Computer Vision (CS763)

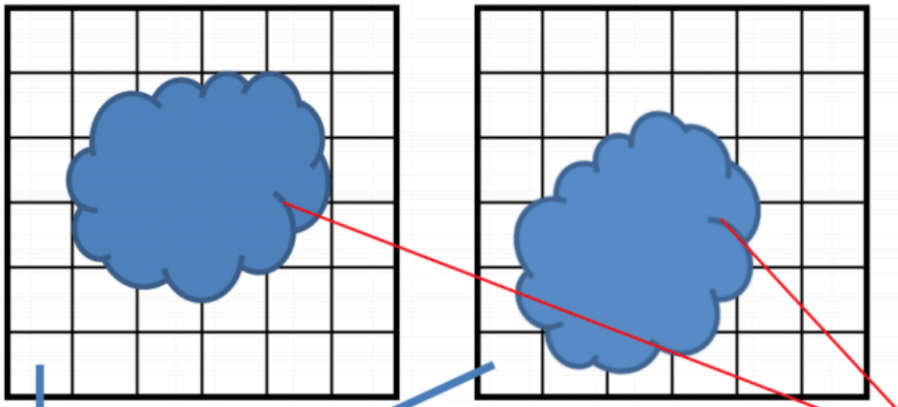
Teaching cameras to "see"

Image Alignment

Arjun Jain

Image Alignment

- Consider images I_1 and I_2 of a scene acquired through different viewpoints.



Pixels in **physical correspondence** (containing measurements of the same physical point, but not necessarily the same coordinate values in the image domain Ω)

Pixels in **digital correspondence** (same coordinate values in the image domain Ω, not necessarily containing measurements of the same physical point)

Slide courtesy Ajit Rajwade

https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/ImageAlignment.pdf

Image Alignment

- I_1 and I_2 are said to be aligned if for every (x,y) in the domain Ω, the pixels in I_1 and I_2 are in physical correspondence.

- Image **alignment** (also called **registration**) is the process of correcting for the relative motion between I_1 and I_2.

Slide courtesy Ajit Rajwade

https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/ImageAlignment.pdf

Basics

- A digital image – a discrete/sampled version of a visual stimulus
- Can be regarded as a function $I = f(x,y)$ where (x,y) are spatial (integer) coordinates in a domain $\Omega = [0,H\text{-}1] \times [0,W\text{-}1]$.
- Each ordered pair (x,y) is called a pixel.
- The pixel is generally square (sometimes rectangular) in shape.

Slide courtesy Ajit Rajwade

https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/ImageAlignment.pdf

Basics

- Pixel dimensions (height/width) relate to the spatial resolution of the sensor in the camera that collects light reflected from a scene.

Slide courtesy Ajit Rajwade

https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/ImageAlignment.pdf

Basics

- In a typical grayscale image, the intensity values *f(x,y)* lie in the range from 0 to 255 (8 bit integers).
- They are quantized versions continuous values corresponding to the actual light intensity that strikes a light sensor in the camera.

Slide courtesy Ajit Rajwade

https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/ImageAlignment.pdf

Image Alignment - Steps

- First Step: Motion Estimation
  1. Feature point based
  2. Direct pixel value based

- Second Step: Image Warping

Slide courtesy Ajit Rajwade

https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/ImageAlignment.pdf

Step 1: Motion Estimation

2D Motion Models

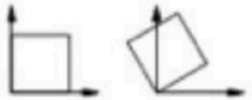
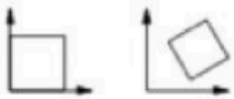
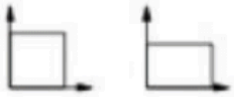
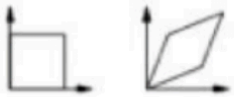
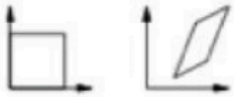
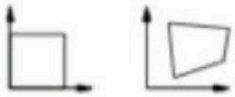
2D Transformation	Figure	d. o. f.	H	H
Translation		2	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} I & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$
Mirroring at y-axis		1	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} Z & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$
Rotation		1	$\begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$
Motion		3	$\begin{bmatrix} \cos\varphi & -\sin\varphi & t_x \\ \sin\varphi & \cos\varphi & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$

Image courtesy: Schindler

2D Motion Models

Similarity		4	$\begin{bmatrix} a & -b & t_x \\ b & a & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \lambda R & t \\ \mathbf{0}^T & 1 \end{bmatrix}$
Scale difference		1	$\begin{bmatrix} 1+m/2 & 0 & 0 \\ 0 & 1-m/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} D & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$
Shear		1	$\begin{bmatrix} 1 & s/2 & 0 \\ s/2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} S & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$
Asym. shear		1	$\begin{bmatrix} 1 & s' & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} S' & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$
Affinity		6	$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} A & t \\ \mathbf{0}^T & 1 \end{bmatrix}$
Projectivity		8	$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$	$\begin{bmatrix} A & t \\ \boldsymbol{p}^T & 1/\lambda \end{bmatrix}$

2D Homography

Image courtesy: Schindler

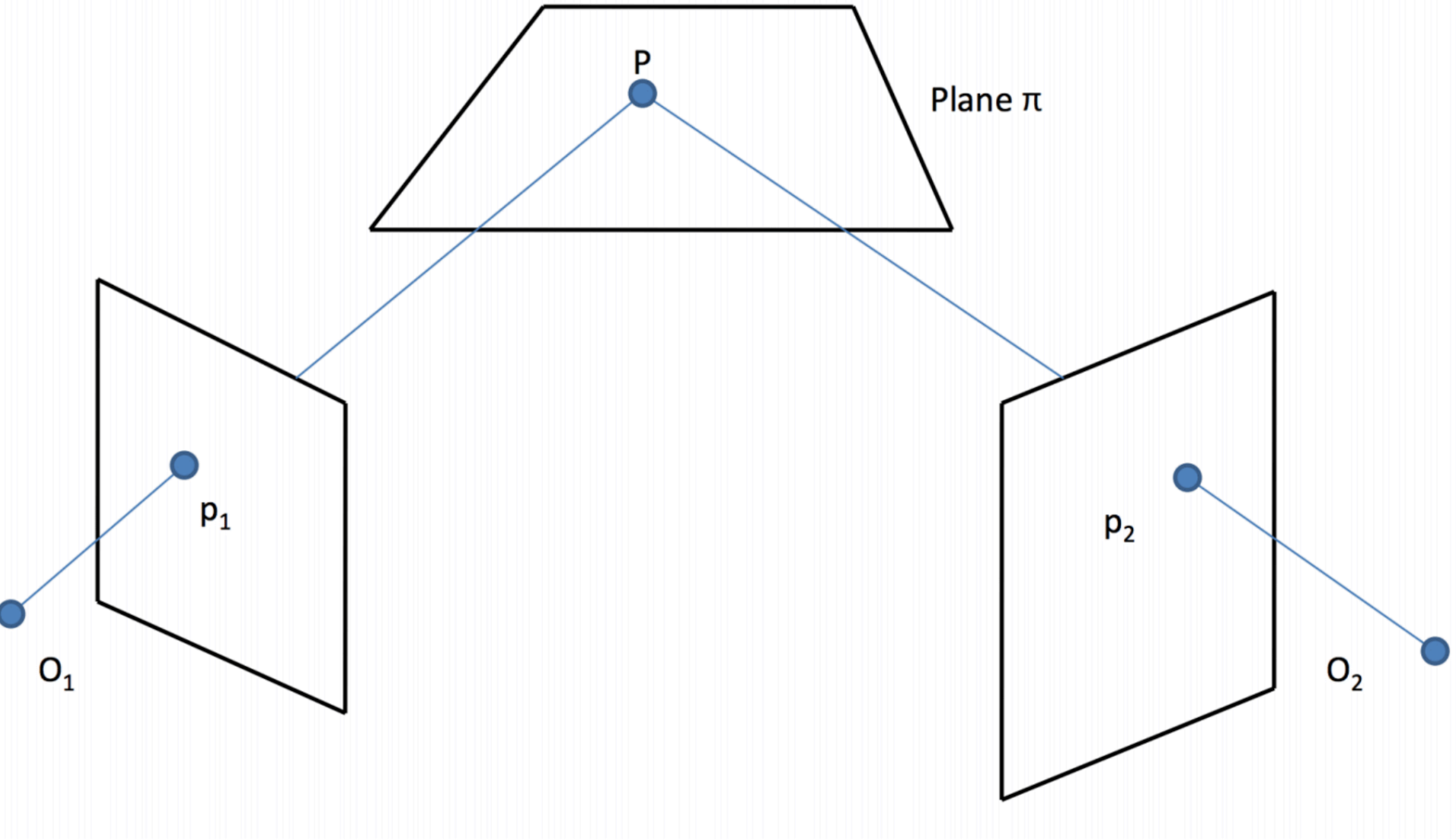
2D Motion Models

- Consider two images (perspective projection) of a plane in 3D space.

- These images are said to be related by a motion model called as a 2D homography

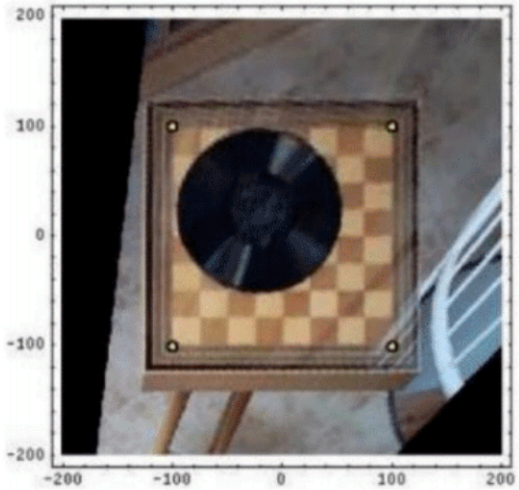
Slide courtesy Ajit Rajwade

https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/ImageAlignment.pdf



Slide courtesy Ajit Rajwade

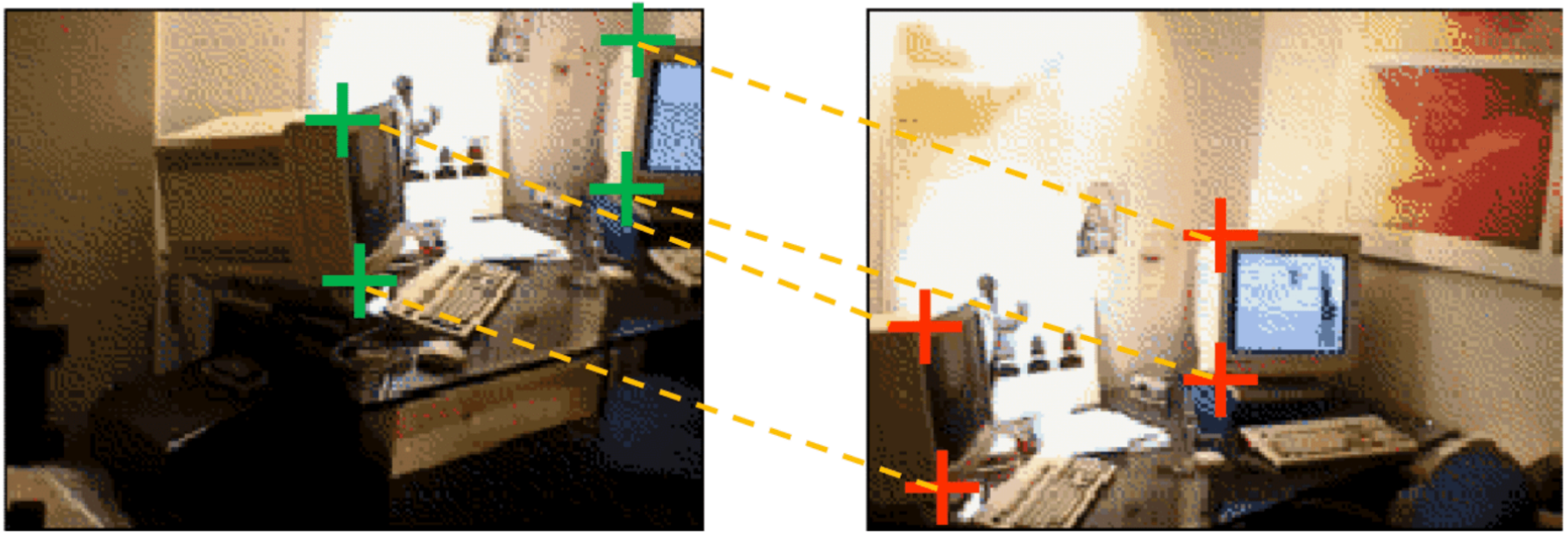
https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/ImageAlignment.pdf



Slide courtesy Ajit Rajwade

https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/ImageAlignment.pdf

Alignment with Control Points



Alignment with Control Points - Output

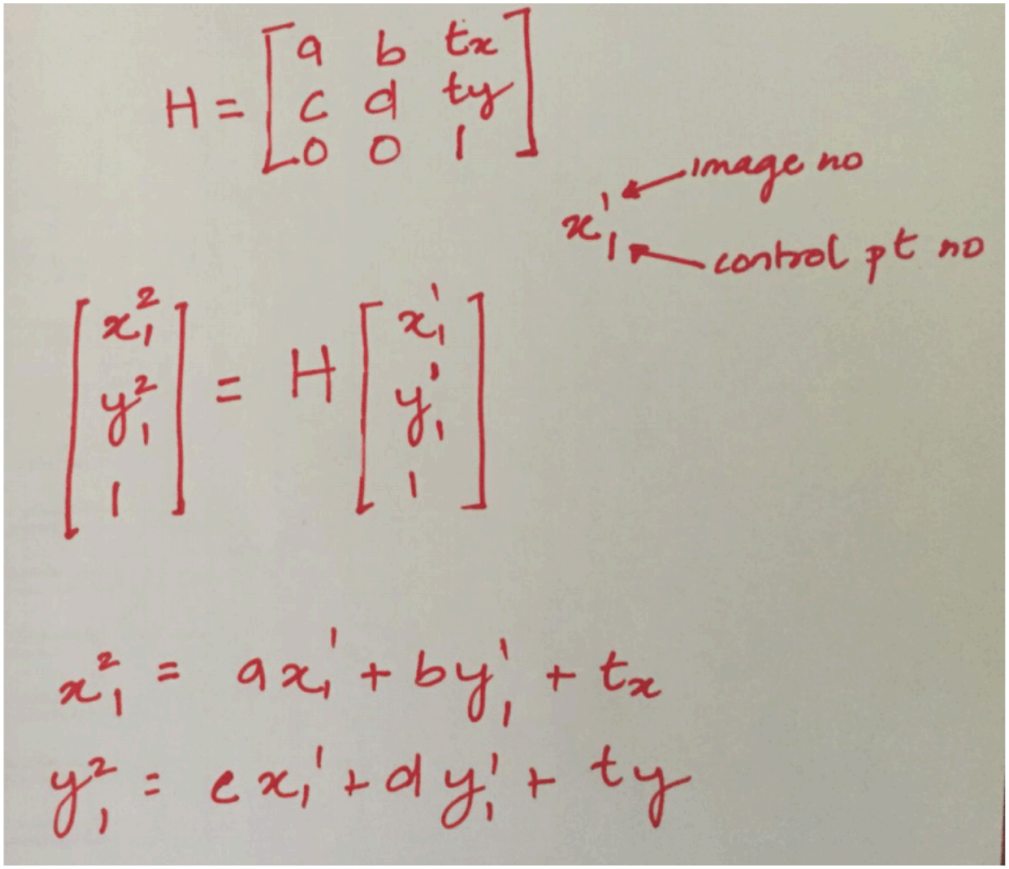
Alignment with Control Points

- Control points: pairs of physically corresponding points – maybe marked out manually, or automatically (we will see how)

- Given a motion model, how many 2D control points needed? (say we have **u** parameters)

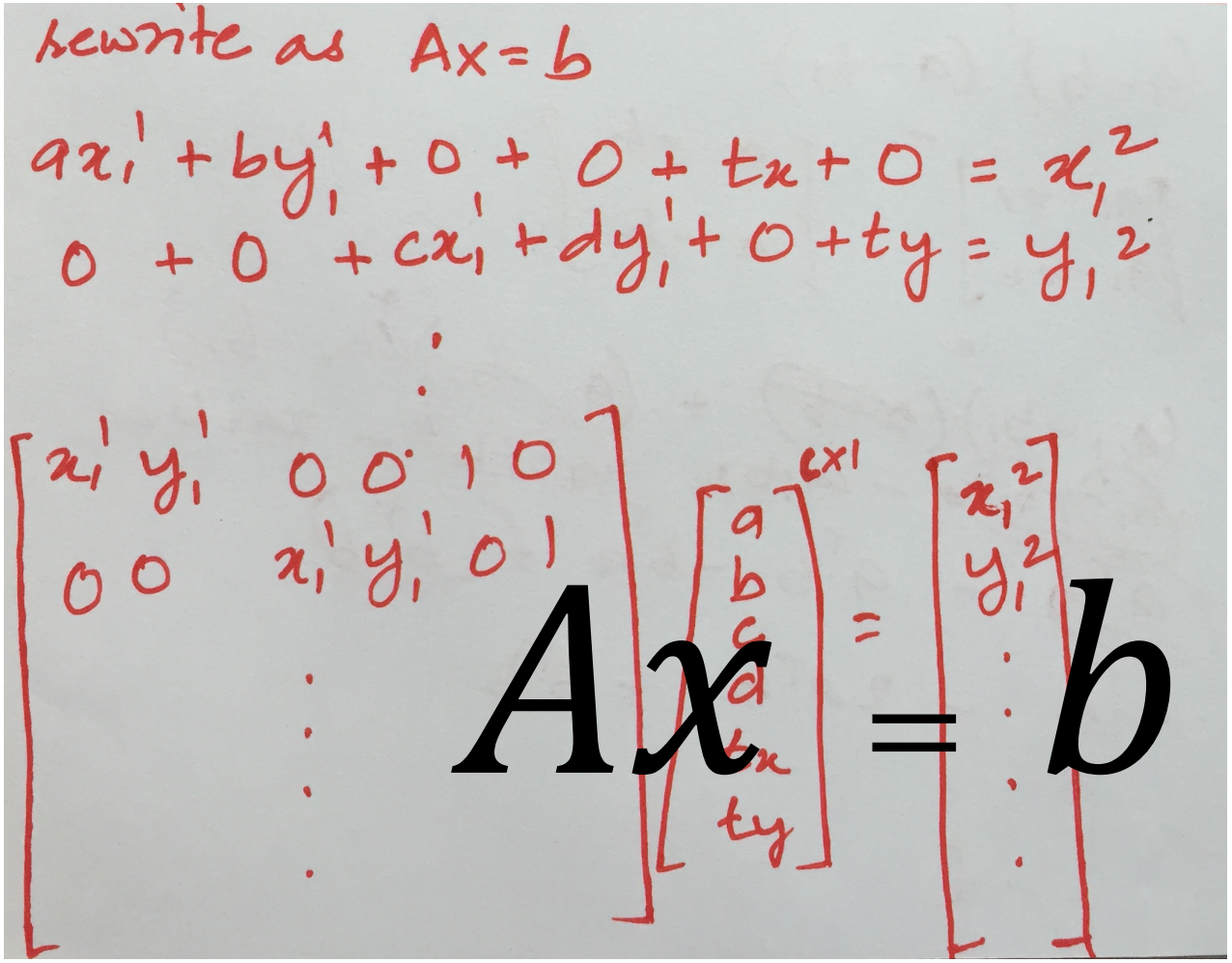
>= u/2, where u = number of unknown parameters in the motion model as each 2D point gives us 2 equations

Least Squares Motion Estimation: Affine



$$\begin{bmatrix} A & \boldsymbol{t} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix}$$

Least Squares Motion Estimation: Affine



Least Squares Motion Estimation: Affine

- Can not directly take by $x = A^{-1}b$ why?
- Solve using least squares:

  $argmin_x \parallel Ax - b \parallel^2 = (Ax - b)^T(Ax - b)$

  $Let\ f(x) = (Ax - b)^T(Ax - b)$
- To find x that minimizes $f(x)$, lets set $f'(x) = 0$

Least Squares Motion Estimation: Affine

- $f(x) = (Ax - b)^T(Ax - b)$
  $f(x) = (x^T A^T - b^T)(Ax - b)$
  $f(x) = x^T A^T Ax \; - x^T A^T b - b^T Ax - b^T b$

- $f'(x) = 2A^T Ax - 2A^T b$

- Set $f'(x) = 0$ i.e. $x = (A^T A)^{-1} A^T b$
- What about $f''(x)$?

Least Squares Motion Estimation: Homography

- We have seen how to estimate the homgraphy matrix when doing Zhang's calibration (also homework)

Least Squares Motion Estimation: Rigid Motion

- What does it mean – Rigid Motion?
- Like affine but a,b,c,d have to form a Rotation matrix, so simple pseudo-inverse with $x = (A^T A)^{-1} A^T b$ will not work
- Detailed by Prof. Ajit Rajwade below assuming that t = 0 (the solution can be extended to the case where t is not 0)

http://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/procrustes.pdf

Also see:

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

Other Motion Models: Non-Rigid

- Higher degree polynomials, non-rigid models (example: motion of an amoeba, motion of the heart during the cardiac cycle, facial expressions, etc.)
- Per-pixel (DOF?)
- Piecewise affine, linear, spline-based (thin, B, cubic), etc.
- Will look into Optical Flow, etc. in future classes

Marking Control Points

- Control points are also called key-points or feature points. Marking feature points in an image can be automated using key-point detectors

- Properties of key-points detected:
  - Should have high **repeatability** under geometric transformations
  - Should be discriminative from its neighborhood. E.g. Corners, edges

- An excellent example of a detector is SIFT(Scale Invariant Feature Transform). www.cs.ubc.ca/~lowe/keypoints/

DOG to detect points

- Extrema's over Laplacian of Gaussian ($\sigma^2\nabla^2 G$) of the image used for detection. Here,

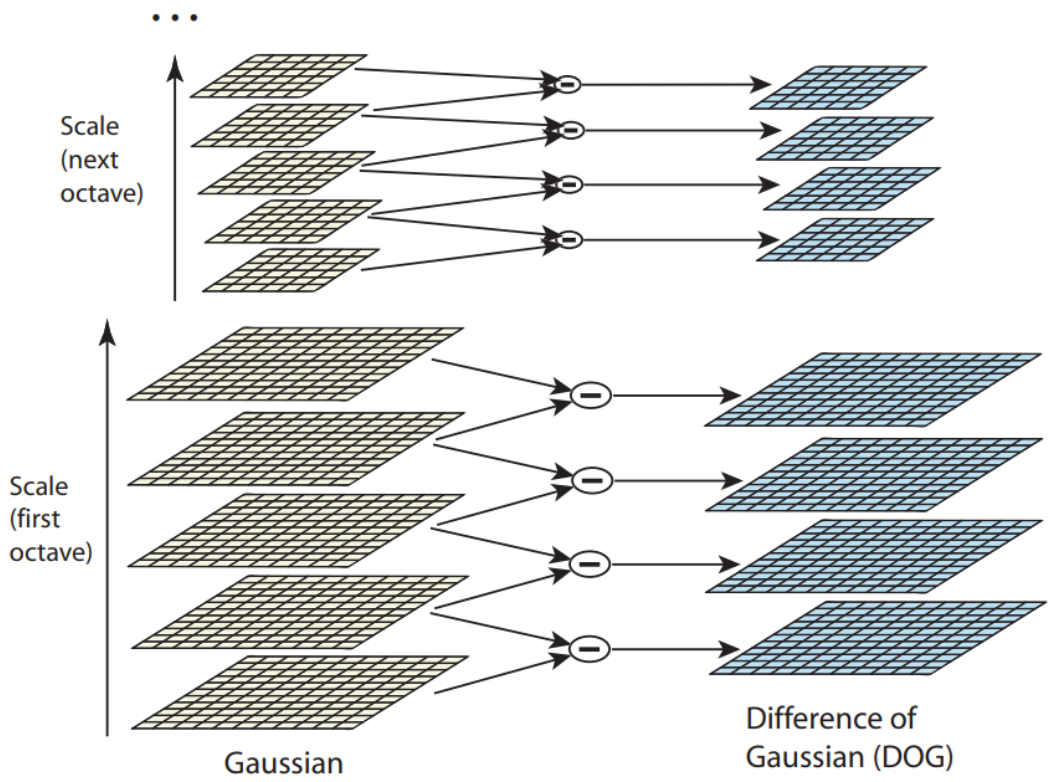
$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

- Laplacian of Gaussian (LoG) can be approximated by Difference of Gaussian (DoG),

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

- Check figure in next slide.

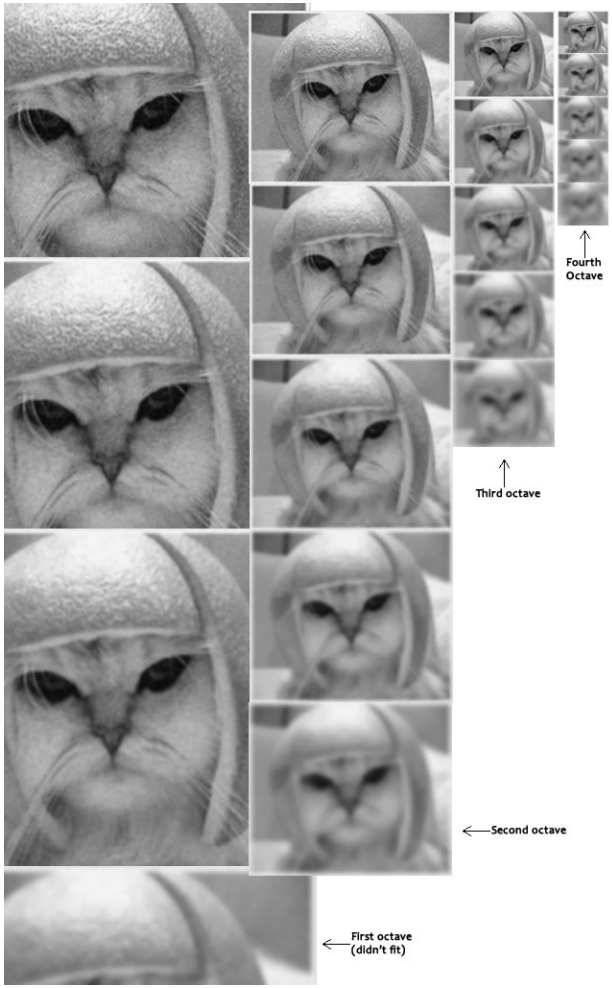
DOG from Gaussian Pyramid



- Shows a Gaussian Pyramid on the left
- Each layer in an octave(layer of the pyramid) is convolved with a Gaussian of k * (sigma of previous layer)

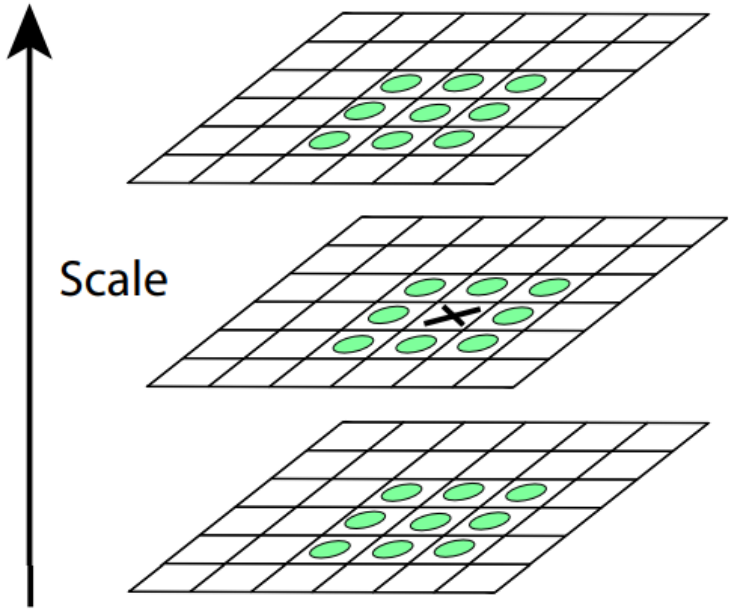
(Image Courtesy: Lowe et al. IJCV 2004)

SIFT Octaves



(Image Courtesy: http://aishack.in)

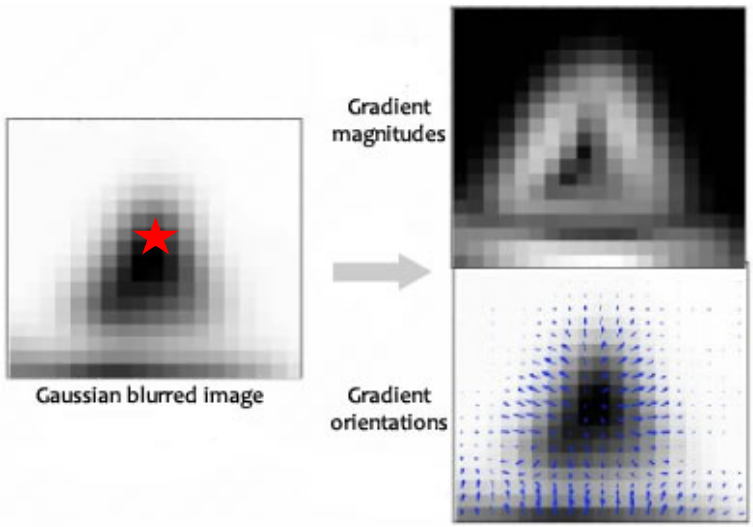
Marking Control Points



- Extremum from selected from the neighboring 26 values are selected as key-points

(Image Courtesy: Lowe et al. IJCV 2004)

Orientation Estimation



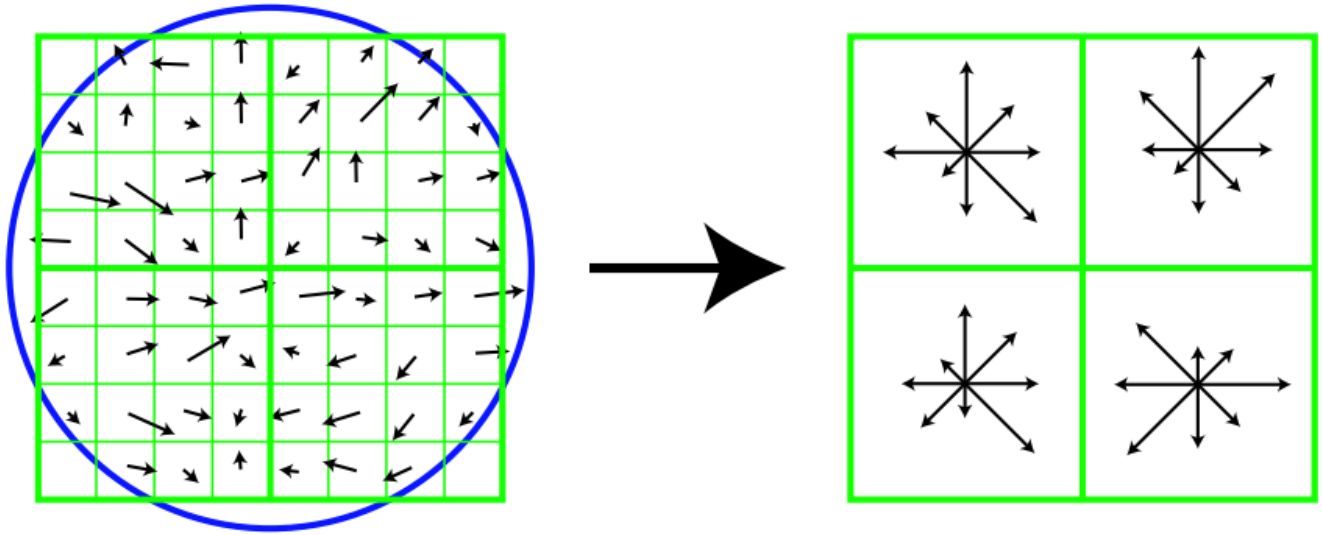
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

- Support window size is 1.5*sigma.
- Orientations of neighbor points are binned into a uniform spaced histogram. The value of bin with the highest frequency is assigned as the orientation.

(Image Courtesy: http://aishack.in)

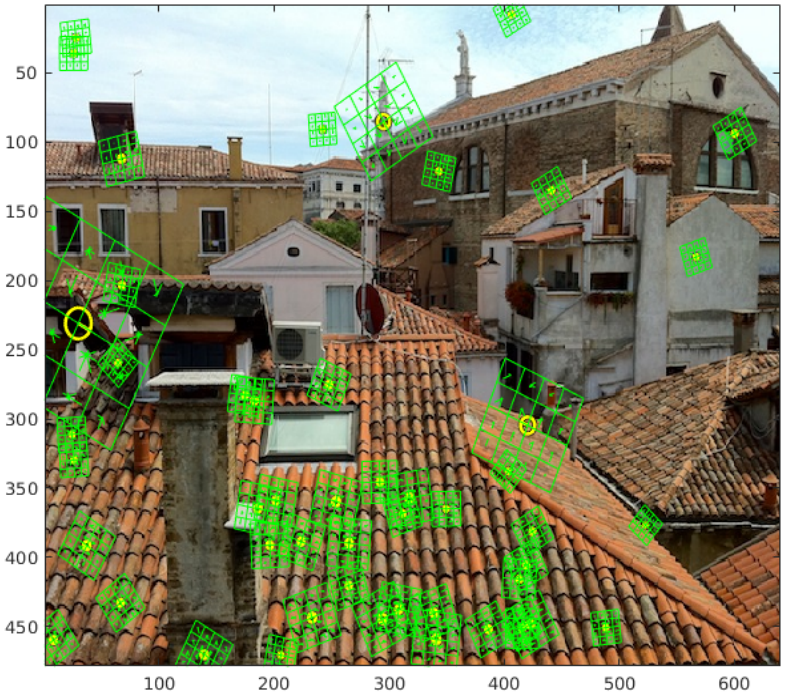
SIFT Descriptor From Control Points



- Support window size is scale normalized to 16 x 16 pixels. (Image shows 8 x 8).
- Support window is divided in 4 x 4 grid. Histogram of gradients weighted by Gaussian filter (blue circle) for each grid stacked forms the descriptor.

(Image Courtesy: Lowe et al. IJCV 2004)

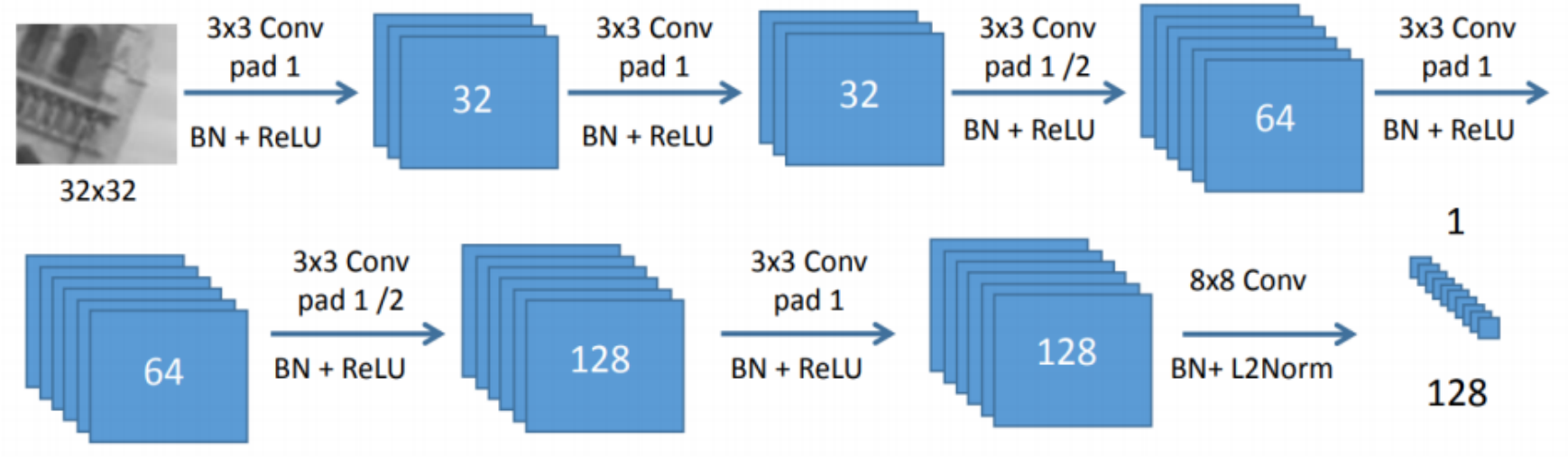
Examples of Detected Control Points



- Right image is annotated with key points. Yellow circle provides location and green grids are its support window.

(Image Courtesy: Lowe et al. IJCV 2004)

CNN Descriptor From Control Points



- A **scale-rotation normalized** patch is taken around the point.
- Output of the network is vector just like SIFT.

CNN Descriptor From Control Points

- Network is trained using triplets of patches {**a**, **p**, **n**}. Here **a** and **p** forms correspondence while **n** is not.

$$\delta_+ = ||f(\boldsymbol{a}) - f(\boldsymbol{p})||_2 \qquad \delta_- = ||f(\boldsymbol{a}) - f(\boldsymbol{n})||_2$$

$$\lambda(\delta_+, \delta_-) = max(0, \mu + \delta_+ - \delta_-)$$

- μ = 1.0. Since the descriptor has unit norm, the loss not only ensures separation between {**p**, **n**} but also distance between {**a**, **p**} at most 1.

Estimating Motion Model

- A combination of transformation invariant detector and descriptor can be used to match salient points from one image to another.

- A motion model can then be estimated from the matched points (say Homography).

Step 2: Image Warping

Image Warping

- After motion estimation, the next step is to warp one of the images (say) I_1 using the estimated motion represented by a matrix (say **T**).

Image Warping

- Forward warping:

  -Apply the motion **Tv** to every coordinate vector **v** = [x_1 y_1 1] in the original image (i.e. I_1).

  -Copy the intensity value from **v** in I_1 to the new location (**Tv**) in the warped image. If the new location is not an integer (most likely), then round off to nearest integer.

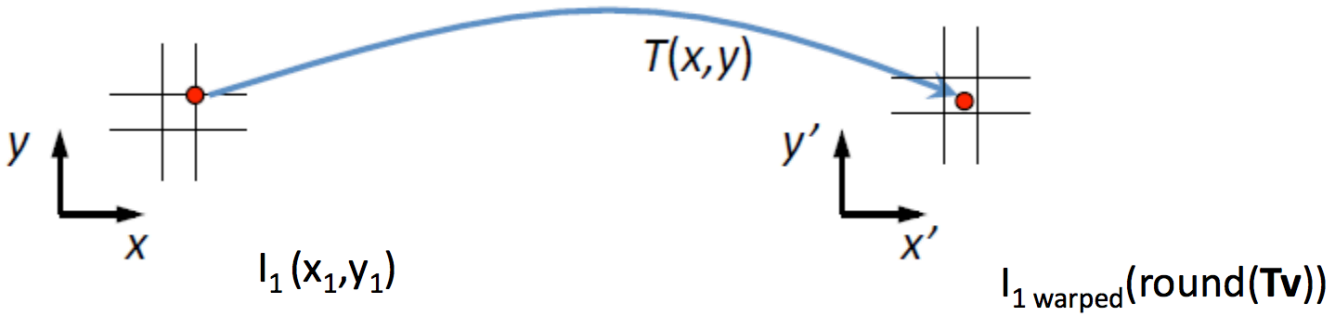


Image Warping

- Forward warping:

-Can leave the destination image with some holes if you scale up.

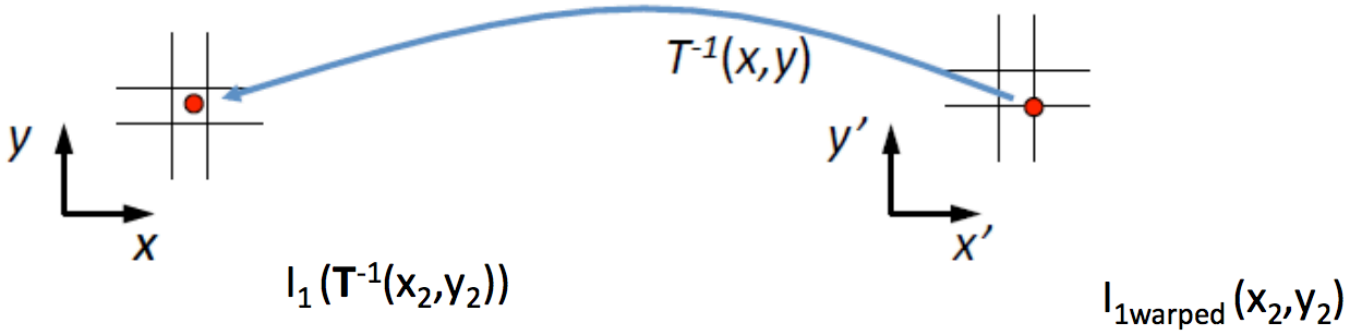
-Can lead to multiple answers in one pixel if you scale down.

Image Warping

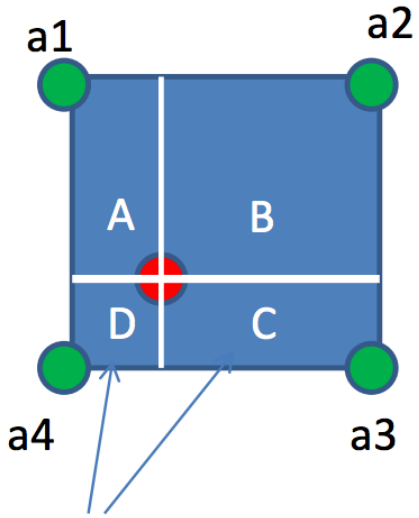
Note: this assumes that **T** is an invertible transformation, which is guaranteed in the applications we are dealing with. It is **not** guaranteed in case of non-rigid deformations, given the pixelized (i.e. discrete) image coordinate systems.

- Reverse warping:

  -For every coordinate $\mathbf{v} = [x_2\ y_2\ 1]$ in the destination image, copy the intensity value from coordinate $\mathbf{T}^{-1}\mathbf{v}$ in the original image. In case of non-integer value, perform interpolation (nearest neighbor or bilinear)



Interpolation



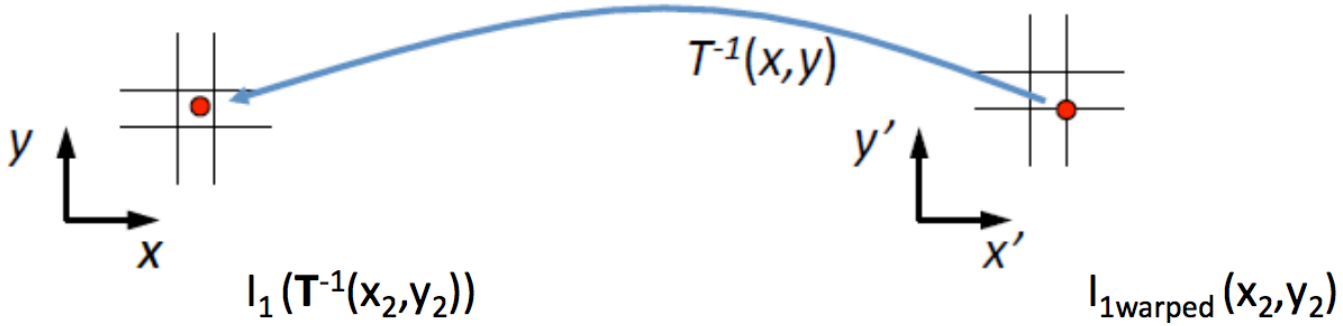
Nearest neighbor method:

- Use value a_4 (as the pixel that is nearest to the red point contains value a_4)

Bilinear method:

- Use the following value, a weighted combination of the four neighboring pixel values, with more weight to nearer values:

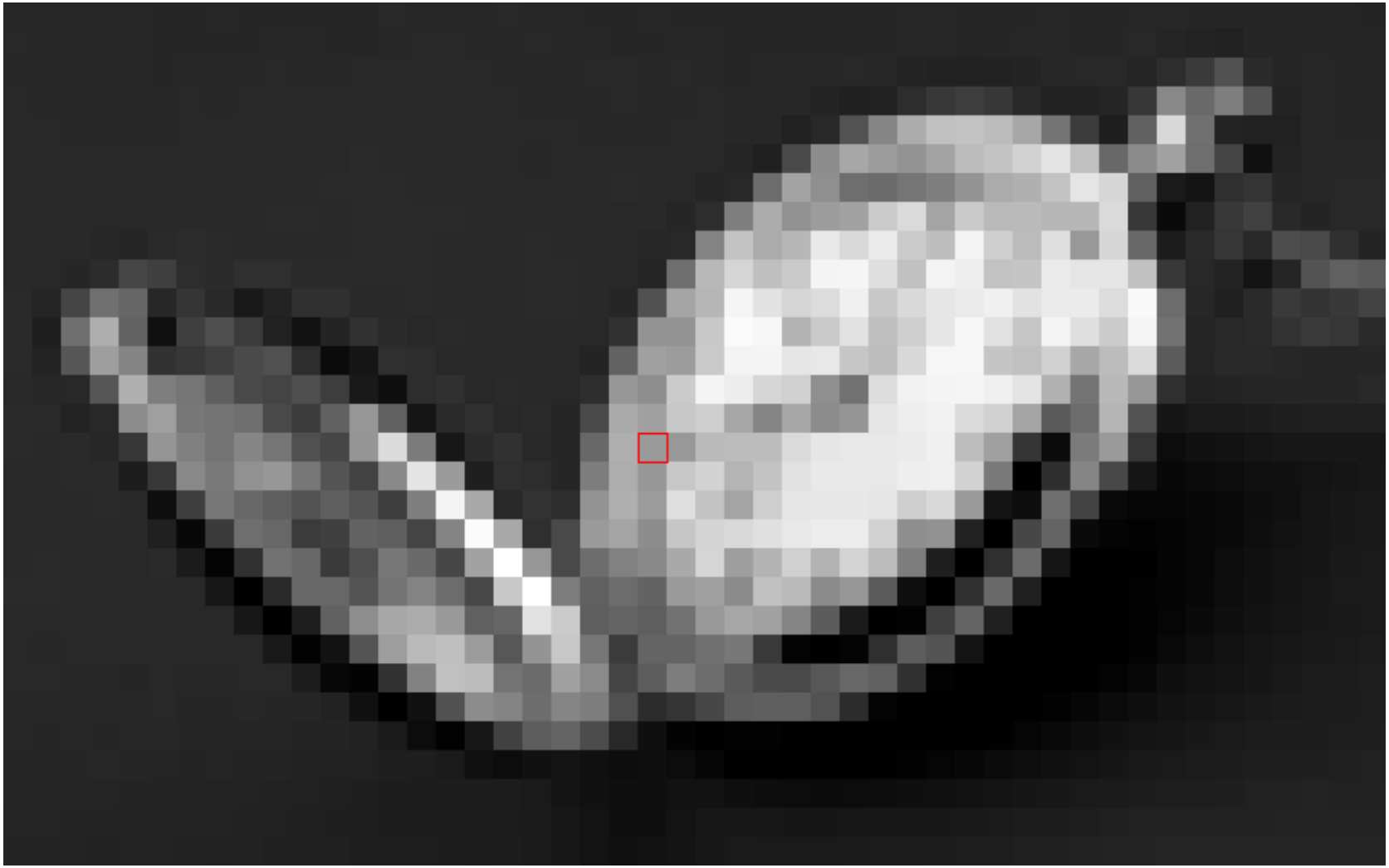
$$\frac{Ba_4 + Aa_3 + Ca_1 + Da_2}{(A+B+C+D)} = Ba_4 + Aa_3 + Ca_1 + Da_2$$



Interpolation

- Suppose we want to scale (zoom) this image 16x

Interpolation - NN



Slide: Prof. Noah Snavely
CS1114 http://www.cs.cornell.edu/courses/cs1114

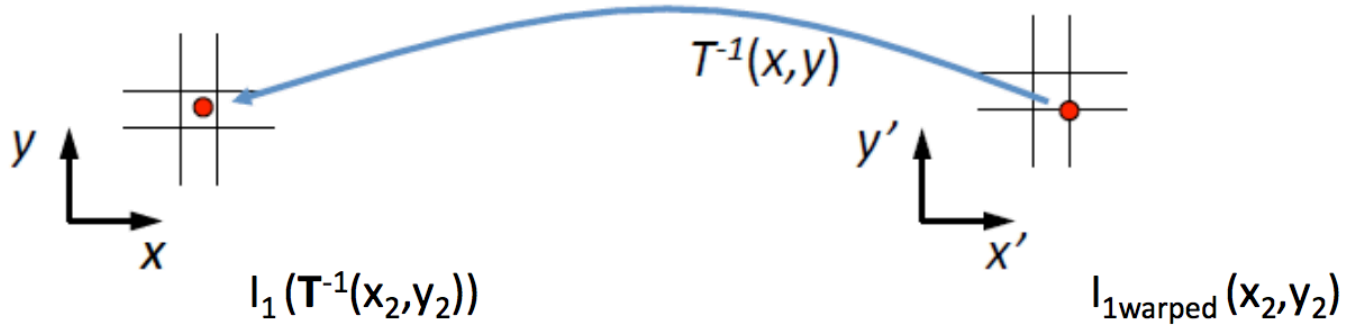
Interpolation - Bilinear

Slide: Prof. Noah Snavely
CS1114 http://www.cs.cornell.edu/courses/cs1114

Warping with 2D Homograpy

- Loop over all pixels in second image and interpolate from first image (homework)

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$



Slide Information

- The slide template has been created by Cyrill Stachniss (cyrill.stachniss@igg.uni-bonn.de) as part of the photogrammetry and robotics courses.
- A lot of material from Ajit Rajwade's CS763 course
- **I tried to acknowledge all people from whom I used images or videos. In case I made a mistake or missed someone, please let me know.**
- Thanks to Rahul Mitra for slides 24 to 30

Arjun Jain, ajain@cse.iitb.ac.in