图像拼接

周晓巍
浙江大学CAD&CG实验室

Some slides adapted from Noah Snavely, Lingqi Yan

Panorama (全景图)

360度VR

360度VR

全景相机

GoPro odyssey

图像拼接

VR眼镜

Huawei

How to combine two images?

How to combine two images?

What is image warping?
How to compute it?

Part I
Image Warping
图像变形

Image Warping

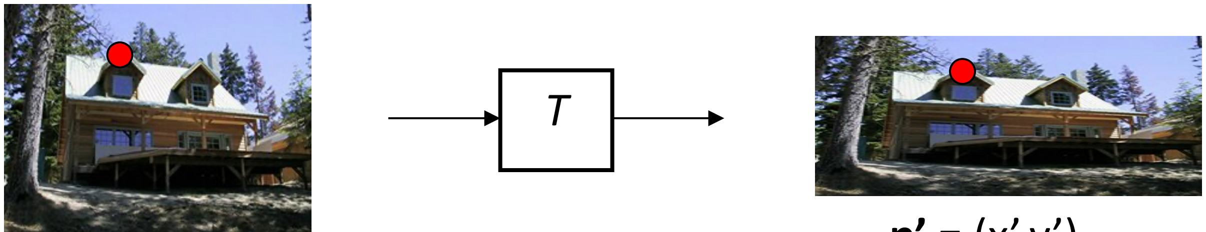
- **image filtering**: change *intensity* of image

f  $\longrightarrow$ g 

- image warping: change *shape* of image

f  $\longrightarrow$ g 

Parametric (global) warping



p = (x,y) **p’** = (x’,y’)

- Transformation T is a coordinate transformation:

$$p' = T(p)$$

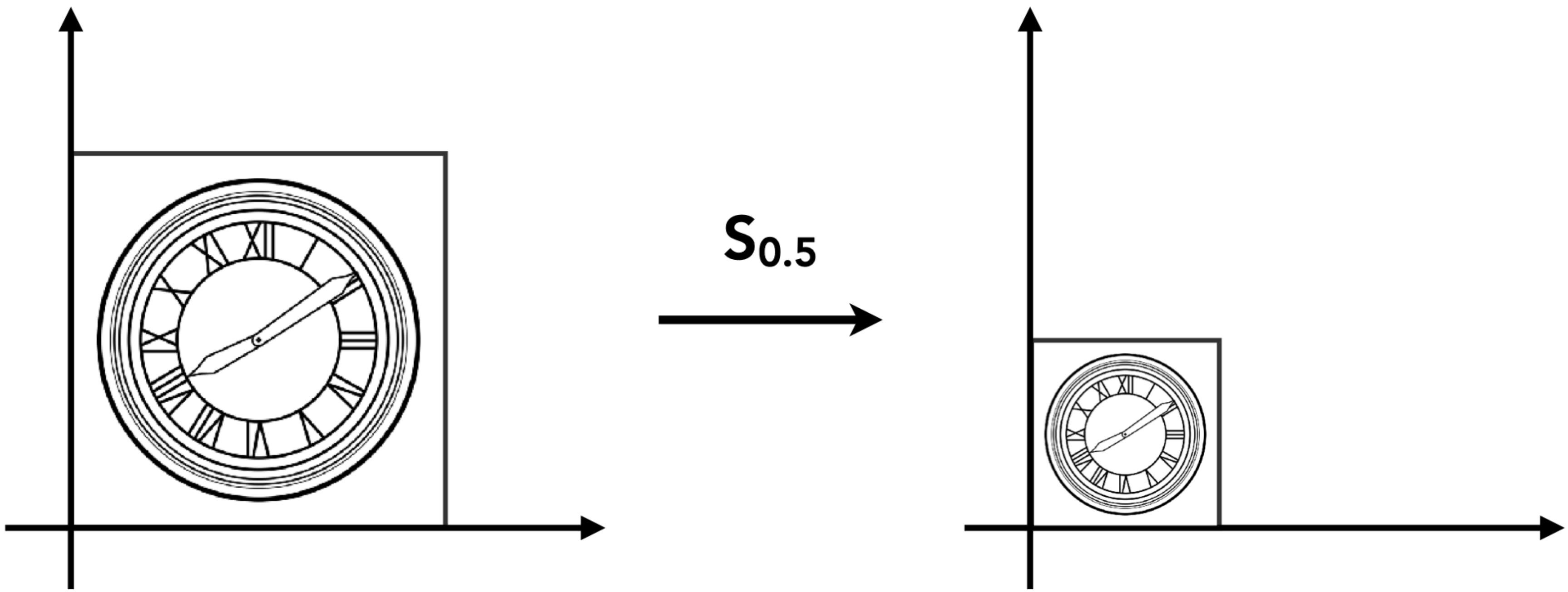
- Examples:

translation

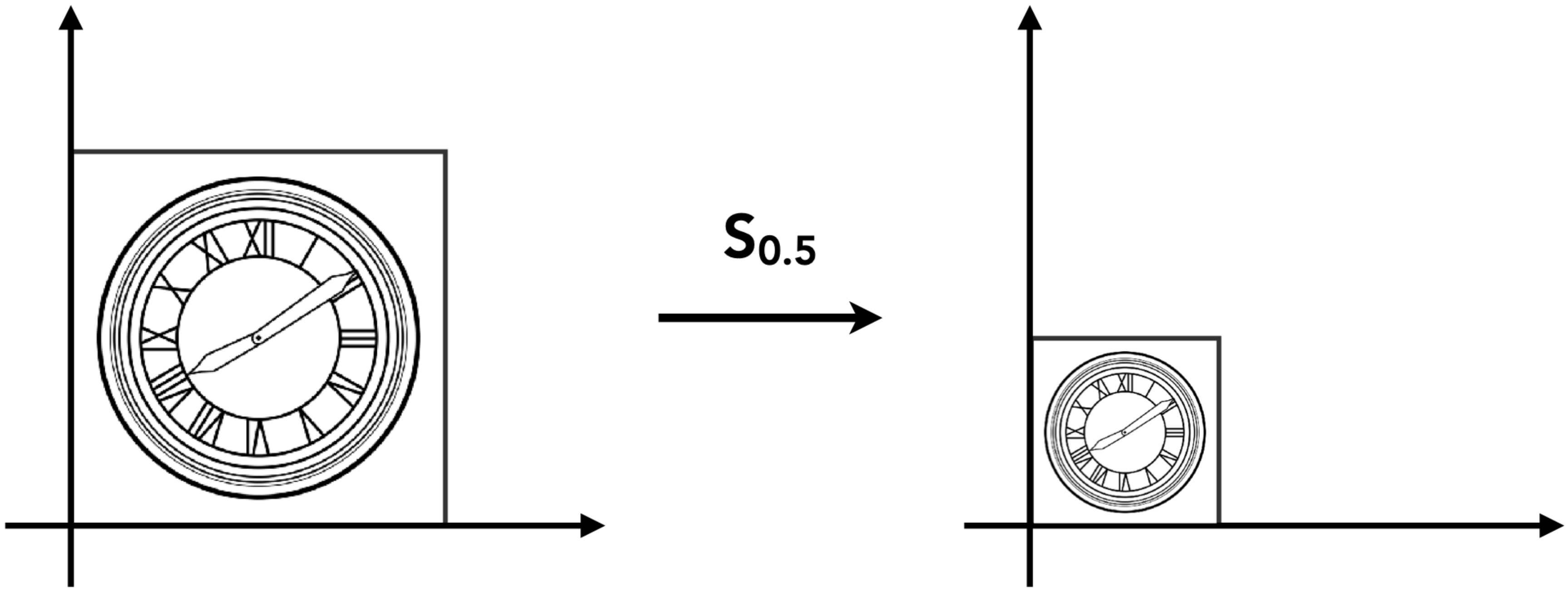
rotation

aspect

Scale



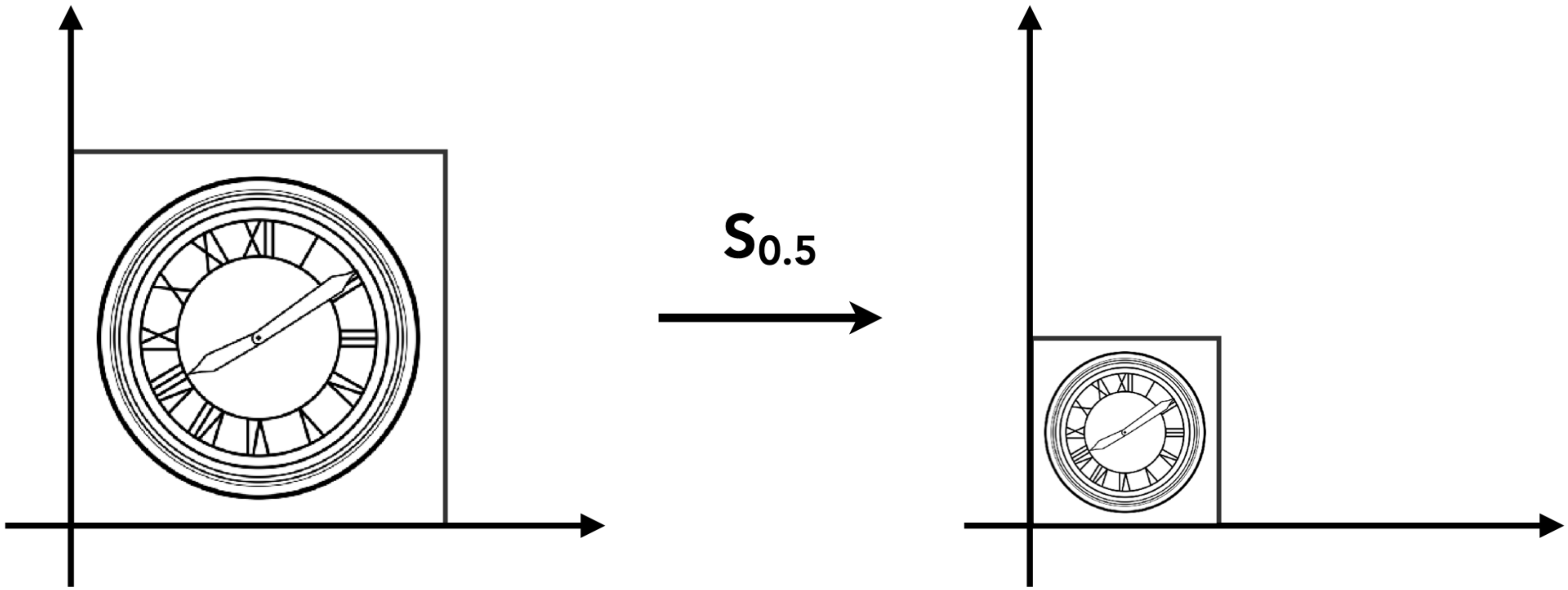
Scale Transform



$$x' = sx$$

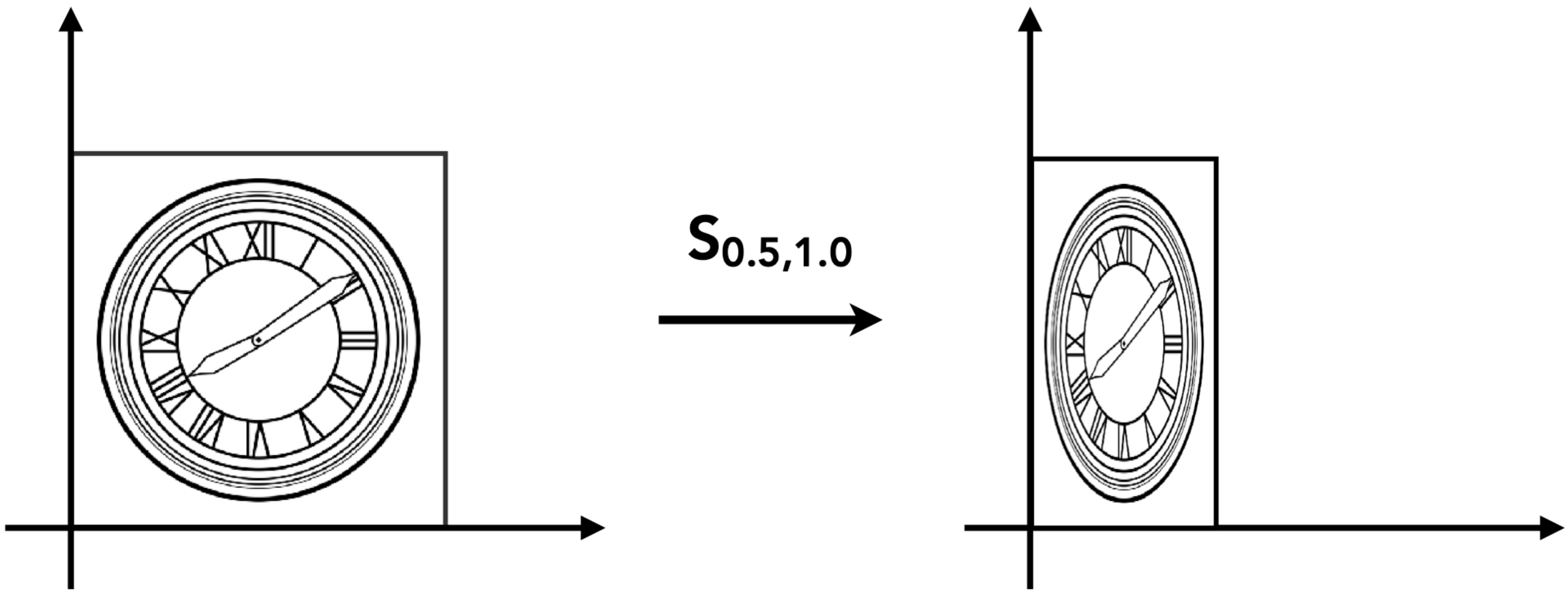
$$y' = sy$$

Scale Matrix



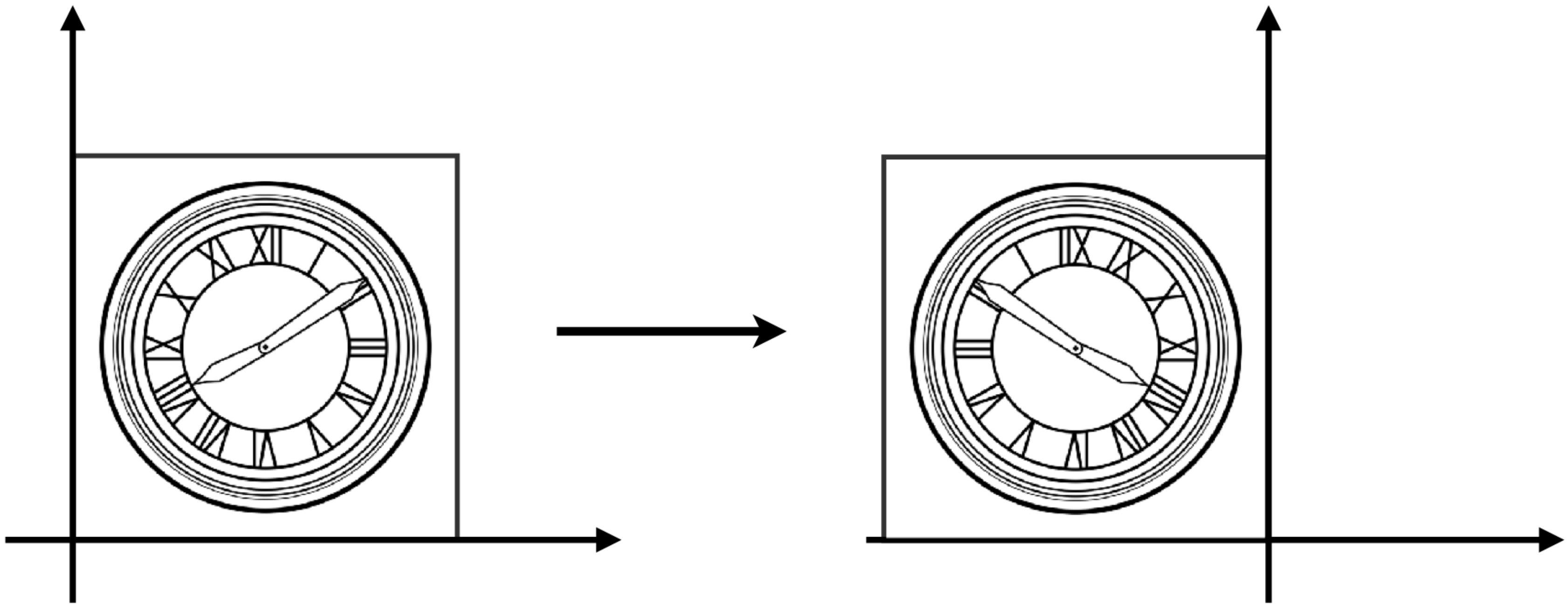
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale (Non-Uniform)



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Reflection Matrix

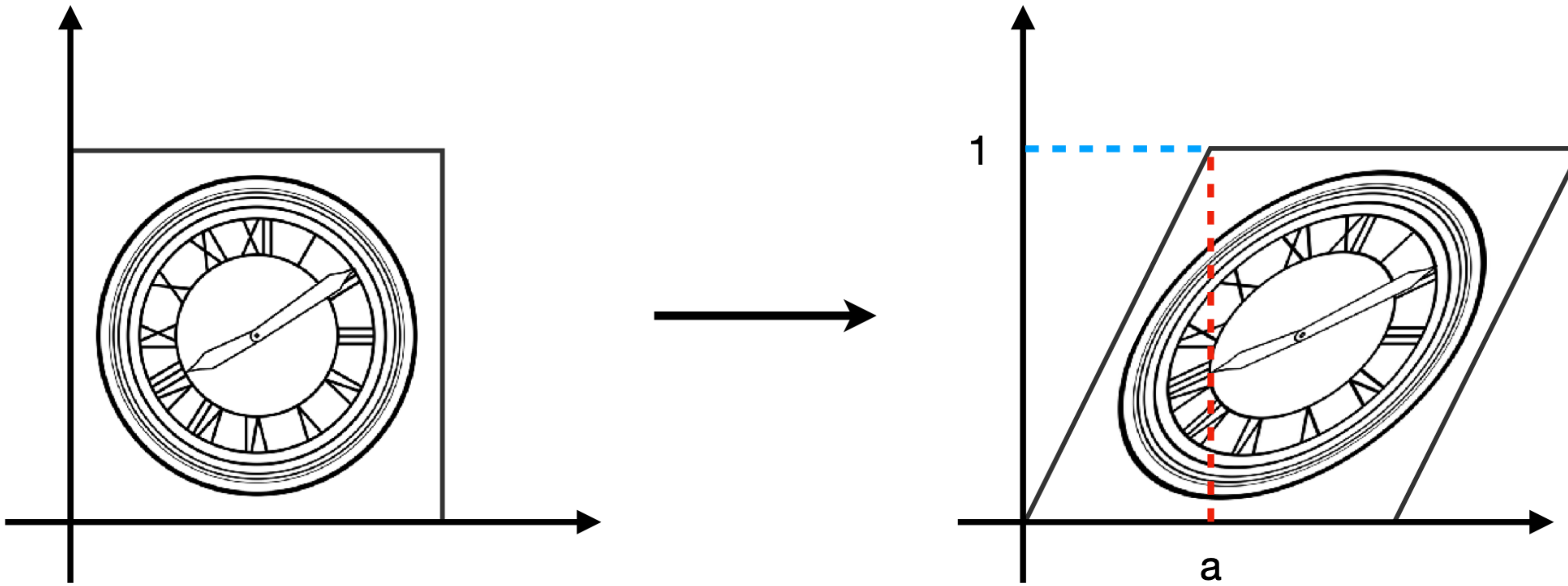


Horizontal reflection:

$$x' = -x$$
$$y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear Matrix



Hints:

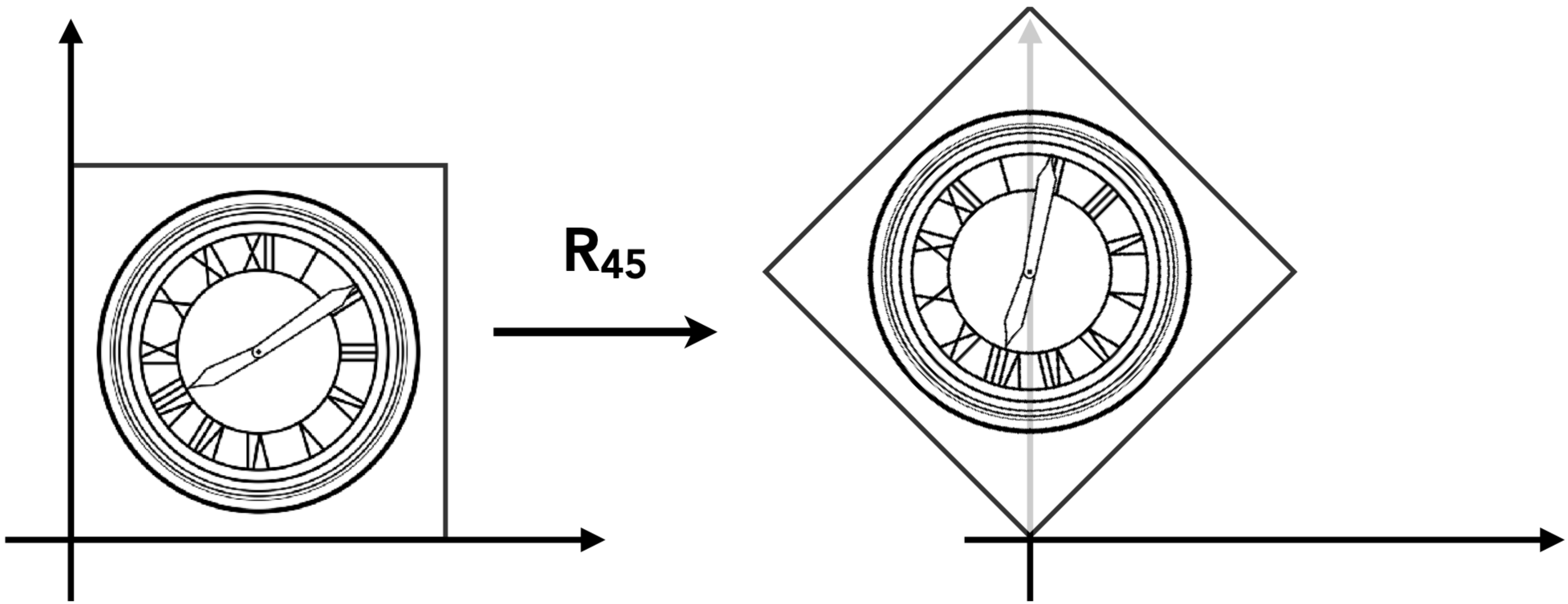
Horizontal shift is 0 at y=0

Horizontal shift is a at y=1

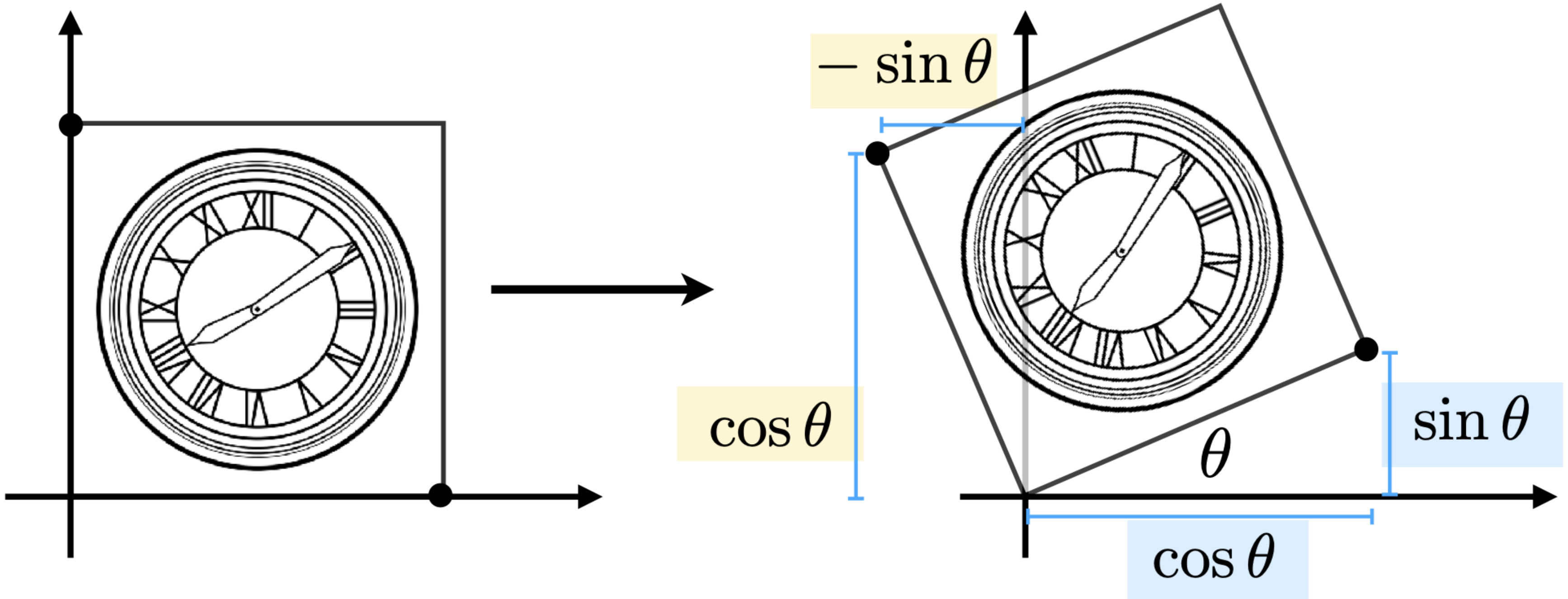
Vertical shift is always 0

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate (about the origin (0, 0), CCW by default)



Rotation Matrix



$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Linear Transforms = Matrices

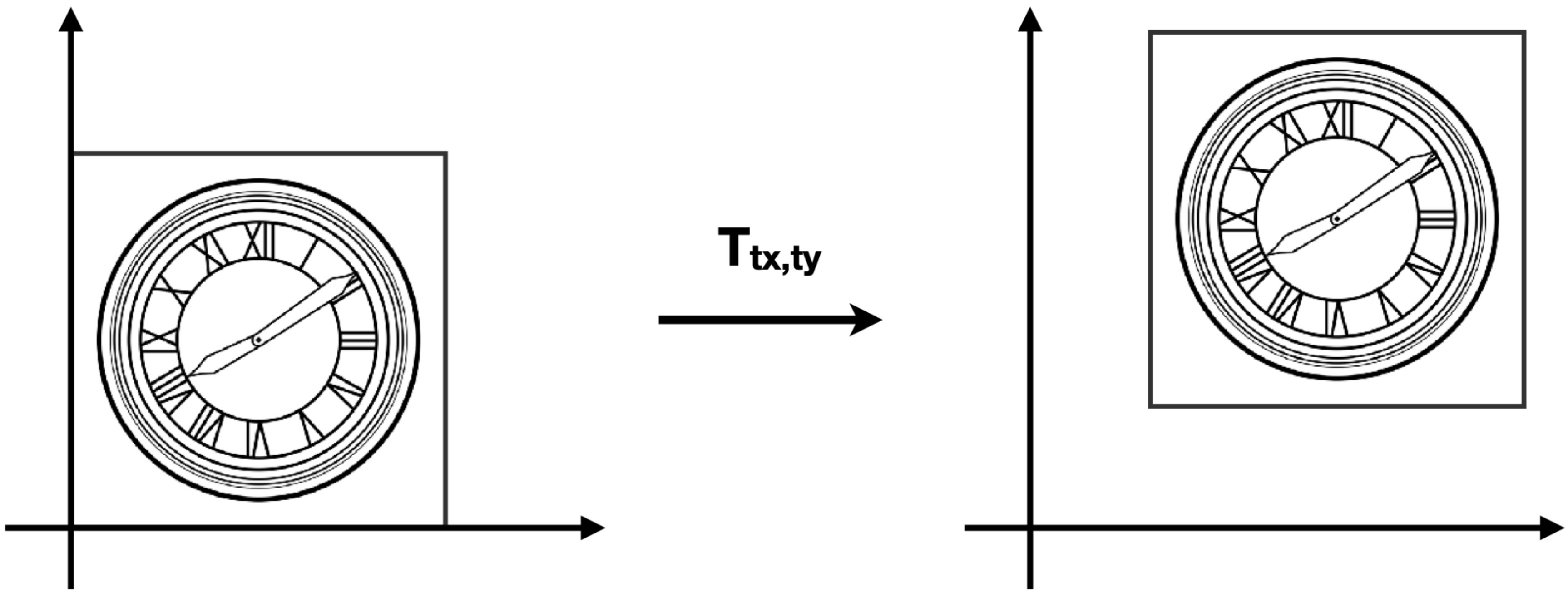
(of the same dimension)

$$x' = a\,x + b\,y$$
$$y' = c\,x + d\,y$$

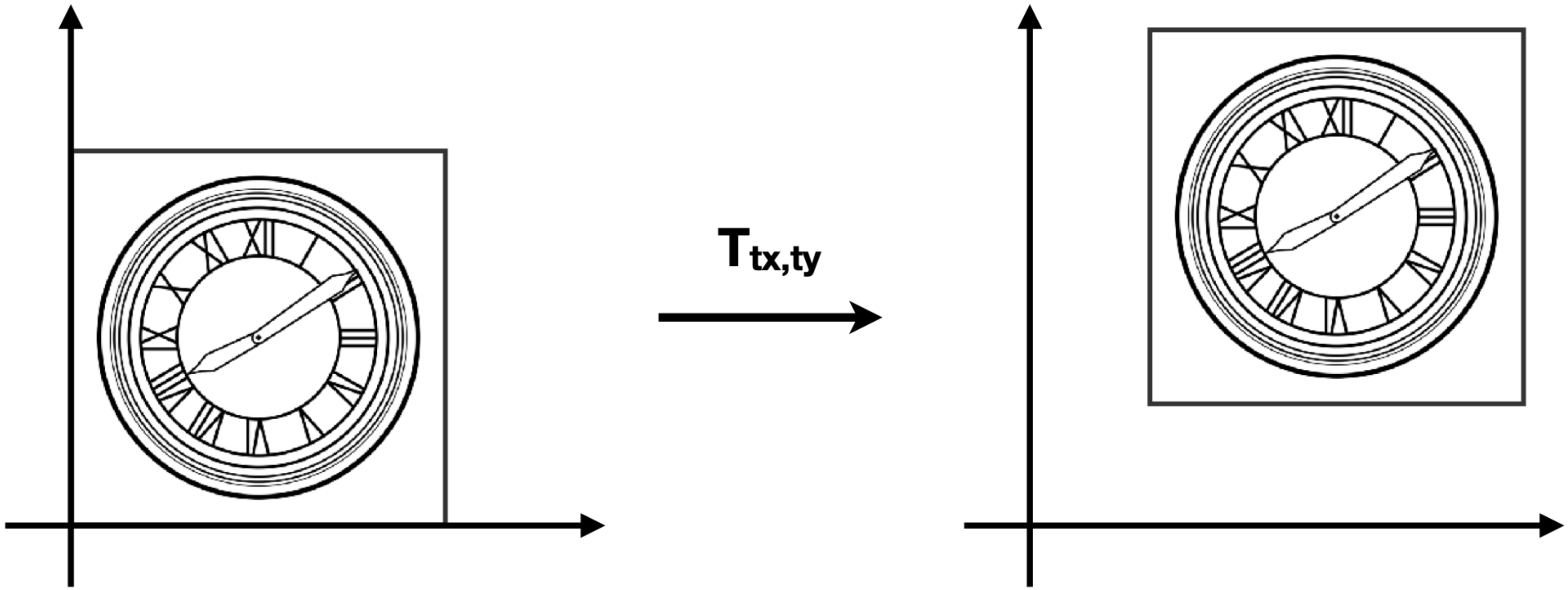
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{x}' = \mathbf{M}\,\mathbf{x}$$

Translation



Translation??



$$x' = x + t_x$$
$$y' = y + t_y$$

Why Homogeneous Coordinates

- Translation cannot be represented in matrix form

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

(So, translation is NOT linear transform!)

- But we don’t want translation to be a special case

- Is there a unified way to represent all transformations? (and what’s the cost?)

Solution: Homogenous Coordinates

Add a third coordinate (*w-coordinate*)

- 2D point = $(x, y, 1)^T$
- 2D vector = $(x, y, 0)^T$

Matrix representation of translations

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

What if you translate a vector?

Homogenous Coordinates

Valid operation if w-coordinate of result is 1 or 0

- vector + vector = vector
- point – point = vector
- point + vector = point
- point + point = ??

In homogeneous coordinates,

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} \text{ is the 2D point } \begin{pmatrix} x/w \\ y/w \\ 1 \end{pmatrix}, \ w \neq 0$$

2D Transformations

Scale

$$\mathbf{S}(s_x, s_y) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation

$$\mathbf{R}(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Translation

$$\mathbf{T}(t_x, t_y) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Affine Transformations

Affine map = linear map + translation

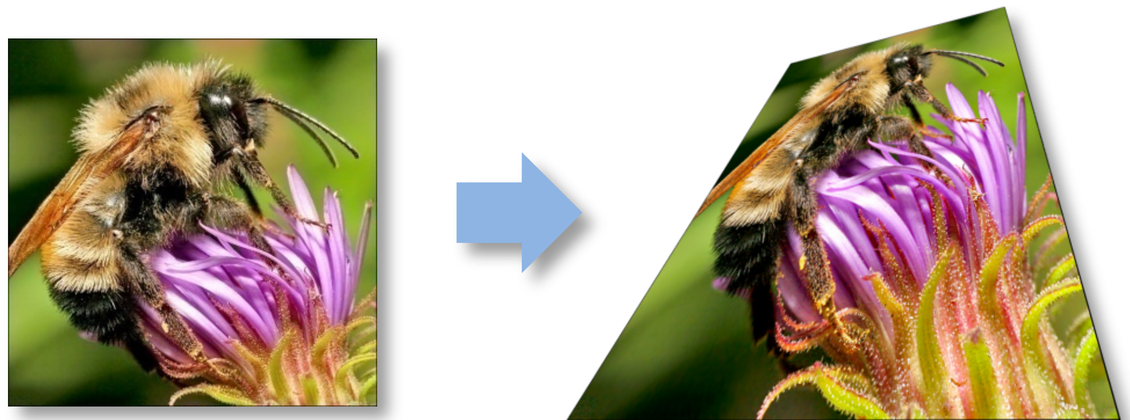
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

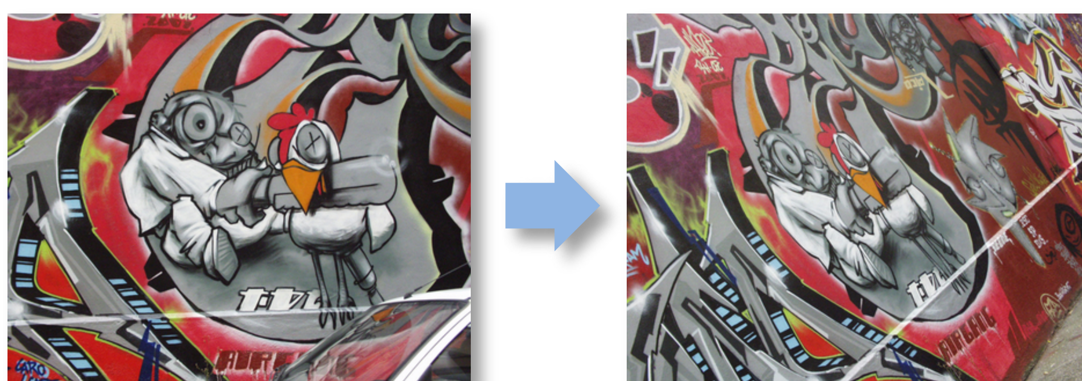
Using homogenous coordinates:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

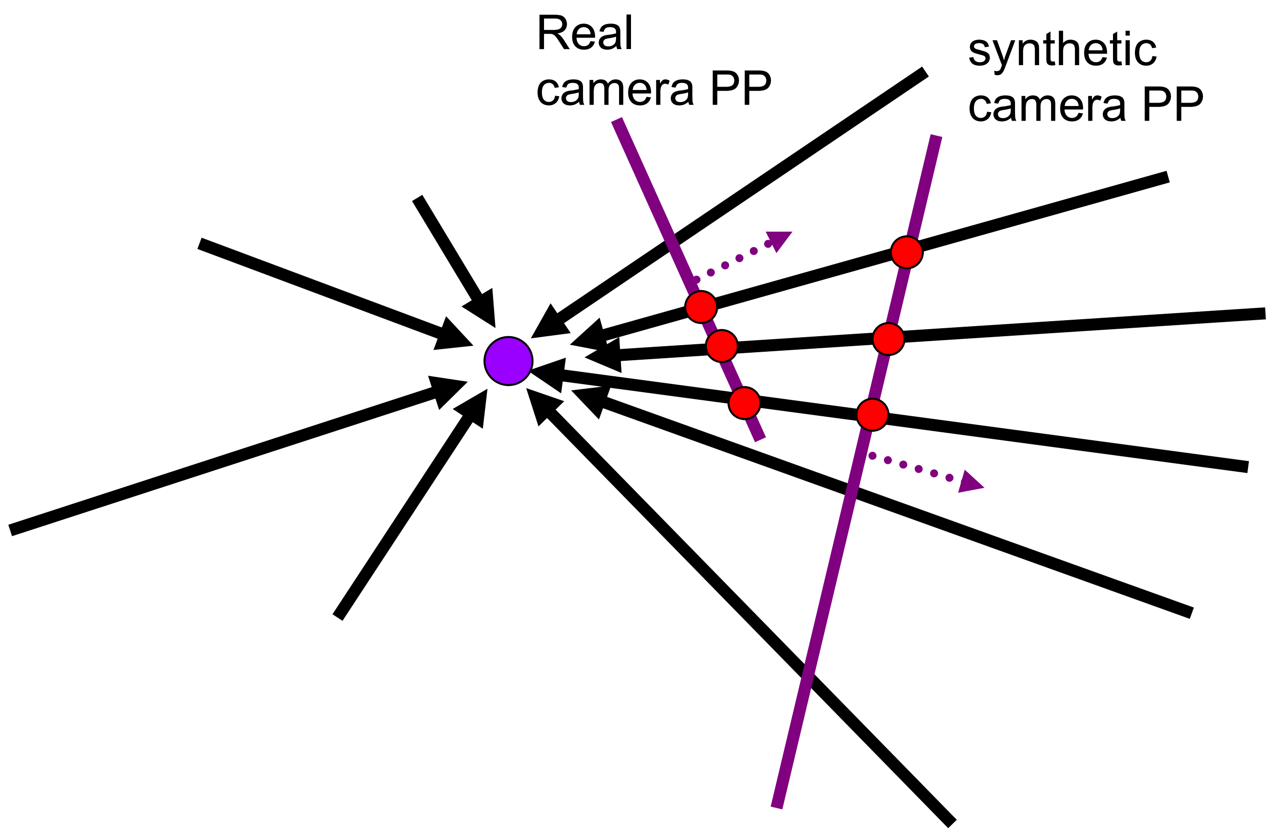
any transformation represented by a 3x3 matrix with last row [0 0 1] we call an *affine* transformation

Projective Transformation (Homography)





Change projection plane (pp)



Can generate any synthetic camera view
as long as it has **the same center of projection**!

Fun with homography

Original image

St.Petersburg
photo by A. Tikhonov

Virtual camera rotations

Projective Transformation (Homography)

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

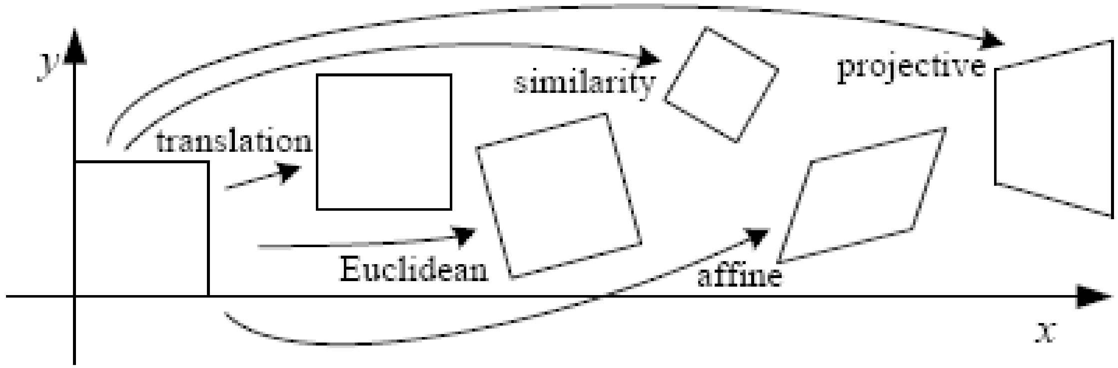
Maybe nonzero

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

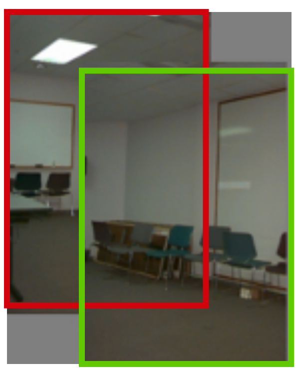
$$y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

- We usually constrain the length of the vector [h_{00} h_{01} … h_{22}] to be 1, which means the degree of freedom is 8

Summary of 2D transformations

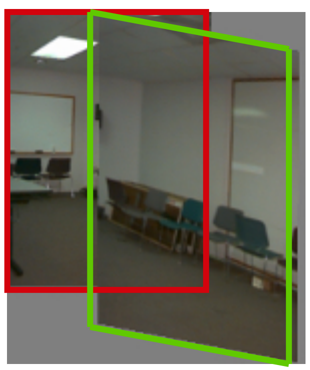


Translation



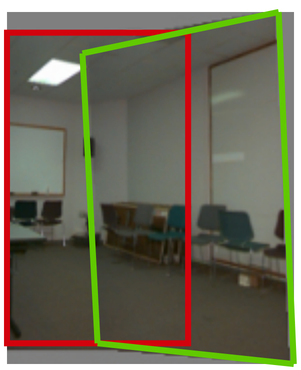
2 unknowns

Affine



6 unknowns

Projective

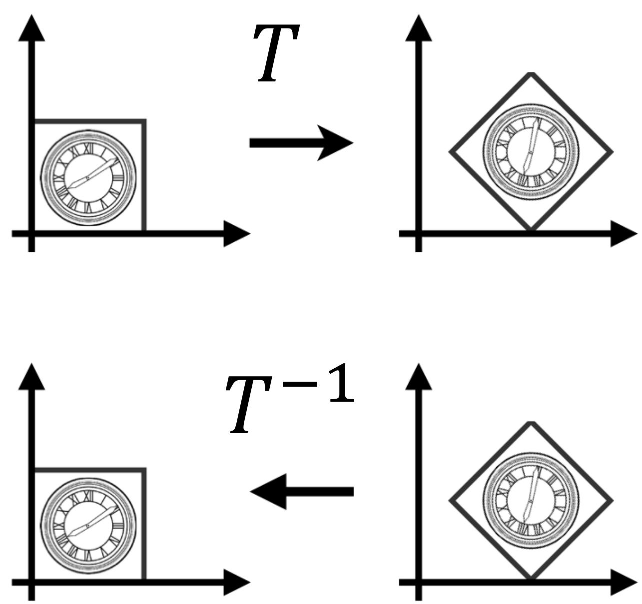


8 unknowns

Inverse Transform

$$T^{-1}$$

T^{-1} **is the inverse of transform** T **in both a matrix and geometric sense**



Implementing image warping

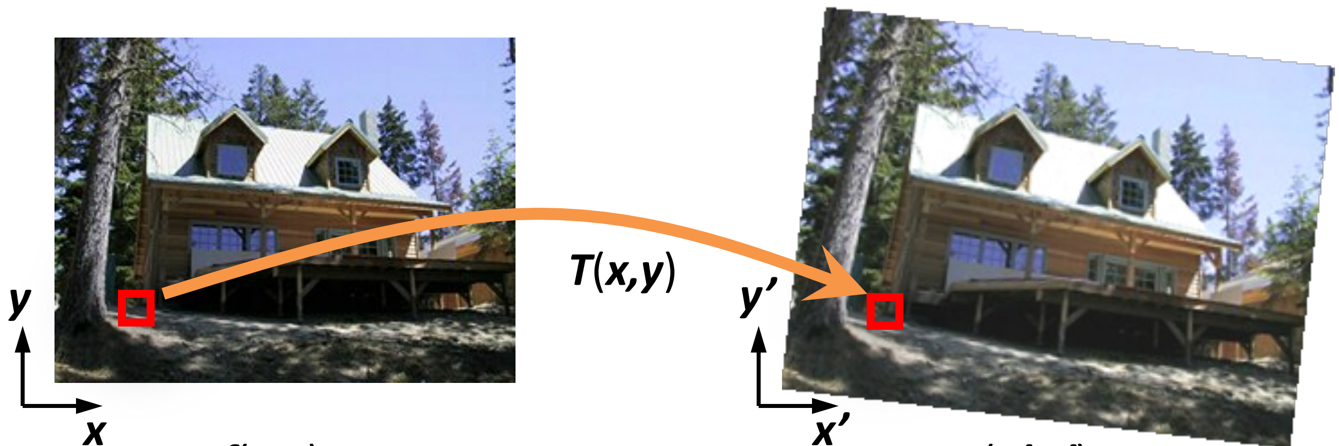
- Given a coordinate transform $(\boldsymbol{x'},\boldsymbol{y'}) = \boldsymbol{T}(\boldsymbol{x},\boldsymbol{y})$ and a source image $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})$, how do we compute an transformed image $\boldsymbol{g}(\boldsymbol{x'},\boldsymbol{y'}) = \boldsymbol{f}(\boldsymbol{T}(\boldsymbol{x},\boldsymbol{y}))$?

$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})$

$\boldsymbol{g}(\boldsymbol{x'},\boldsymbol{y'})$

Forward Warping

- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $(\boldsymbol{x'},\boldsymbol{y'}) = \boldsymbol{T}(\boldsymbol{x},\boldsymbol{y})$ in $\boldsymbol{g}(\boldsymbol{x'},\boldsymbol{y'})$



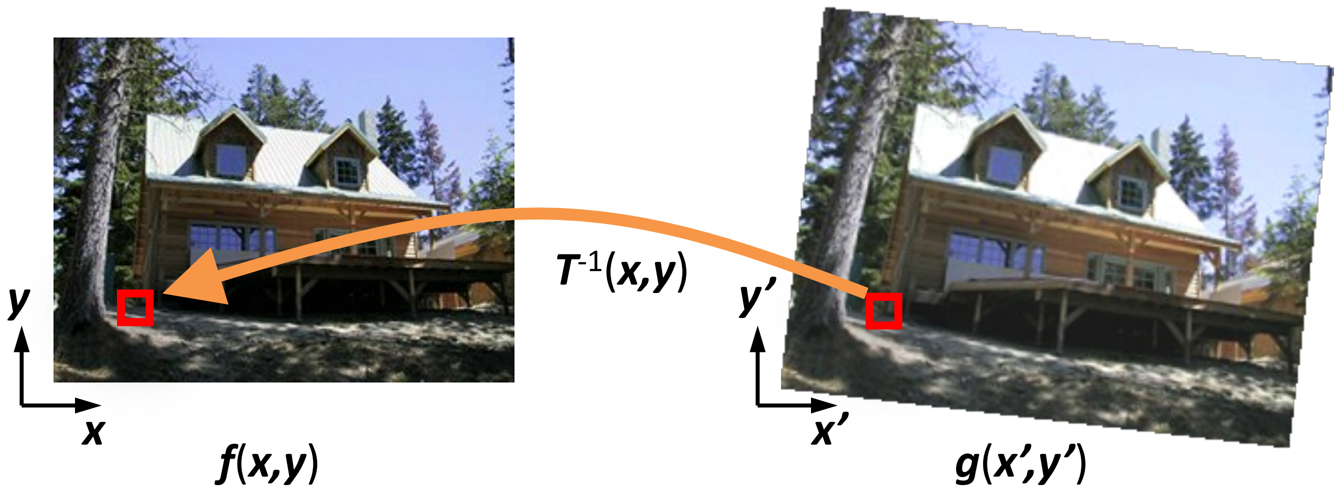
$f(x,y)$ $g(x',y')$

Forward Warping

- What if pixel lands “between” pixels?

Inverse Warping

- Get each pixel $\boldsymbol{g}(\boldsymbol{x'},\boldsymbol{y'})$ from its corresponding location $(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{T}^{-1}(\boldsymbol{x},\boldsymbol{y})$ in $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})$



Inverse Warping

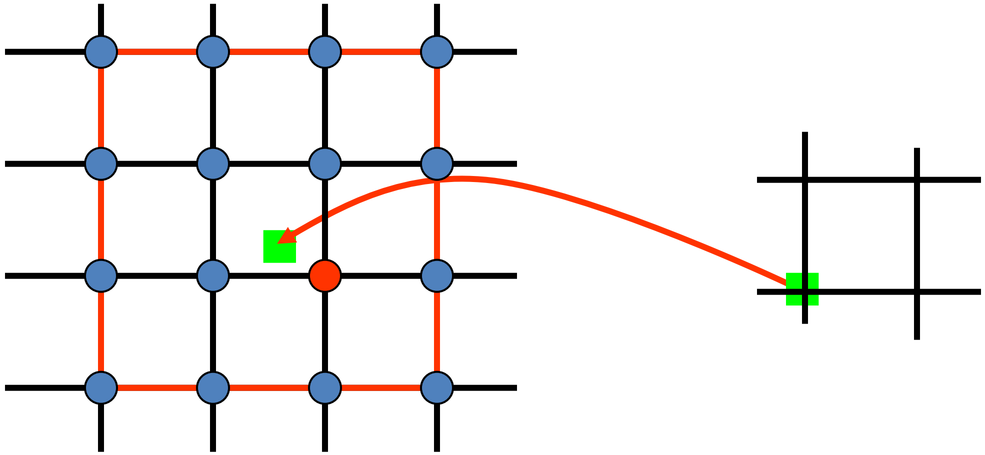
- What if pixel lands “between” pixels?

Answer: interpolate color values from neighboring pixels

Interpolation

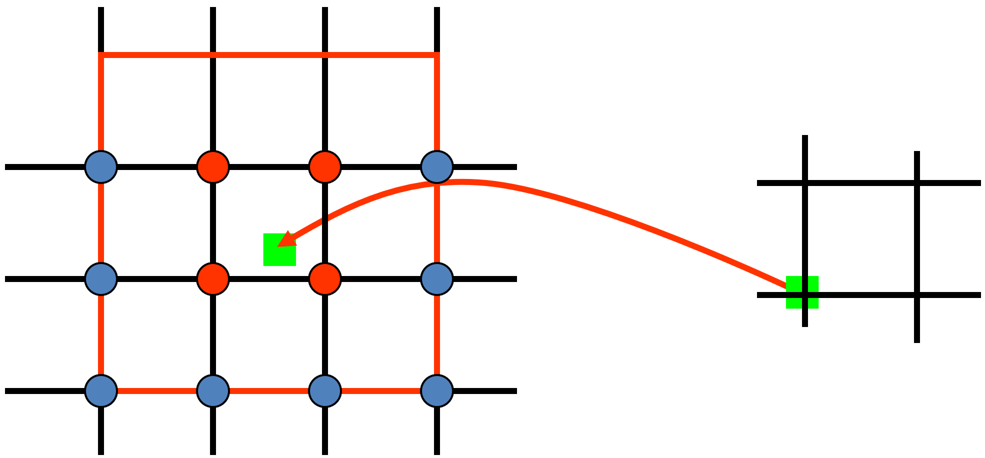
Nearest neighbor

- Copies the color of the pixel with the closest integer coordinate



Interpolation

Weighted sum of four neighboring pixels



Interpolation

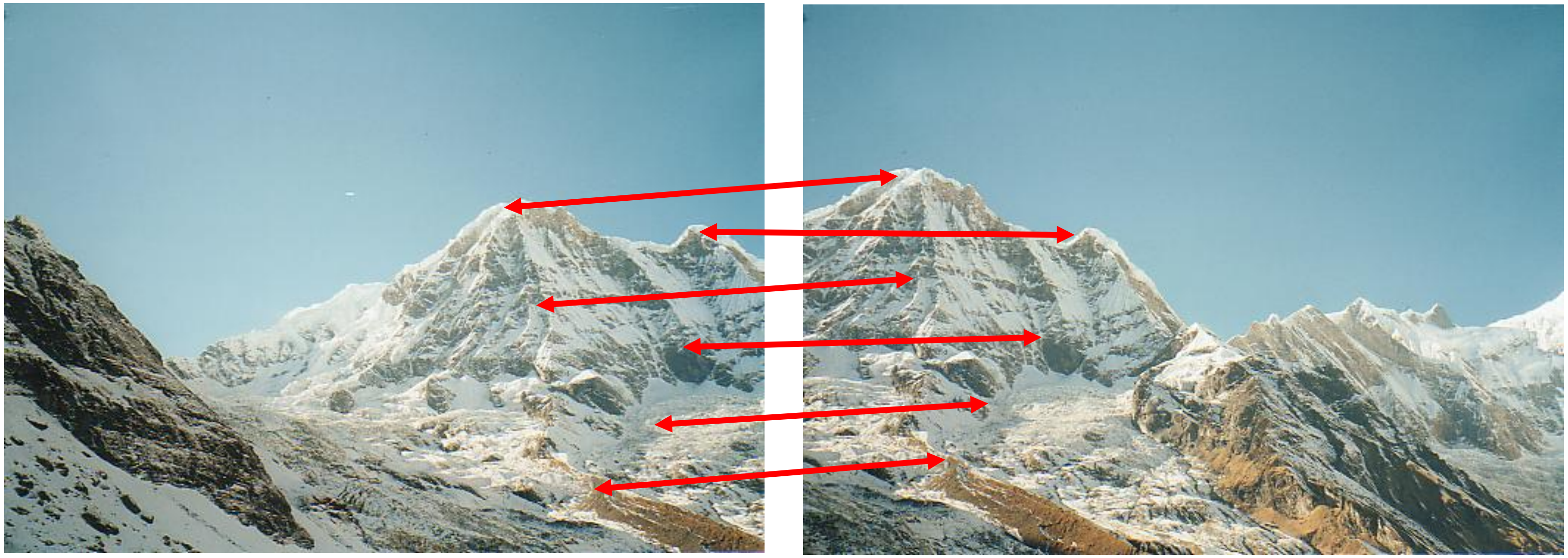
- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic
  - sinc

Questions?

Part II
Image Stitching

How to compute transformation?

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \cong T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



1. Image matching (each match gives an equation)
2. Solve T from the obtained matches

Affine transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

- How many unknowns?
- How many equations per match?
- How many matches do we need?

Affine transformations

- For each match, we have

$$x' = ax + by + c$$
$$y' = dx + ey + f$$

- Matrix form

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} \mathrm{a} \\ b \\ c \\ d \\ e \\ f \end{bmatrix}$$

Affine transformations

- For n matches

$$
\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_1 & y_1 & 1 \\
x_2 & y_2 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_2 & y_2 & 1 \\
 & & & \vdots & & \\
x_n & y_n & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & x_n & y_n & 1
\end{bmatrix}
\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}
=
\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}
$$

$$
\underset{2n \times 6}{\mathbf{A}} \qquad \underset{6 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}
$$

How to solve t?

- Least squares: find **t** that minimizes

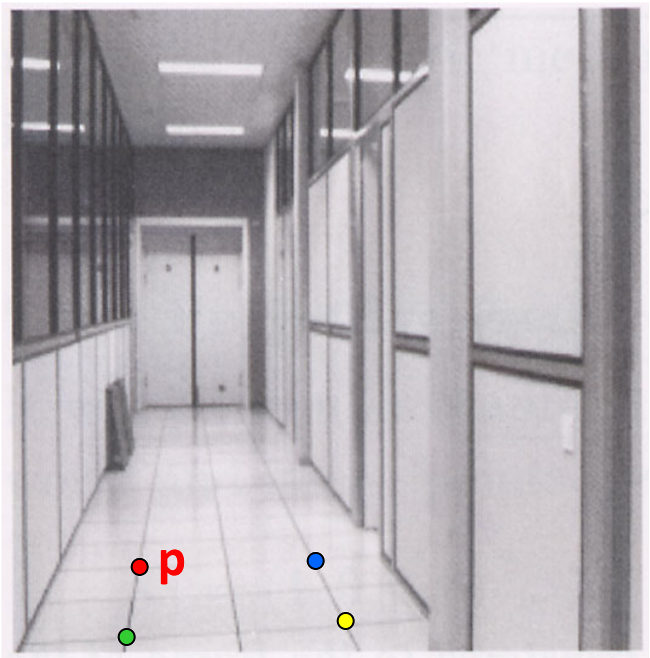
$$||\mathbf{A}\mathbf{t} - \mathbf{b}||^2$$

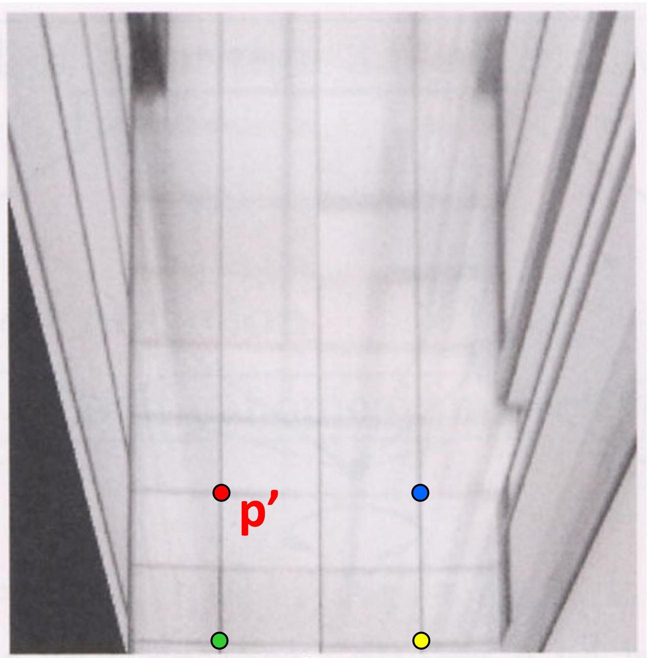
- To solve, form the *normal equations*

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{t} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{t} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

Projective transformations





P’ Homography P

$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} \sim H_{10} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}$$

Solving for homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\begin{aligned} x'_i &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y'_i &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{aligned} x'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

Solving for homographies

$$
\begin{aligned}
x_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\
y_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12}
\end{aligned}
$$

$$
\begin{bmatrix}
x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\
0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i'
\end{bmatrix}
\begin{bmatrix}
h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22}
\end{bmatrix}
=
\begin{bmatrix}
0 \\ 0
\end{bmatrix}
$$

Solving for homographies

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{2n \times 9}{\mathbf{A}} \qquad \underset{9}{\mathbf{h}} \qquad \underset{2n}{\mathbf{0}}$$

Defines a least squares problem: minimize $\|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2$

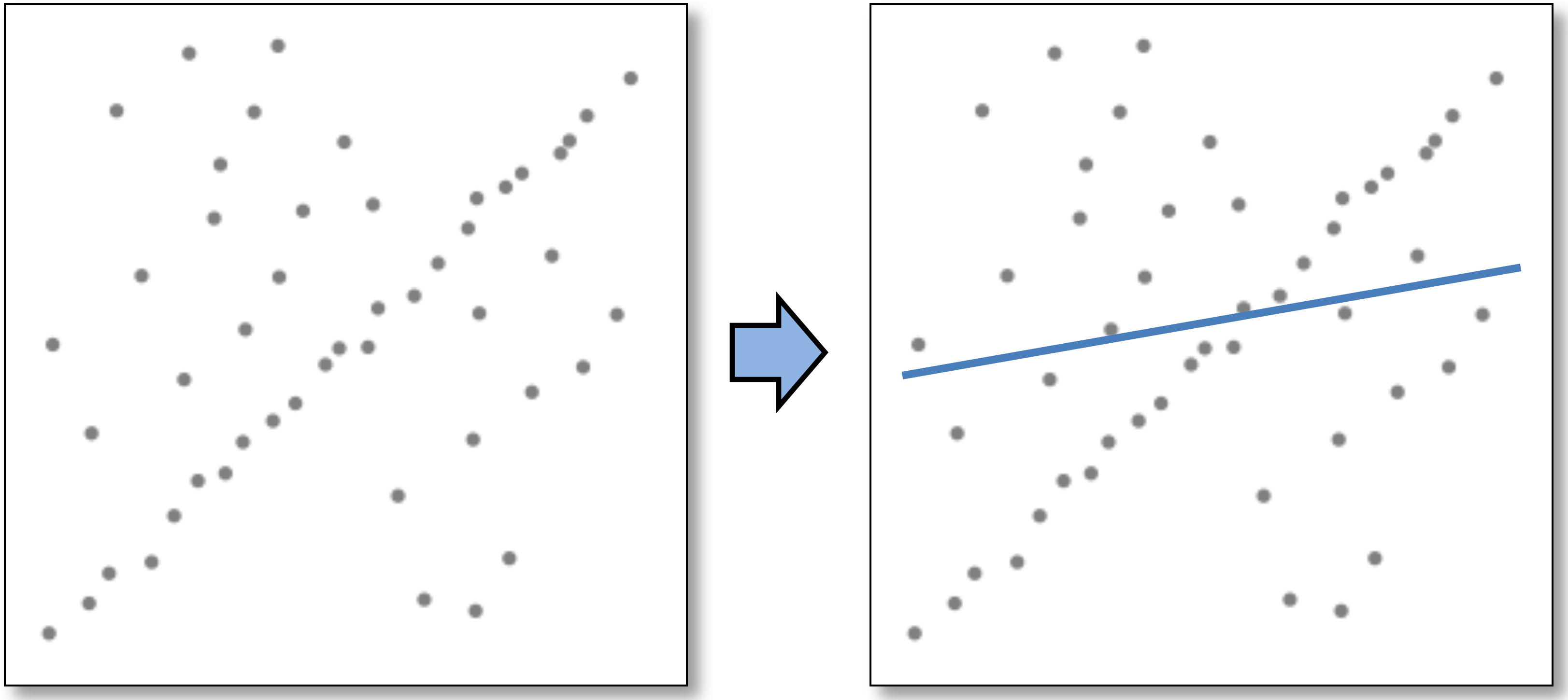
- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

Outliers

outliers
MAKING
COMICS
STORYTELLING SECRETS OF COMICS, MANGA, AND GRAPHIC NOVELS
FROM THE AUTHOR OF UNDERSTANDING COMICS
SCOTT McCLOUD
SOUTH WEST
DVD
THE COMPLETE
BLACK ADDER
BBC
inliers

Robustness

- Let’s consider a simpler example… linear regression

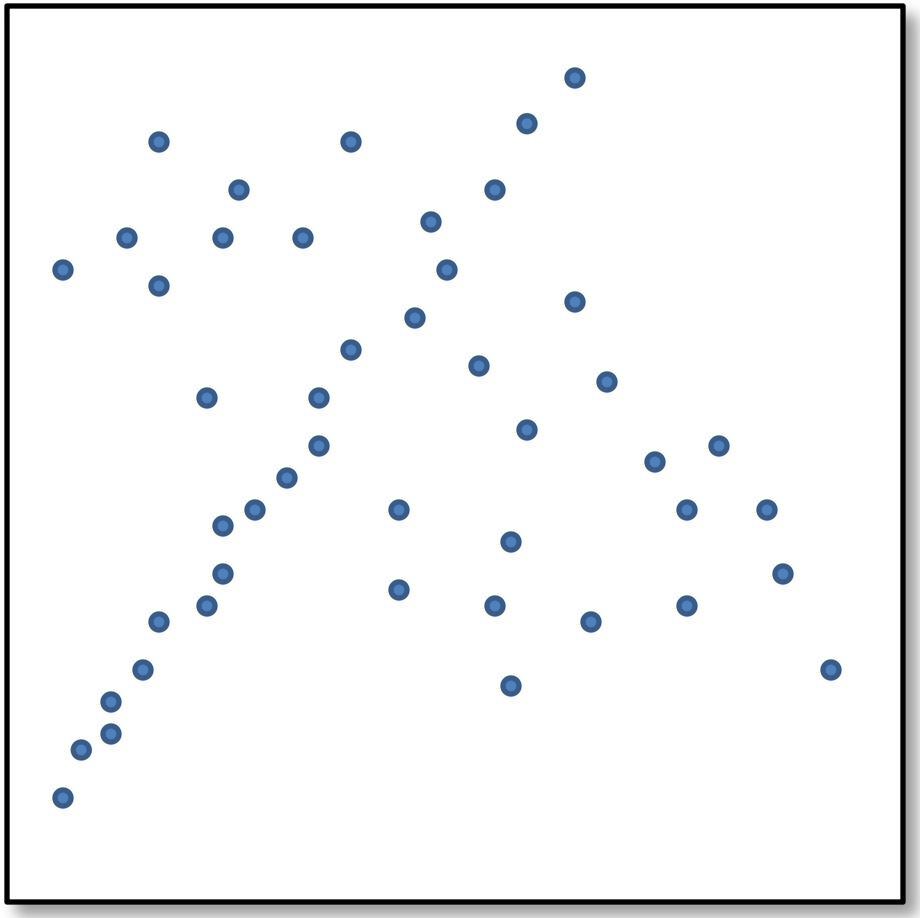


Problem: Fit a line to these datapoints

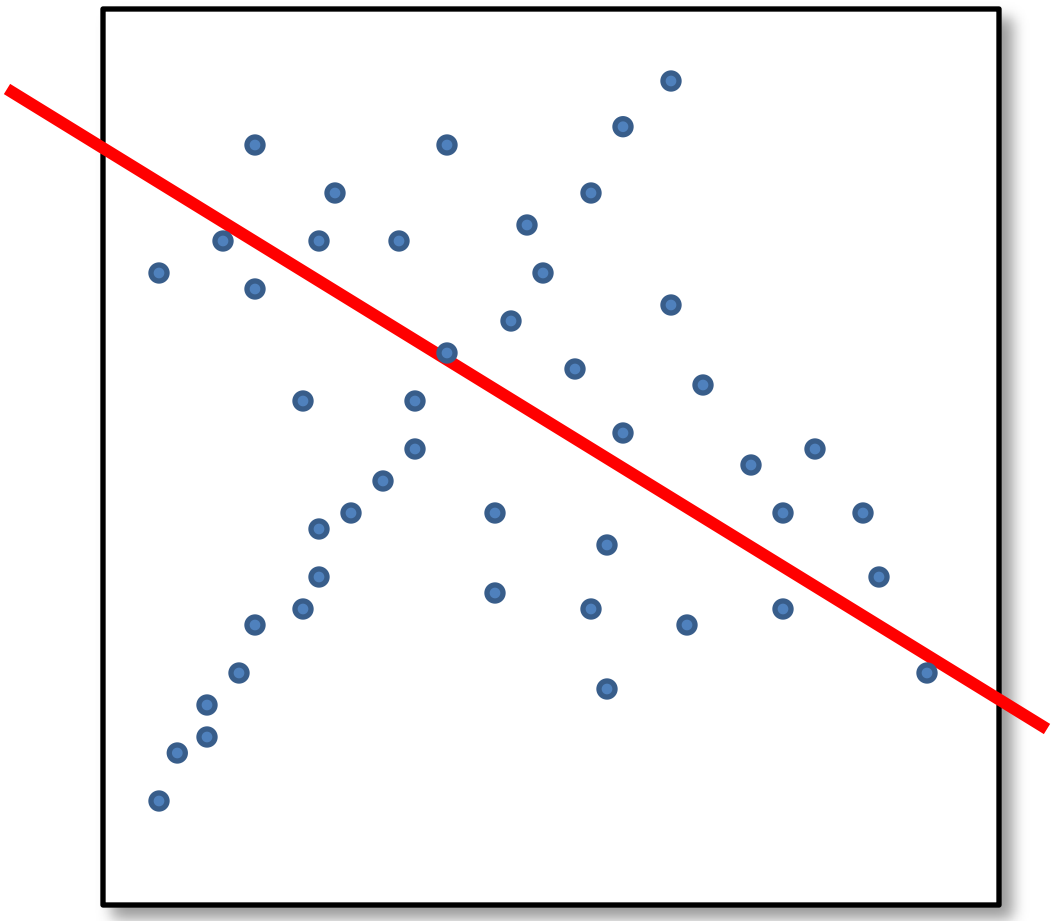
Least squares fit

- How can we fix this?

RANSAC

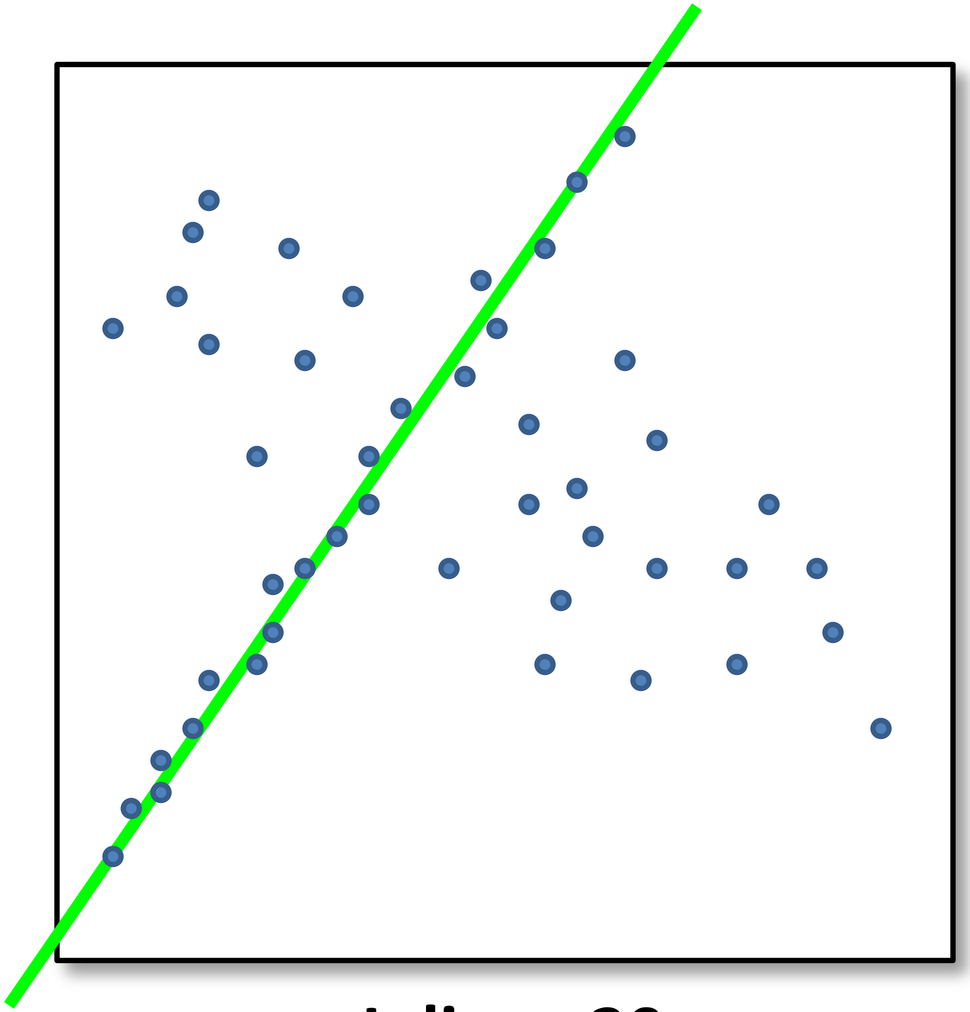


RANSAC



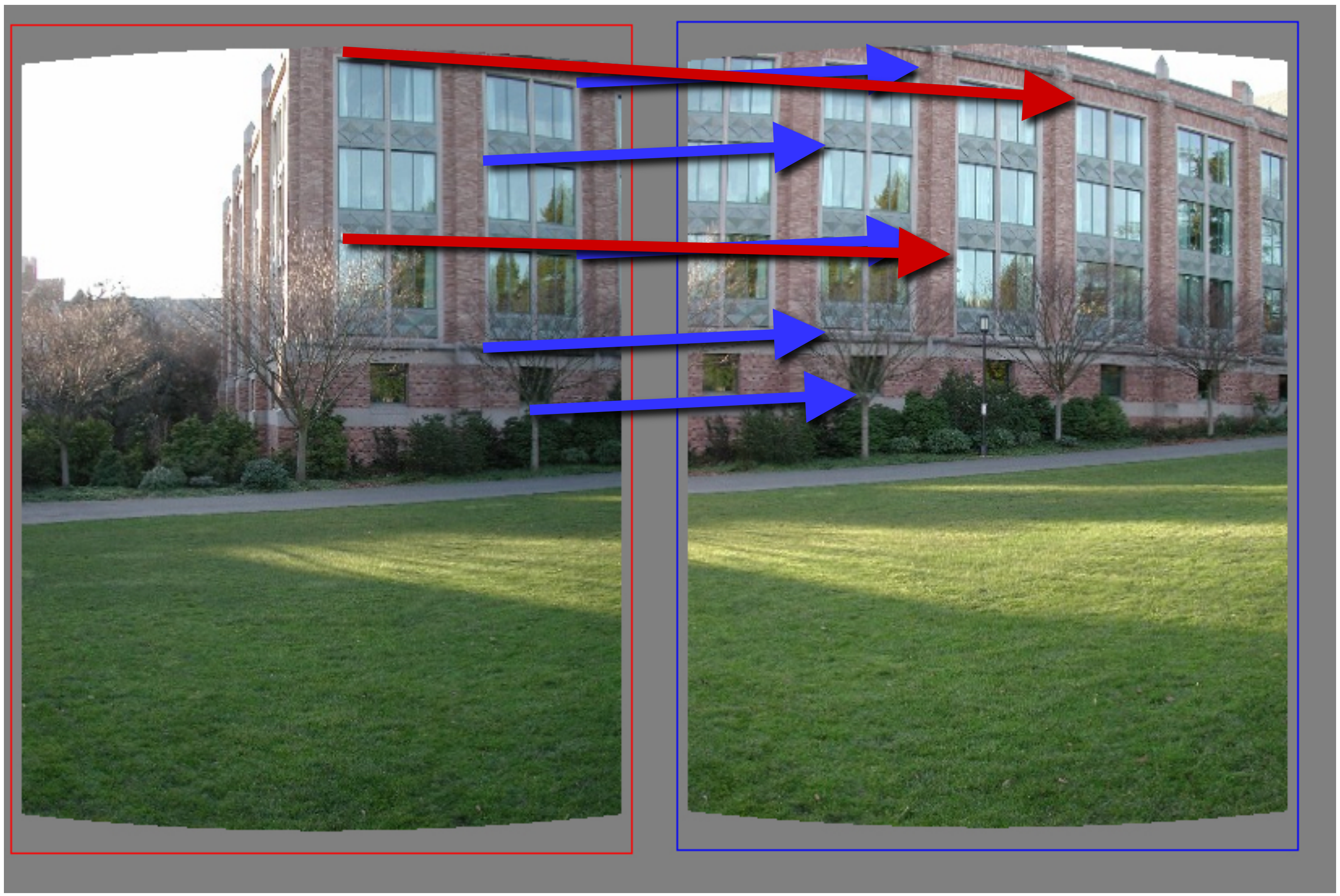
Inliers: 3

RANSAC

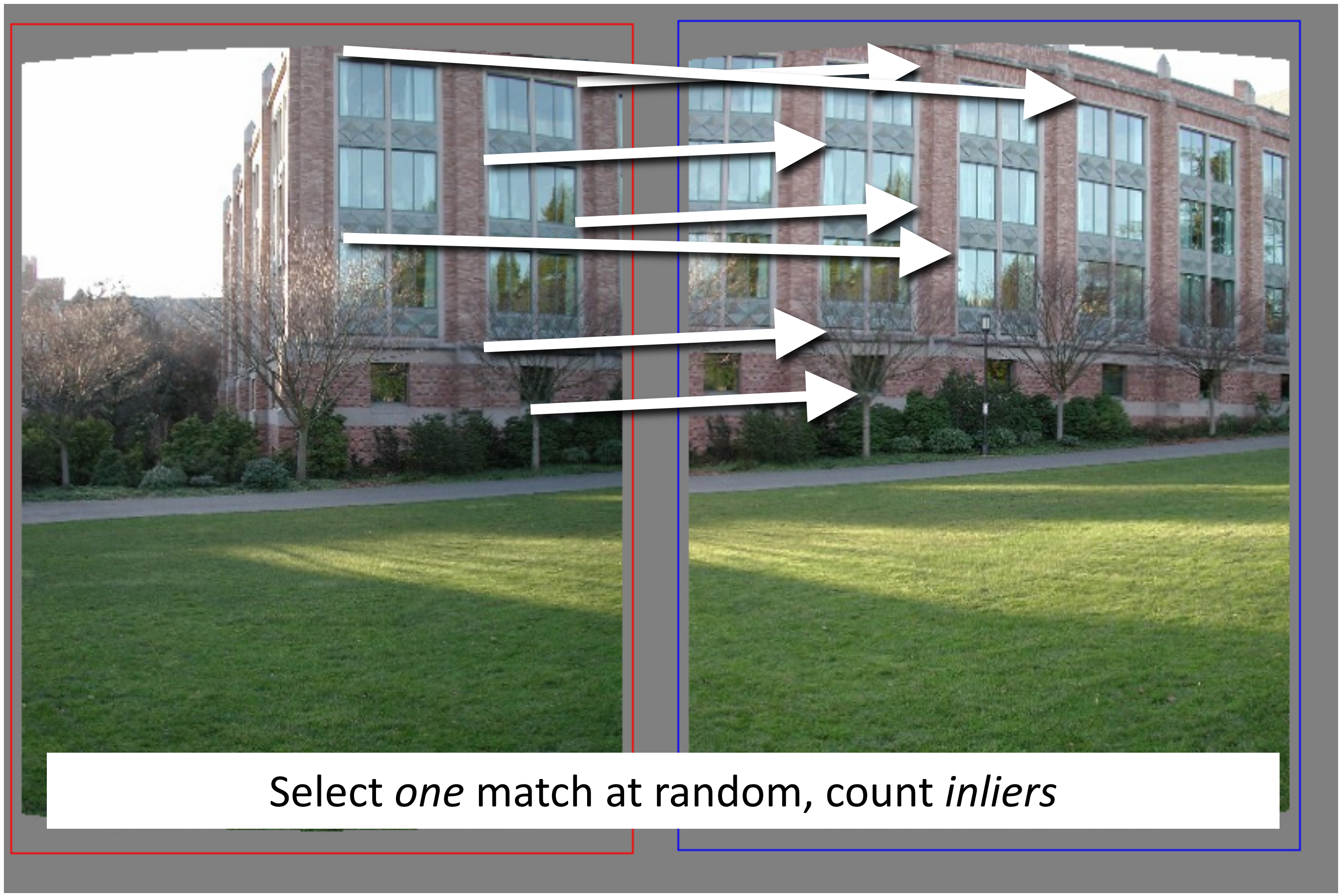


Inliers: 20

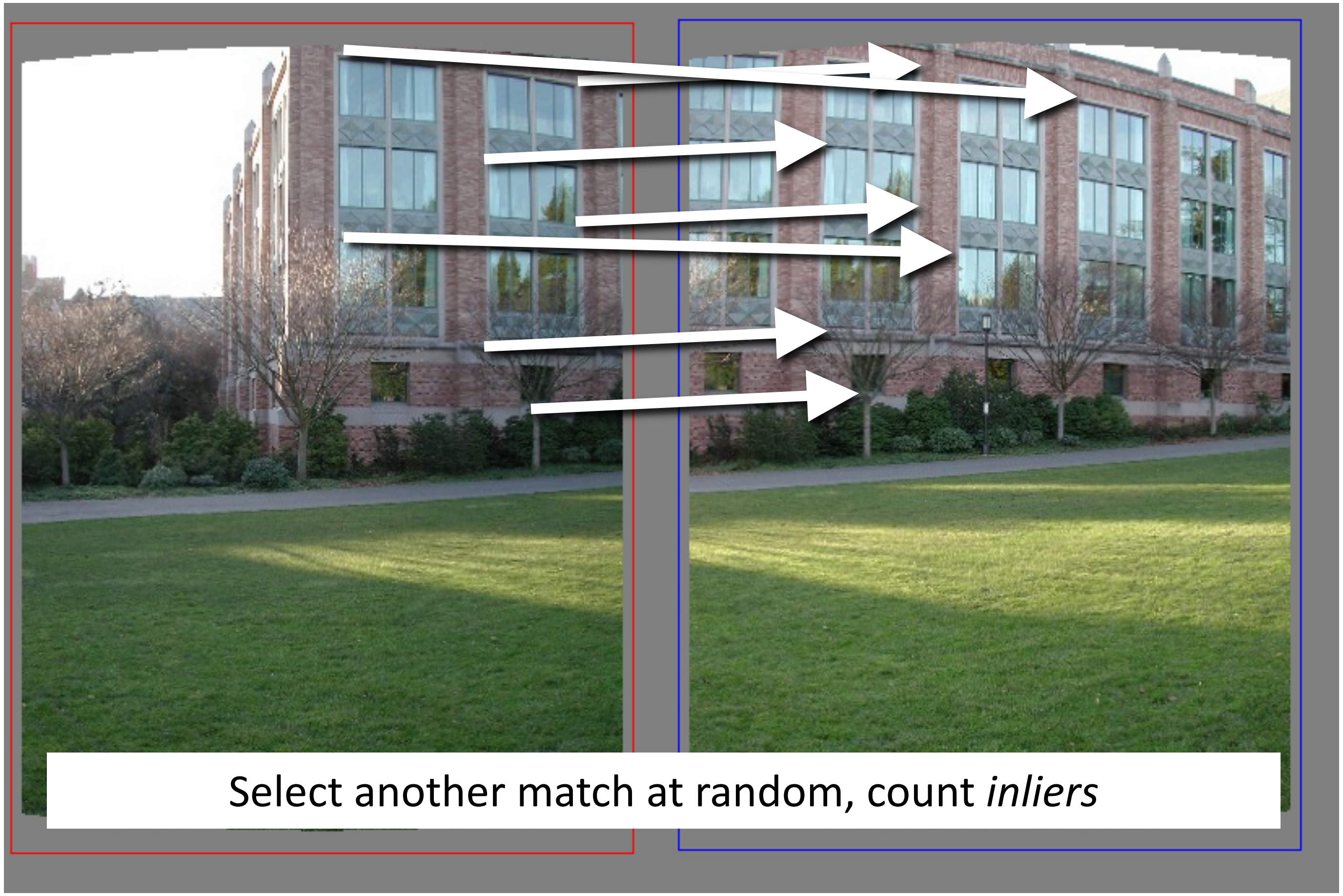
RANSAC for Translation



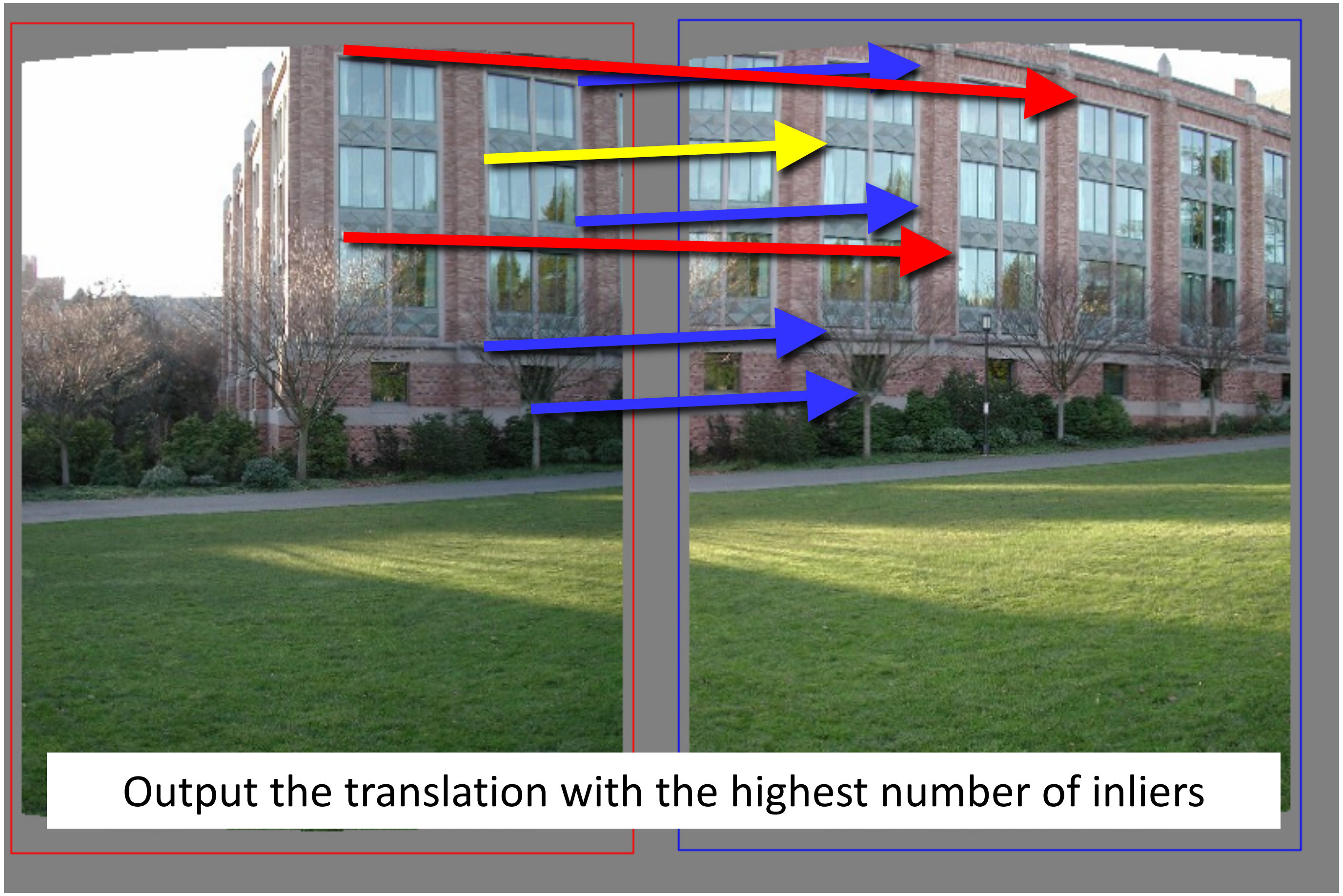
RANSAC for Translation



RANSAC for Translation



RANSAC for Translation



RANSAC

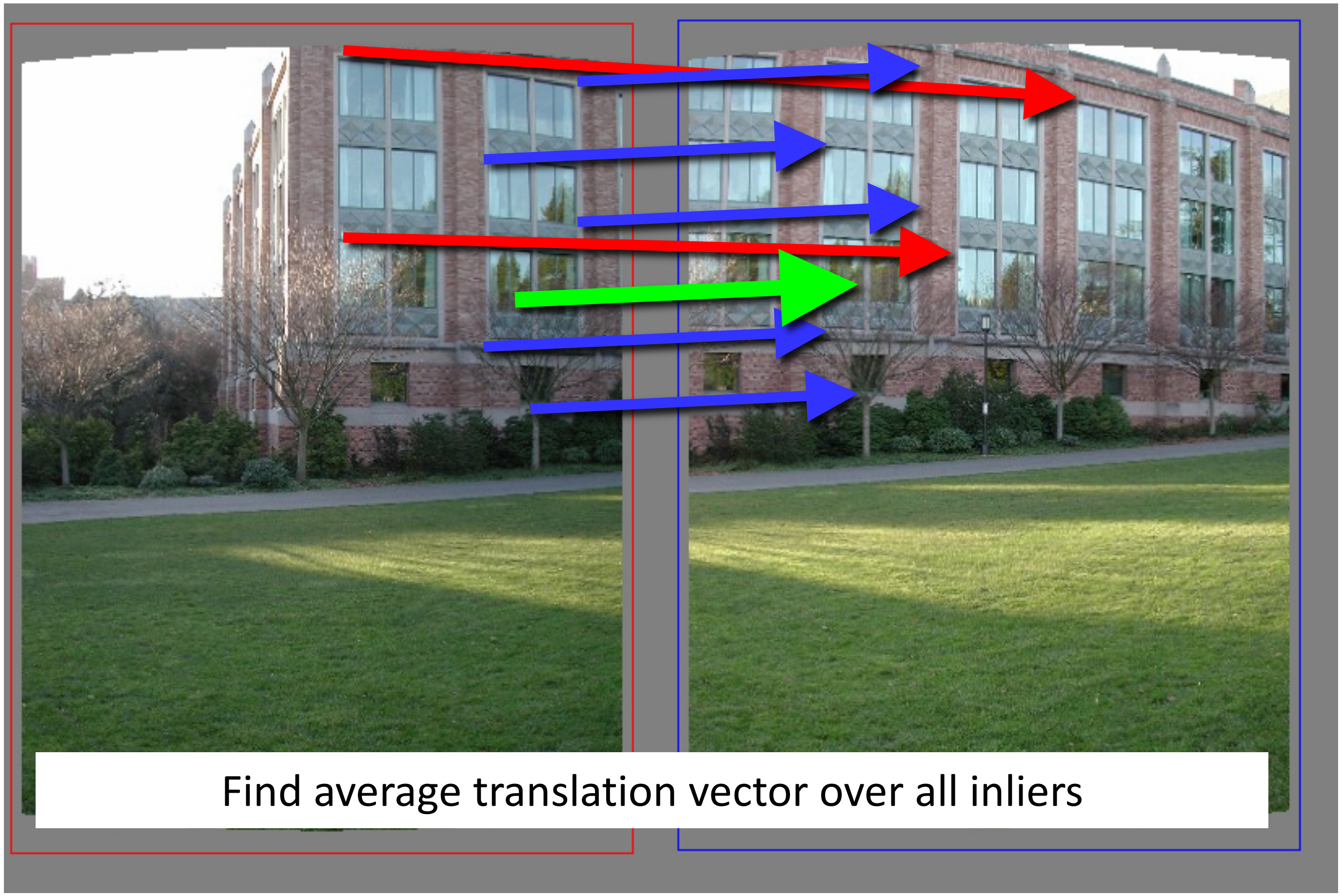
- Idea:
  - All the inliers will agree with each other on the translation vector;
  - The outliers will disagree with each other
    - RANSAC only has guarantees if there are < 50% outliers
  - "All good matches are alike; every bad match is bad in its own way."

– Tolstoy via Alyosha Efros

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., transformation matrix) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat *N* times
  5. Choose the model that has the largest set of inliers

Final step: least squares fit

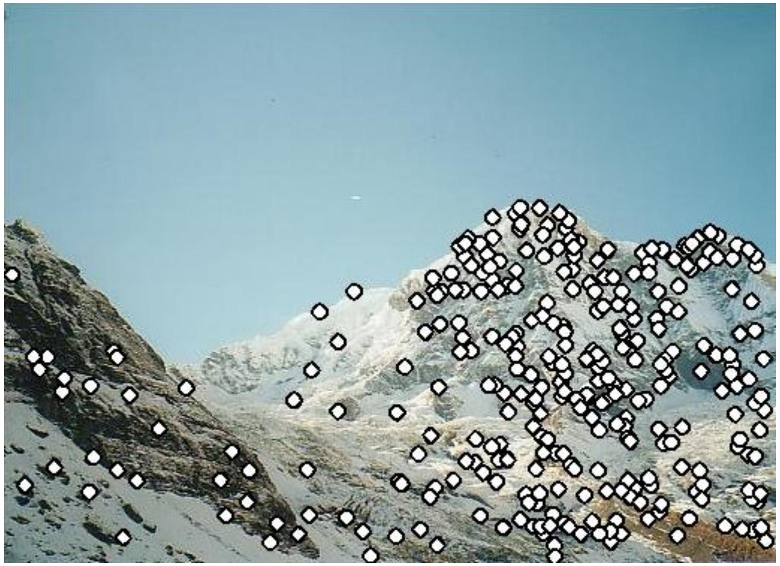


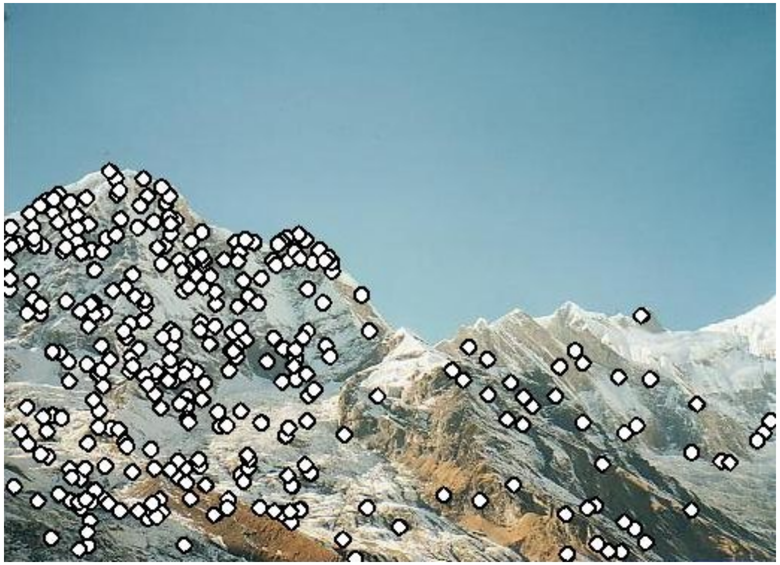
Find average translation vector over all inliers

图像拼接

输入图像

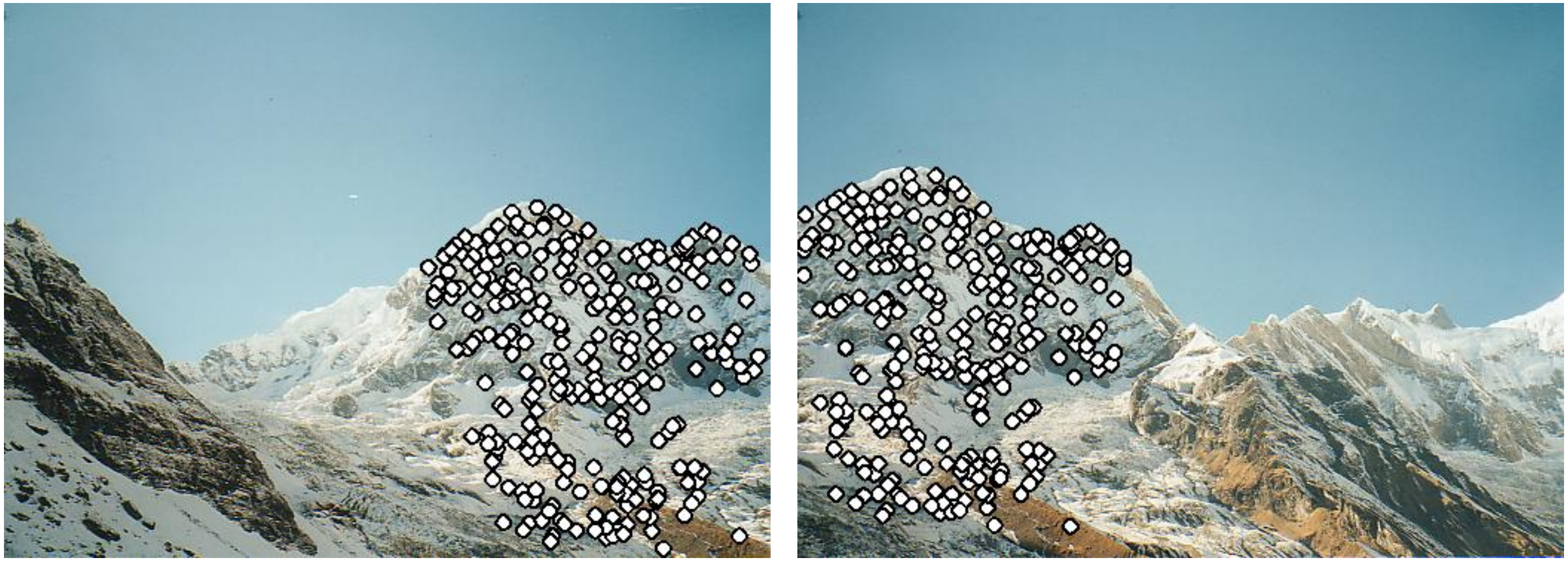
图像拼接





特征匹配

图像拼接



RANSAC计算变换矩阵

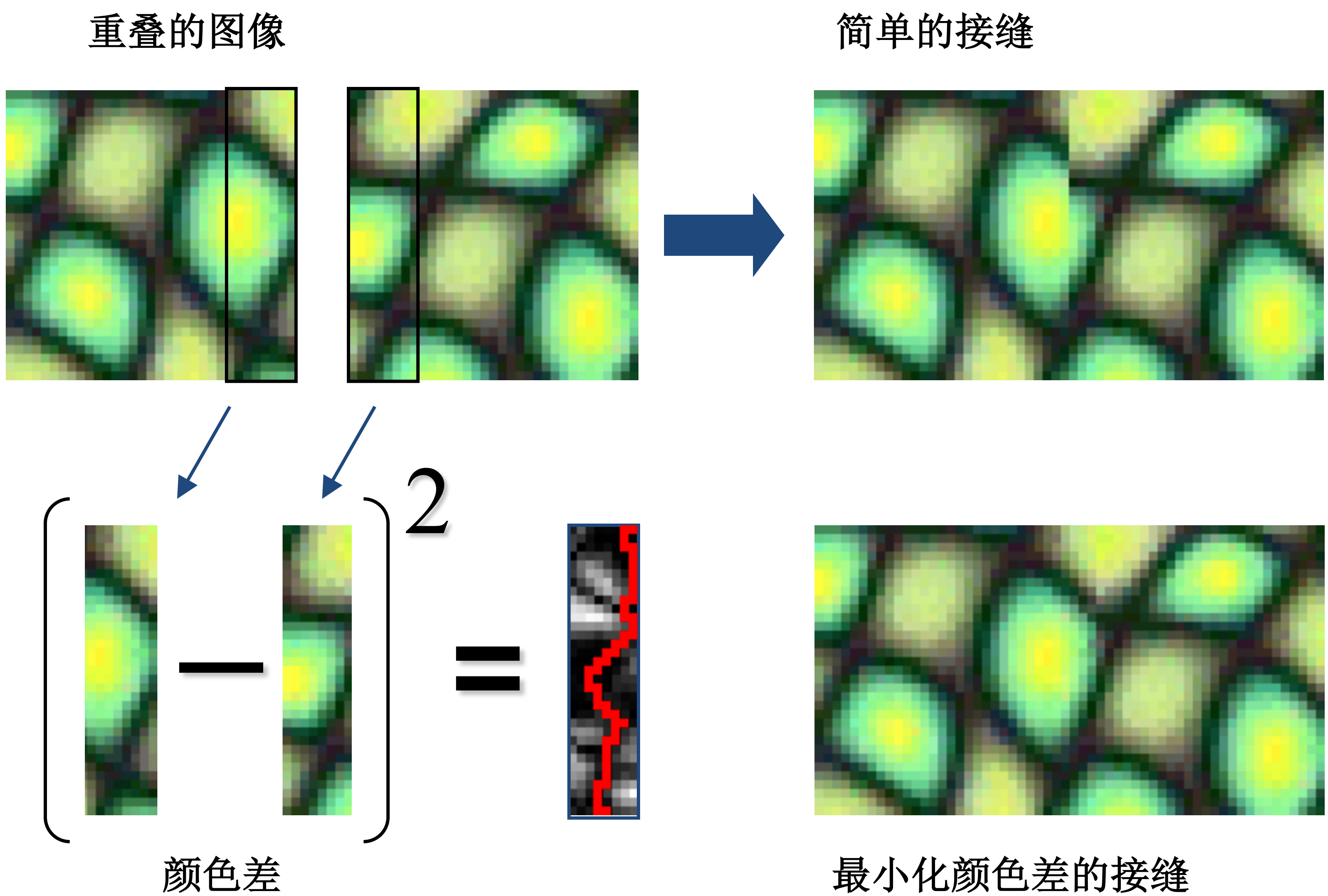
图像拼接

固定第一幅图，变换第二幅图

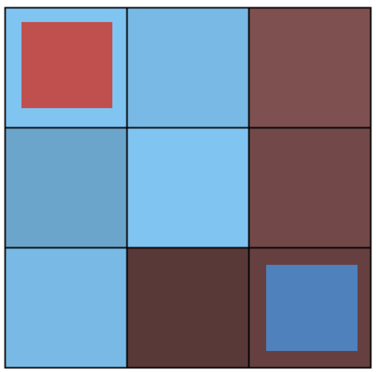
无缝融合

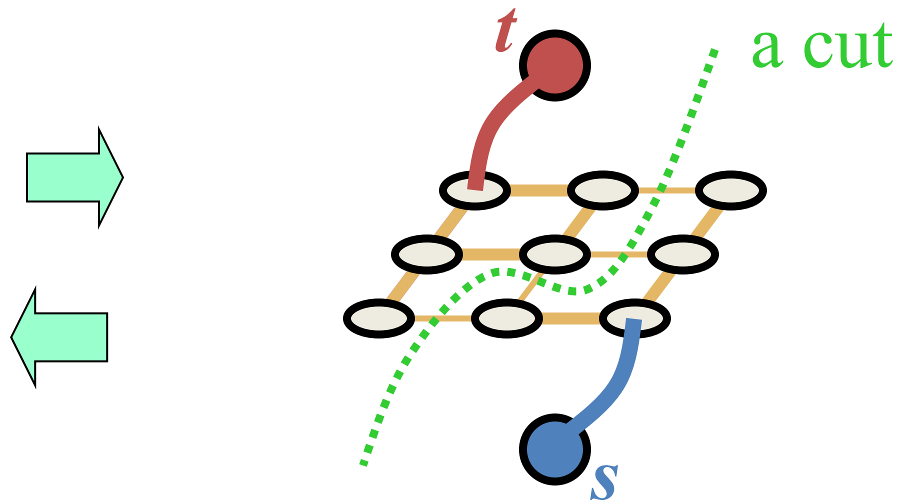
- Graphcut
- Poisson Image Editing

无缝融合



无缝融合



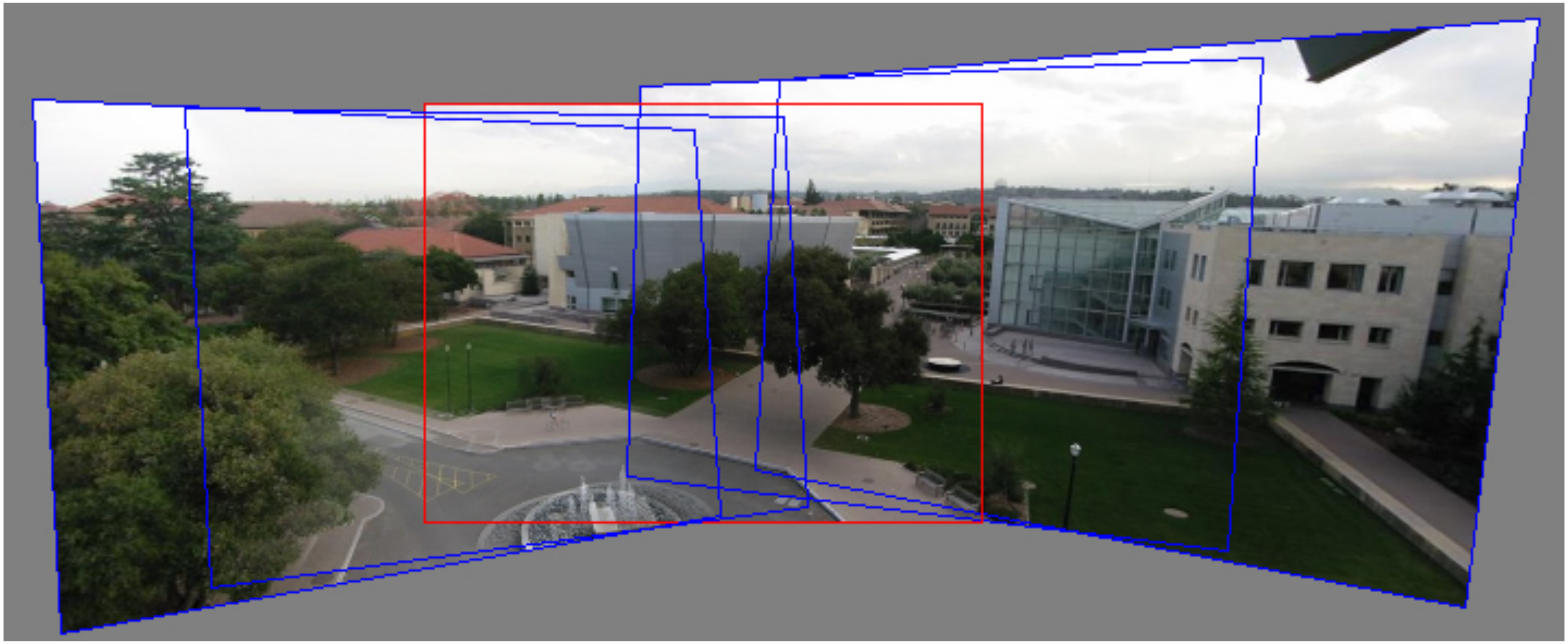


最大流最小割算法

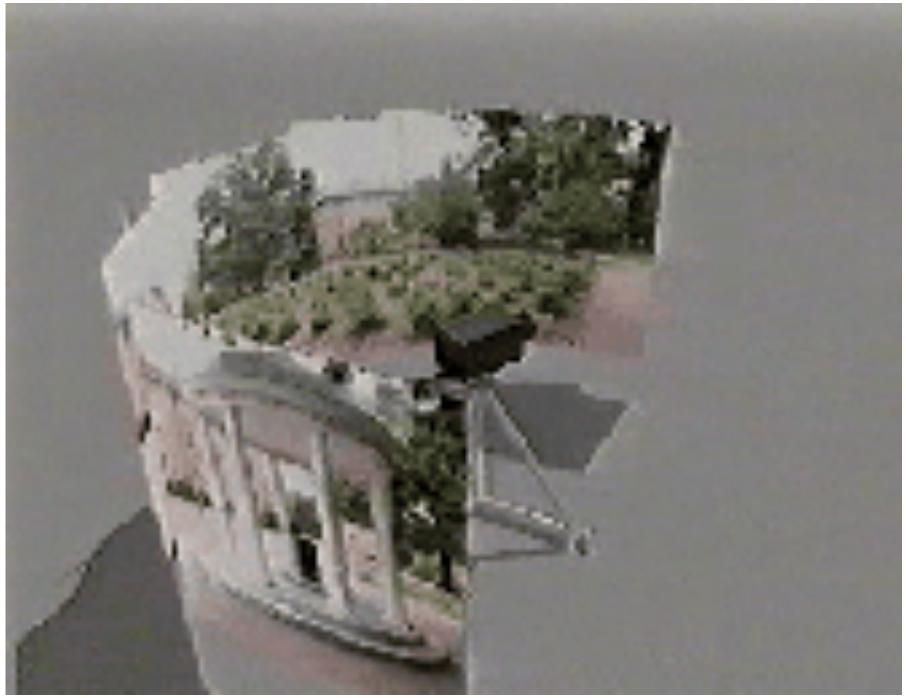
多项式时间

Panoramas

- Now we know how to create panoramas!
  - 1) Warp all images to a reference image; 2) merge them



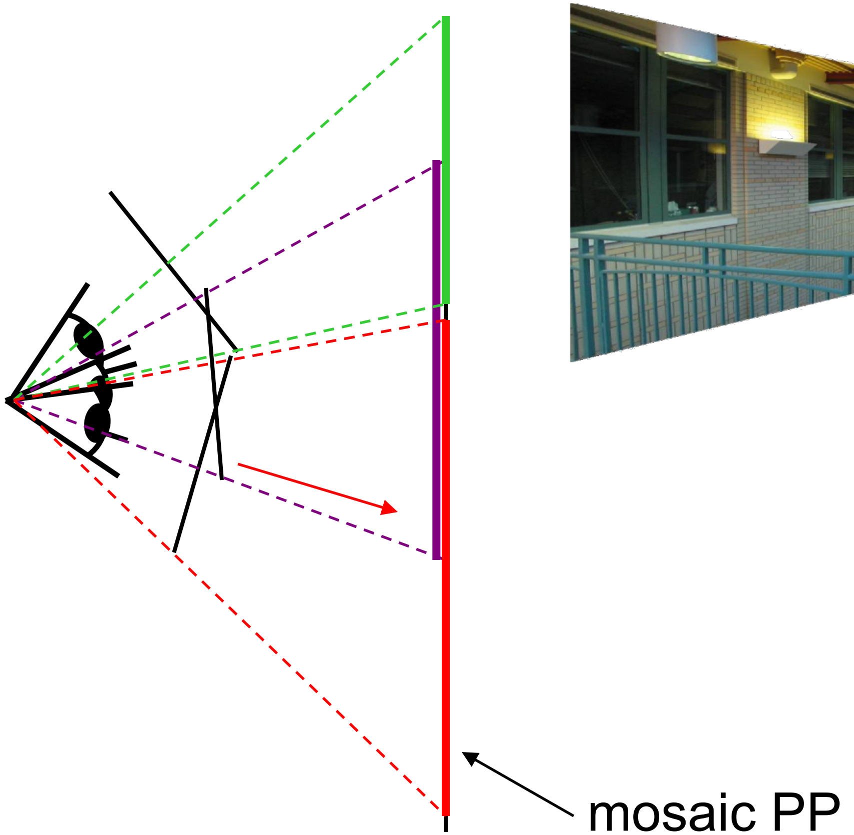
Rotation about vertical axis



- What if our camera rotates on a tripod?
- What's the structure of H?

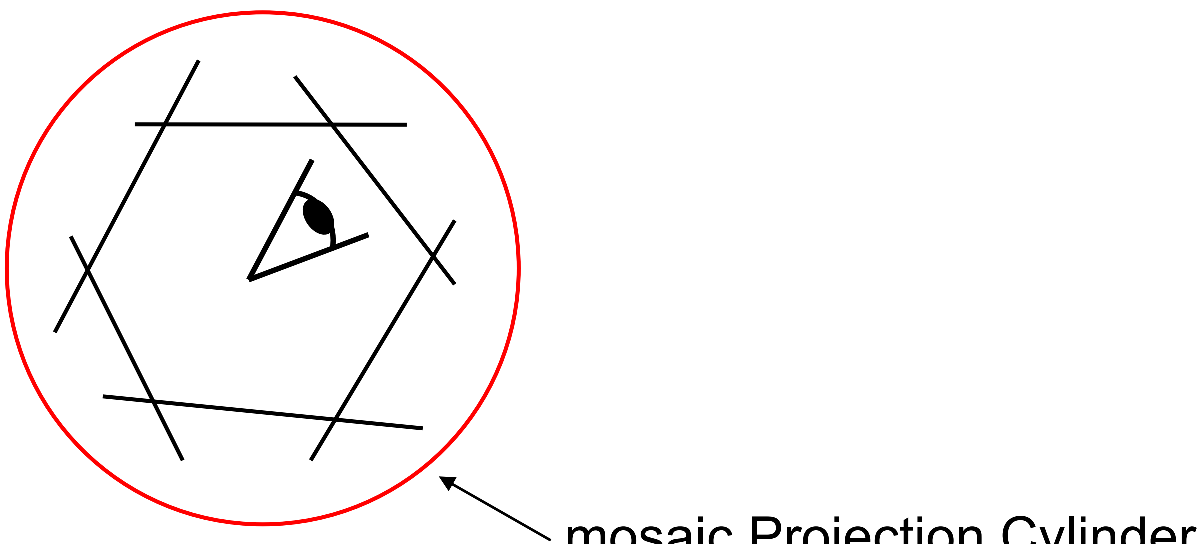
$$H = KRK^{-1}$$

Do we have to project onto a plane?

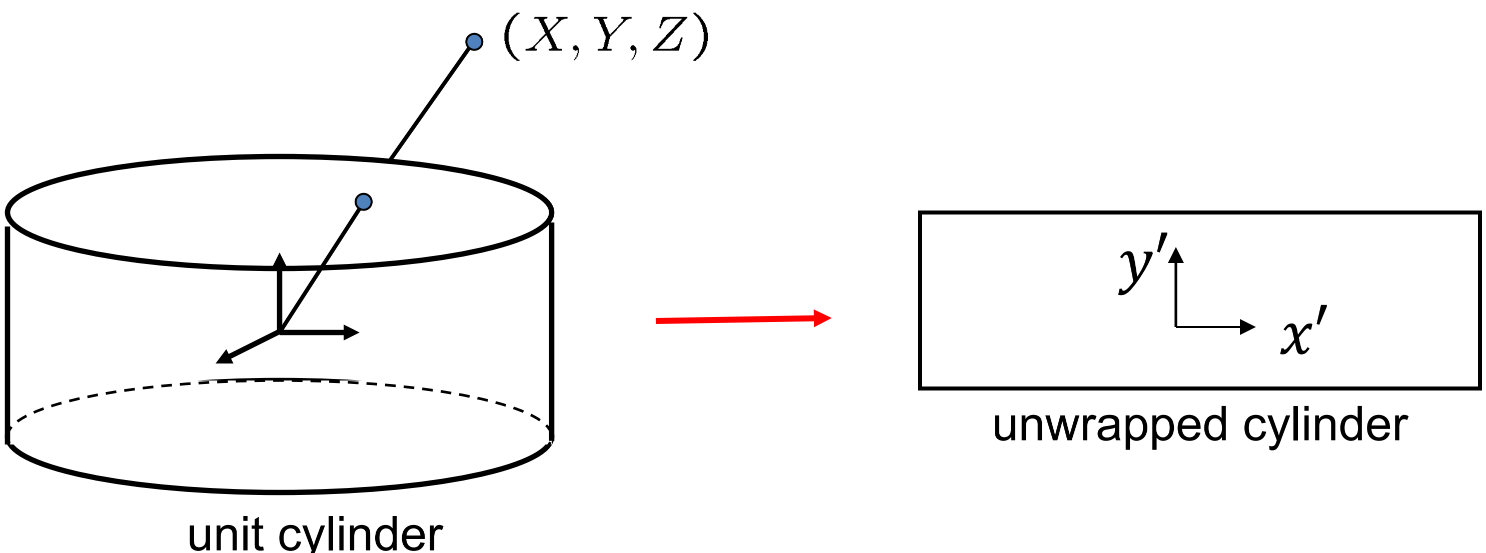


Full Panoramas

- What if you want a 360° field of view?



Cylindrical projection

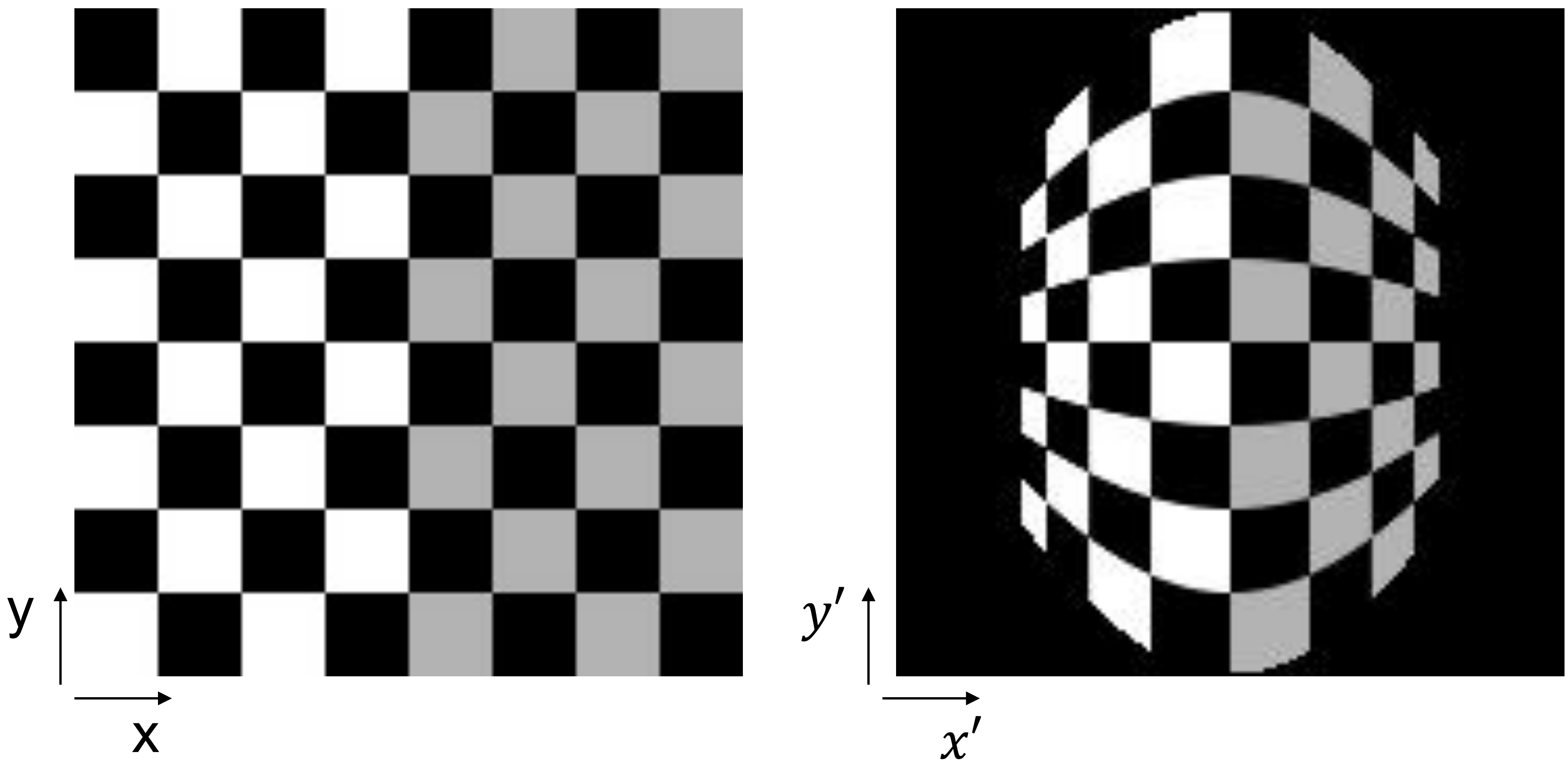


$$x' = r tan^{-1}(\frac{x}{f})$$

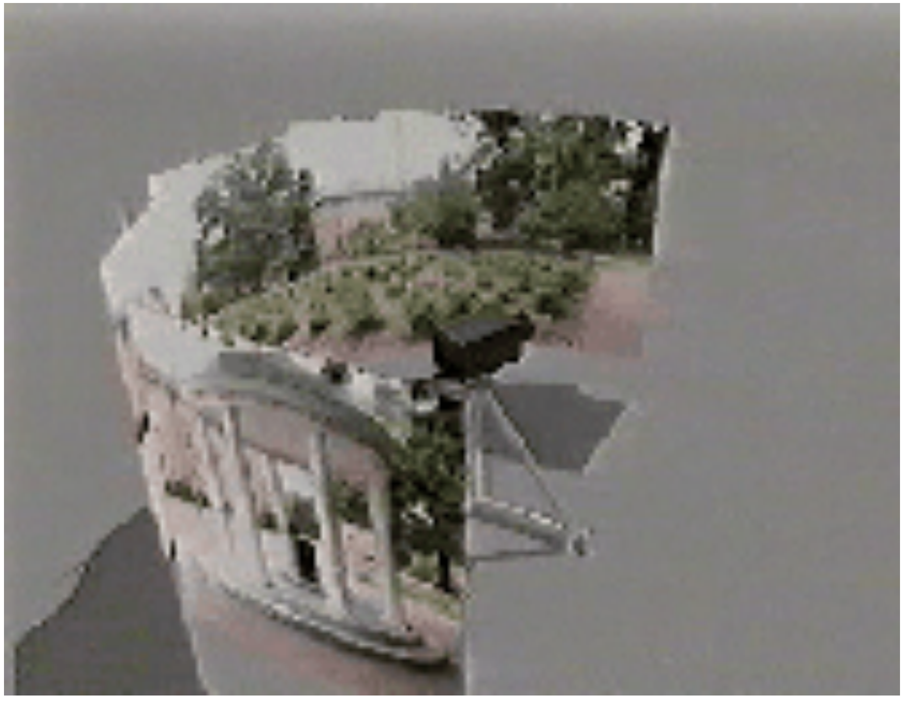
$$y' = \frac{ry}{\sqrt{x^2+f^2}}$$

其中(x', y')为柱面上的坐标,(x, y)为平面图像坐标，其坐标原点都已移至图像中心，r 为柱面半径，f 为焦距。

Cylindrical projection



Cylindrical panoramas

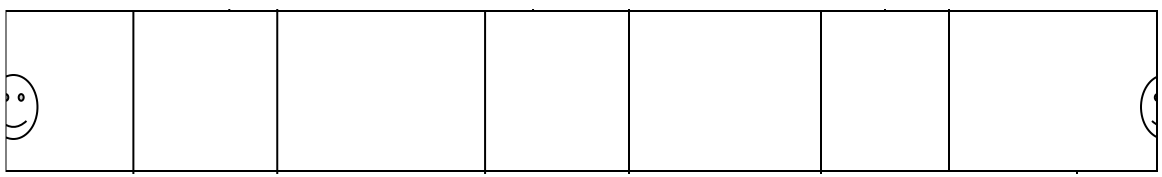


- Steps
  - Reproject each image onto a cylinder
  - Blend
  - Output the resulting mosaic

Cylindrical image stitching

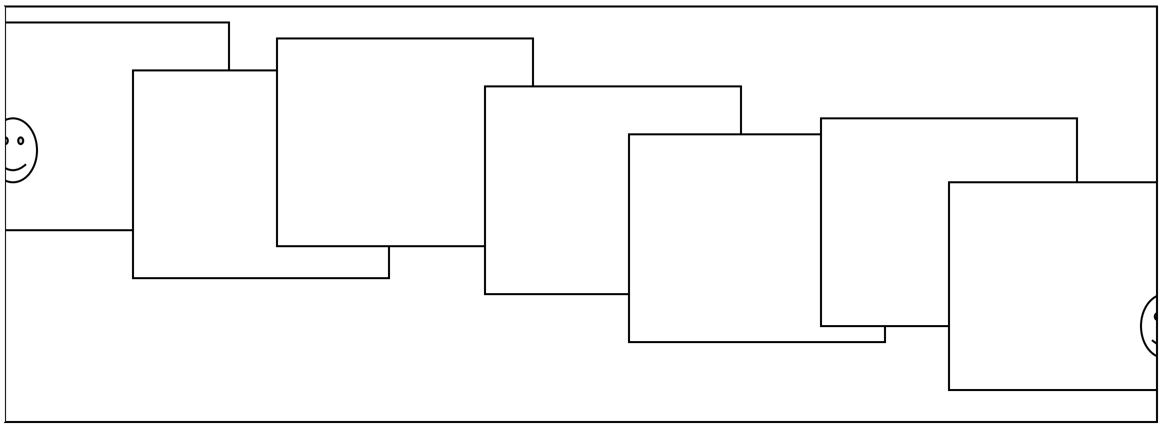
- What if you don’t know the camera rotation?
  - Solve for the camera rotations
    - Note that a rotation of the camera is a **translation** of the cylinder!

Assembling the panorama



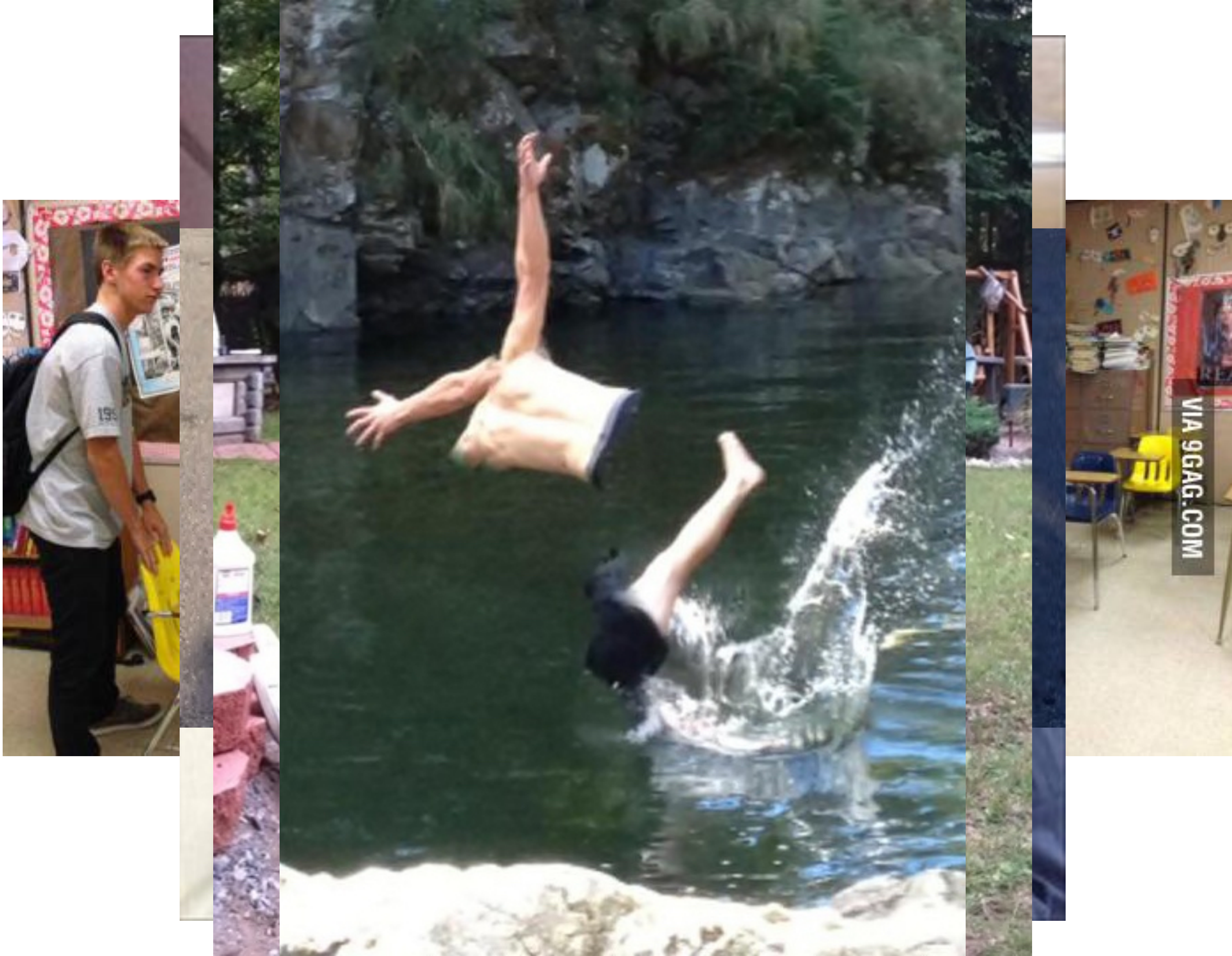
- Stitch pairs together, blend, then crop

Problem: Drift



- small (vertical) errors accumulate over time
- apply correction so that sum = 0 (for 360° pan.)

Full-view (360°) panoramas

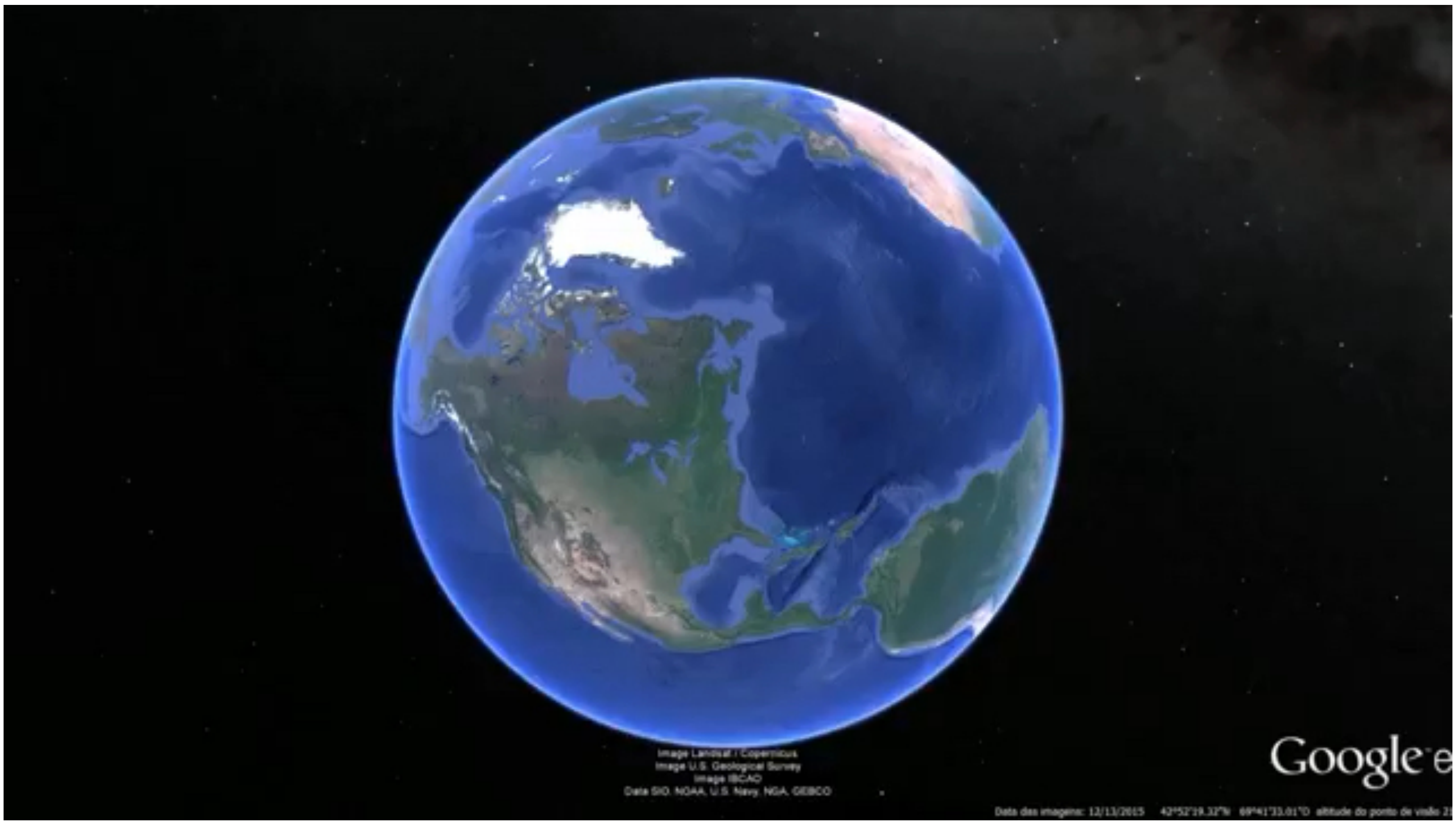

VIA 9GAG.COM

Questions?

基于单视图的三维重建

周晓巍

3D Navigation (Free Viewpoint)

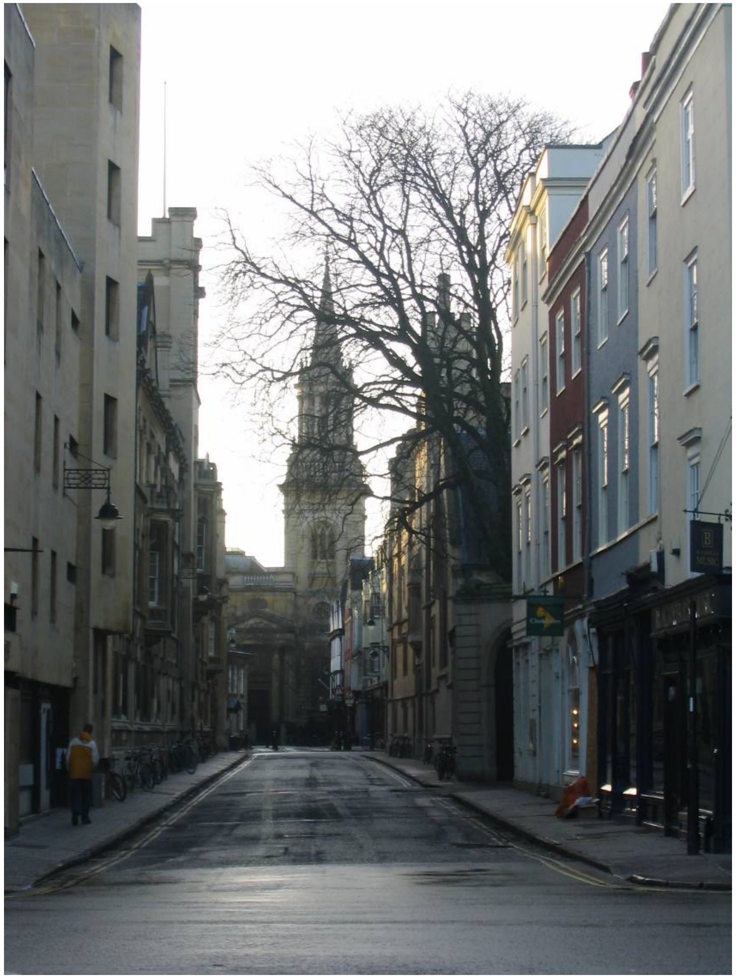


3D Navigation (Free Viewpoint)

- Need 3D models
- Difficult to obtain high-quality models

3D Navigation (Free Viewpoint)

Can we do it from a single photograph?



3D modeling from a photograph

St. Jerome in his Study, H. Steenwick

3D modeling from a photograph

3D modeling from a photograph

Flagellation, Piero della Francesca

3D modeling from a photograph

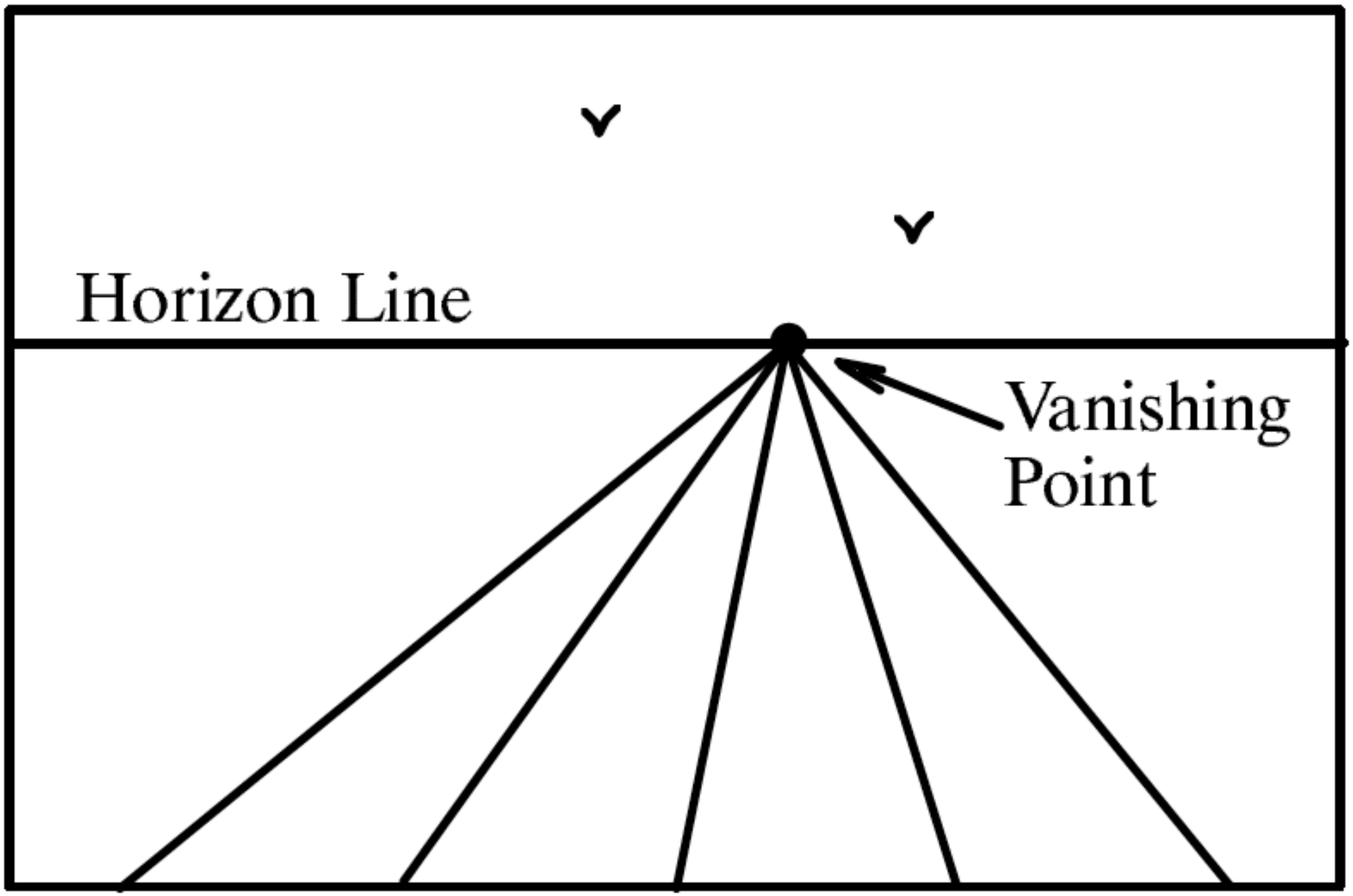
video by Antonio Criminisi

Some preliminaries: projective geometry

3D to 2D: perspective projection

Projection:

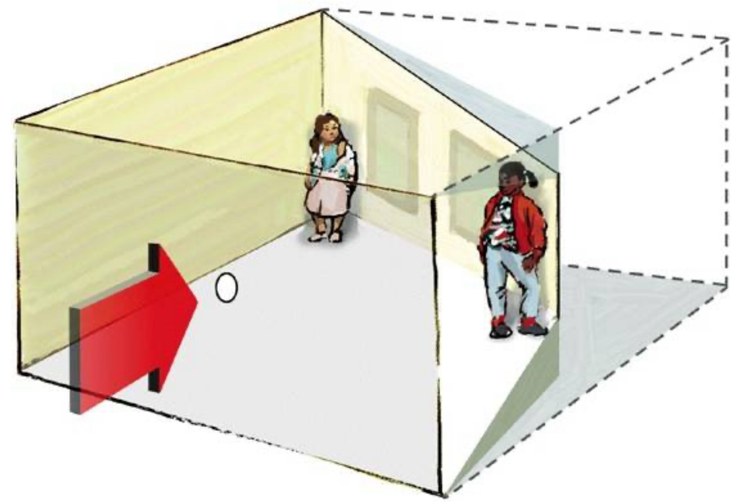
$$\mathbf{p} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi P}$$



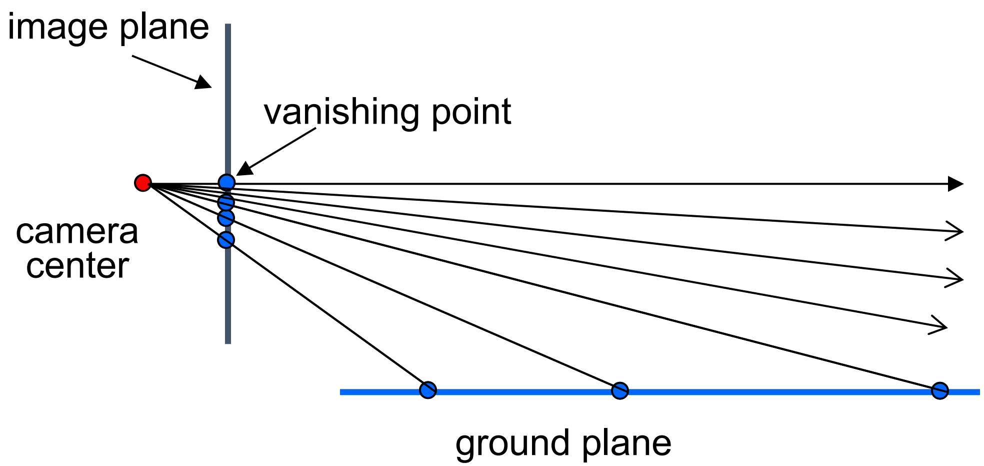
Vanishing point and line tell us a lot about camera position and orientation

Figure 23.4
A perspective view of a set of parallel lines in the plane. All of the lines converge to a single vanishing point.

Ames Room

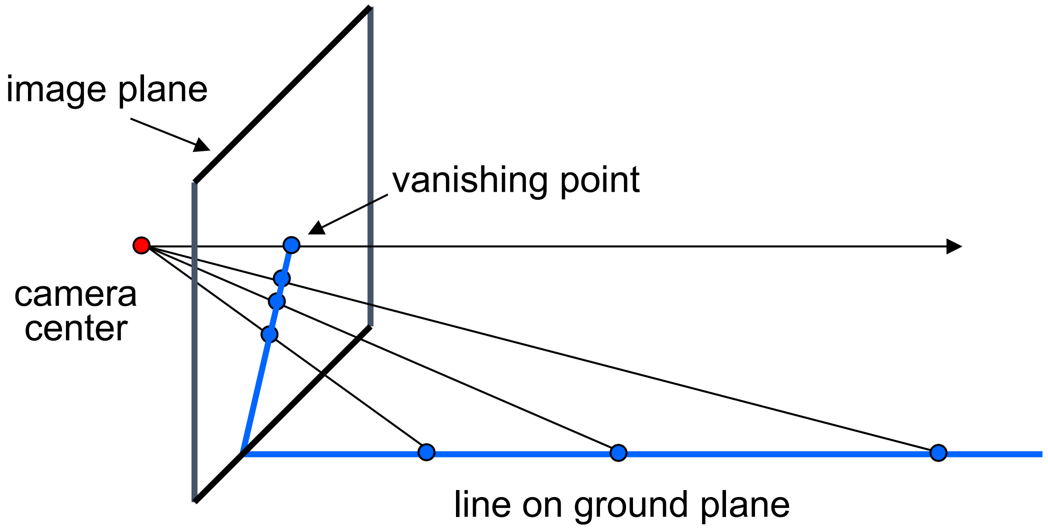


Vanishing points

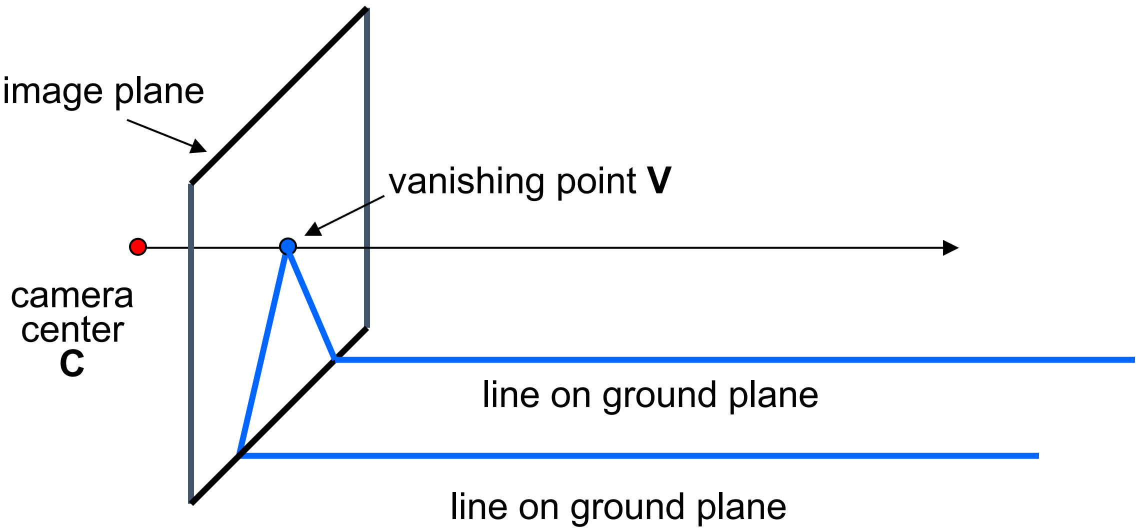


- Vanishing point
  - projection of a point at infinity

Vanishing points (2D)

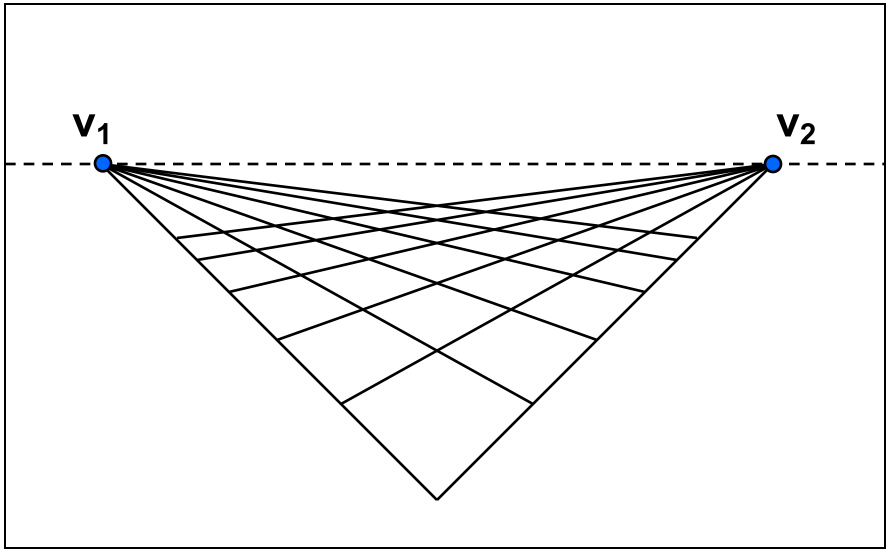


Vanishing points



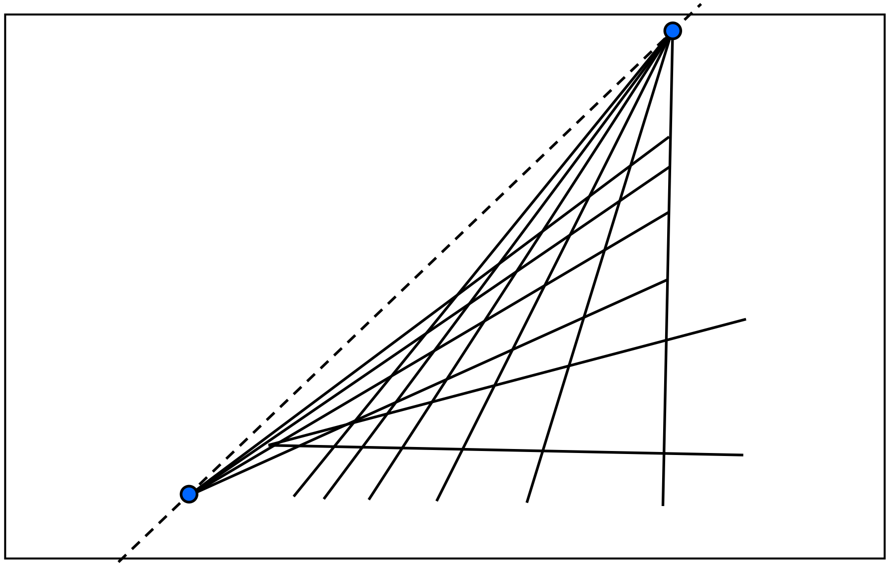
- Properties
  - Any two parallel lines have the same vanishing point **v**
  - The ray from **C** through **v** is parallel to the lines
    - v tells us the direction of the lines
  - An image may have more than one vanishing point

Vanishing lines



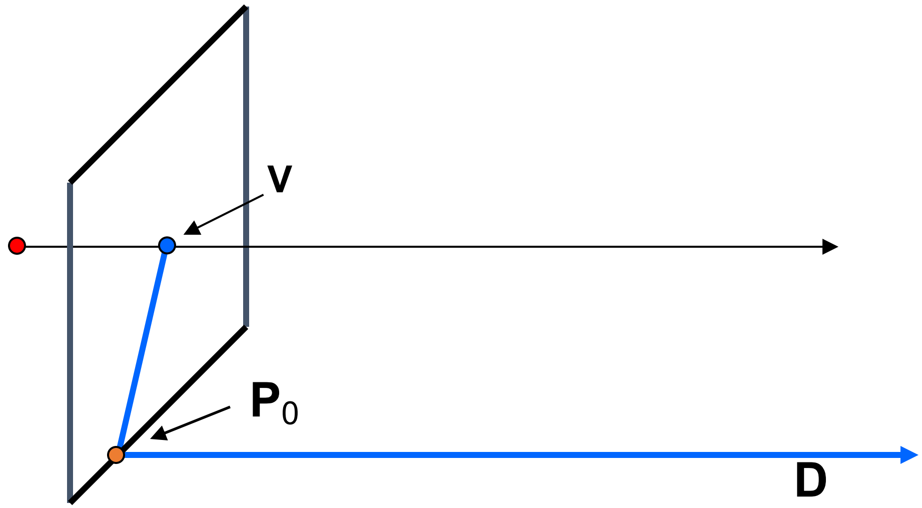
- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
    The union of all of these vanishing points is the *horizon line*
    - also called *vanishing line*
  - Note that different planes define different vanishing lines

Vanishing lines



- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
    The union of all of these vanishing points is the *horizon line*
    - also called *vanishing line*
  - Note that different planes define different vanishing lines

Computing vanishing points

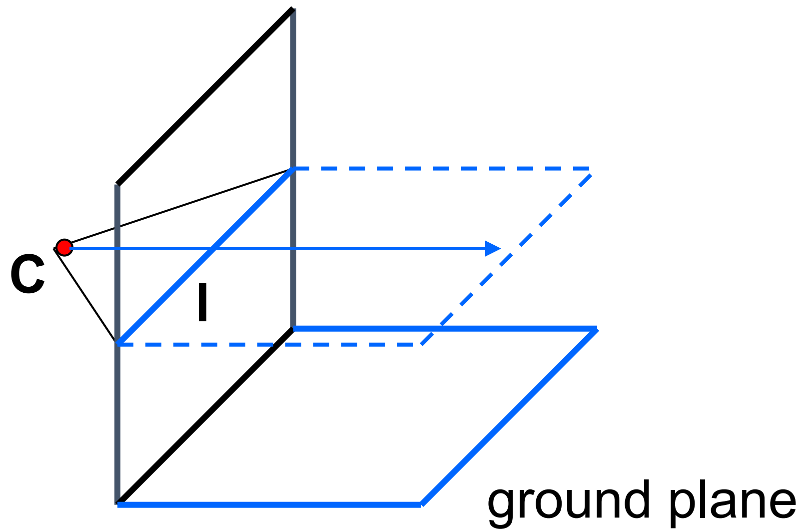


$$\mathbf{P} = \mathbf{P}_0 + t\mathbf{D}$$

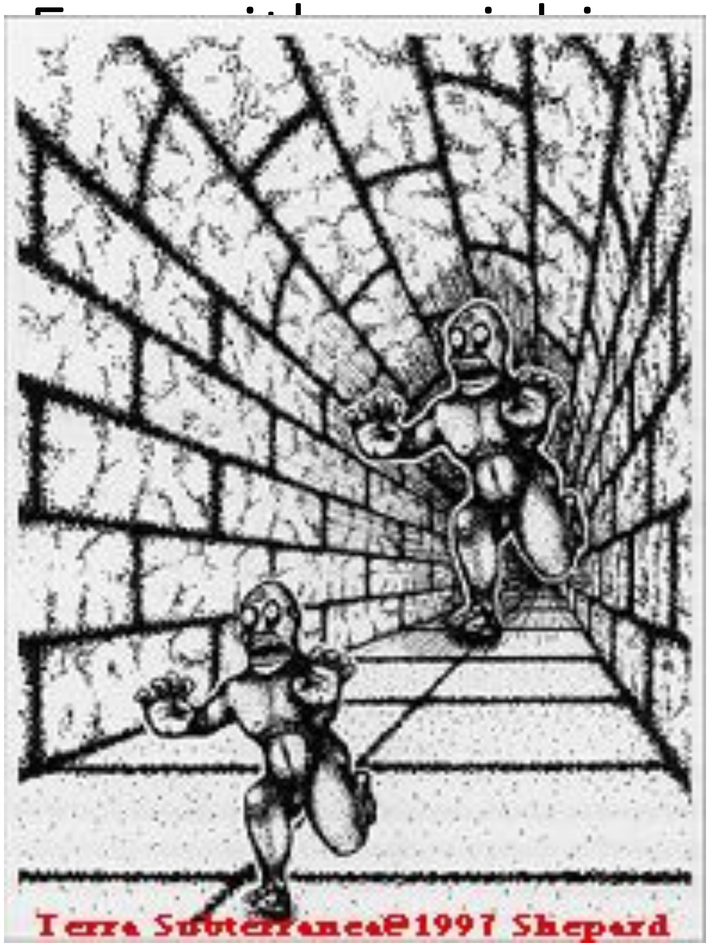
$$\mathbf{P}_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix} \cong \begin{bmatrix} P_X / t + D_X \\ P_Y / t + D_Y \\ P_Z / t + D_Z \\ 1 / t \end{bmatrix} \qquad t \to \infty \qquad \mathbf{P}_\infty \cong \begin{bmatrix} D_X \\ D_Y \\ D_Z \\ 0 \end{bmatrix}$$

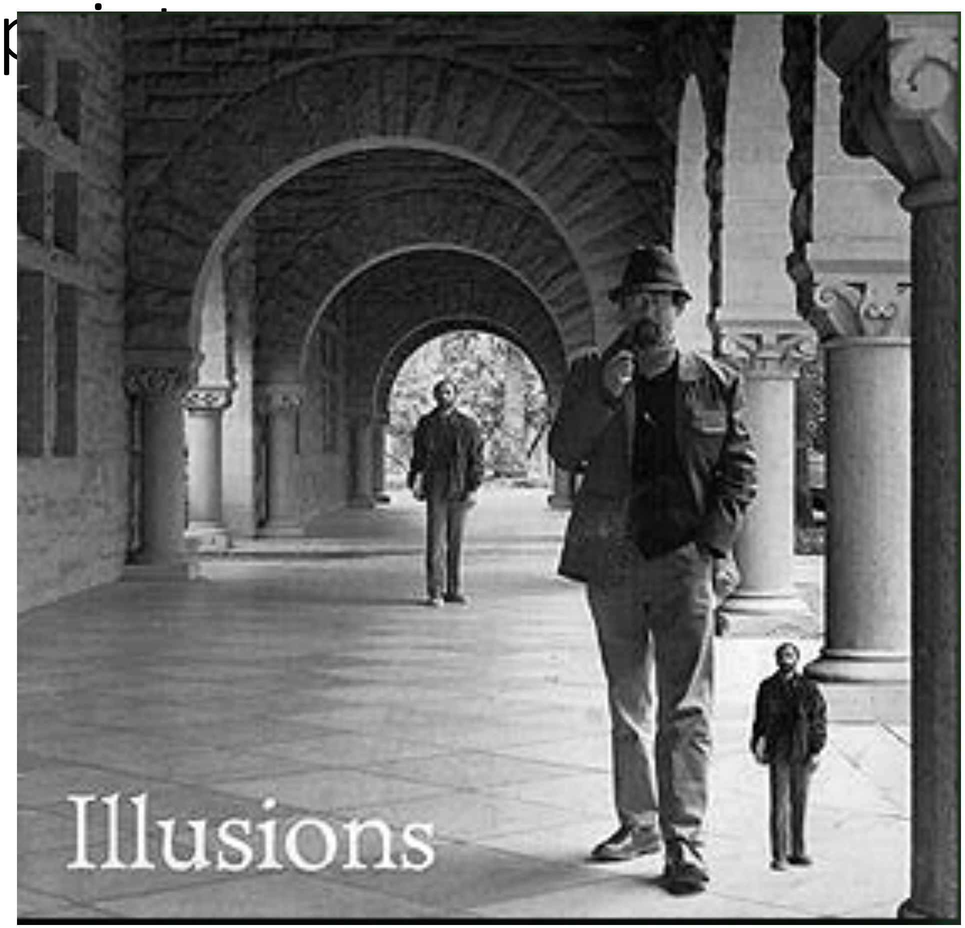
- Properties
  - $\mathbf{P}_\infty$ is a point at *infinity*, **v** is its projection $\mathbf{v} = \mathbf{\Pi}\mathbf{P}_\infty$
  - They depend only on line *direction (angle between the line and optical axis)*
  - Parallel lines $\mathbf{P}_0$ + t**D**, $\mathbf{P}_1$ + t**D** intersect at $\mathbf{P}_\infty$

Computing vanishing lines



- Compute **l** from two sets of parallel lines on ground plane
- Properties
  - **l** is intersection of horizontal plane through **C** with image plane
  - **l** depends on the orientation of the camera
  - All points at same height as **C** project to **l**
    - points higher than C project above l, vice versa


Terra Subterranea©1997 Shepard


Illusions

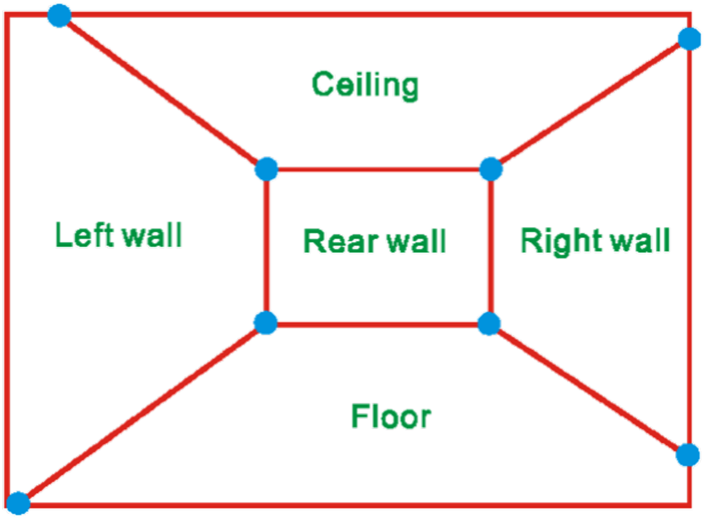
“Tour into the Picture” (SIGGRAPH ’97)

•Create a 3D “theatre stage” of five billboards

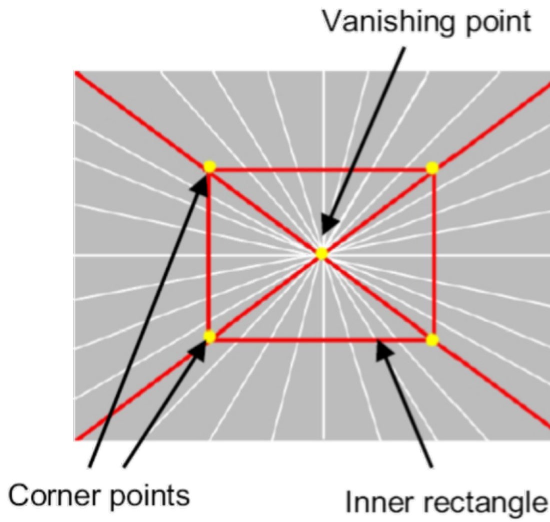
•Use camera transformations to navigate through the scene

The idea

- Many scenes (especially paintings), can be represented as an axis-aligned box volume (i.e. a stage)
- Key assumptions:
  - All walls of volume are orthogonal
  - Camera view plane is parallel to back of volume
  - Camera up is normal to volume bottom
- How many vanishing points does the box have?
  - Three, but two at infinity
  - Single-point perspective
- Can use the vanishing point
- to fit the box to the particular
- Scene!



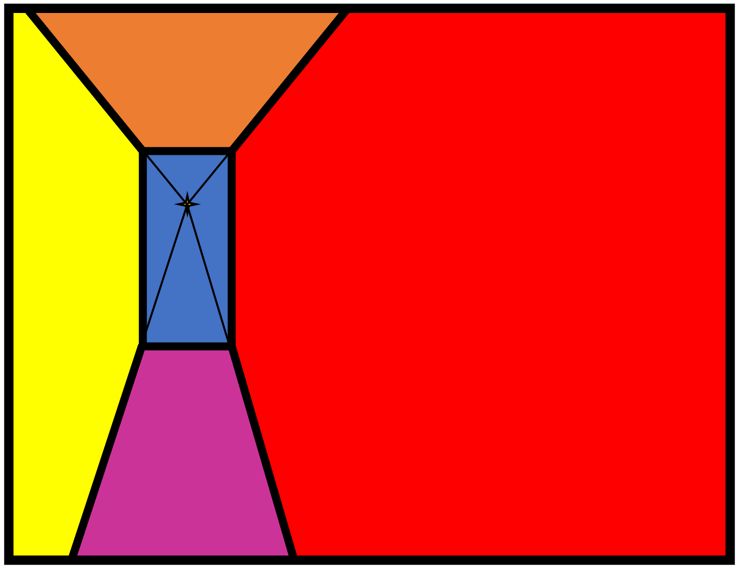
Fitting the box volume



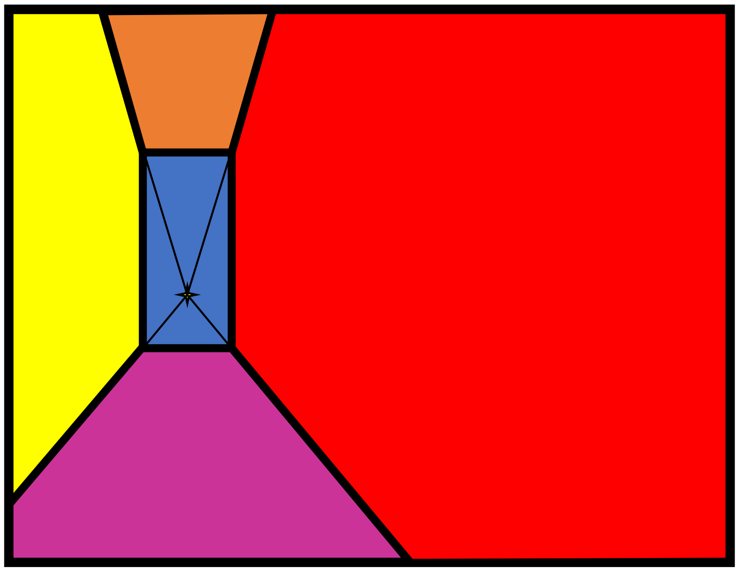
- User controls the inner box and the vanishing point placement (# of DOF???)

- Q: What's the significance of the vanishing point location?
- A: It's at eye level: ray from COP to VP is perpendicular to image plane.

Comparison of how image is subdivided based on two different camera positions. You should see how moving the vanishing point corresponds to moving the eyepoint in the 3D world.

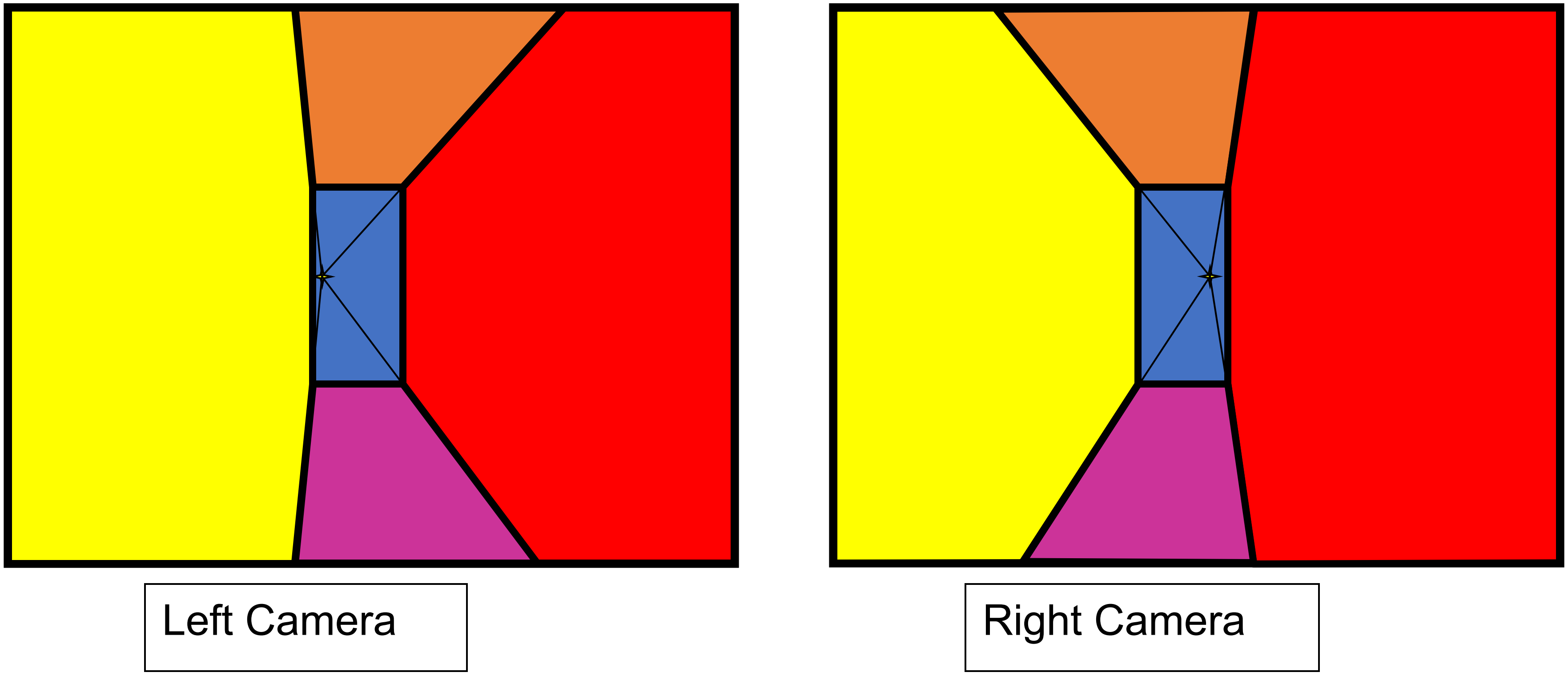


High Camera



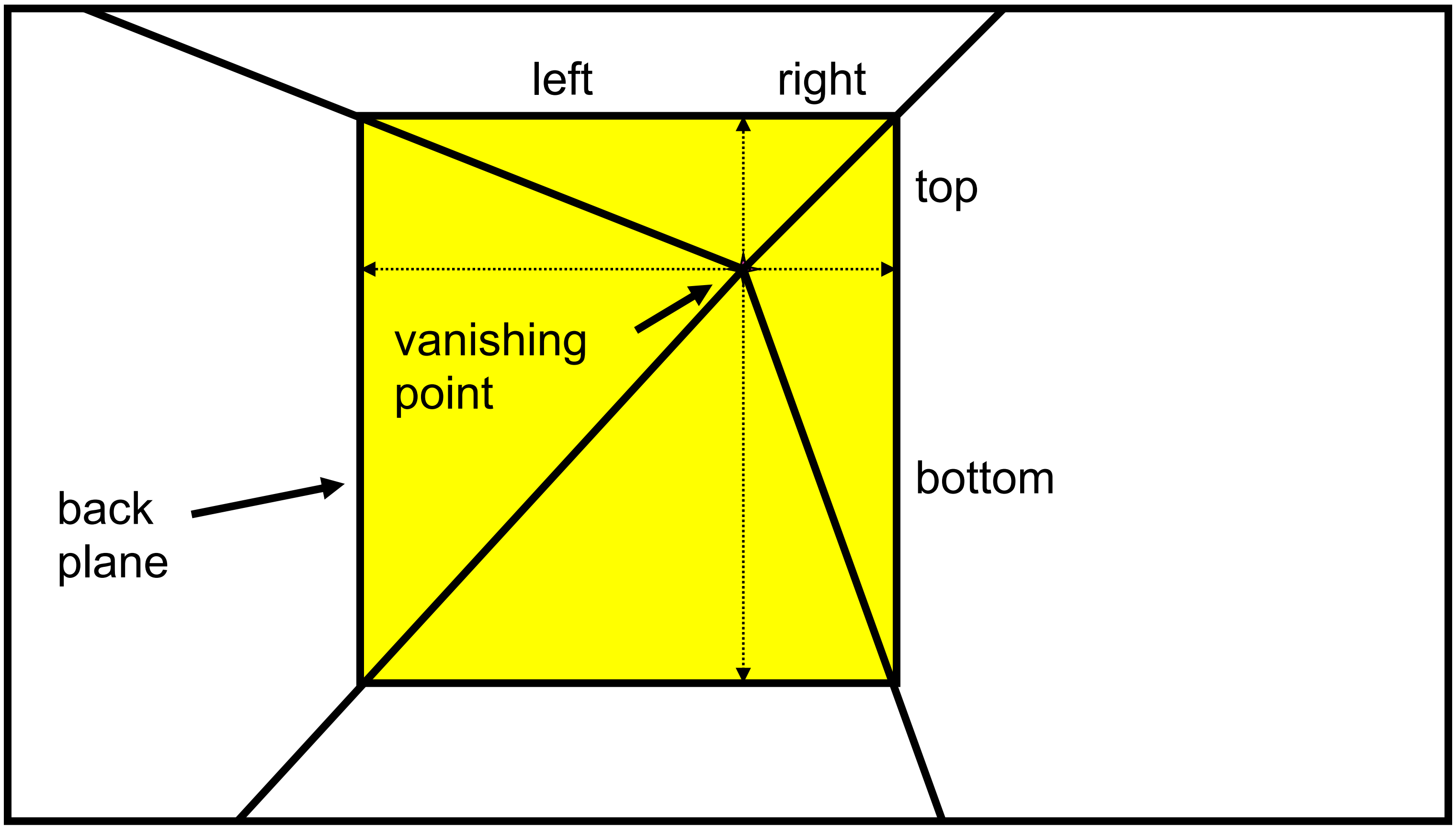
Low Camera

Comparison of two camera placements – left and right. Corresponding subdivisions match view you would see if you looked down a hallway.



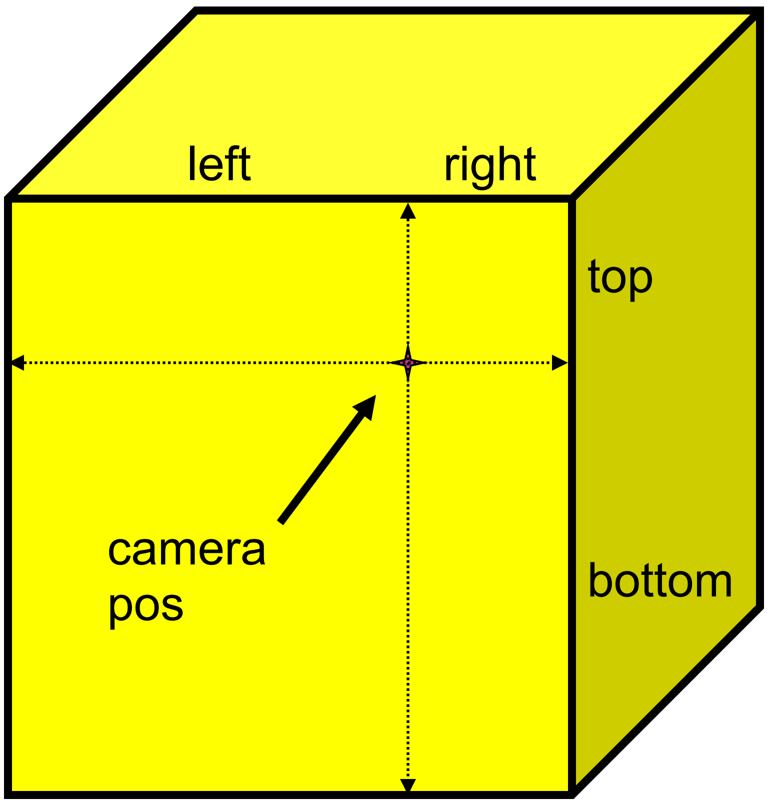
2D to 3D conversion

- First, we can get ratios

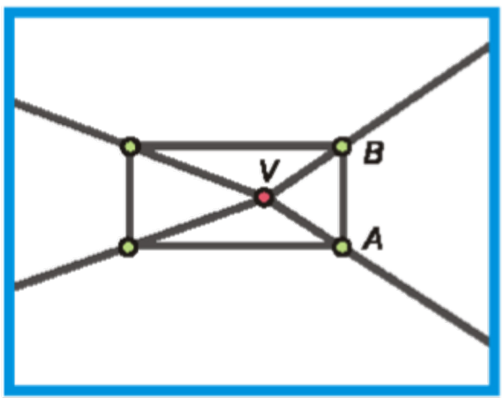


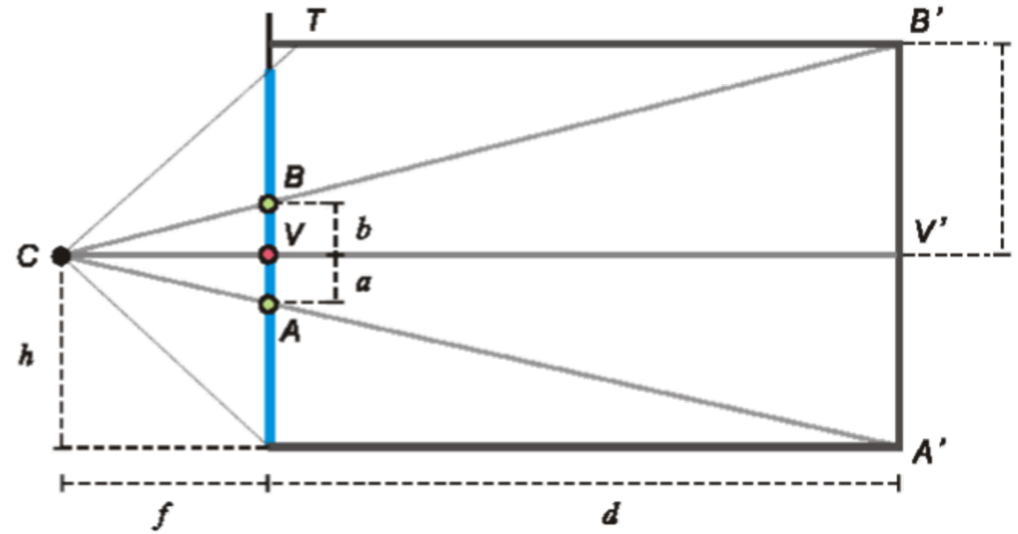
2D to 3D conversion

- Size of user-defined back plane must equal size of camera plane (orthogonal sides)
- Use top versus side ratio to determine relative height and width dimensions of box
- Left/right and top/bot ratios determine part of 3D camera placement



Depth of the box





- Can compute by similar triangles (CVA vs. CV'A')
- Need to know focal length f (or FOV)

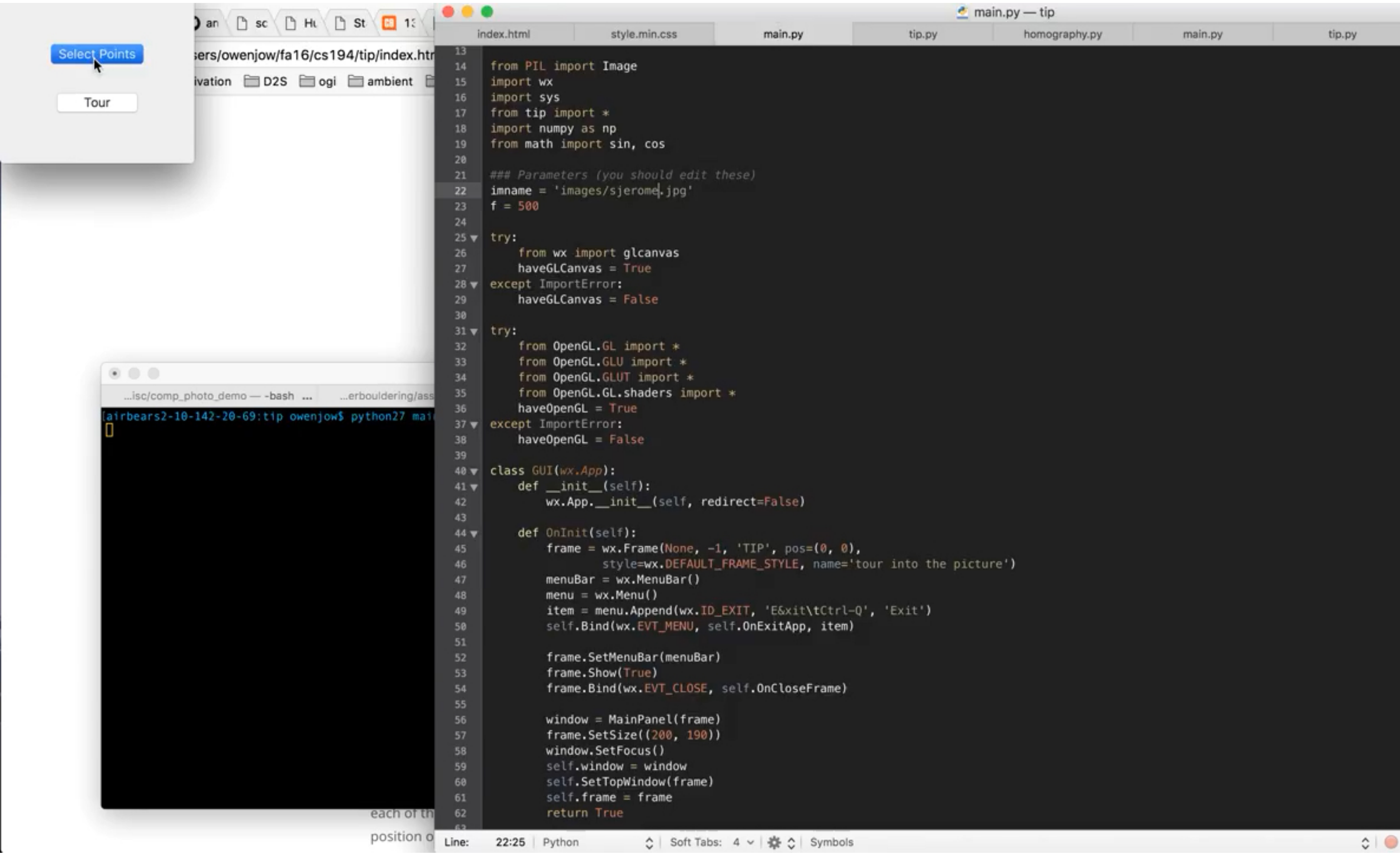
- Note: can compute position on any object on the ground
  - Simple unprojection
  - What about things off the ground?

DEMO

- Now, we know the 3D geometry of the box
- We can texture-map the box walls with texture from the image

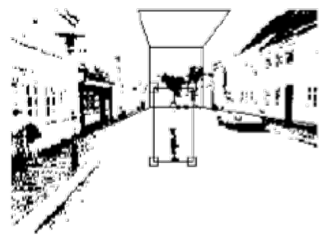
DEMO

- Now, we know the 3D geometry of the box
- We can texture-map the box walls with texture from the image

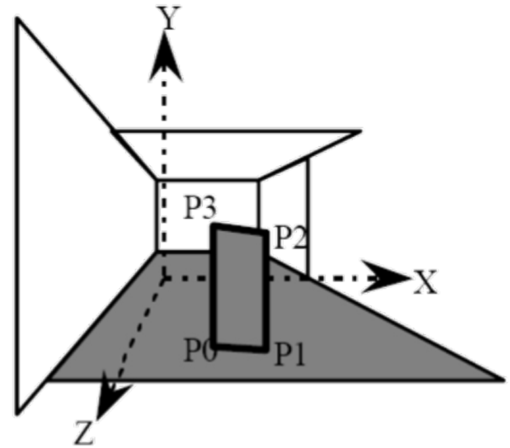

Select Points
Tour
sers/owenjow/fa16/cs194/tip/index.htm
ivation D2S ogi ambient
main.py — tip
index.html
style.min.css
main.py
tip.py
homography.py
main.py
tip.py
from PIL import Image
import wx
import sys
from tip import *
import numpy as np
from math import sin, cos
Parameters (you should edit these)
imname = 'images/sjerome.jpg'
f = 500
try:
    from wx import glcanvas
    haveGLCanvas = True
except ImportError:
    haveGLCanvas = False
try:
    from OpenGL.GL import *
    from OpenGL.GLU import *
    from OpenGL.GLUT import *
    from OpenGL.GL.shaders import *
    haveOpenGL = True
except ImportError:
    haveOpenGL = False
class GUI(wx.App):
    def __init__(self):
        wx.App.__init__(self, redirect=False)
    def OnInit(self):
        frame = wx.Frame(None, -1, 'TIP', pos=(0, 0),
                style=wx.DEFAULT_FRAME_STYLE, name='tour into the picture')
        menuBar = wx.MenuBar()
        menu = wx.Menu()
        item = menu.Append(wx.ID_EXIT, 'E&xit\tCtrl-Q', 'Exit')
        self.Bind(wx.EVT_MENU, self.OnExitApp, item)
        frame.SetMenuBar(menuBar)
        frame.Show(True)
        frame.Bind(wx.EVT_CLOSE, self.OnCloseFrame)
        window = MainPanel(frame)
        frame.SetSize((200, 190))
        window.SetFocus()
        self.window = window
        self.SetTopWindow(frame)
        self.frame = frame
        return True
...isc/comp_photo_demo — -bash
...erbouldering/ass
airbears2-10-142-20-69:tip owenjow$ python27 mai
Line: 22:25 Python Soft Tabs: 4 Symbols

Foreground Objects

- Add vertical rectangles for each foreground object
- Can compute 3D coordinates P0, P1 since they are on known plane.
- P2, P3 can be computed as before (similar triangles)



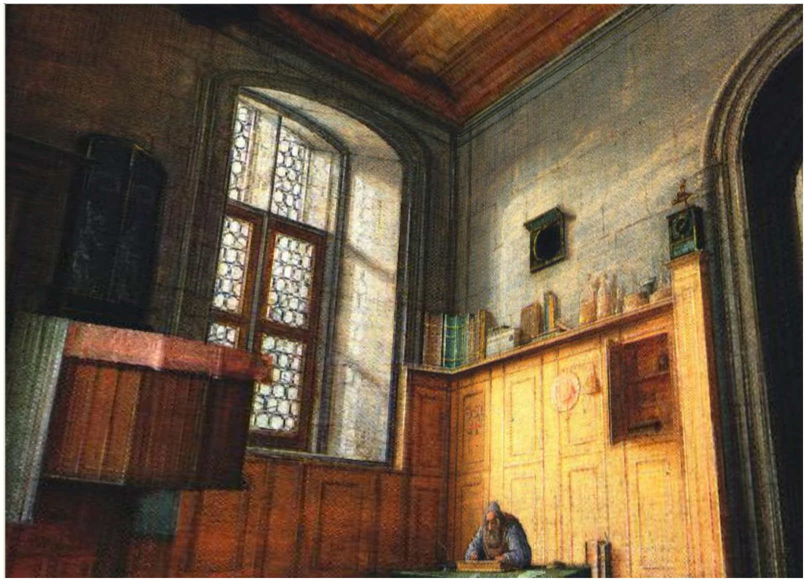
(a) Specifying of a foreground object



(b) Estimating the vertices of the foreground object model

(c) Three foreground object models

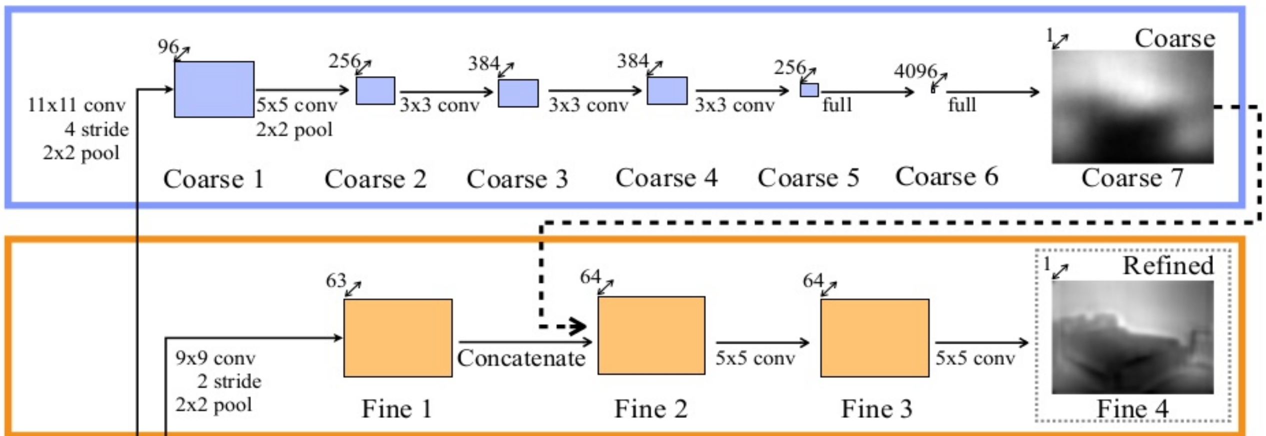
Foreground DEMO (and video)

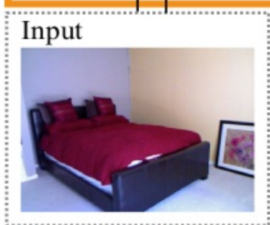


[redacted]

Single View Modeling using Learning

Depth estimation using CNNs





		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

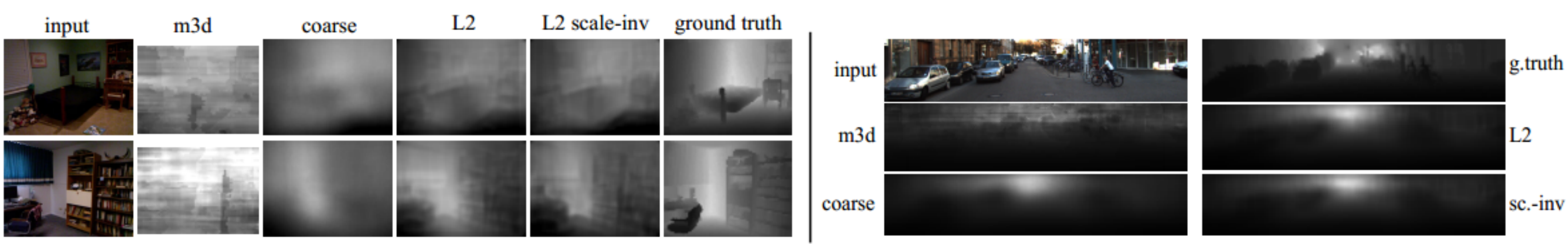
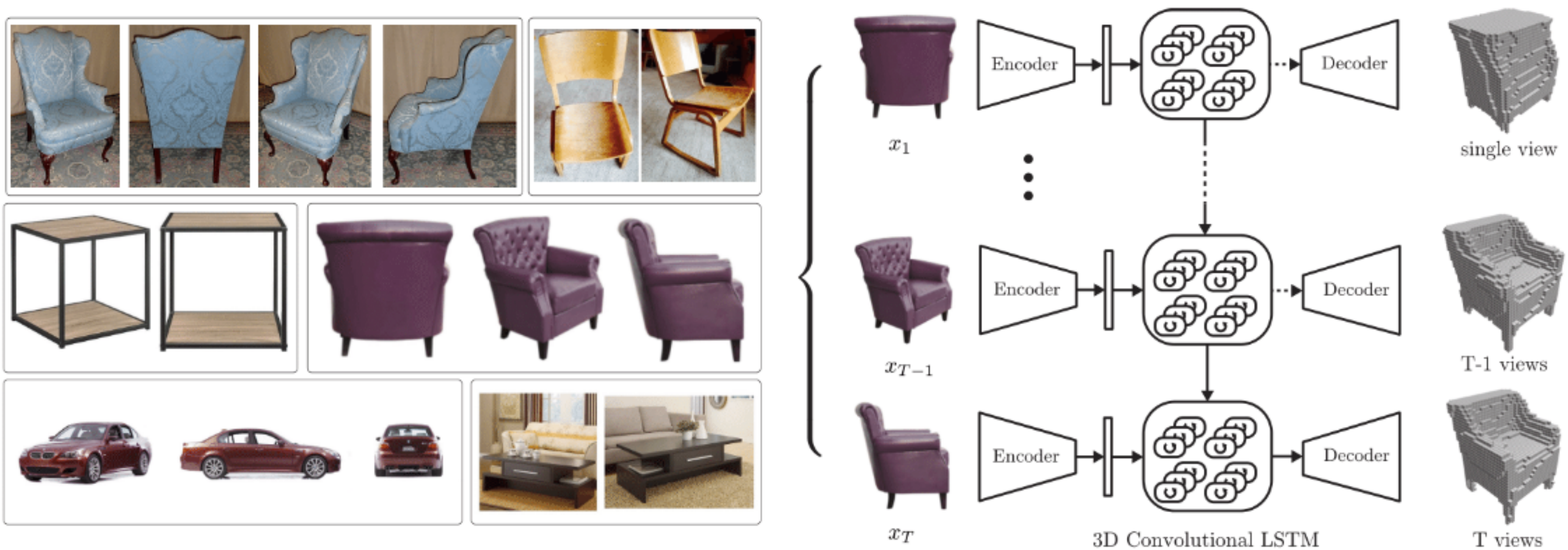


Figure 3: Qualitative comparison of Make3D, our method trained with l_2 loss ($\lambda = 0$), and our method trained with both l_2 and scale-invariant loss ($\lambda = 0.5$).

Depth Map Prediction from a Single Image using a Multi-Scale Deep. David Eigen, Christian Puhrsch, Rob Fergus. NIPS 2014. Cited by 1482

3D-R2N2

通过Encoder-3DLSTM-Decoder 的网络结构建立2D images -to -3D voxel model 的映射。



(a) Images of objects we wish to reconstruct (b) Overview of the network

3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, Silvio Savarese. ECCV 2016. Cited by 584.

Point Set Generation Network

主要思想：

利用深度网络通过单张图像直接生成点云，解决了基于单个图片对象生成3D几何的问题。

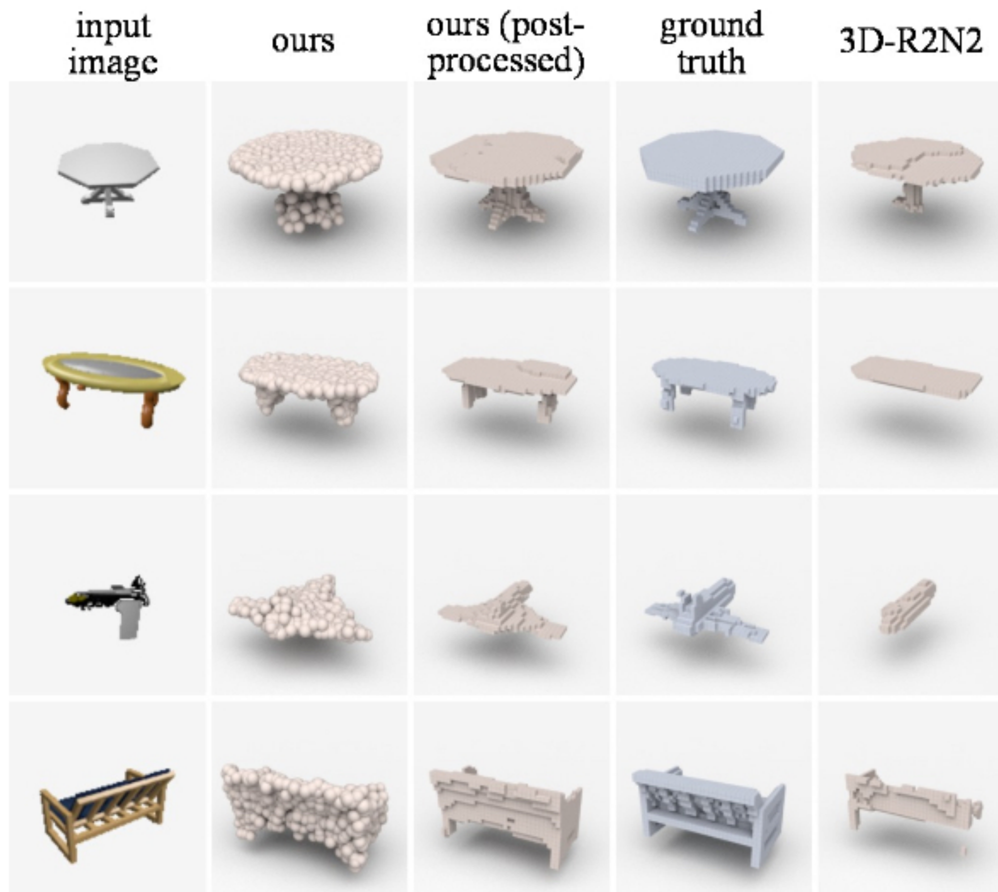


Figure 6. Visual comparison to 3D-R2N2. Our method better preserves thin structures of the objects.

A Point Set Generation Network for 3D Object Reconstruction from a Single Image, Haoqiang Fan, Hao Su, Leonidas Guibas. CVPR 2017. Cited by 476.)

Pixel2Mesh

主要思想：用一个椭球作为任意物体的初始形状，然后逐渐将这个形状变成目标物体。不借助点云、深度或者其他更加信息丰富的数据，而是直接从单张彩色图片直接得到 3D mesh

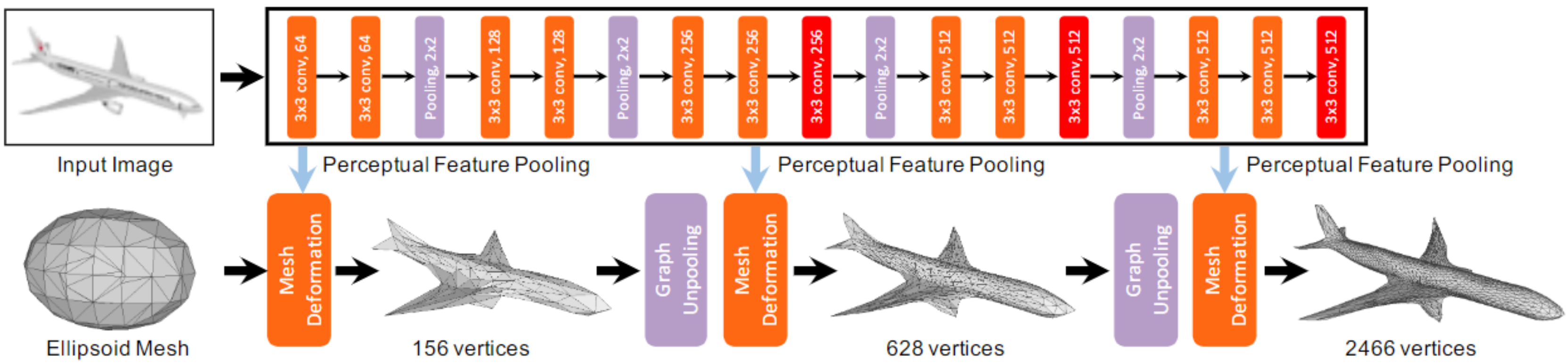


Fig. 2. The cascaded mesh deformation network. Our full model contains three mesh deformation blocks in a row. Each block increases mesh resolution and estimates vertex locations, which are then used to extract perceptual image features from the 2D CNN for the next block.

Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images，Nanyang Wang, Yinda Zhang, Zhuwen Li et.al. ECCV 2018, cited by 184.

Mesh R-CNN

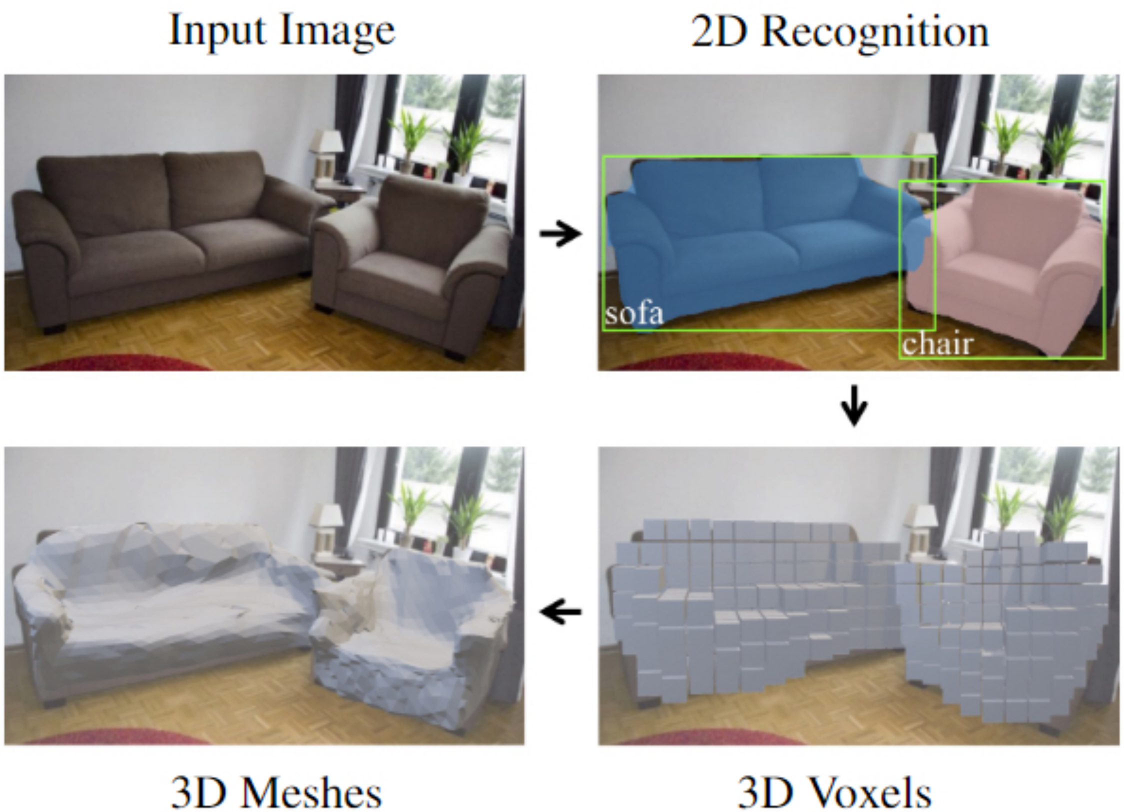


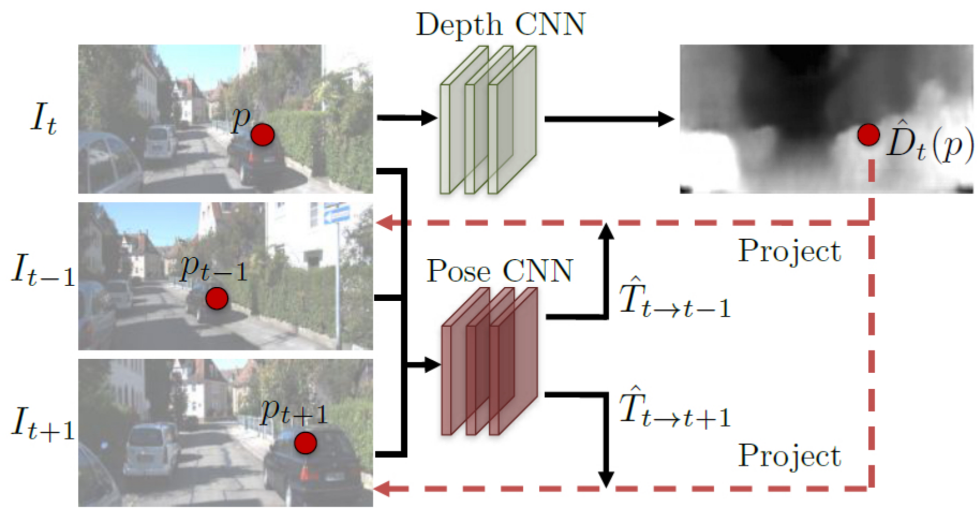
Figure 1. Mesh R-CNN takes an input image, predicts object instances in that image and infers their 3D shape. To capture diversity in geometries and topologies, it first predicts coarse voxels which are refined for accurate mesh predictions.

Mesh R-CNN是基于Mask R-CNN的增强网络，输入一个图像，检测图像中的所有对象，并输出所有对象的类别标签，边界框、分割掩码以及三维三角形网格。

Mesh R-CNN. Georgia Gkioxari, Jitendra Malik, Justin Johnson. 2020

Learning-based SFM

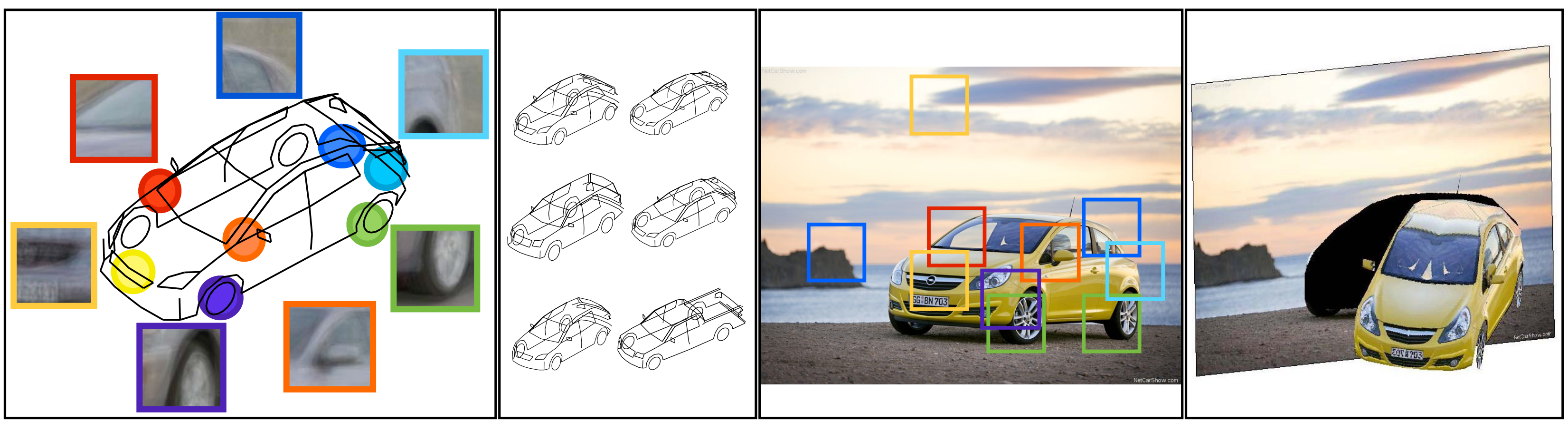
- Learning without ground-truth depth information
- Modeling the learning target with video sequences



Unsupervised Learning of Depth and Ego-Motion from Video.
Tinghui Zhou, Matthew Brown, Noah Snavely, David Lowe. In CVPR 2017

Single Image Pop-Up from Discriminatively Learned Parts

Menglong Zhu* Xiaowei Zhou* Kostas Daniilidis
Computer and Information Science, University of Pennsylvania
{menglong,xiaowz,kostas}@cis.upenn.edu

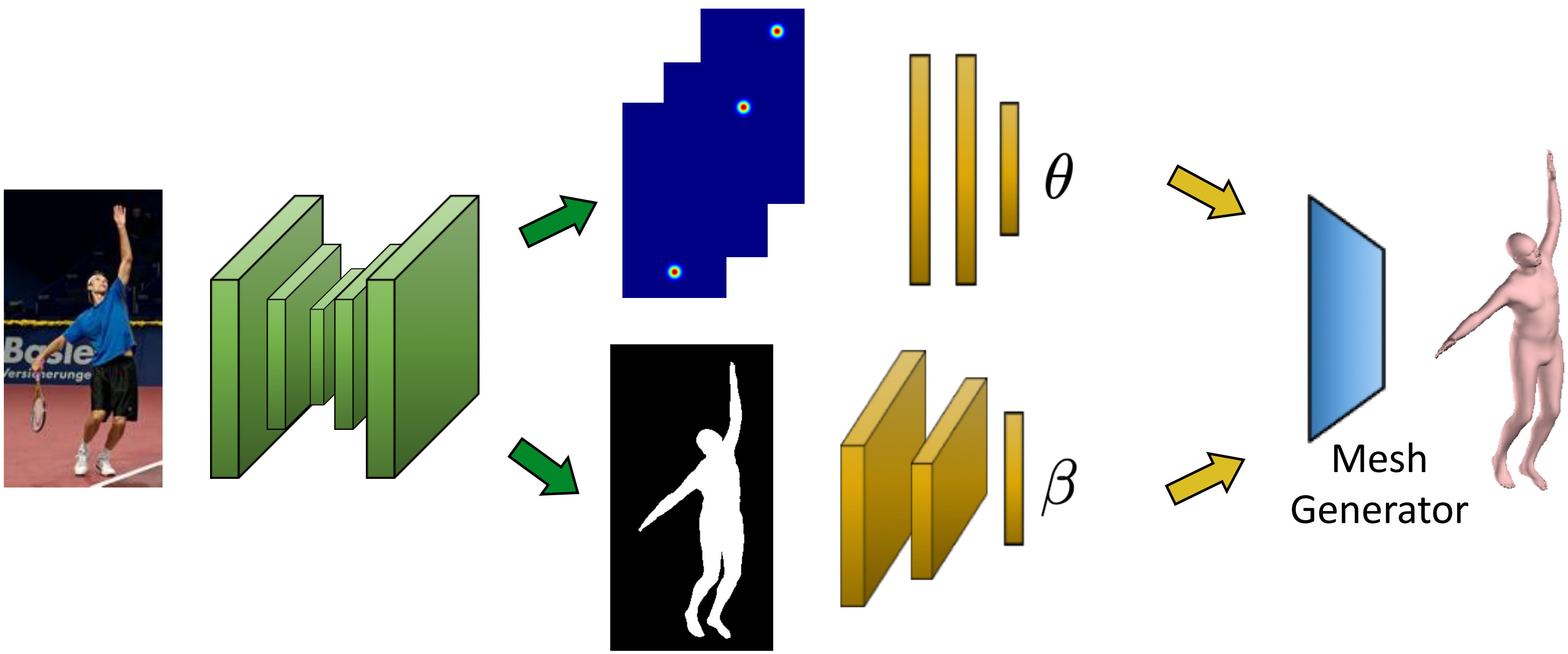


3D Shape Constrained Discriminative Parts + 3D Shape Space + Individual Part Detection Hypotheses = Pop-Up !


NetCarShow.com
GG BN 703
NetCarShow.com
NetCarShow.com

Learning to Estimate 3D Human Pose and Shape from a Single Color Image

Georgios Pavlakos[1], Luyang Zhu[2], Xiaowei Zhou[3], Kostas Daniilidis[1]
[1] University of Pennsylvania [2] Peking University [3] Zhejiang University




西瓜视频
100
西瓜视频
100

Questions?