# What have we leaned so far?

- Camera structure
- Eye structure

Project 1: High Dynamic Range Imaging

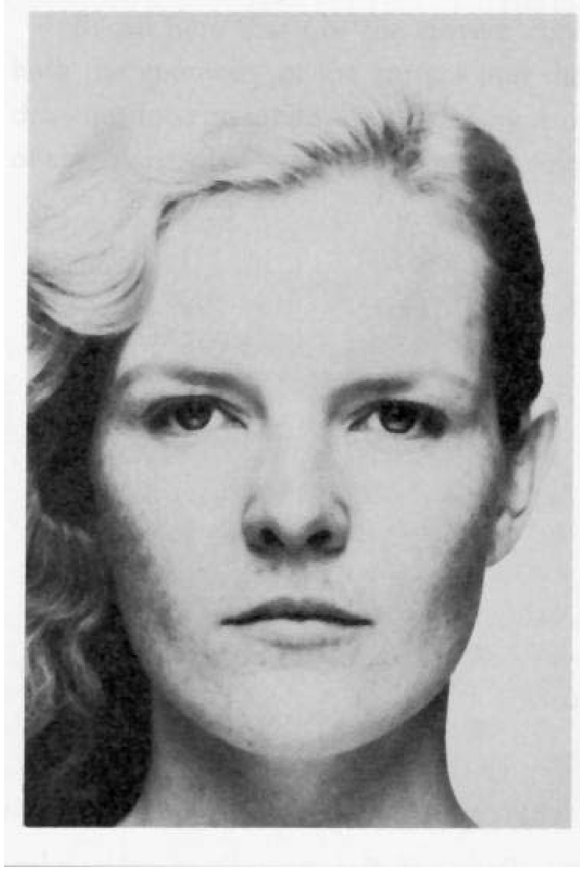
# What have we learned so far?

- Image Filtering
- Image Warping
- Camera Projection Model

Project 2: Panoramic Image Stitching

# What have we learned so far?

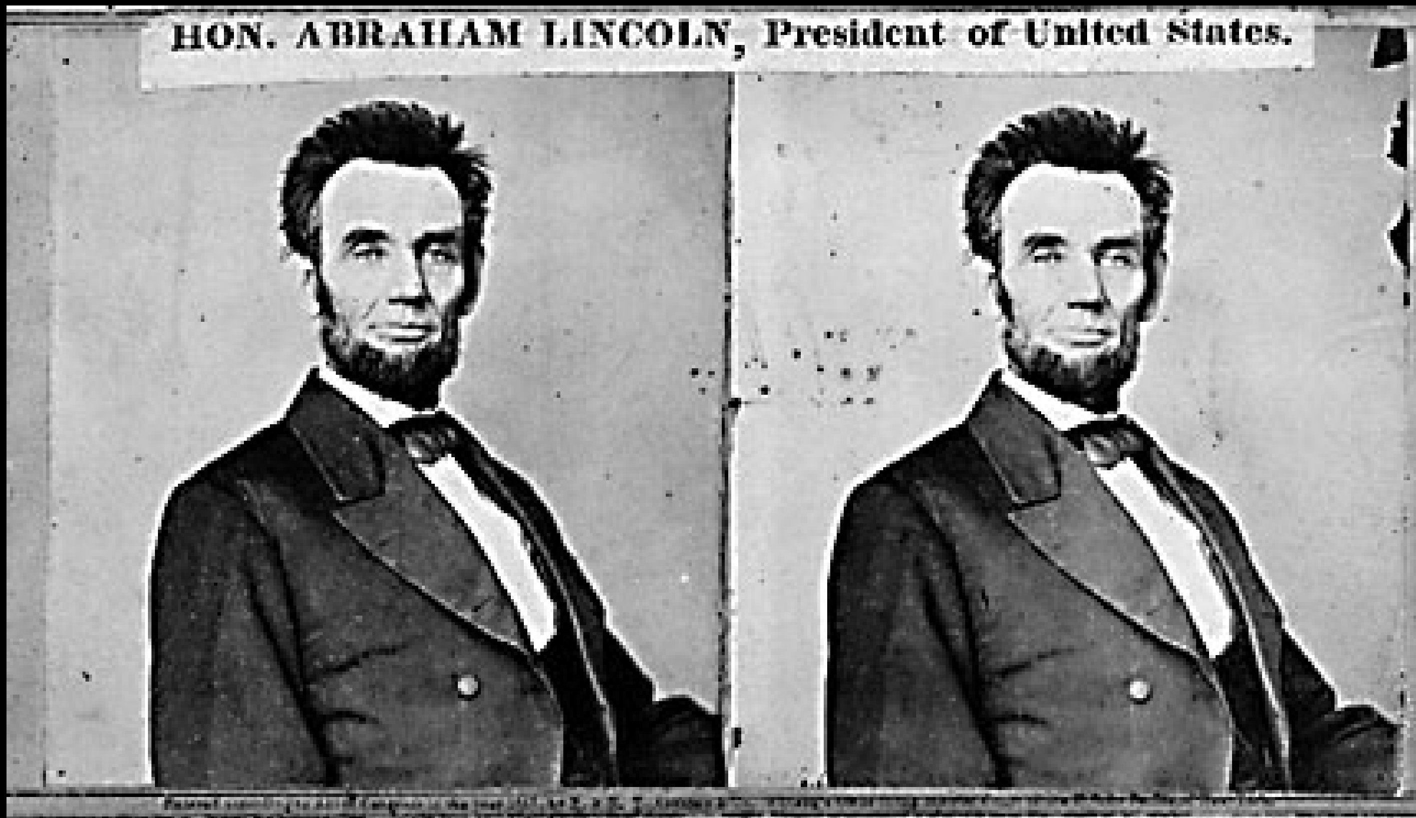
- Projective Geometry
- Single View Modeling
- Shading Model



Project 3: Photometric Stereo

# Today

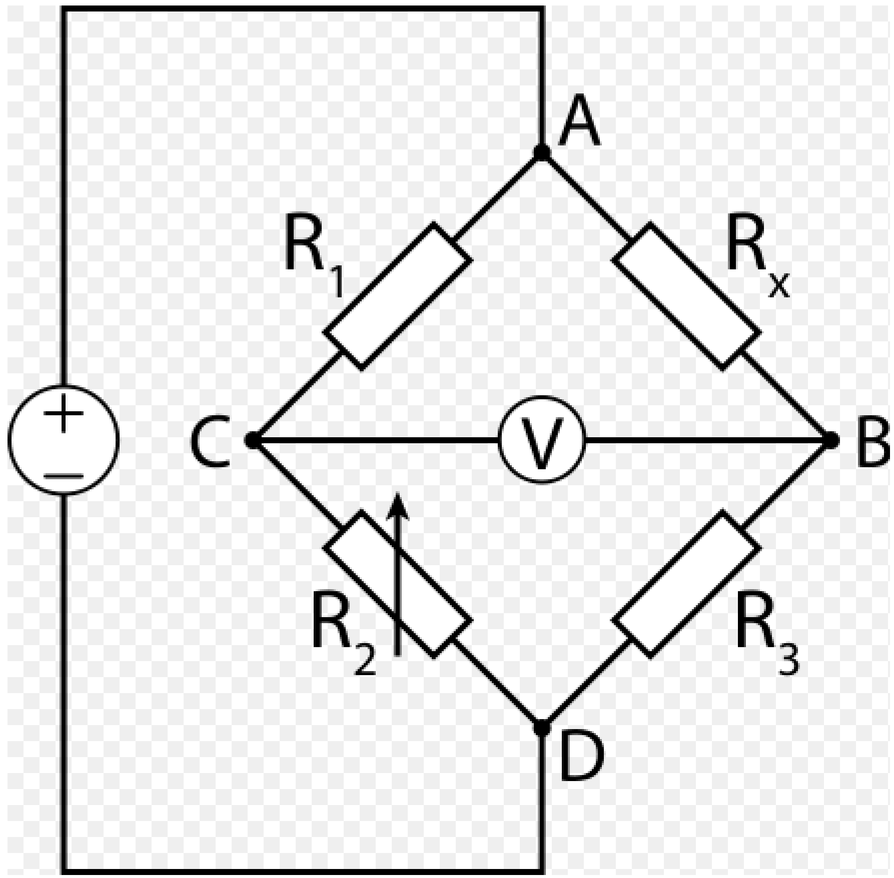
- 3D modeling from two images – Stereo


HON. ABRAHAM LINCOLN, President of United States.


**Public Library, Stereoscopic Looking Room, Chicago, by Phillips, 1923**

Inventor: Sir Charles Wheatstone, 1802 - 1875

http://en.wikipedia.org/wiki/Sir_Charles_Wheatstone




Inventor: Sir Charles Wheatstone, 1802 - 1875

http://en.wikipedia.org/wiki/Wheatstone_bridge

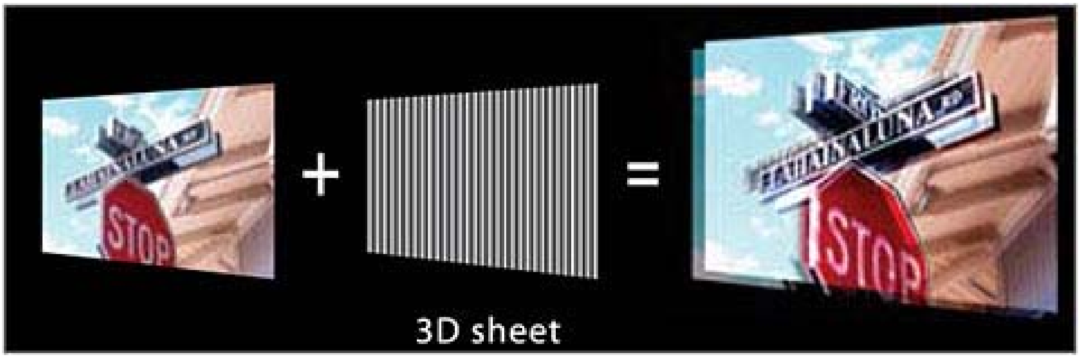
# Stereograms online

- UCR stereographs
  - http://www.cmp.ucr.edu/site/exhibitions/stereo/
- The Art of Stereo Photography
  - http://www.photostuff.co.uk/stereo.htm
- History of Stereo Photography
  - http://www.rpi.edu/~ruiz/stereo_history/text/historystereog.html
- Double Exposure
  - http://home.centurytel.net/s3dcor/index.html
- Stereo Photography
  - http://www.shortcourses.com/book01/chapter09.htm
- 3D Photography links
  - http://www.studyweb.com/links/5243.html
- National Stereoscopic Association
  - http://204.248.144.203/3dLibrary/welcome.html
- Books on Stereo Photography
  - http://userwww.sfsu.edu/~hl/3d.biblio.html

A free pair of red-blue stereo glasses can be ordered from Rainbow Symphony Inc

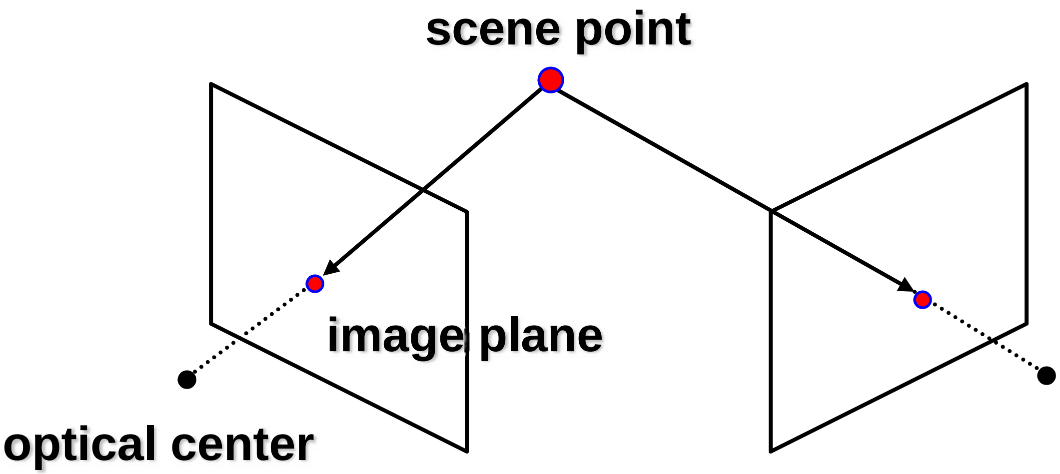
- http://www.rainbowsymphony.com/freestuff.html

# FUJIFILM, September 23, 2008

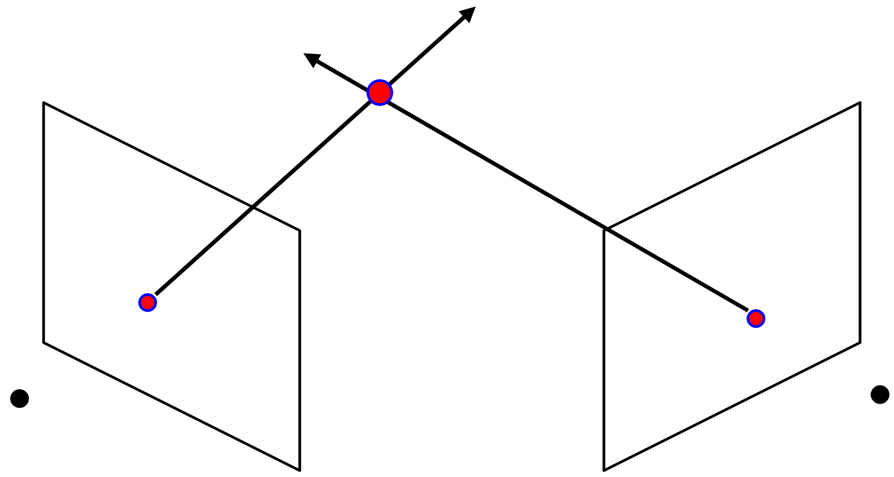




Fuji 3D printing

# Stereo



# Stereo

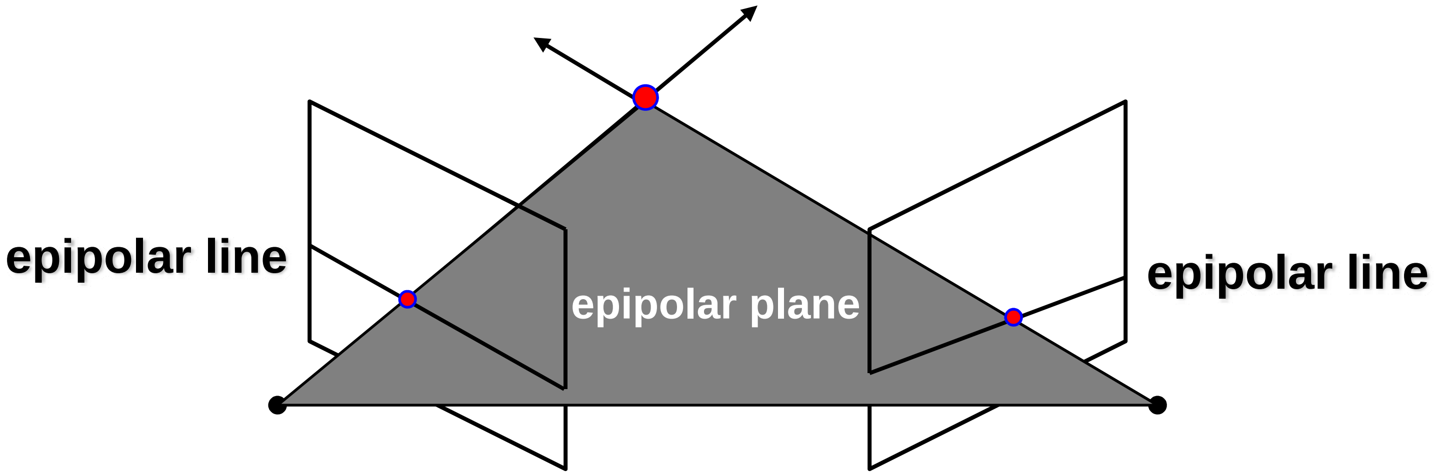


Basic Principle:  Triangulation

- Gives reconstruction as intersection of two rays
- Requires
  - calibration
  - ***point correspondence***

# Stereo correspondence

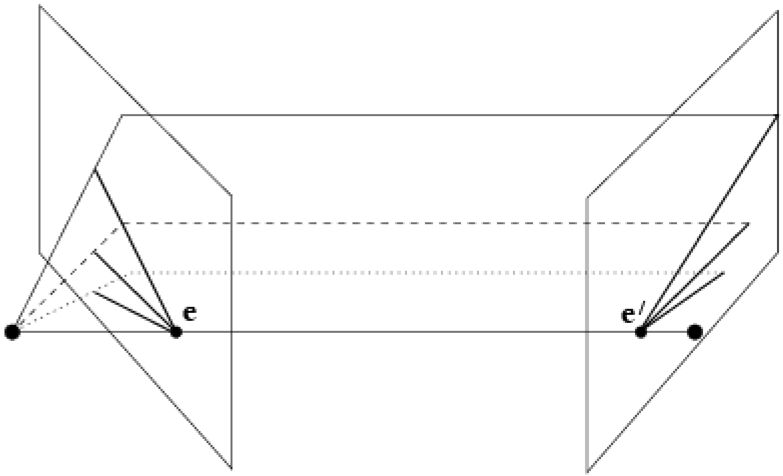
- Determine Pixel Correspondence
  - Pairs of points that correspond to same scene point

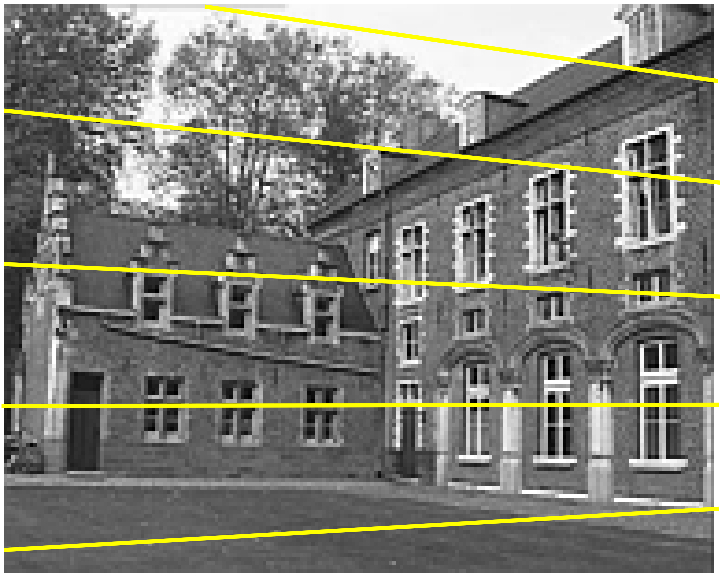


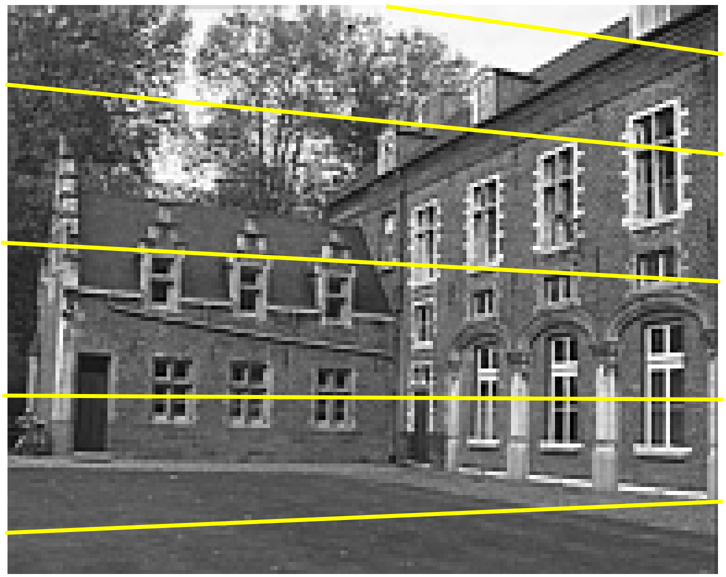

Epipolar Constraint

- Reduces correspondence problem to 1D search along *conjugate epipolar lines*
- Java demo: http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html

# Epipolar Line Example

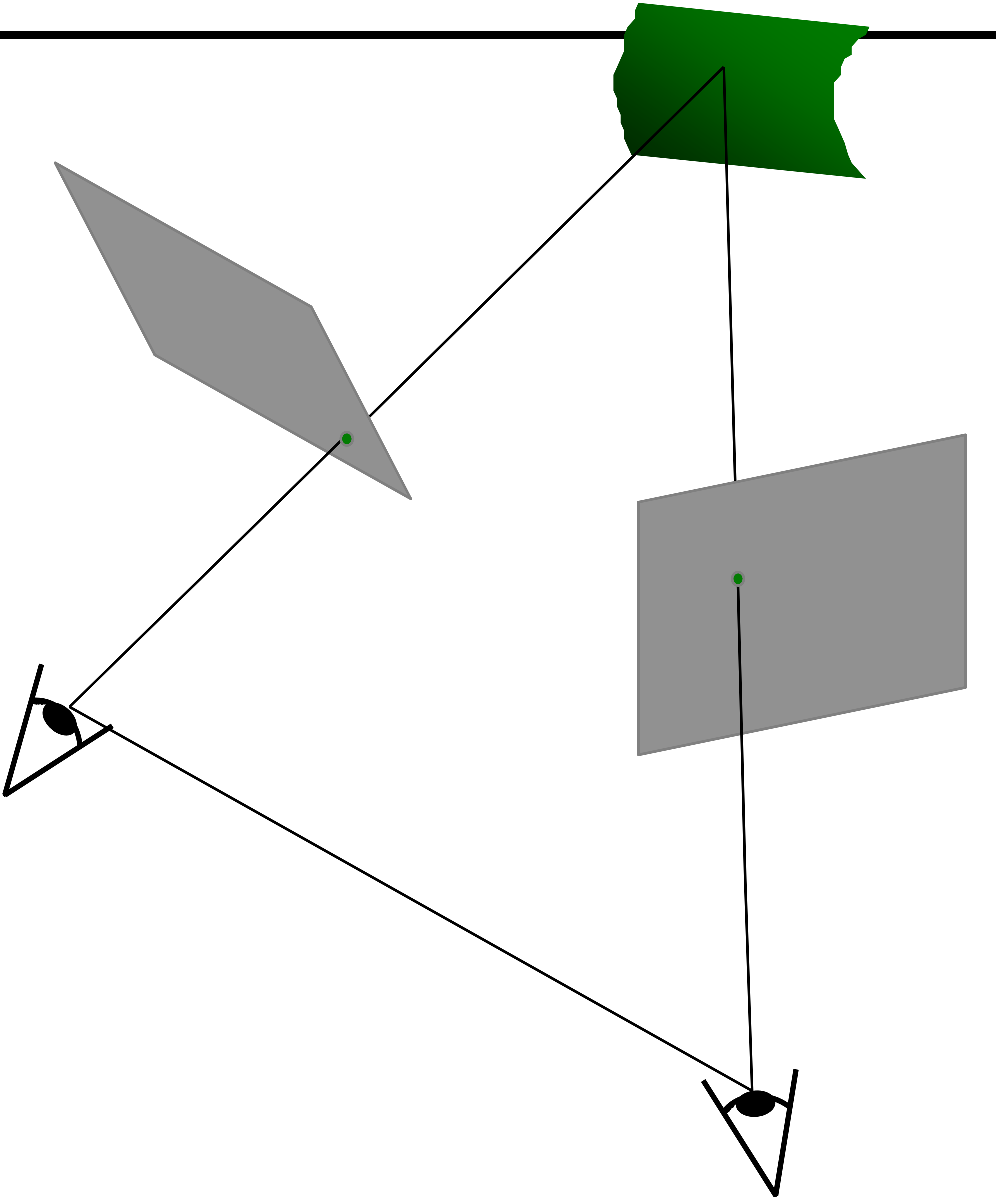


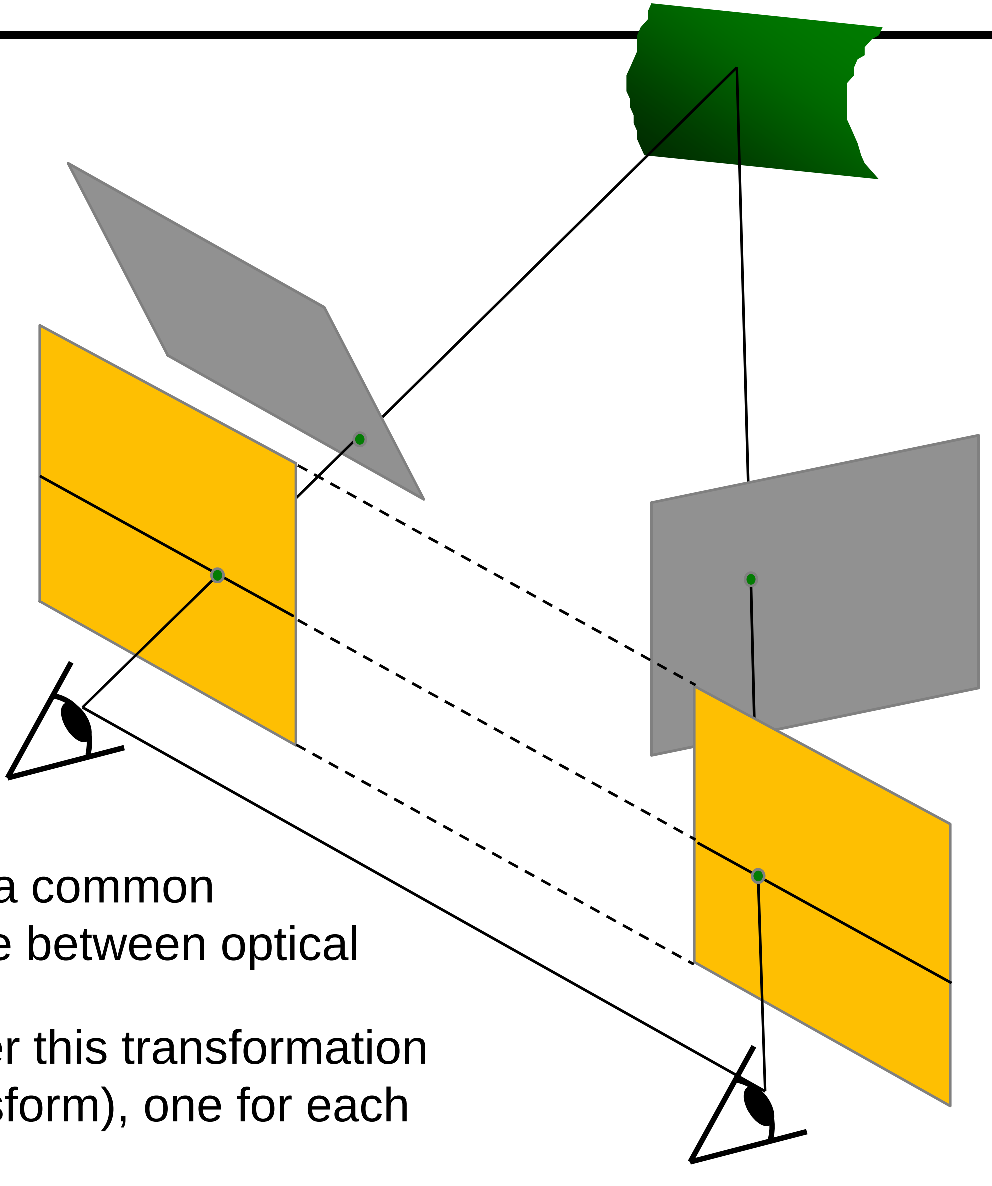


courtesy of Marc Pollefeys

# Stereo image rectification



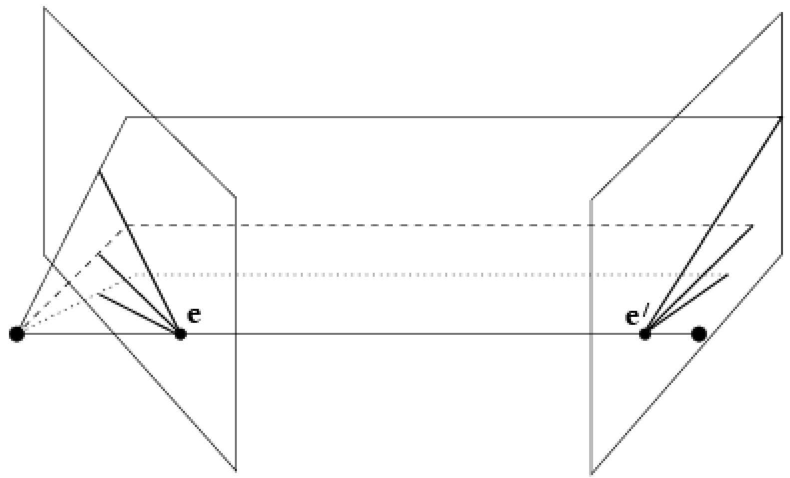
# Stereo image rectification

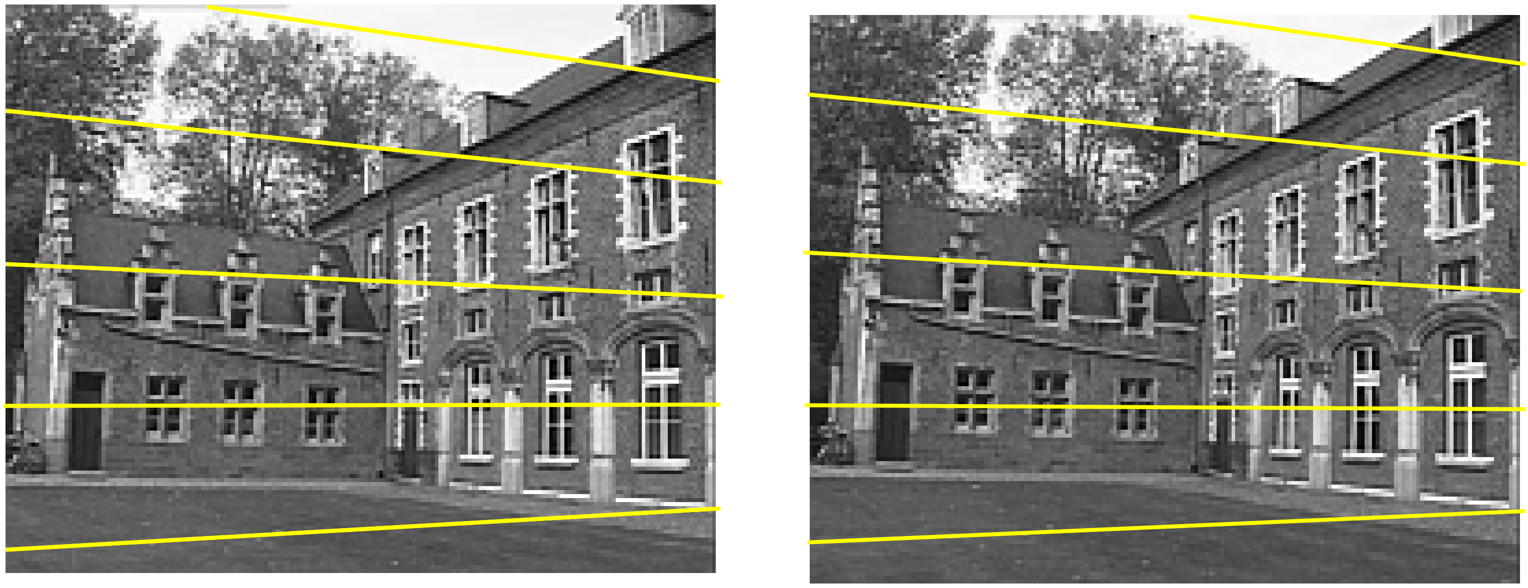


- reproject image planes onto a common
-  plane parallel to the line between optical centers
- pixel motion is horizontal after this transformation
- two homographies (3x3 transform), one for each input image reprojection

➢ C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. IEEE Conf. Computer Vision and Pattern Recognition, 1999.

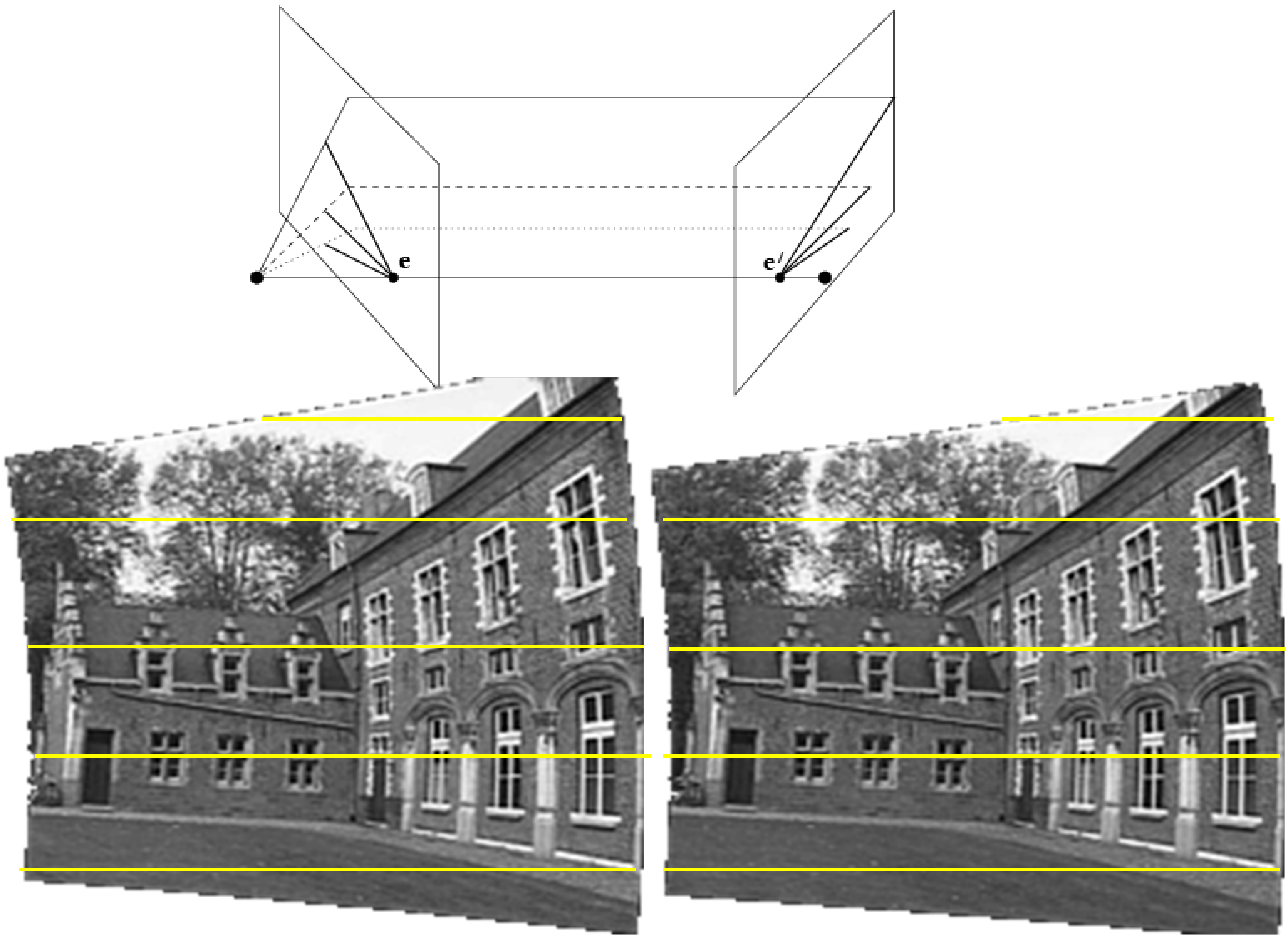
# Epipolar Line Example



courtesy of Marc Pollefeys

# Epipolar Line Example

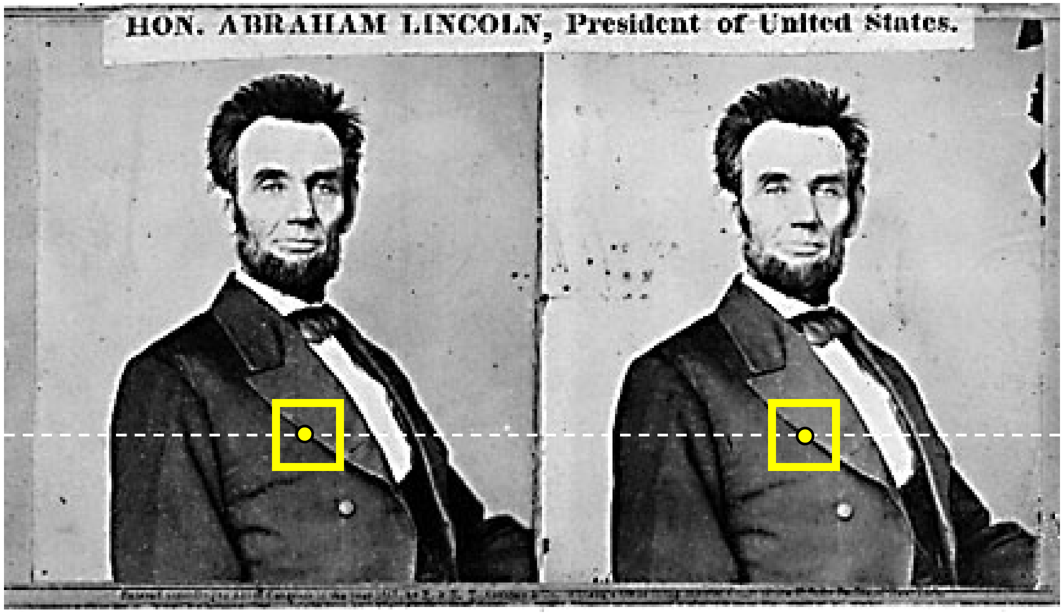



courtesy of Marc Pollefeys

# Stereo matching algorithms

- Match Pixels in Conjugate Epipolar Lines
  - Assume brightness constancy
  - This is a tough problem
  - Numerous approaches
    - A good survey and evaluation: http://www.middlebury.edu/stereo/

# Basic stereo algorithm




For each epipolar line

For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match *windows*

# Basic stereo algorithm

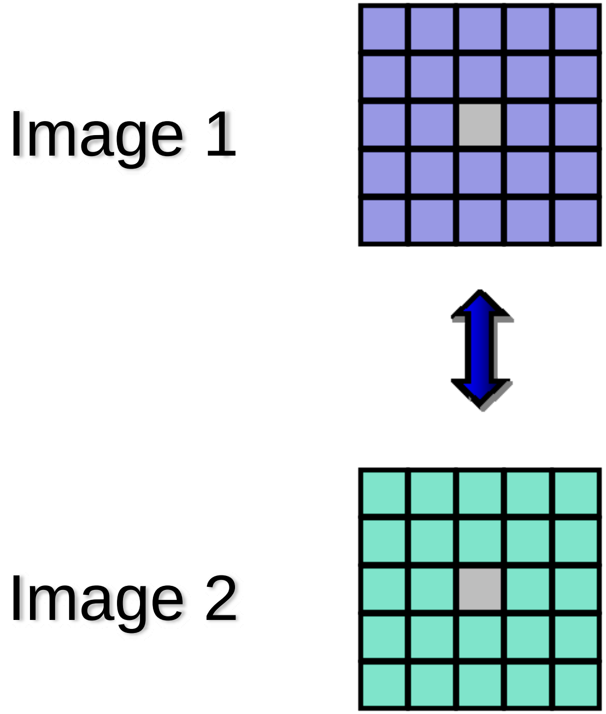
- For each pixel
  - For each disparity
    - – For each pixel in window
      - » Compute difference
  - Find disparity with minimum SSD

# Reverse order of loops

- For each disparity
  - For each pixel
    - For each pixel in window
      - Compute difference
- Find disparity with minimum SSD at each pixel

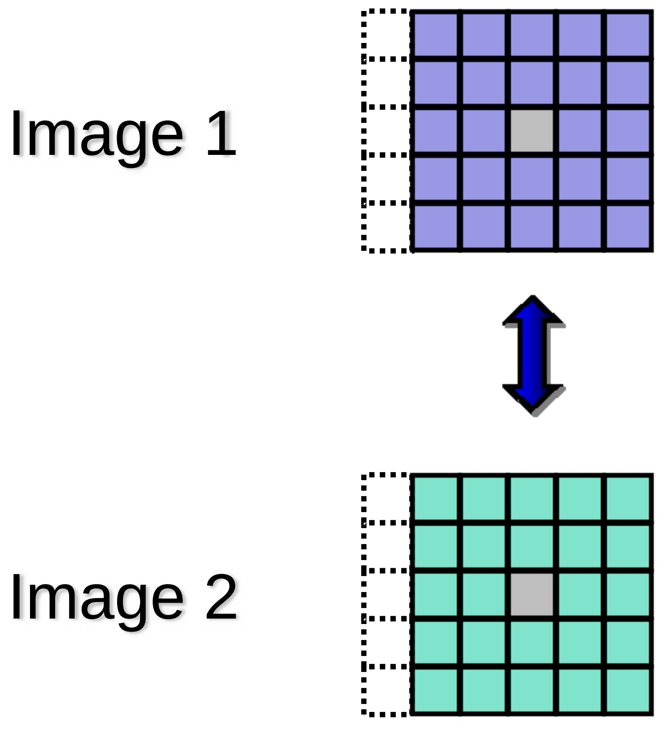
# Incremental computation

- Given SSD of a window, at some disparity



# Incremental computation

- Want: SSD at next location



# Incremental computation

- Subtract contributions from leftmost column, add contributions from rightmost column

Image 1

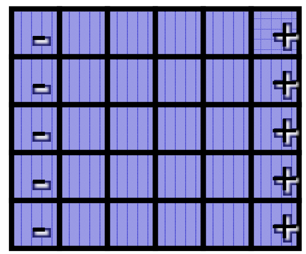
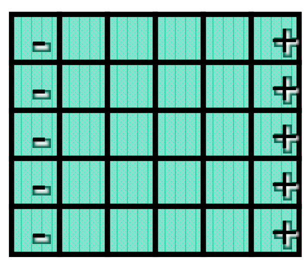


Image 2

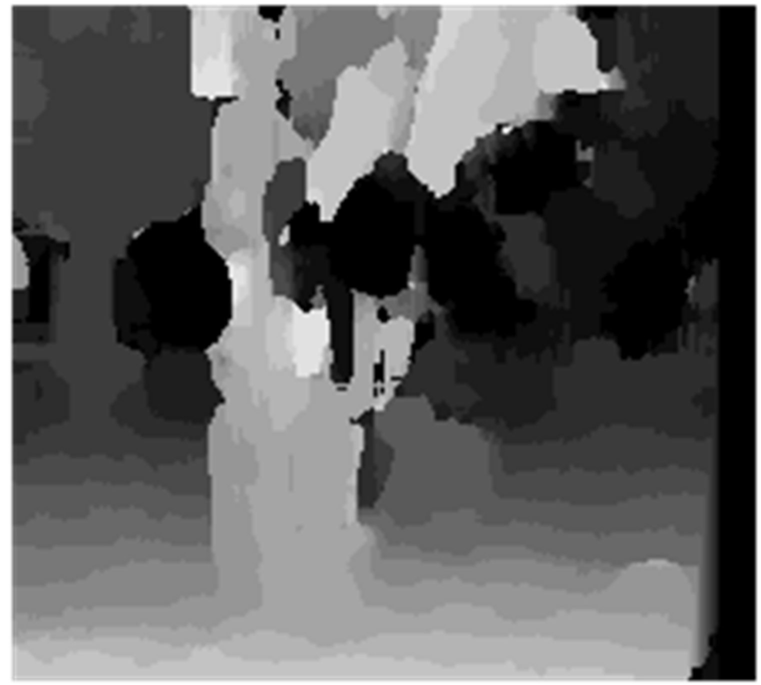


# Selecting window size

- Small window: more detail, but more noise
- Large window: more robustness, less detail
- Example:

# Selecting window size

3 pixel window



20 pixel window

Why?

# Non-square windows

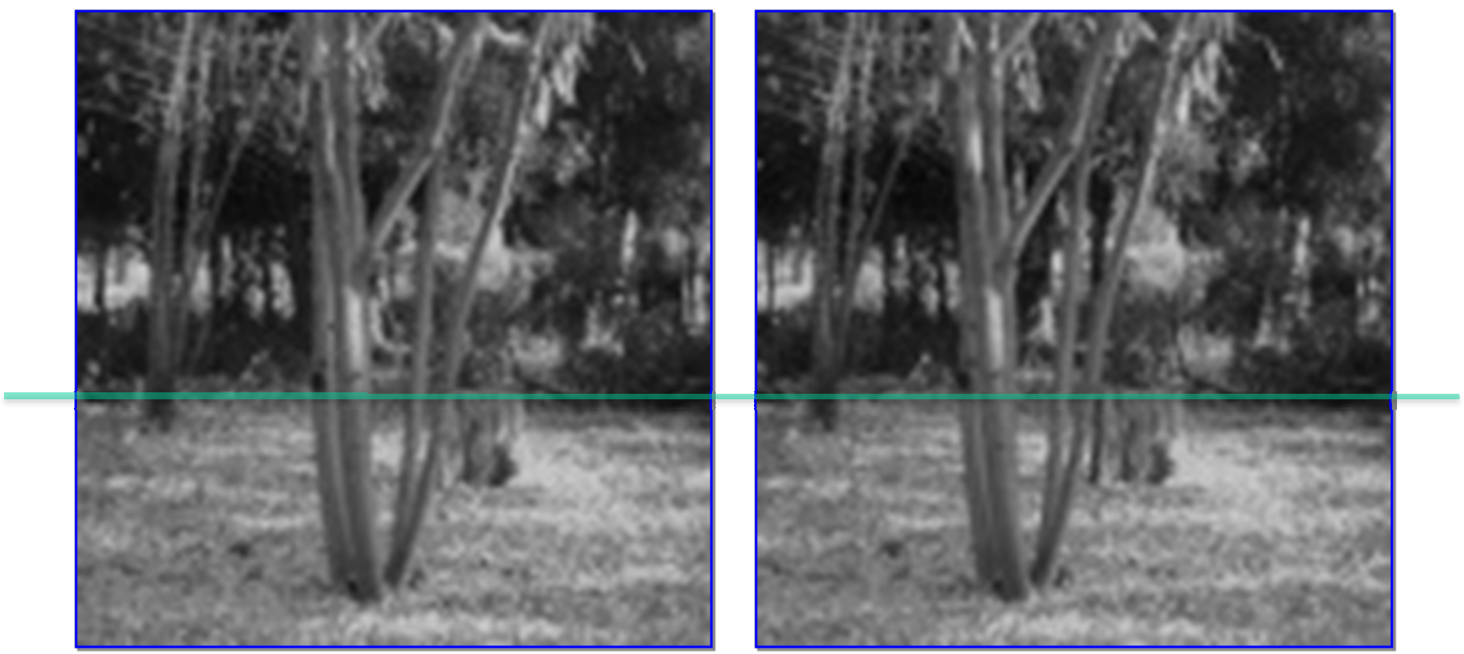
- Compromise: have a large window, but higher weight near the center
- Example: Gaussian
- Example: Shifted windows (computation cost?)

# Problems with window matching

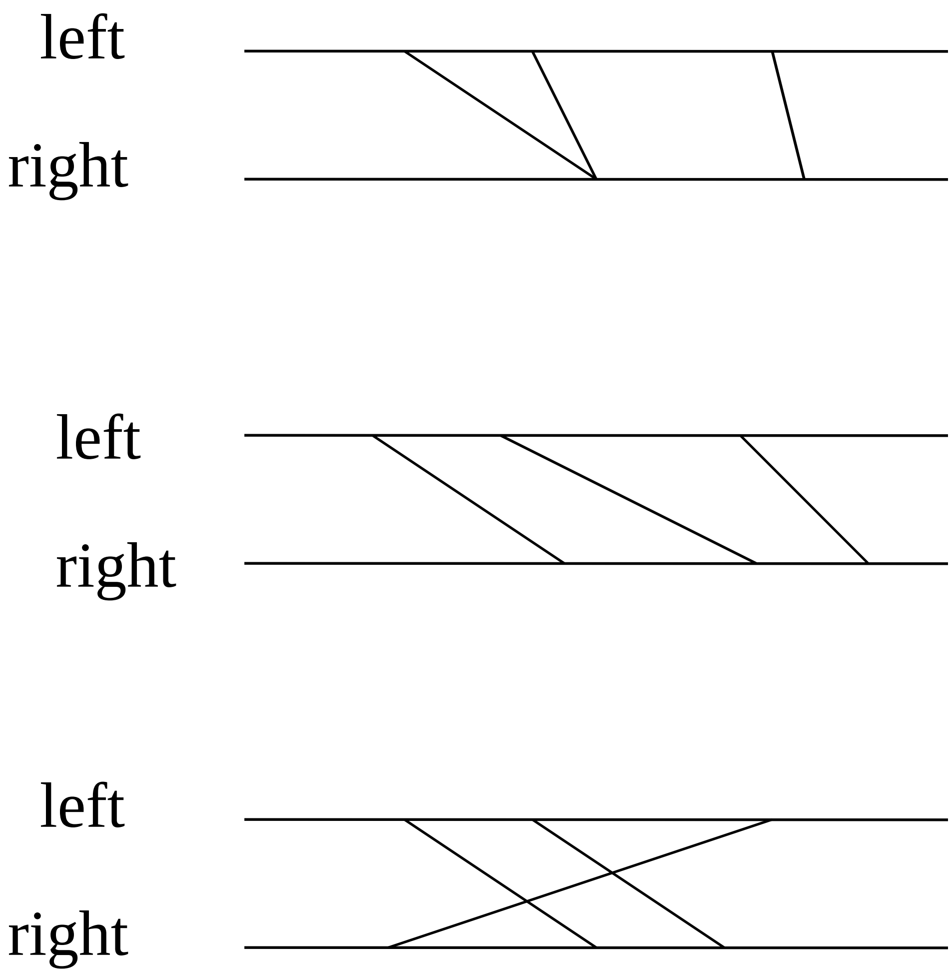
- No guarantee that the matching is one-to-one
- Hard to balance window size and smoothness

# A global approach

- Finding correspondence between a pair of epipolar lines for all pixels simultaneously



# A global approach



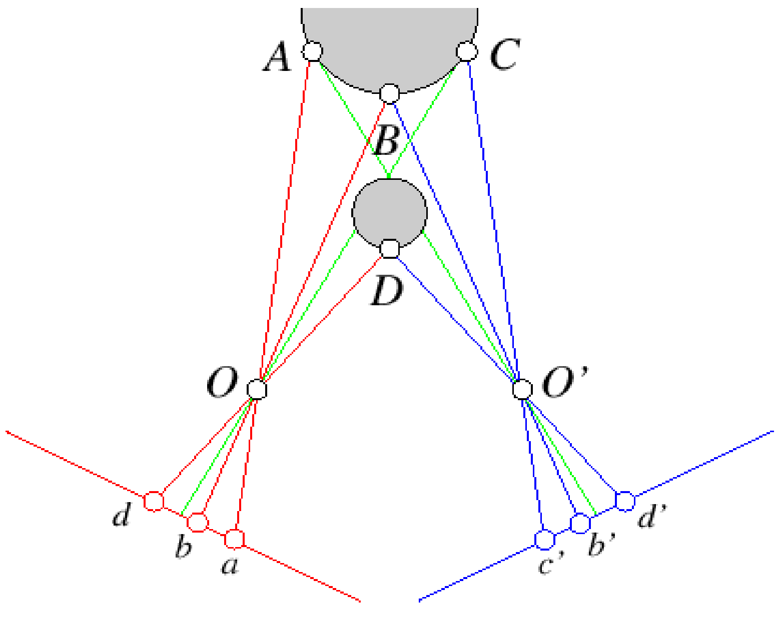

Define an evaluation score for each configuration, choose the best matching configuration

# A global approach

- How to define the evaluation score?
  - How about the sum of corresponding pixel difference?

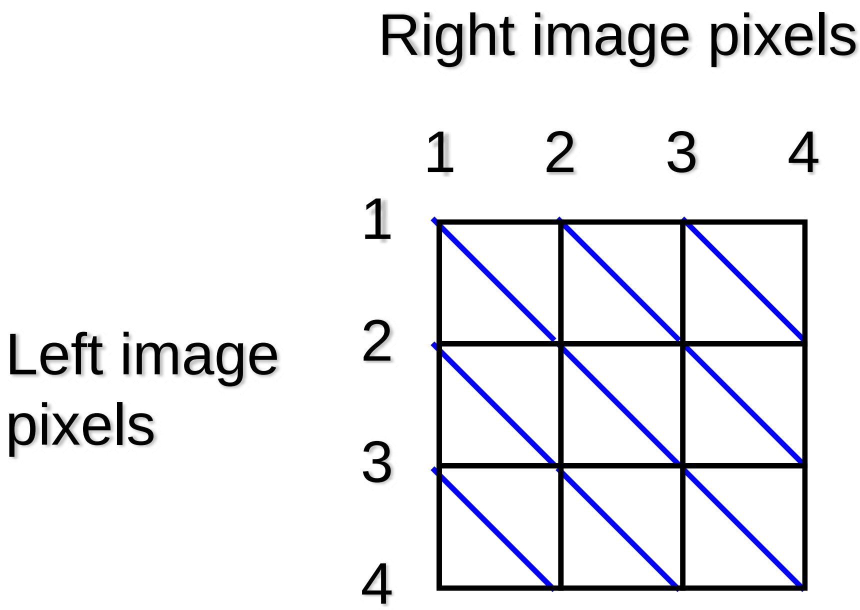
# Ordering constraint

- Order of matching features usually the same in both images
- But not always: occlusion



# Dynamic programming

- Treat pixel correspondence as graph problem

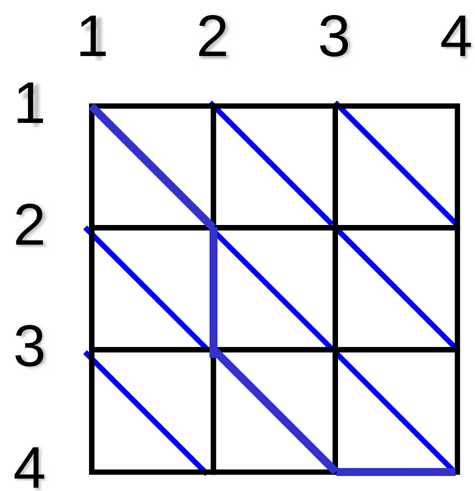


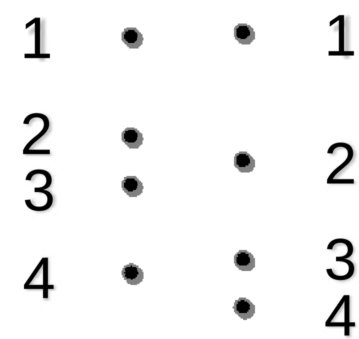
# Dynamic programming

- Find min-cost path through graph

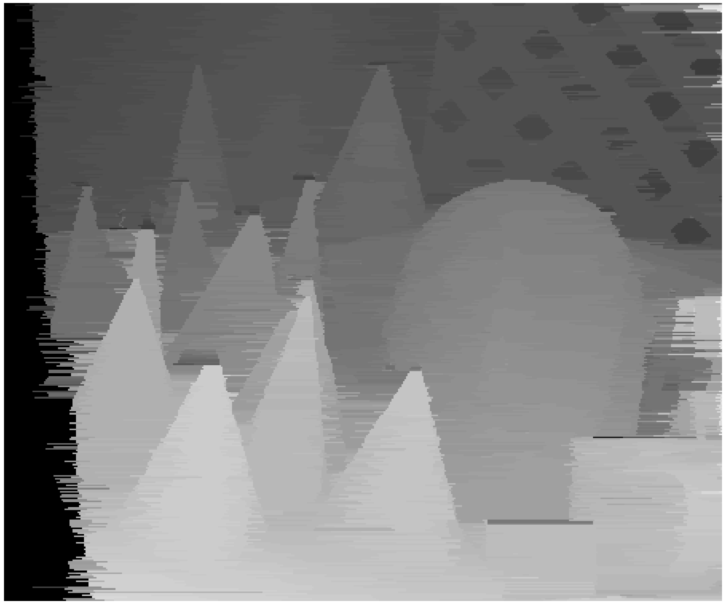
Right image pixels

Left image pixels



# Dynamic Programming Results

# Energy minimization

- Another global approach to improve quality of correspondences
- Assumption: disparities vary (mostly) smoothly
- Minimize energy function:

$$E_{data}+\lambda E_{smoothness}$$

- $E_{data}$: how well does disparity match data
- $E_{smoothness}$: how well does disparity match that of neighbors – regularization

# Stereo as energy minimization

- Matching Cost Formulated as Energy
  - "data" term penalizing bad matches

$$D(x, y, d) = \left|\mathbf{I}(x, y) - \mathbf{J}(x + d, y)\right|$$

  - "neighborhood term" encouraging spatial smoothness

$$V(d_1, d_2) = \text{cost of adjacent pixels with labels d1 and d2}$$

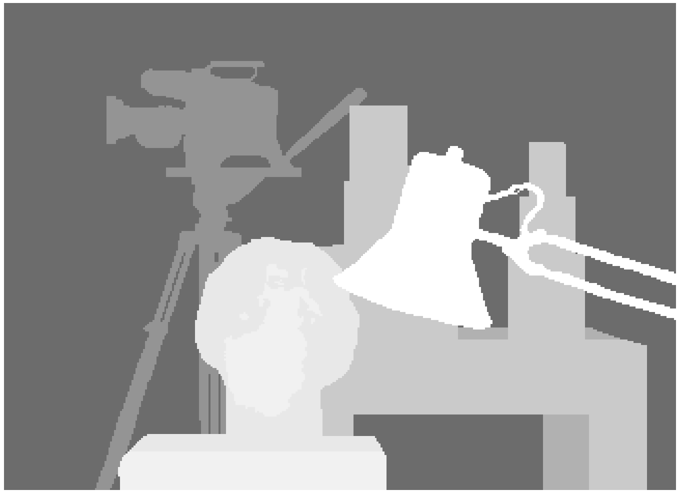
$$= \left|d_1 - d_2\right| \quad \text{(or something similar)}$$

$$E(\{d\}) = \sum_{(x,y)} D(x, y, d_{x,y}) + \sum_{neighbors\ (x1,y1),(x2,y2)} V(d_{x1,y1}, d_{x2,y2})$$

# Energy minimization

$$E(\{d\}) = \sum_{(x,y)} D(x, y, d_{x,y}) + \sum_{neighbors\ (x1,y1),(x2,y2)} V(d_{x1,y1}, d_{x2,y2})$$

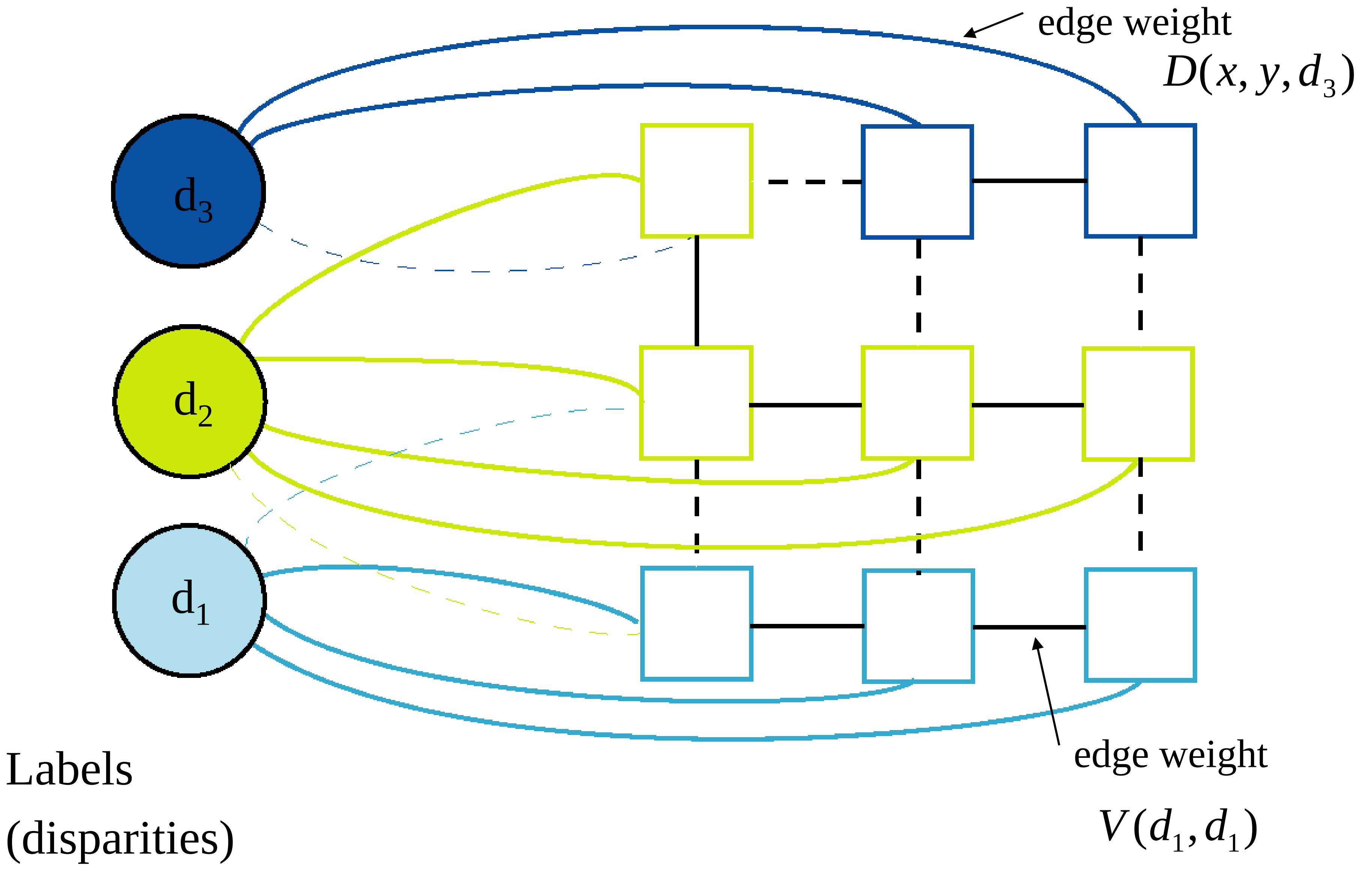
- Many local minimum
  - Why?
  - Gradient descent doesn't work well
- In practice, disparities only piecewise smooth
- Design smoothness function that doesn't penalize large jumps too much
  - Example: $V(\alpha,\beta)=\min(|\alpha-\beta|, K)$
    - – Non-convex



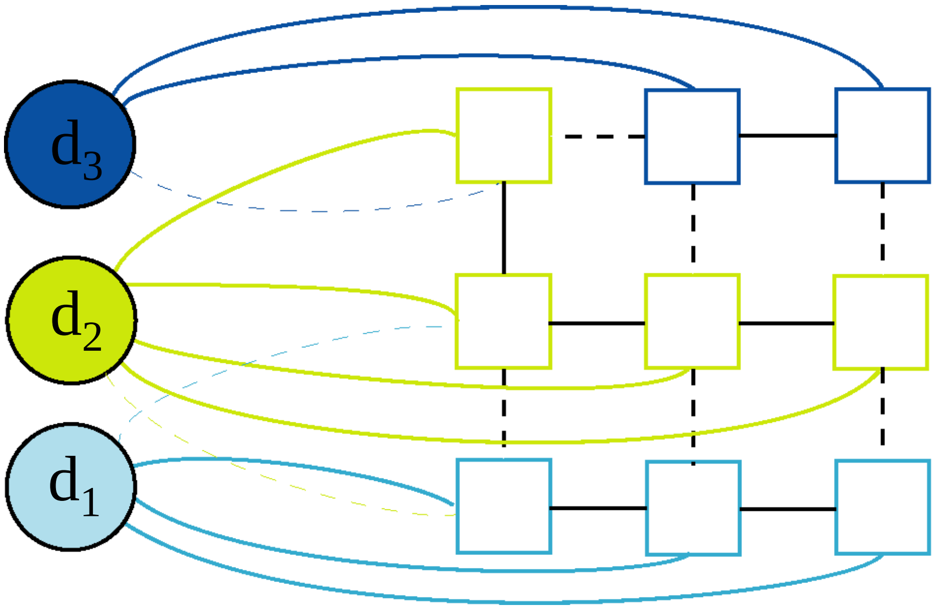
# Energy minimization

- Hard to find global minima of non-smooth functions
  - Many local minima
  - Provably NP-hard
- Practical algorithms look for approximate minima (e.g., simulated annealing)

# Energy minimization via graph cuts

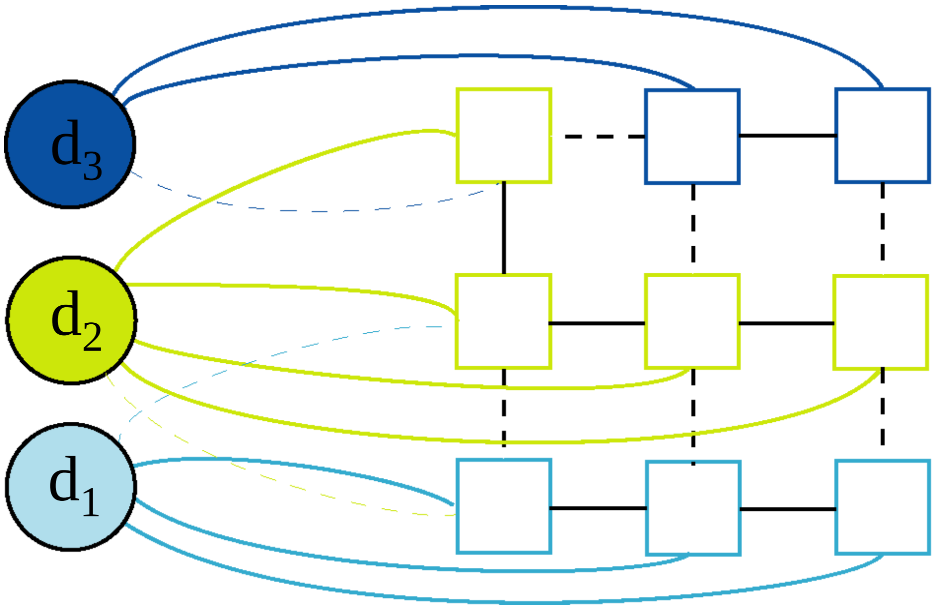


# Energy minimization via graph cuts




- Graph Cost
  - Matching cost between images
  - Neighborhood matching term
  - Goal: figure out which labels are connected to which pixels

# Energy minimization via graph cuts



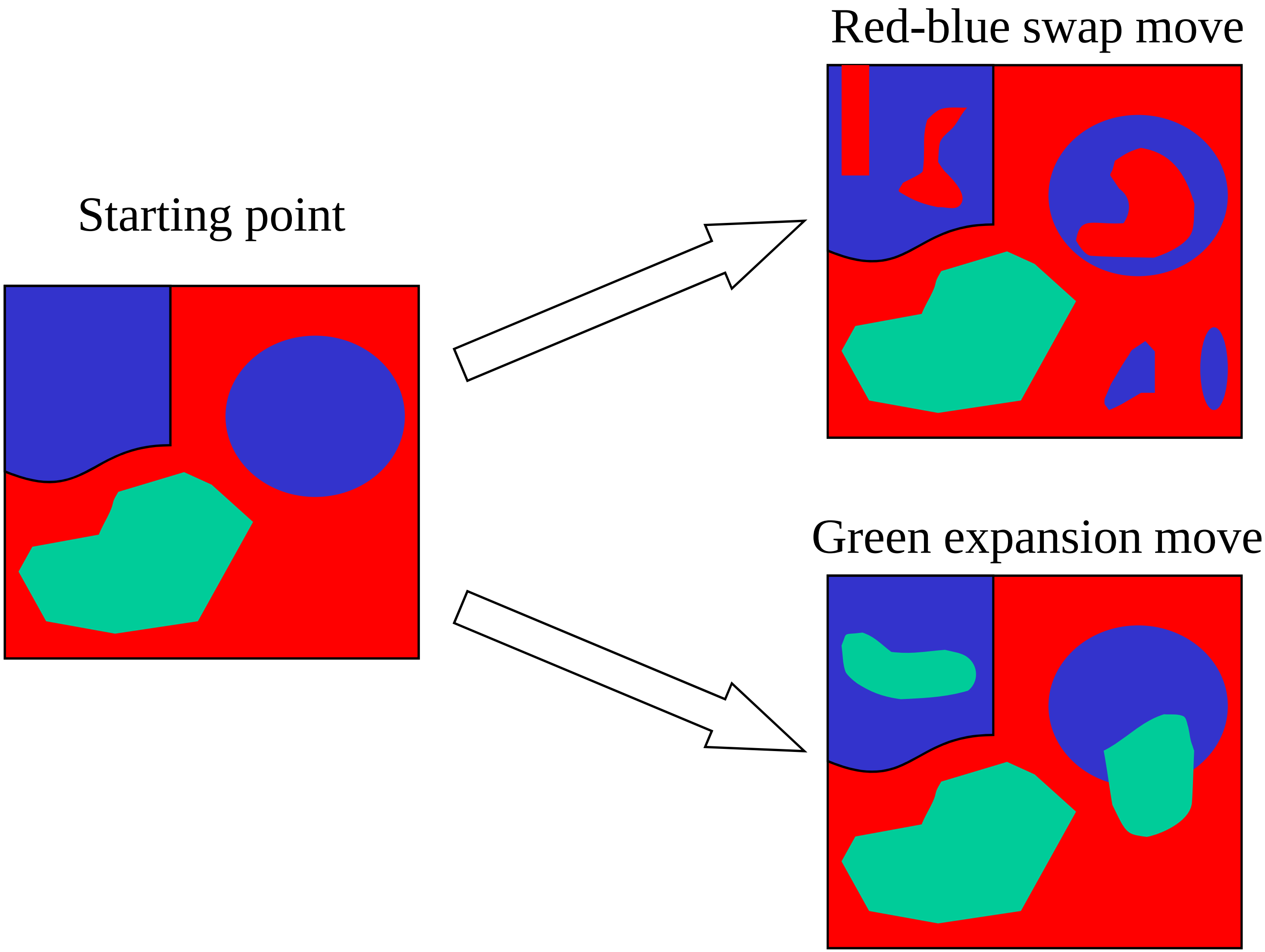

- Graph Cut
  - Delete enough edges so that
    - each pixel is connected to exactly one label node
  - Cost of a cut: sum of deleted edge weights
  - Finding min cost cut equivalent to finding global minimum of energy function

# Computing a multiway cut

- With 2 labels: classical min-cut problem
  - Solvable by standard flow algorithms
    - polynomial time in theory, nearly linear in practice
  - More than 2 terminals: NP-hard

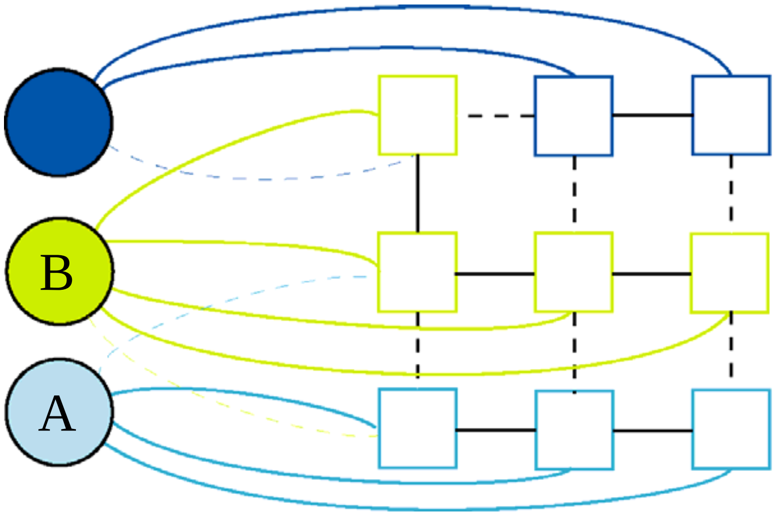
    [Dahlhaus *et al.*, STOC '92]
- Efficient approximation algorithms exist
  - Yuri Boykov, Olga Veksler and Ramin Zabih, Fast Approximate Energy Minimization via Graph Cuts, International Conference on Computer Vision, September 1999.
  - Within a factor of 2 of optimal
  - Computes local minimum in a strong sense
    - even very large moves will not improve the energy

# Move examples

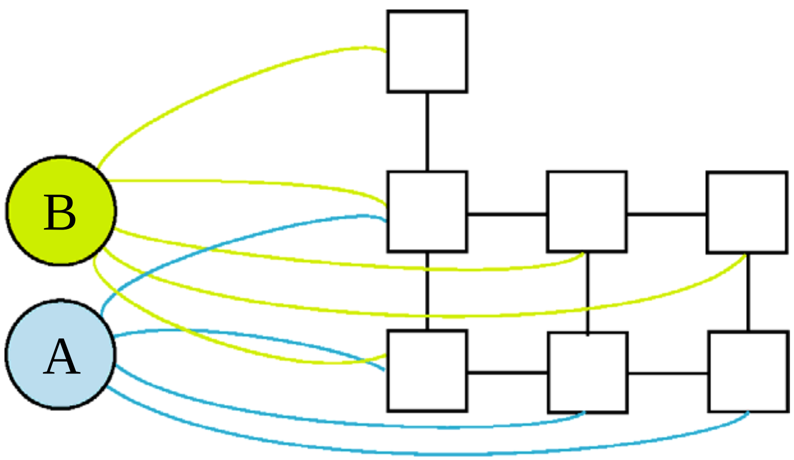


# The swap move algorithm

1. Start with an arbitrary labeling
2. Cycle through every label pair *(A,B)* in some order
   2.1 Find the lowest *E* labeling within a single *AB*-swap
   2.2 Go there if it's lower *E* than the current labeling
3. If *E* did not decrease in the cycle, we're done
   Otherwise, go to step 2




Original graph




AB subgraph
(run min-cut on this graph)

# The expansion move algorithm

1. Start with an arbitrary labeling
2. Cycle through every label *A* in some order

    2.1 Find the lowest *E* labeling within a single *A*-expansion

    2.2 Go there if it's lower *E* than the current labeling

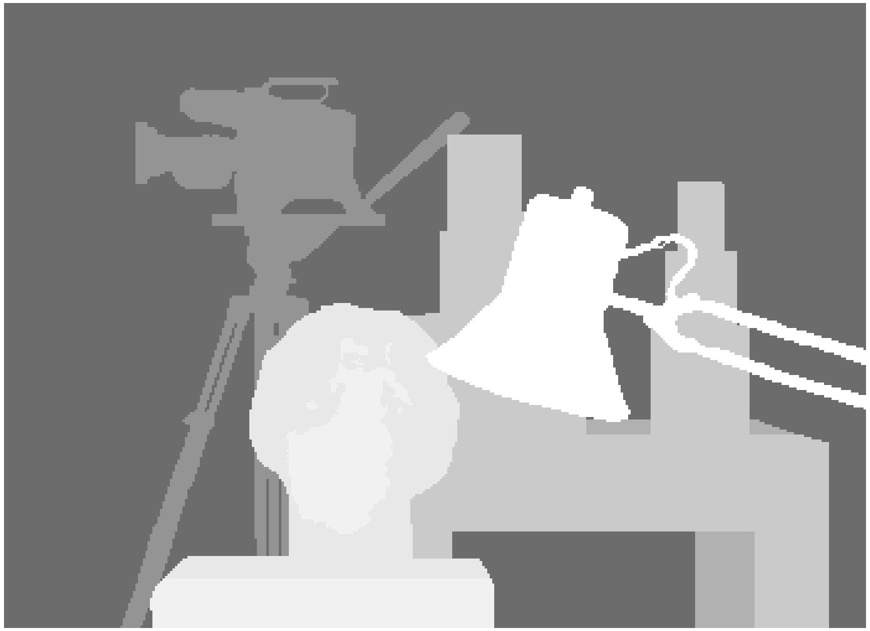
3. If *E* did not decrease in the cycle, we're done     Otherwise, go to step 2

Multi-way cut →A sequence of binary optimization problems

# Stereo results

- Data from University of Tsukuba

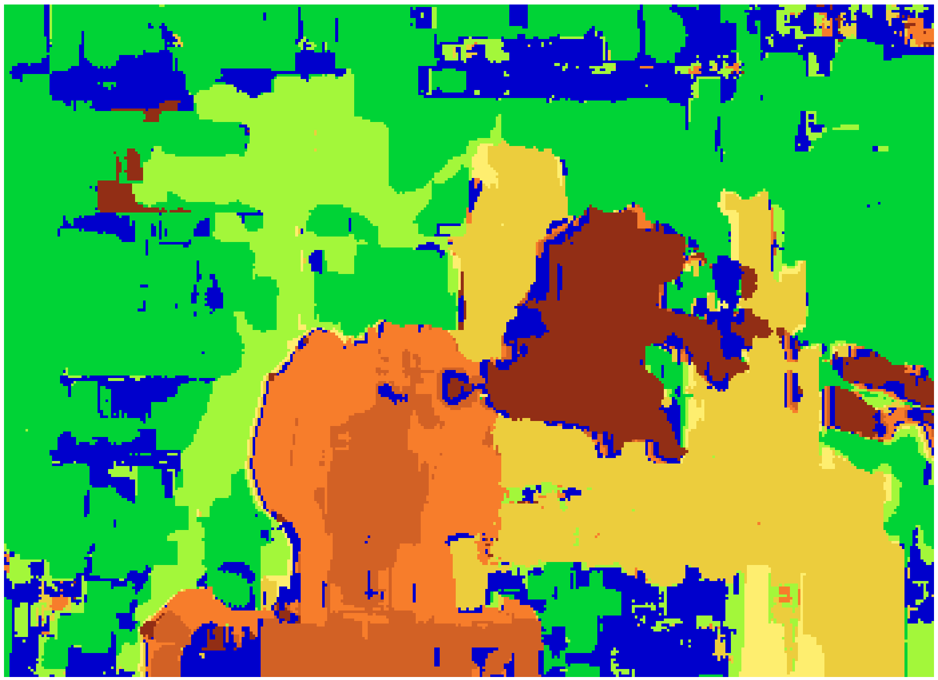
scene



ground truth

http://cat.middlebury.edu/stereo/

# Results with window correlation



normalized correlation
(best window size)

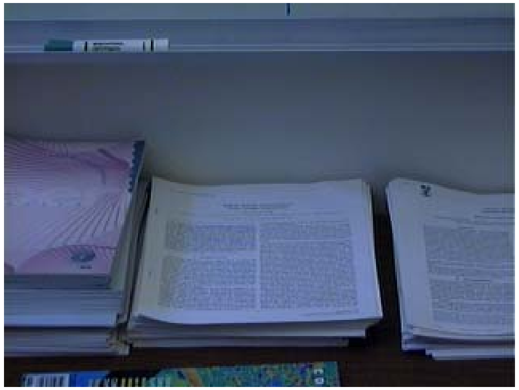
ground truth

# Results with graph cuts

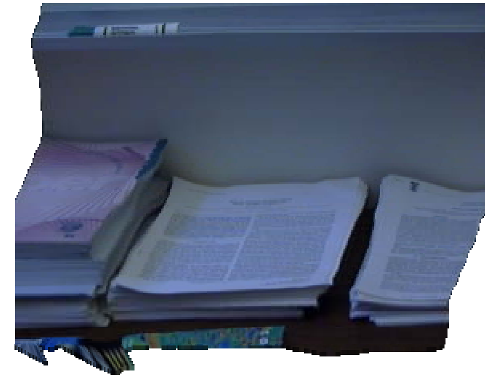
graph cuts
(Potts model *E*,
expansion move algorithm)

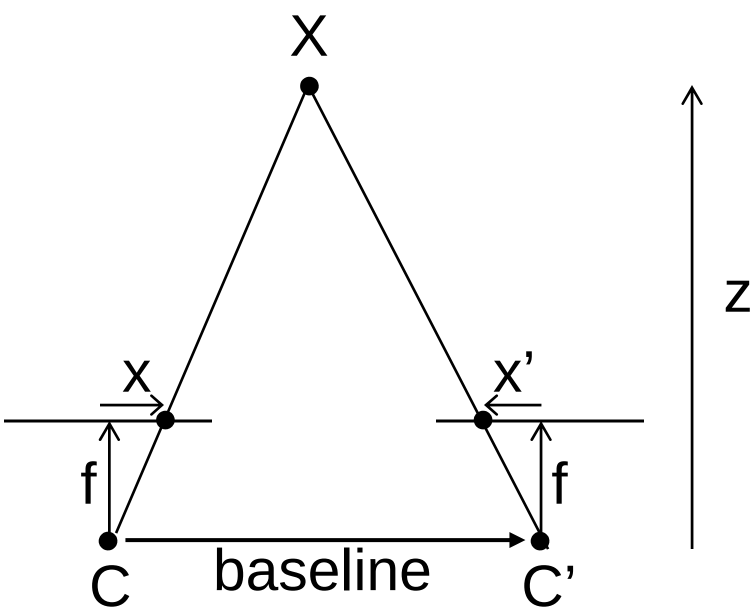
ground truth

# Depth from disparity


input image (1 of 2)


depth map
[Szeliski & Kang ‘95]


3D rendering




$$disparity = x - x' = \frac{baseline * f}{z}$$

# Real-time stereo

Nomad robot searches for meteorites in Antartica
http://www.frc.ri.cmu.edu/projects/meteorobot/index.html

- Used for robot navigation (and other tasks)
  - Several software-based real-time stereo techniques have been developed (most based on simple discrete search)

# Stereo reconstruction pipeline

- Steps
  - Calibrate cameras
  - Rectify images
  - Compute disparity
  - Estimate depth

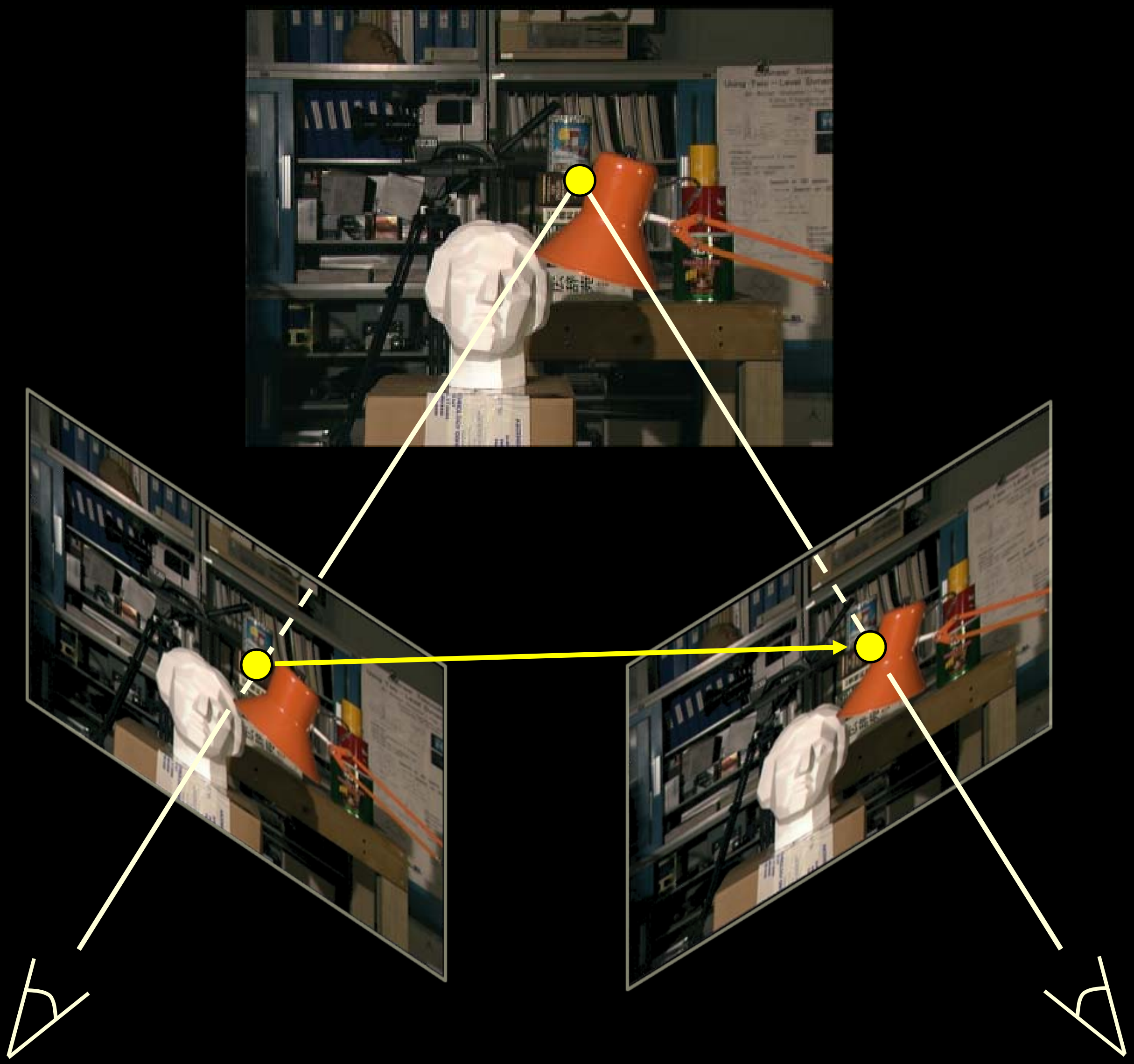
What will cause errors?

  - Camera calibration errors
  - Poor image resolution
  - Occlusions
  - Violations of brightness constancy (specular reflections)
  - Large motions
  - Low-contrast image regions

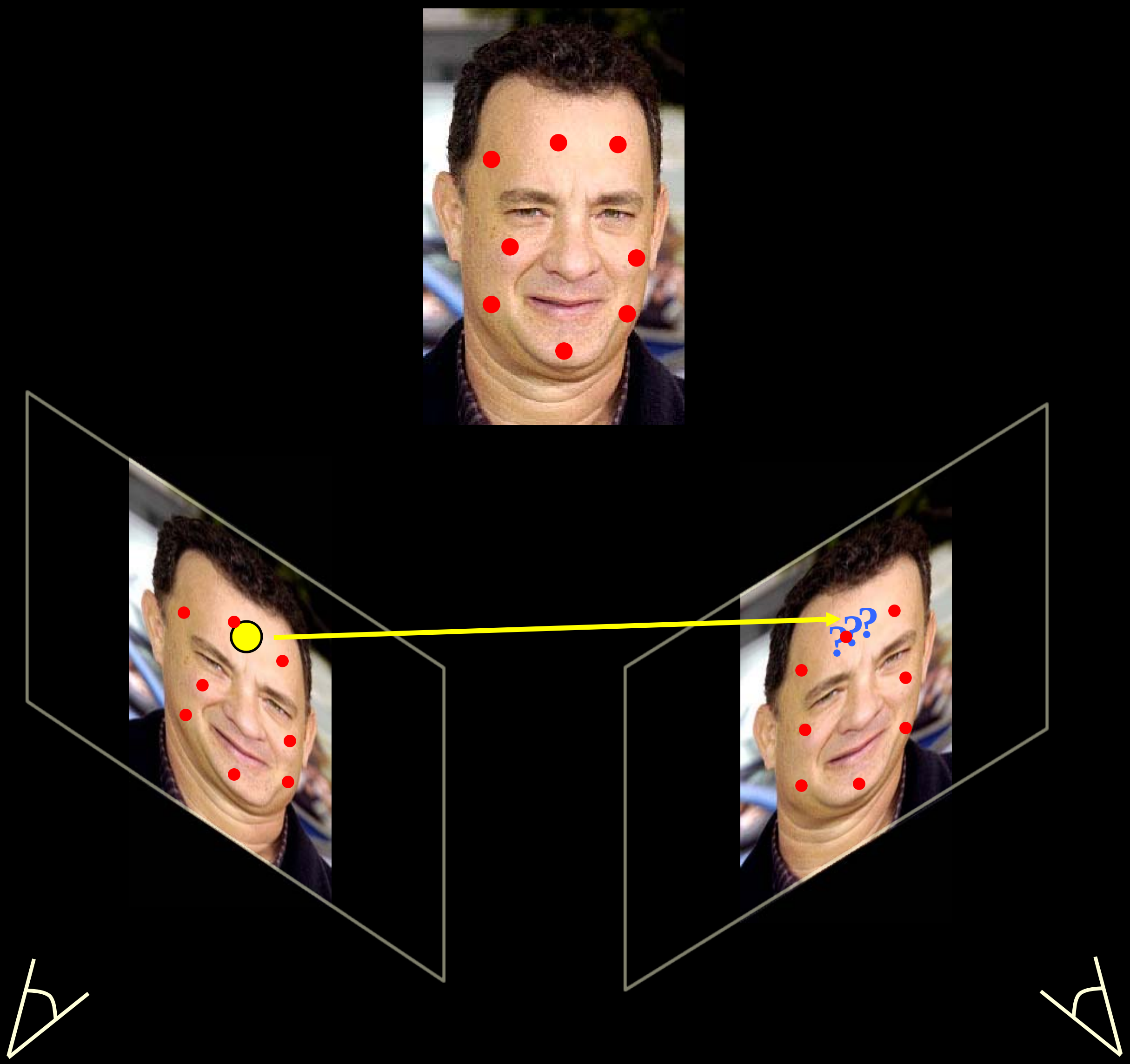
# Spacetime Stereo

Li Zhang, Noah Snavely, Brian Curless, Steven Seitz
CVPR 2003, SIGGRAPH 2004

# Stereo



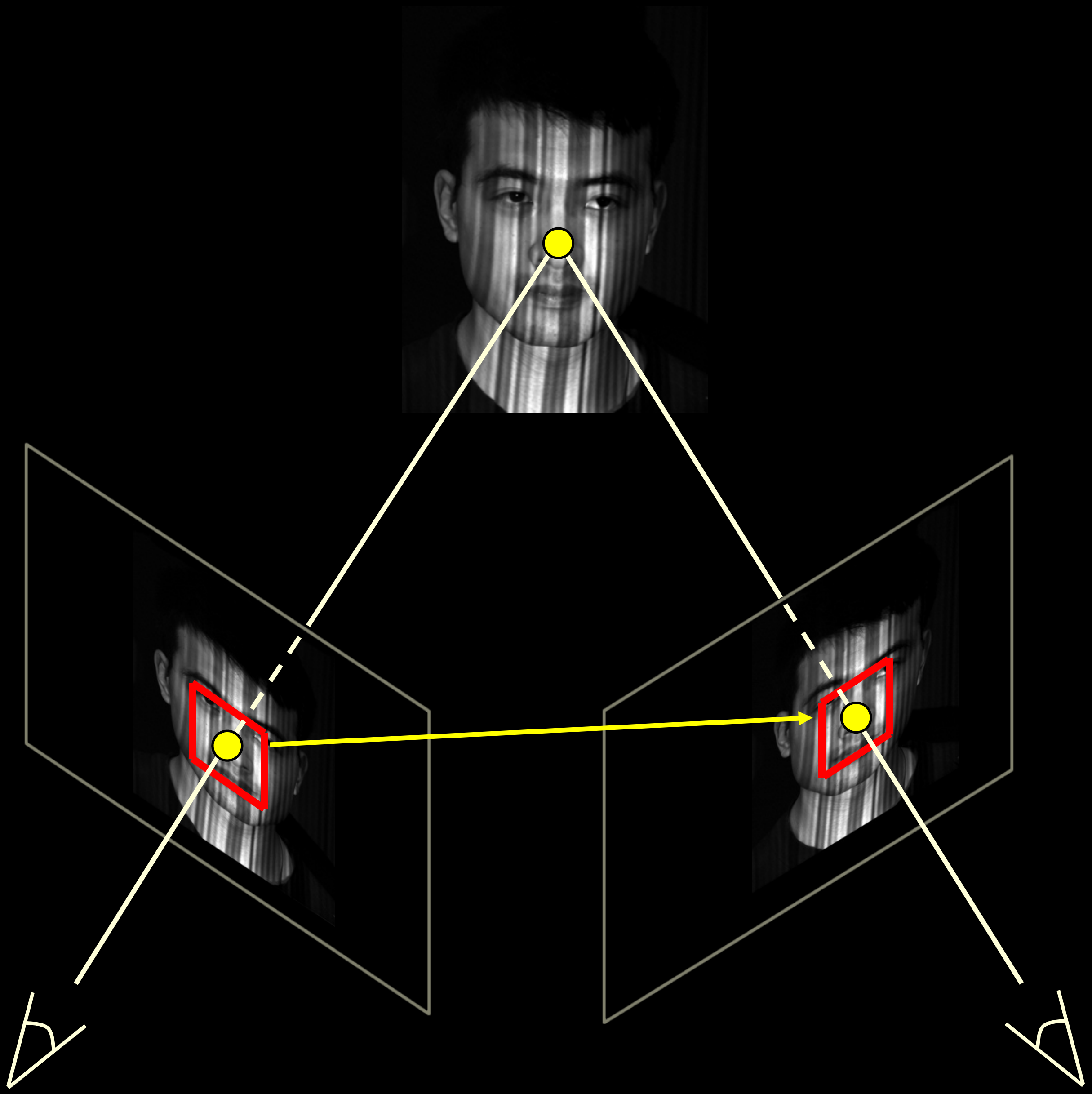
# Stereo



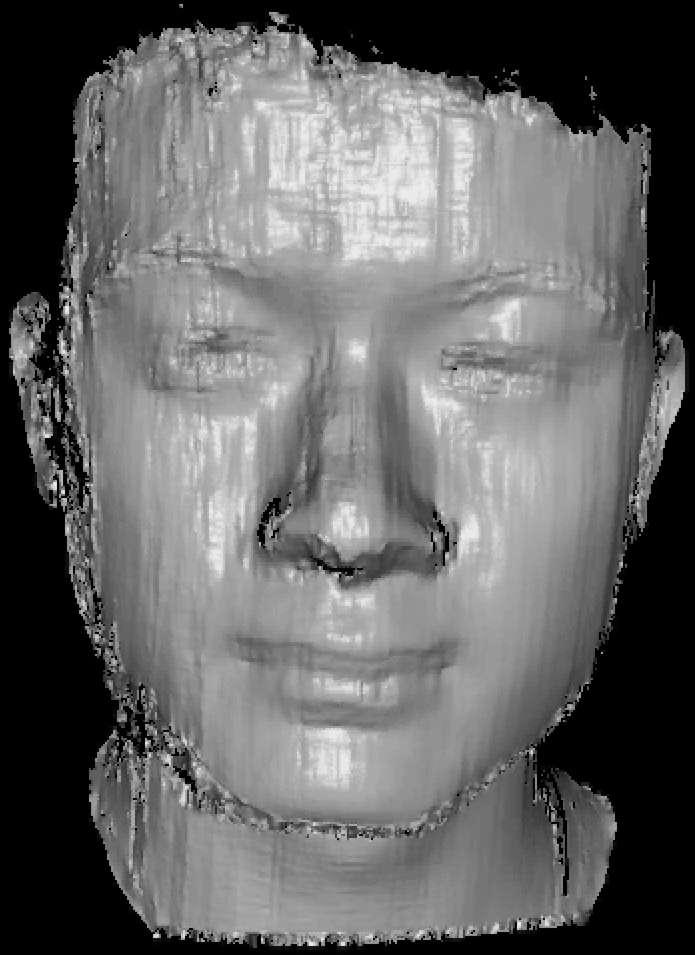
# Marker-based Face Capture

"The largest intractable problem with 'The Polar Express' is that the motion-capture technology used to create the human figures has resulted in a film filled with creepily unlifelike beings."

# Stereo

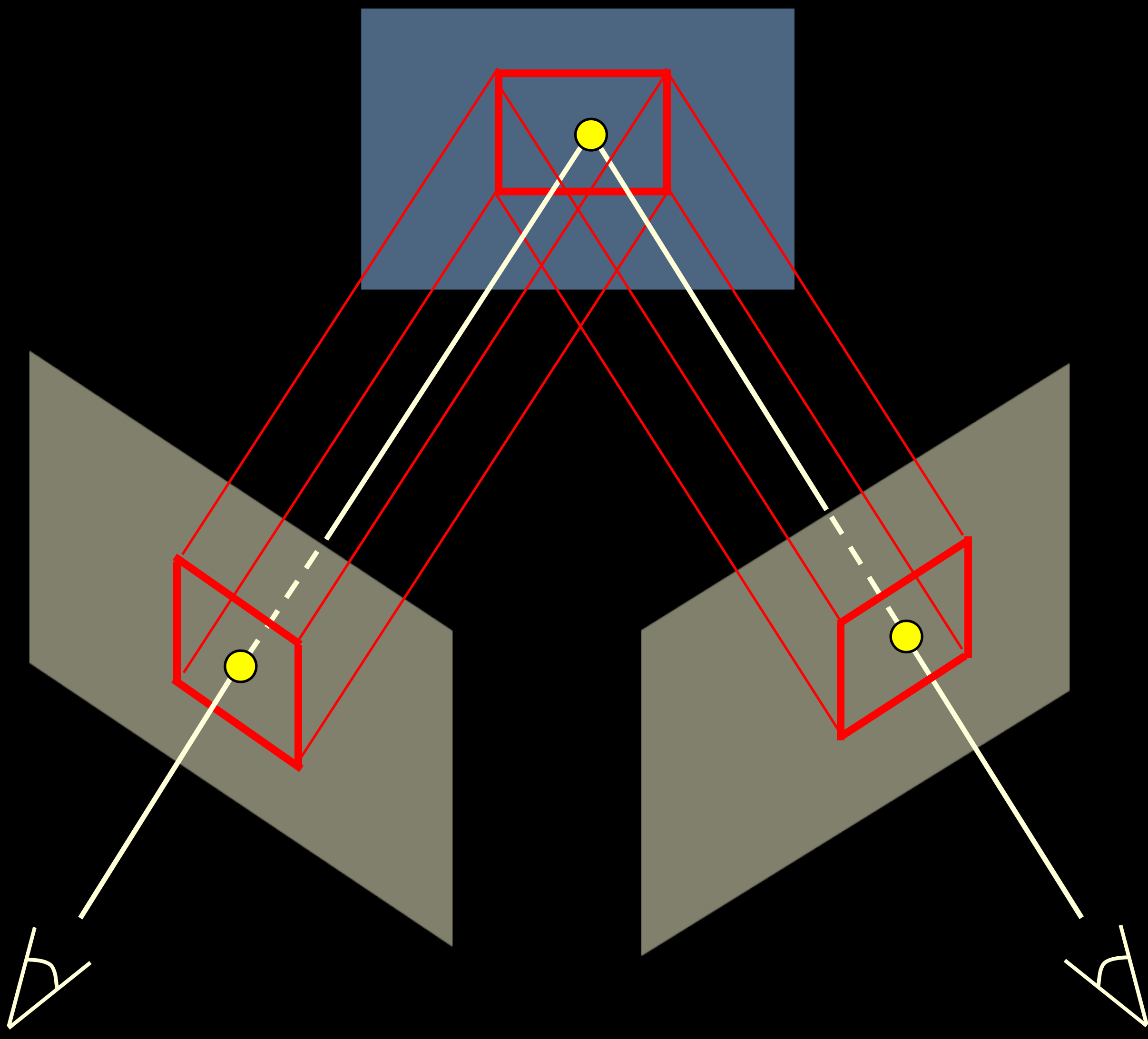


# Stereo

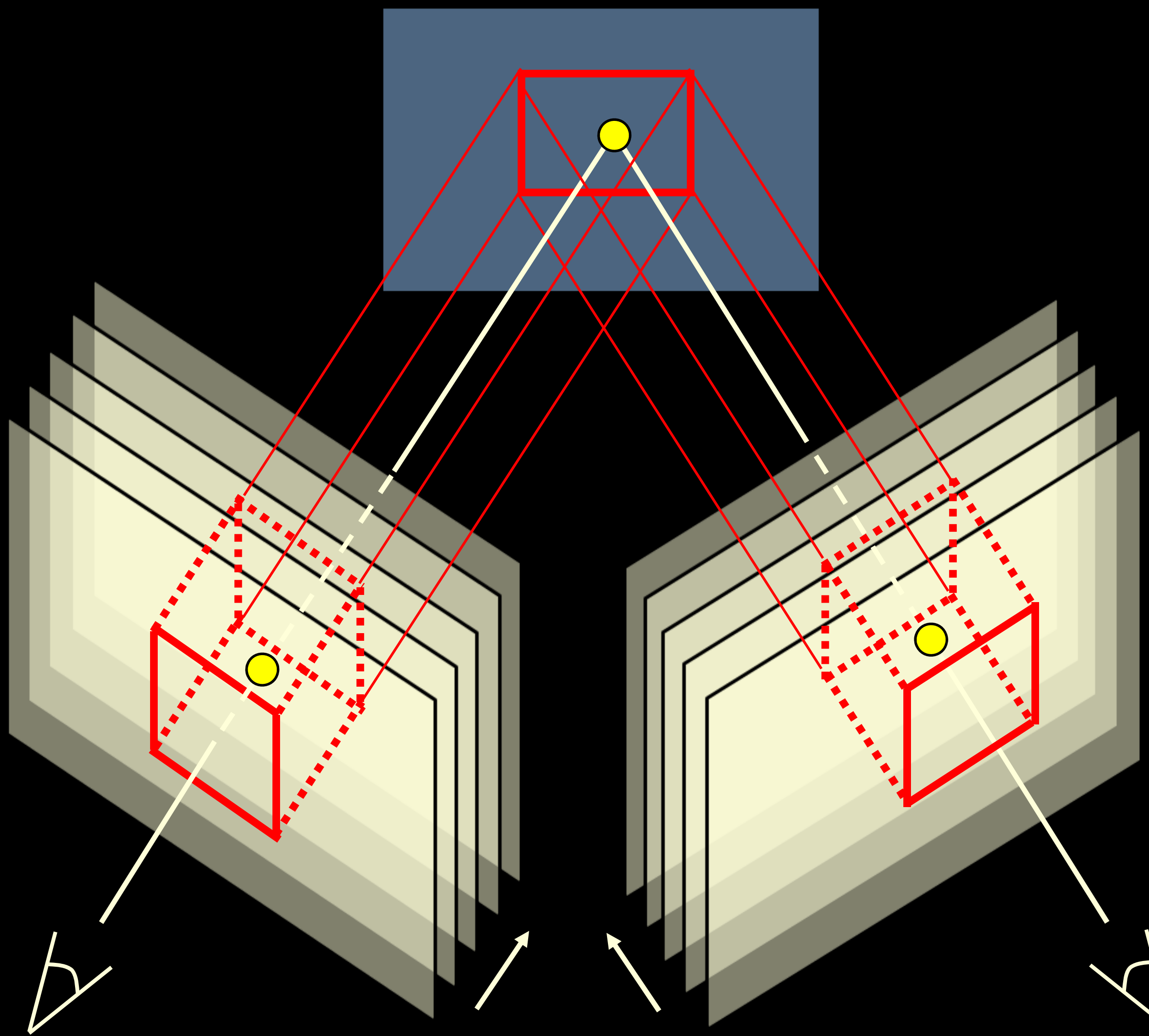


Inaccurate & Jittering

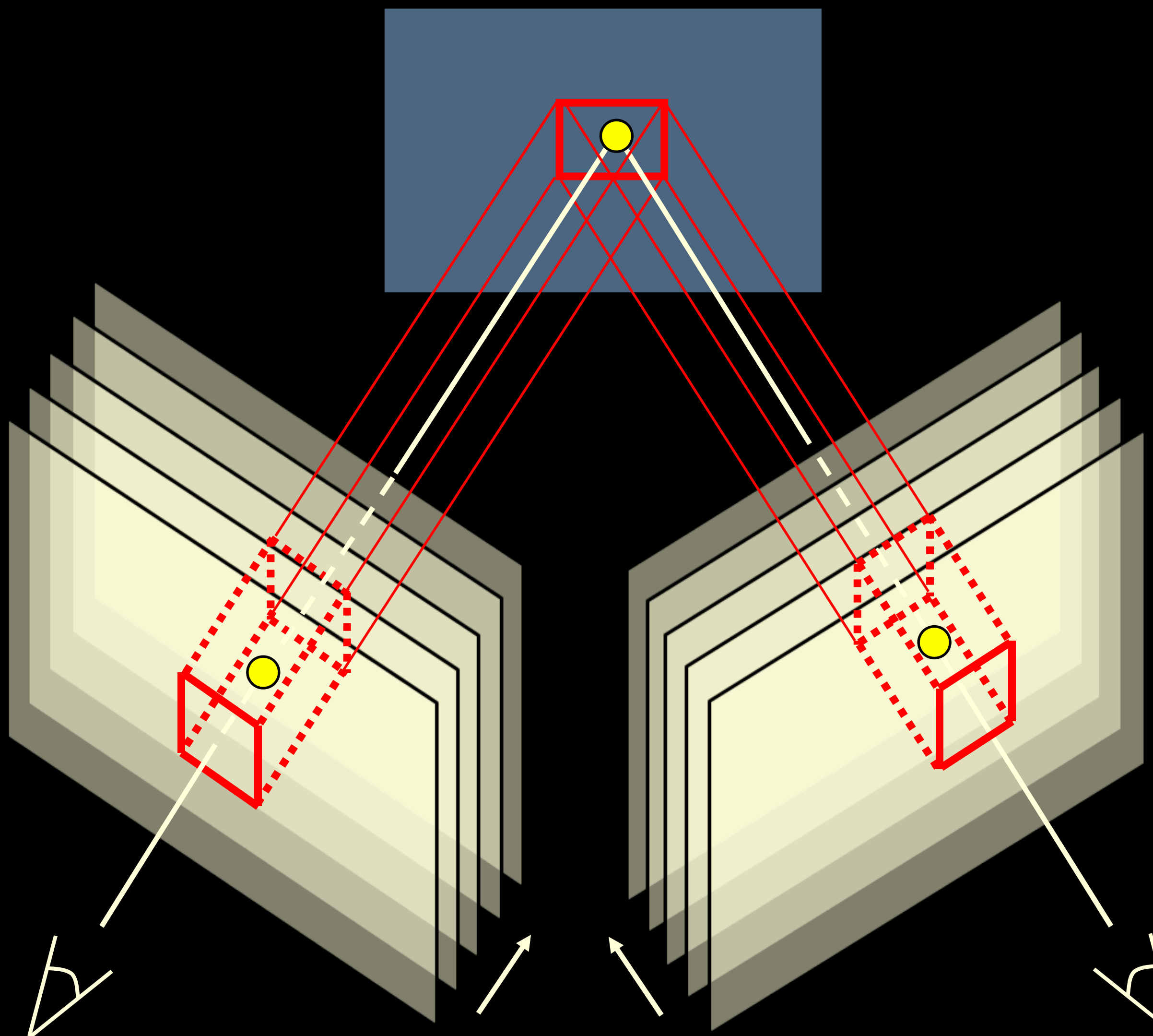
# Spacetime Stereo



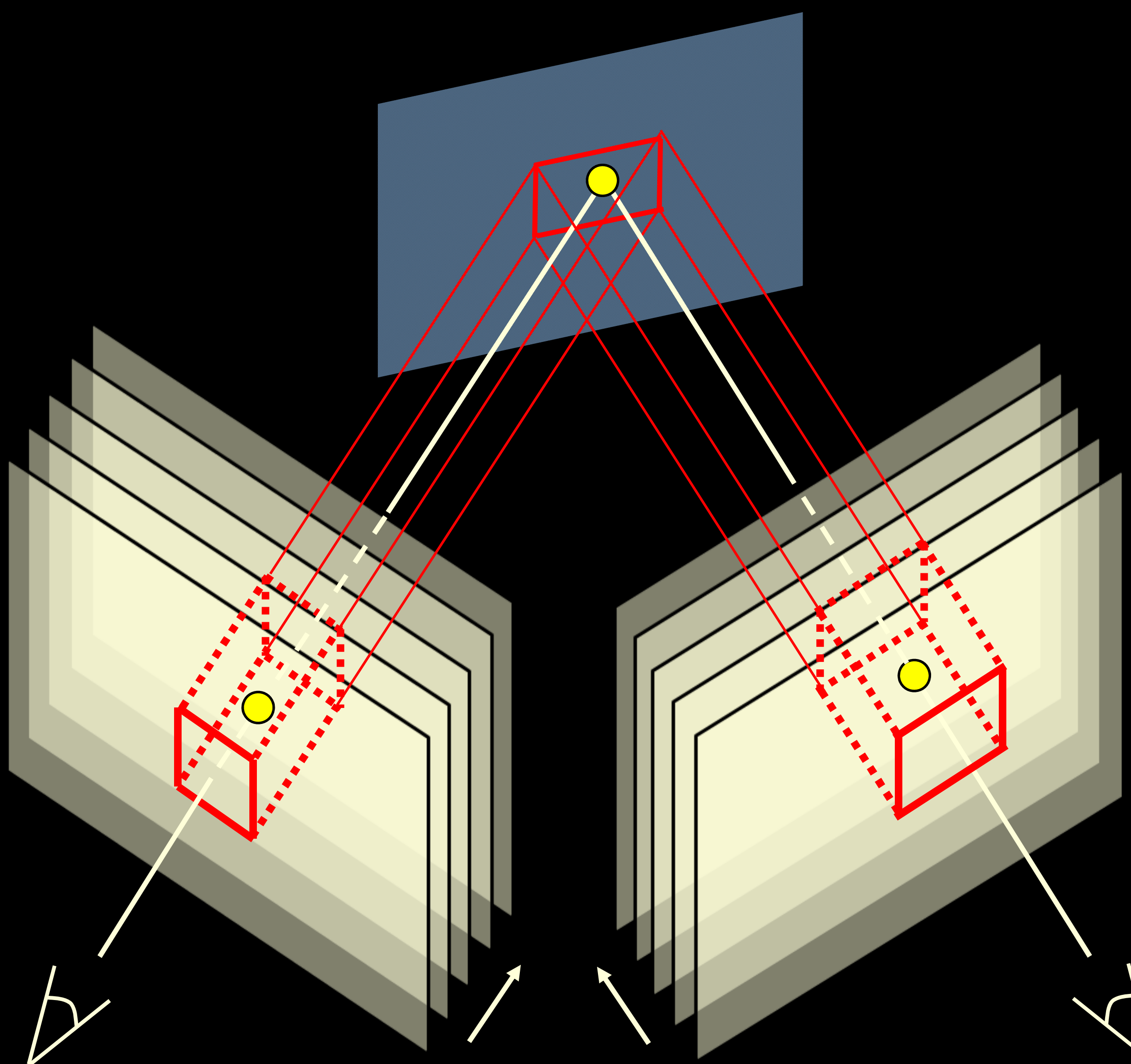
# Spacetime Stereo



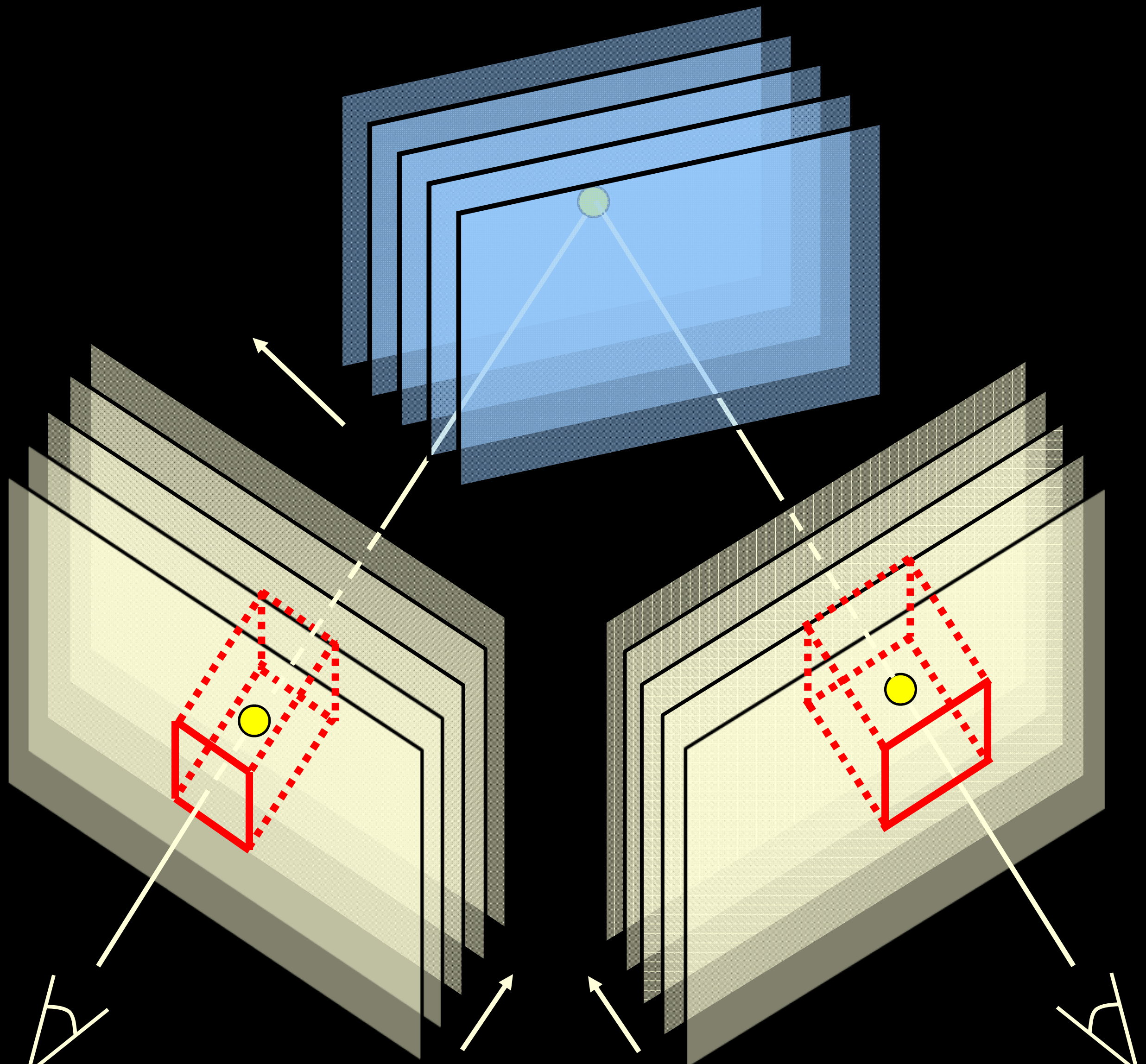
# Spacetime Stereo



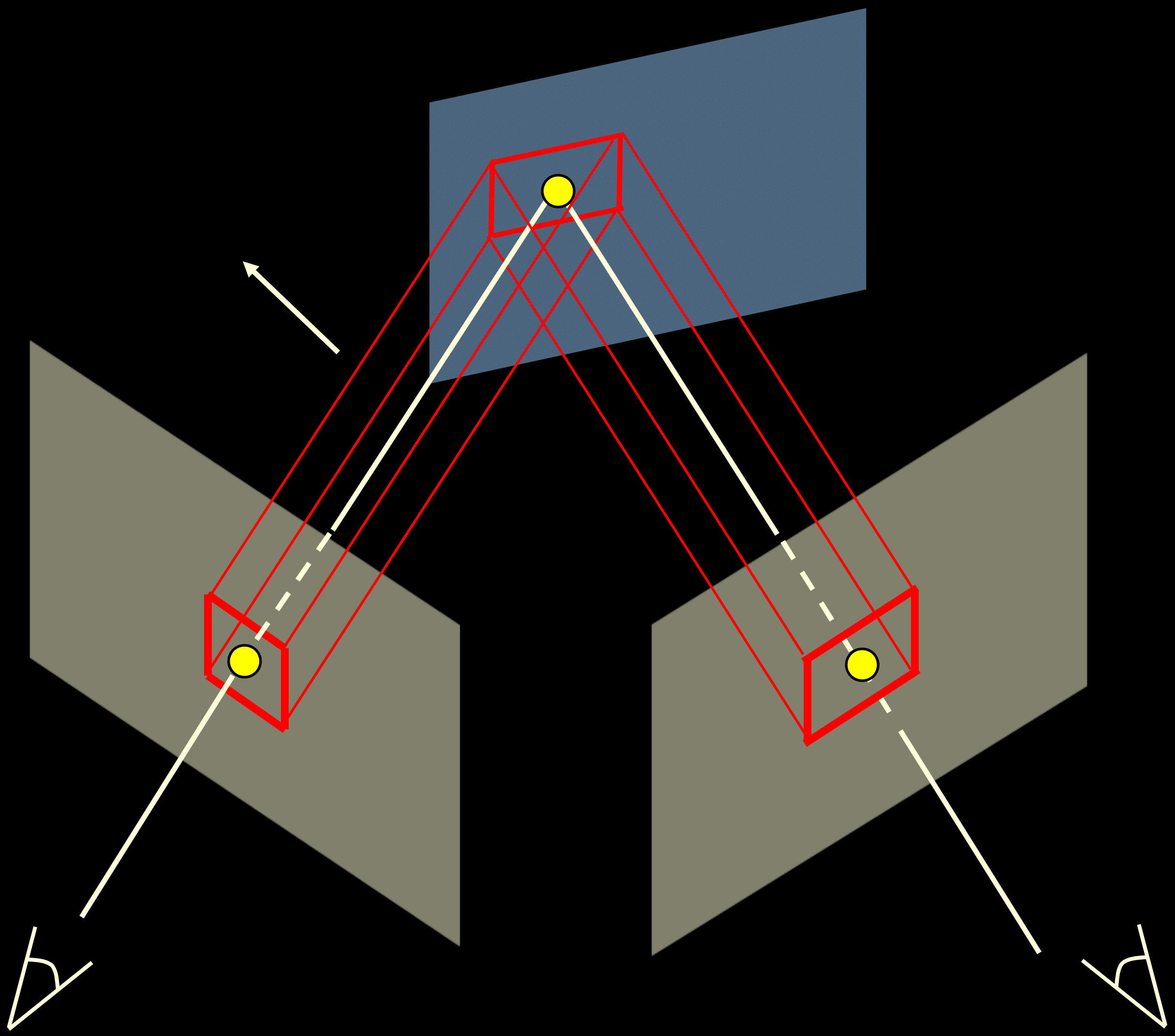
# Spacetime Stereo



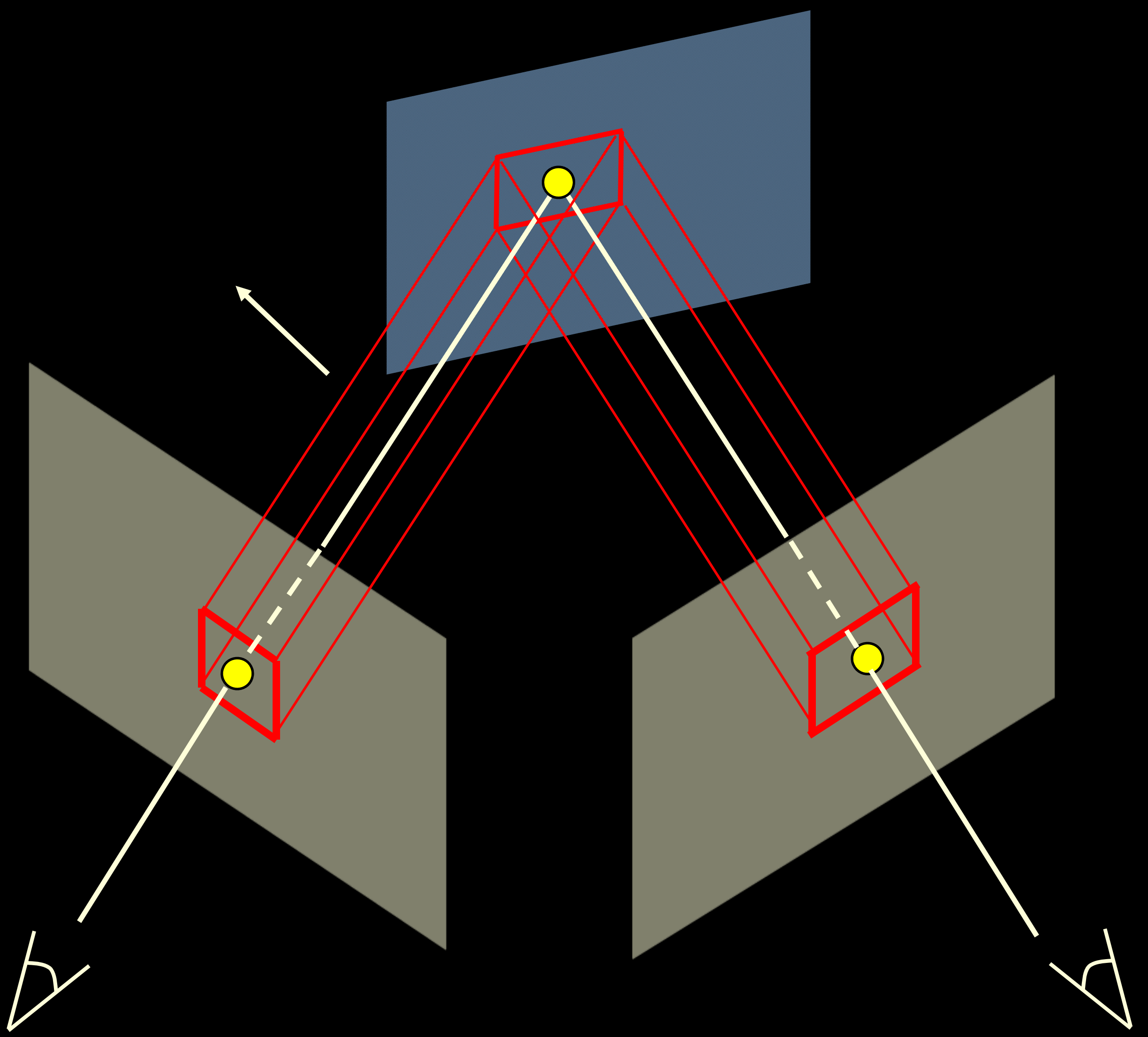
# Spacetime Stereo



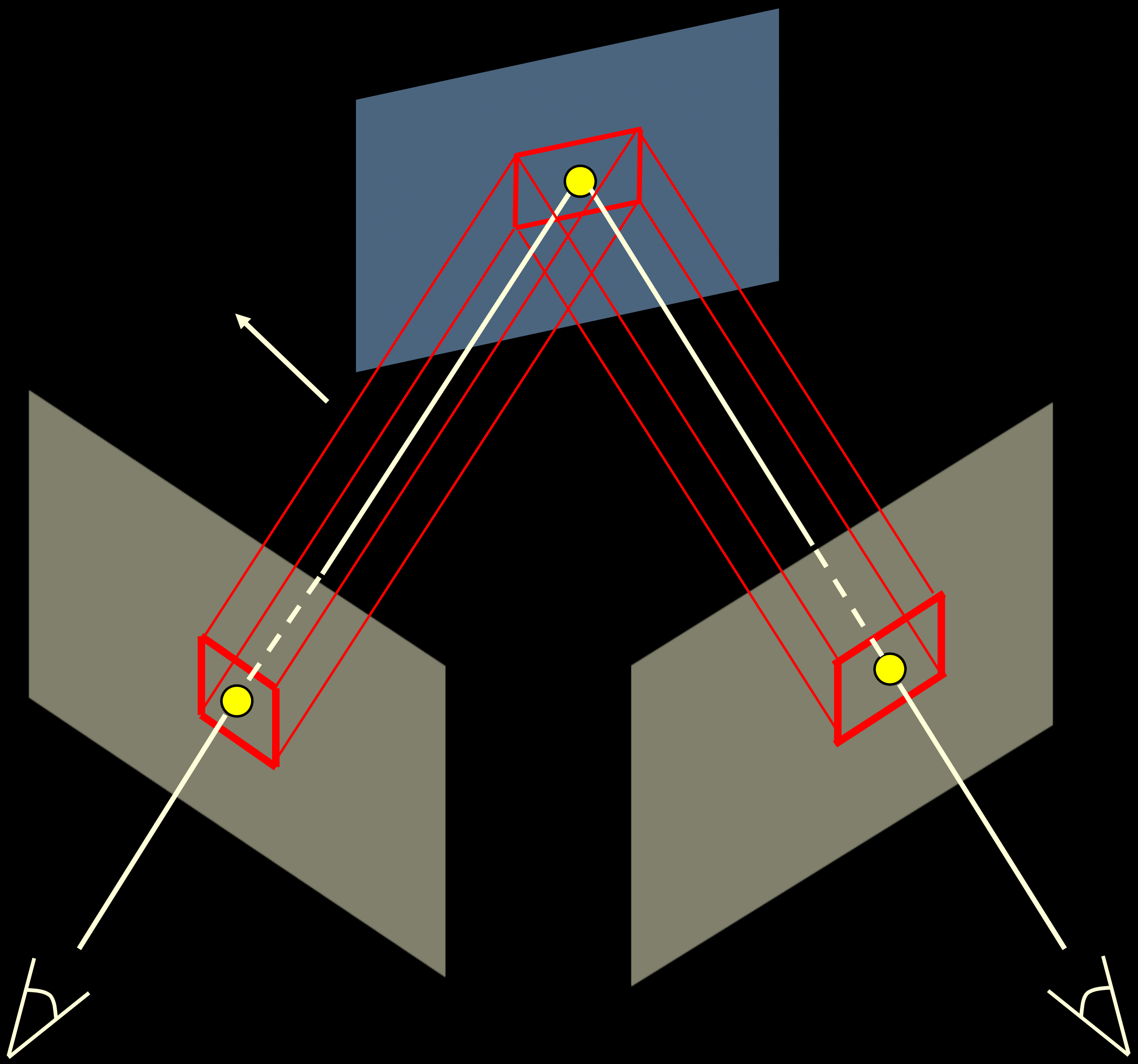
# Spacetime Stereo



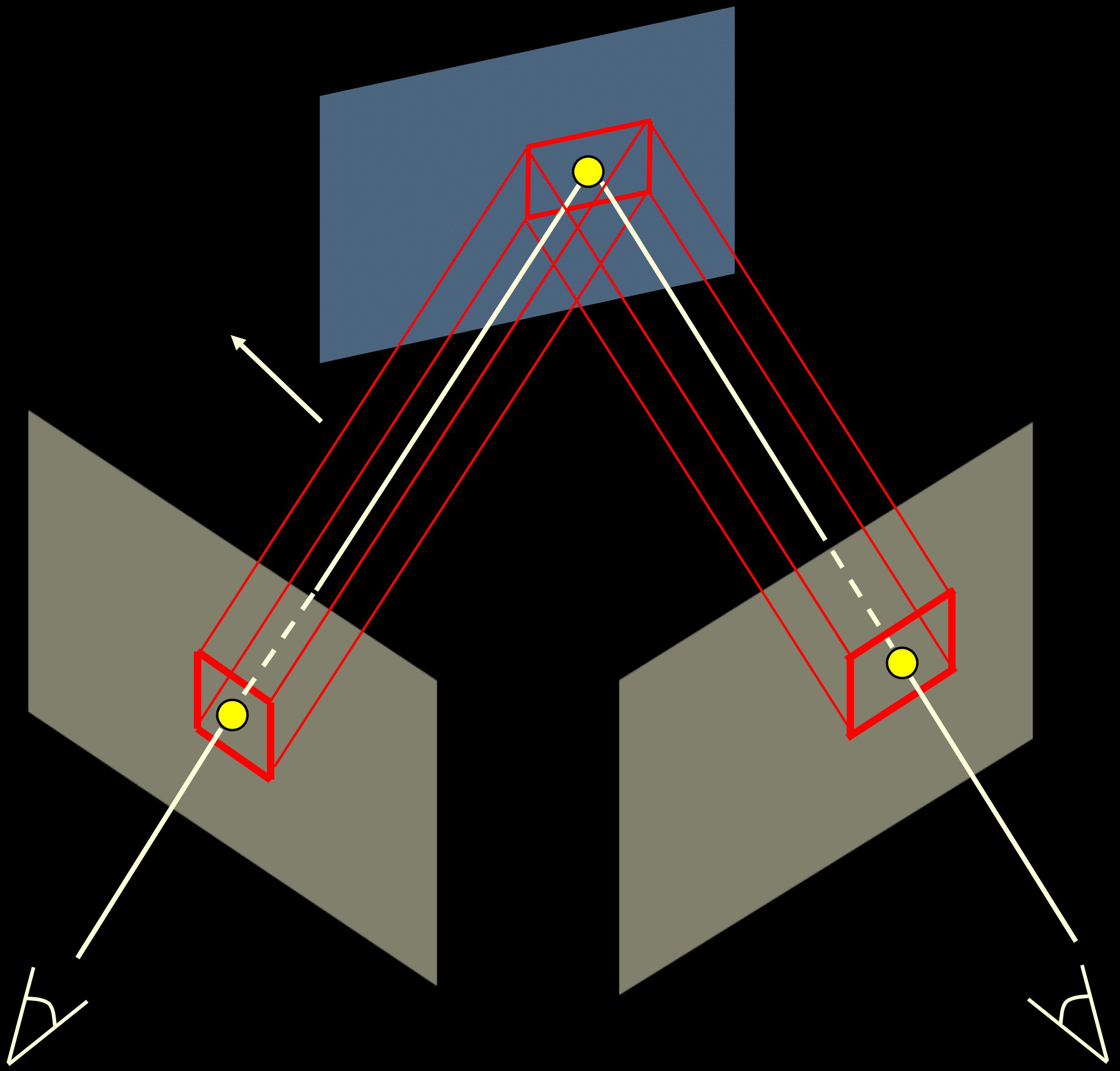
# Spacetime Stereo



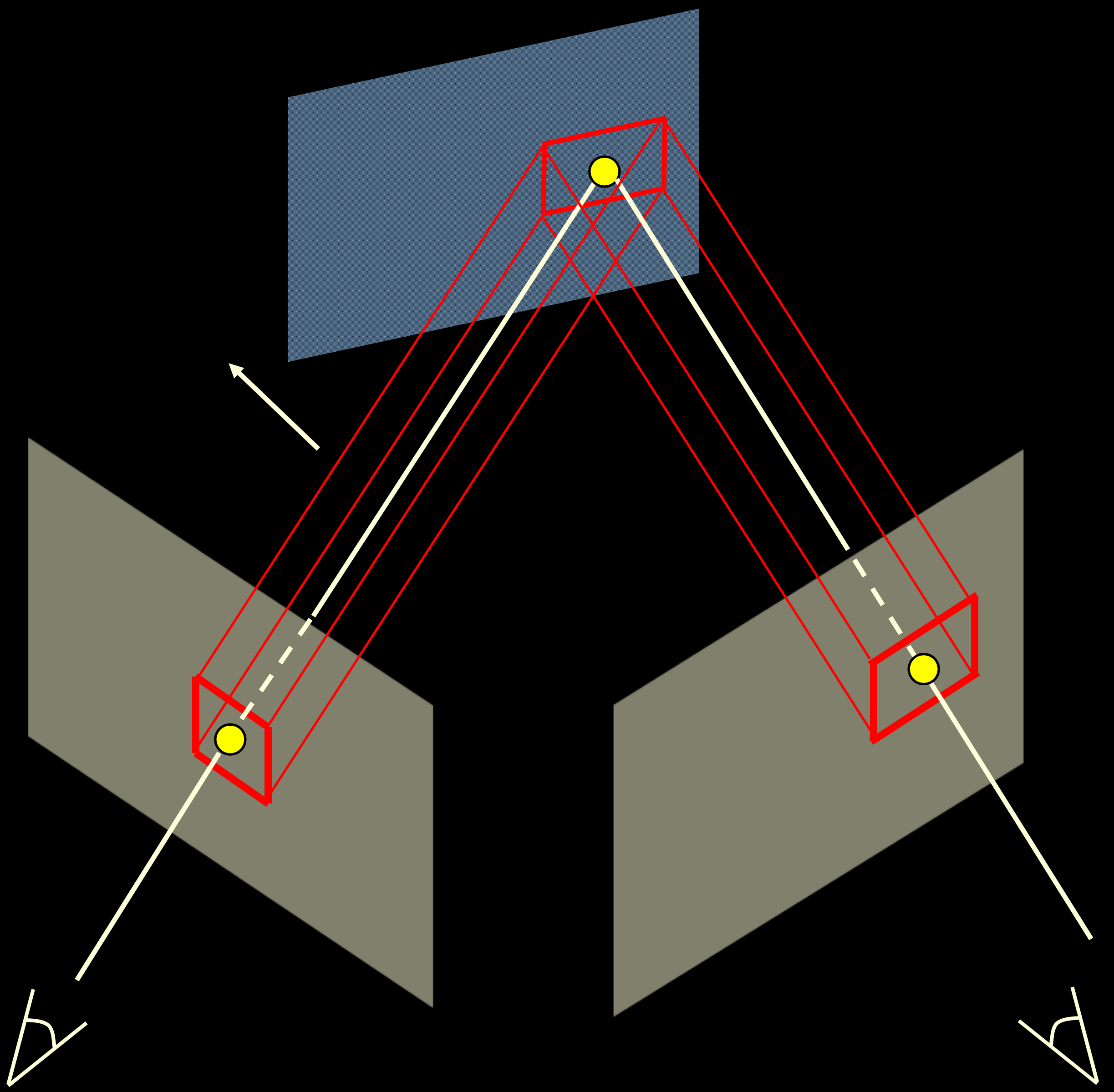
# Spacetime Stereo



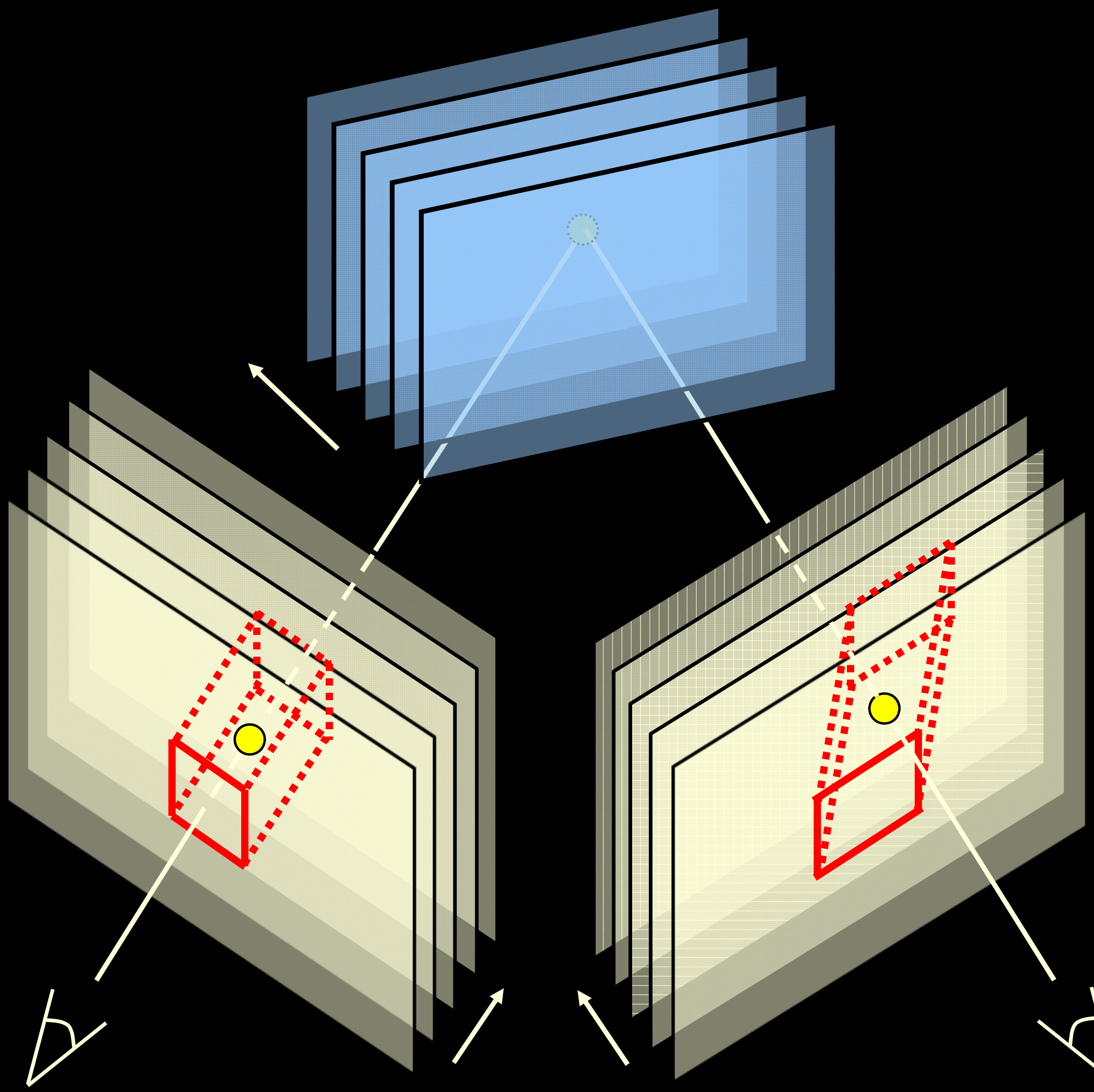
# Spacetime Stereo



# Spacetime Stereo

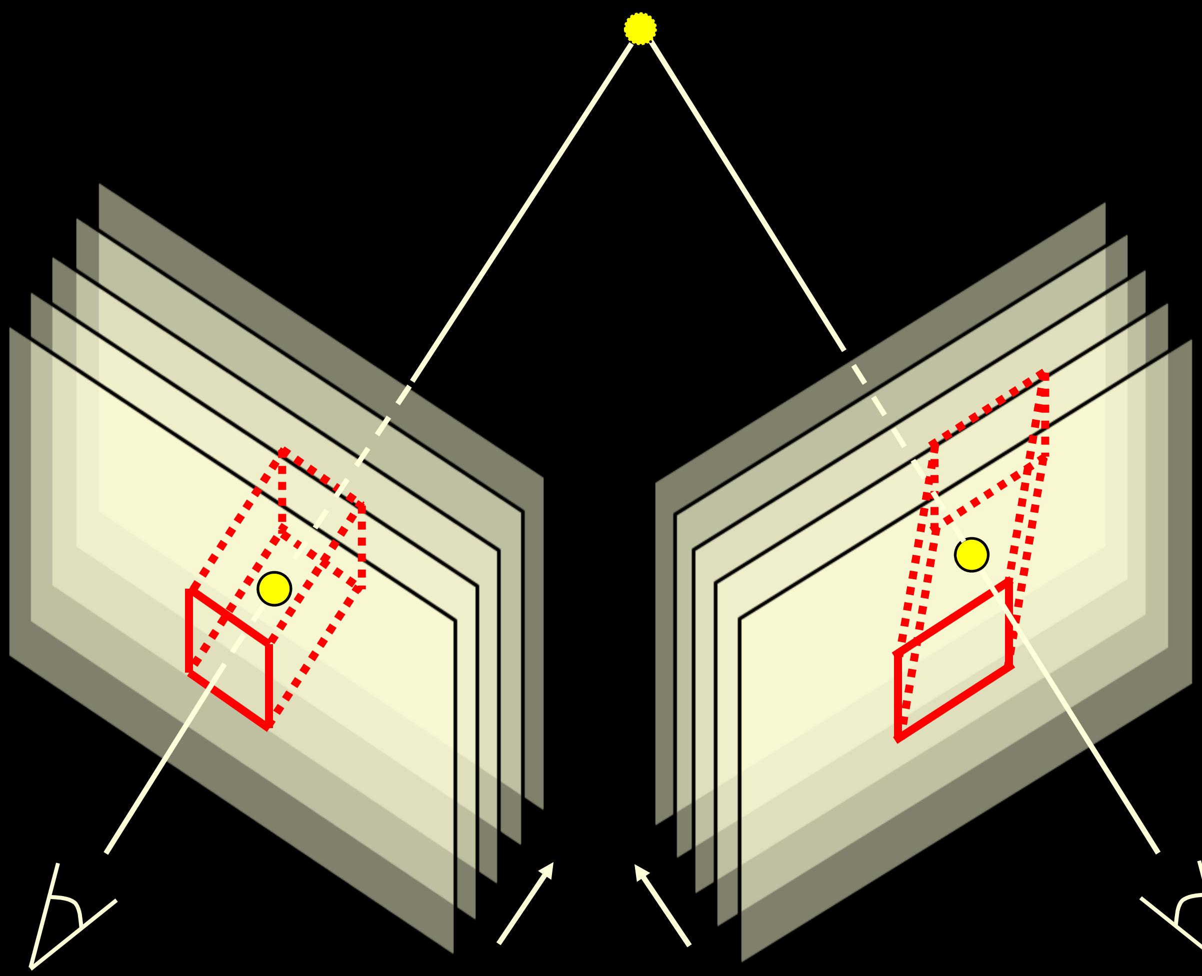


# Spacetime Stereo



# Spacetime Stereo

Given Videos $Left(x, y, t)$ and $Right(x, y, t)$



# Spacetime Stereo

**Given Videos** $Left(x, y, t)$ and $Right(x, y, t)$

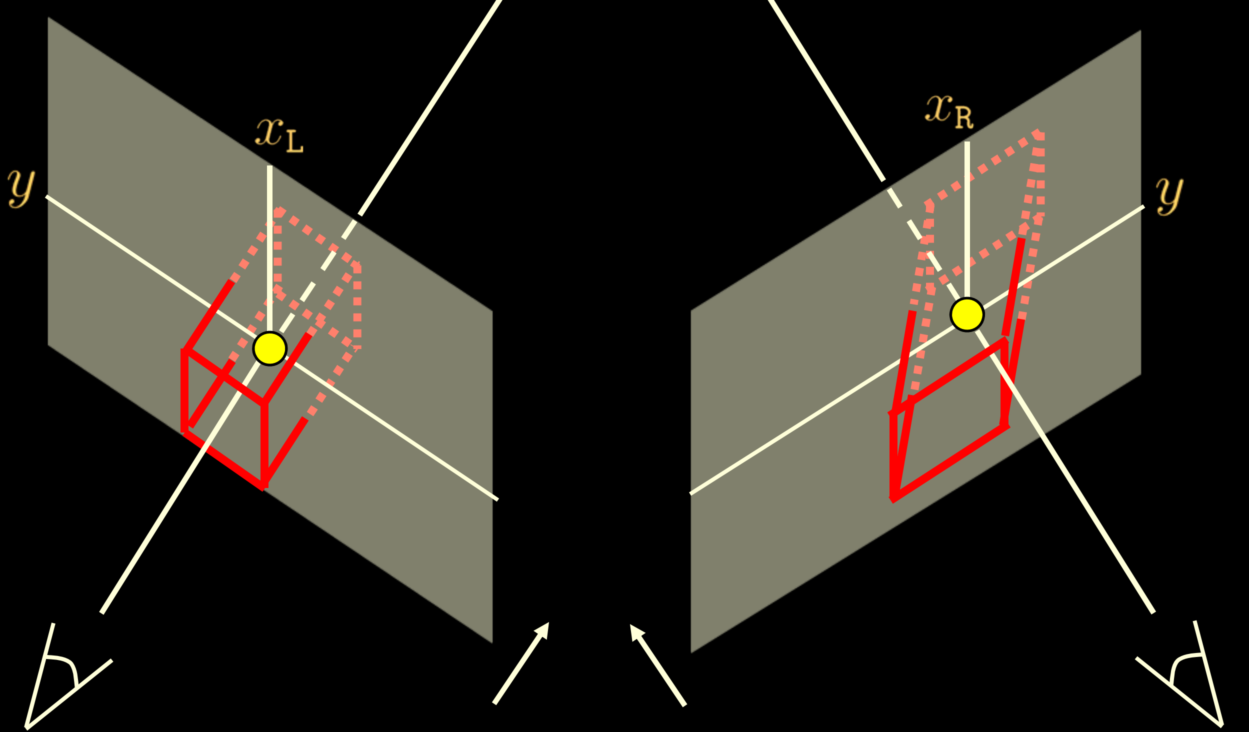
**Goal** Correspondence $\stackrel{\text{def}}{=} (x_L, y, t) \longrightarrow (x_R, y, t)$

Disparity $\stackrel{\text{def}}{=} x_L - x_R$

**Assumption** Locally Linear Disparity Model

$\Downarrow$

Affine Volume Deformation: $\{Disparity, Stretch, Shear\}$



# Spacetime Stereo

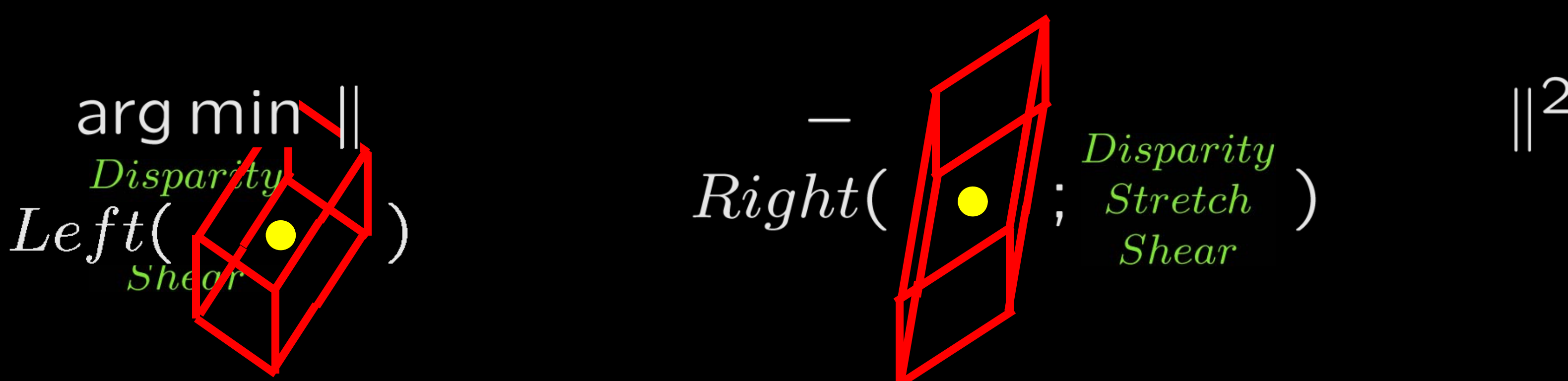
Given Videos $Left(x, y, t)$ and $Right(x, y, t)$

Goal $Disparity \stackrel{\text{def}}{=} x_{\mathrm{L}} - x_{\mathrm{R}}$

Assumption Locally Linear Disparity Model

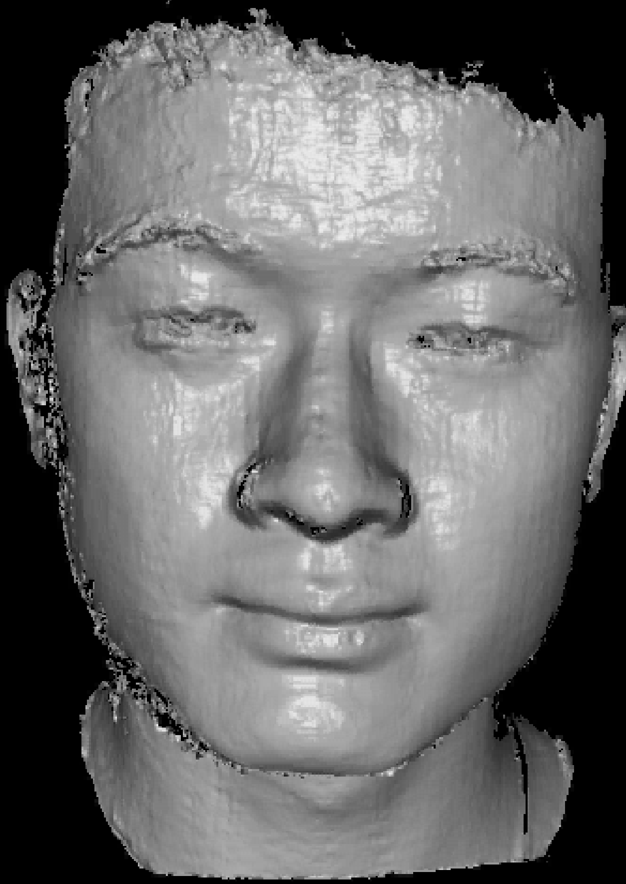
$\Downarrow$

Affine Volume Deformation: $\{Disparity, Stretch, Shear\}$

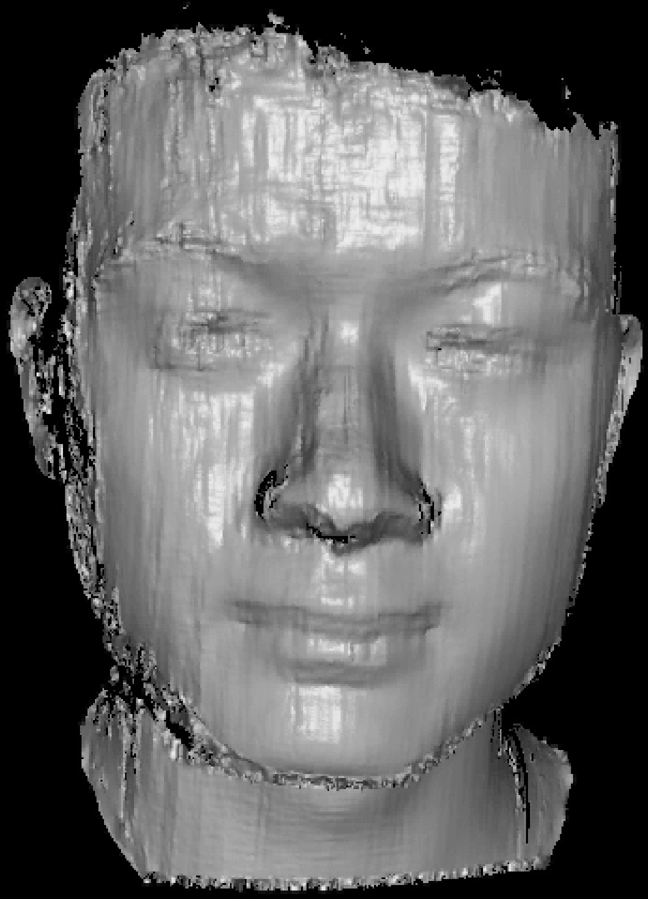


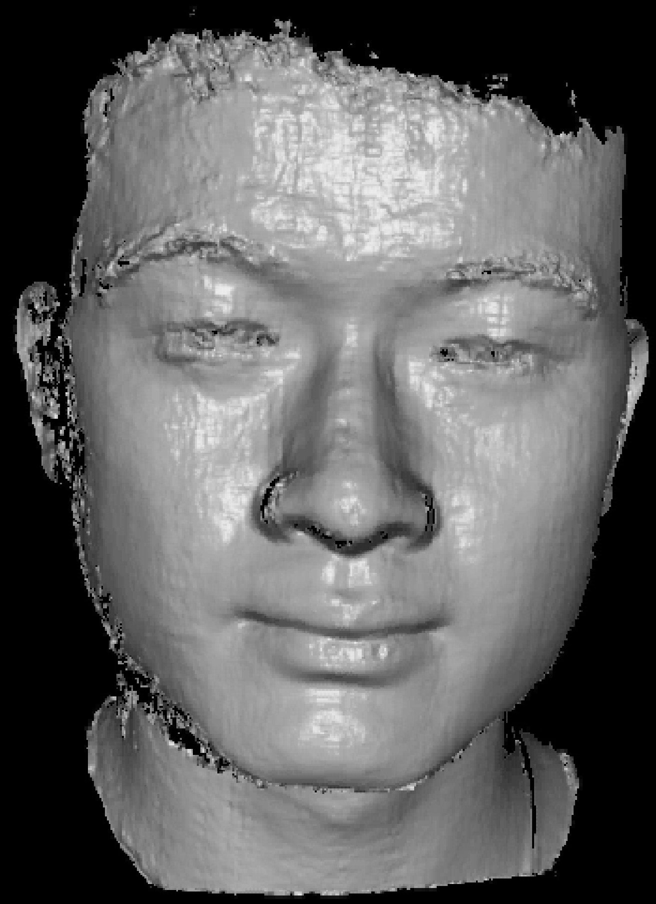

Gauss-Newton Solution, [Zhang *et al.* CVPR2003]
[Davis *et al.* CVPR2003]

# Spacetime Stereo



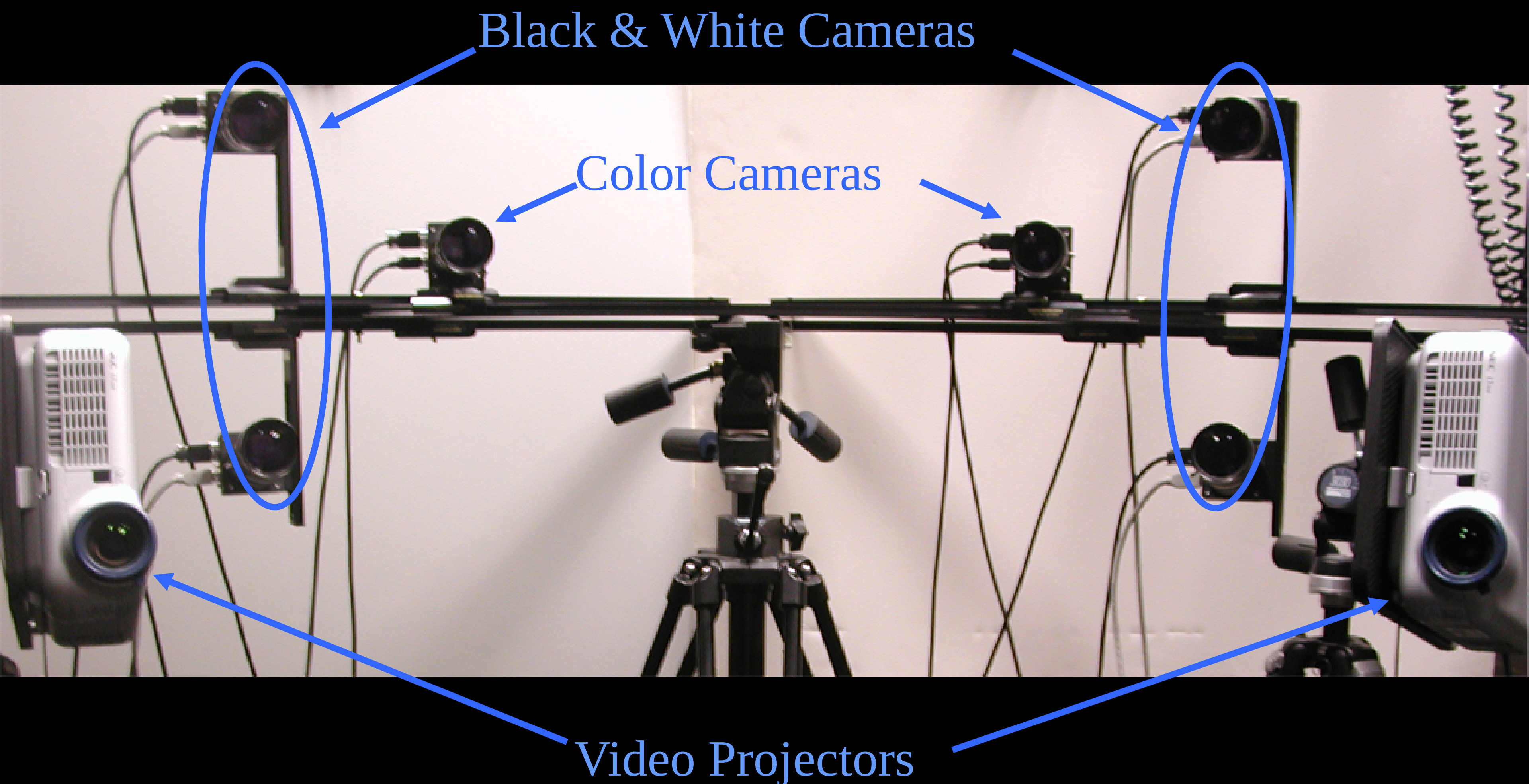
# Frame-by-Frame vs. Spacetime Stereo





Spatially More Accurate
Temporally More Stable

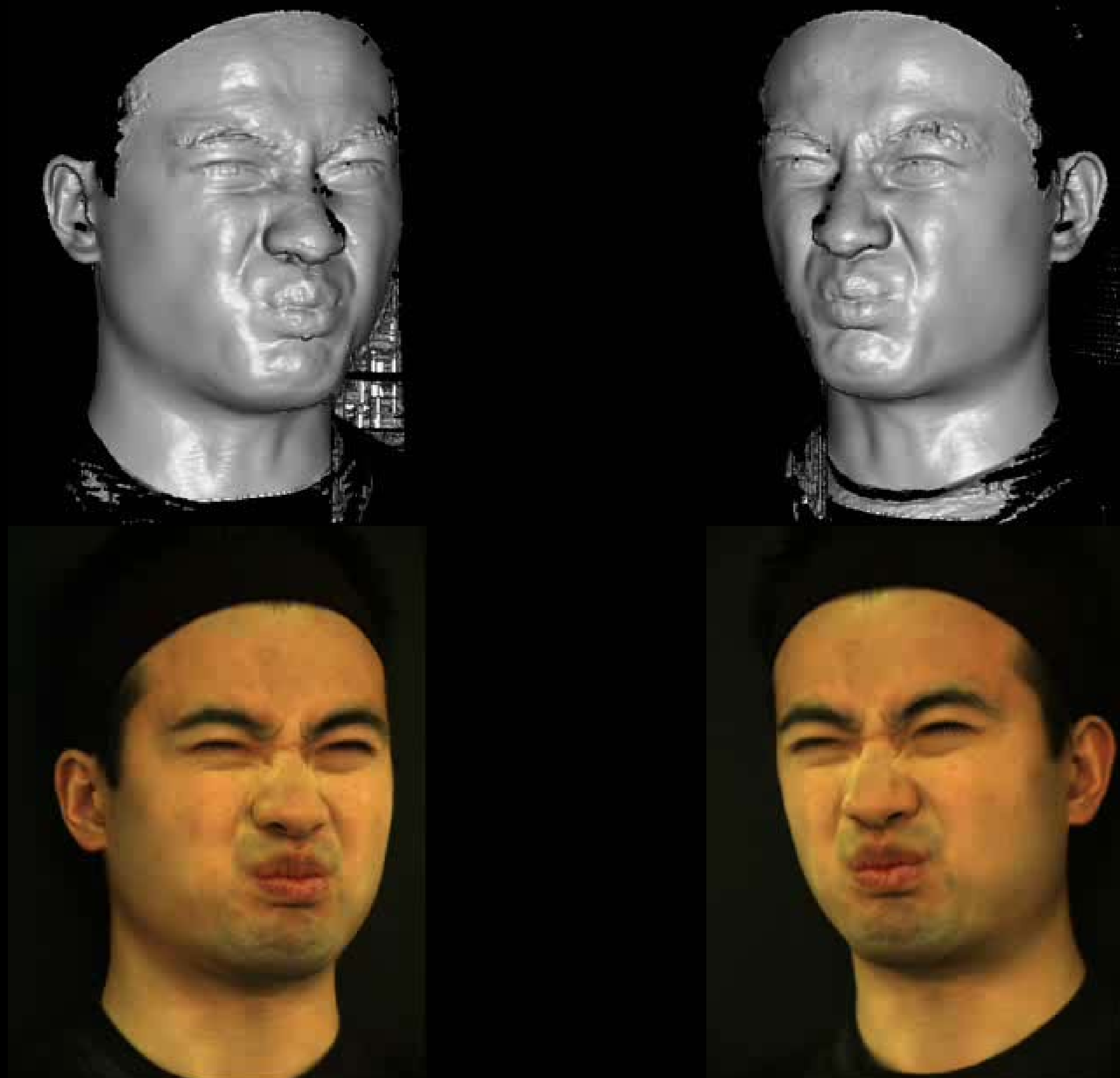
# Spacetime Face Capture System



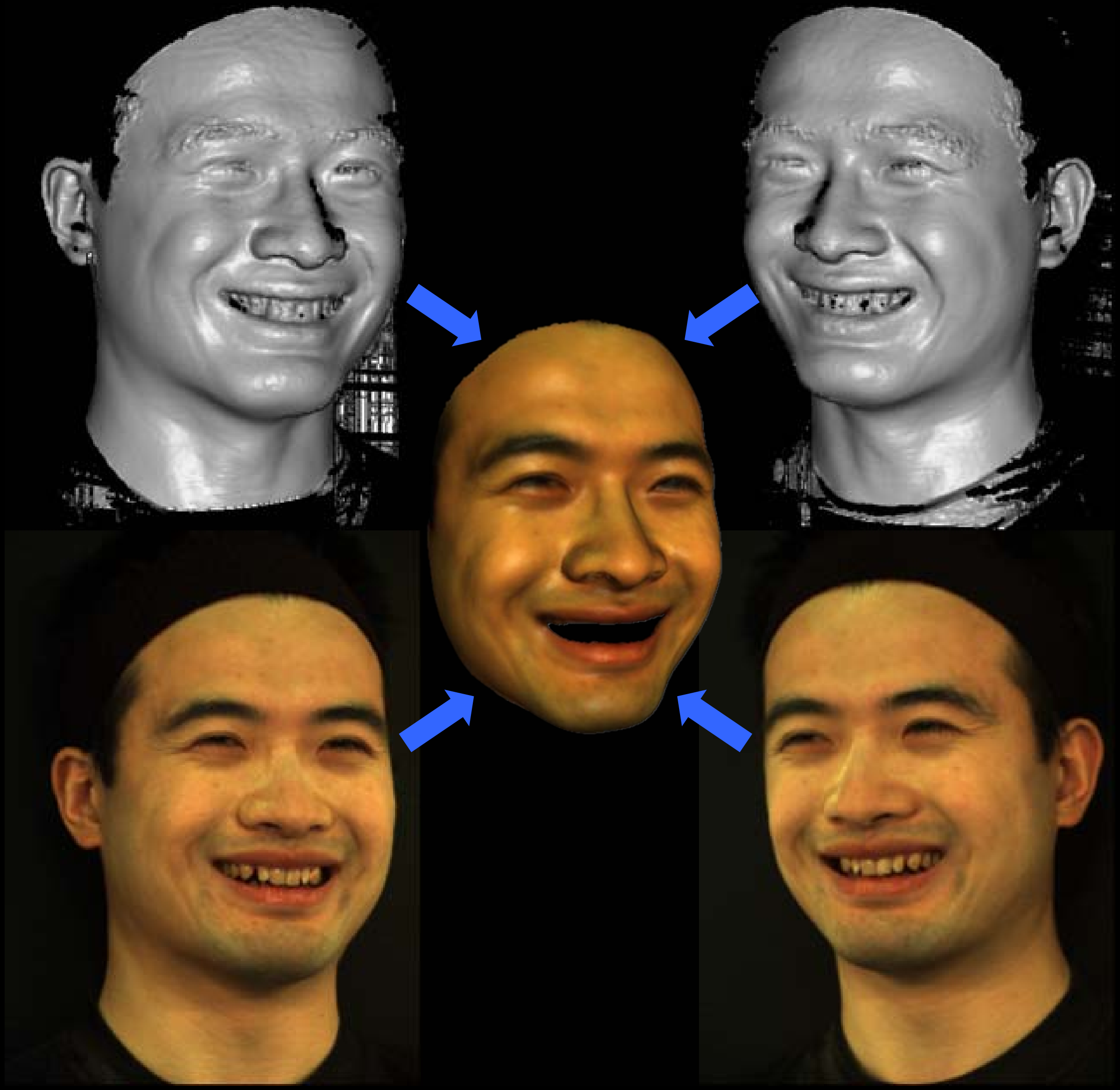
# System in Action

# Input Videos (640×480, 60fps)

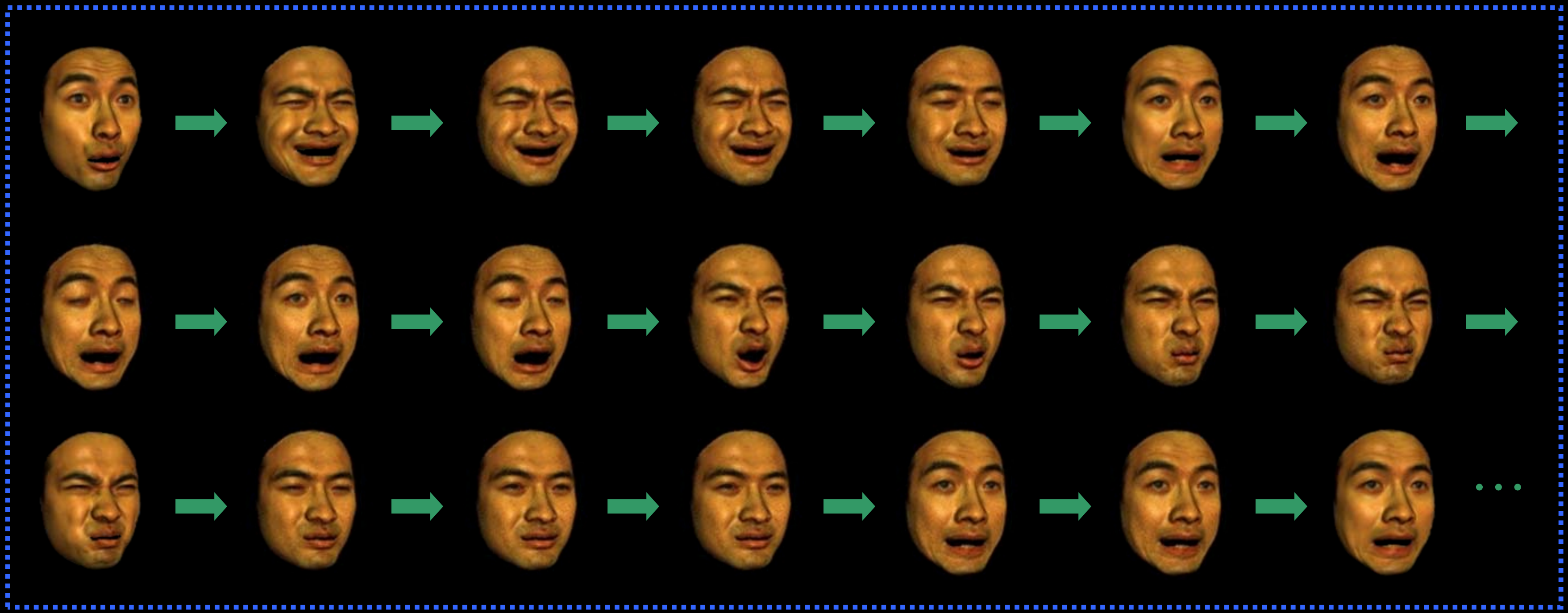
# Spacetime Stereo Reconstruction



# Creating a Face Database

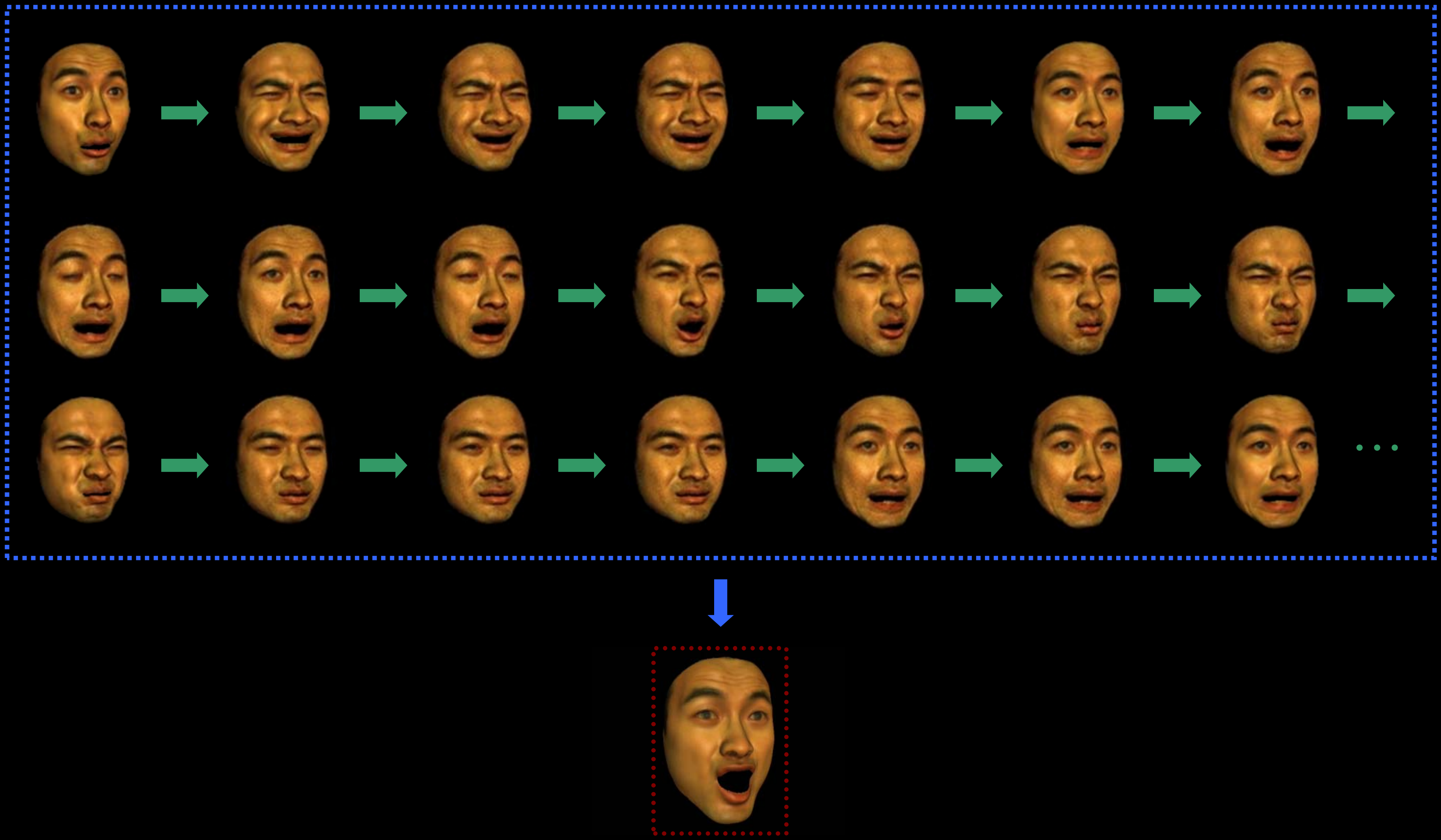


# Creating a Face Database



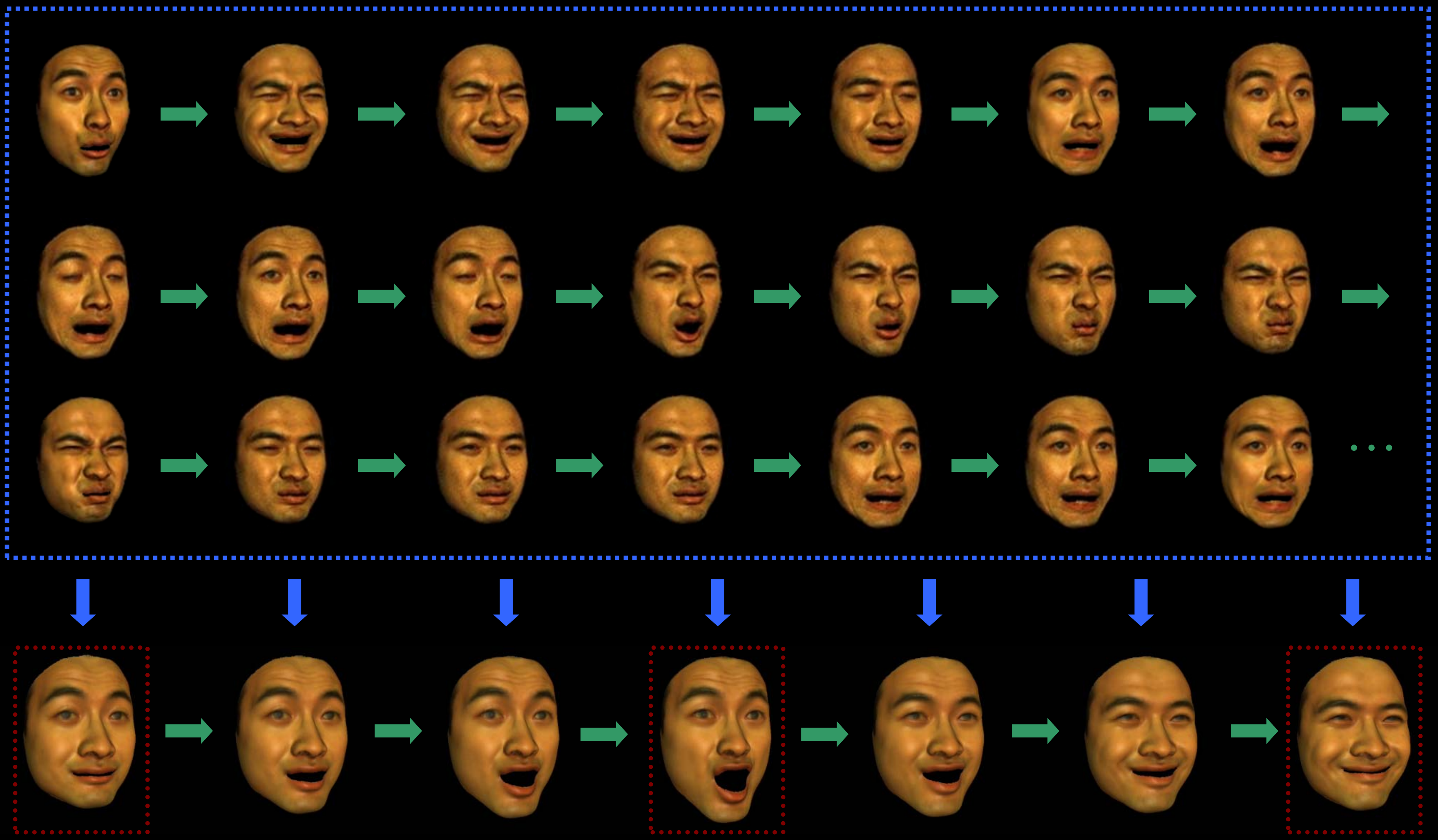
[Zhang *et al.* SIGGRAPH'04]

# Application 1: Expression Synthesis



[Zhang *et al*. SIGGRAPH'04]

# Application 2: Facial Animation



[Zhang *et al.* SIGGRAPH'04]

# Keyframe Animation