Manipulating and Understanding Images via Edges, Features and Alignment

Harpreet S Sawhney

Microsoft / Vision & Mixed Reality

May 7th and 12th , 2020

(Dynamic) Images Afford Two Types of Inferences

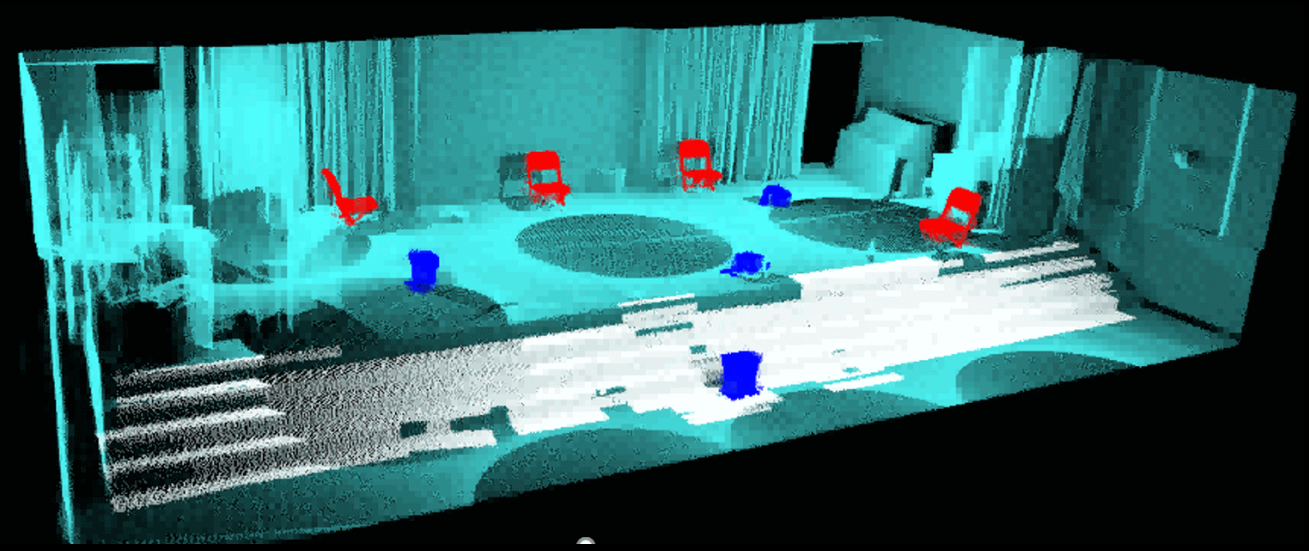
Inferences on scene content without object identity

- Frame-to-frame motions
- Foreground/background and independent object motions
- 3D structure / 2-2.5D layers
- …



Inferences on scene content involving object identity

- Object/event recognition
- Object/scene classification
- Object/scene fingerprinting
- Extended space-time tracking
- …



A Bit of Historical Perspective : My Work

Dynamic Images Afford Two Types of Inferences

- **Analysis / Manipulation of scene content without object / scene identity**

  Pixels In ➔ Pixels Out

  - Stabilization, Mosaicking, 3D Structure,…

- **Inferences on scene content involving scene /object / event identity**

  Pixels In ➔ Decisions / Labels Out

  - Model-based object/event recognition
  - Object/scene classification
  - Object/scene fingerprinting
  - Extended space-time tracking
  - Multi-modal Video Analysis

(Dynamic) Images Afford Two Types of Inferences

- **Inferences on scene content without object identity**
  - Frame-to-frame motions
  - Foreground/background and independent object motions
  - 3D structure / 2-2.5D layers
  - …

FOUR LECTURES

- **Inferences on scene content involving object identity**
  - Model-based object/event recognition
  - Object/scene classification
  - Object/scene fingerprinting
  - Extended space-time tracking
  - …

Plan of Two Lectures : May 7th (and 12th)

- Why and How of Edges in Images
- Why Image / Video Alignment and Matching
- A Swift Primer on Sum of Squares Optimization and Fitting
- Short Range Matching with Patches
- Image Transformations and Robust Fitting with RANSAC

Plan of Two Lectures : May 12th

- Long Range Matching
  - Quasi-Invariant Feature Detectors and Descriptors: SIFT, HOG
- Location and Image based Object Instance Retrieval
- Learned Features through Metric Learning with CNNs
- Long Range Matching and Localization : Contrast between Hand-crafted and Learned Features
- Take Home Lessons

What do edges tell us about the Physical World?

Figure-Ground

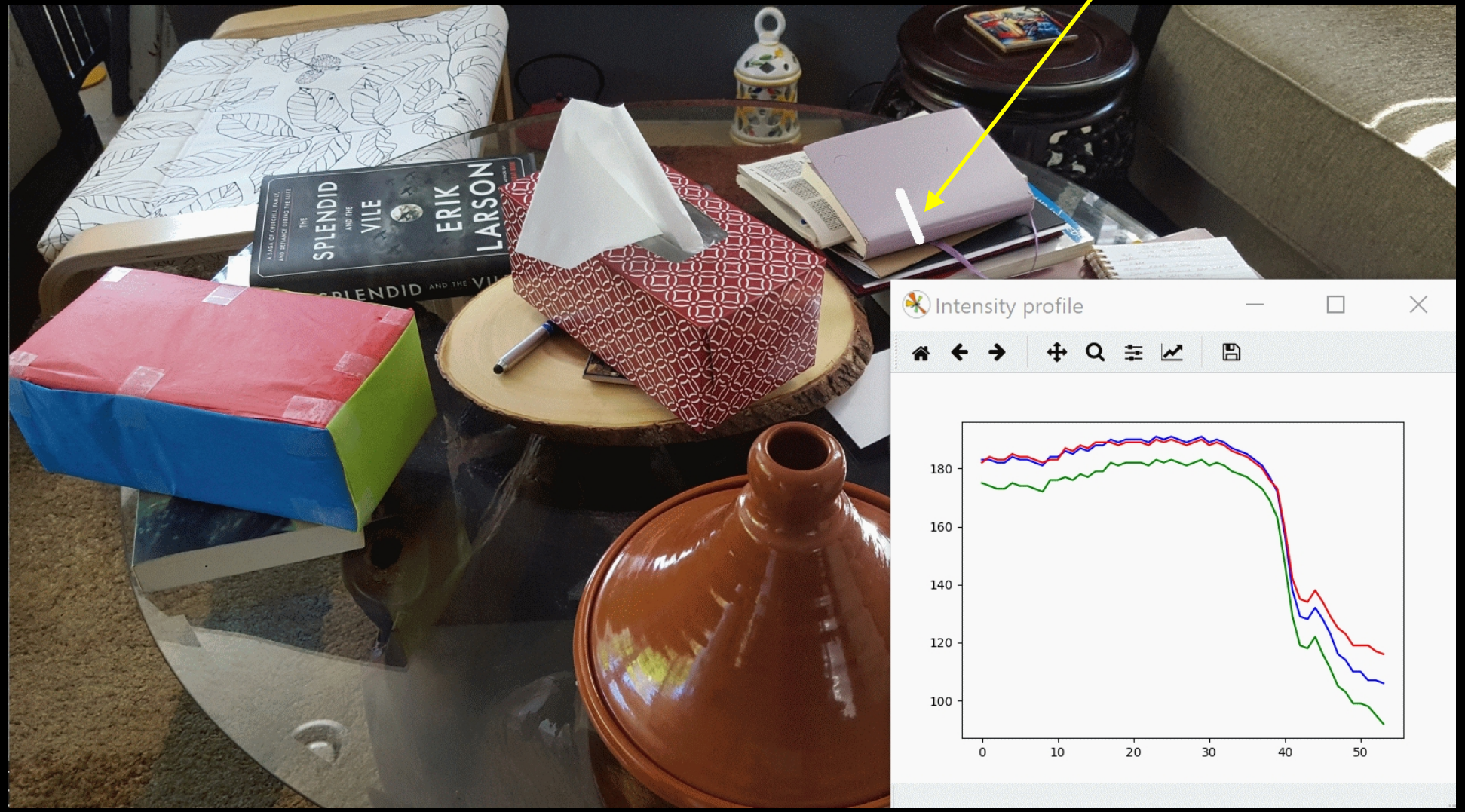


3D Perception

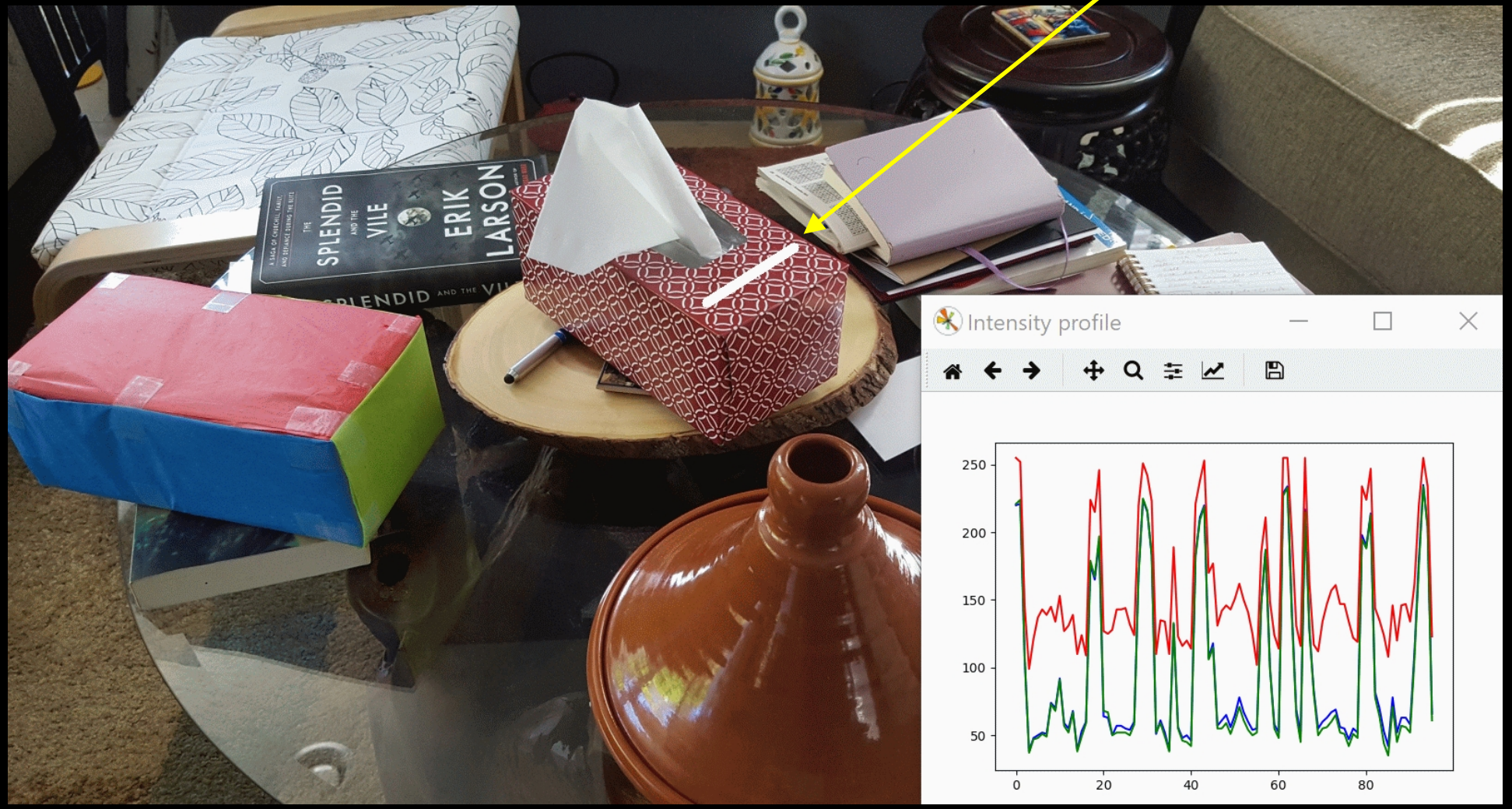



THE SPLENDID AND THE VILE
ERIK LARSON
SUDOKU
HAND SANITIZER

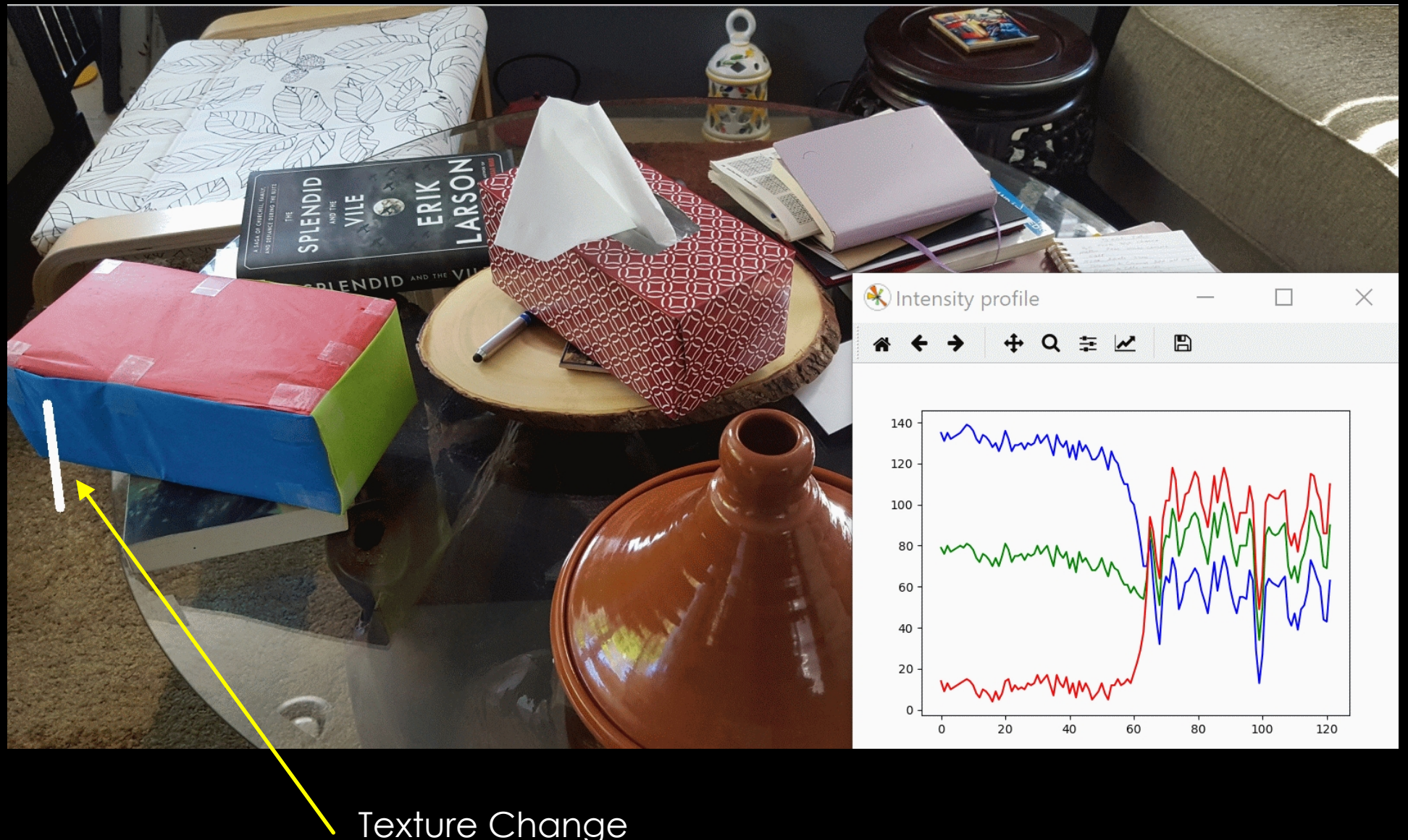
Edges have Physical / Geometric causes



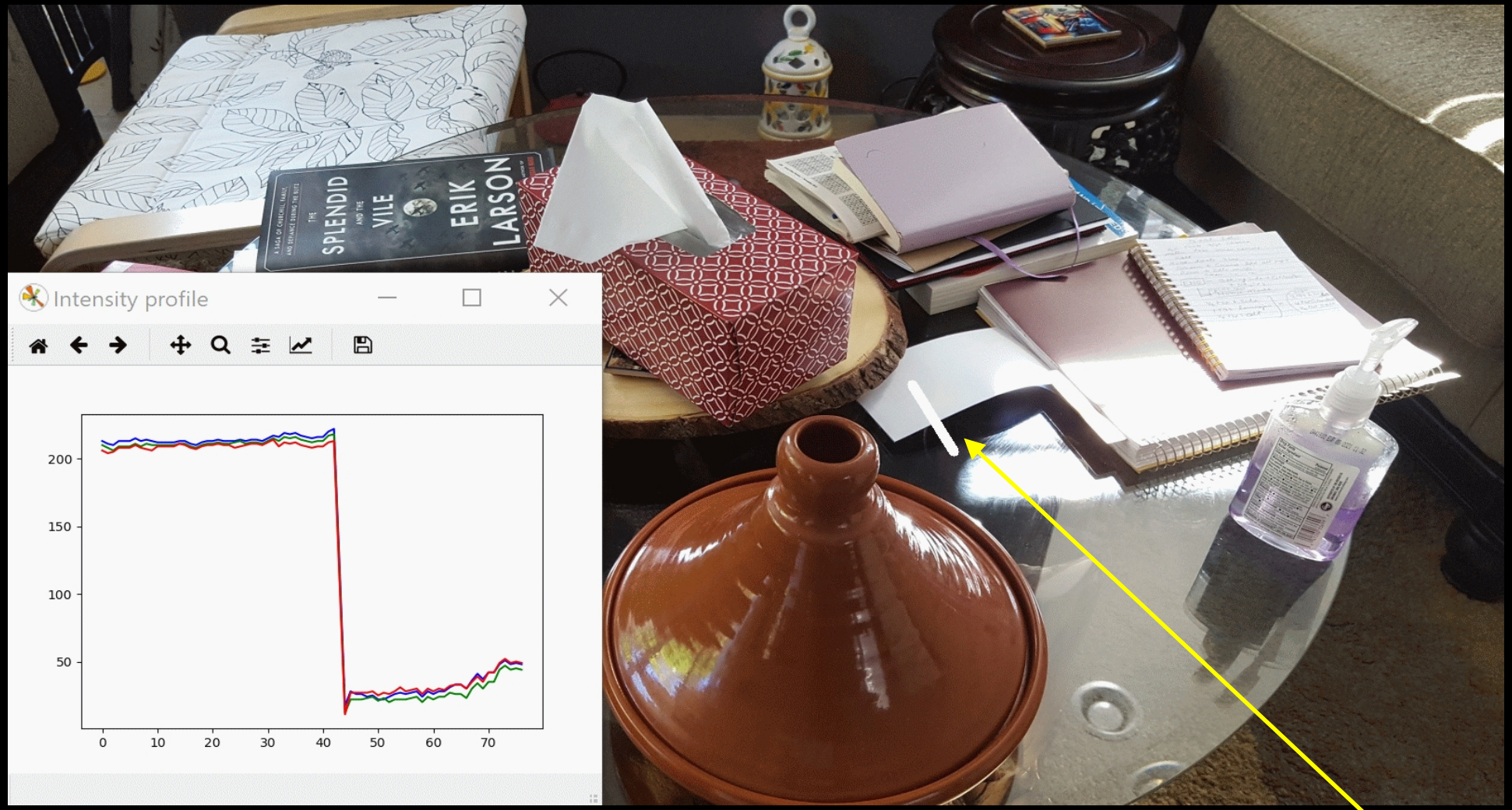
Edges have Physical / Geometric causes



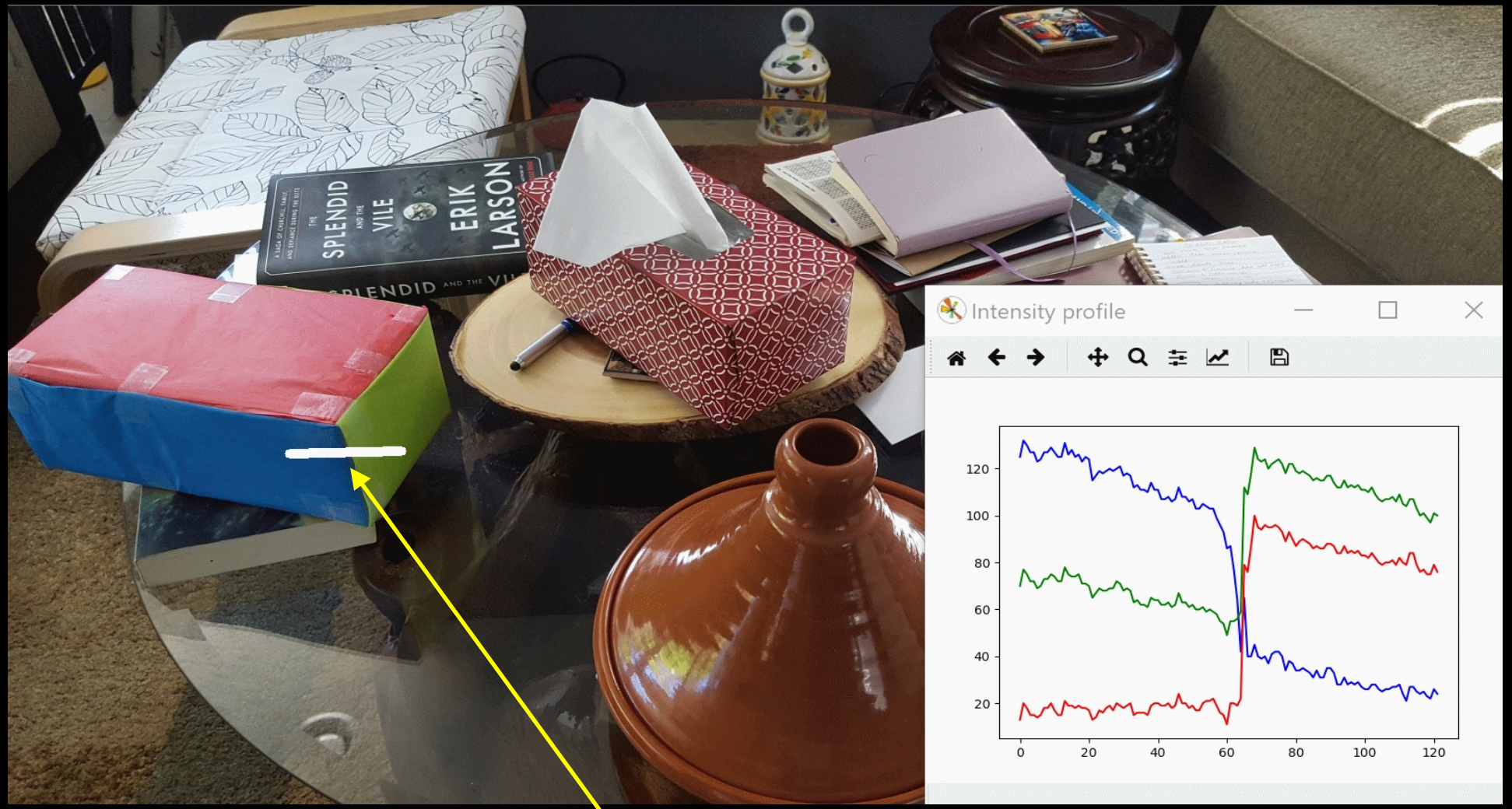
Edges have Physical / Geometric causes



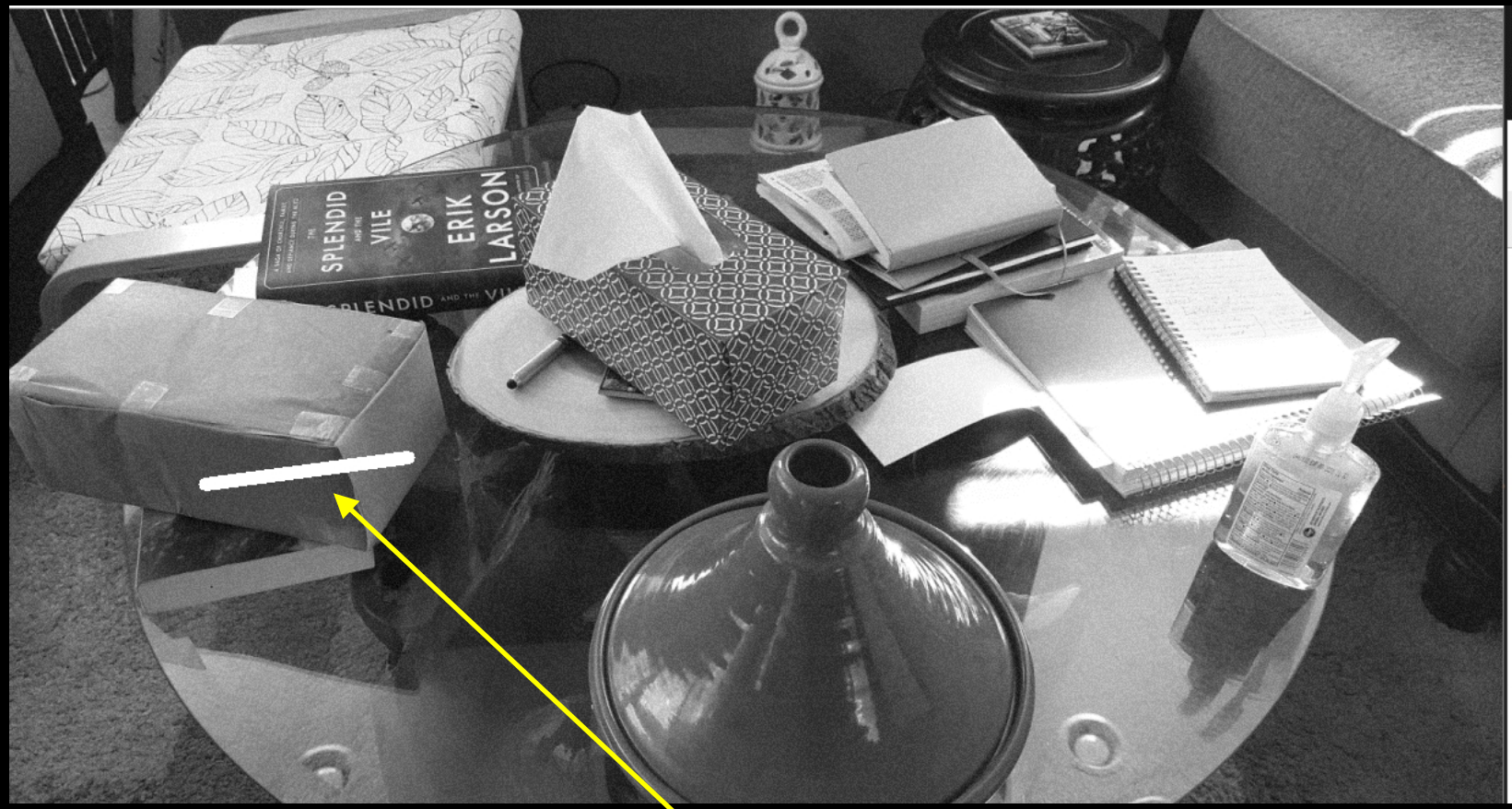
Edges have Physical / Geometric causes

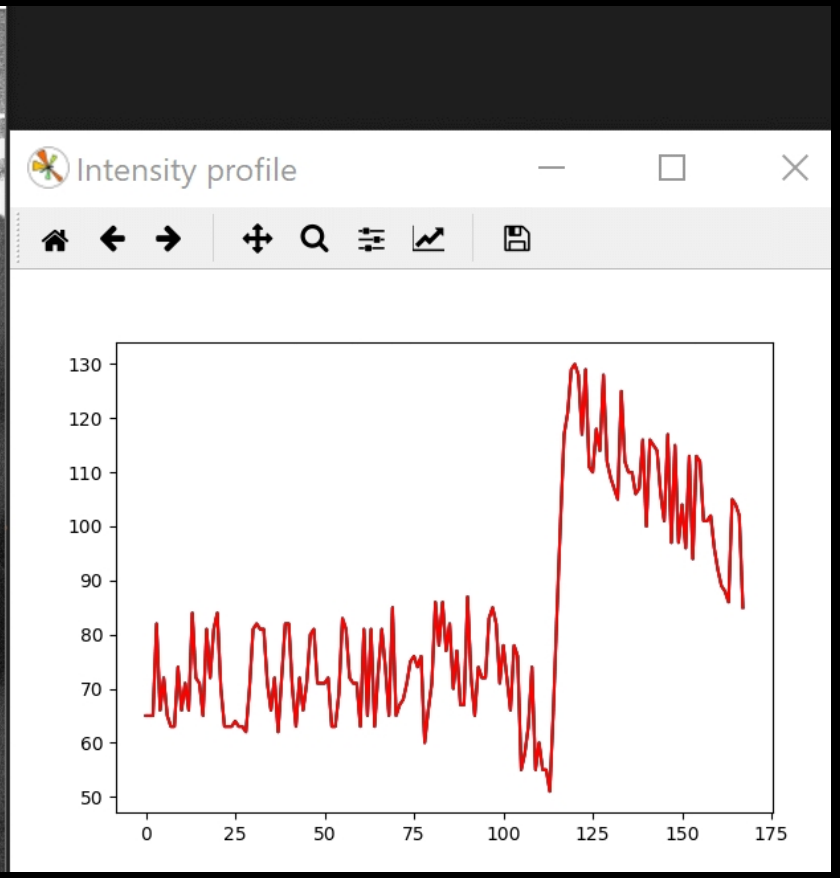


Edges have Physical / Geometric causes



Edges have Physical / Geometric causes

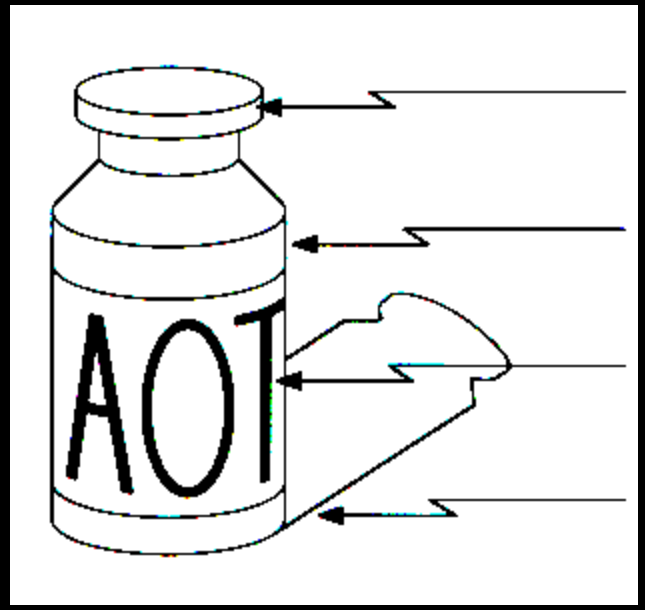




Noisy Edge


SPLENDID
VILE
ERIK
LARSON

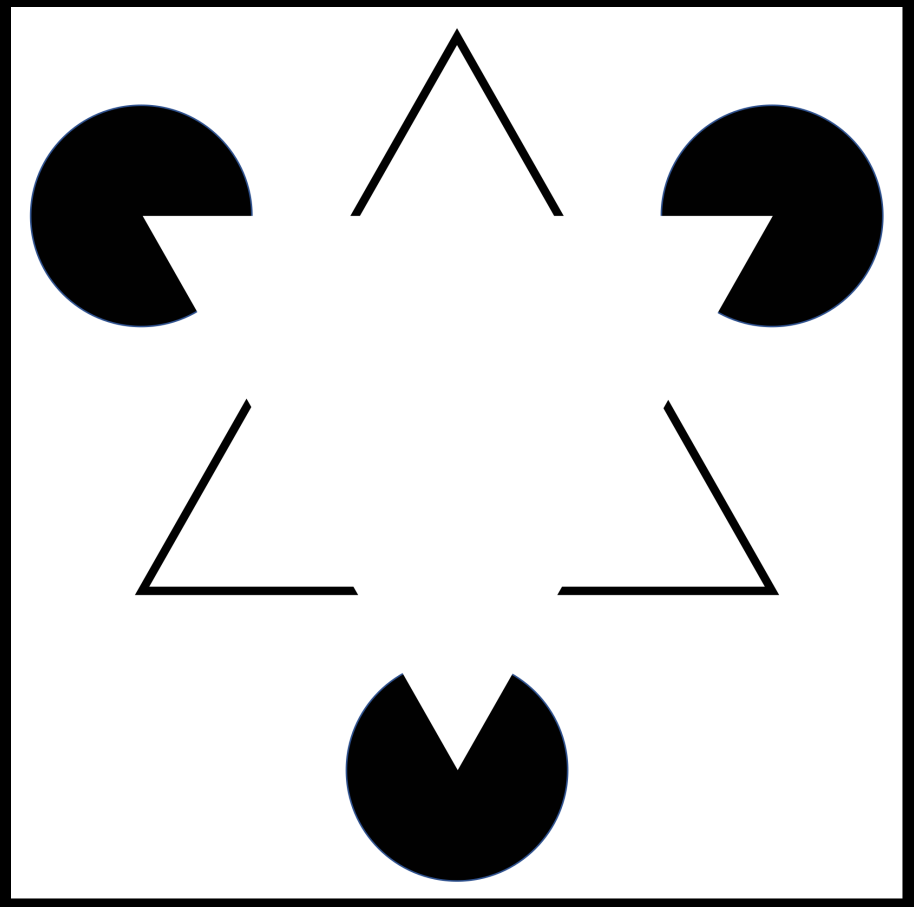
Origin of Edges



Edges are caused by a variety of factors

Source: Steve Seitz

Subjective Contours: 3D Perception / Perceptual Organization





Complex Subjective “Contours”

High-Level meets Low-Level ?



Early Work In Computer Vision: Interpretation of Line Drawings as 3D

MACHINE PERCEPTION OF THREE-DIMENSIONAL SOLIDS

by

LAWRENCE GILMAN ROBERTS

S.B., Massachusetts Institute of Technology
(1961)

M.S., Massachusetts Institute of Technology
(1961)

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June, 1963

Signature of Author _____

Certified by _____
Thesis Supervisor

Accepted by _____
Chairman, Departmental Committee on Graduate Students

Machine Perception of 3D Solids

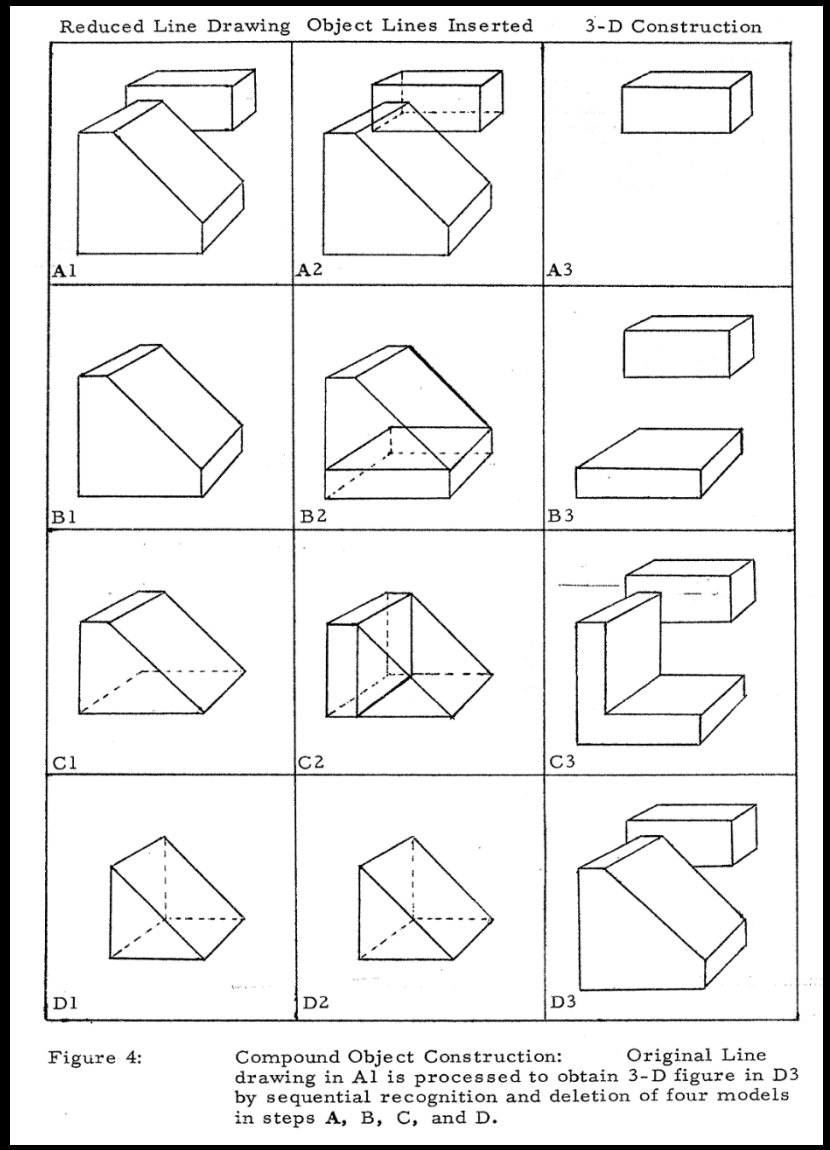


Figure 4: Compound Object Construction: Original Line drawing in A1 is processed to obtain 3-D figure in D3 by sequential recognition and deletion of four models in steps A, B, C, and D.

Compound Object Construction

International Journal of Computer Vision, 1, 73–103 (1987)
© 1987 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands

Interpreting Line Drawings of Curved Objects

JITENDRA MALIK
Computer Science Division, Department of EECS, University of California, Berkeley, CA 94720

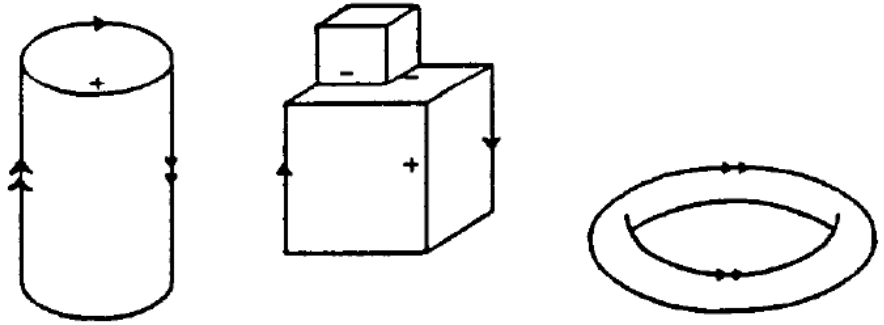


Fig. 1. **Different kinds of line labels.**

"+" label represents a convex edge

"-" label represents a concave edge

→ or a ← represents an occluding convex edge

← or a → represents a limb

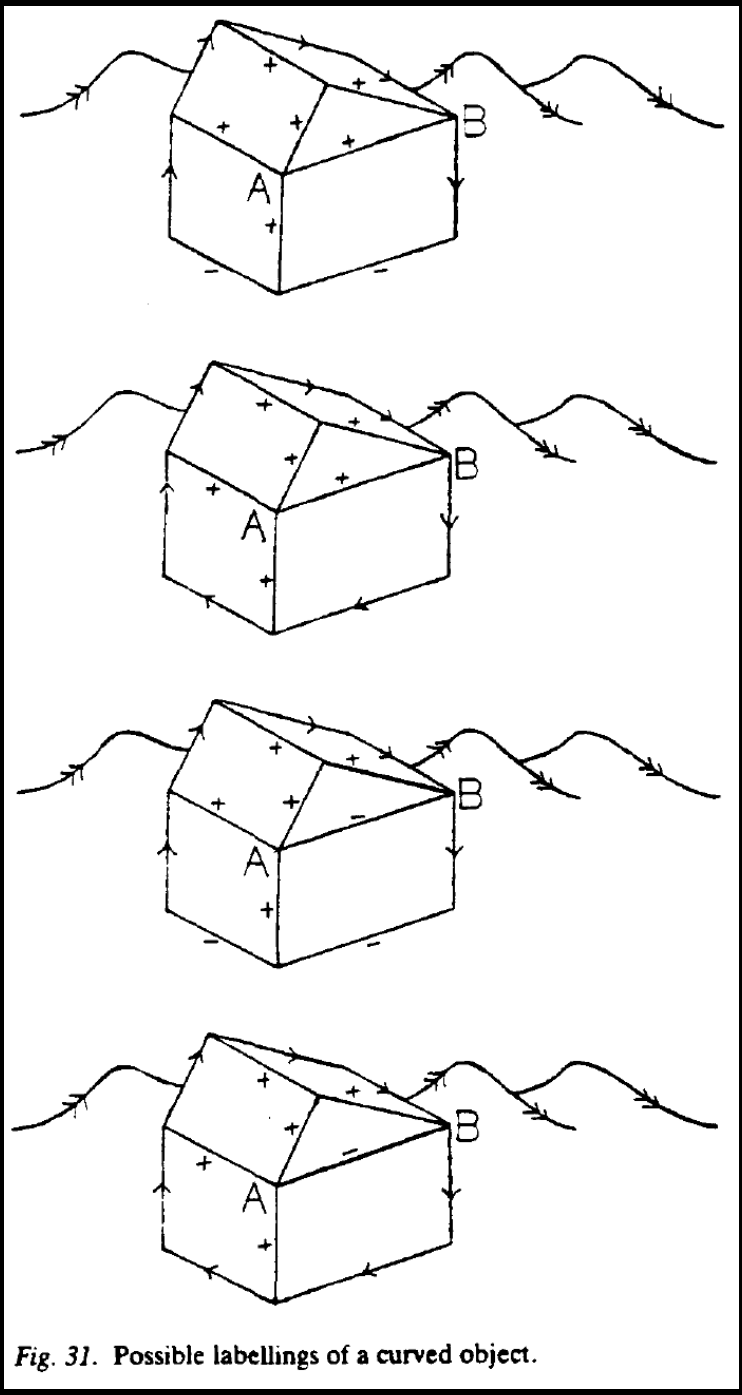


Fig. 31. **Possible labellings of a curved object.**

Perceptual Organization

PERCEPTUAL COMPLETION OF OCCLUDED SURFACES

A Dissertation Presented

by

LANCE R. WILLIAMS

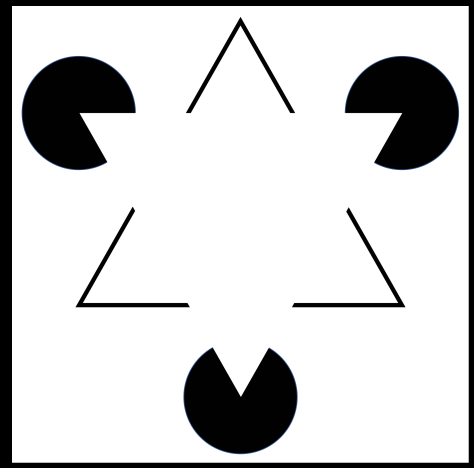
PREVIEW

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 1994

Department of Computer Science

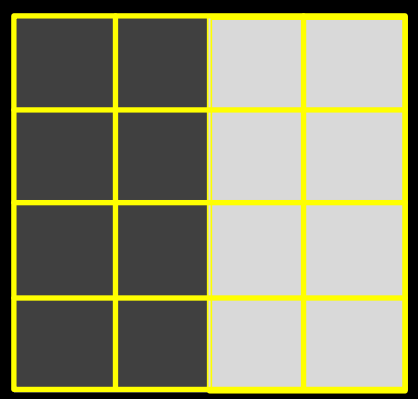


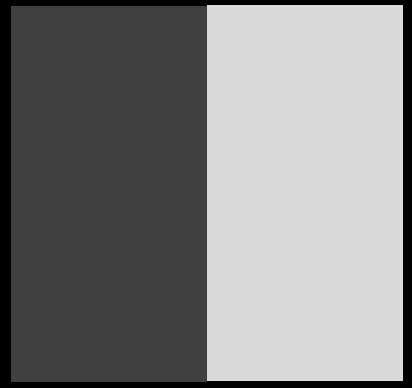
- We evolved to infer the World as 3D because you do not want to mistake a tiger in the bushes as a painting on the Wall!
- Perceptual Organization makes sense the World as Figure-Ground / 3D Percepts

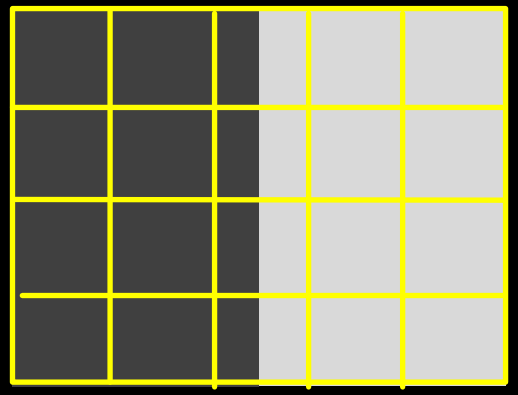
Perceptual Organization (Lance's view)

- Perceptual organization is the process of deriving the topological component of a representation of visible and partially occluded surfaces.
- The topological component is represented using labeled knot diagrams.
- Labeled knot diagrams can be derived from images through a process of figural completion.

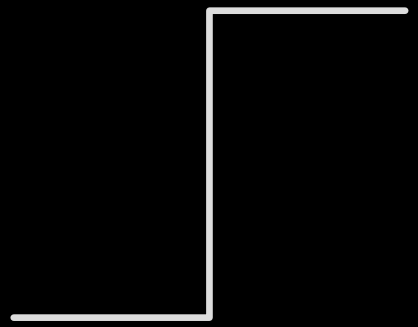
Edges correspond to changes in image intensities



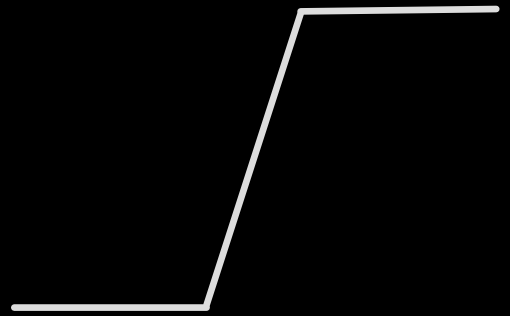




On a discrete pixel grid as above:

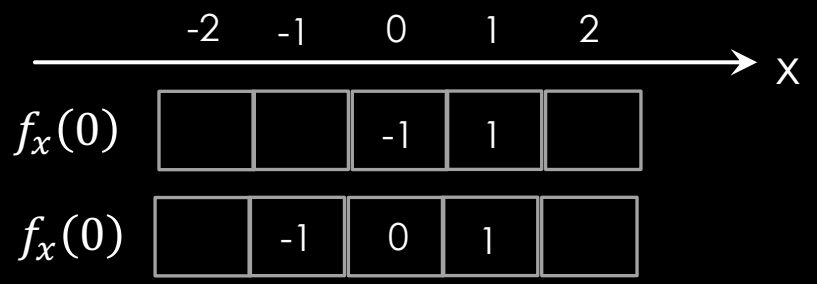






Computing Edges via Derivatives on a Pixel Grid: 1D Case for Simplicity

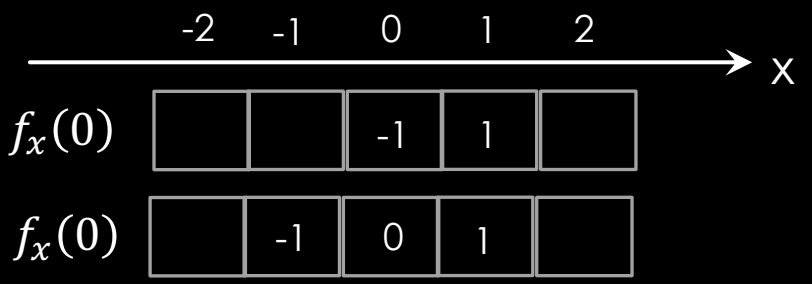
Two Approximations:



Which one is better?

Computing Edges via Derivatives on a Pixel Grid: 1D Case for Simplicity

Two Approximations:



To compute derivatives of f(x) at x=0, let us use Taylor series expansion

- $f(1) = f(0) + 1.f_x(0) + \frac{1^2}{2!}f_{xx}(0) + \frac{1^3}{3!}f_{xxx}(0) + O(4)$
- $f(-1) = f(0) - 1.f_x(0) + \frac{1^2}{2!}f_{xx}(0) - \frac{1^3}{3!}f_{xxx}(0) + O(4)$
- First Approximation: $f_x(0) = -1.f(0) + 1.f(1) = f_x(0) + \mathbf{\frac{1}{2}f_{xx}(0)} + O(3)$
- Second Approximation: $f_x(0) = \frac{1}{2}(-1.f(-1) + 1.f(1)) = f_x(0) + O(3)$

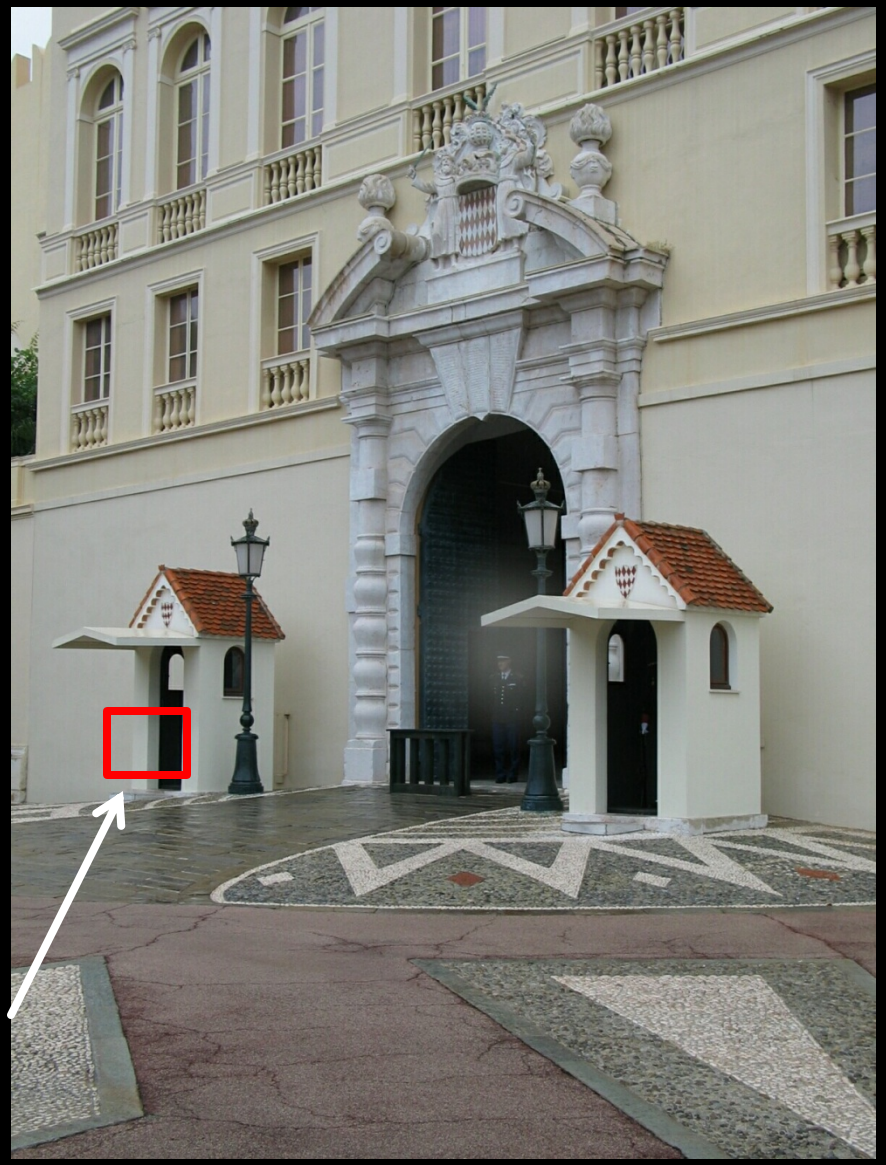
Good exercise to derive the second derivative approximation: $f_{xx}(0)$

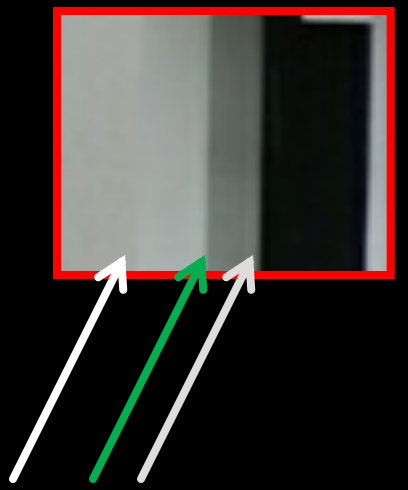
Closeup of edges



via DerekHoiem

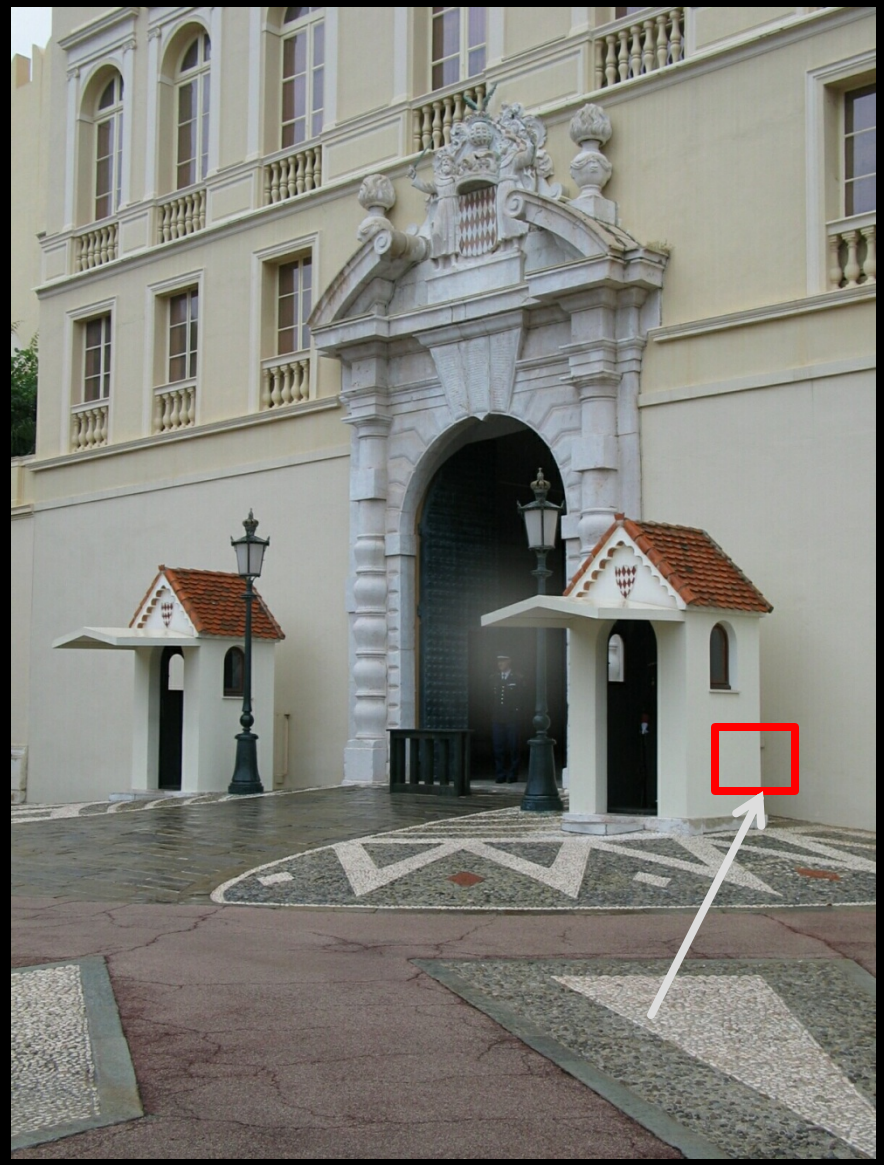
Closeup of edges

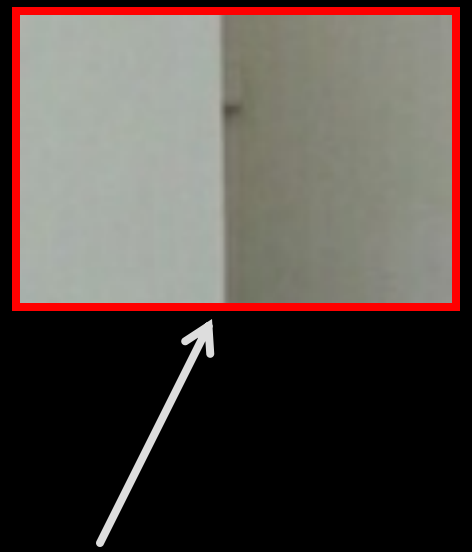




via DerekHoiem

Closeup of edges





via DerekHoiem

Closeup of edges

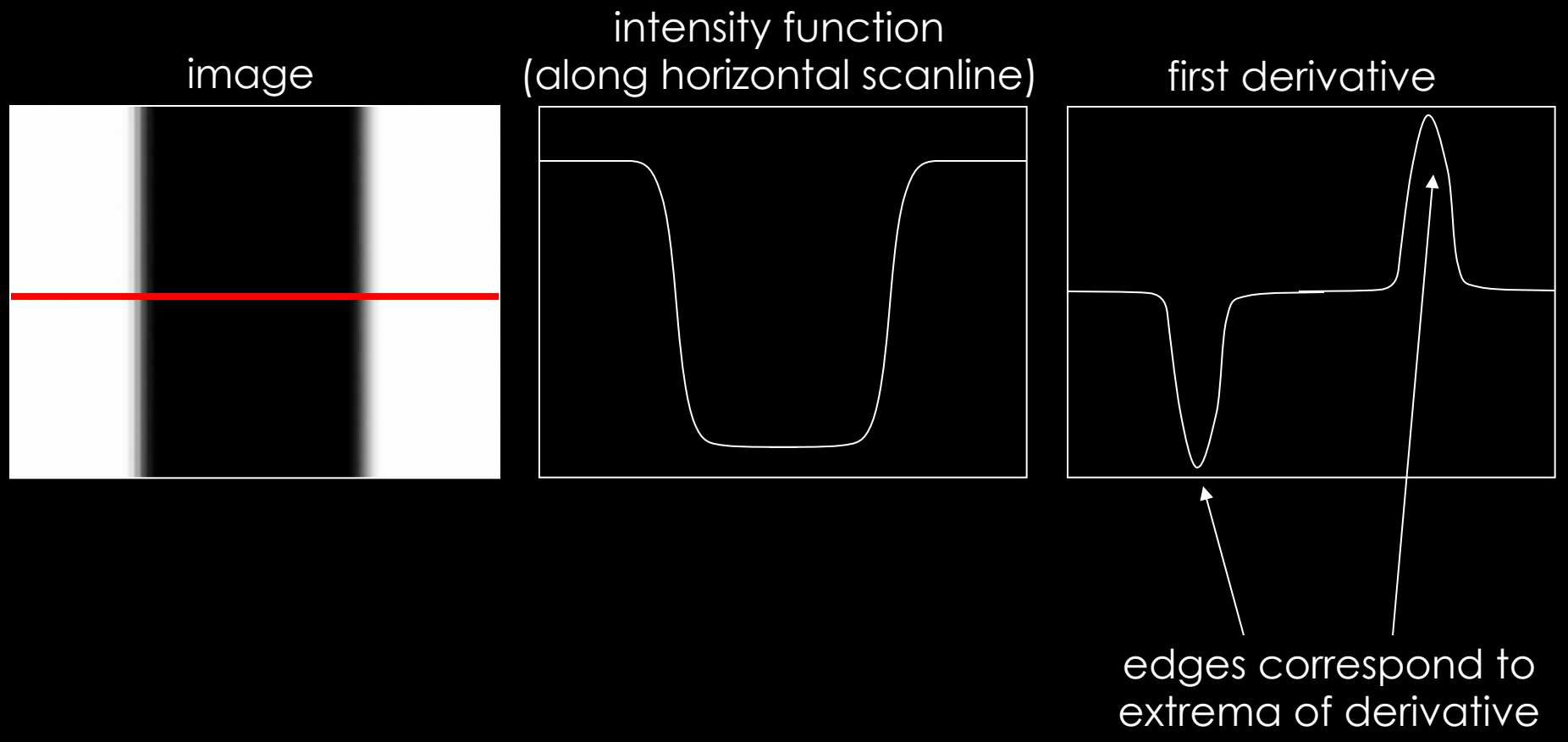




via DerekHoiem

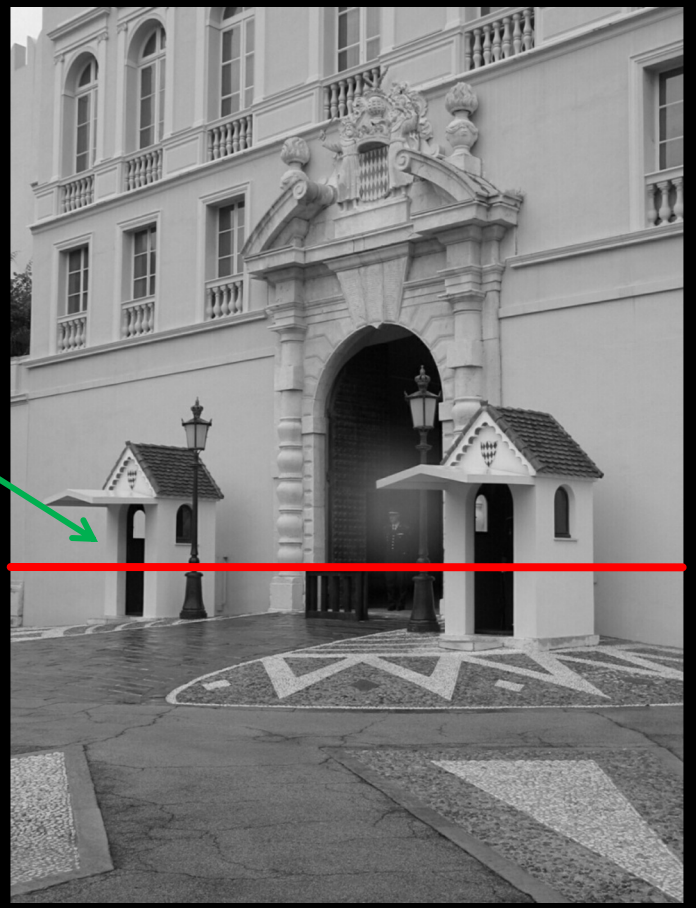
Characterizing edges

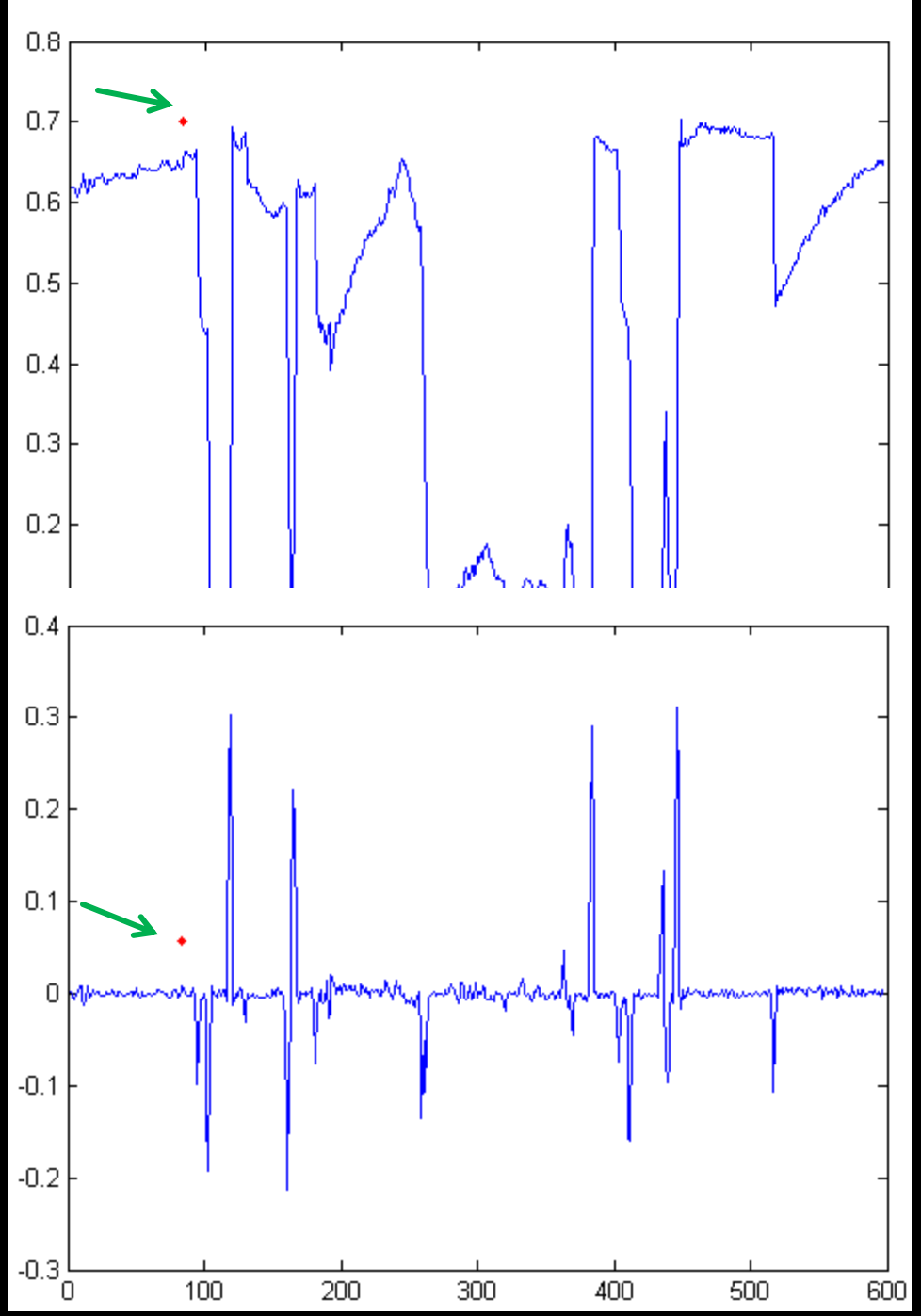
- An edge is a place of rapid change in the image intensity function



via DerekHoiem

Intensity profile

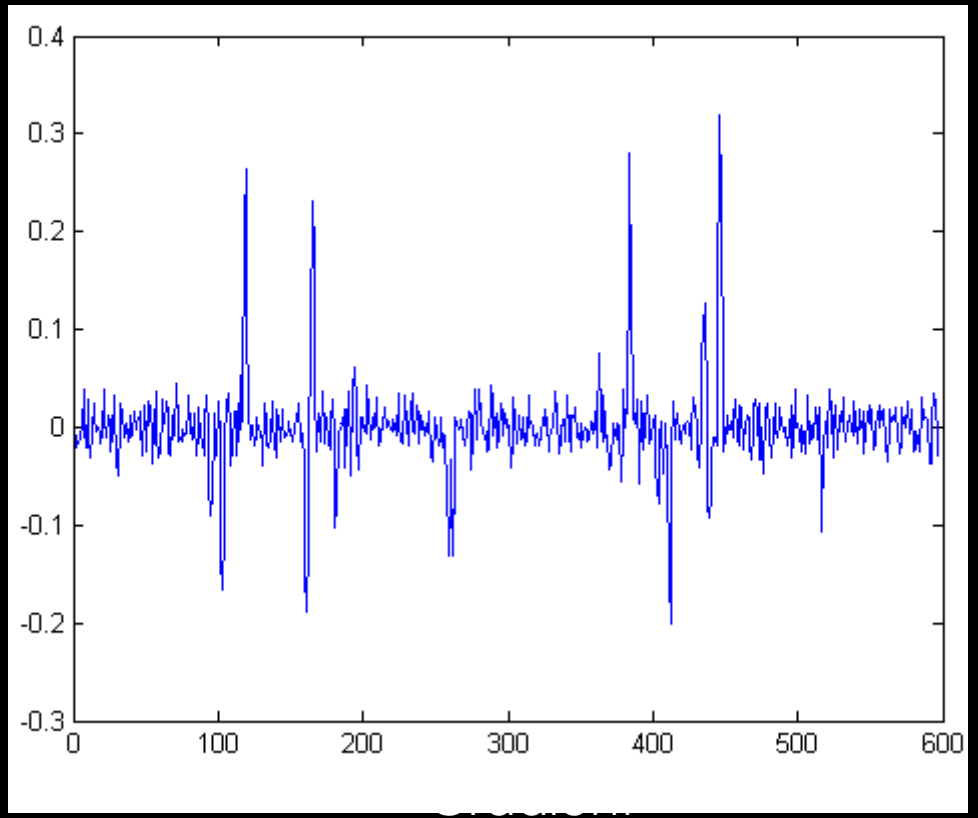




via DerekHoiem

With a little Gaussian noise





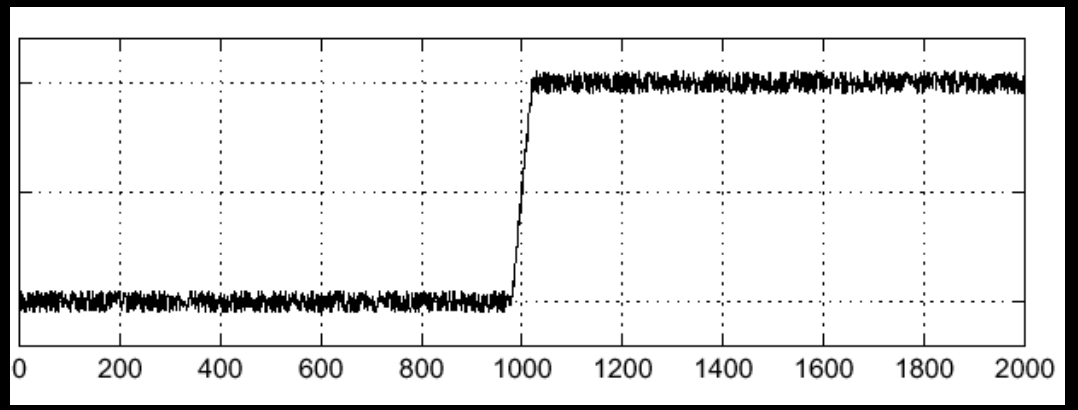
via DerekHoiem

Effects of noise

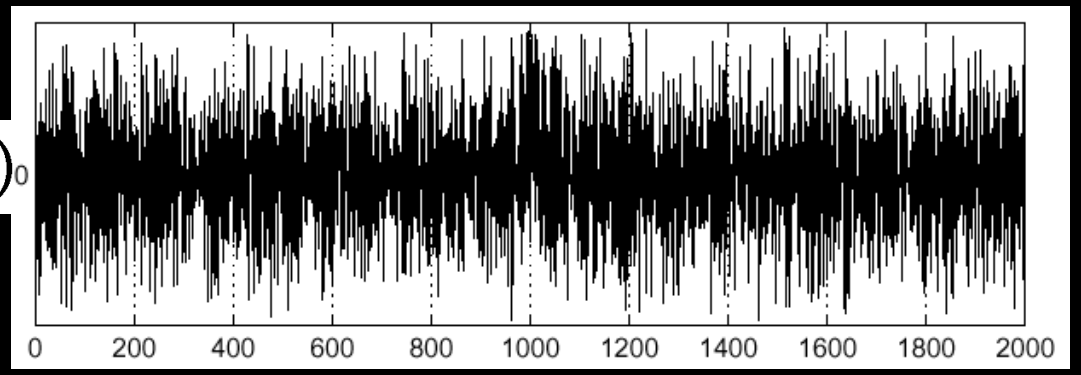
Consider a single row or column of the image

Plotting intensity as a function of position gives a signal

$f(x)$



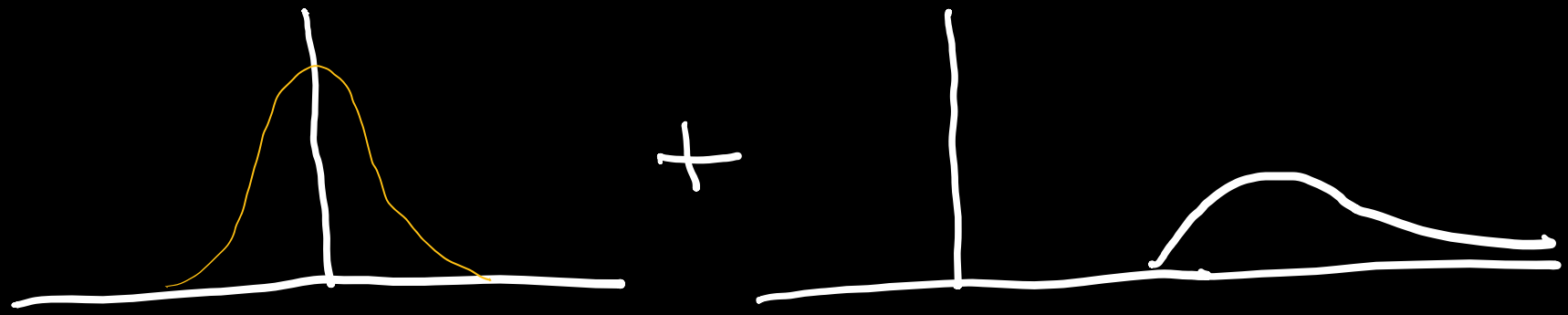
$\frac{d}{dx}f(x)$

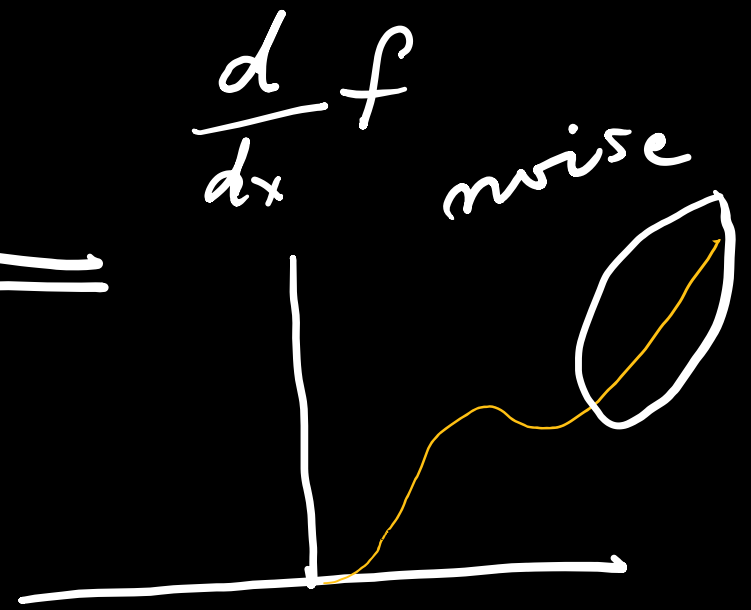


Where is the edge?

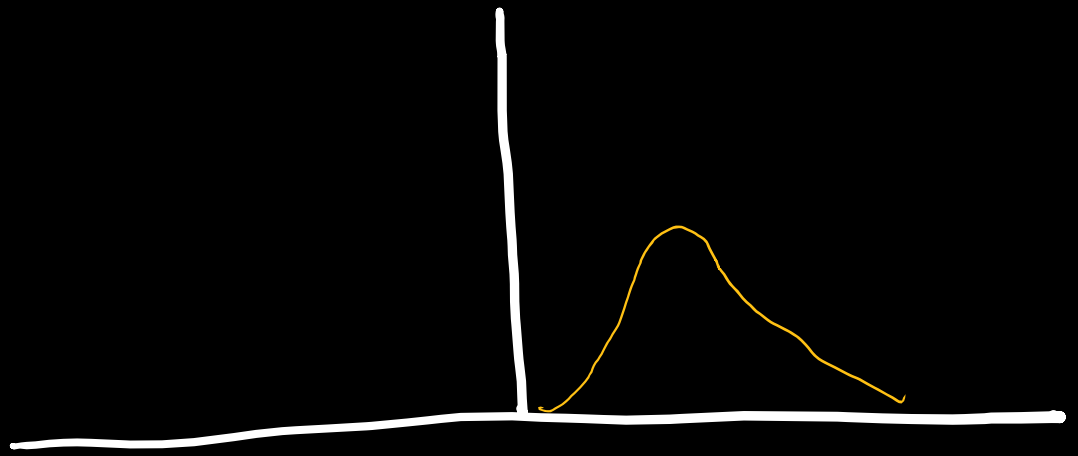
Source: S. Seitz

Differentiating noisy signals : Frequencey Domain Intuition

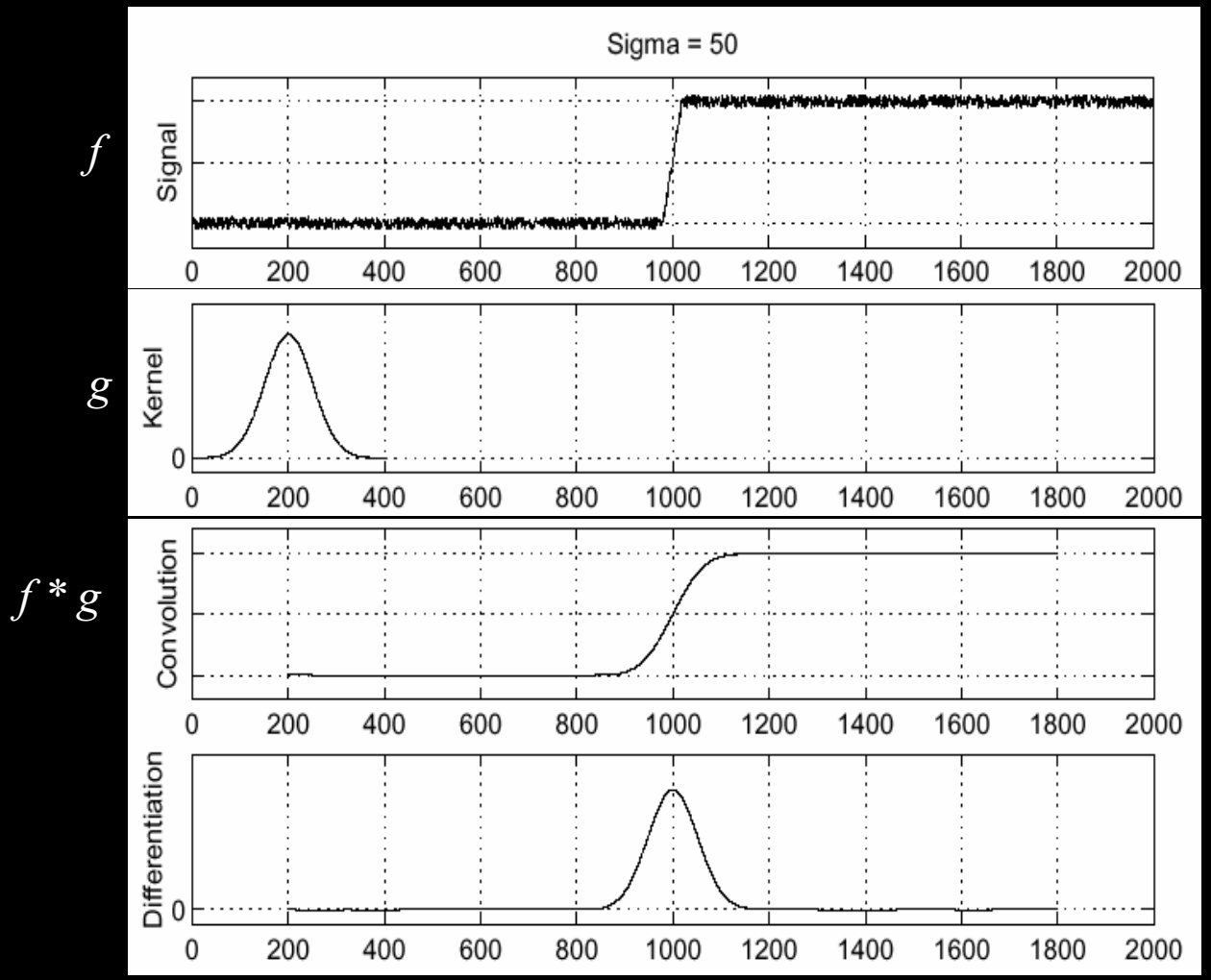




Filter w/ a Gaussian & then $\frac{d}{dx}$



Solution: Smooth First



f

g

$f * g$

- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

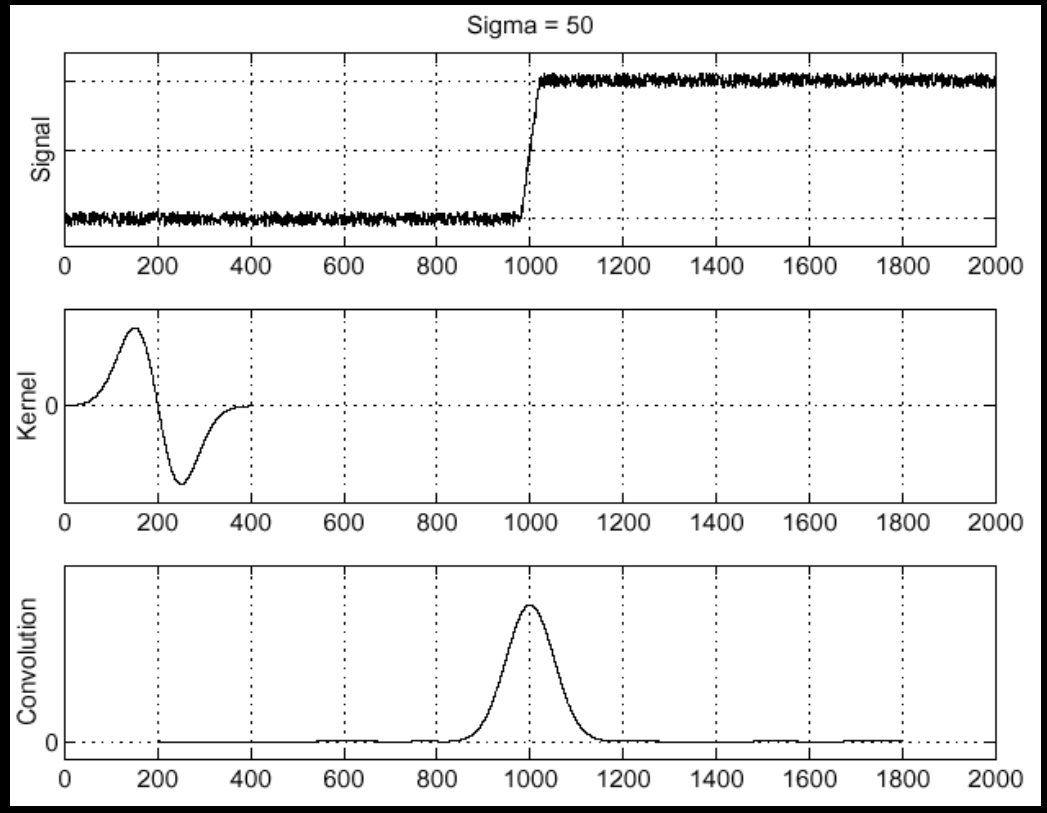
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

f

$\frac{d}{dx} g$

$f * \frac{d}{dx} g$

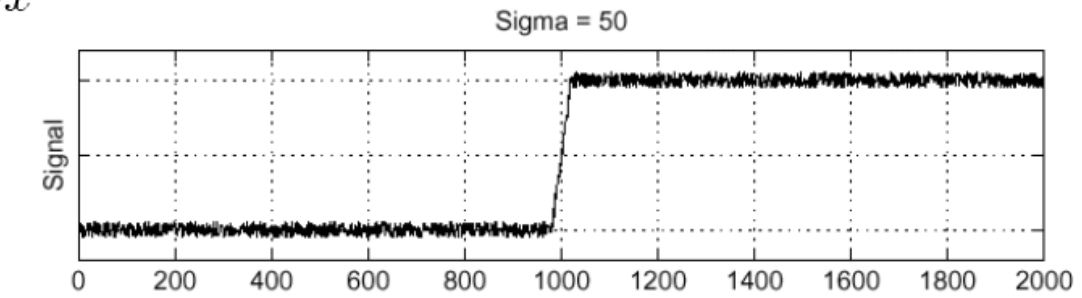


Source: S. Seitz

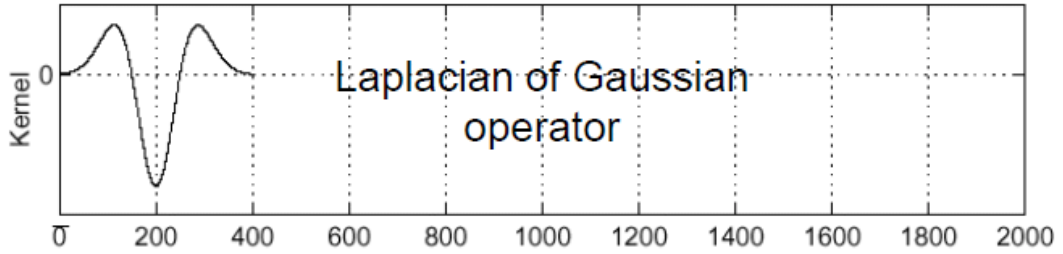
Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

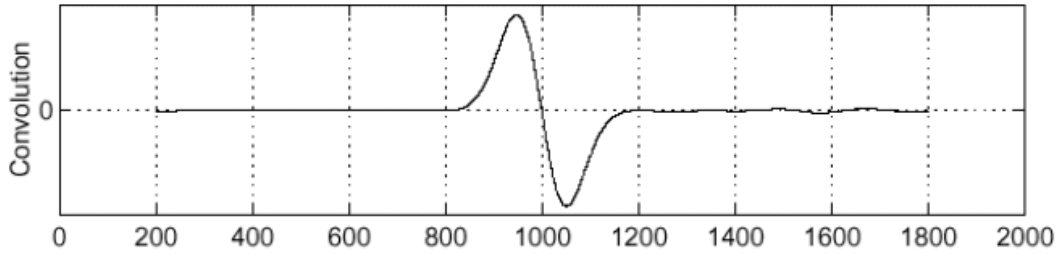
f



$\frac{\partial^2}{\partial x^2}h$



$(\frac{\partial^2}{\partial x^2}h) \star f$



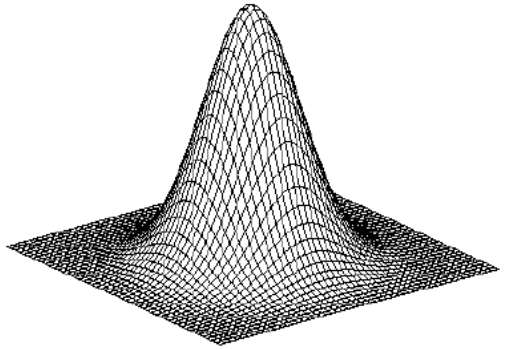
Where is the edge? Zero-crossings of bottom graph

via DerekHoiem

- Derivative of Gaussian is a Matched Filter for a Step / Smooth Edge (Max response)
- Laplacian is also a Matched Filter at appropriate scales for a bar detector (1D) and Blob detector (2D)
- This property is effectively exploited by SIFT Detector

  More on that later.

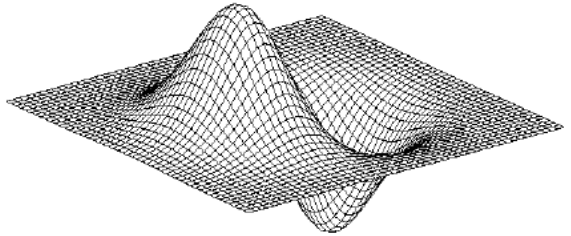
2D edge detection filters



Gaussian

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

SMOOTHING

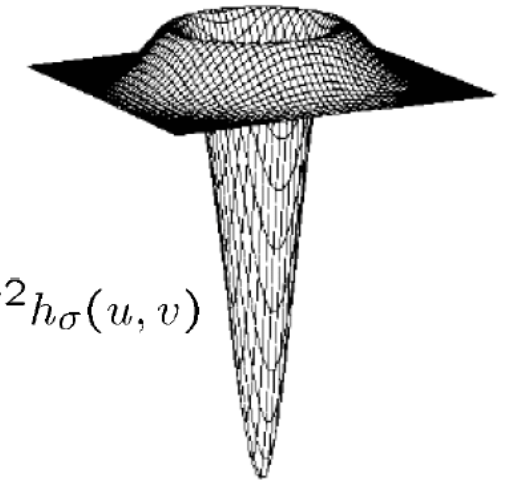


derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

EXTREMA

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

ZERO-CROSSING

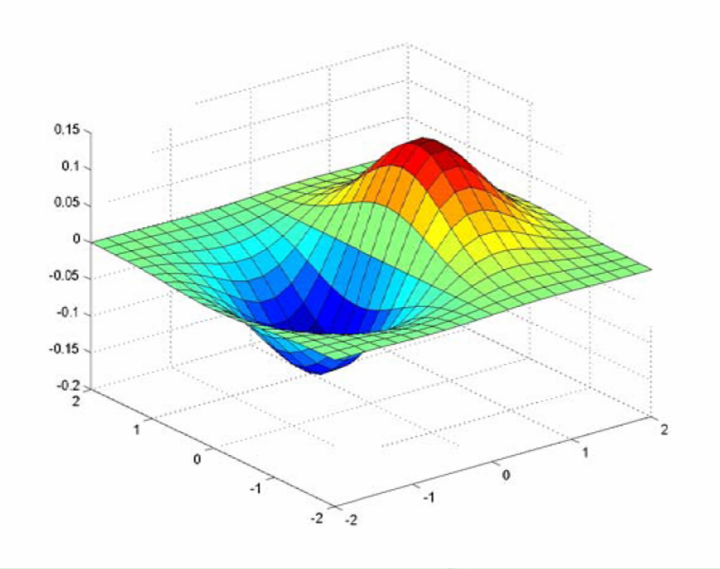
∇^2 is the **Laplacian** operator:

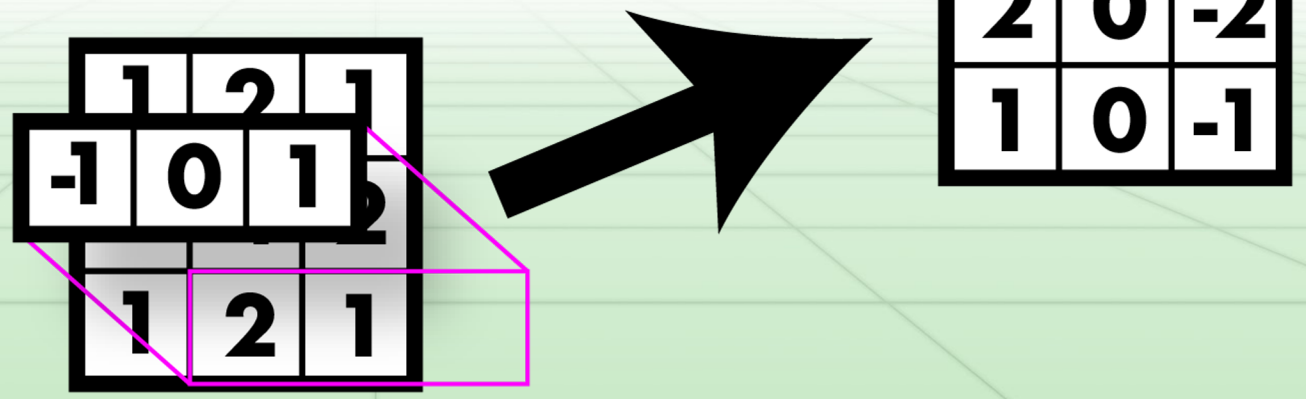
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

via DerekHoiem

Sobel filter! Smooth & derivative

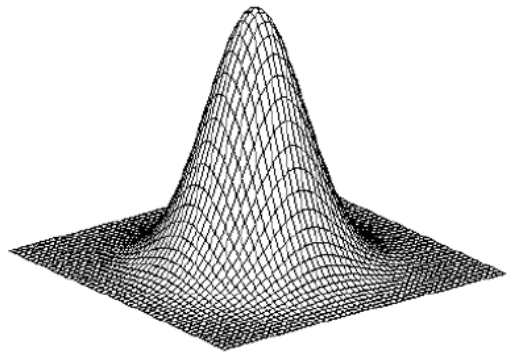
$$\frac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$





via JoeRedmon

2D Gaussian is Separable…



Gaussian

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

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

Separable kernel

- Factors into product of two 1D Gaussians
- Discrete example:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

- Given an arbitrary sigma, 95% of the AUC for Gaussian is between -2*sigma and 2*sigma
- 99.7% of the AUC is between -3*sigma to 3*sigma
- Choose one of these for your implementation
- Separable kernels (M x M) reduce the complexity of smoothing for an N x N image to $O(N^2M)$ instead of $O(N^2 M^2)$

http://onlinestatbook.com/2/calculators/normal_dist.html

Edges in 2D Domains : Images I(x, y)

The gradient of an image:

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

$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$

$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$

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

The gradient points in the direction of most rapid increase in intensity

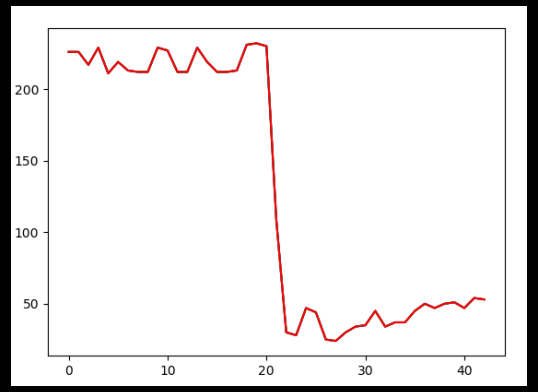
The gradient direction is given by:

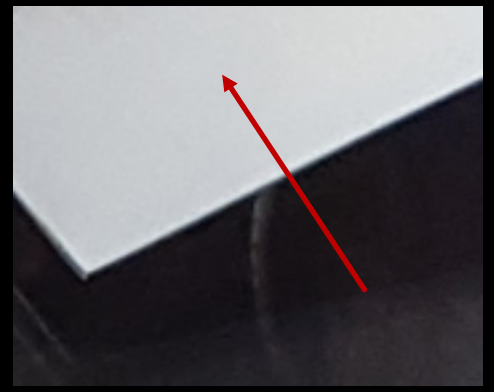
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

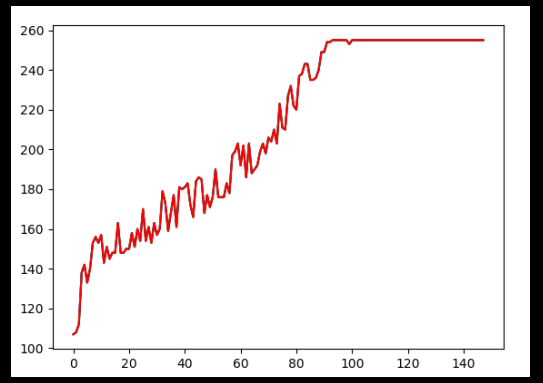
- how does this relate to the direction of the edge?

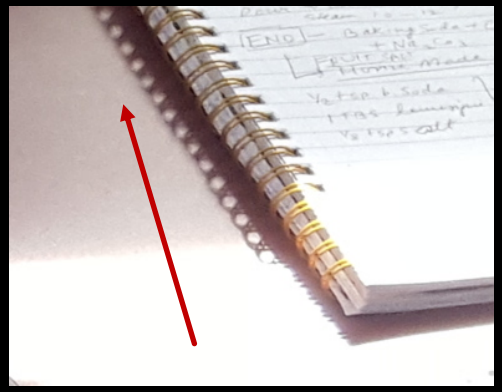
The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$



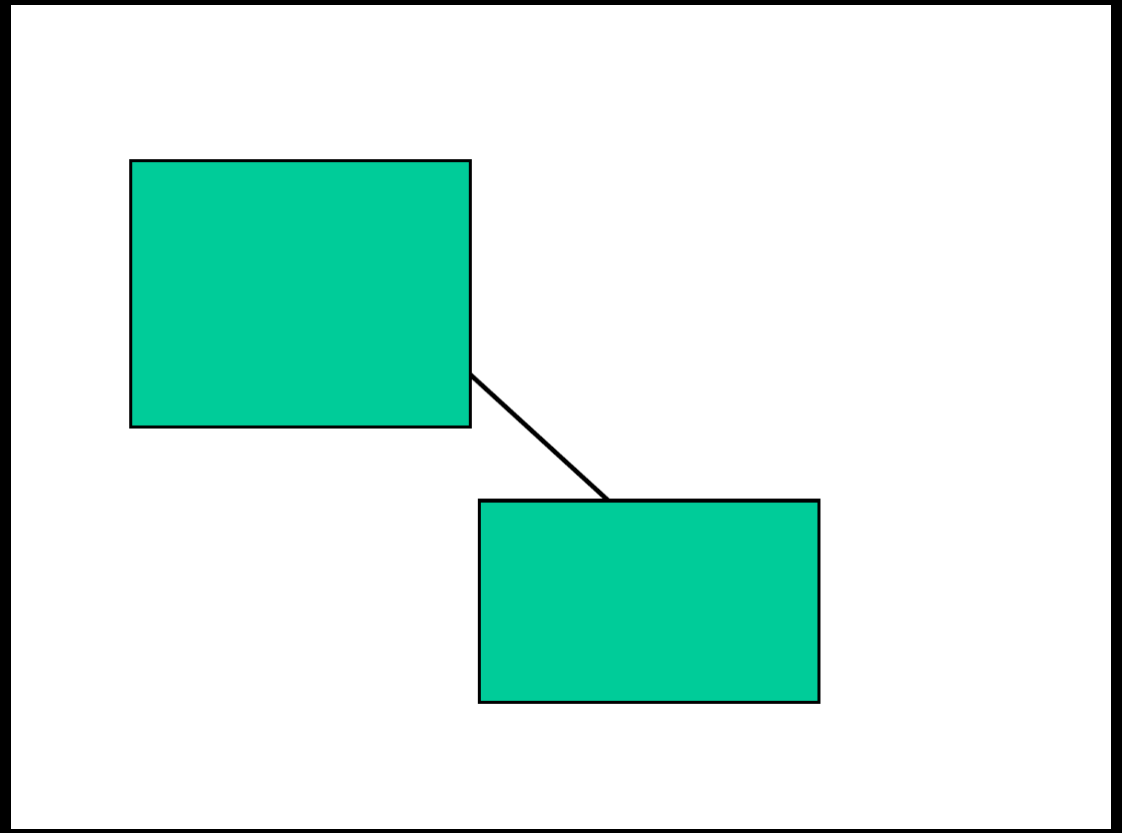


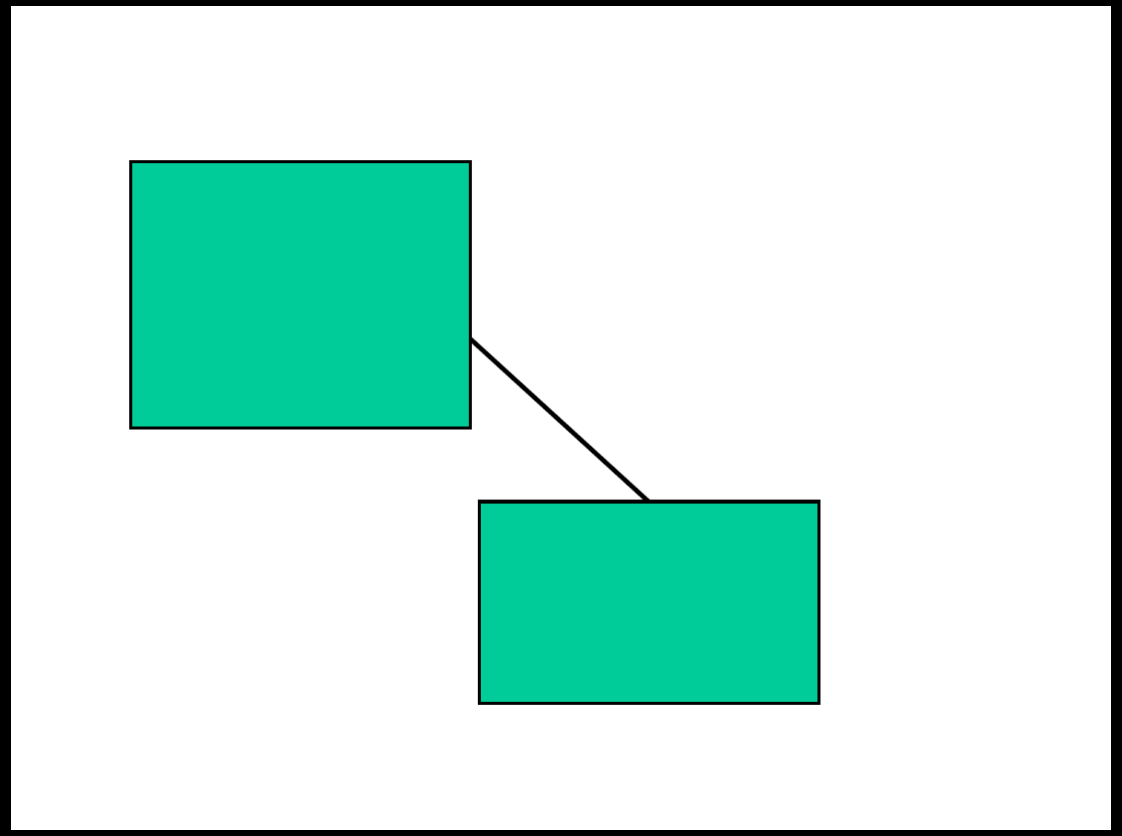


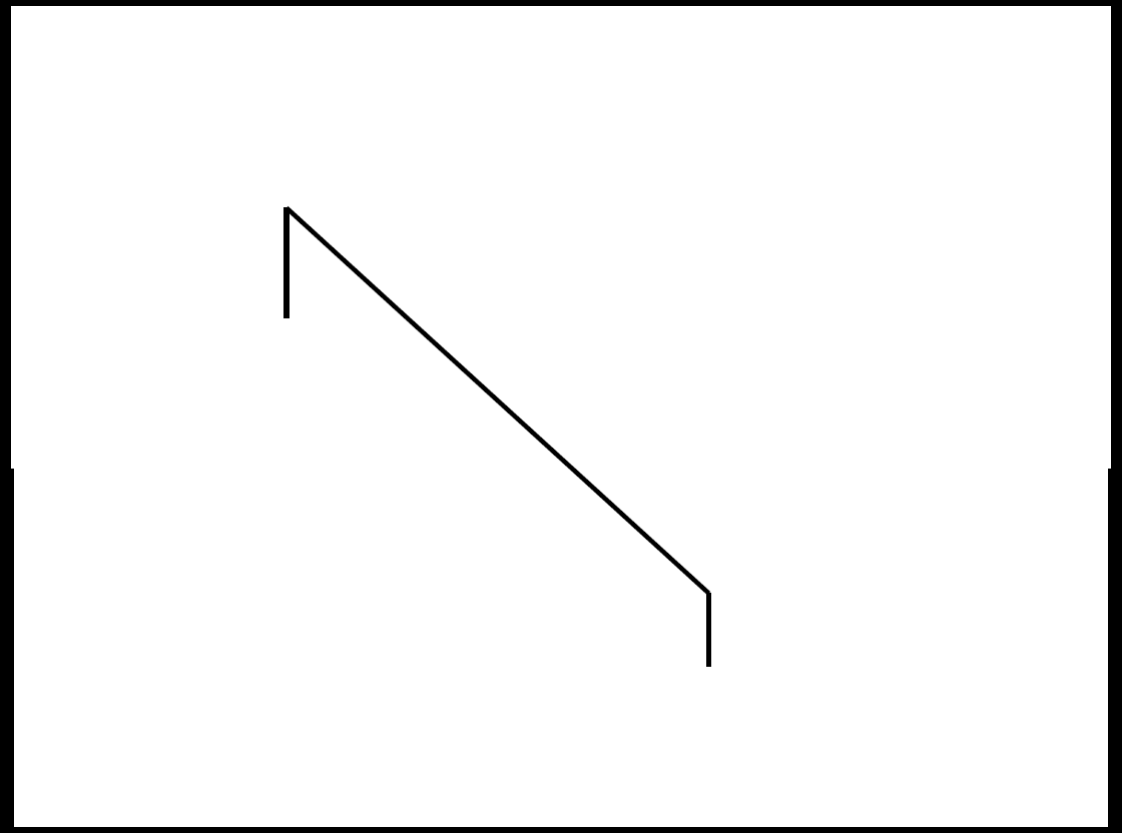


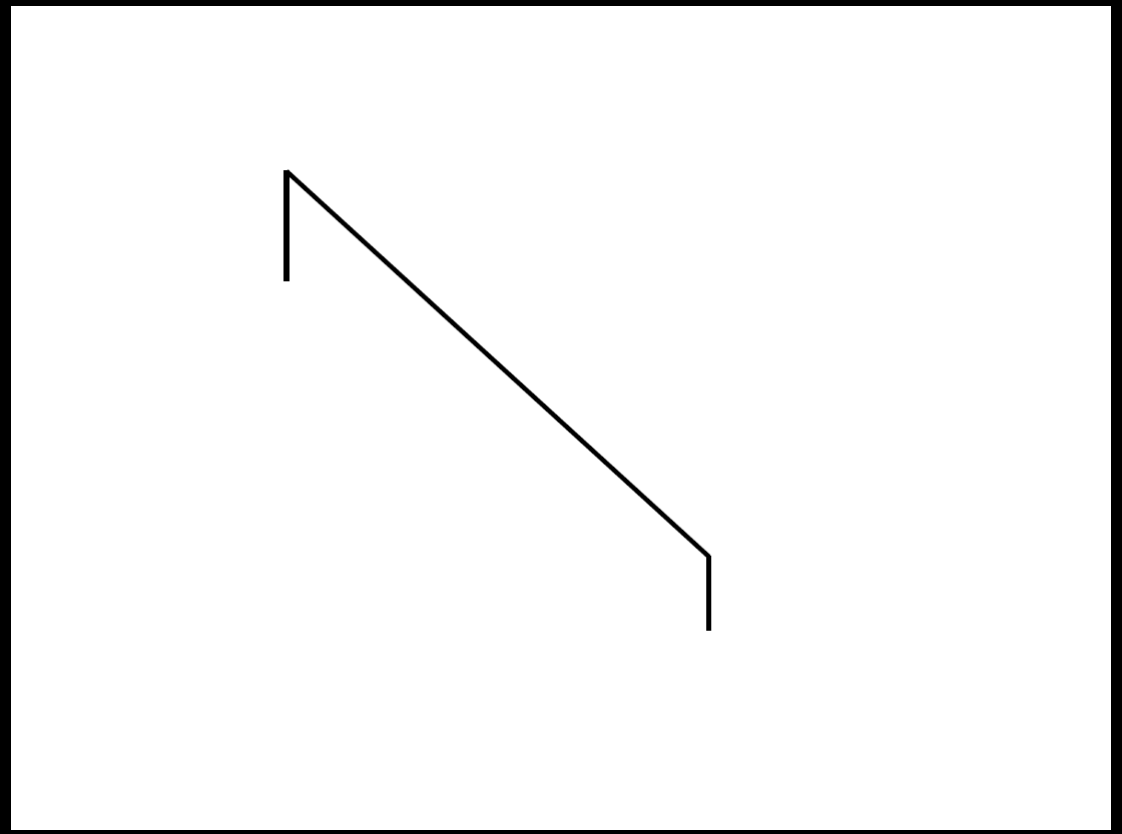
- Edges have an Orientation and a Magnitude

via SteveSeitz

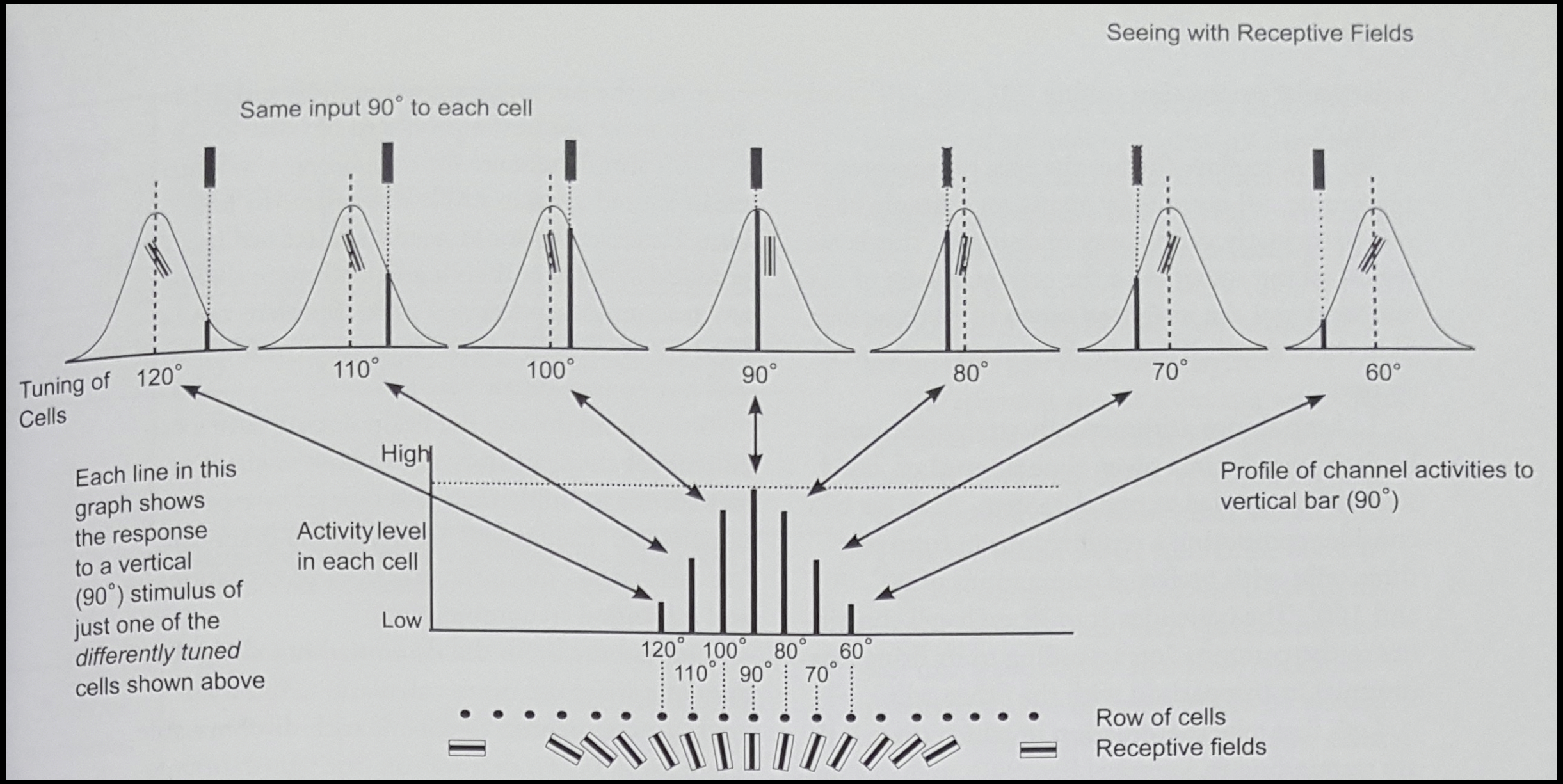






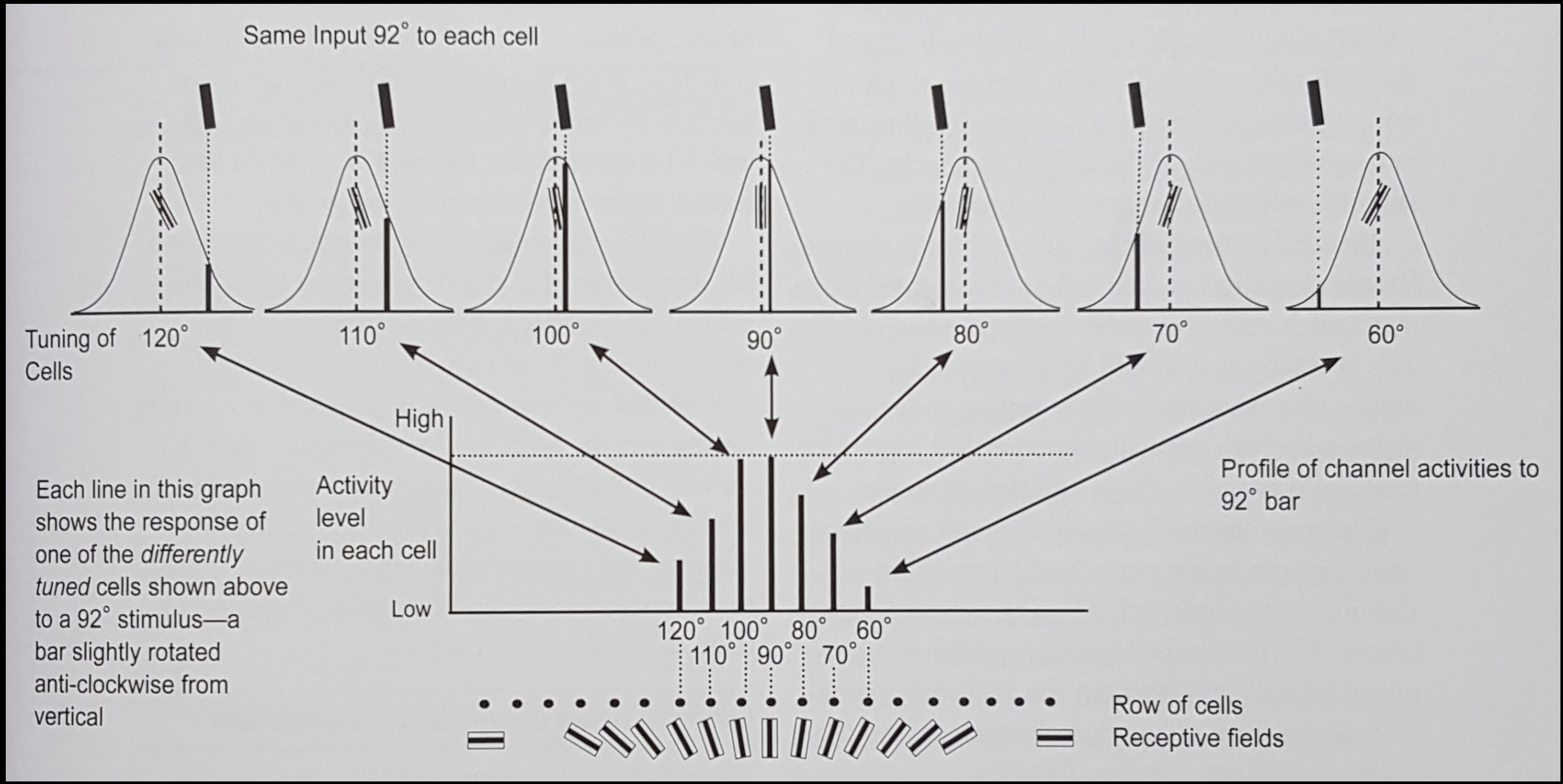


Seeing with Receptive Fields: From John Frisby



via JohnFrisby “Seeing”

Seeing with Receptive Fields: From John Frisby



via JohnFrisby “Seeing”

Edges Occur at Different Scales

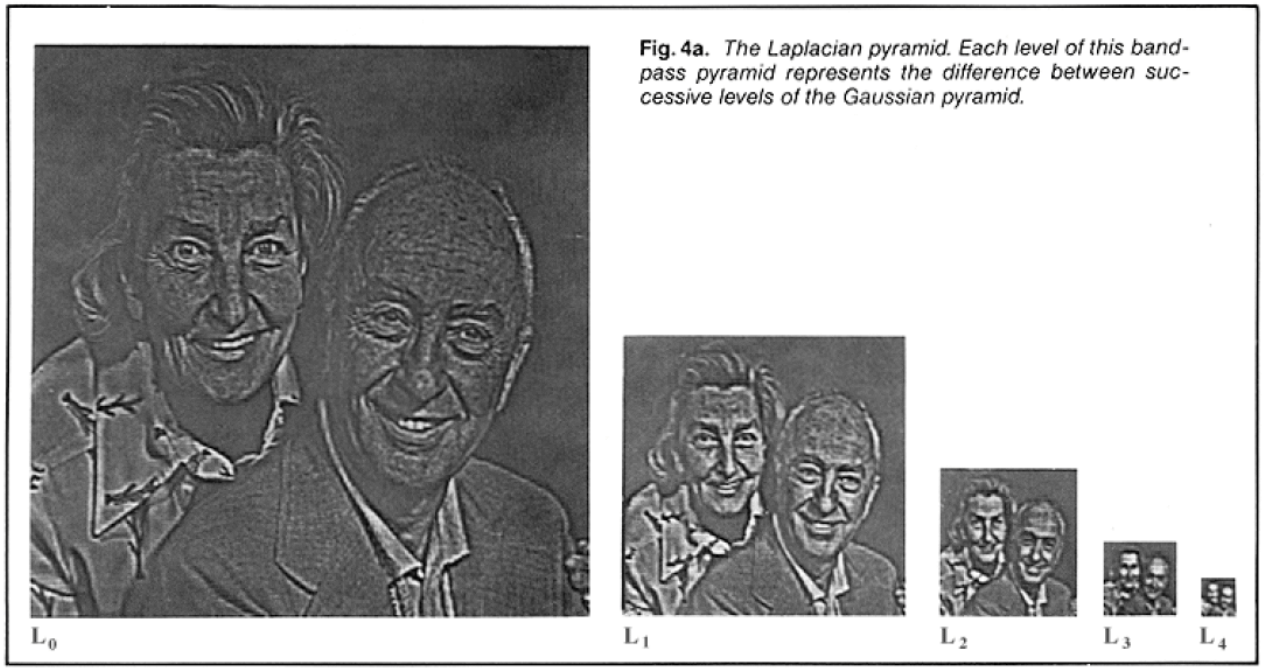


Fig. 4a. *The Laplacian pyramid. Each level of this band-pass pyramid represents the difference between successive levels of the Gaussian pyramid.*



Fig. 4b. *Levels of the Laplacian pyramid expanded to the size of the original image. Note that edge and bar features are enhanced and segregated by size.*

E. H. Adelson | C. H. Anderson | J. R. Bergen | P. J. Burt | J. M. Ogden

Pyramid methods in image processing

The image pyramid offers a flexible, convenient multiresolution format that mirrors the multiple scales of processing in the human visual system.

RCA Engineer • 29-6 • Nov/Dec 1984

Edges Occur at Different Scales

This will be discussed in more details in
Wide Baseline Features and Matching (SIFT).

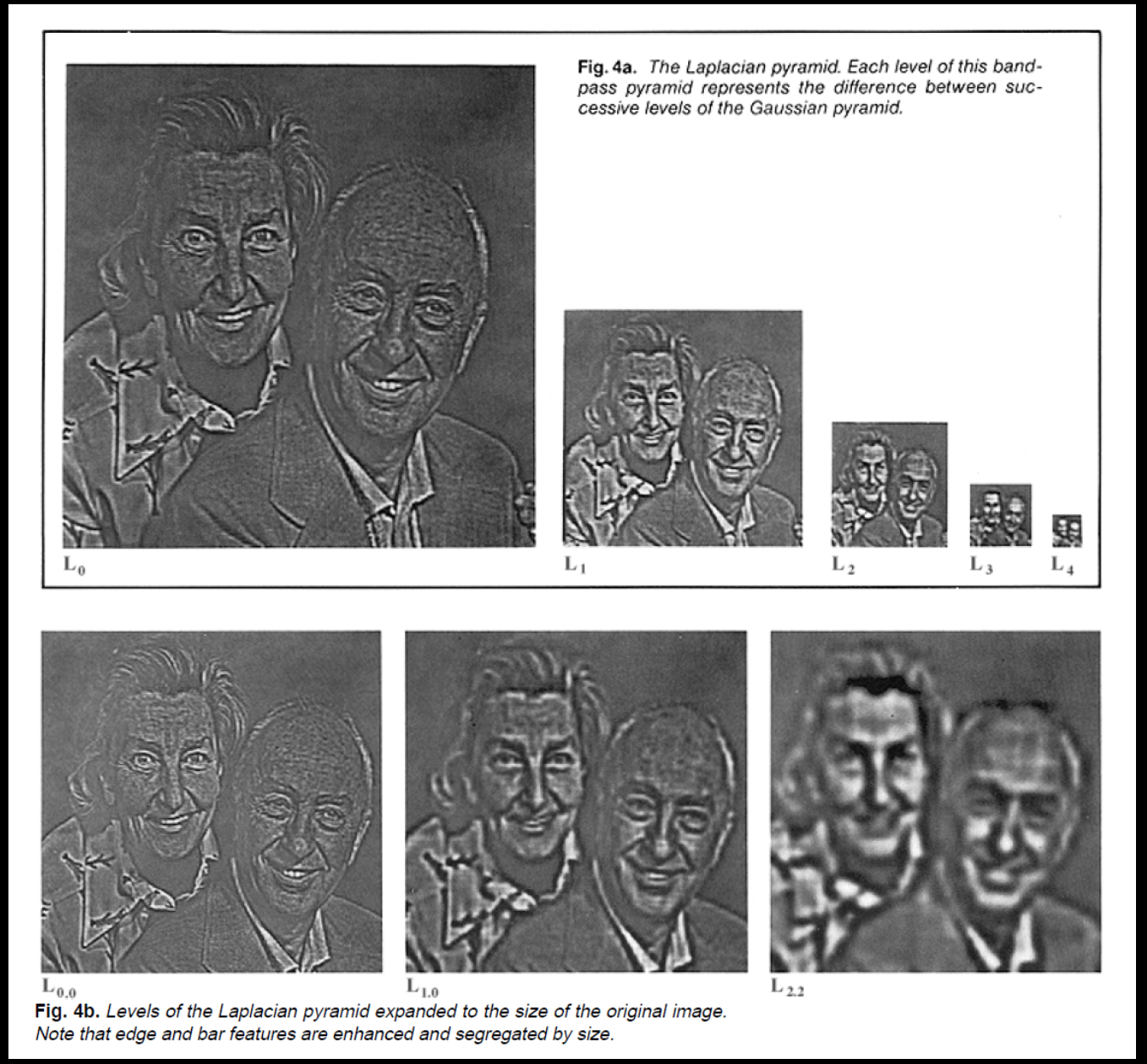


Fig. 4a. *The Laplacian pyramid. Each level of this band-pass pyramid represents the difference between successive levels of the Gaussian pyramid.*

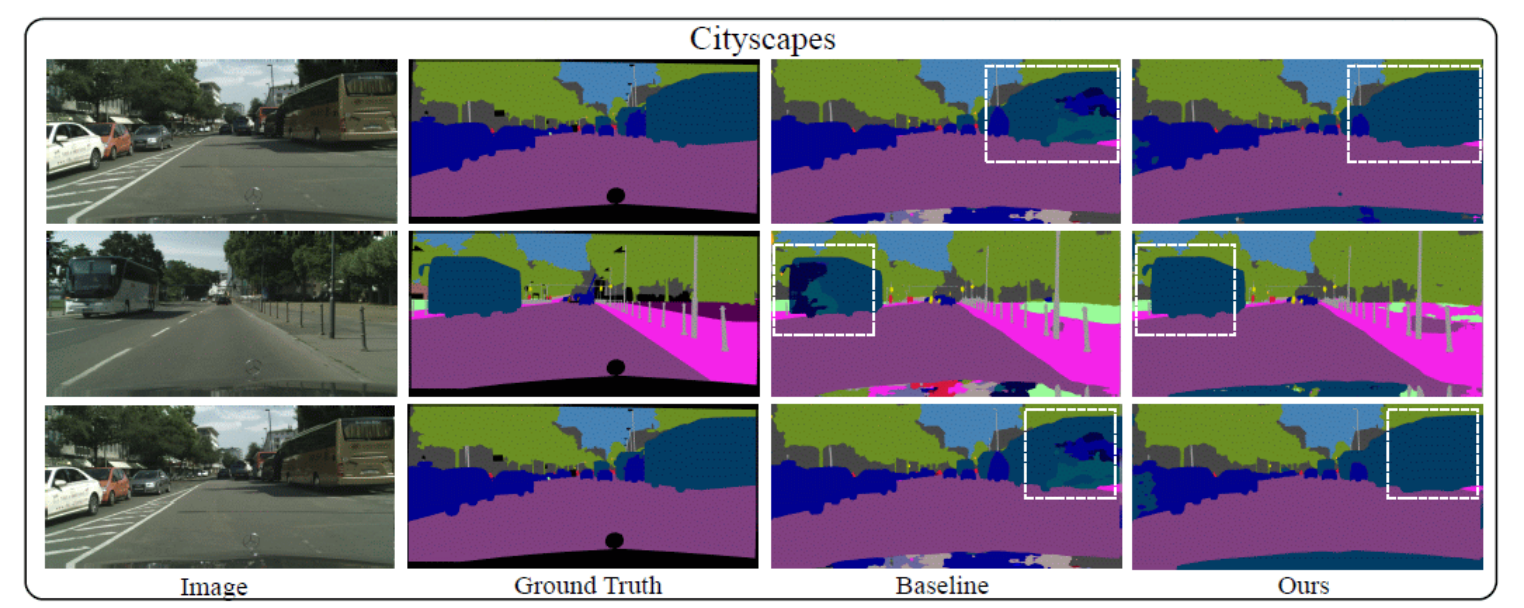
Fig. 4b. *Levels of the Laplacian pyramid expanded to the size of the original image. Note that edge and bar features are enhanced and segregated by size.*

Role of Edges ?

- From the time when Edge Detection was considered essential to understanding and manipulating images:

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

- To Modern Learning to Segment, Detect Boundaries, Objects and much more…



Object-Contextual Representations for Semantic Segmentation
Yuhui Yuan, Xilin Chen, Jingdong Wang

https://arxiv.org/abs/1909.11065v2

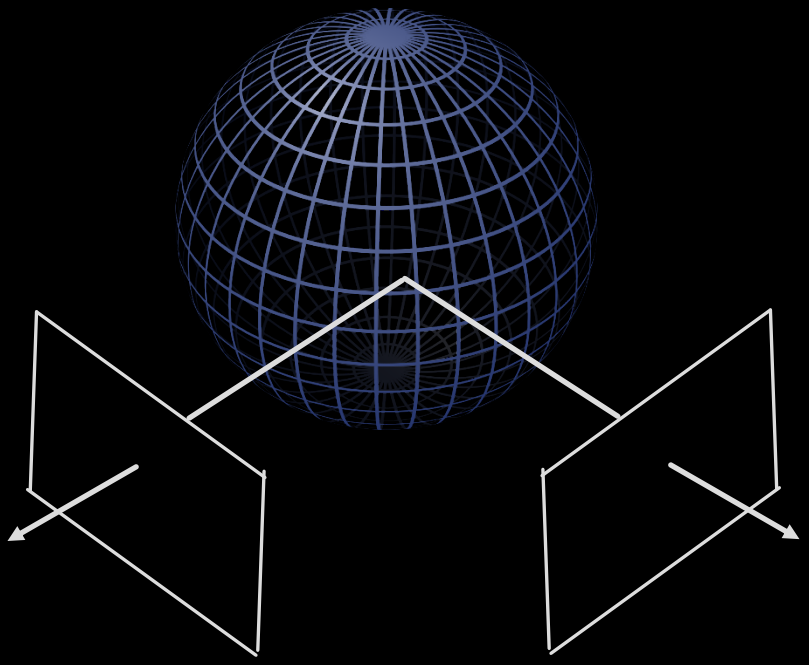
Image Matching & Alignment

Why Align Images?

- Match to create Panoramas for Single Center of Projection Images





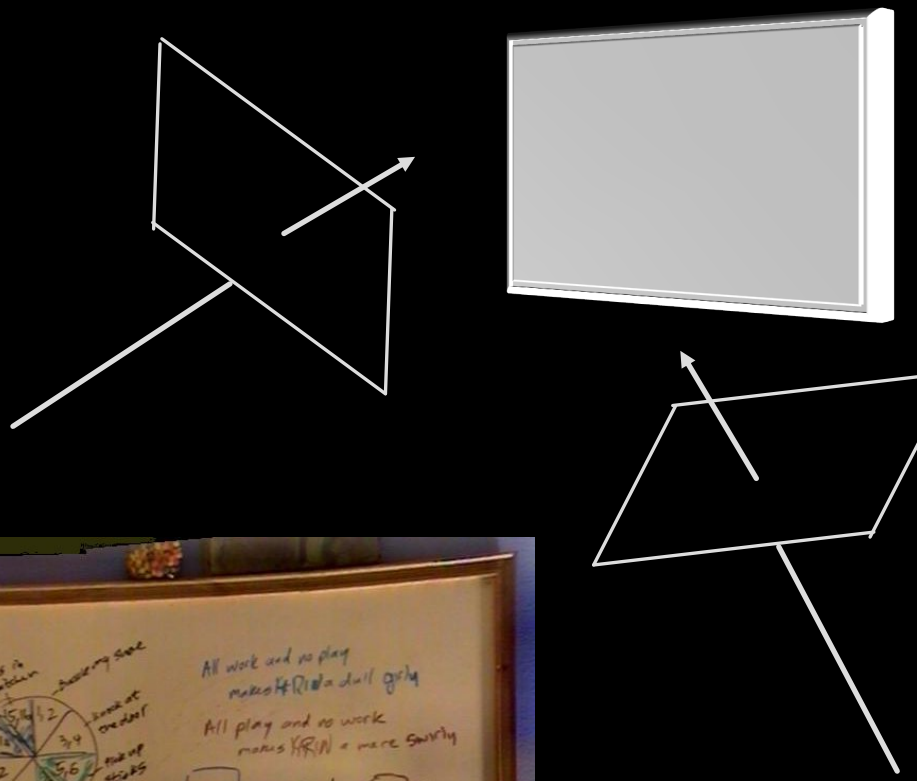


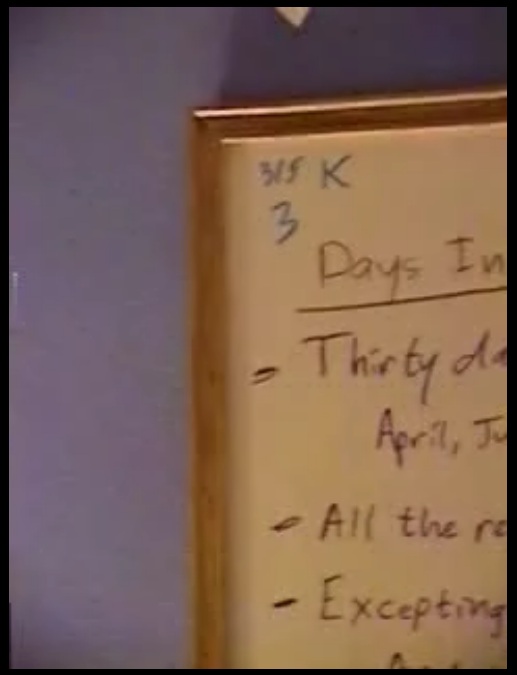
Mercator projection of spherical mosaic

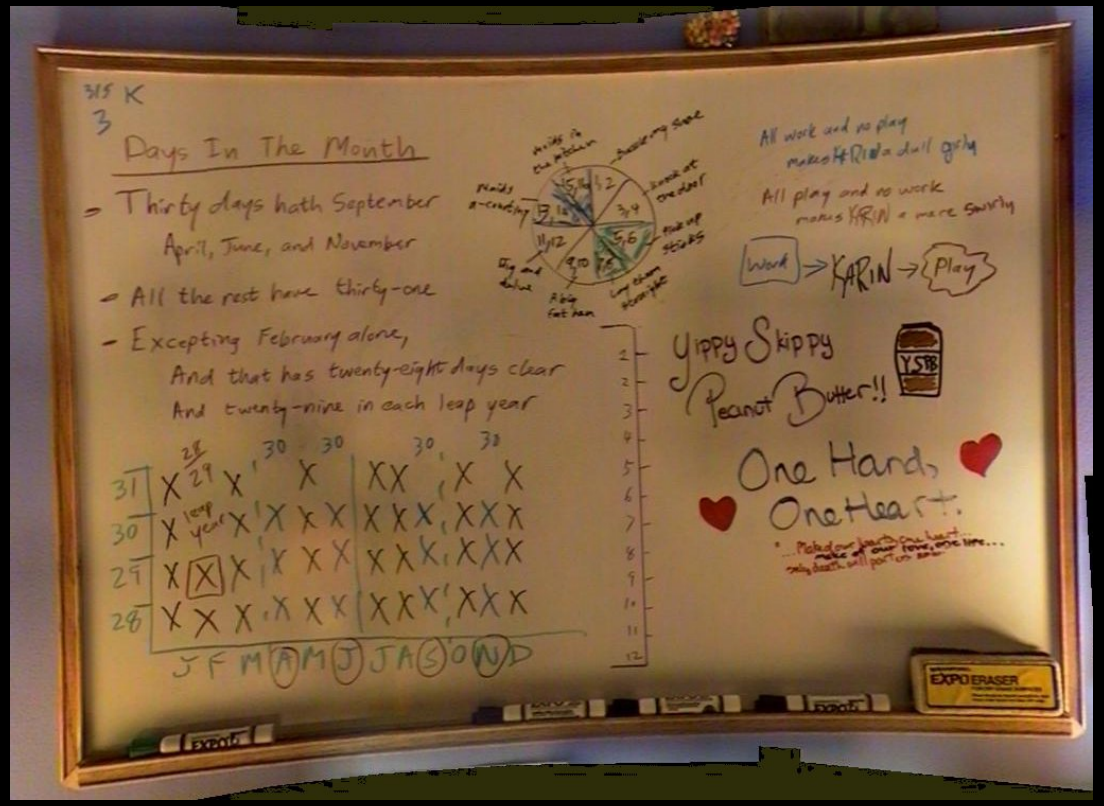


Why Align Images?

- Match to create Planar Mosaics



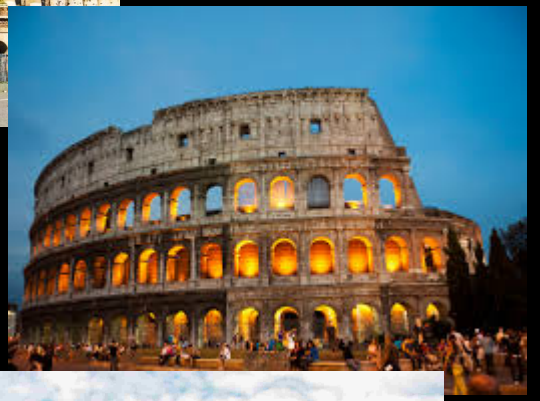


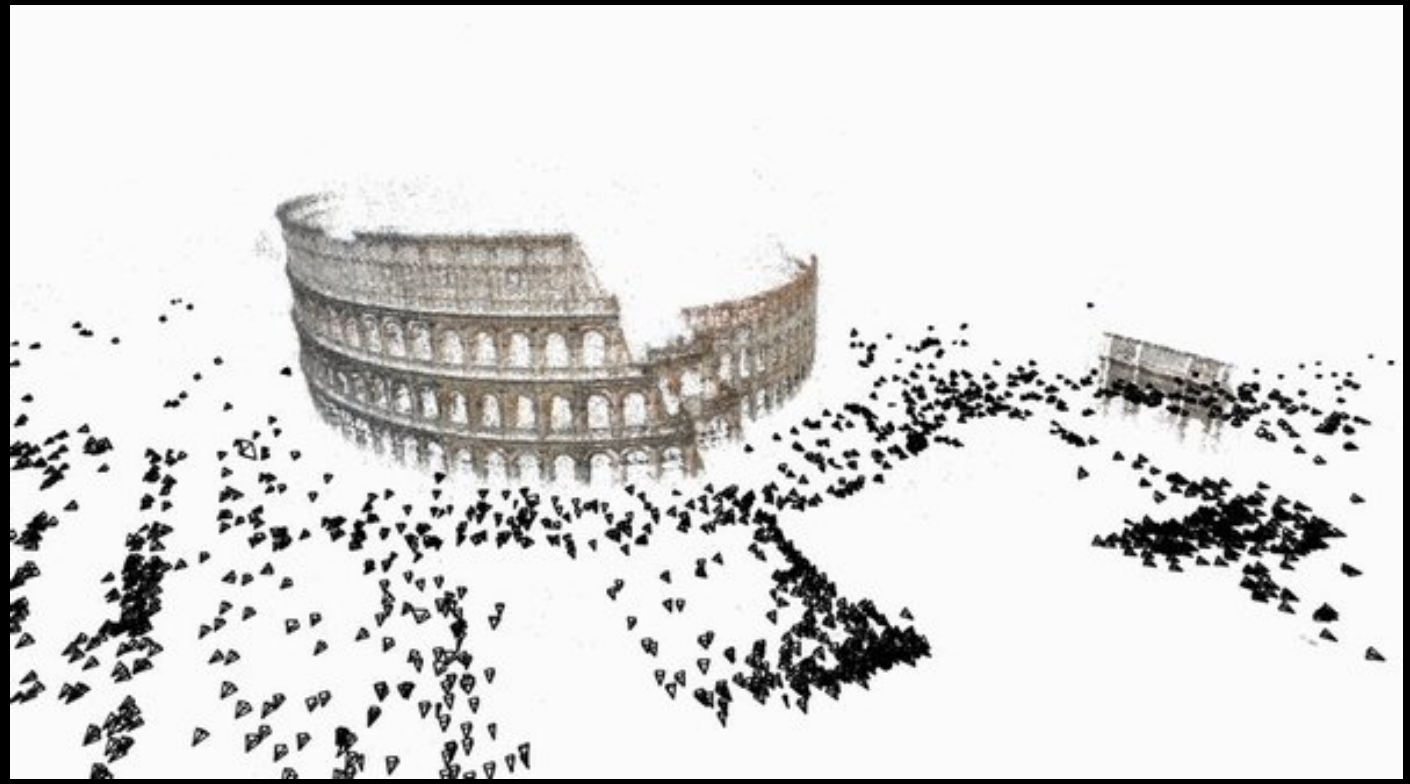


Why Align Images?

- “Build Rome in a Day”









Why Align Images?

- Image / Location / Object Retrieval from a Large Database



Figure 10. A snapshot of the CD-cover recognition running. With 40000 images in the database, the retrieval is still real-time and robust to occlusion, specularities, viewpoint, rotation and scale changes. The camera is directly connected to the laptop via firewire. The captured frames are shown on the top left, and the top of the query is displayed on the bottom right. Some of the CD-covers are also connected to music that is played upon successful recognition.



Figure 5: Examples of 3D object recognition with occlusion.



Figure 9. Typical examples of the 278 query images (left) and the corresponding top matches returned from the database (right) using a 1000^2 vocabulary tree with $N = 4$.

via DavidNister

Matching & Alignment

- Matching:

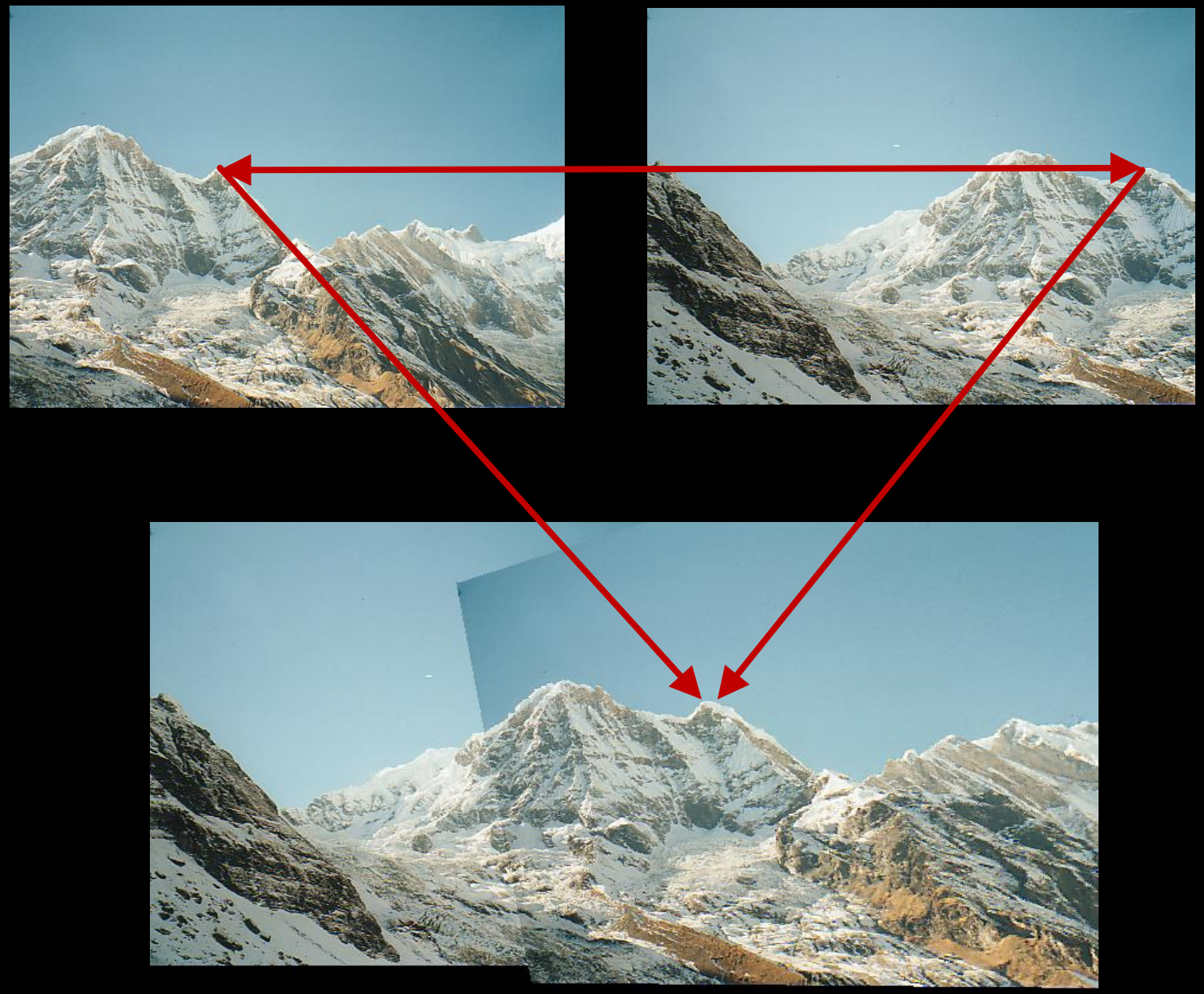
  Find locations / spatial coordinates in images that correspond to the same location / direction in the World

via Feature Detectors and Feature Descriptors

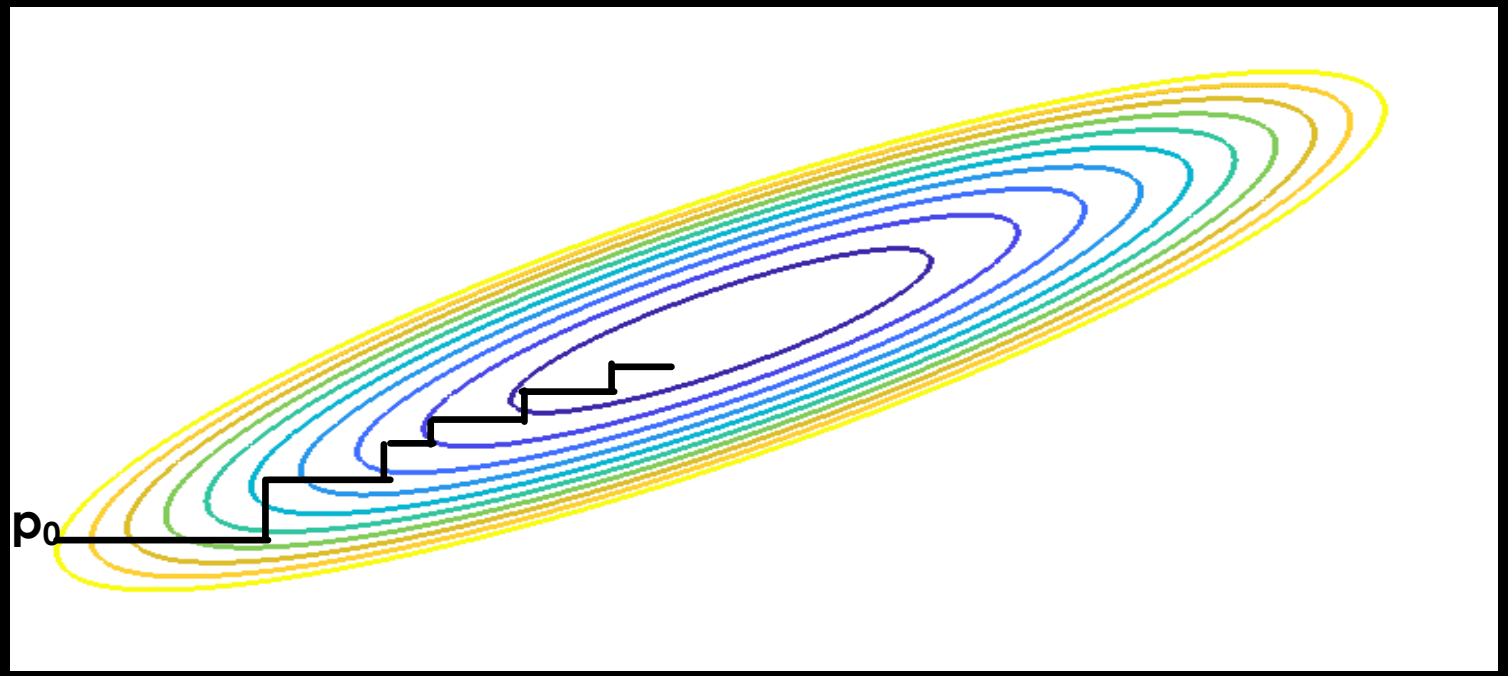
- Alignment:

  Use the matching locations to find Transformations for Camera Estimation and Warping

via Linear and Non-linear Optimization



A Swift Primer on Optimization Methods for Linear and Non-Linear (Sum of Squares) Cost Functions



If f(p) is quadratic: $f(p) = \frac{1}{2}p^T Hp - p^T \mathrm{b} + f_0$ Gradient: $g(p) = Hp - b$

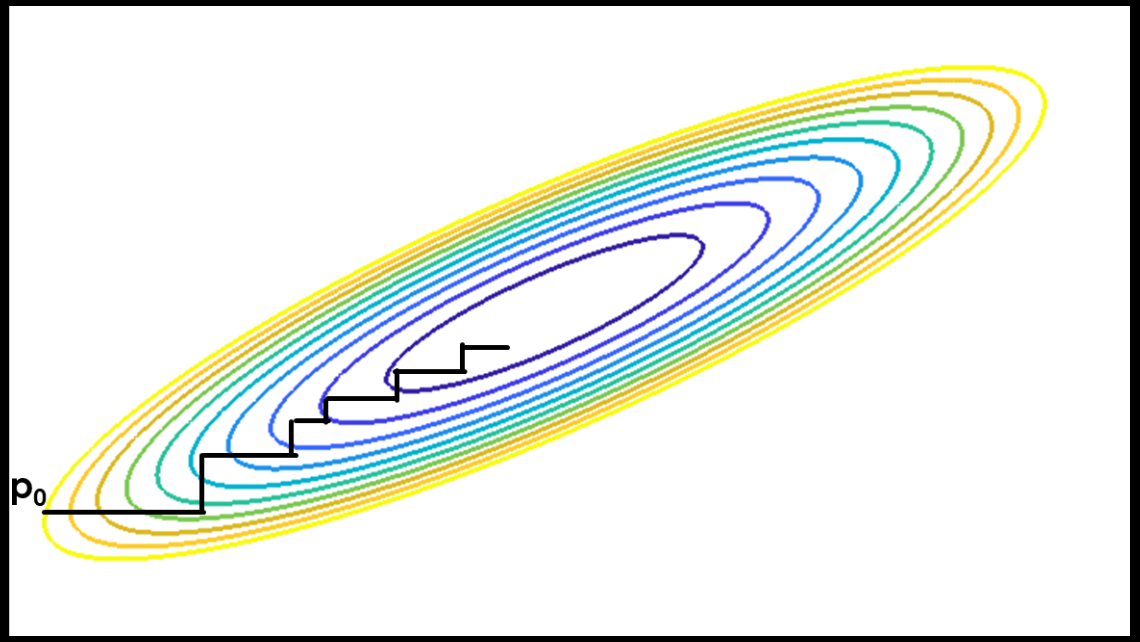
Steepest Descent: $p_{k+1} = p_k - \alpha_k g_k$ $g_k = Hp_k - b$ But how much do we move along g?

$$f(p_k - \alpha_k g_k) = \frac{1}{2}(p_k - \alpha_k g_k)^T H(p_k - \alpha_k g_k) - (p_k - \alpha_k g_k)^T b$$

$$\frac{\partial}{\partial \alpha_k} = 0 \Rightarrow \alpha_k = \frac{{g_k}^T g_k}{g_k H g_k}$$

Rate of convergence governed by: $\left(\frac{r-1}{r+1}\right)^2$ $r = \frac{\lambda_{max}}{\lambda_{min}}$

Observations



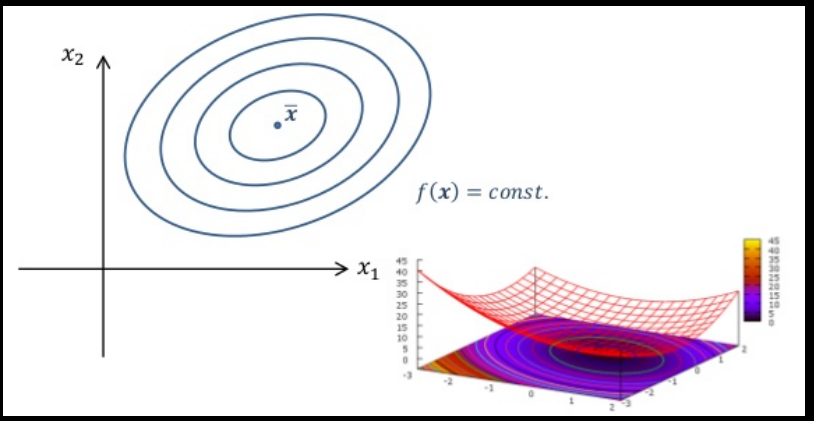
- Gradient Descent “looks” at the cost function VERY LOCALLY.

- r = 1 => Single step convergence
- r >> 1 => Very slow convergence

CAN WE DO BETTER?

- Line search does not let the descent overshoot.
  - Too expensive in Backprop for CNNs because of millions of parameters
  - So learning rate used as a hyperparameter that is “globally” set and varied a bit

Quadratic Cost Functions



- Model the 2nd order variation (CURVATURE) in addition to the Gradient
- Convex second order cost functions have a unique global minimum
- f(p) is convex quadratic: $f(p) = \frac{1}{2}p^T Hp - p^T \mathrm{b} + f_0$ H is Positive Definite (Concave Upwards)

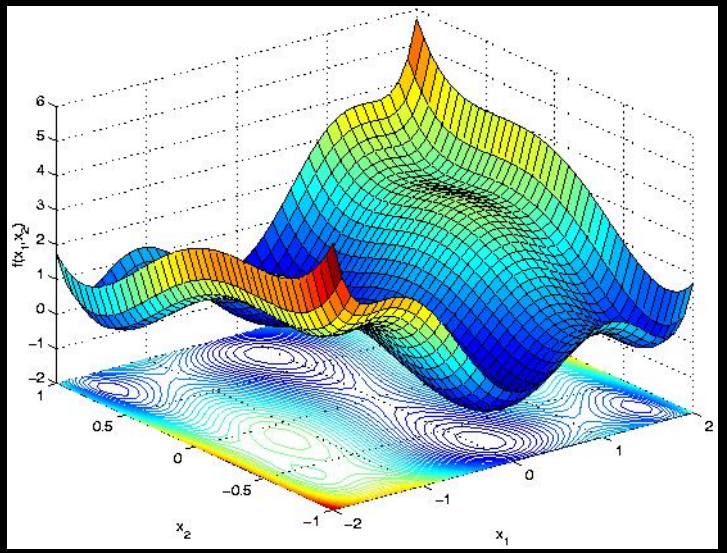
$$\nabla f(p) = Hp - b \qquad Setting\ \nabla f(p) = 0 \quad \Rightarrow \quad p_{opt} = H^{-1}b$$

- p_{opt} is a Global minimum : f(p) >= f(p_{opt})
- By modeling the globally convex function we obtain minimum solution in a single step!

Optimization of General Cost Functions



- We can model local quadratic behavior and apply descent type methods to general cost functions
- Only local minima can be found
- For local descent to happen the function needs to be PD (positive definite)
- Let us see why



Optimization of General Cost Functions



- At any point p_k use quadratic approximation to define a descent direction: $\delta p = -H(p_k)^{-1}\ \nabla f(p_k)$
- For generality define a descent direction as: $p_{k+1} = p_k\ - \alpha_k M_k g_k$
- $f(p_{k+1}) = f(p_k)\ - \alpha_k {g_k}^T M_k g_k$ => For this to do descent ${g_k}^T M_k g_k > 0$ (Positive Definite)
- $M_k = \mathrm{I}$ guarantees PD but is gradient descent that converges linearly.
- $M_k = H(p_k)^{-1}$ has quadratic convergence but not guaranteed to be PD.

Optimization of General Cost Functions



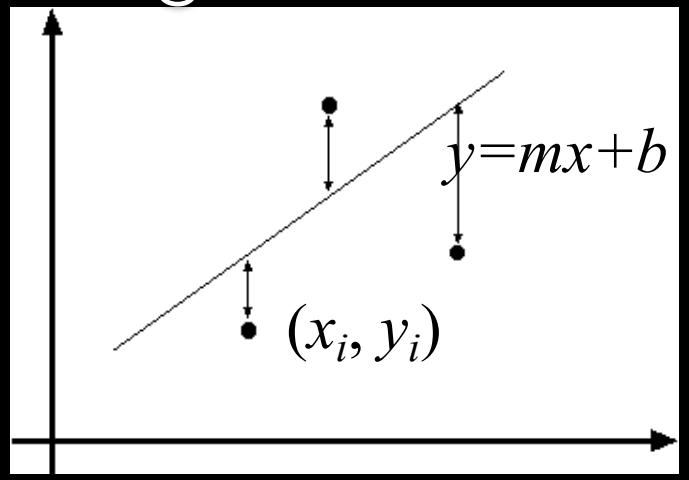
- $M_k = \mathrm{I}$ guarantees PD but is gradient descent that converges linearly.
- $M_k = H(p_k)^{-1}$ has quadratic convergence but not guaranteed to be PD.
- By melding both aspects: $M_k = [\epsilon_k I + H(p_k)]^{-1}$
- ϵ_k >> 1 is Steepest Descent. ϵ_k = 0 is Second Order method such as Newton's method.
- Special case for Sum of Squares problems (like in Vision) called Levenberg-Marquardt method.
- If descent possible use current M_k otherwise scale up ϵ_k.
- Close to the solution decrease ϵ_k towards zero to approximate second order descent.

Special Common Case : Sum of Squares Cost Functions

- $Y_i = \mathbf{A}_i X + \eta \qquad \eta \sim N(0, \Lambda_i) \qquad Y_i$ is a vector of measurements corrupted by noise. X is the unknown.
- Assuming independence of measurements Y_i, the negative log likelihoods of the measurements given X is:
- $\Sigma_i \, (Y_i - \mathbf{A}_i X)^T {\Lambda_i}^{-1} (Y_i - \mathbf{A}_i X) \quad = \quad \Sigma_i \, ({Y_i}^T {\Lambda_i}^{-1} Y_i - 2 {Y_i}^T \mathbf{A}_i X + X^T \mathbf{A}_i^T {\Lambda_i}^{-1} \mathbf{A}_i X)$
- To find the optimal X we take the derivative w.r.t. X and set it to zero:
- $\frac{\partial}{\partial X} = \; 2 \left[\Sigma_i \mathbf{A}_i^T {\Lambda_i}^{-1} \mathbf{A}_i\right] X \; - 2 \left[\Sigma_i \mathbf{A}_i^T Y_i\right] \; = 0$
- $X = \left[\Sigma_i \mathbf{A}_i^T {\Lambda_i}^{-1} \mathbf{A}_i\right]^{-1} \left[\Sigma_i \mathbf{A}_i^T Y_i\right]$
- Solvable in Closed form. For large systems iterative methods are used.

Special Case : Least squares line fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m\,x_i + b$
- Find (m, b) to minimize



$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

$$E = \sum_{i=1}^{n} \left(\begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$
$$= \mathbf{y}^T\mathbf{y} - 2(\mathbf{Ap})^T\mathbf{y} + (\mathbf{Ap})^T(\mathbf{Ap})$$

$$\frac{dE}{dp} = 2\mathbf{A}^T\mathbf{Ap} - 2\mathbf{A}^T\mathbf{y} = 0$$

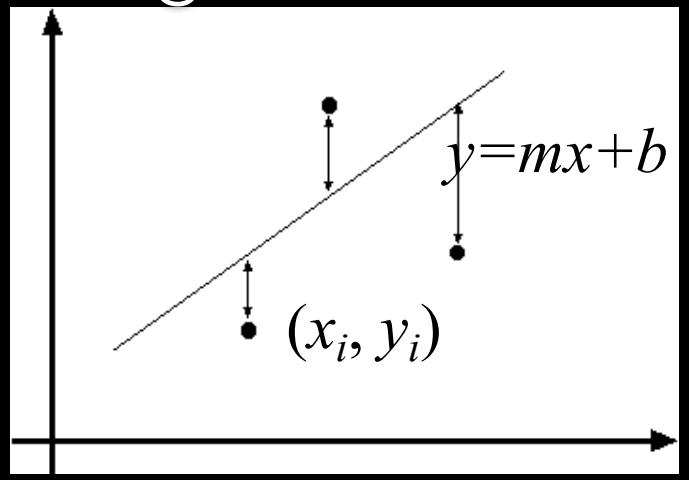
$$\mathbf{A}^T\mathbf{Ap} = \mathbf{A}^T\mathbf{y} \Rightarrow \mathbf{p} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$$

Modified from S. Lazebnik

Special Case : Least squares line fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m\,x_i + b$
- Find (m, b) to minimize

$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$



$$E = \sum_{i=1}^{n} \left(\begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$
$$= \mathbf{y}^T\mathbf{y} - 2(\mathbf{Ap})^T\mathbf{y} + (\mathbf{Ap})^T(\mathbf{Ap})$$

$$\frac{dE}{dp} = 2\mathbf{A}^T\mathbf{Ap} - 2\mathbf{A}^T\mathbf{y} = 0$$

$$\mathbf{A}^T\mathbf{Ap} = \mathbf{A}^T\mathbf{y} \Rightarrow \mathbf{p} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$$

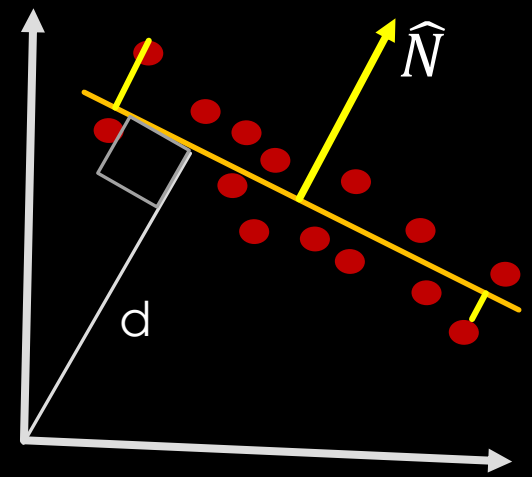
Caveat:

Errors are in y's and not in x's.

Say x is time for example.

Modified from S. Lazebnik

Line Fitting with Realistic Error Model



- Typically errors are in all the measured quantities: x and y here
- A proper error measure to optimize is normal distance of measured points to a best fit line
- Line: $n^T p = d$
- $\min_{n,d} \sum_i \|n^T p_i - d\|^2 \quad = \quad n^T \overbrace{\left[\sum_i p_i \, p_i^T\right]}^{\mathbf{S}} n - 2\, d\, p_i^T\, n + \, d^2$
- $\frac{\partial}{\partial d} = 0 \qquad \Rightarrow \qquad d = \, p_0^T \, n$ where p_0 is the centroid of all the points
- $\frac{\partial}{\partial n} = 0 \qquad \Rightarrow \qquad \overline{\boldsymbol{S}} n = \, \lambda n \quad n^T n = 1$ Non-trivial solution is the Null space vector of $\overline{\boldsymbol{S}}$

  Eigenvector with the smaller (~ 0) eigenvalue

Why is Fitting and Features Important for Image Matching?

- Find 2D / 3D Transformations
- Create Panoramas

Fit to what : Coordinates defined by Features

Features have two Components:

Feature Detector

Feature Descriptor

Two Types of Matching Problems

SHORT BASELINE

- Limited change in viewpoint & illumination
- Typically Videos / Image Sequences
- Very similar looking images



Video Example

WIDE BASELINE

- Larger changes in viewpoint & illumination
- Jumble of frames
- “Distant” in time:
  - Retrieve Locations, Objects, “THINGS”, Images, …





Two Types of Matching Problems

Today's Topic

SHORT BASELINE

- Limited change in viewpoint & illumination
- Typically Videos / Image Sequences
- Very similar looking images



Video Example

WIDE BASELINE

- Larger changes in viewpoint & illumination
- Jumble of frames
- "Distant" in time:
  - Retrieve Locations, Objects, "THINGS", Images, …



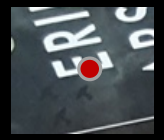


Features

- Detector / Keypoints: Discrete “distinctive” locations in the image.



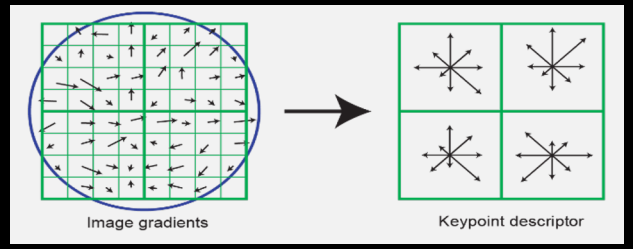
- Descriptors: Raw / Filtered Representations of Patches around the Keypoints



Intensity Patch



Filtered Patch



Quasi-invariant Feature
SIFT

Learned
Detectors
&
Descriptors

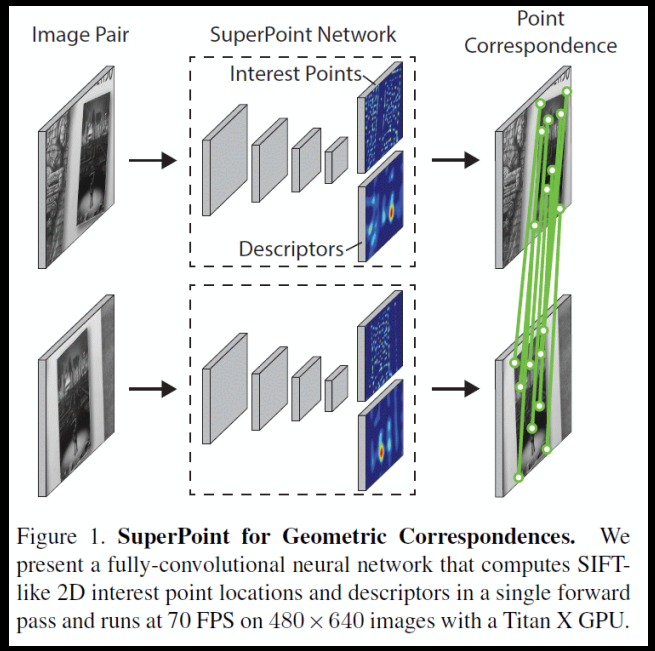
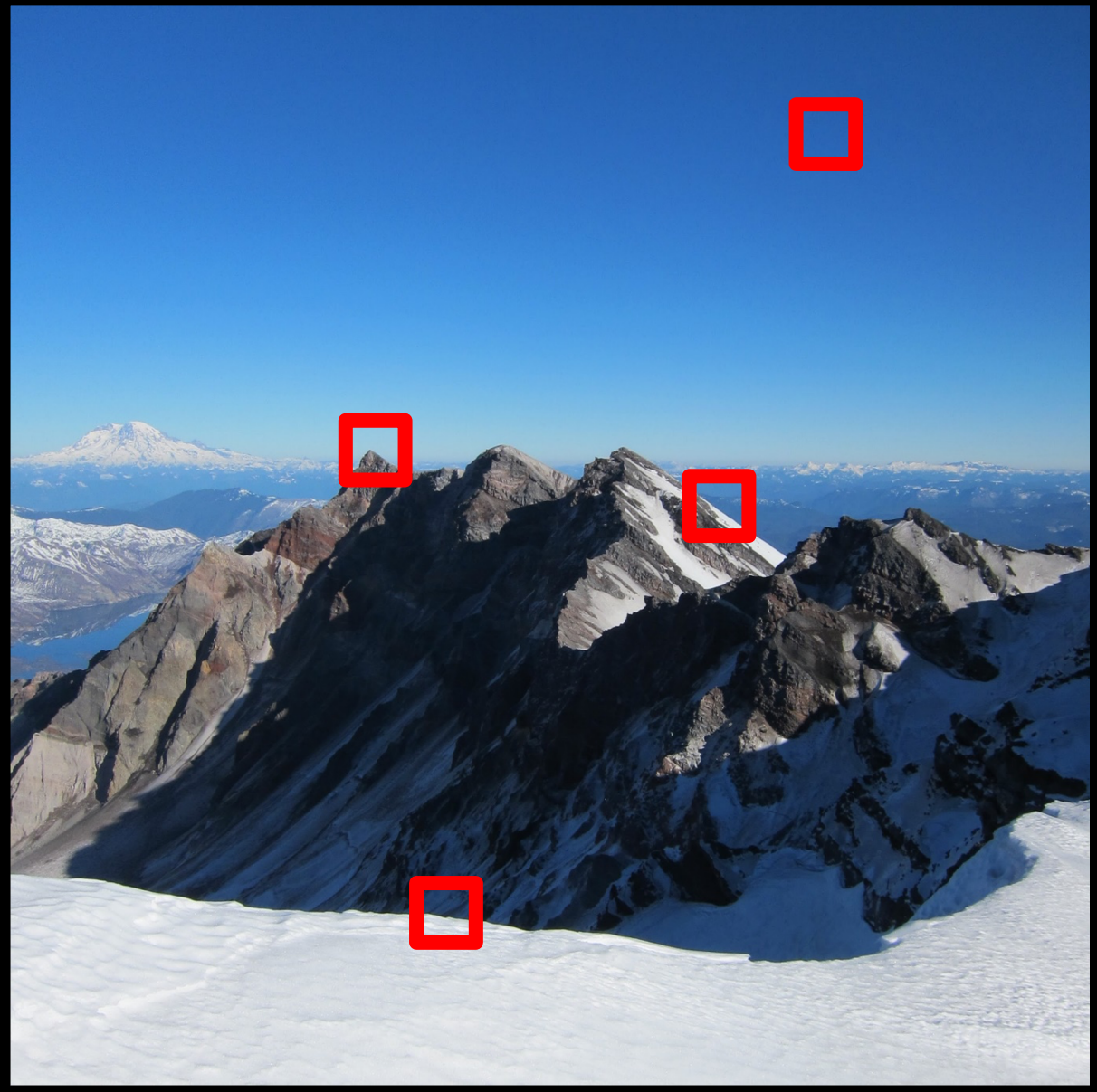


Figure 1. **SuperPoint for Geometric Correspondences.** We present a fully-convolutional neural network that computes SIFT-like 2D interest point locations and descriptors in a single forward pass and runs at 70 FPS on 480×640 images with a Titan X GPU.

via JoeRedmon

Where would you define good features?



via JoeRedmon

How can we find unique patches?

Sky: bad

- Very little variation
- Could match any other sky



via JoeRedmon

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



via JoeRedmon

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



via JoeRedmon

Making Panoramas

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





via JoeRedmon

How close are two patches?

Sum squared difference

Images I, J

$\Sigma_{x,y} (I(x,y) - J(x,y))^2$

via JoeRedmon

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:

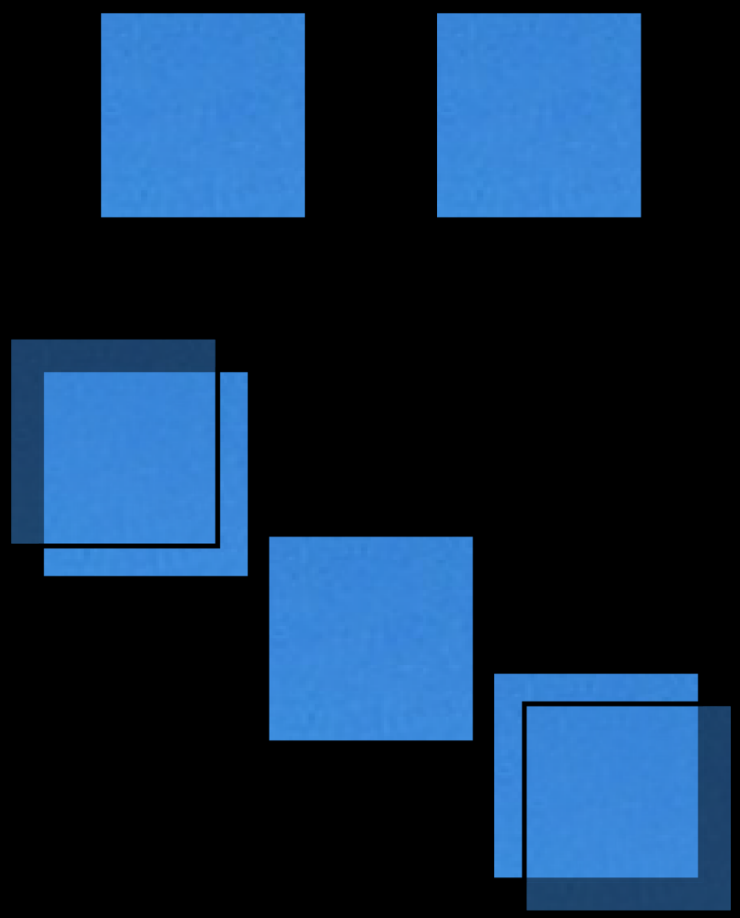
How well does image match shifted version of itself?

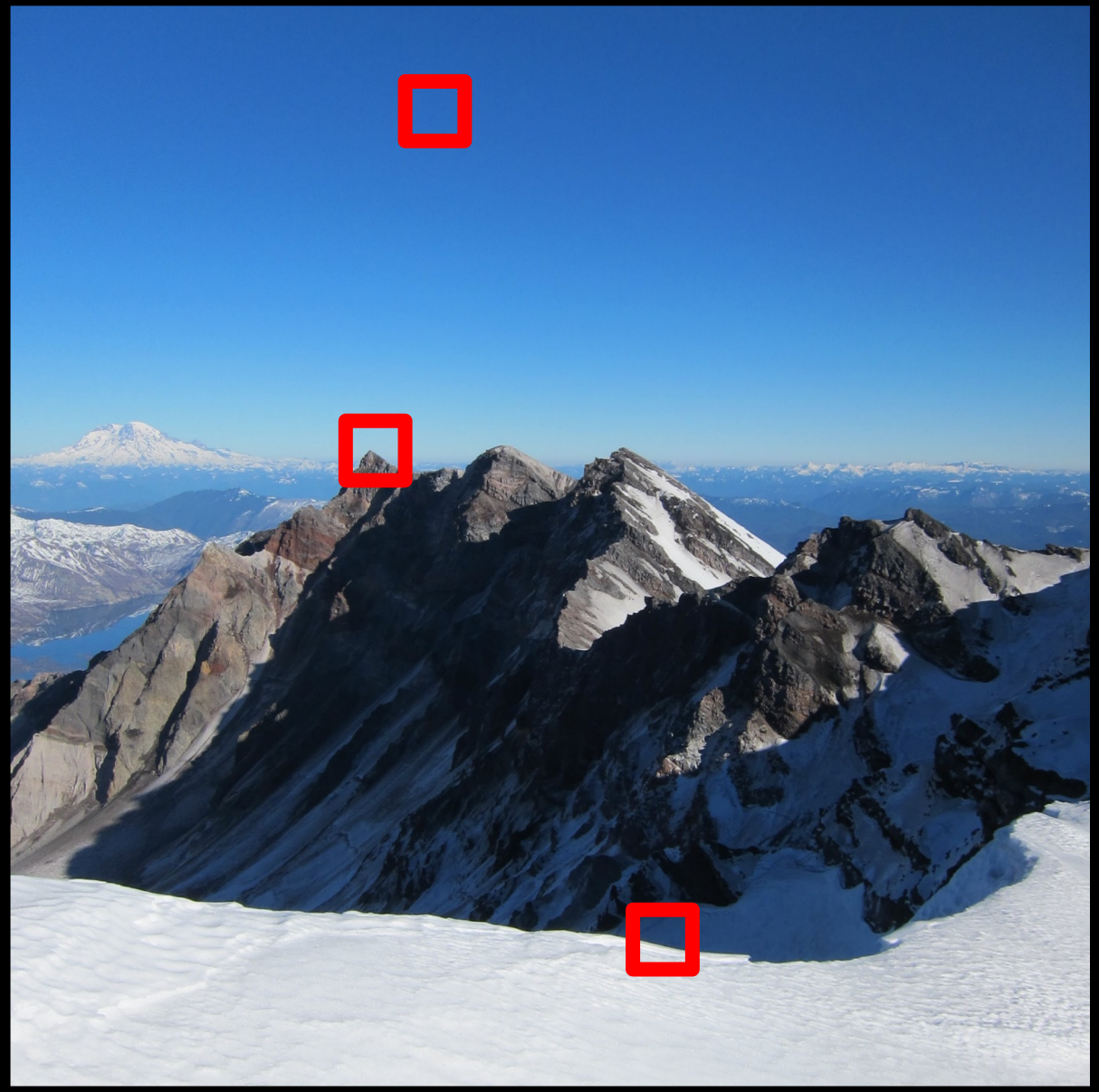
$$\Sigma_{\mathbf{d}}\Sigma_{x,y} (I(x,y) - I(x+\mathbf{d}_x,y+\mathbf{d}_y))^2$$

Measure of self-difference (how am I not myself?)

via JoeRedmon

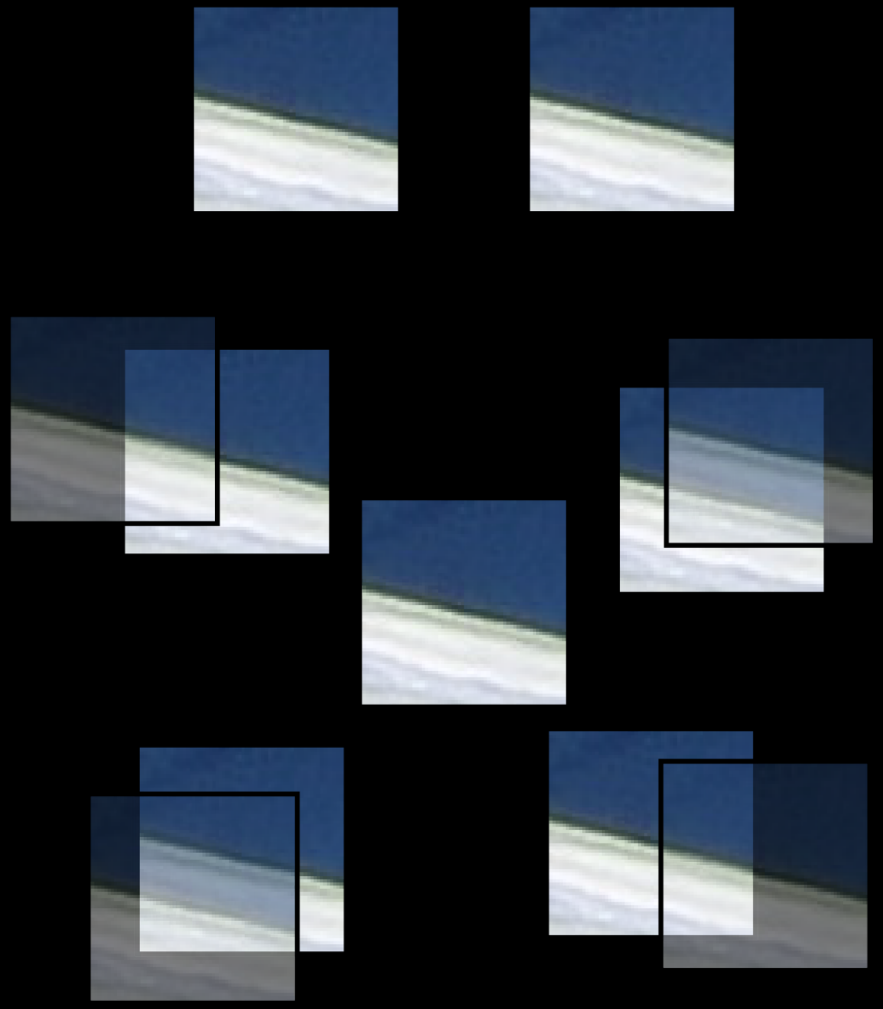
Self-difference

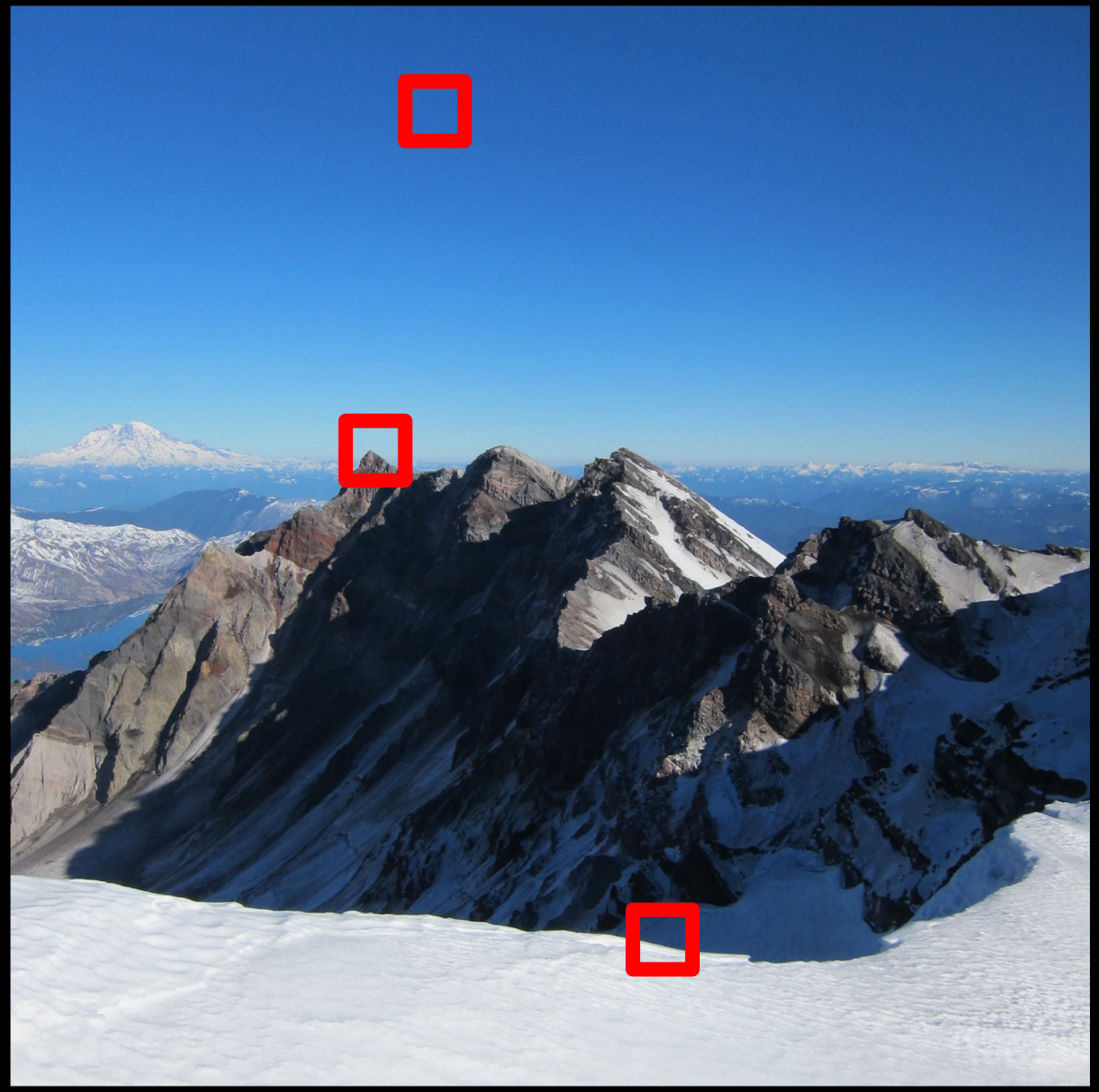




via JoeRedmon

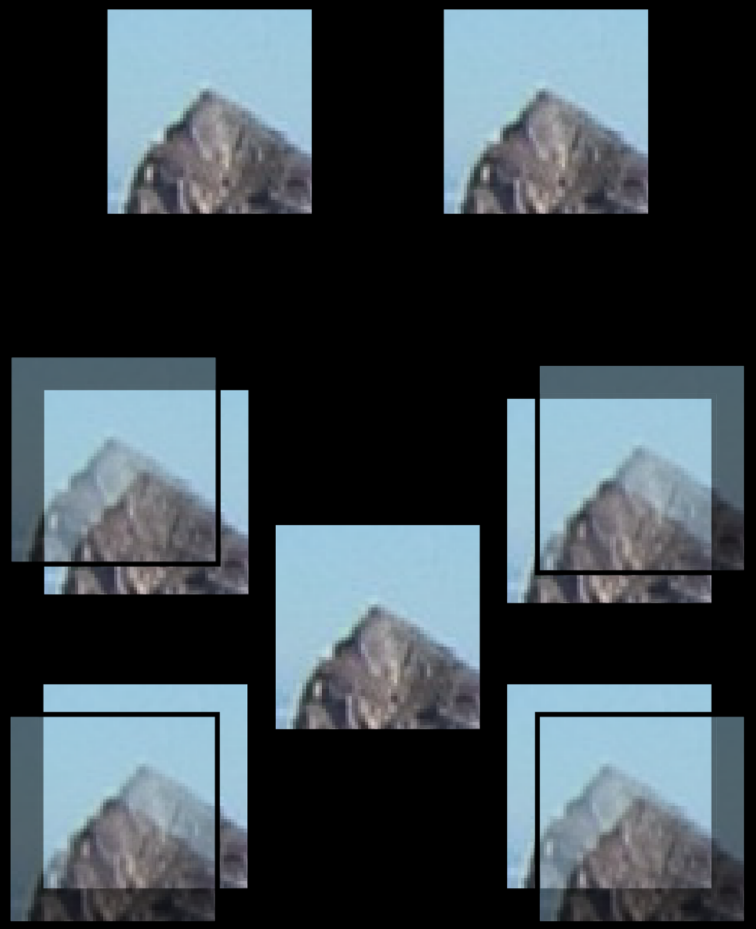
Self-difference

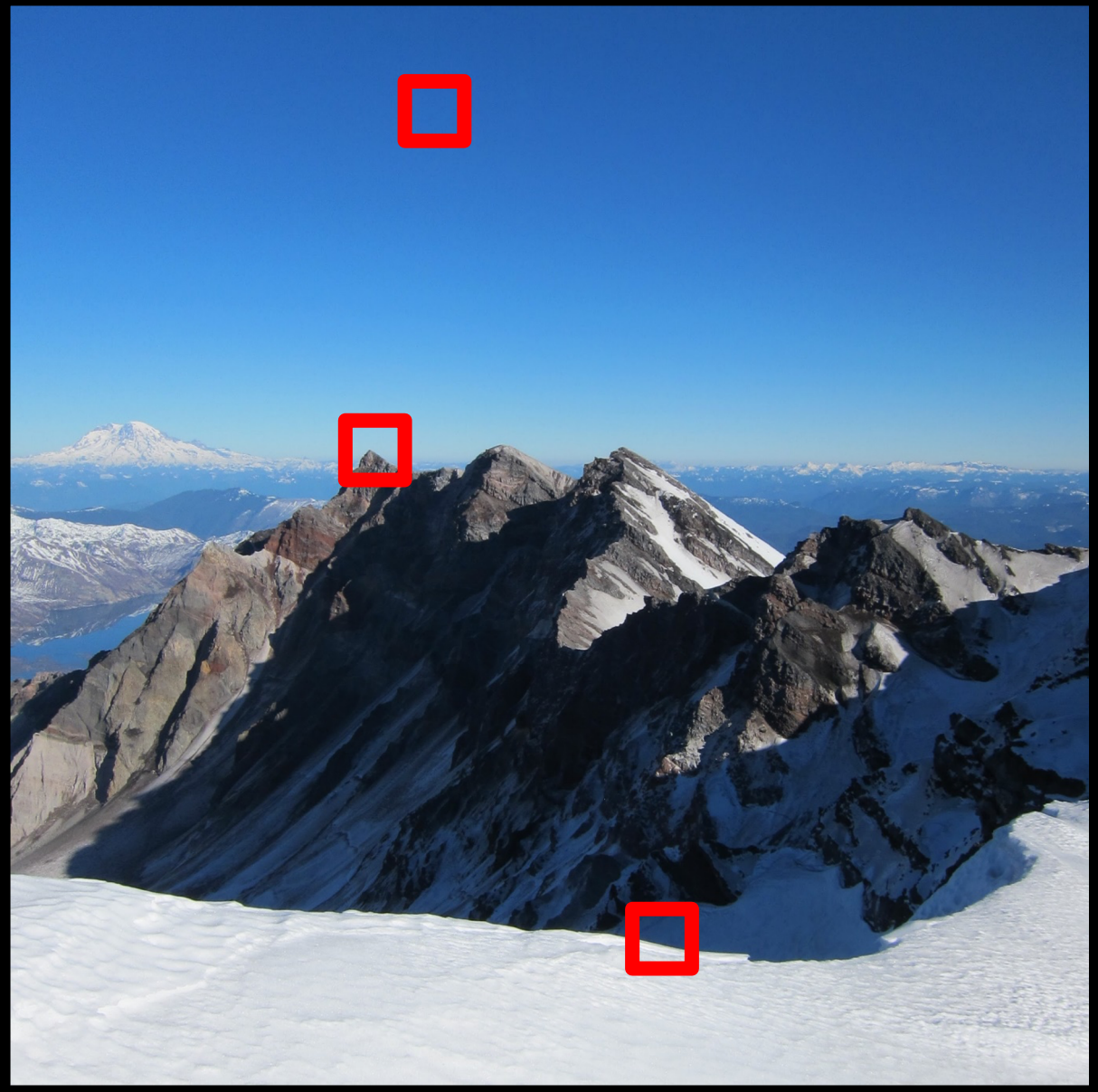




via JoeRedmon

Self-difference





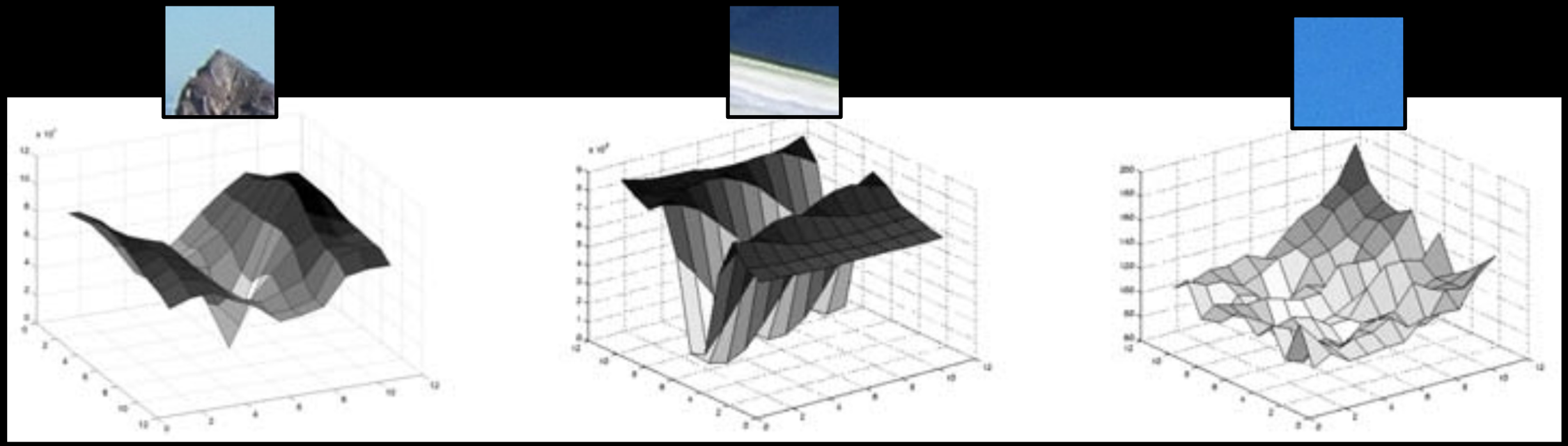
via JoeRedmon

Self-difference

Sky: low everywhere

Edge: low along edge

Corner: mostly high



Characterize Self-Difference to Compute Keypoints

- Compute an Auto-SSD or Auto-Correlation Surface locally at each point

$$E_{SSD}(\Delta u) = \sum_i w(\mathbf{x}_i)[I_0(\mathbf{x}_i + \Delta\mathbf{u}) - I_0(\mathbf{x}_i)]^2$$ Measures the sum-of-square differences as a function of (Δu)

- With first order approximation of image patch locally, we get: $E_{SSD}(\Delta u) =$ $\Delta\mathbf{u}^T \mathbf{A} \Delta\mathbf{u}$,

$$\mathbf{A} = \underbrace{\Sigma_i \, w_i \, * \begin{bmatrix} I_{x_i}^2 & I_{x_i} I_{y_i} \\ I_{x_i} I_{y_i} & I_{y_i}^2 \end{bmatrix}}$$

Structure Matrix

- Best Practice: Compute Smoothed Gradients with derivatives of Gaussian

- Eigen-structure of A captures the notion of flat or peaky self-similarity:

λ_1 and λ_2 are eigenvalues

λ_1 and λ_2 both small: no gradient (flat

$\lambda_1 >> \lambda_2$: gradient in one direction (edge)

λ_1 and λ_2 similar: multiple gradient directions, corner

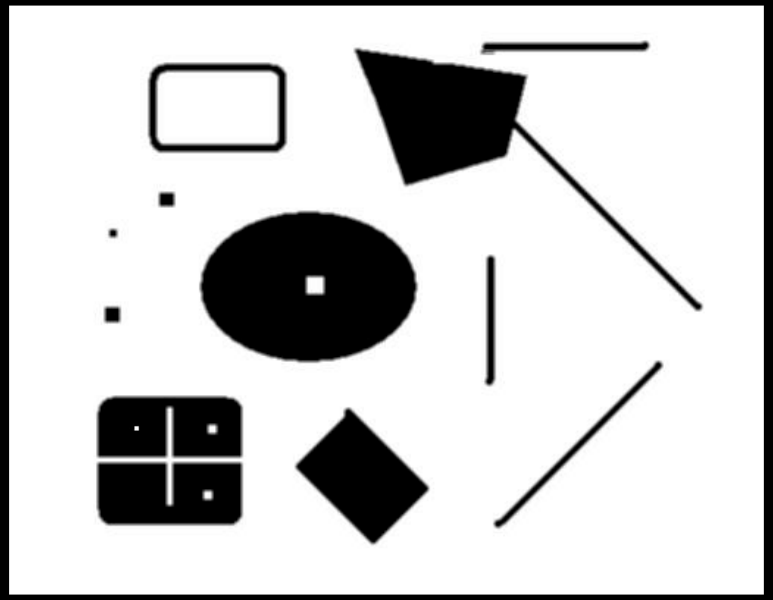
Harris Detector [Harris88]

via DerekHoeim

- "Second moment matrix" / Structure Matrix

$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

Weighted Summation can be implemented as Gaussian Smoothing of Squared Gradient Images

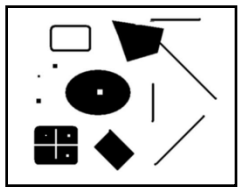


Intuition: Search for local neighborhoods where the image content has two main directions (eigenvectors).

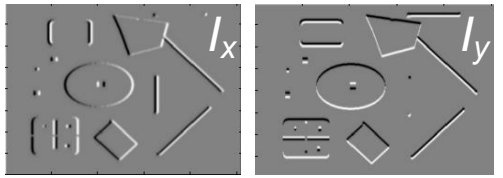
Harris Detector [Harris88]

- Second moment matrix

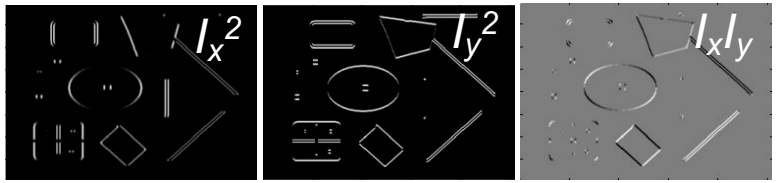
$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$



1. Image derivatives (optionally, blur first)



2. Square of derivatives



3. Gaussian filter $g(\sigma_I)$



$$\det M = \lambda_1 \lambda_2$$
$$\text{trace}\, M = \lambda_1 + \lambda_2$$

4. Cornerness function – both eigenvalues are strong

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I, \sigma_D))^2] =$$

$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$



5. Non-maxima suppression

Harris Detector: Mathematics

$$M = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Want large eigenvalues, and small ratio $\frac{\lambda_1}{\lambda_2} < t$

$$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

2. We know

$$\det M = \lambda_1 \lambda_2 \qquad \det M = m_{11}\, m_{22} - m_{12}\, m_{21}$$

$$\text{trace}\, M = \lambda_1 + \lambda_2 \qquad trace\, M = m_{11} + m_{22}$$

3. **Leads to**

$$\det M - k \cdot \text{trace}^2(M) > t$$

(k :empirical constant, k = 0.04-0.06)

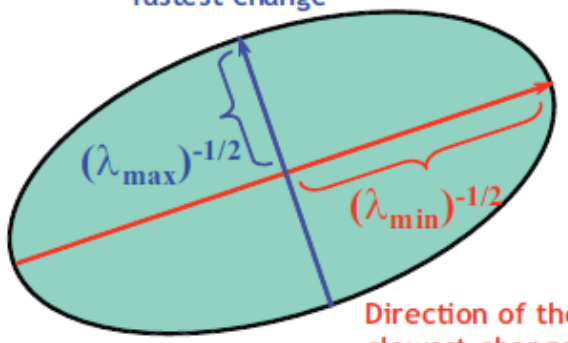
Nice brief derivation on wikipedia

Explanation of Harris Criterion

via DerekHoeim

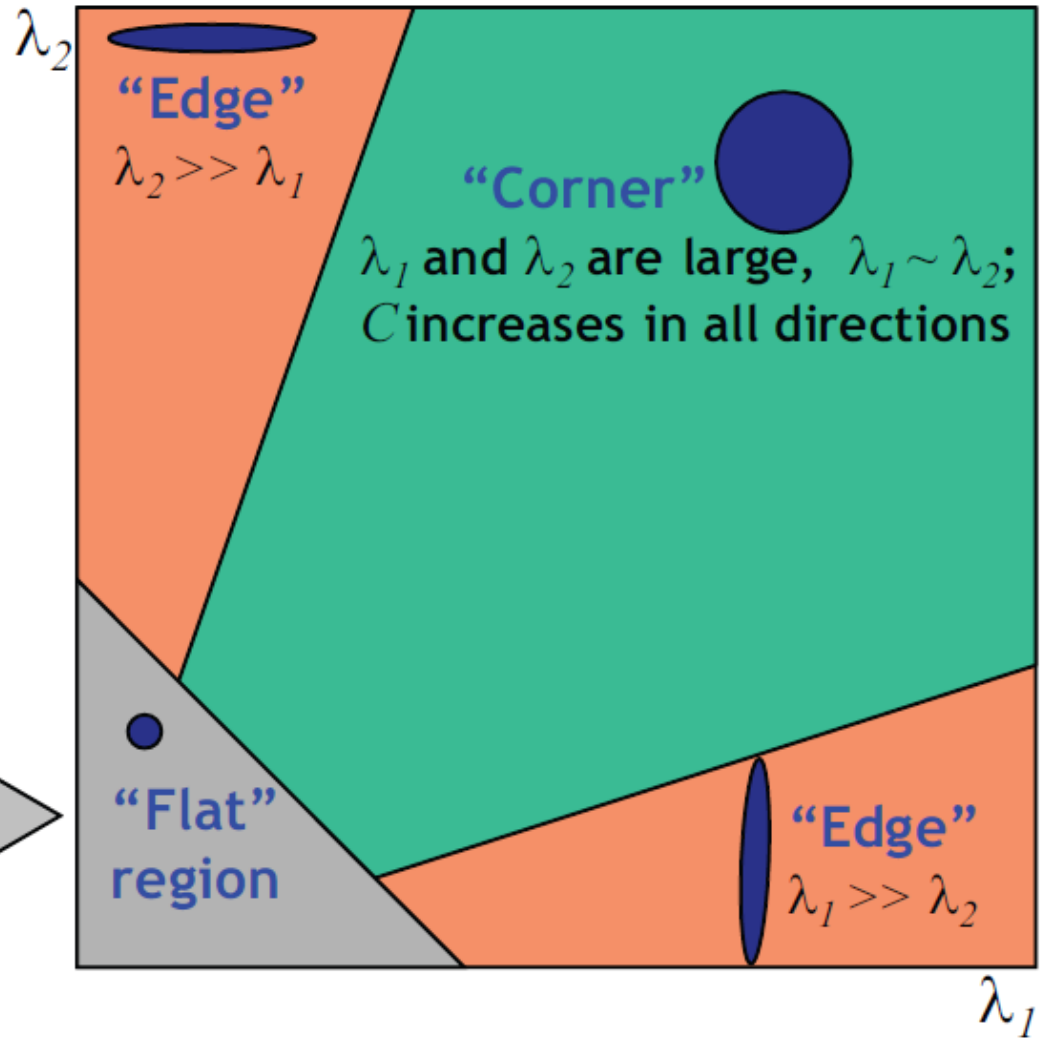
$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

Direction of the fastest change



Direction of the slowest change

λ_1 and λ_2 are small; C is almost constant in all directions



From Grauman and Leibe

Non-Maximal Suppression (NMS)

- Harris corner responses beyond a certain threshold typically create bunched up corner pixels
- Too much data to handle
- Suppress responses other than MAX within an NxN neighborhood everywhere. And choose top K responses.
- Alternatively use Adaptive NMS.
  - For each response compute the min distance to a local MAX response
  - Choose the Top K points with the largest radii

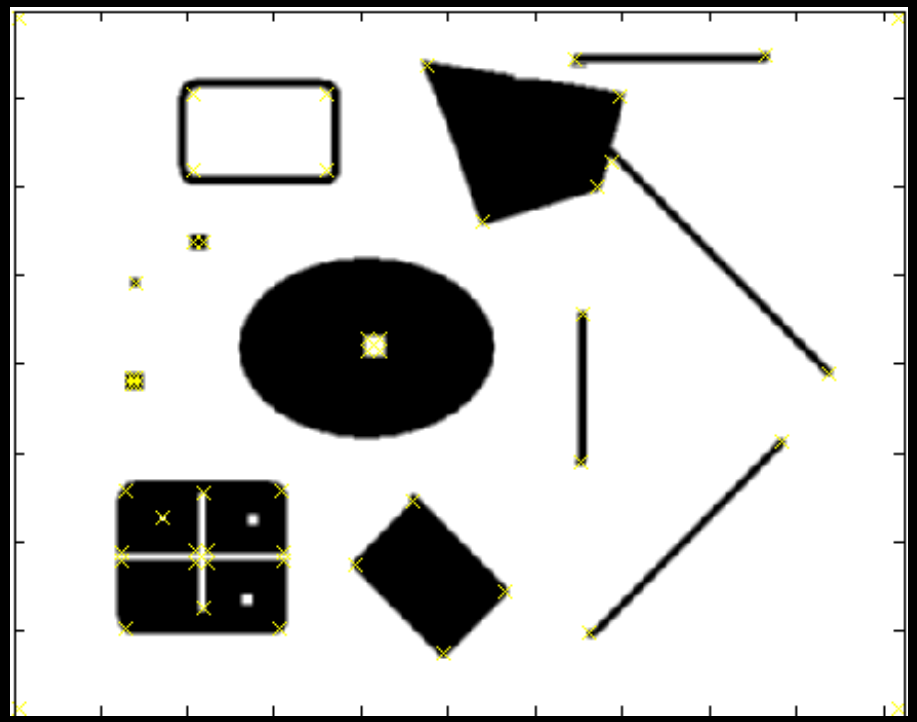


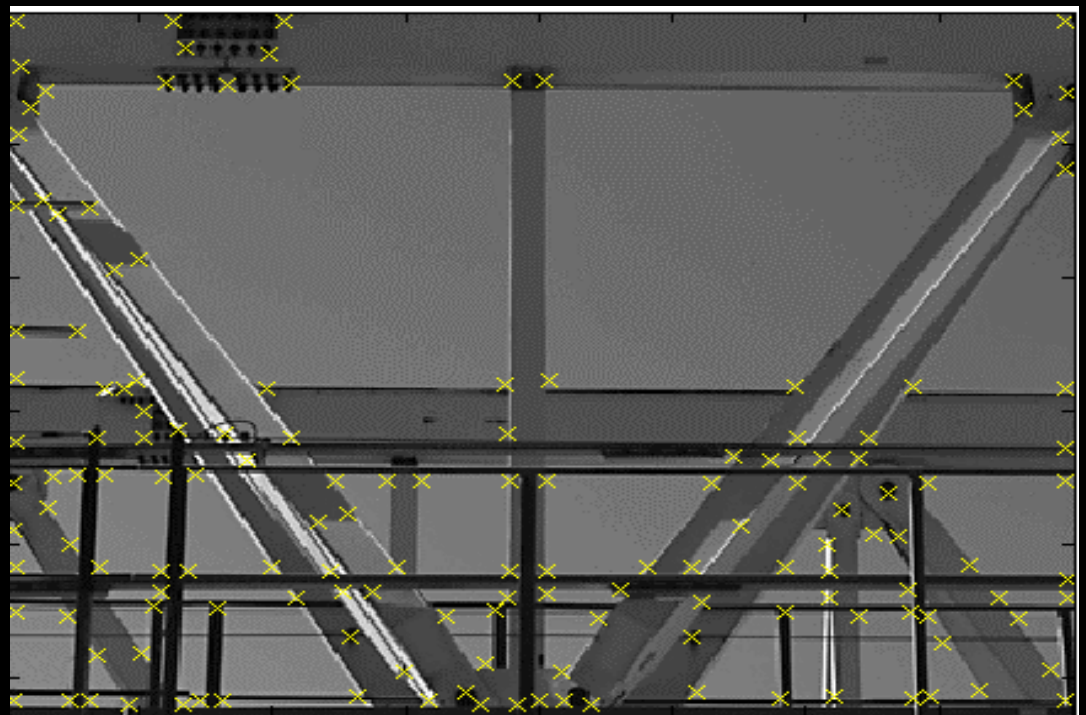


via MIT

Harris Detector – Responses [Harris88]

via DerekHoeim

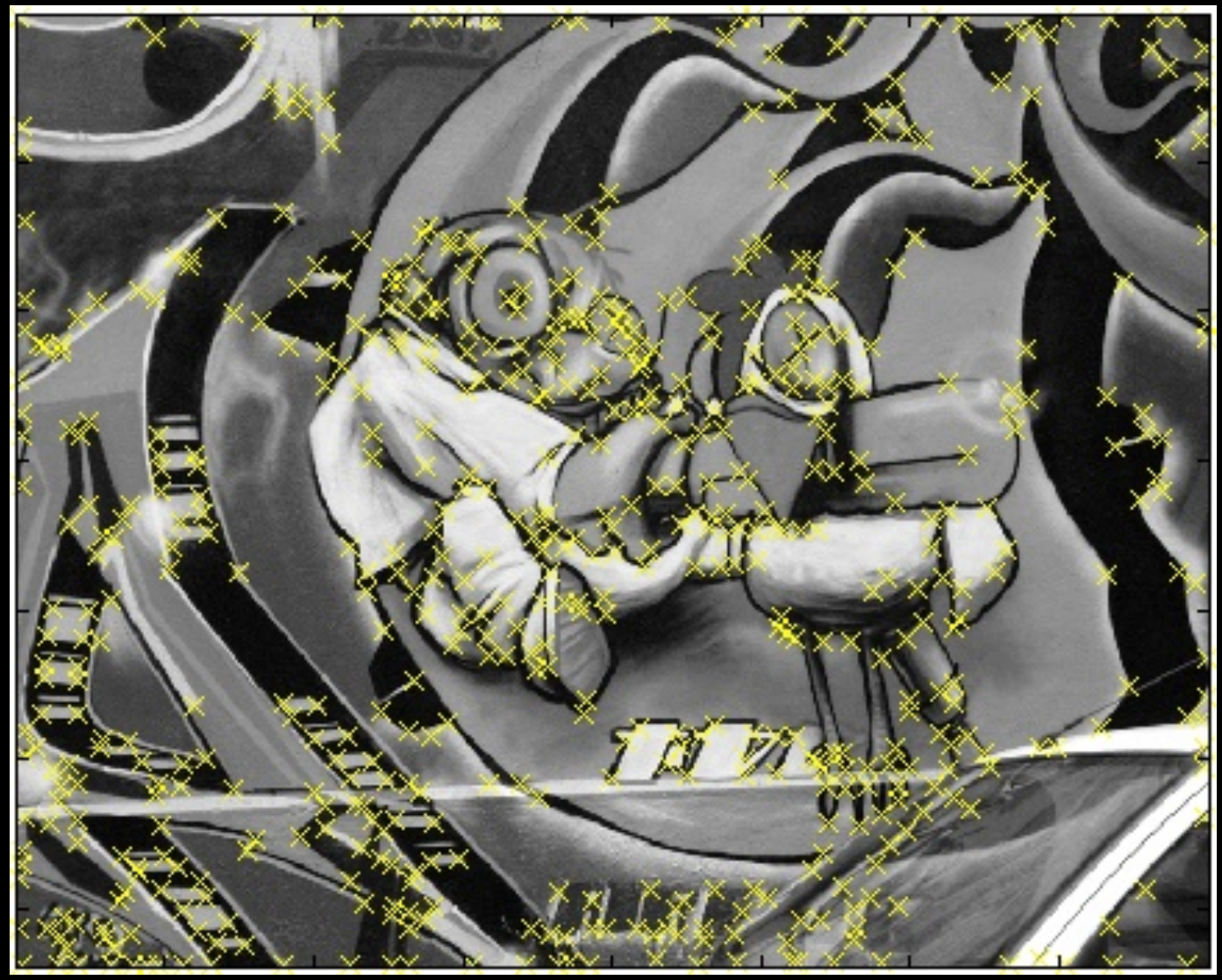




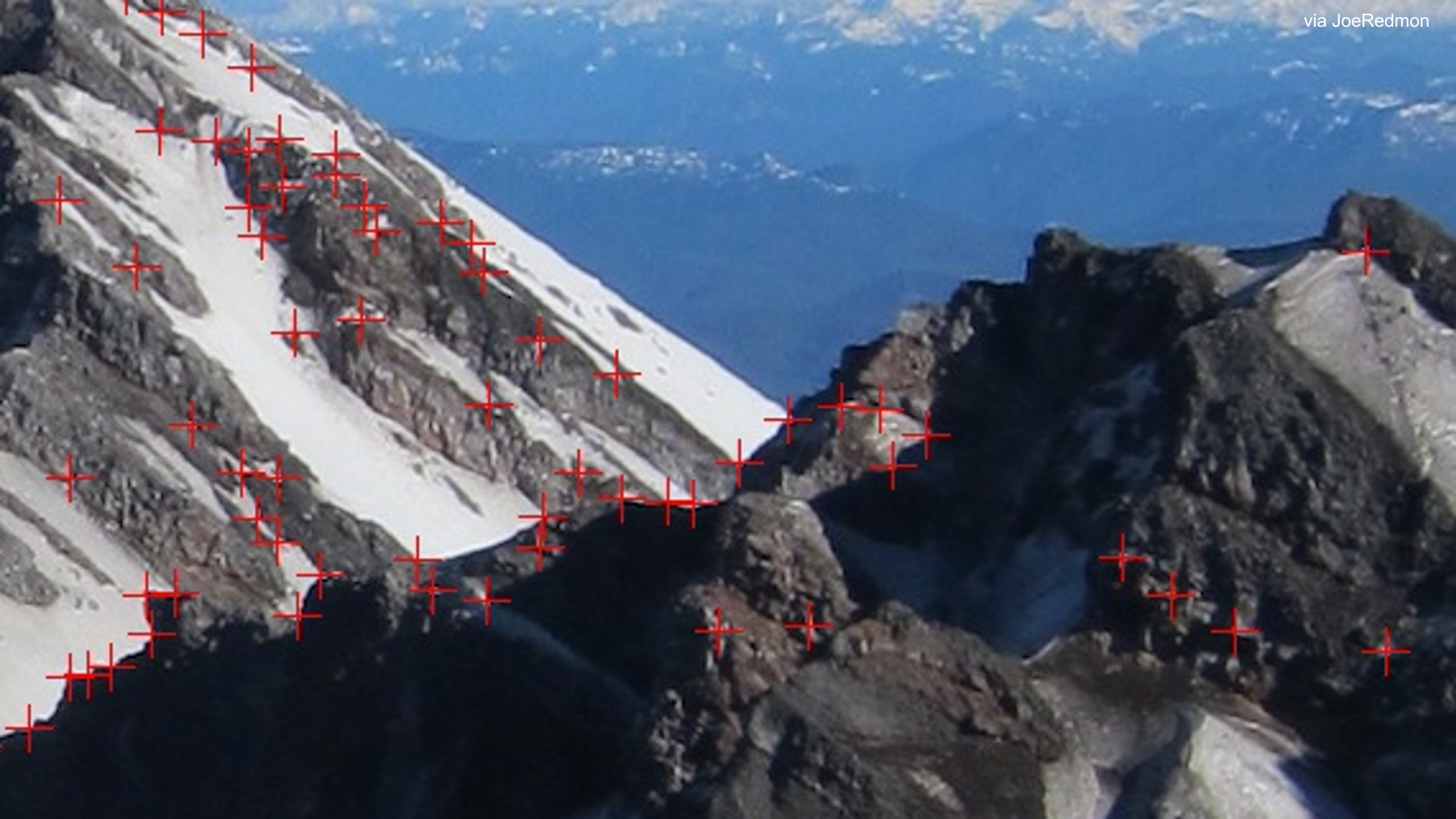
***Effect*: A very precise corner detector.**

Harris Detector - Responses [Harris88]

via DerekHoeim




via JoeRedmon


via JoeRedmon

via JoeRedmon

Ok, we found corners, now what?

Need to match image patches to each other

Need to figure out transform between images





via JoeRedmon

Ok, we found corners, now what?

Need to match image patches to each other

What is a patch? How do we look for matches? Pixels?

Need to figure out transform between images

How can we transform images?

How do we solve for this transformation given matches?





Matching patches: descriptors!

We want a way to represent an image patch

Can be very simple, just pixels!

Finding matching patch is easy, distance metric:

$\Sigma_{x,y} (I(x,y) - J(x,y))^2$

What problems are there with just using pixels?





Matching patches: descriptors!

We want a way to represent an image patch

Can be very simple, just pixels!

Finding matching patch is easy, distance metric:

$\Sigma_{x,y} (I(x,y) - J(x,y))^2$

What problems are there with just using pixels?





Short Baseline: Matching patches (descriptors!)

We want a way to represent an image patch

Can be very simple, just pixels!

Finding matching patch is easy, L2 distance metric:

$\Sigma_{x,y} (I(x,y) - J(x,y))^2$

What problems are there with just using pixels?

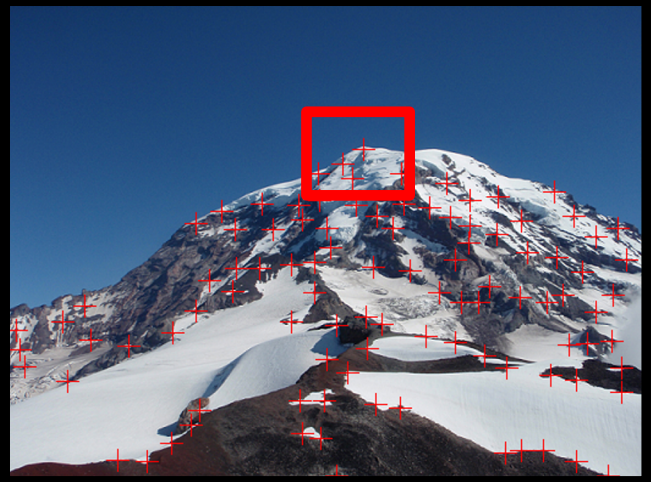
Descriptors can be more complex

Gradient information

How much context?

Edges, etc. we'll talk more about descriptors later!

Match Measures for matching Descriptors / Patches



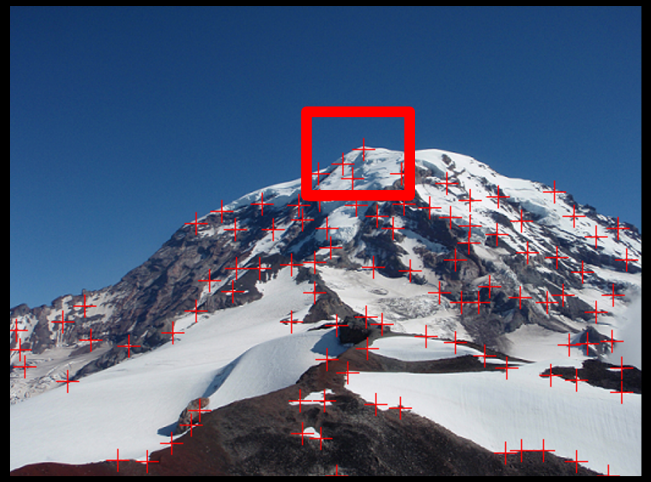


- $M = \sum_p (I(p) - J(p))^2$
- L2 Norm
- Sensitive to outliers
- Sensitive to brightness / contrast changes

- $M = \sum_p |I(p) - J(p)|$
- L1 Norm
- Less Sensitive to outliers
- Sensitive to brightness / contrast changes

- $M = 1 - \sum_p \frac{(I(p) - \bar{I})(J(p) - \bar{J})}{\sigma_{I_p} \sigma_{J_p}}$
- Inverse Normalized Correlation
- Sensitive to outliers
- Invariant to Local offset and scale changes in intensity/color

Match Measures for matching Descriptors / Patches





Can be used for subpixel matching:

- $M = \sum_p (I(p) - J(p))^2$
  - L2 Norm
  - Sensitive to outliers
  - Sensitive to brightness / contrast changes
- $M = \sum_p |I(p) - J(p)|$
  - L1 Norm
  - Less Sensitive to outliers
  - Sensitive to brightness / contrast changes
- $M = 1 - \sum_p \frac{(I(p) - \bar{I})(J(p) - \bar{J})}{\sigma_{I_p} \sigma_{J_p}}$
  - Inverse Normalized Correlation
  - Sensitive to outliers
  - Invariant to Local offset and scale changes in intensity/color

via JoeRedmon

Matching patches: descriptors!

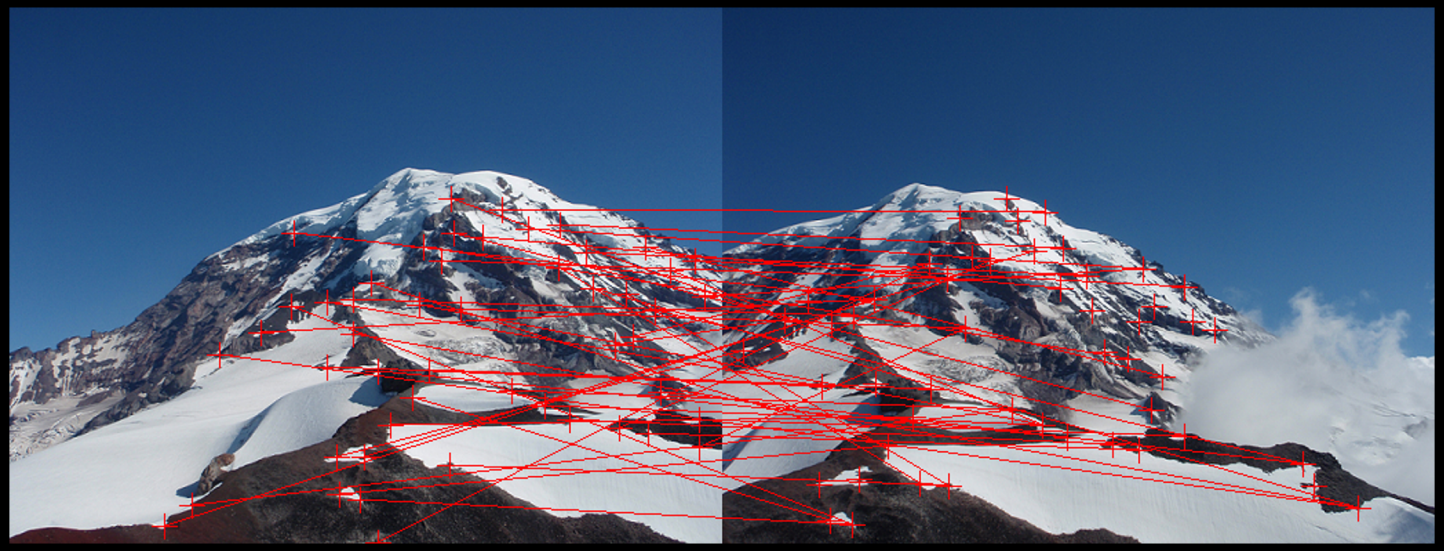
Already have our patches that are likely “unique”-ish

Loop over good patches in one image

    Find best match in other image

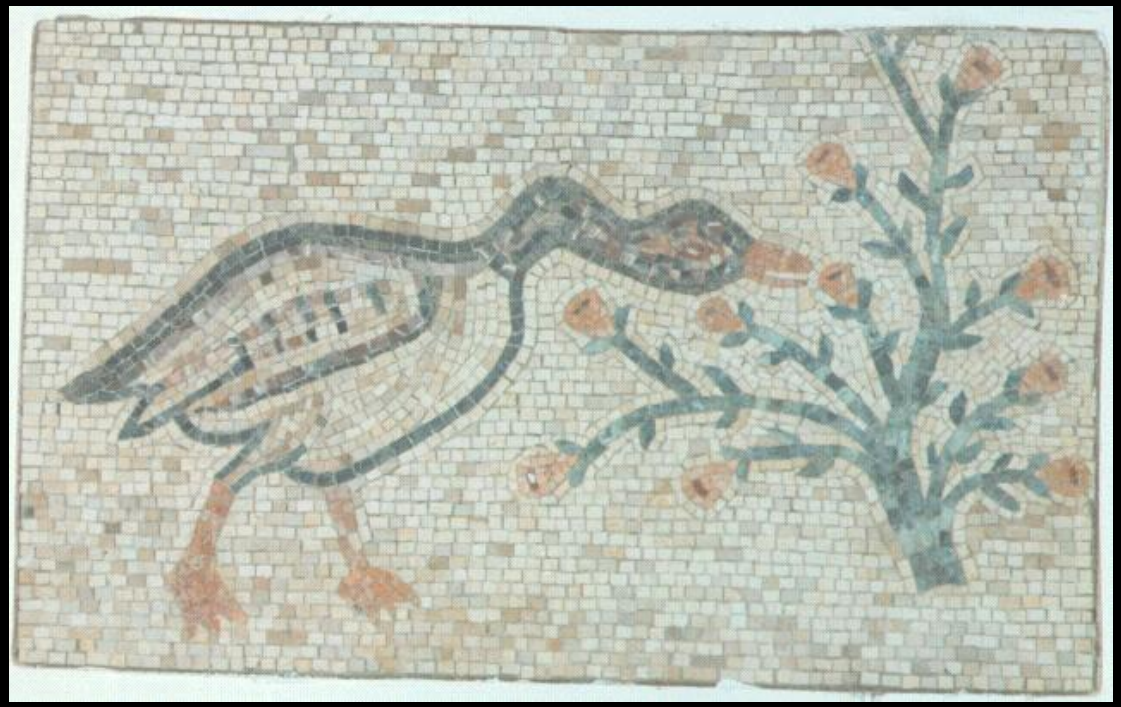
Find mutually best matches:

A = BestMatch(B) & B = BestMatch(A)



MOSAICS IN ART

…combine individual chips to create a big picture...



Part of the Byzantine mosaic floor that has been preserved in the Church of Multiplication in Tabkha (near the Sea of Galilee).
www.rtlsoft.com/mmmosaic

Image Transformations : Induced by Camera Motion

3D Rotations : Pan / Tilt

Image Displacment is Independent of Depth

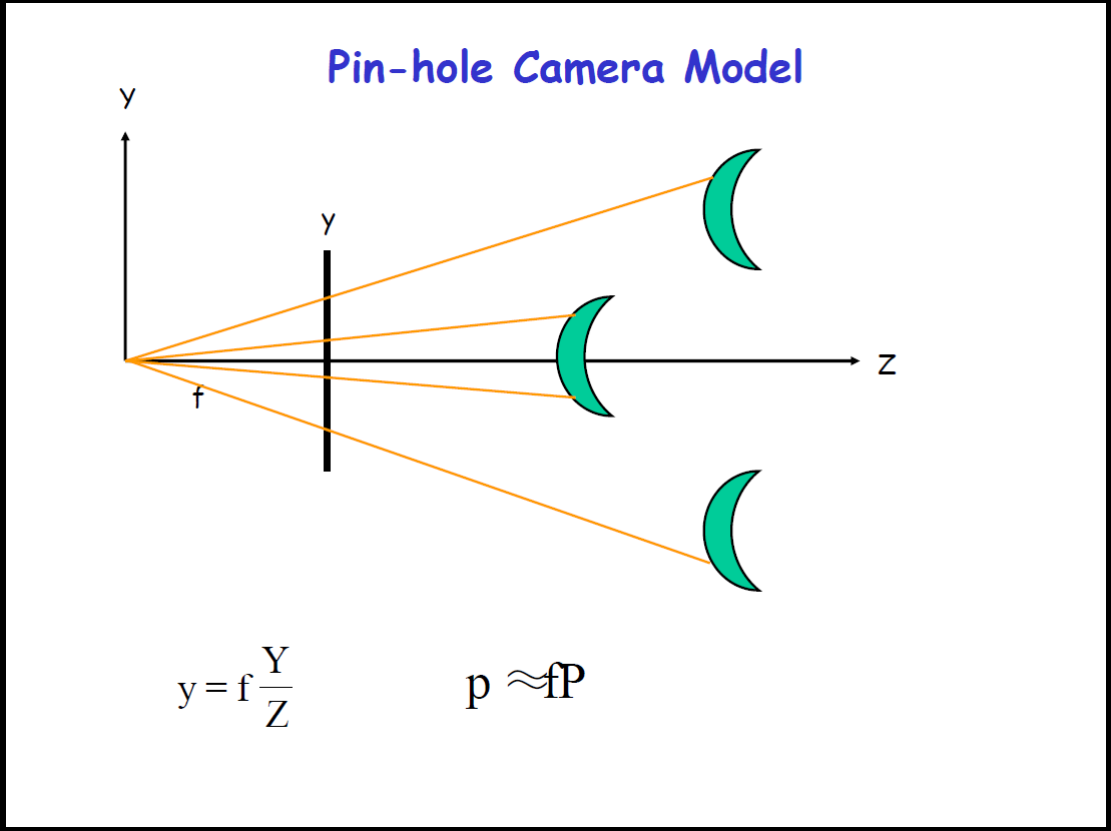


Image Transformations : Induced by Camera Motion

3D Rotations : Pan / Tilt

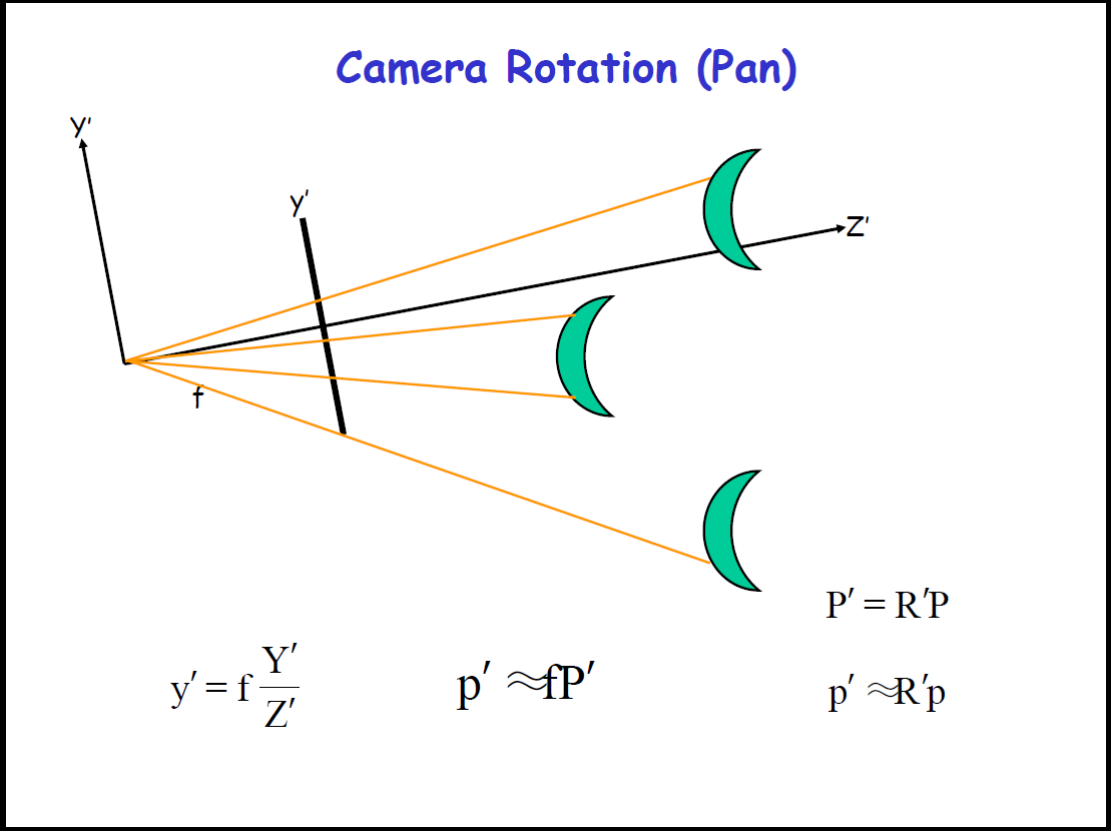
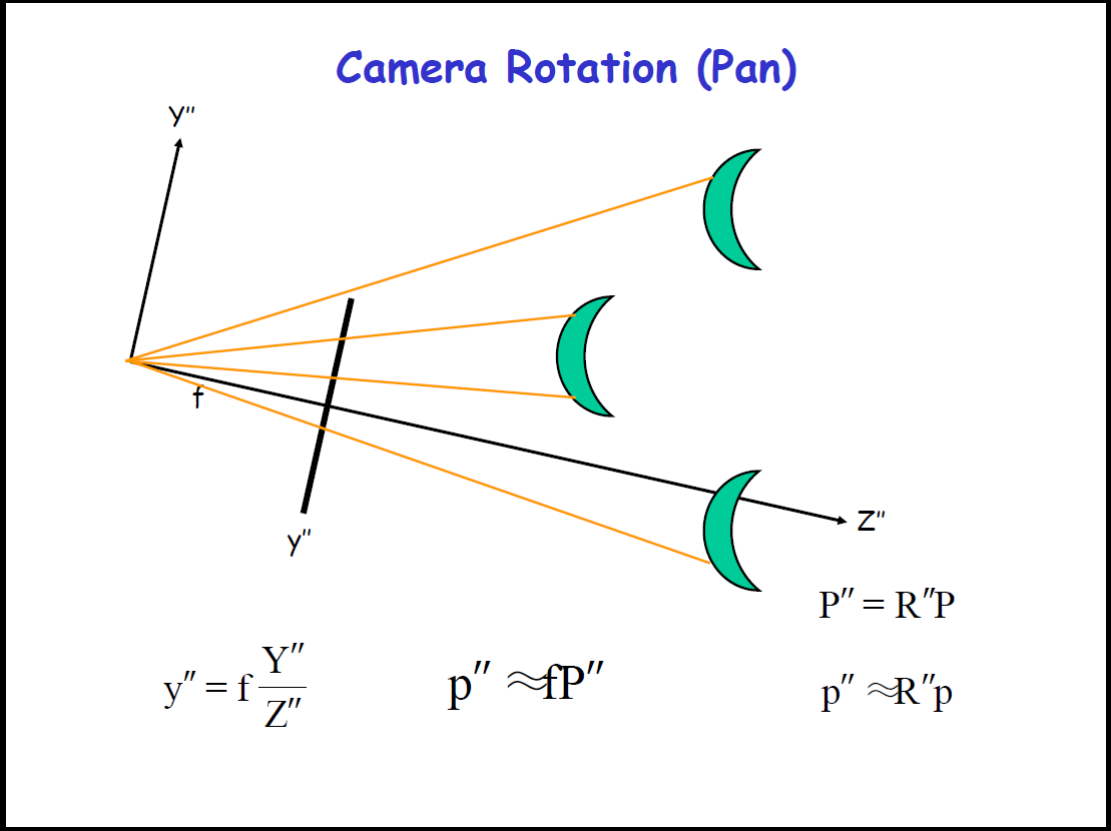
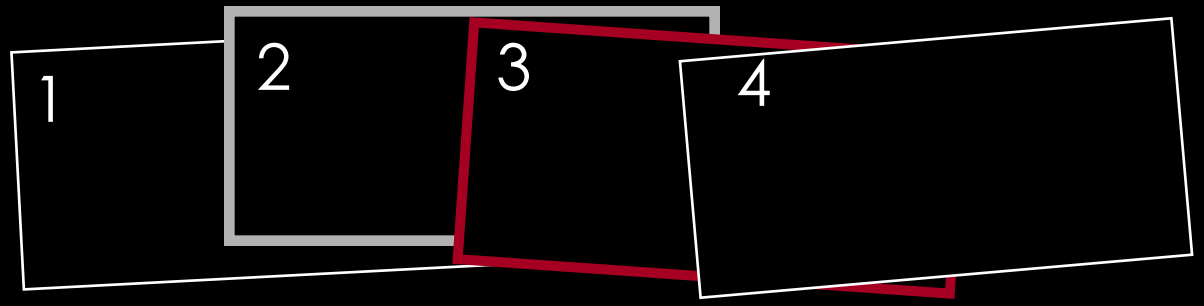


Image Transformations : Induced by Camera Motion

3D Rotations : Pan / Tilt



Basic Concept : Planar Mosaic Construction

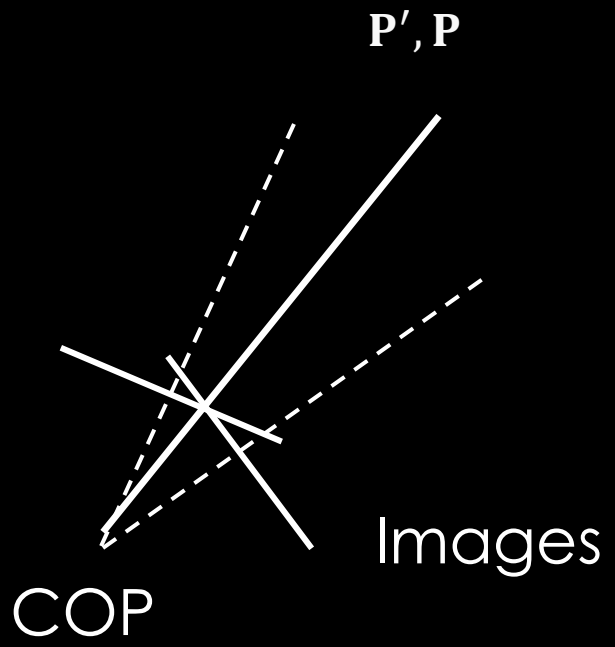


- Align Pairwise: 1:2, 2:3, 3:4, ...
- Select a Reference Frame
- Align all Images to the Reference Frame
- Combine into a Single Mosaic

Virtual Camera (Pan)
Image Surface - Plane
Projection - Perspective

Image Transformation : Rotations

What is the mapping from image rays to the mosaic coordinates ?



Rotations/Homographies
Plane Projective Transformations

$$\mathbf{P}' = \mathbf{RP}$$

$$\mathbf{p}'_\mathbf{c} \approx \mathbf{Rp}_\mathbf{c}$$

$$\mathbf{K}'\mathbf{p}' \approx \mathbf{RKp}$$

$$\mathbf{p}' \approx \mathbf{K}'^{-1}\mathbf{RKp}$$

$$\mathbf{p}' \approx \mathbf{H}_\infty\mathbf{p}$$

Image Transformations : Induced by Camera Motion

3D Translations : Image Displacement is a function of Depth (3D Parallax)

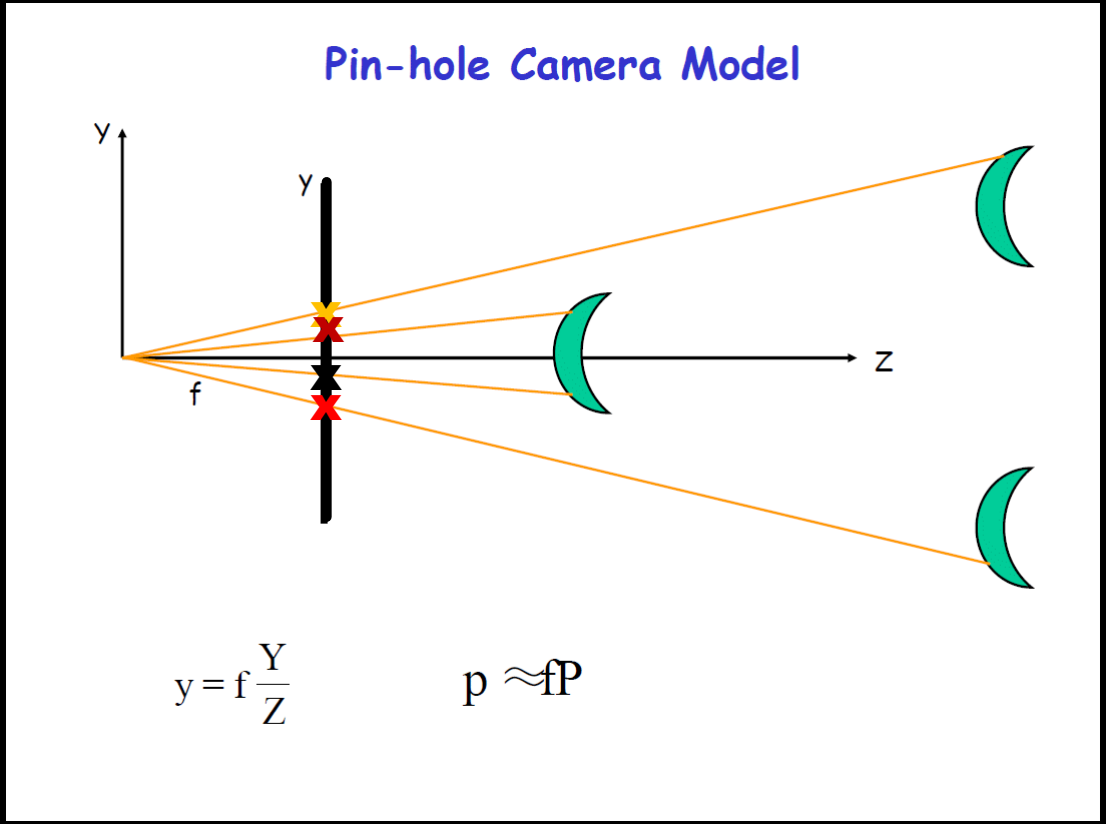


Image Transformations : Induced by Camera Motion

3D Translations : Image Displacement is a function of Depth (3D Parallax)

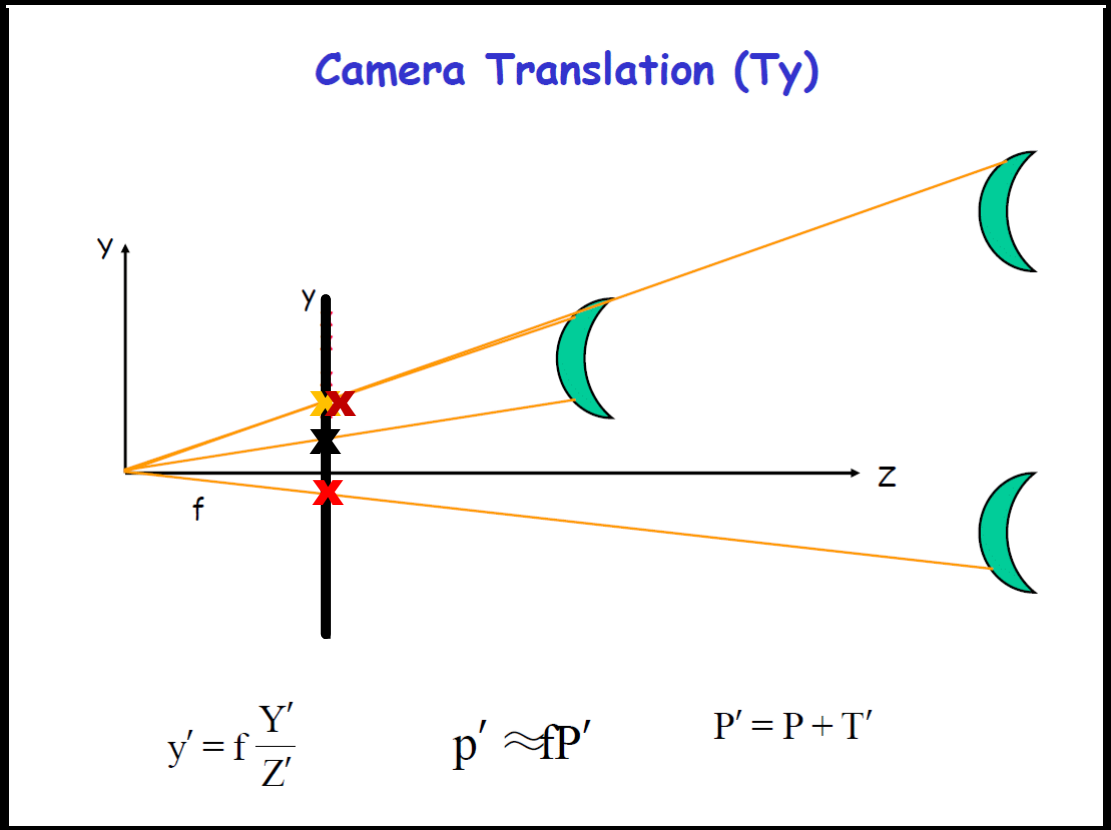


Image Transformation : Translations

Translational Displacement

$$y' = f\frac{Y'}{Z'} \qquad y' = f\frac{Y'}{Z'}$$

$$y' = f\frac{Y + Ty}{Z} \qquad y' = f\frac{Y}{Z + Tz}$$

$$y' - y = f\frac{Ty}{Z} \qquad y' - y = -y'\frac{Tz}{Z}$$

Image Motion due to Translation is a function of the depth of the scene

Image Transformation : Special Cases

Planar Scene (e.g. A Whiteboard)

- Plane Equation: $\frac{P^T N}{d} = 1$
- 3D Camera Motion: $P' = RP + T$
- Imaging a Plane Under Motion leads to:
- $p' \approx \left(R + \frac{T\,N^T}{d}\right) P$
- $p' \approx H\,p$

Scene at a Distance (e.g. Rainier)

- $\frac{\|T\|}{Z} \ll 1$ ➔ Image Displacement due to T is 0
- Same as the case of Rotational Motion
- Image Displacements well approximated by:

$$\mathbf{p}' \approx \mathbf{H}_\infty \mathbf{p}$$

Closer Look at Homography

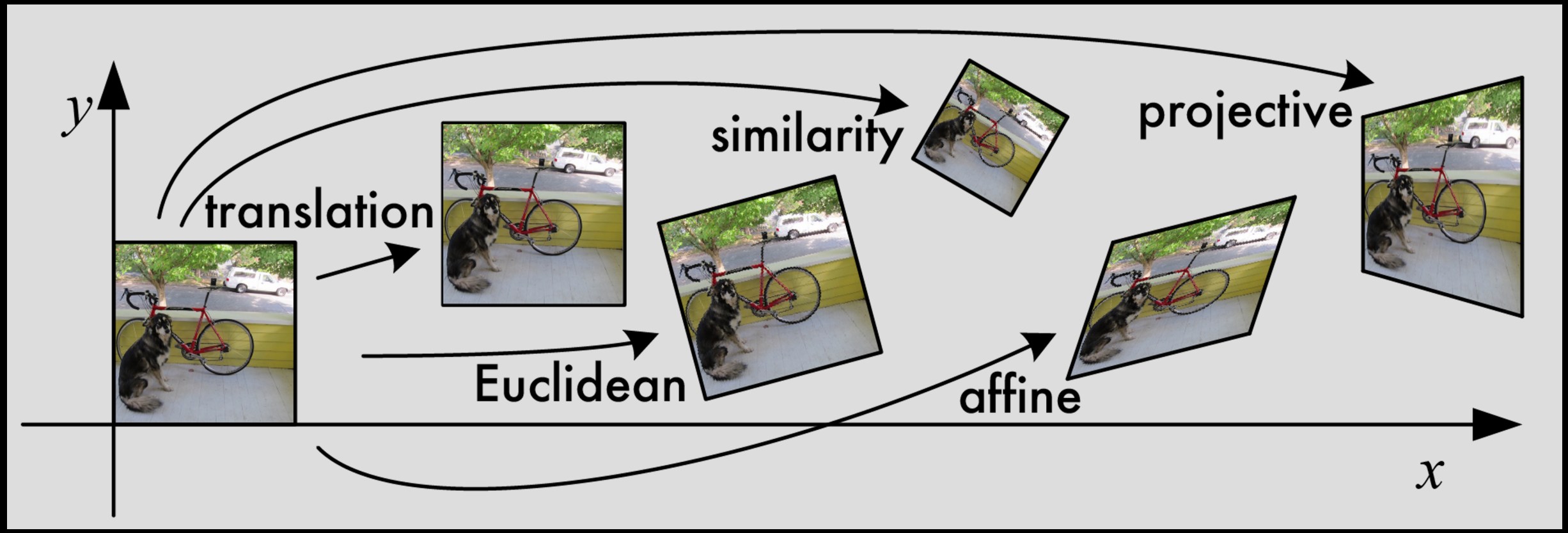
- $p' \approx H\,p$ $\quad p' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$ $\quad p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ are called Homogeneous coordinates.
- $H = \begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$ a 3x3 matrix relates view rays in images.

Transformation is upto an arbitrary scale

- View rays are represented as homogeneous coordinates. $p \approx \lambda\,p$ upto an arbitrary constant λ.
- Observed 2D image coordinates are related via:
- $x' = \frac{p'_x}{p'_z} = \frac{(1+h_{00})\,x + h_{01}\,y + h_{02}}{h_{20}x + h_{21}\,y + 1}$ $\quad y' = \frac{p'_y}{p'_z} = \frac{(h_{10})x + (1+h_{11})\quad y + h_{12}}{h_{20}x + h_{21}\,y + 1}$

via JoeRedmon

Approximations of Homography



Approximations of Homography

- Translation: $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$ — Preserves Orientations, Angles, Lengths, Areas
- Euclidean: $\begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$ — Preserves Angles, Lengths, Areas
- Similarity: $s * \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$ — Preserves Angles
- Affine: $\begin{bmatrix} a_{00} & a_{01} & t_x \\ a_{10} & a_{11} & t_y \\ 0 & 0 & 1 \end{bmatrix}$ — Preserves Parallelism
- Homography: : $\begin{bmatrix} (1+h_{00}) & h_{01} & h_{02} \\ h_{10} & (1+h_{11}) & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$ — Preserves Cross-Ratios

via JoeRedmon

Fitting Transformations to Noisy Matches

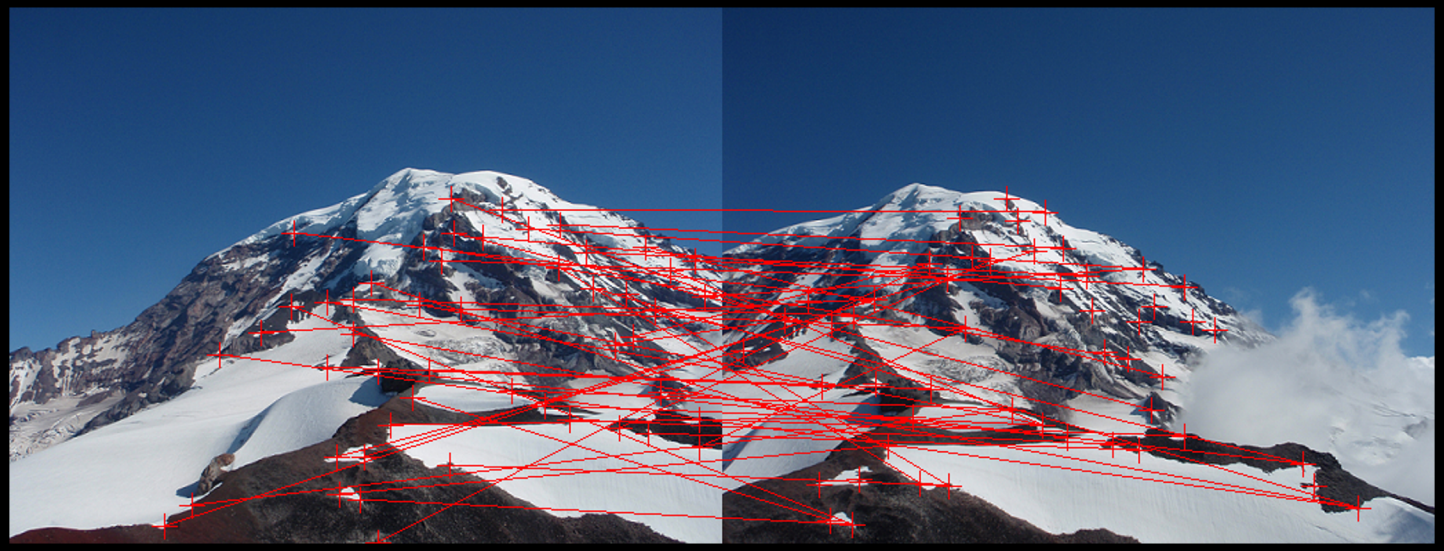
Already have our patches that are likely “unique”-ish

Loop over good patches in one image

    Find best match in other image

Find mutually best matches:

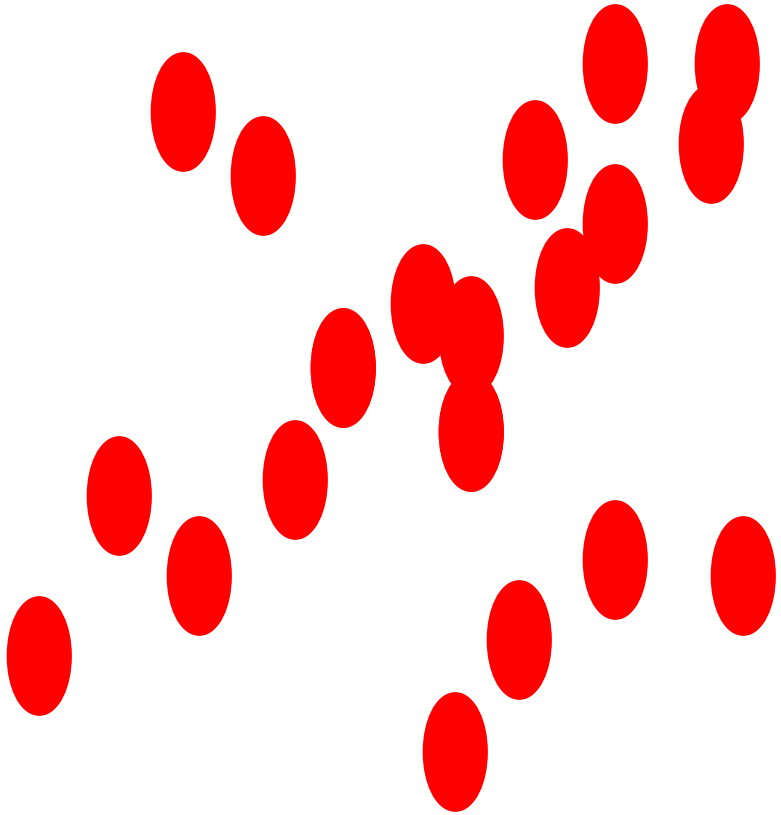
A = BestMatch(B) & B = BestMatch(A)



RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



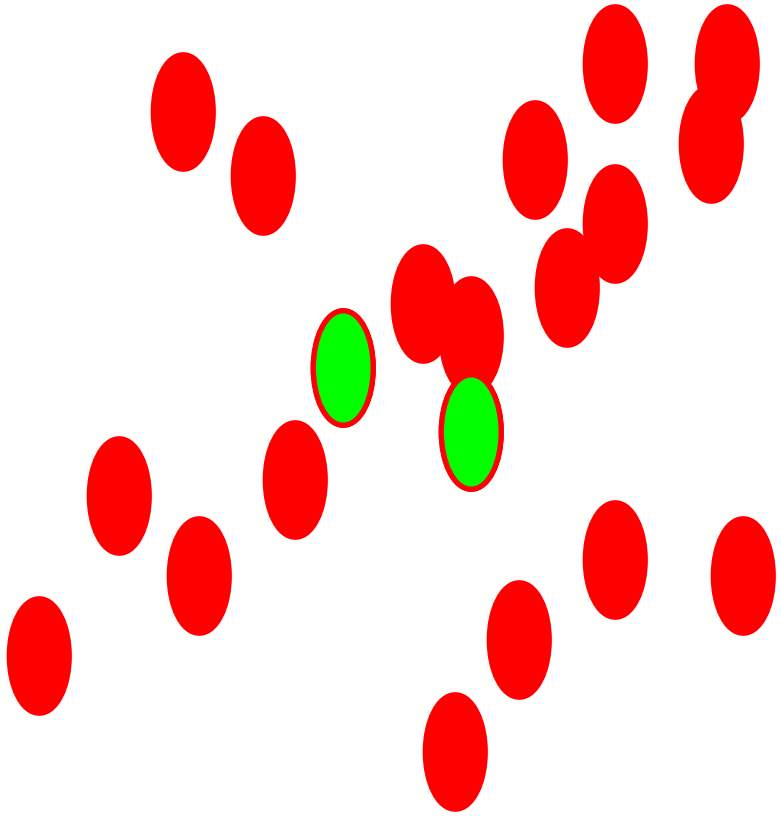
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



Algorithm:

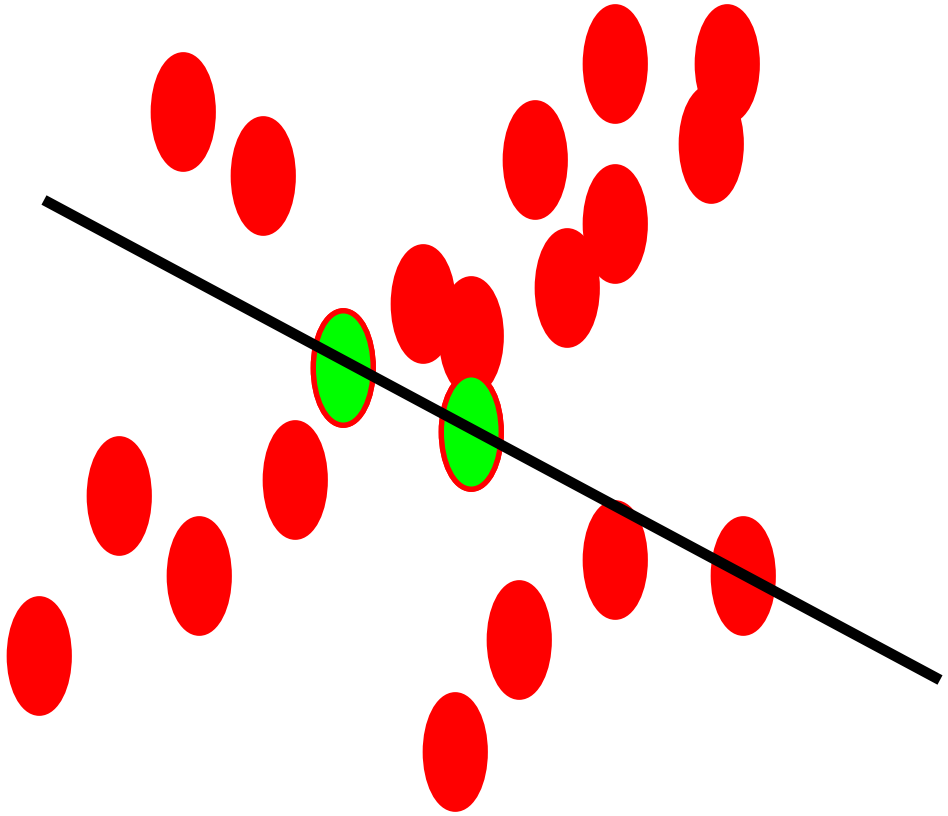
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

RANSAC

Line fitting example



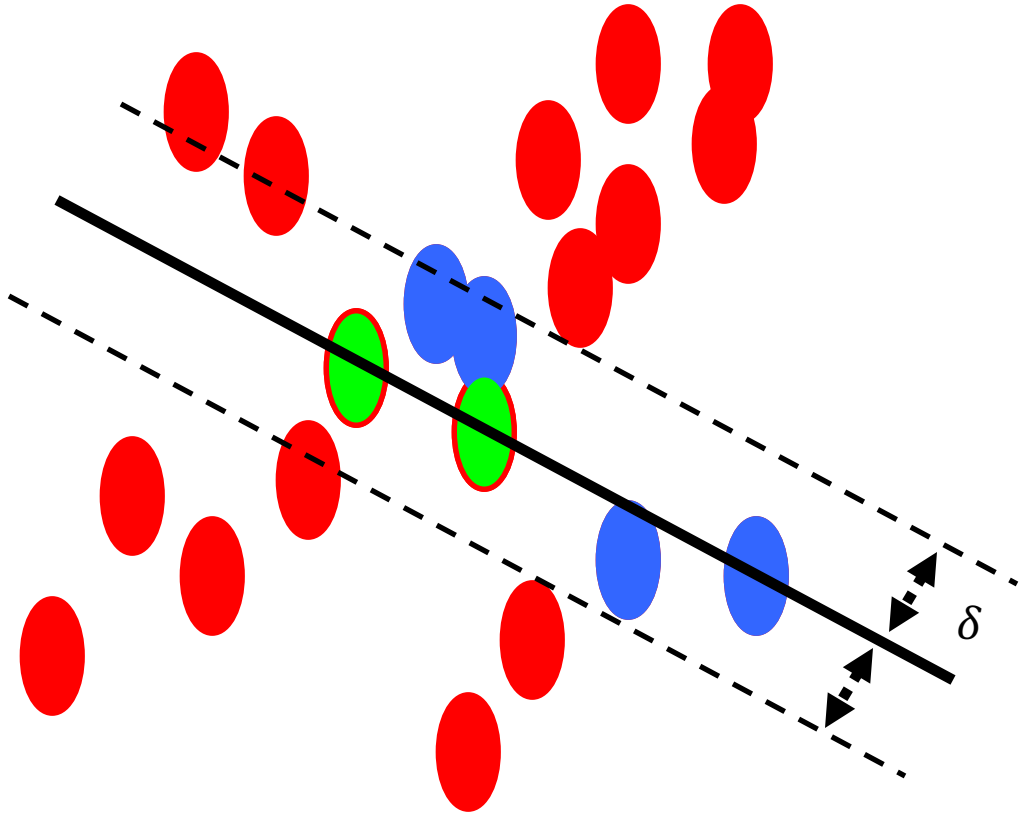
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



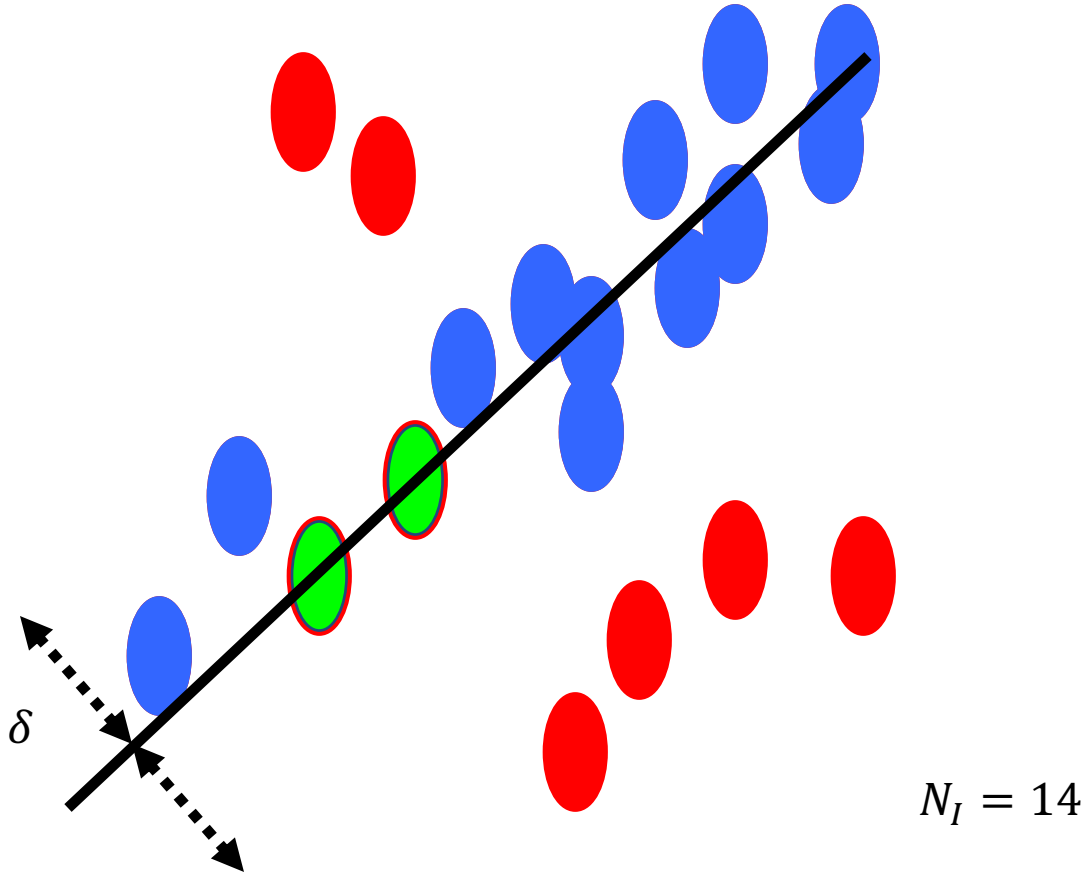
$N_I = 6$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

How to choose parameters?

- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)
- Number of sampled points *s*
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

$$N = \log(1-p) \,/\, \log(1-(1-e)^s)$$

	proportion of outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

modified from M. Pollefeys

RANSAC

Good

- Robust to outliers
- Applicable for larger number of objective function parameters than Hough transform
- Optimization parameters are easier to choose than Hough transform

Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not as good for getting multiple fits (though one solution is to remove inliers after each fit and repeat)

Common applications

- Computing a homography (e.g., image stitching)
- Estimating fundamental matrix (relating two views)

Solving for Homography with Correspondences: Linear Least Squares

- $x' = \frac{p'_x}{p'_z} = \frac{(1+h_{00})\,x + h_{01}\,y + h_{02}}{h_{20}x + h_{21}\,y + 1}$ $\qquad$ $y' = \frac{p'_y}{p'_z} = \frac{(h_{10})x + (1+h_{11})\quad y + h_{12}}{h_{20}x + h_{21}\,y + 1}$
- Given corresponding matches in two images (p, p'), find the best fit H
- Convert the rational equations above into linear algebraic form:
- For each correspondence:

Image Displacement

- $$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -\hat{x}'x & -\hat{x}'y \\ 0 & 0 & 0 & x & y & 1 & -\hat{y}'x & -\hat{y}'y \end{bmatrix} \begin{bmatrix} h_{00} \\ \vdots \\ h_{21} \end{bmatrix} = \begin{bmatrix} \hat{x}' - x \\ \hat{y}' - y \end{bmatrix}$$
- $A_i^T\ h = d_i$

Solving for Homography with Correspondences: Linear Least Squares

- $x' = \frac{p'_x}{p'_z} = \frac{(1+h_{00})\,x + h_{01}\,y + h_{02}}{h_{20}x + h_{21}\,y + 1}$ $\qquad$ $y' = \frac{p'_y}{p'_z} = \frac{(h_{10})x + (1+h_{11})\quad y + h_{12}}{h_{20}x + h_{21}\,y + 1}$
- Given corresponding matches in two images (p, p'), find the best fit H
- Convert the rational equations above into linear algebraic form:
- For each correspondence:

What is the minimum # of correspondences needed?

Image Displacement

- $$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -\hat{x}'x & -\hat{x}'y \\ 0 & 0 & 0 & x & y & 1 & -\hat{y}'x & -\hat{y}'y \end{bmatrix} \begin{bmatrix} h_{00} \\ \vdots \\ h_{21} \end{bmatrix} = \begin{bmatrix} \hat{x}' - x \\ \hat{y}' - y \end{bmatrix}$$
- $A_i^T\ h = d_i$

Solving for Homography with Correspondences: Linear Least Squares

- $A_i^T\ h = d_i$
- $\min_h \sum_i \left\| A_i^T\ h - d_i \right\|^2 \quad \rightarrow \quad \min_h \left(h^T \left(\sum_i A_i\ A_i^T \right) h - 2 \sum_i A_i^T\ d_i h \right)$
- Setting dervivative w.r.t. h to zero gives us:
- $\left(\sum_i A_i\ A_i^T \right) h = \sum_i A_i^T\ d_i \quad \leftrightarrow \quad \mathrm{A}\, h = b \quad \leftrightarrow \quad h = \mathrm{A}^{-1}\, b$
- Sub-optimal since Algebraic error does not account for measurement errors in keypoints and matches
- Can normalize the error with Jacobian (see Book)

Optimal Solution for Homography : Non-Linear Least Squares with Levenberg-Marquardt

- $x' = \frac{p'_x}{p'_z} = \frac{(1+h_{00})\,x + h_{01}\,y + h_{02}}{h_{20}x + h_{21}\,y + 1}$ $\qquad y' = \frac{p'_y}{p'_z} = \frac{(h_{10})x + (1+h_{11})\;\; y + h_{12}}{h_{20}x + h_{21}\,y + 1}$ (Non-linear in h)

- $\min_{\Delta h} \sum_i \; \|f(p_i;\, h + \Delta h) - p'_i\|^2 + \lambda_h \|\Delta h\|^2 \quad \approx \quad \min_{\Delta h} \sum_i \; \|J(p_i;\, h)\Delta h - r_i\|^2 + \lambda_h \|\Delta h\|^2$

  $= \min_{\Delta h} \; \Delta h^T \left[\sum_i J_i^T J_i + \lambda_h I\right]\Delta h - 2\,\Delta h^T \left[\sum_i J_i^T r_i\right] + \sum_i \|r_i\|^2$: Computed at a current estimate h

- Setting derivative w.r.t. Δh to zero gives us:

- $\Delta h = \left[\sum_i J_i^T J_i + \lambda_h I\right]^{-1}\left[\sum_i J_i^T r_i\right]$ $\qquad \lambda_h \gg 1$ Gradient Descent $\qquad \lambda_h = 0$ Gauss-Newton

- Far from the local optimal solution, keep λ_h high so that descent is guaranteed.

- Closer to the minimum $\lambda_h = 0$ to get faster convergence

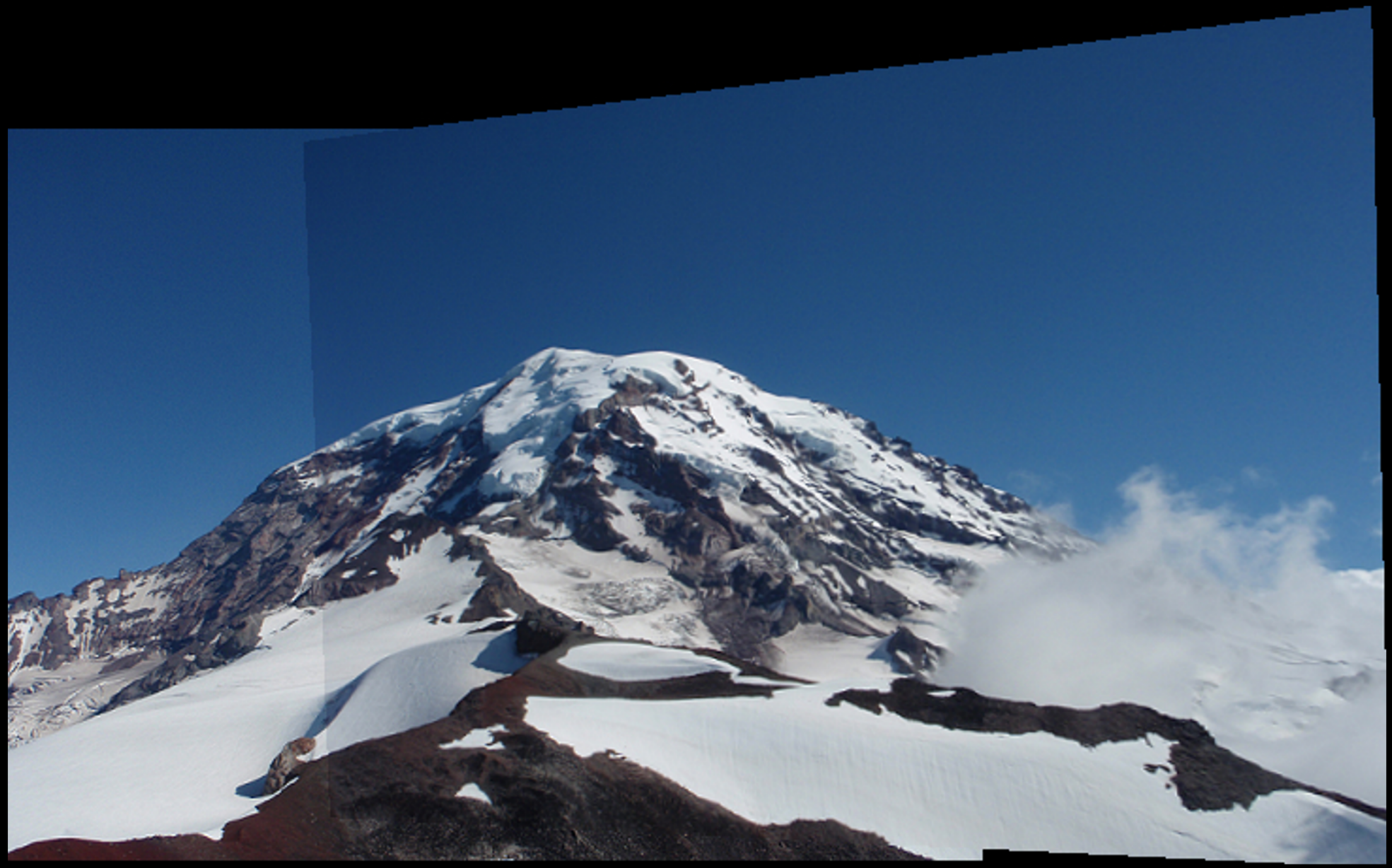
via JoeRedmon

In practice:

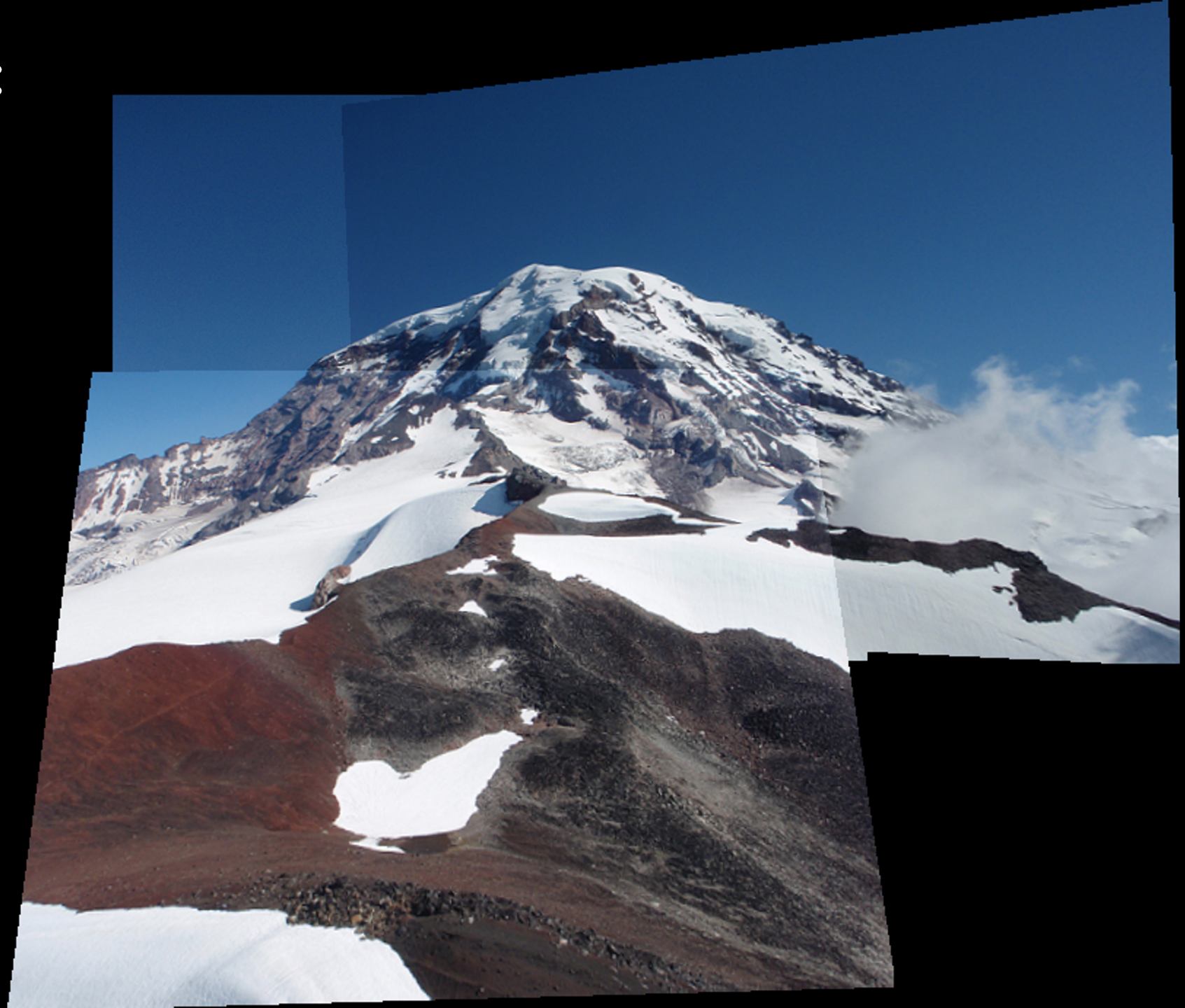


via JoeRedmon

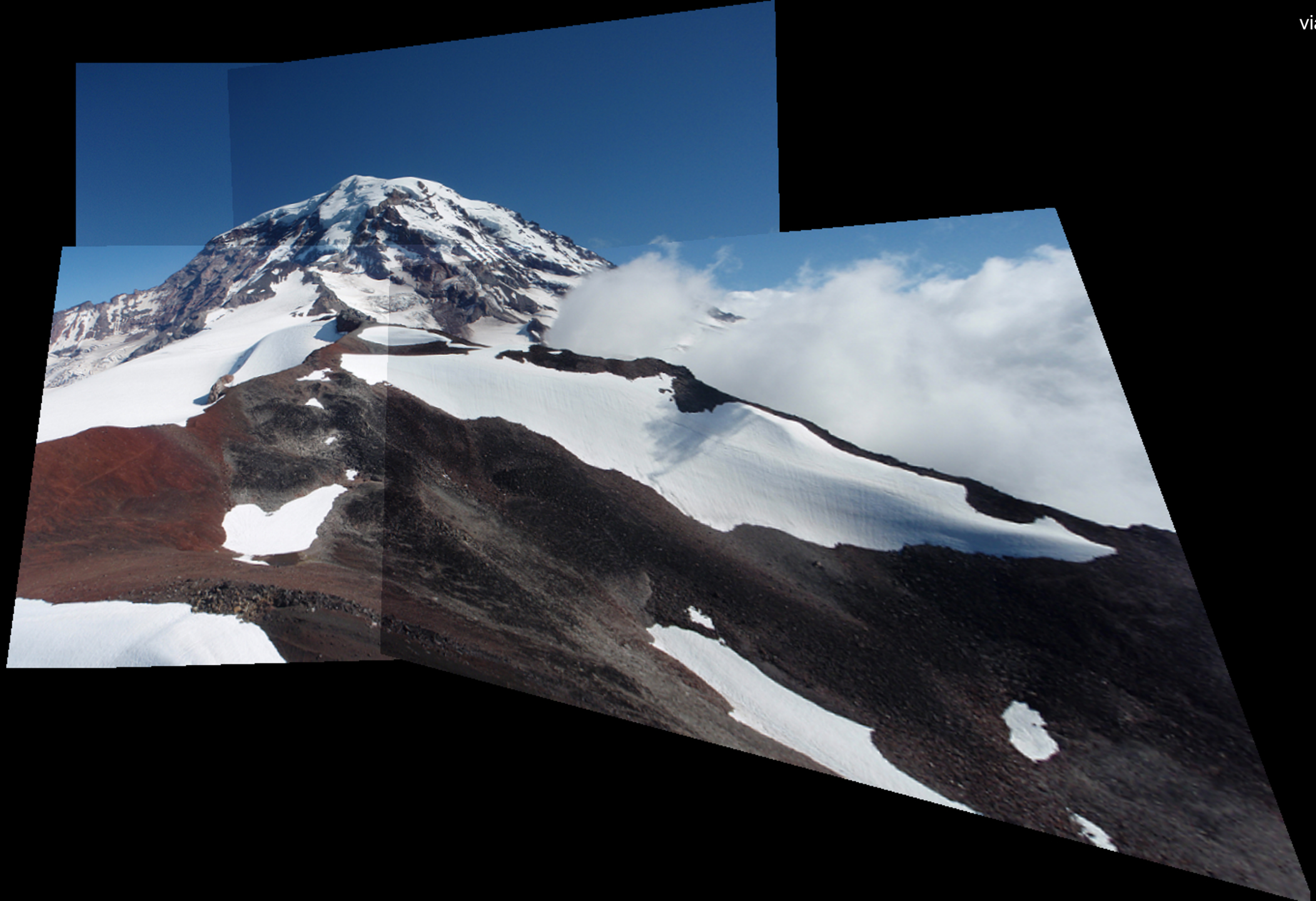
In practice:

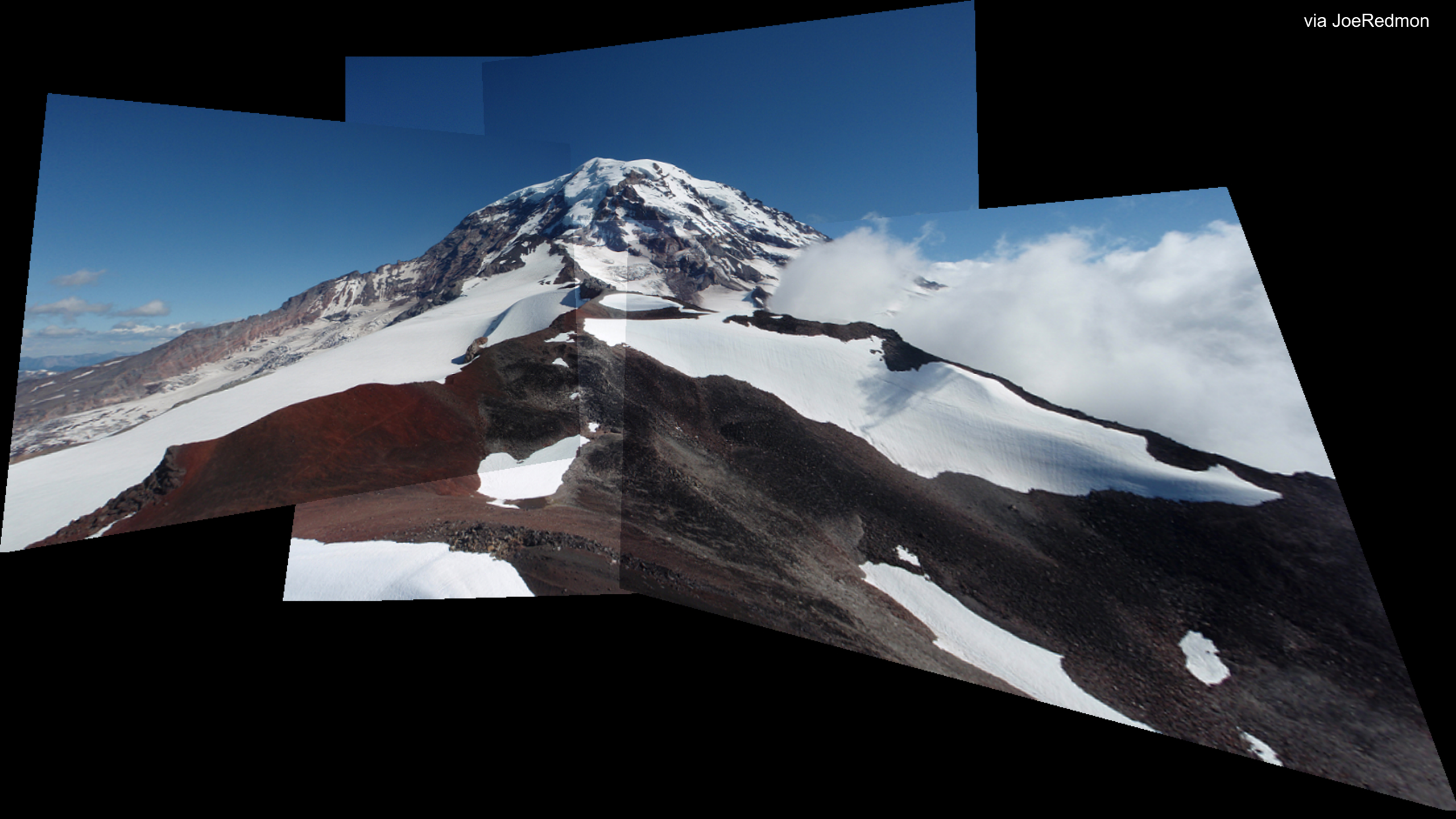


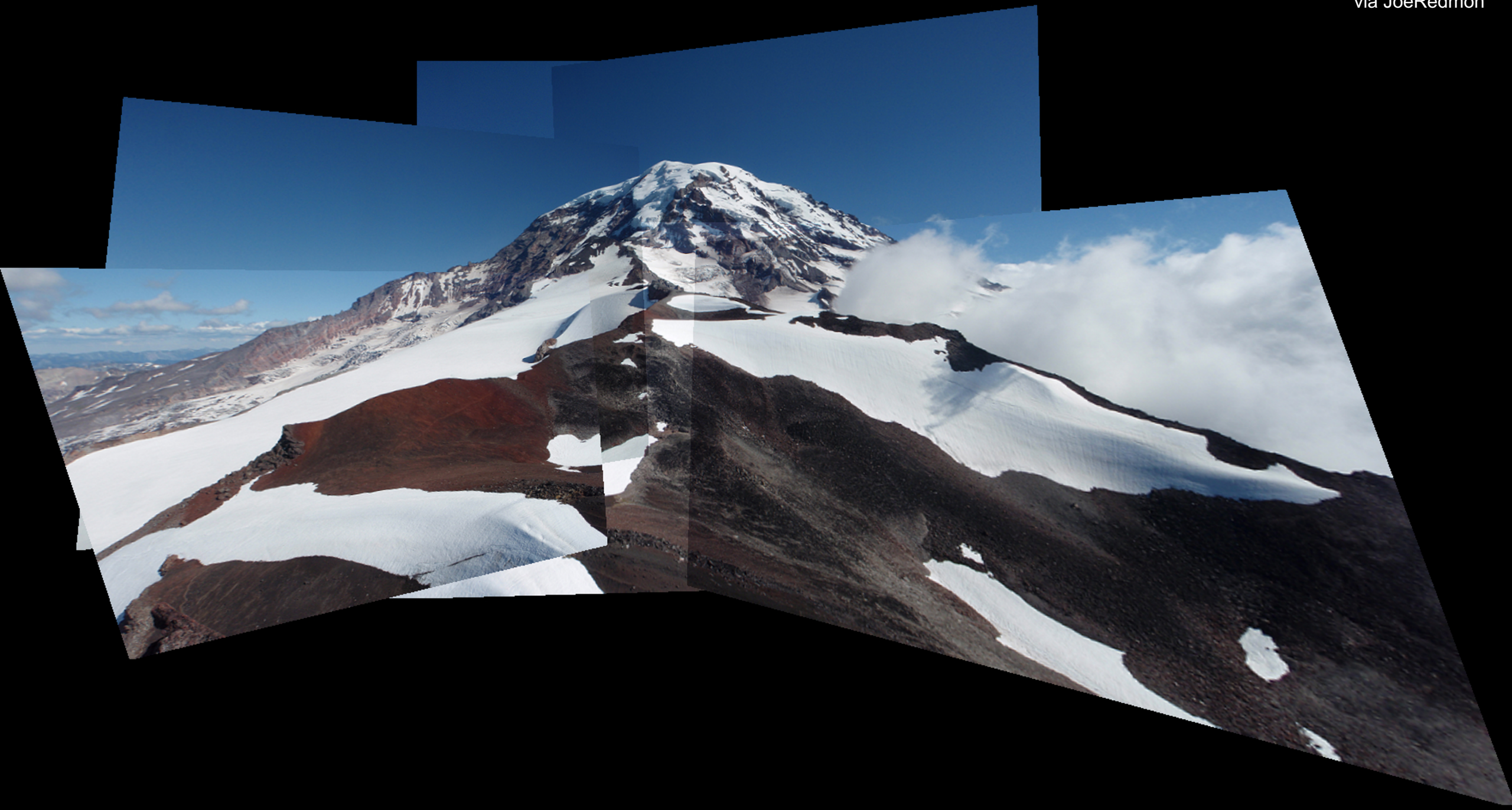
In practice:



via JoeRedmon

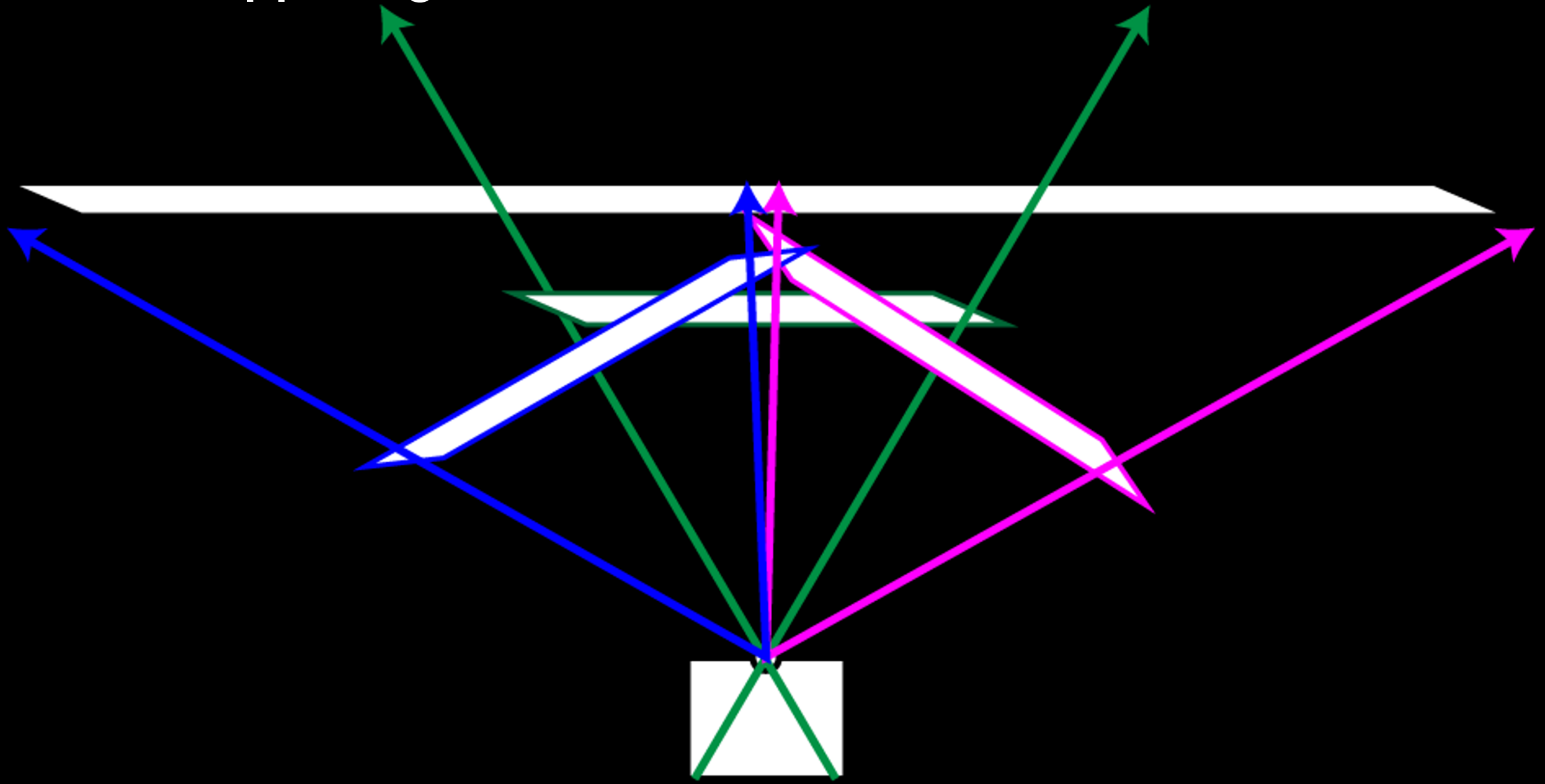

via JoeRedmon


via JoeRedmon


via JoeRedmon

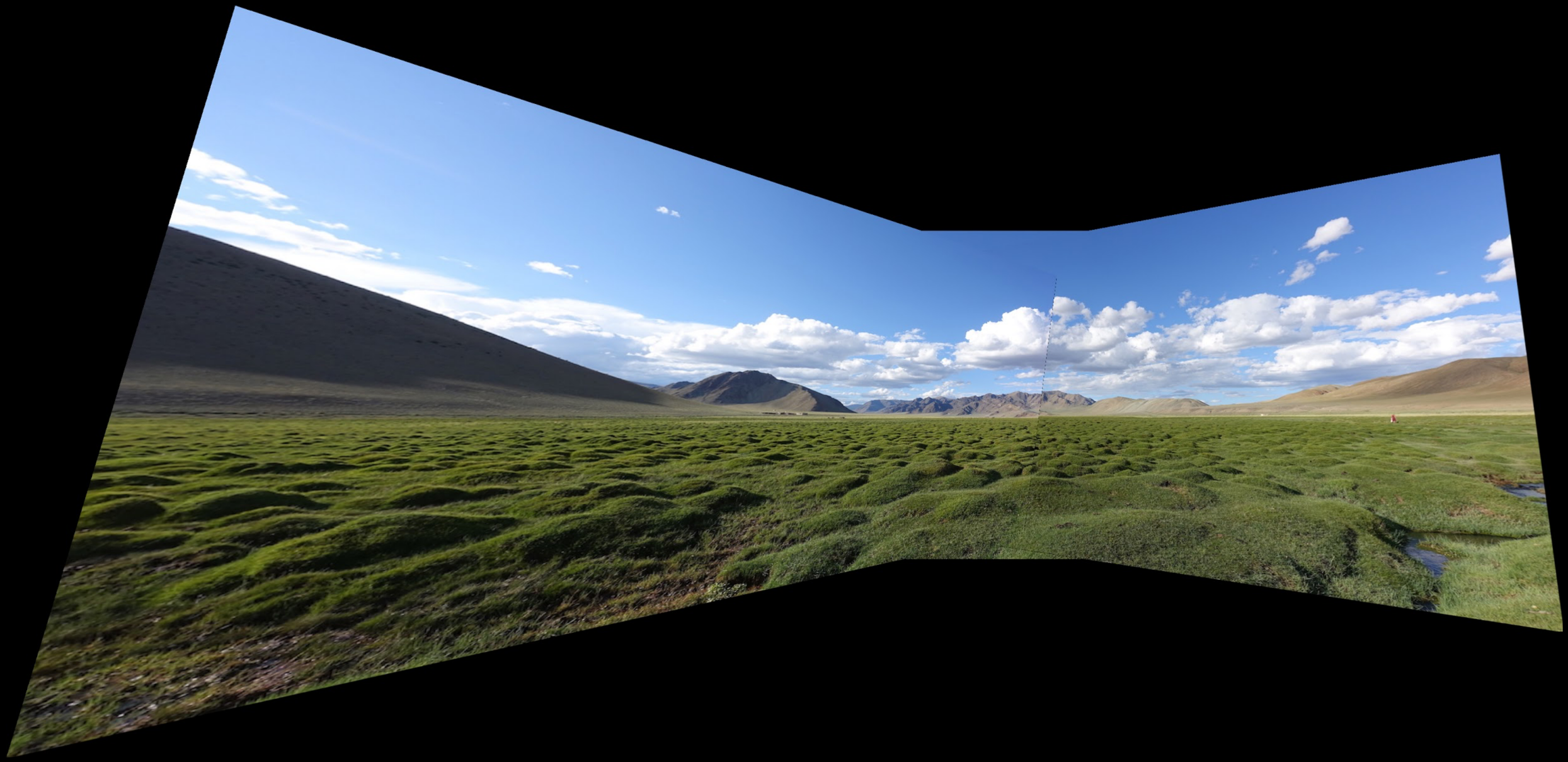
via JoeRedmon

What's happening?



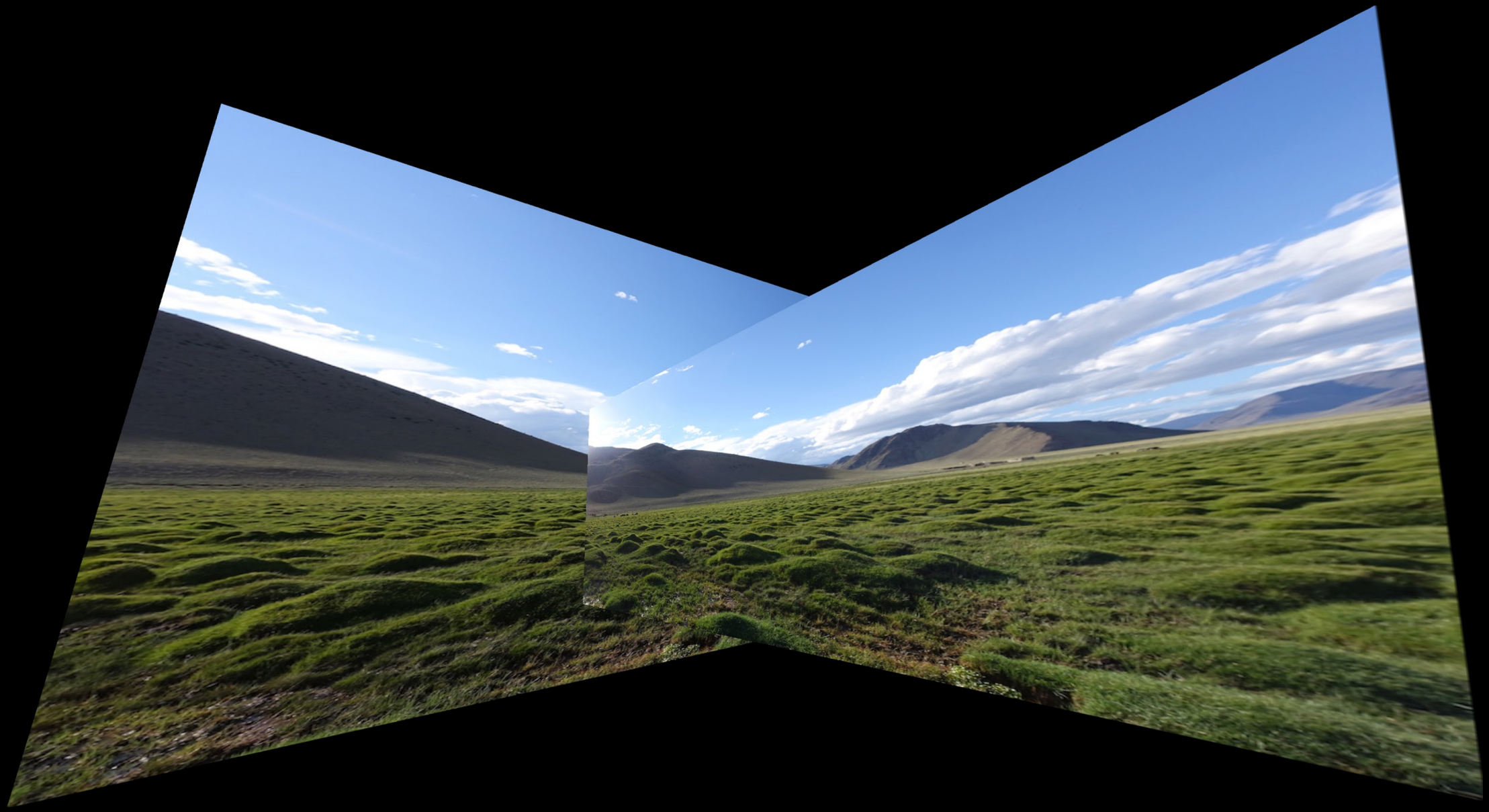
via JoeRedmon

Very bad for big panoramas!



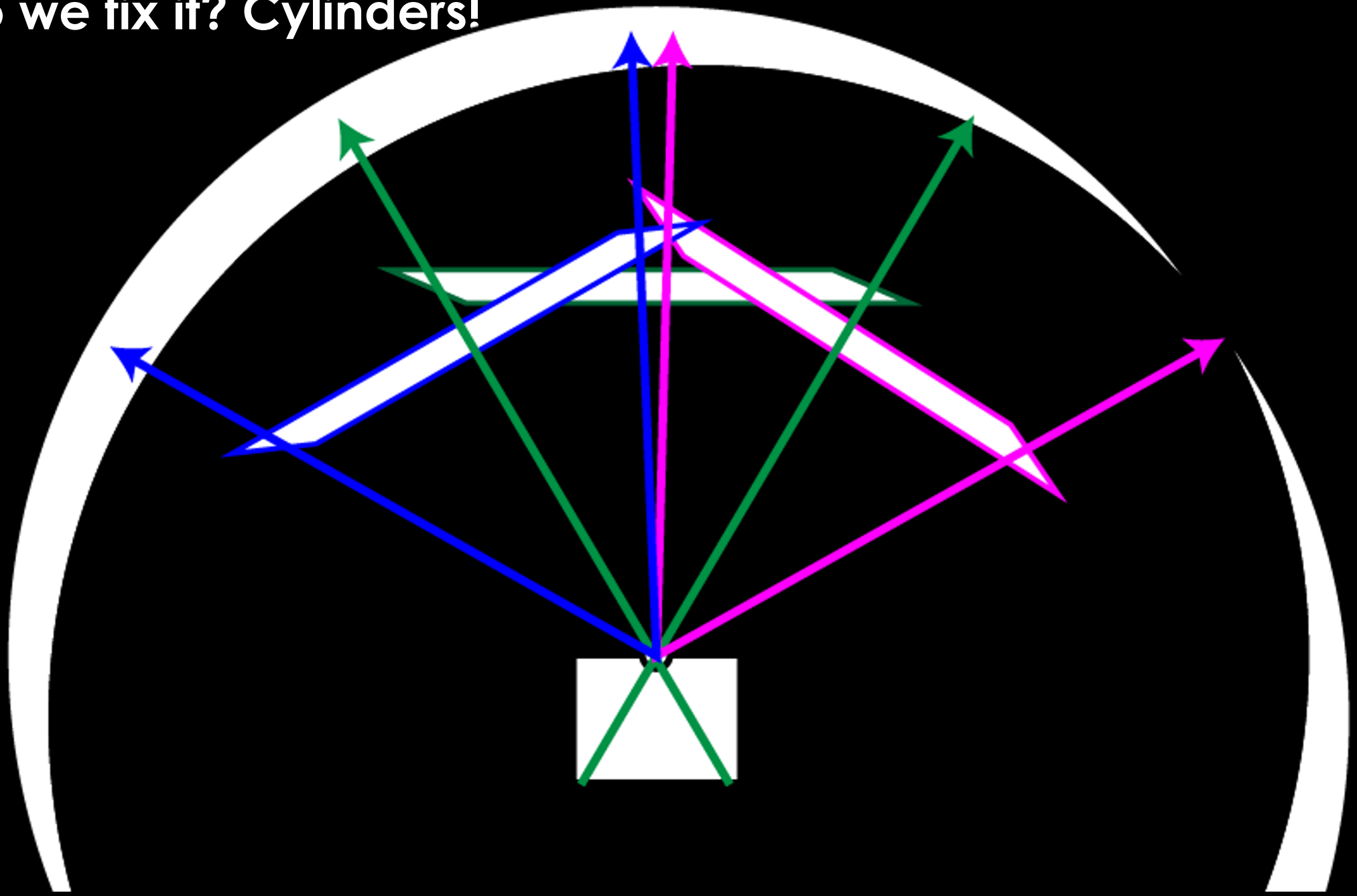
via JoeRedmon

Fails :-(



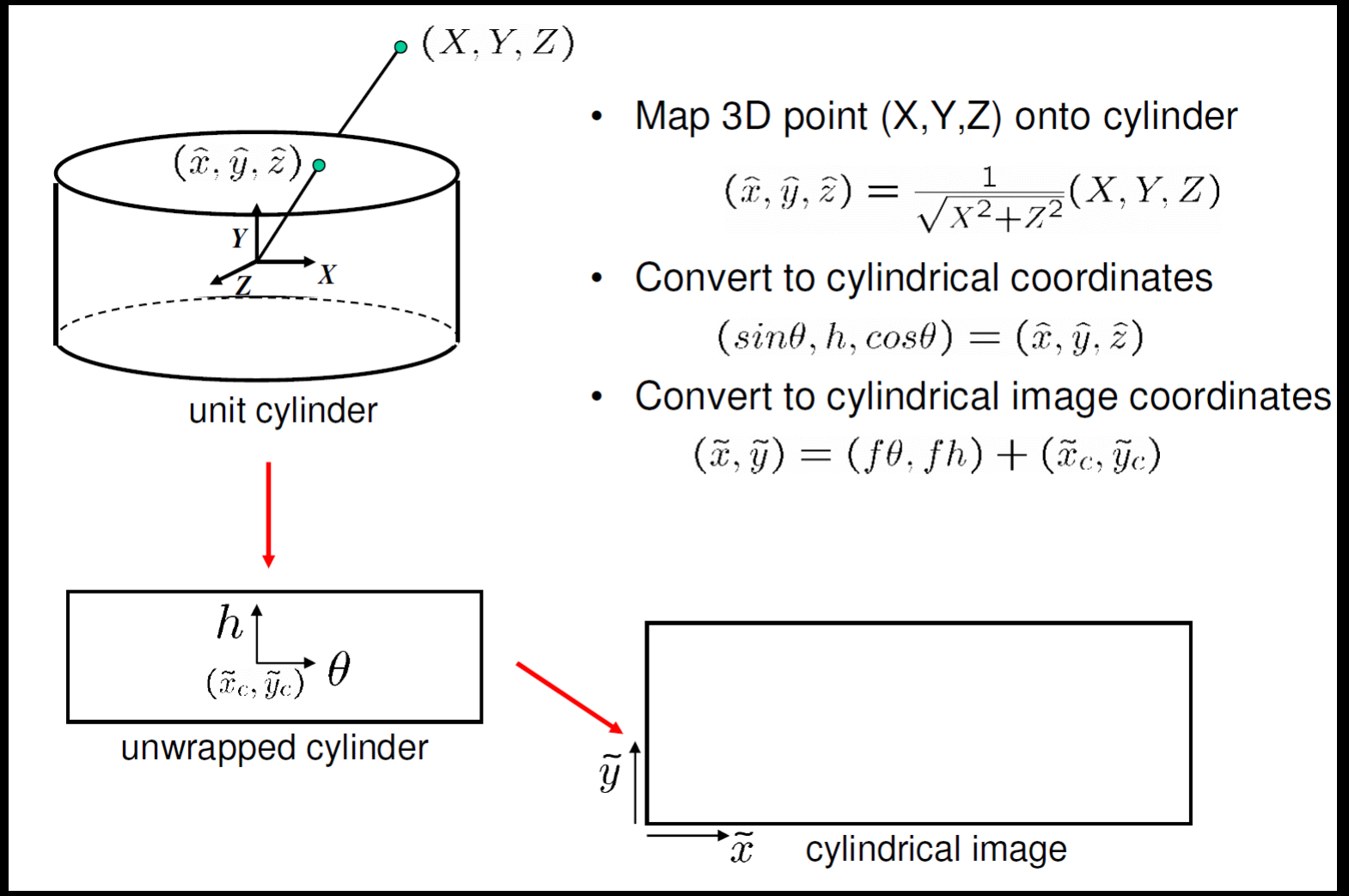
How do we fix it? Cylinders!

via JoeRedmon



Cylindrical Projection

via CMU



- Map 3D point (X,Y,Z) onto cylinder

$$(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X, Y, Z)$$

- Convert to cylindrical coordinates

$$(sin\theta, h, cos\theta) = (\hat{x}, \hat{y}, \hat{z})$$

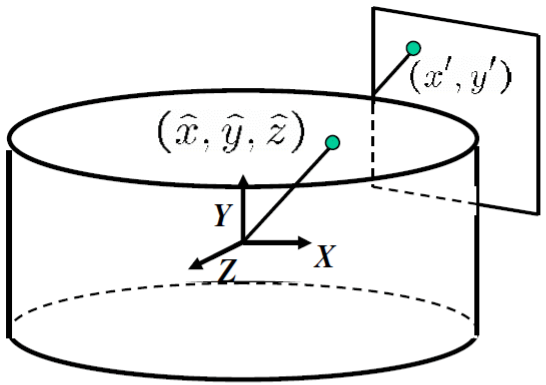
- Convert to cylindrical image coordinates

$$(\tilde{x}, \tilde{y}) = (f\theta, fh) + (\tilde{x}_c, \tilde{y}_c)$$

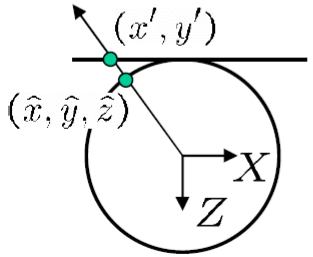
- Mapping planar image to Cylindrical Coordinates

Cylindrical Re-Projection

via CMU



side view



top-down view

- Apply camera projection matrix
  - w = image width, h = image height

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} -f & 0 & w/2 & 0 \\ 0 & -f & h/2 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ 1 \end{bmatrix}$$

- Convert to image coordinates
  - divide by third coordinate (w)

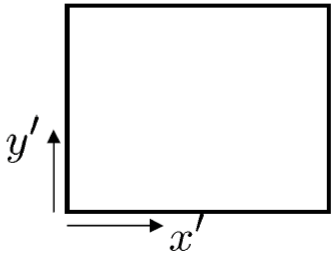
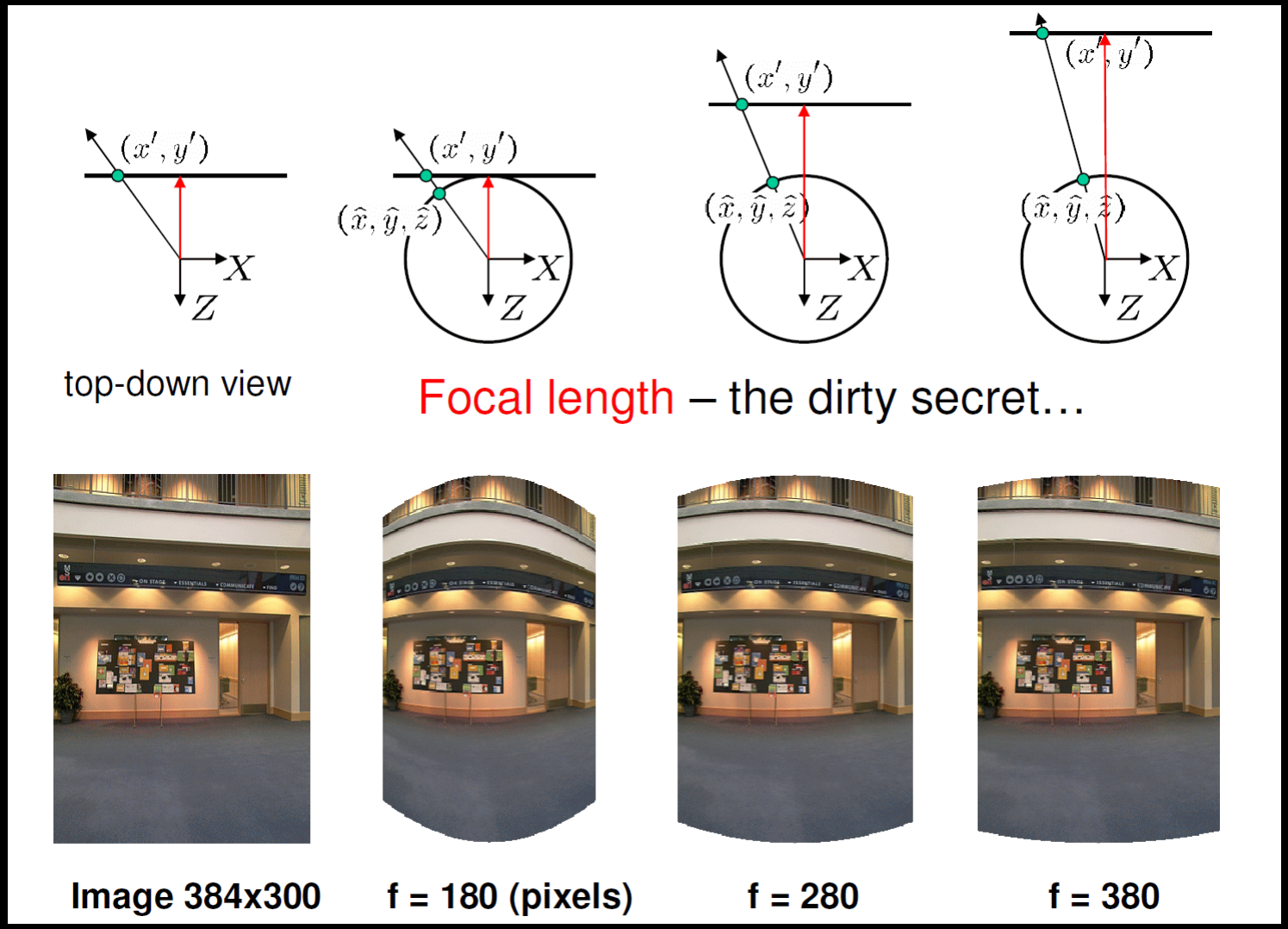


image coords

- Developing / Reprojecting a Cylinder onto a plane to create a Planar Viewable Panorama
- Try different focal lengths or use the focal length of the camera

Cylindrical Reprojection: Dependence on f

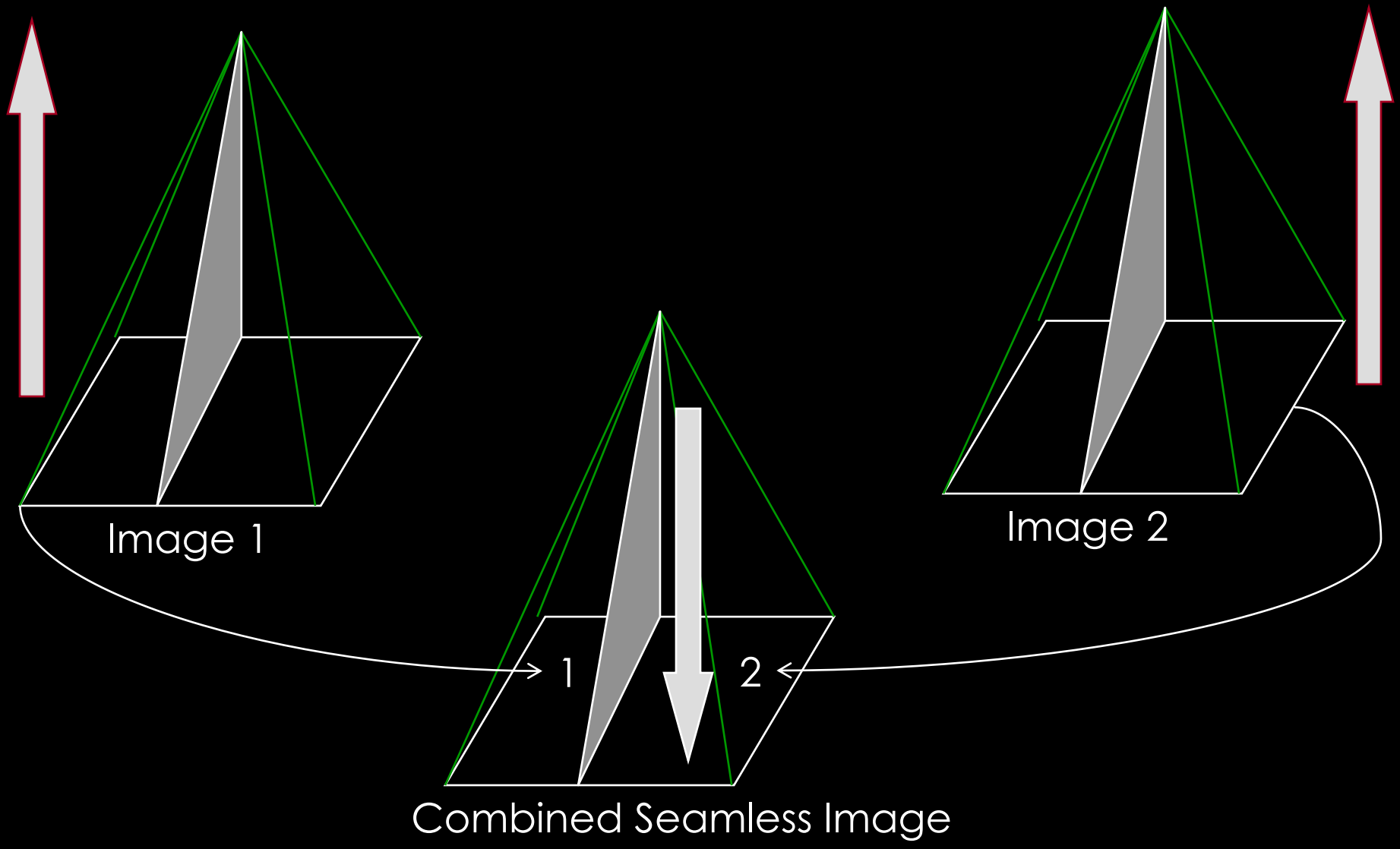


- Fill the panorama image to the max or use f

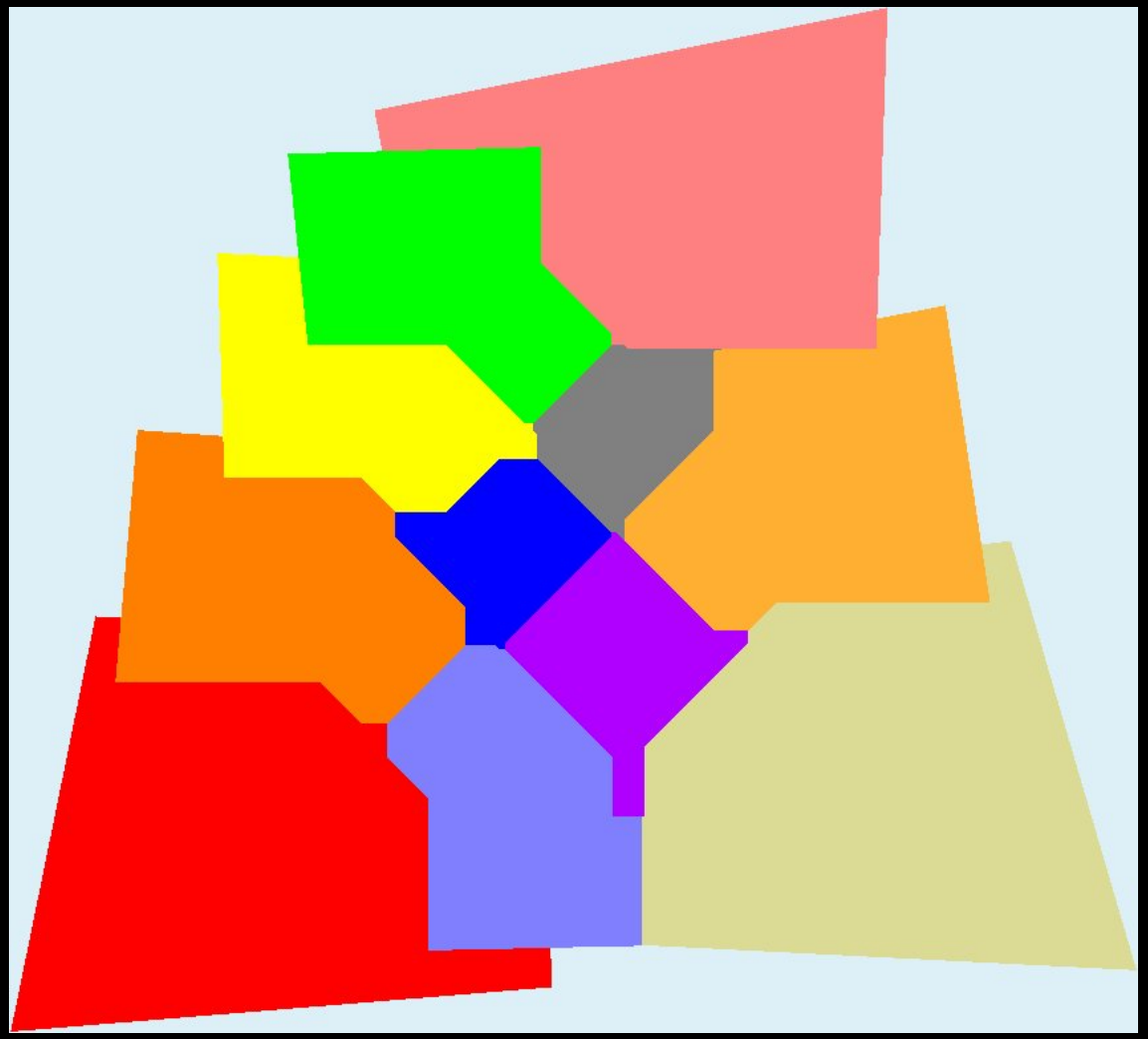
UNBLENDED MOSAIC



Image Merging withLaplacian Pyramids



VORONOI TESSELATIONS W/ L1 NORM



BLENDED MOSAIC



UNBLENDED MOSAIC



How does iPhone panoramic stitching work?

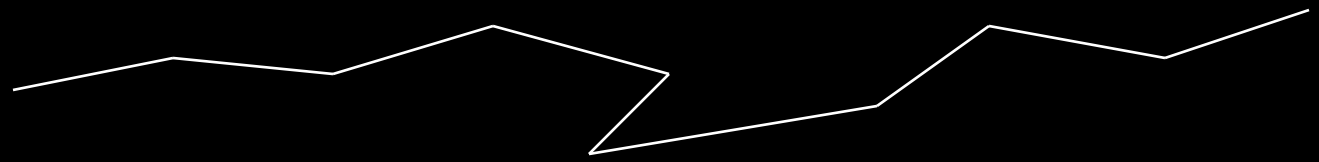
via DerekHoeim

- Capture images at 30 fps
- Stitch the central 1/8 of a selection of images
  - Select which images to stitch using the accelerometer and frame-to-frame matching
  - Faster and avoids radial distortion that often occurs towards corners of images
- Alignment
  - Initially, perform cross-correlation of small patches aided by accelerometer to find good regions for matching
  - Register by matching points (KLT tracking or RANSAC with FAST (similar to SIFT) points) or correlational matching
- Blending
  - Linear (or similar) blending, using a face detector to avoid blurring face regions and choose good face shots (not blinking, etc)

http://www.patentlyapple.com/patently-apple/2012/11/apples-cool-iphone-5-panorama-app-revealed-in-5-patents.html

1D vs. 2D SCANNING

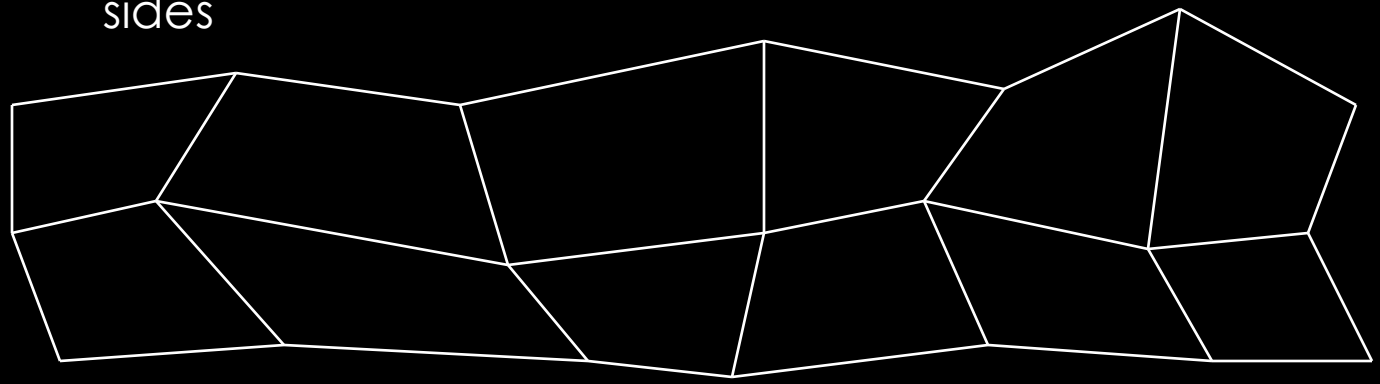
- 1D : The topology of frames is a ribbon or a string.
  Frames overlap only with their temporal neighbors.



(A 300x332 mosaic captured by mosaicing a 1D sequence of 6 frames)



- 2D : The topology of frames is a 2D graph
  Frames overlap with neighbors on many sides



1D vs. 2D SCANNING



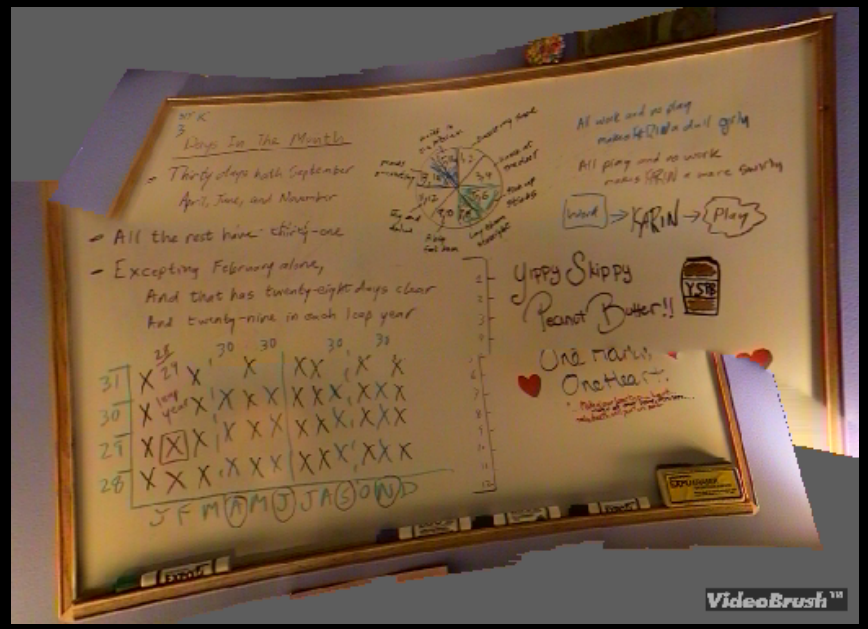
The 1D scan scaled by 2 to 600x692



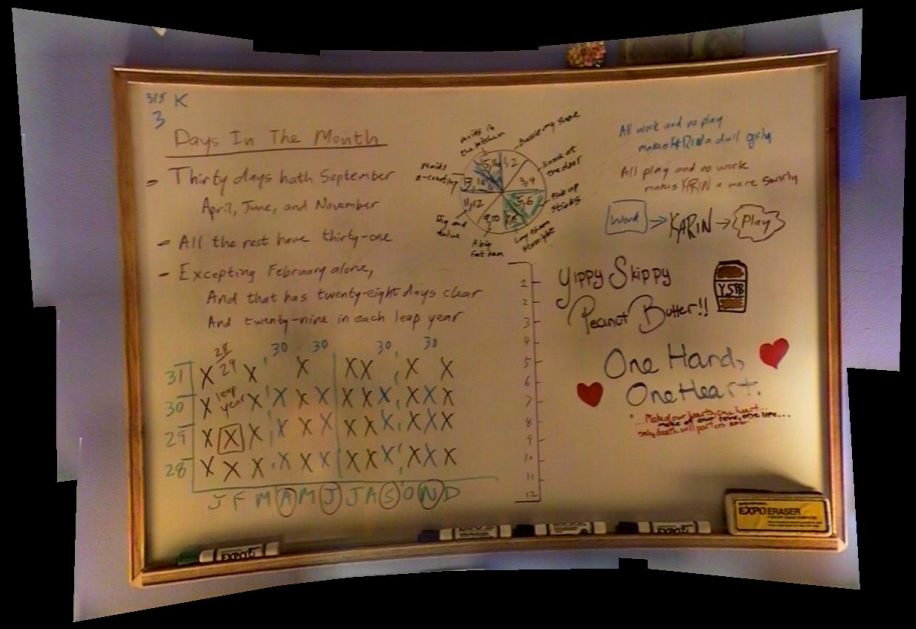
A 2D scanned mosaic of size 600x692

Frame-to-Mosaic vs. Local-to-Global Alignment

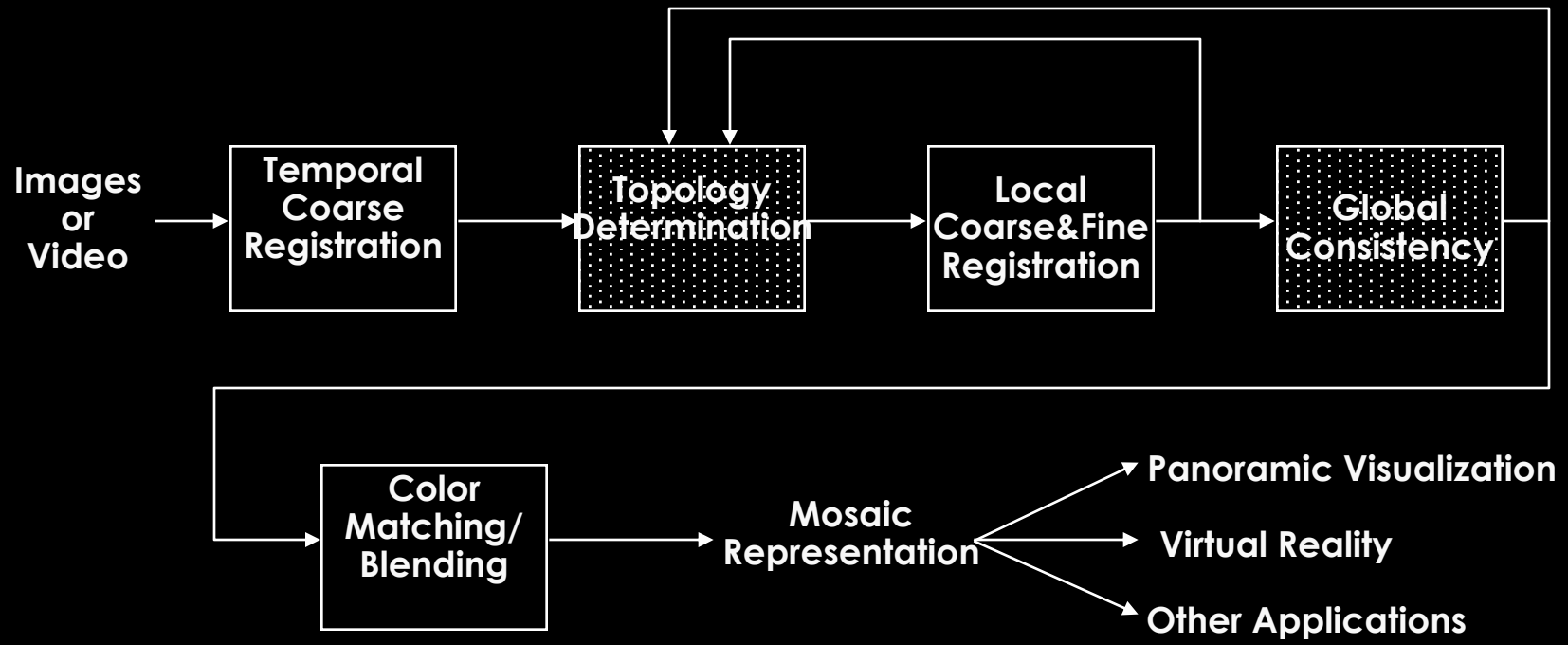
- Uses limited 2D spatial context
- Causal commitment to parameters cannot be corrected
- Demands large overlap between frames



- Uses all the available frame-to-frame constraints
- Global solution is optimal subject to local frame-to-frame constraints
- Works even with small overlap between frames



LOCAL TO GLOBAL MOSAIC ALGORITHM



SPECIFIC EXAMPLES : 2. SPHERICAL MOSAICS

- Frame to mosaic transformation model: $\mathbf{u} \approx \mathbf{F}\mathbf{R}_\mathbf{i}^{T}\mathbf{X}$

- Local Registration
  - Coarse 2D translation & fine 2D projective alignment
- Parameter Initialization
  - Compute **F** and **R**'s from the 2D projective matrices

- Topology :
  - Initial graph topology computed with the 2D R & T estimates on a plane
  - Subsequently the topology defined on a sphere
  - Iterative refinement using arcs based on alignment with **F** and **R**'s

- Global Alignment

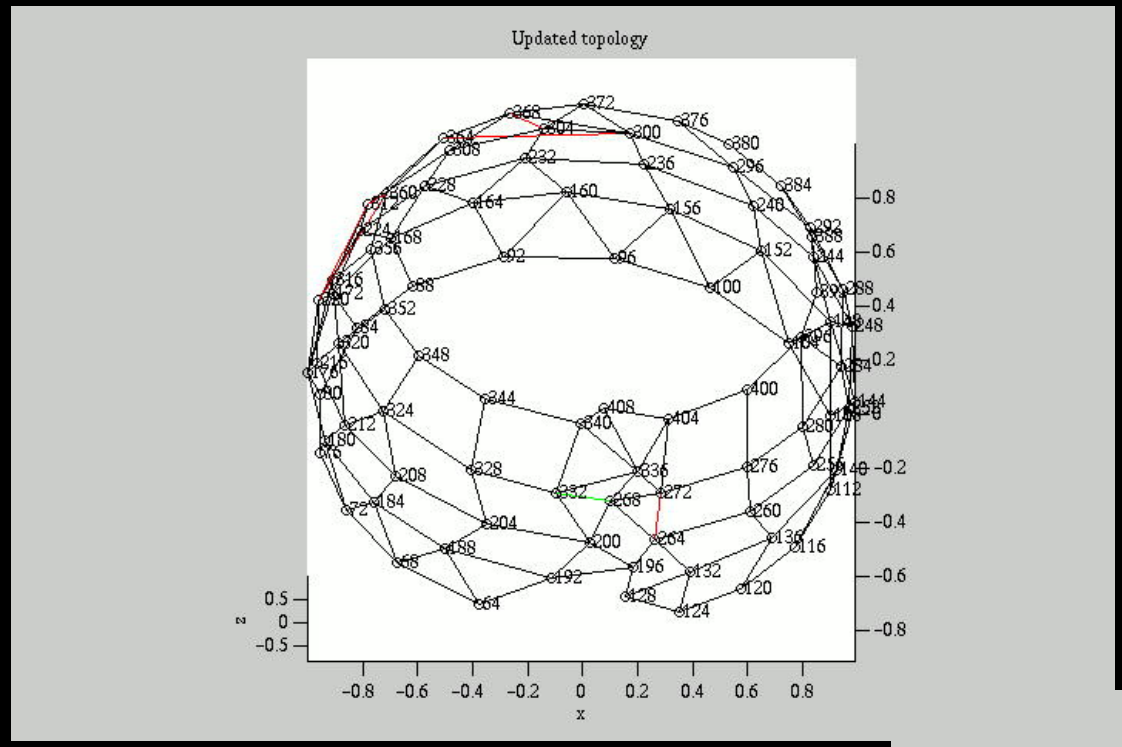
$$E_{ij} = \sum_{k} |\mathbf{R_i}\mathbf{F^{-1}}\mathbf{u_{ik}} - \mathbf{R_j}\mathbf{F^{-1}}\mathbf{u_{jk}}|^2$$

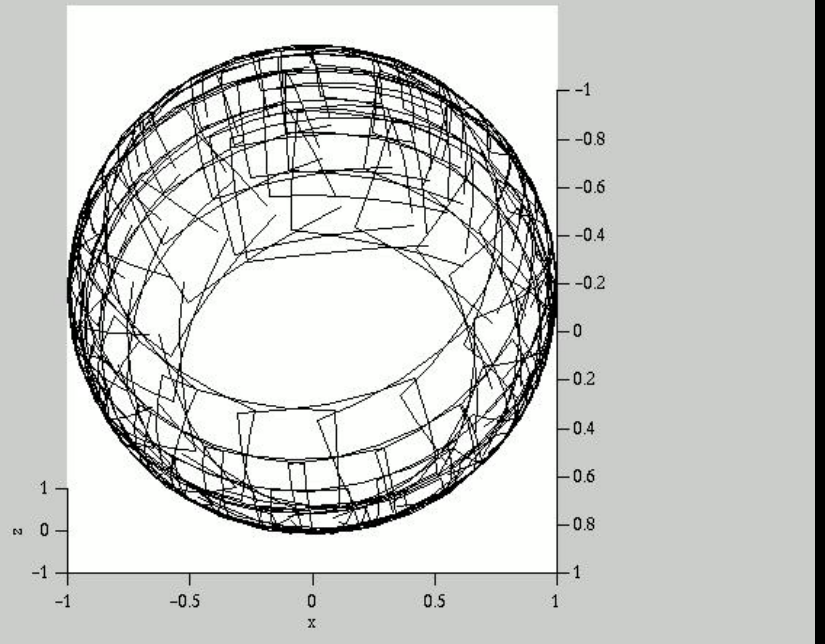
SPHERICAL MOSAICS



Video
Captures almost the complete sphere
with 380 frames

SPHERICAL TOPOLOGY EVOLUTION





SPHERICAL MOSAIC

Sarnoff Library

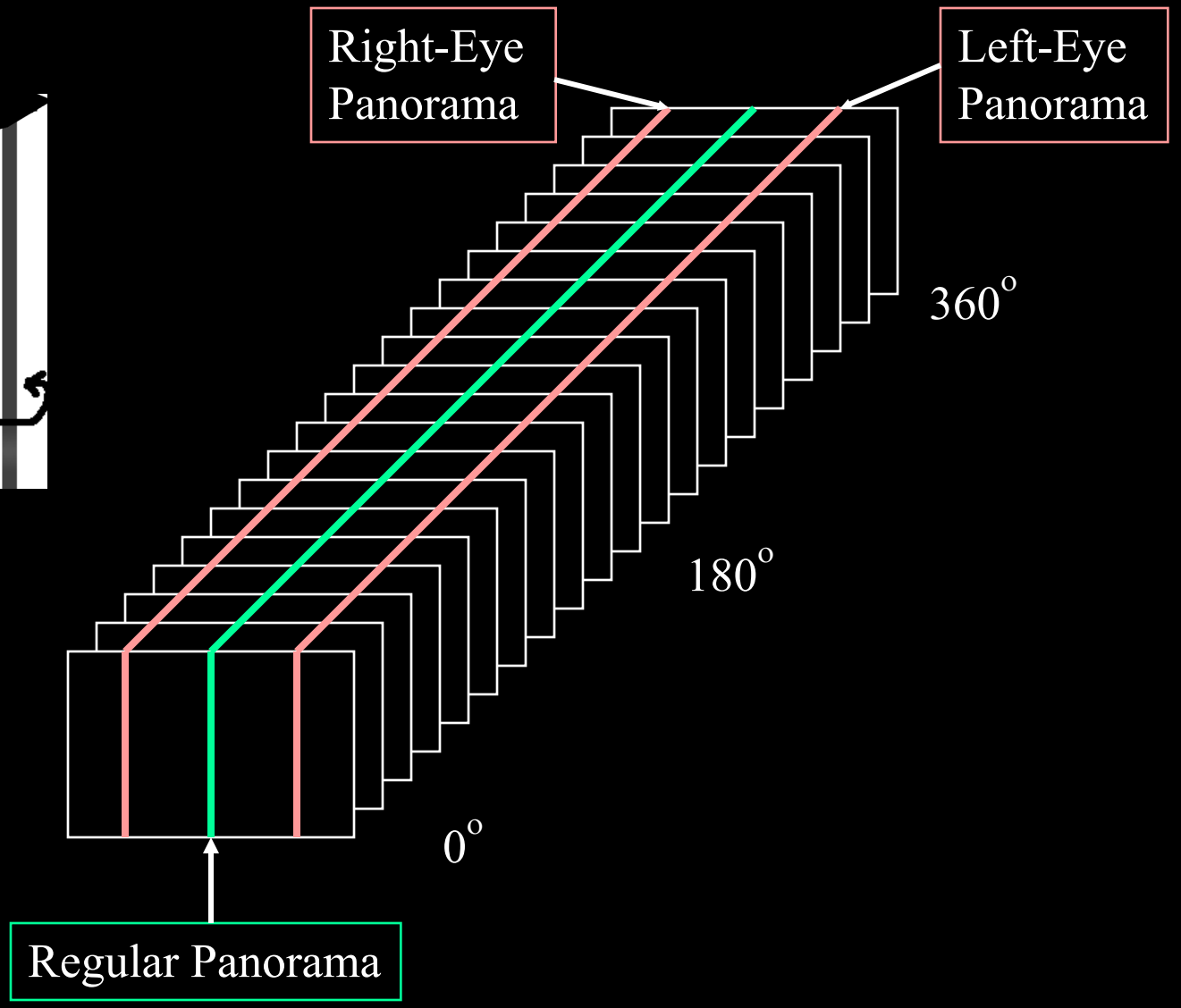


Mercator projection of spherical mosaic



MultiView Panoramas





Stereo Panorama from Video



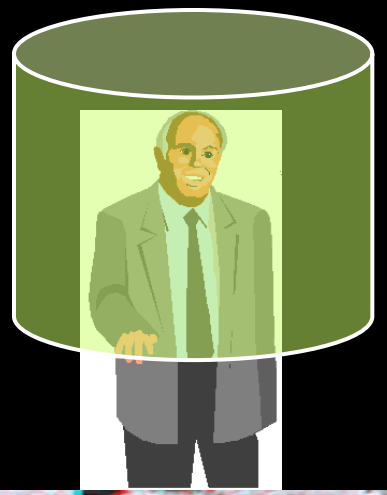
Stereo viewing with Red/Blue Glasses

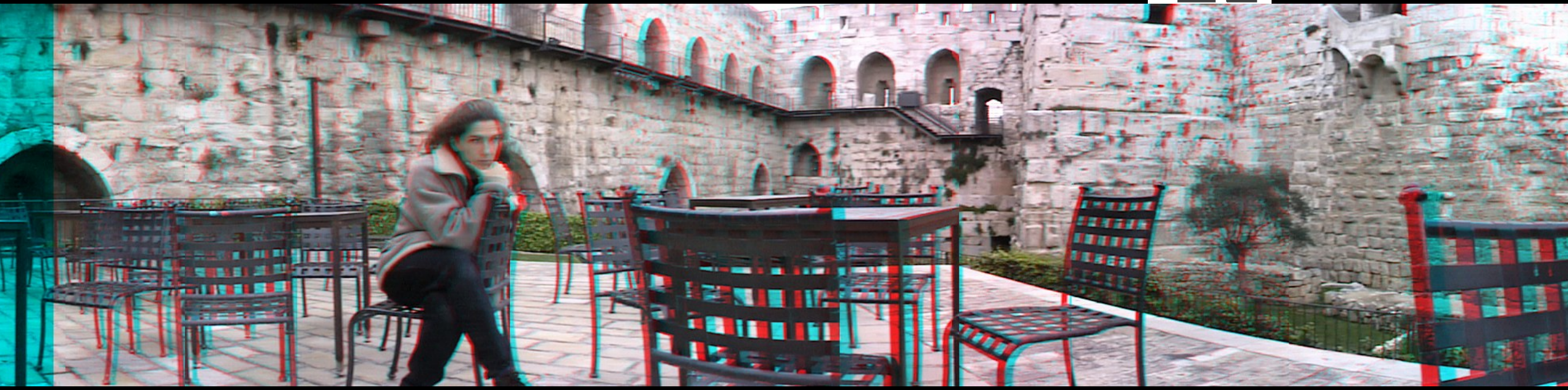


Viewing Panoramic Stereo

Printed Cylindrical Surfaces

- Print panorama on a cylinder
- No computation needed!!!





DYNAMIC MOSAICS

Original Video



Video Stream with
deleted moving object



Dynamic Mosaic Video



SYNOPISIS MOSAICS

