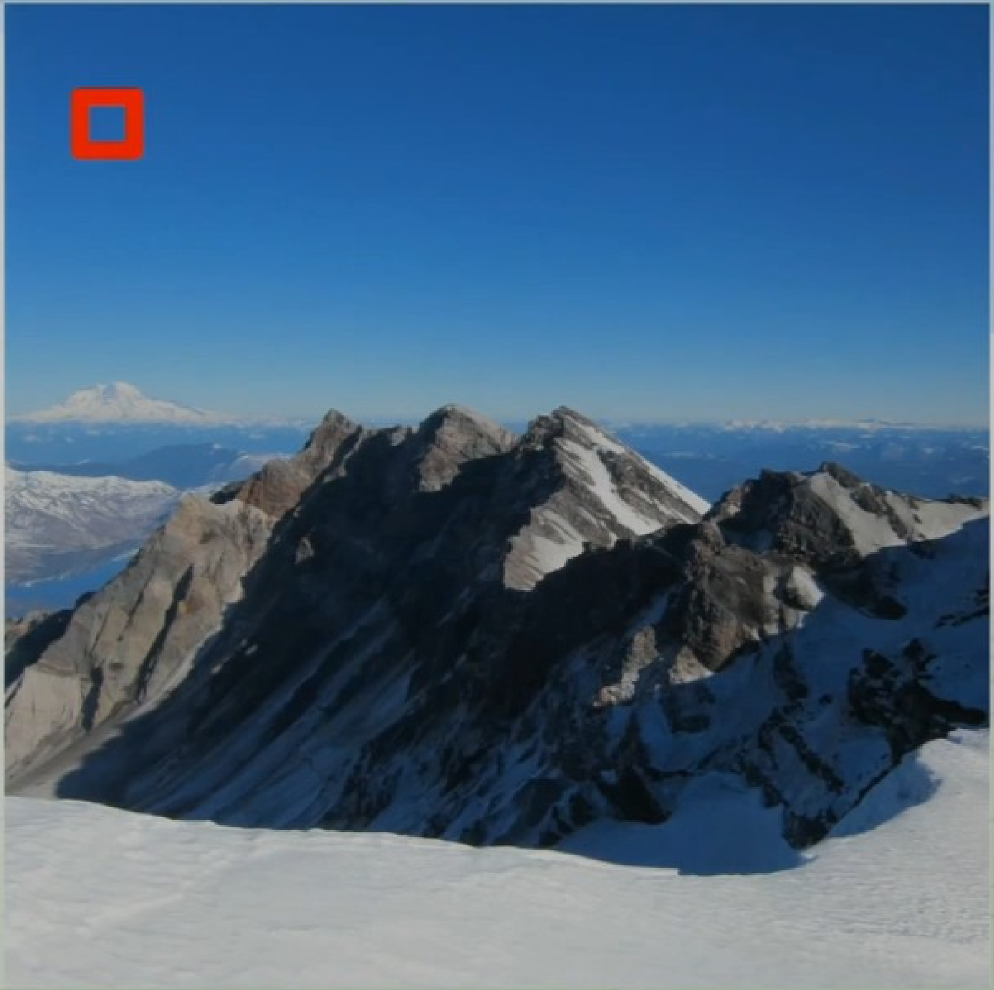
How can we find unique patches?

- Say we are stitching a panorama
- Want patches in image to match to other image
- Need to only match one spot



How can we find unique patches?

- Sky: bad
  - Very little variation
  - Could match any other sky

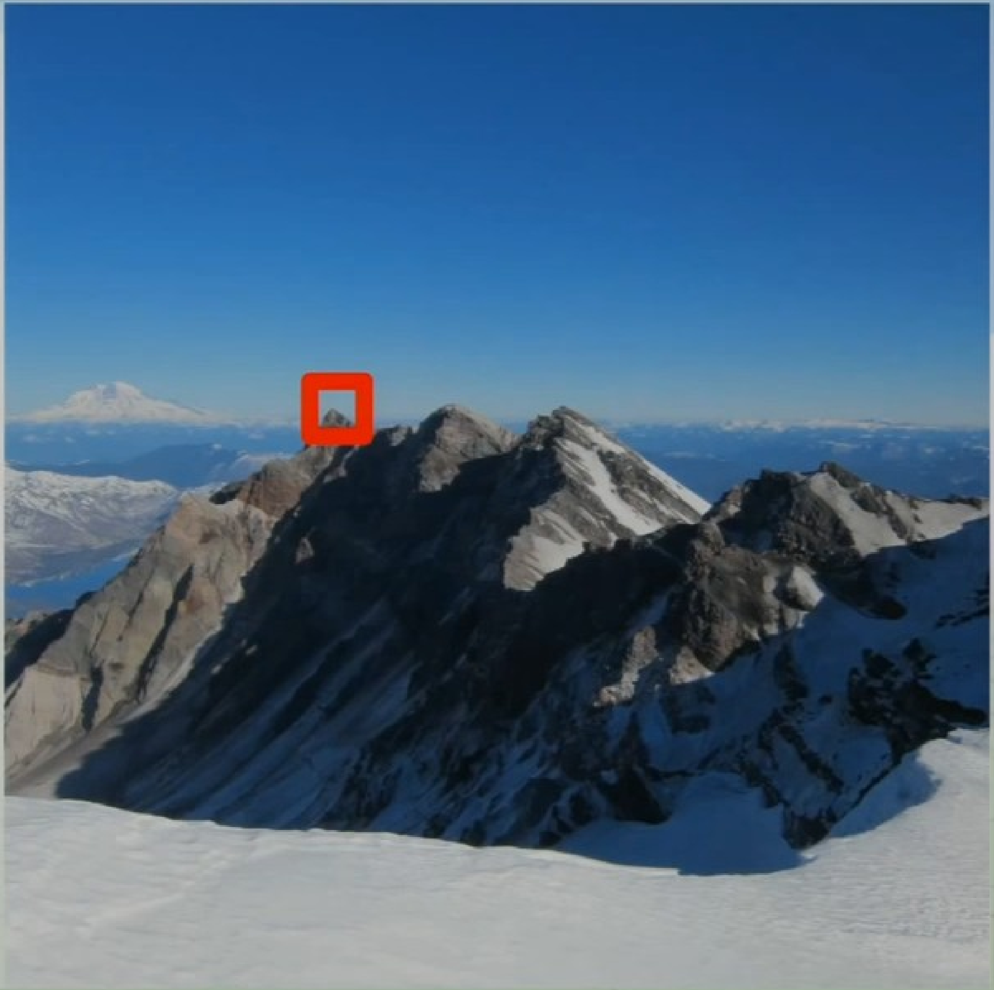


How can we find unique patches?

- Sky: bad
  - Very little variation
  - Could match any other sky
- Edge: ok
  - Variation in one direction
  - Could match other patches along same edge

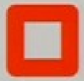
How can we find unique patches?

- Sky: bad
  - Very little variation
  - Could match any other sky
- Edge: ok
  - Variation in one direction
  - Could match other patches along same edge
- Corners: good!
  - Only one alignment matches



How can we find unique patches?

- Want a patch that is unique in the image
- Can calculate distance between patch and every other patch, lot of computation

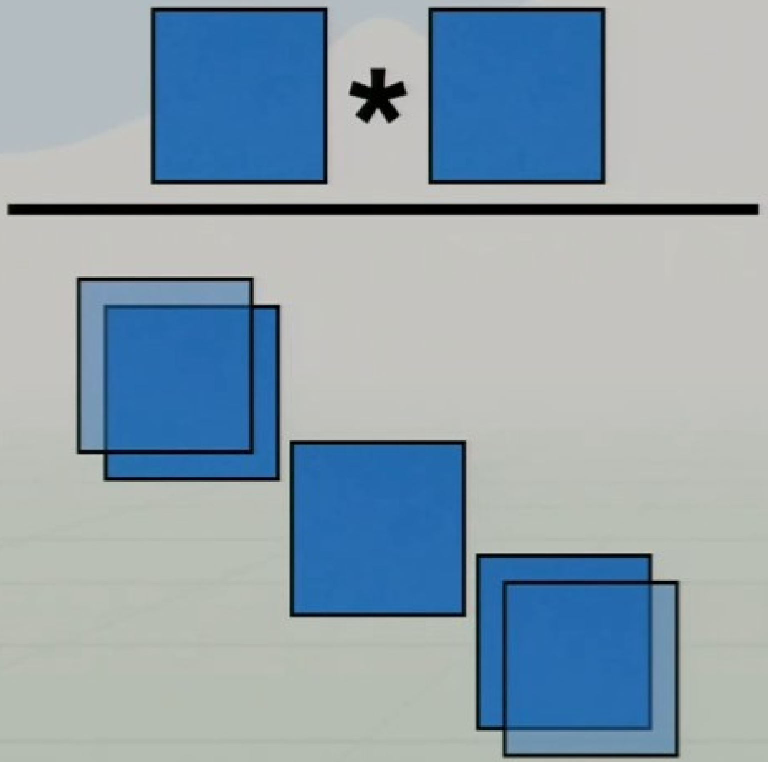


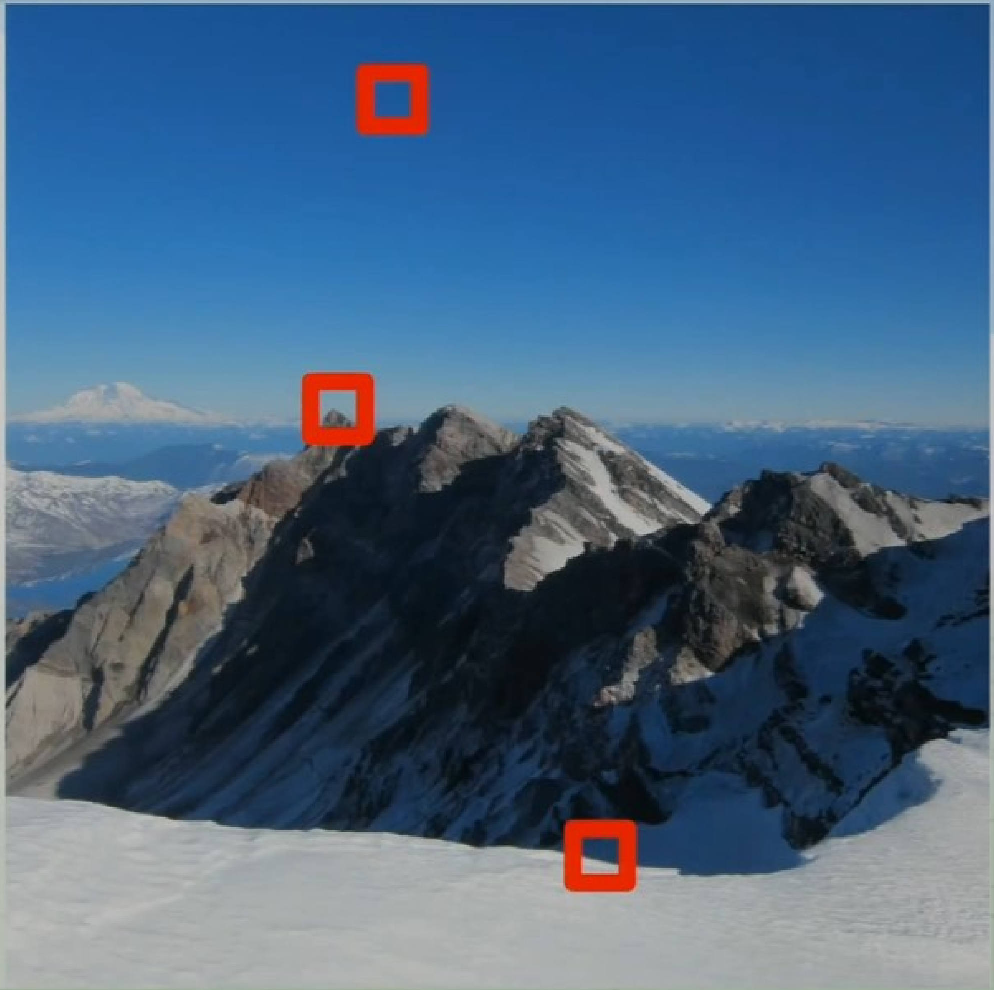
How can we find unique patches?

- Want a patch that is unique in the image
- Can calculate distance between patch and every other patch, lot of computation
- Instead, we could think about auto-correlation:
  - How well does image match shifted version of itself?
- $\Sigma_{\mathbf{d}}\Sigma_{x,y} (I(x,y) - I(x+\mathbf{d}_x,y+\mathbf{d}_y))^2$
- Measure of self-difference (how am I not myself?)

Self-difference

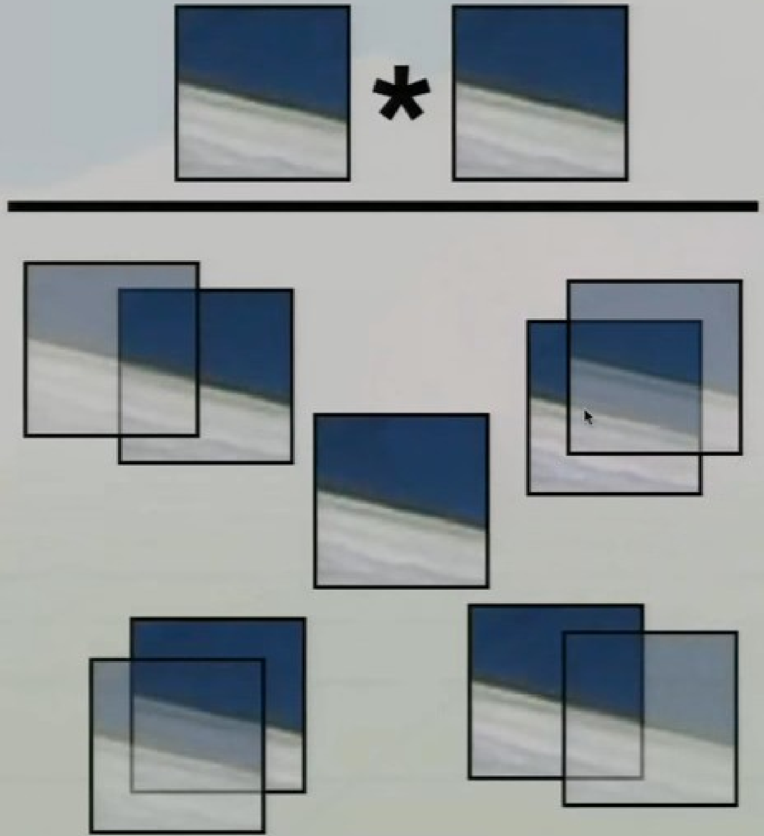
Sky: low everywhere

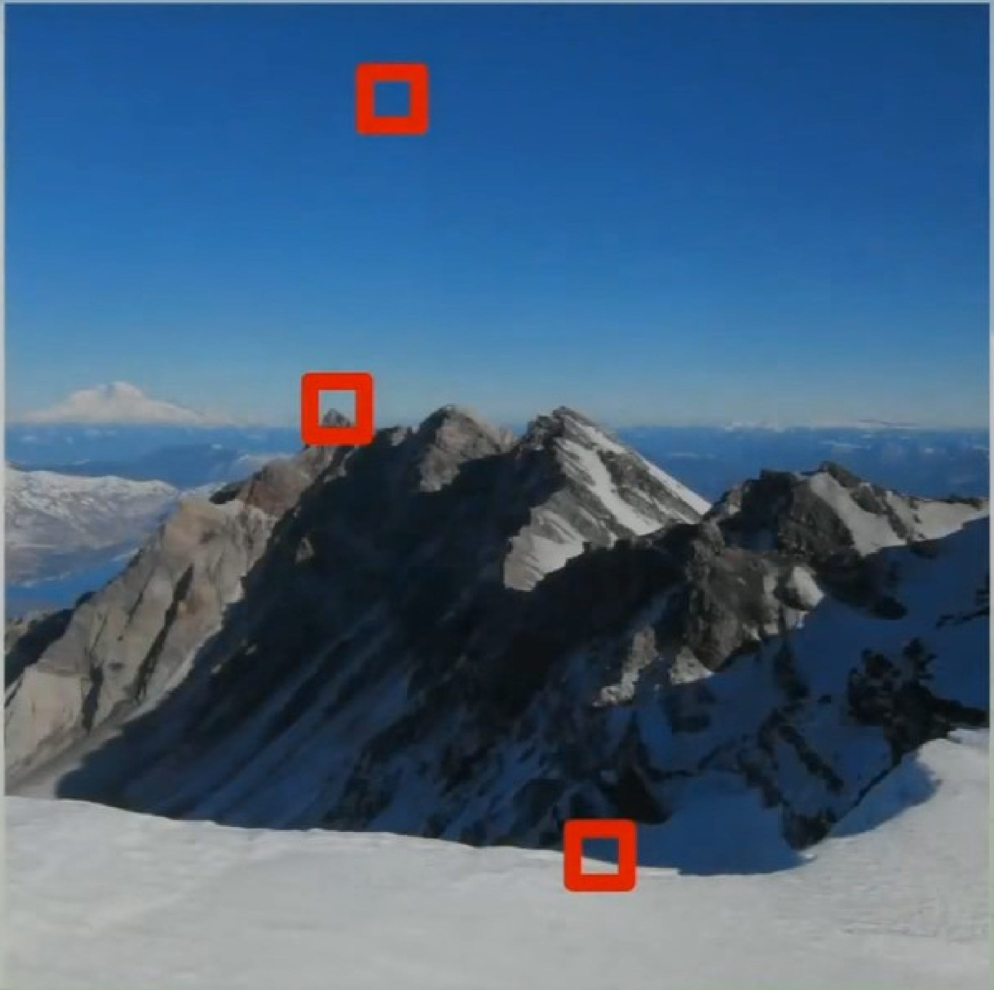




Self-difference

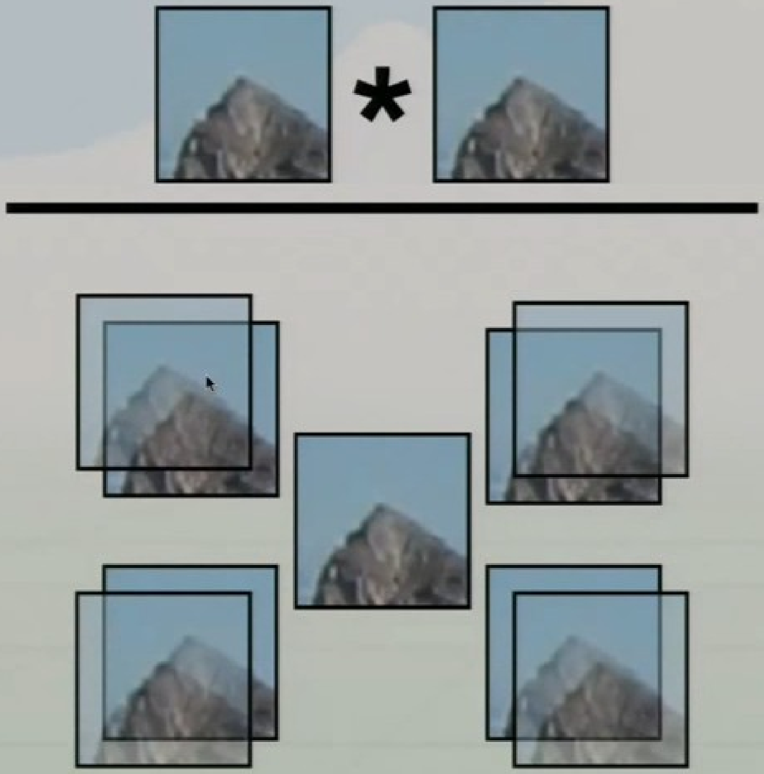
Edge: low along edge

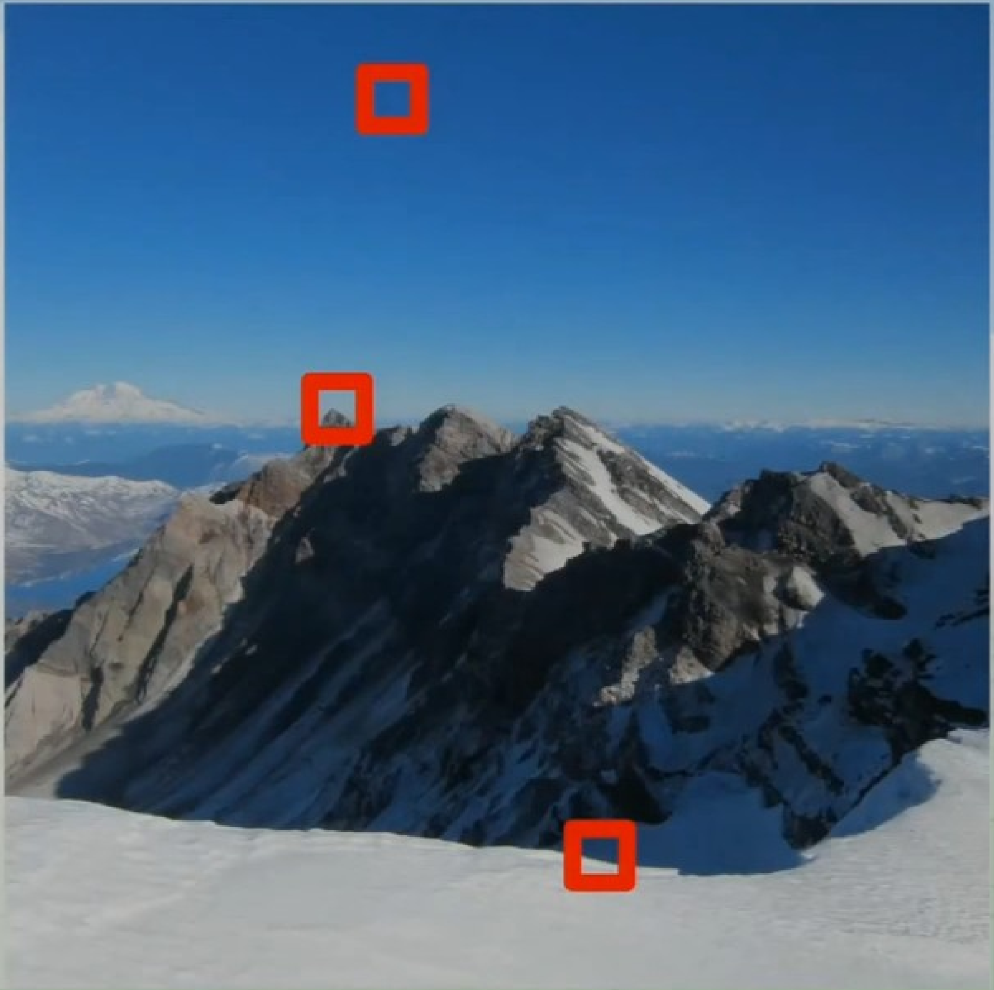




Self-difference

Corner: mostly high



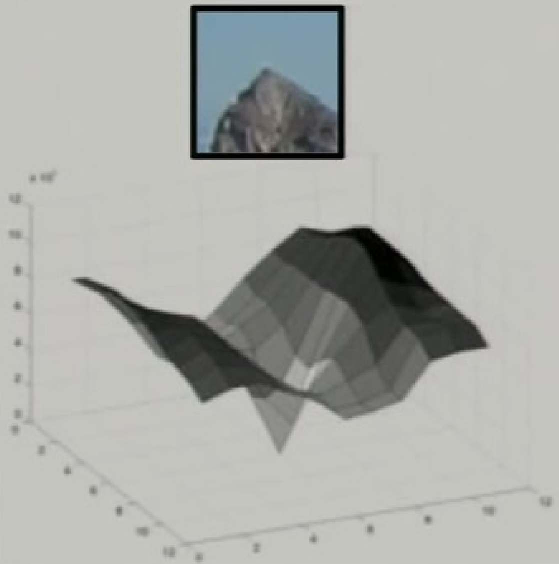
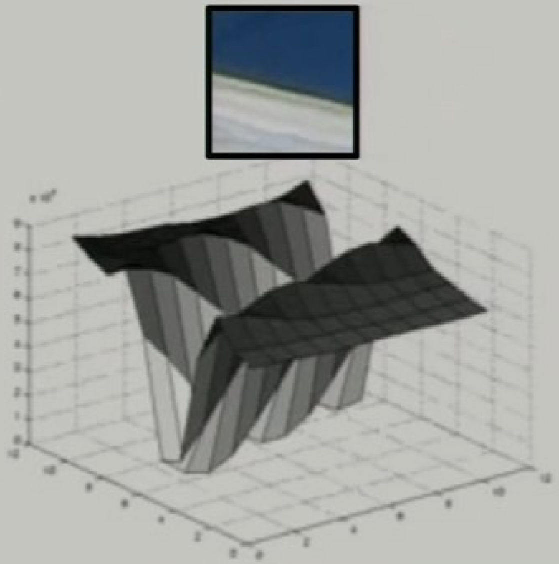
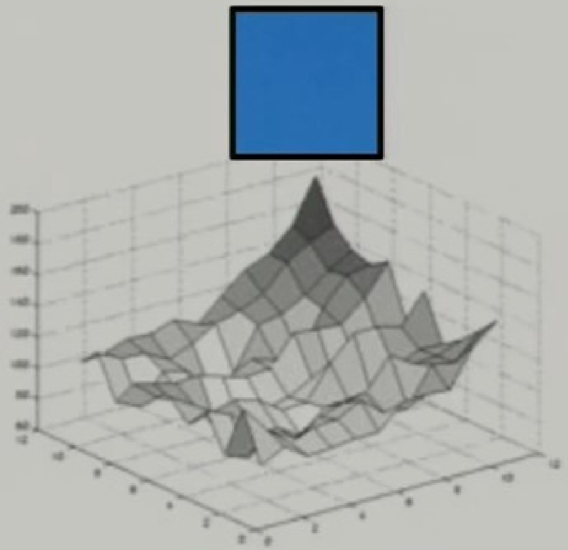


Self-difference

Sky: low everywhere

Edge: low along edge

Corner: mostly high

Self-difference is still expensive

- $\Sigma_{\mathbf{d}}\Sigma_{x,y} \ (I(x,y) - I(x+\mathbf{d}_x, y+\mathbf{d}_y))^2$
- Lots of summing
- Need an approximation
  - If you want the mathy details, Szeliski pg 212

Approximate self-difference

- Look at nearby gradients I_x and I_y
- If gradients are mostly zero, not a lot going on
  - Low self-difference
- If gradients are mostly in one direction, edge
  - Still low self-difference
- If gradients are in twoish directions, corner!
  - High self-difference, good patch!

Approximate self-difference

- How do we tell what's going on with gradients?
- Eigen vectors/values!
- Need structure matrix for patch, just a weighted sum of nearby gradient information

$$S_w[p] = \begin{bmatrix} \sum_r w[r](I_x[p-r])^2 & \sum_r w[r]I_x[p-r]I_y[p-r] \\ \sum_r w[r]I_x[p-r]I_y[p-r] & \sum_r w[r](I_y[p-r])^2 \end{bmatrix}$$

- Not as complex as it looks, weighted sum of gradients near pixel

Structure matrix

- Weighted sum of gradient information
  - $| \Sigma_i w_i I_x(i) I_x(i) \quad \Sigma_i w_i I_x(i) I_y(i) |$
  - $| \Sigma_i w_i I_x(i) I_y(i) \quad \Sigma_i w_i I_y(i) I_y(i) |$
- Can use Gaussian weighting (so many gaussians)
- Eigen vectors/values of this matrix summarize the distribution of the gradients nearby
- λ_1 and λ_2 are eigenvalues
  - λ_1 and λ_2 both small: no gradient
  - $\lambda_1 >> \lambda_2$: gradient in one direction
  - λ_1 and λ_2 similar: multiple gradient directions, corner

Estimating smallest eigen value

- A few methods:
  - $\det(S) = \lambda_1 * \lambda_2$
  - $\mathrm{trace}(S) = \lambda_1 + \lambda_2$
- $\det(S) - \alpha\ \mathrm{trace}(S)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1+\lambda_2)^2$
- $\det(S) / \mathrm{trace}(S) = \lambda_1\lambda_2/(\lambda_1+\lambda_2)$
- If these estimates are large, λ_2 is large

Harris Corner Detector

- Calculate derivatives I_x and I_y
- Calculate 3 measures I_xI_x, I_yI_y, I_xI_y
- Calculate weighted sums
  - Want a weighted sum of nearby pixels, guess what this is?
  - Gaussian!
- Estimate response based on smallest eigen value
- Non-max suppression (just like canny)