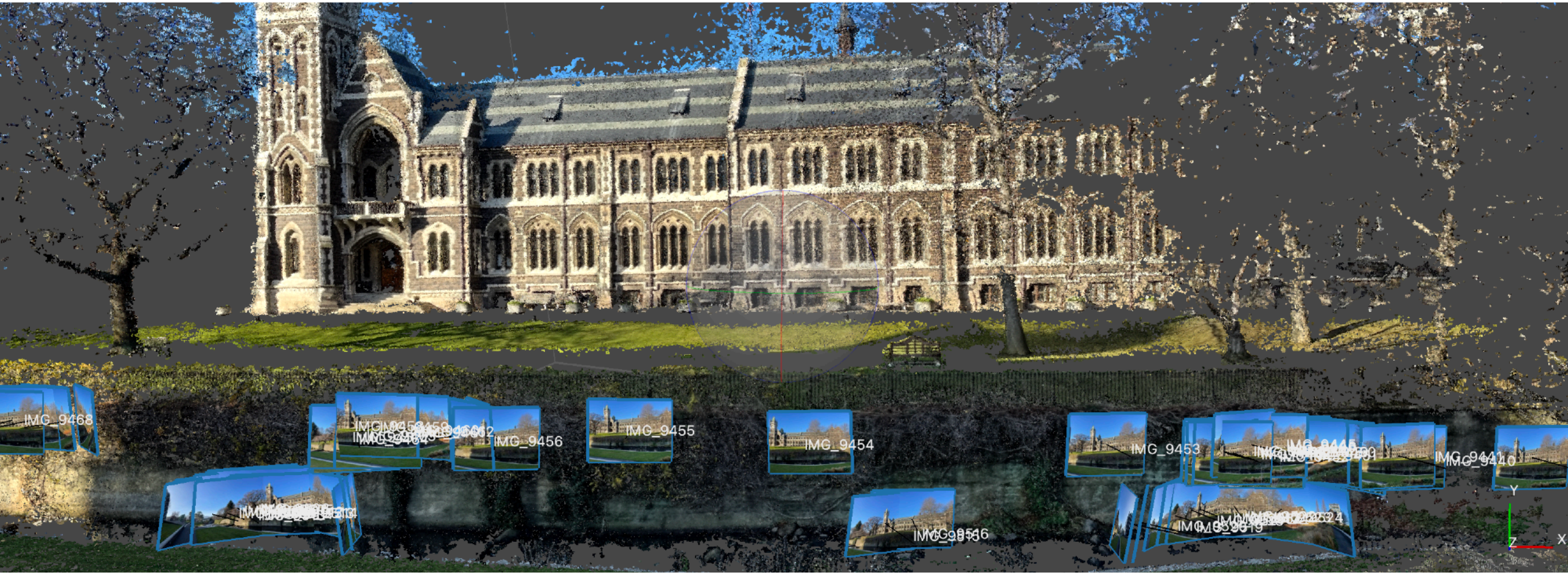
Visual Computing I:

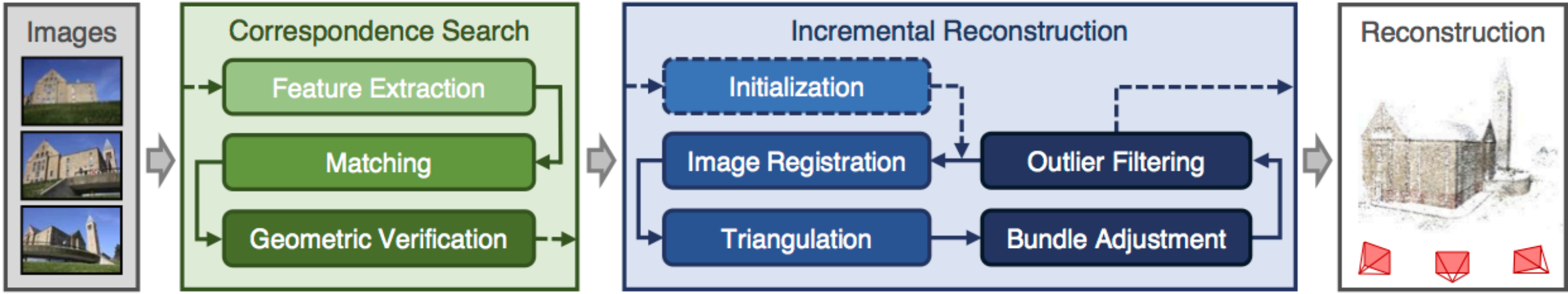
Interactive Computer Graphics and Vision



3D Reconstruction/Recap Lecture

Stefanie Zollmann and Tobias Langlotz

Colmap Workflow



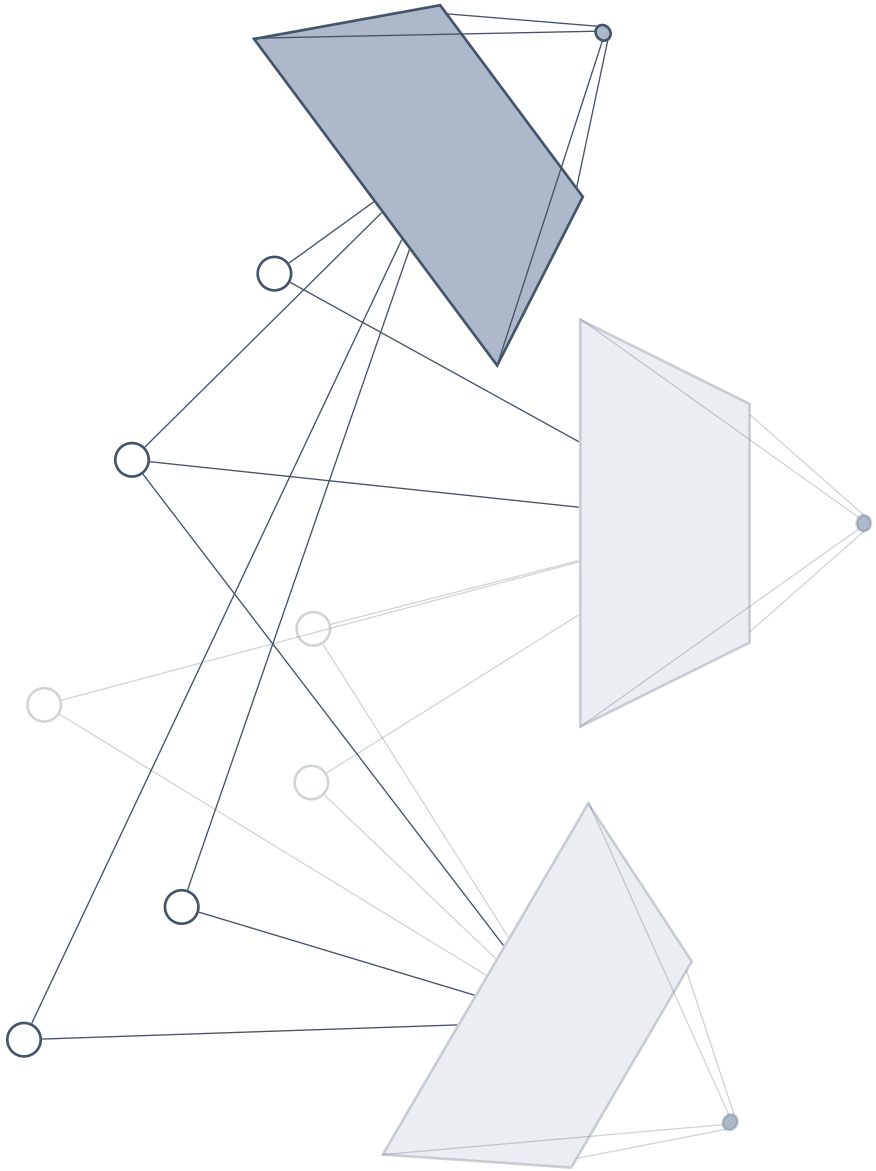
https://colmap.github.io/tutorial.html

Absolute Pose

Multi-View Stereo Pipeline

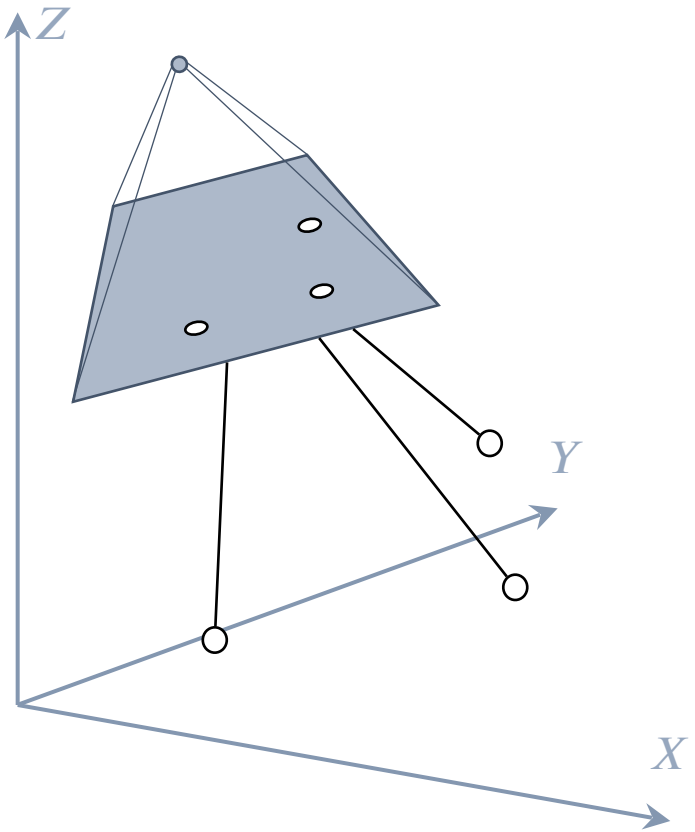
- Identify best matching pair
  - Two-view stereo for relative pose
  - Reconstruct 3D points
  - Minimise reprojection error

- While still images to add
  - Select next best view
  - *Determine absolute pose*
  - Reconstruct more 3D points
  - Minimise reprojection error



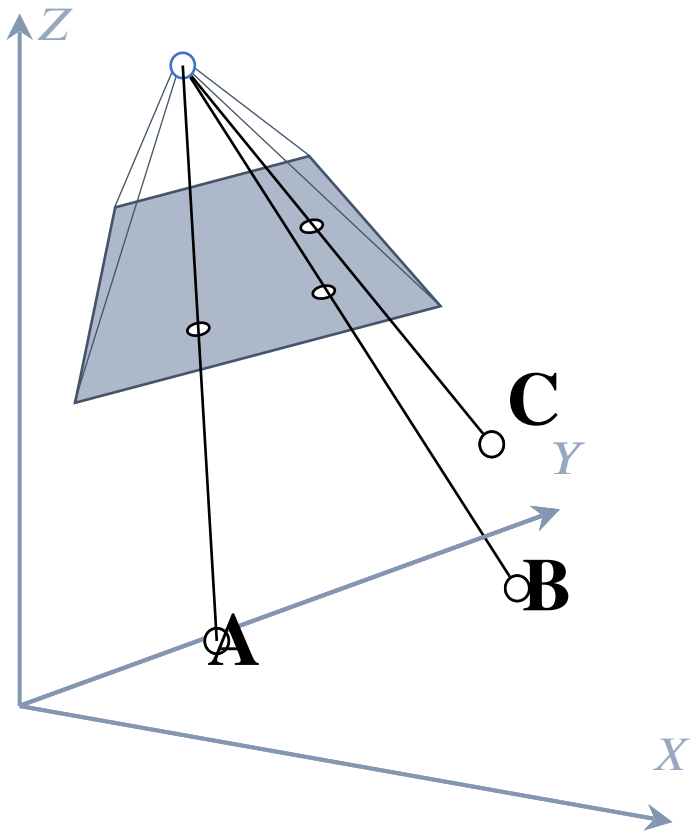
Absolute Pose Problem

- We have a new image taken with a calibrated camera
- We have already established a coordinate frame
- We have 3D points and their corresponding 2D points from the first image pair
- We found matches from new image to first image pair
- Need to find pose (rotation and translation) of the camera when the new image was taken



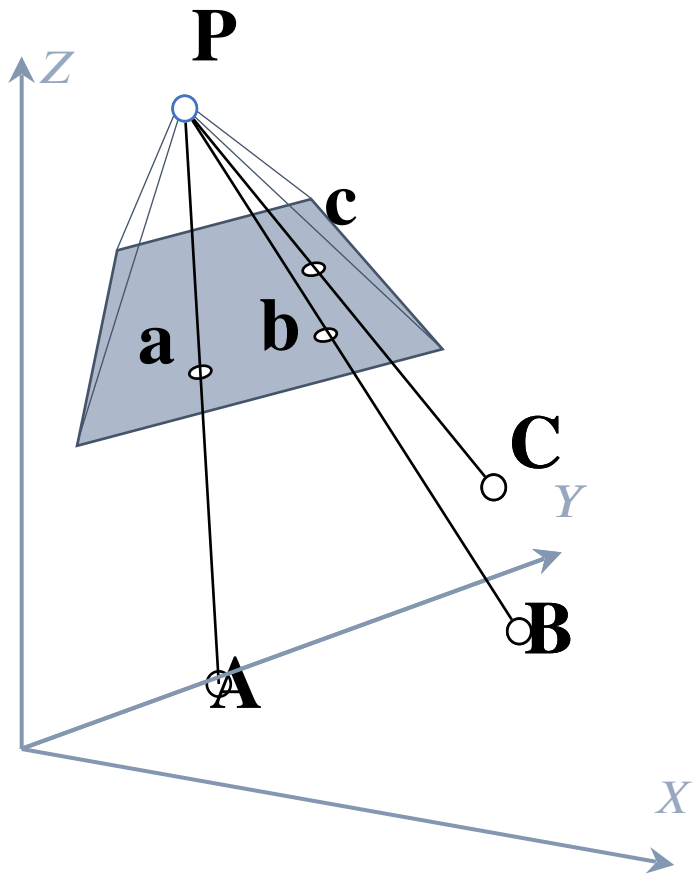
Perspective-n-Point Pose (PnP)

- Pose from 2D-3D matches
  - Have six unknowns ($\mathbf{R},\ \mathbf{t}$)
  - Each 2D-3D match $\mathbf{u}_i \leftrightarrow \mathbf{x}_i$ gives $k_i\mathbf{u}_i = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\mathbf{x}_i$
- How many matches?
  - Each point gives 3 equations
  - Adds 1 unknown (k_i)
  - $6 + n$ unknowns, $3n$ equations, $\mathbf{n} \geq 3$



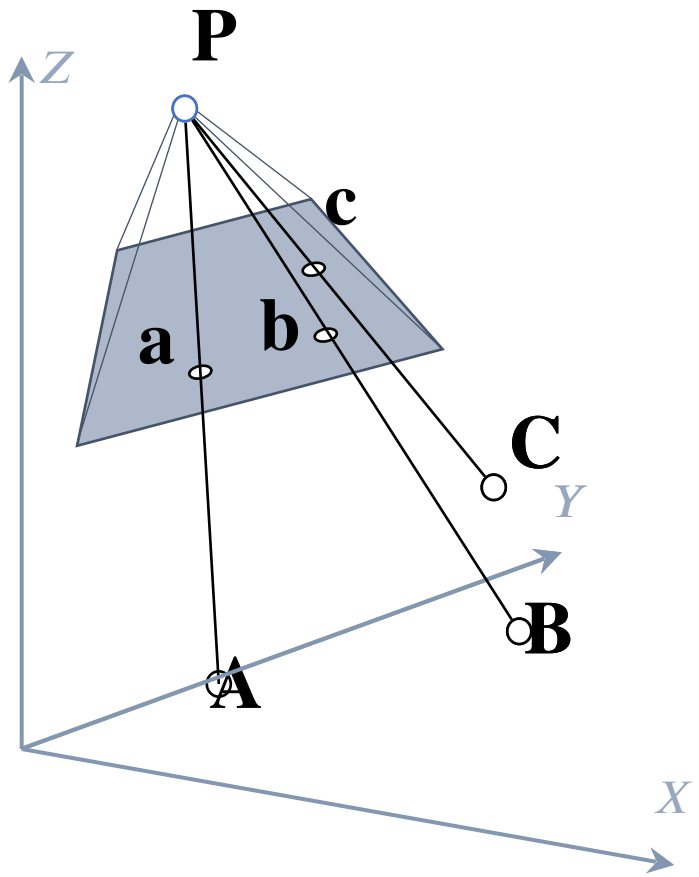
Perspective-3-Point Pose (P3P)

- Input:
  - The 3D coordinates of three known points in the world (e.g., landmarks)
  - The 2D coordinates of their projections in the camera image
  - The camera's intrinsic calibration (focal length, principal point, etc.)
- Output:
  - The rotation (camera orientation)
  - The translation (camera position)



Perspective-3-Point Pose (P3P)

- We have three 2D-3D matches
  - Call the 3D points $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$
  - The matching image points are $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$
  - The (unknown) camera location is $\mathbf{P}$
- Overview of solution
  - Use angles between $\vec{\mathbf{PA}}$, $\vec{\mathbf{PB}}$, and $\vec{\mathbf{PC}}$
  - Find the distances $\left|\vec{\mathbf{PA}}\right|$, $\left|\vec{\mathbf{PB}}\right|$, $\left|\vec{\mathbf{PC}}\right|$
  - This gives us equations for $\mathbf{P}$



Perspective-3-Point Pose (P3P)

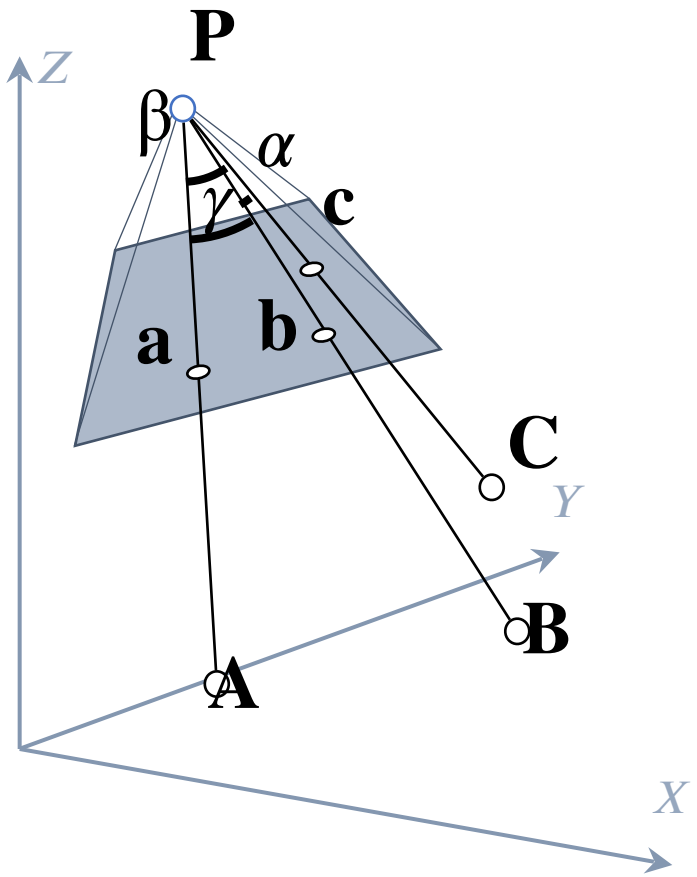
- Define the angles between rays

  $\alpha = \angle BPC \quad \beta = \angle APC \quad \gamma = \angle APB$

- Using dot products:

$$\cos\alpha = \frac{\mathbf{b} \cdot \mathbf{c}}{|\mathbf{b}|\,|\mathbf{c}|}$$

$$\cos\beta = \frac{\mathbf{a} \cdot \mathbf{c}}{|\mathbf{a}|\,|\mathbf{c}|} \qquad \cos\gamma = \frac{\mathbf{a} \cdot \boldsymbol{b}}{|\mathbf{a}|\,|\mathbf{b}|}$$



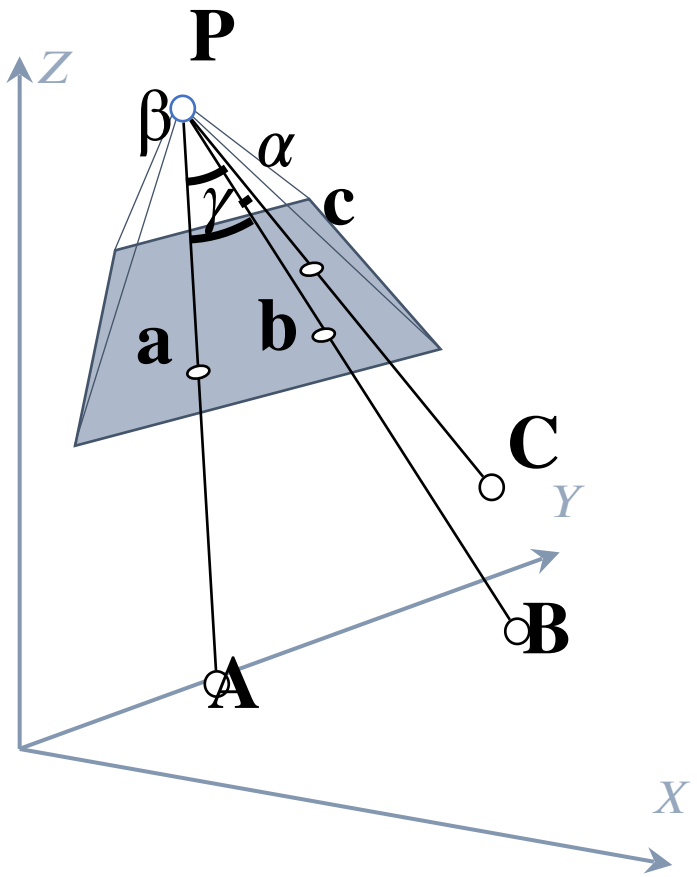
Perspective-3-Point Pose (P3P)

- The law of cosines tells us that

$$\left|\vec{AB}\right|^2 = \left|\vec{PA}\right|^2 + \left|\vec{PB}\right|^2 - 2\left|\vec{PA}\right|\left|\vec{PB}\right|\cos\gamma$$

$$\left|\vec{AC}\right|^2 = \left|\vec{PA}\right|^2 + \left|\vec{PC}\right|^2 - 2\left|\vec{PA}\right|\left|\vec{PC}\right|\cos\beta$$

$$\left|\vec{BC}\right|^2 = \left|\vec{PB}\right|^2 + \left|\vec{PC}\right|^2 - 2\left|\vec{PB}\right|\left|\vec{PC}\right|\cos\alpha$$

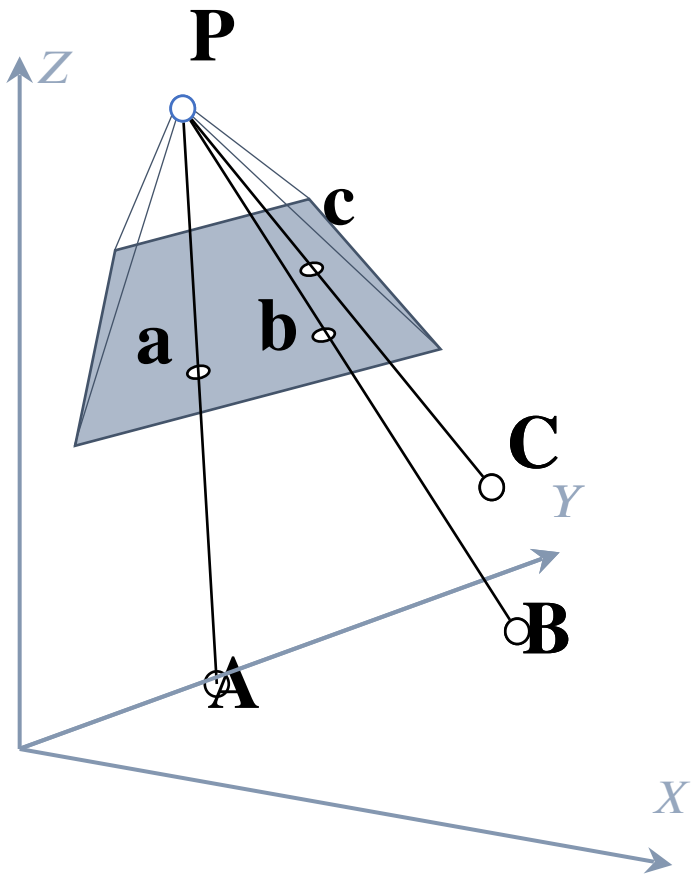


- Solve for

$$\left|\vec{PA}\right|, \left|\vec{PB}\right|, \text{ and } \left|\vec{PC}\right|$$

Perspective-3-Point Pose (P3P) - Summary

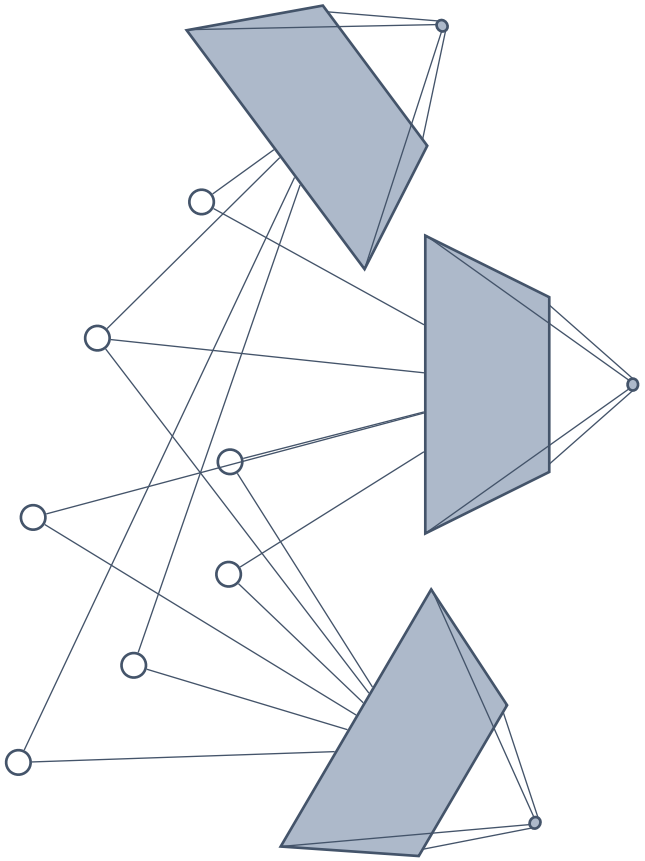
- Input: 3 known 3D world points and their corresponding 2D image points
- Form camera rays: Convert the 2D points into unit direction vectors in the camera frame
- Use the law of cosines: Relate angles between rays (from image) to distances between 3D points (from world) to solve for the distances from the camera to each point
- From these distances, compute the camera center (position) and the rotation that aligns the 3D world points with their camera rays



Bundle Adjustment

Multi-View Stereo Pipeline

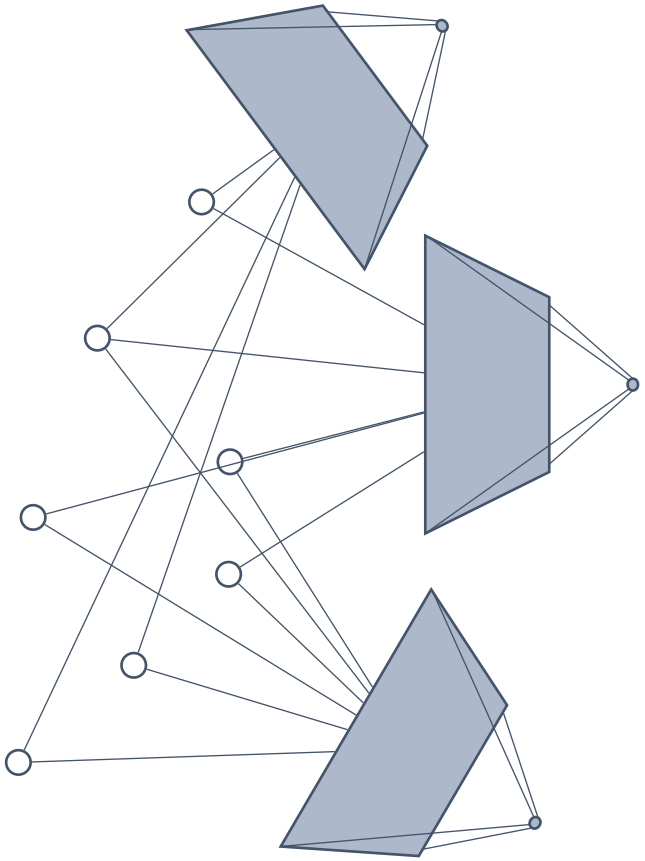
- Identify best matching pair
  - Two-view stereo for relative pose
  - Reconstruct 3D points
  - Minimise reprojection error
- While still images to add
  - Select next best view
  - Determine absolute pose
  - Reconstruct more 3D points
  - *Minimise reprojection error*



Bundle Adjustment

- Minimising reprojection error for multi-view stereo
- Jointly refines:
  - Camera parameters (poses + intrinsics)
  - 3D point positions (scene structure)
- Goal: minimize reprojection error across all images

- Geometry forms **bundles** of rays for each camera
- **Adjusting** these gives an optimal reconstruction solution
- Large sparse non-linear least squares
- Levenberg-Marquardt is commonly used for this

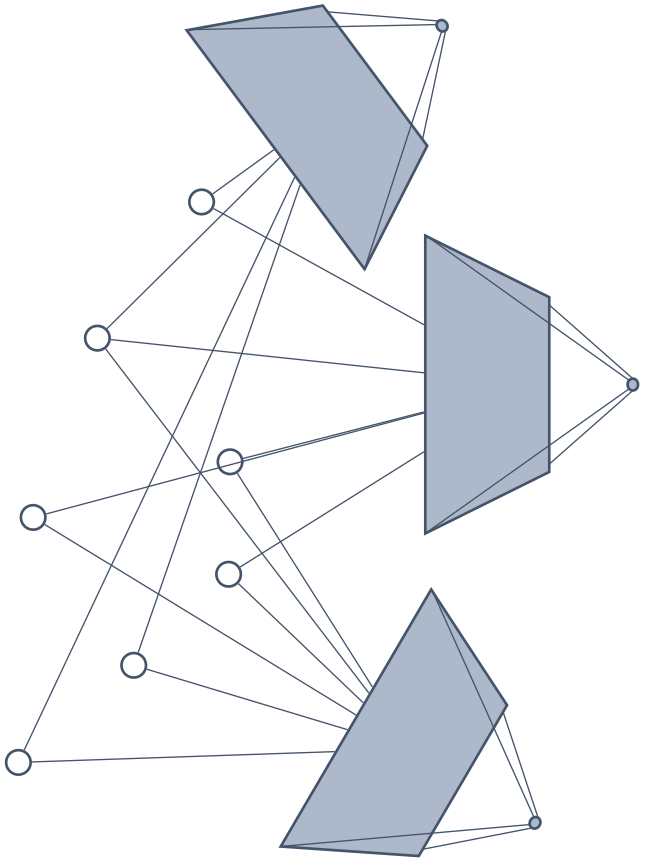


Reprojection Error

- If we have f features in n cameras, reprojection error is:

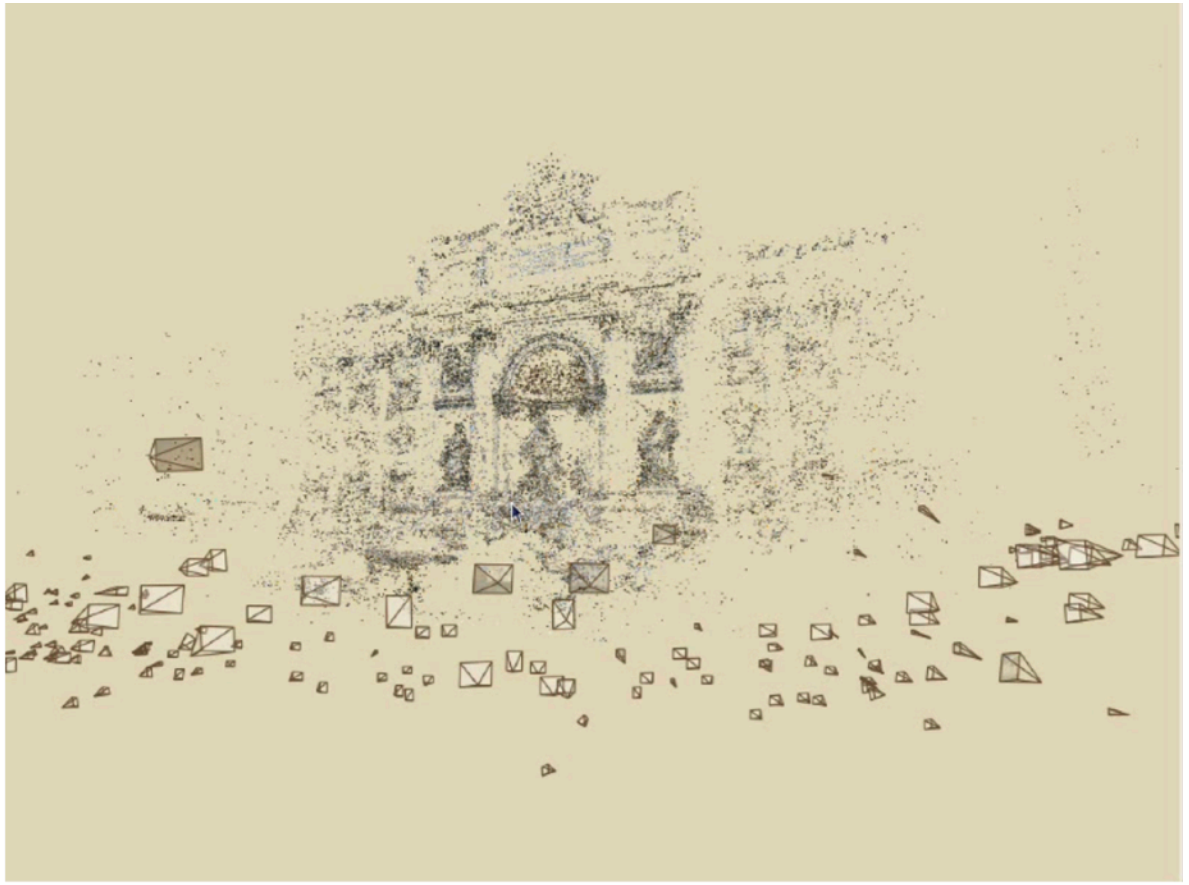
$$\sum_{i=1}^{n}\sum_{j=1}^{f}\left(\left(\mathbf{u}_{i,j}-\mathrm{K}_i\left[\begin{array}{cc}\mathrm{R_i} & \mathbf{t}_i\end{array}\right]\mathbf{x}_j\right)^2\right)$$

Measures Predictions



Bundle Adjustment

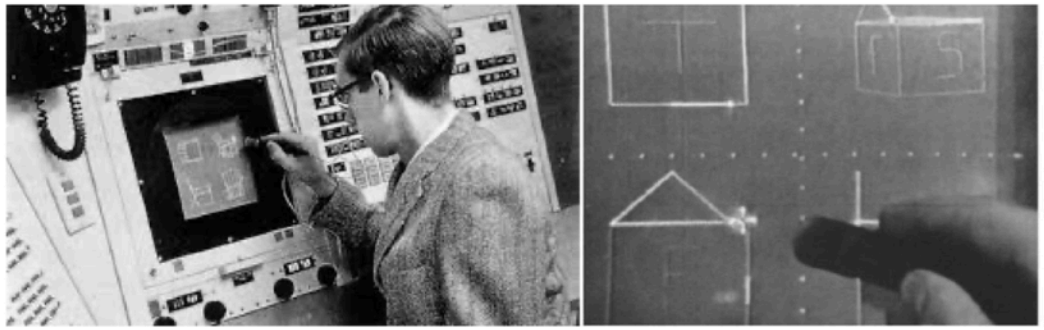
- Why is it Needed?
  - Individual pose or triangulation estimates are noisy
  - Errors accumulate across many views
- Bundle adjustment enforces global consistency



Revision

Introduction

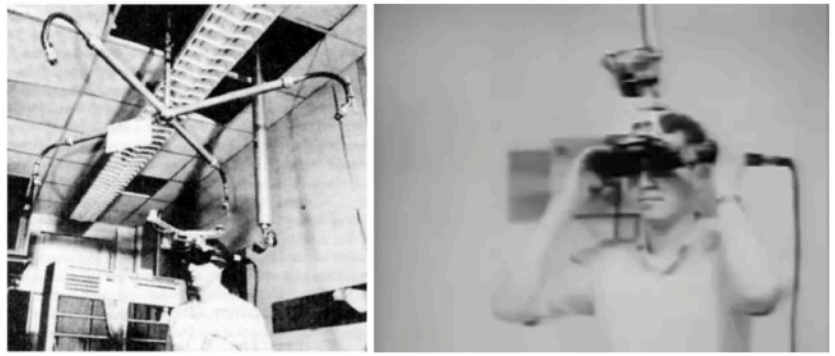
A Short History of Visual Computing



Ivan Sutherland, *Sketchpad, a Man–Machine Graphical Communication System*,1963

65

A Short History of Visual Computing



Ivan Sutherland, 1968 “A head-mounted three dimensional display”

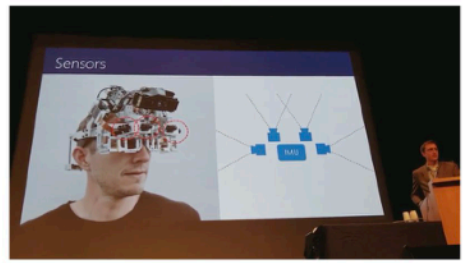
66

A Short History of Visual Computing

SIFT by David G. Lowe “Object Recognition from Local Scale-Invariant Features” (1999)

77

A Short History of Visual Computing



George Klein “Parallel tracking and mapping for small AR workspaces” (PTAM, 2007)

79

Digital Images

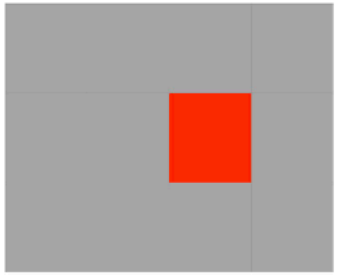
Digital Images and 2D Image Representation

- 2D array of pixels – a raster
- Pixels have a value, or list of values
  - value: usually brightness
  - values: brightness, colour, transparency

```
# Create a 2D array
image2D = create_array(height, width)

# Fill with example grayscale values
for y from 0 to height-1:
    for x from 0 to width-1:
        image2D[y][x] = 125

# Access pixel at (row=2, col=3)
image2D[1][2]= ?
```



16

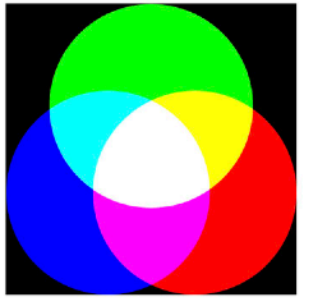
Representing Colour

- We perceive 3 colour dimensions
  - Short, medium, long cones
  - Intensity, Red-Green, Blue-Yellow
- Colour representations are usually 3 dimensional
  - Red-Green-Blue -> RGB
  - Cyan-Magenta-Yellow -> CMY (+K)
  - Hue-Saturation-Value ->HSV or HSL
  - And others (e.g. YUV, CIE)

34

RGB Model

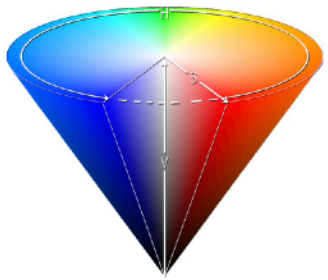
- Most common in computing
  - Most monitors and projectors
  - Additive colour model
  - Not perceptually uniform
- Record/transmit RGB channels
  - Aligns to the types of cone in the eye
  - Mixes three primary light colours
  - Can represent most colours (not all?)
  - Typically within the range [0,255] or [0,1]
    - E.g. Red as (255,0,0) or (1,0,0)
  - Can be abbreviated as hex values:
    - (183, 24, 92) -> #B7185C



38

HSV Colour Space

- How do we describe colour?
  - The sky is a bright blue
  - The trees are a dark green
  - That shirt is a bold colour
- HSV is closer to this:
  - Hue – what general colour is it
  - Saturation – how strong the colour is (distance from grey)
  - Value – how light or dark it is
  - Not perceptually uniform
- Also HSL, L = lightness/luminance
  - from dark to light



60

CMY Colour Space

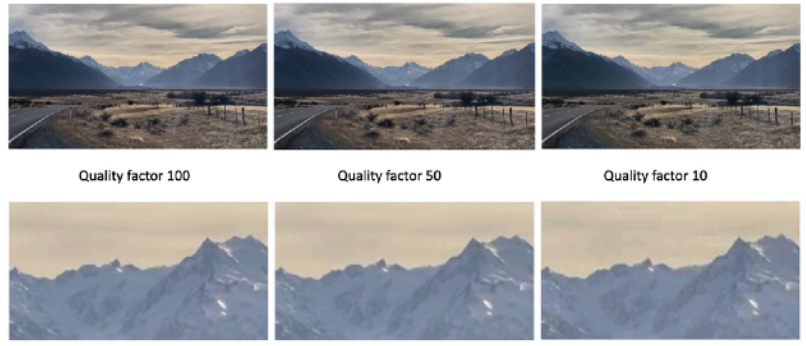
- Most common for printing
  - Subtractive Colour Model
  - Not perceptually uniform
- CMY is the inverse of RGB
  - Cyan Magenta Yellow (Black)
  - Cyan pigment absorbs red light
  - Magenta pigment absorbs green
  - Yellow pigment absorbs blue
- In theory C+M+Y = Black
  - In practice a 'true Black' ink is used
  - Black is represented by K (Key)


FINAL CMYK


DETAIL VIEW

43

JPEG Compression Artifacts

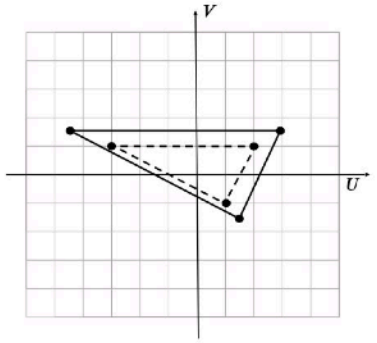


61

2D Transforms

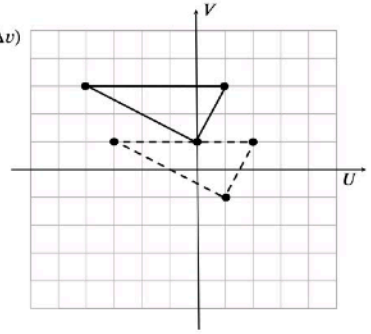
Transformations - Scaling

- Scaling points by factor, s
  $(u, v) \rightarrow (su,\ sv)$
- In vector terms
  $\begin{bmatrix} u' \\ v' \end{bmatrix} = s\begin{bmatrix} u \\ v \end{bmatrix}$
- Example:
  $\begin{bmatrix} u' \\ v' \end{bmatrix} = 1.5\begin{bmatrix} u \\ v \end{bmatrix}$



Transformations - Translation

- Shifting a point, $(u,\ v)$, by some offset, $(\Delta u,\ \Delta v)$
  $(u,\ v) \rightarrow (u + \Delta u,\ v + \Delta v)$
- In vector form:
  $\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix}$
- Example:
  $\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} -1 \\ 2 \end{bmatrix}$



Transformations - Rotation

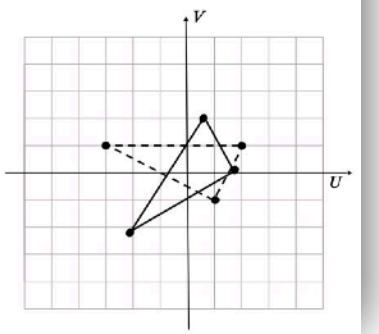
- Example rotation around 45 degrees

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$R\left(\frac{\pi}{4}\right) = \begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix} \quad = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{3}{\sqrt{2}} \end{bmatrix}$$

$$\approx \begin{bmatrix} 0.7071 \\ 2.1213 \end{bmatrix}$$



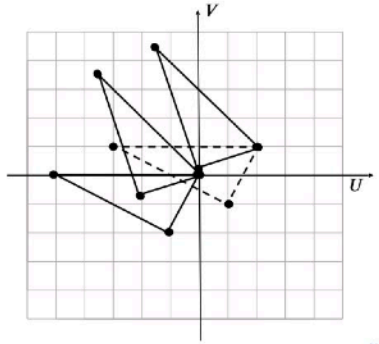
Transformations - Combining Transforms

- Rotate $-45°$ about $(2,1)$
  1. Shift by $(-2,\ -1)$
  2. Rotate by $-45°$
  3. Shift by $(2,\ 1)$

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = R_{-45°}\left(\begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix}\right)$$

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = R_{-45°}\left(\begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \end{bmatrix}\right) + \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$



Homogenous Coordinates

- The solution is counter-intuitive
- 2D points become sets of 3-vectors

$$\begin{bmatrix} u \\ v \end{bmatrix} \rightarrow k\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad \forall k \neq 0$$

- The vector $[a \quad b \quad c]^T$ corresponds to the point $\left(\frac{a}{c}, \frac{b}{c}\right)$
- These are homogeneous coordinates
- All (linear) transformations become 3×3 matrices

Projective Transforms

- To capture perspective, we generalize to Projective Transforms

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

- Affine = special case where: $h_{31} = 0, h_{32} = 0, h_{33} = 0$
- Properties:
  - Lines stay straight
  - Parallelism not preserved
  - Can map any quadrilateral to any other quadrilateral
  - Models perspective projection

$$H = \begin{bmatrix} 2 & 0 & 0 \\ 0.2 & 1 & 0 \\ 0.5 & 0.3 & 1 \end{bmatrix},$$

Quiz

Image Manipulation/Image Filters

Greyscale

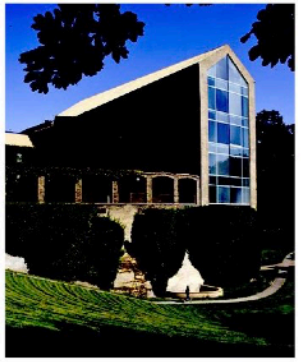
Original

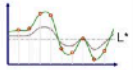
Greyscale
(“black&white” photo)

24

Contrast

- Contrast is the difference in luminance or colour that makes an object (or its representation in an image or display)
- Simpler: Contrast is the difference in luminance or colour for each pixel from the overall mean
  - Compute mean luminance L* over whole image
  - Scale deviation from L* for each pixel





15

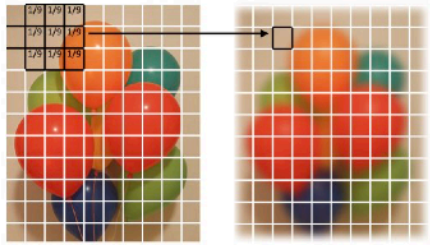
Quantization and Dithering

- Quantization:
  - Reduce intensity/colour resolution
    - Greyscale 8bit to 1bit
    - RGB 12 bit to 8 bit
    - RGB 8bit to 5/6 bits (RGB565)
  - Frame buffers have a limited number of bits per pixel
  - Physical devices have a limited dynamic range
  - Reduce image size

33

Linear Filters and Image Convolutions

- Image filtering with linear filters
  - Construct array **R** of same size as original image **F**
  - Fill each location in **R** with a weighted sum of the pixel values of corresponding location's neighbourhood in **F**
  - Use the same weights each time
  - Different sets of weights represent different filters
  - ***Convolution*** with a (filter) ***kernel***
- **Shift-invariant:** filtered result depends on image neighbourhood but not on position of neighbourhood
- **Linear:** filtering the sum of two images is equivalent as summing the two filtered images
- **Shift-Invariant Linear Systems (SILS)**



55

Image Features & Feature Detection

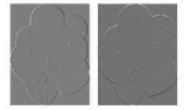
- Often required:
  - Features in an image that can be reliably detected in different images of the same scene
- A 'feature' or 'key point' is
  - A 2D location in an image
  - Accurately/precisely located
  - Can be found in different views
- Feature descriptors
  - Used to tell features apart
  - Ideally robust to changes in lighting, viewpoint, contrast, etc.
- Three common types of features
  - Edges – gradient-based
  - Corners – gradient-based
  - Blobs – bright/dark centre/surround

What are good image features?

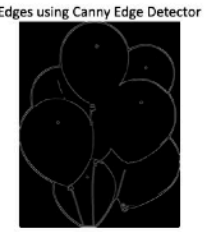
89

Edge Detection - Canny Edge Detector

- Canny Edge Detector
  - Step 1: Reduce noise (> Gaussian filter G_σ)
  - Step 2: Compute gradients in x and y direction (> Sobel operators H_u, H_v), gradient is a vector $\mathbf{g} = \begin{bmatrix} H_u & H_v \end{bmatrix}^T$
  - Step 3: compute gradient magnitudes (edge strength) and gradient directions (edge directions)
  - Step 4: select points with maximum magnitudes (above threshold T1)
  - Step 5: non-maximum suppression with hysteresis:
    - For all points with magnitudes above high-threshold T1
    - Trace along the edge (in edge direction) suppress all pixels that are not on traced line (pixel by pixel) and mark as visited
    - Stop tracing when point's magnitude is below low-threshold T2 or already visited



$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$



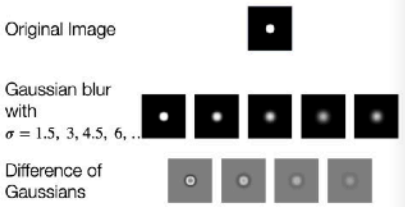
Magnitude

86

Features

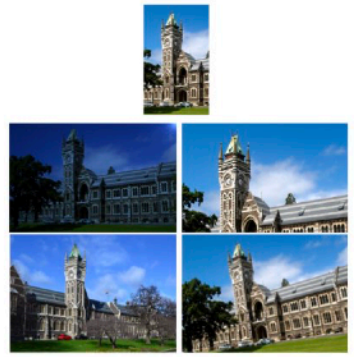
Blob Detection - Difference of Gaussians (DoG)

- 'Blobs' are alternative features
  - A blob is a dark region surrounded by a bright region, or vice-versa
  - Blobs have a scale as well as a location
- Difference of Gaussian filter
  - Two Gaussians of different width
  - Subtract wide from narrow
  - Bright blobs – high positive response
  - Dark blobs – high negative response



Scale Invariant Features Transform (SIFT)

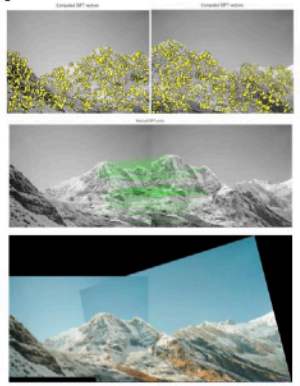
- Properties of SIFT
  - **Robustness:** Invariance to changes in illumination, scale, rotation, affine, perspective.
  - **Locality:** robustness to occlusion and clutter.
  - **Distinctiveness:** Easy to match to a large database of objects (e.g. different images).
  - **Quantity:** Many features (points) can be generated for even small objects.
  - **Efficiency:** Computationally "cheap", real-time performance.



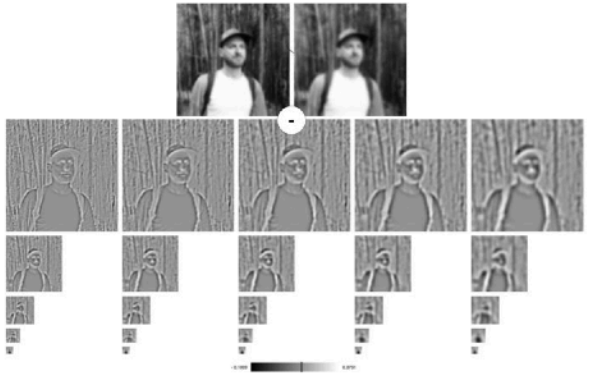
Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.

Scale Invariant Features Transform (SIFT)

- Common problems:
  - Panorama stitching
  - Image Search
  - Object recognition
  - Tracking (covered a different time)
- Scale Invariant Features Transform (SIFT):
  - Computer vision algorithm to
    - Detect,
    - Describe,
    - And match local features in images



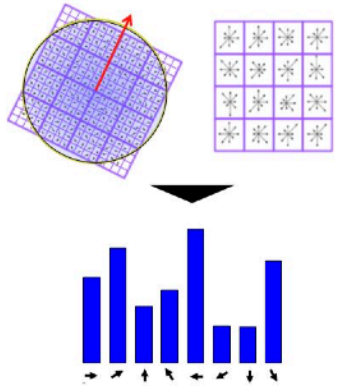
SIFT - Difference of Gaussians



Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

SIFT - Feature description from gradients

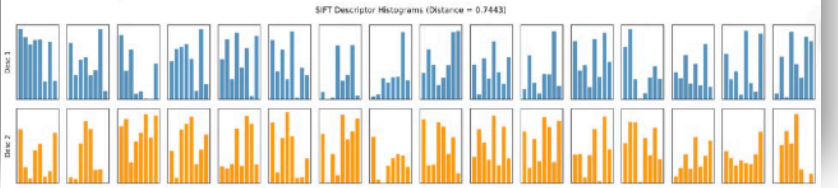
- Feature description (orientation and gradient)
  - Reorient image patch using main orientation
  - Create 16 gradient histograms (8 bins) weighted by magnitude and Gaussian window
  - Final keypoint Descriptor - 128 (4x4x8) element vector



SIFT - Matching

- SIFT descriptor 128 values
  - Usually these are bytes
- Computing the distance between features:

$$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots}$$

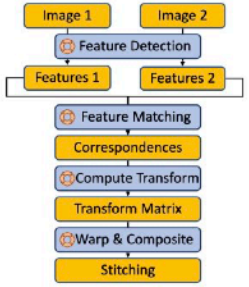


Images Stitching and Homographies

How to Create Panoramic Images?

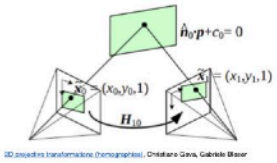
Image Stitching Process

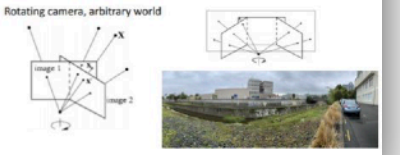
- Four main stages:
  1. Detect features in each image
  2. Match features between images
  3. Estimate a transform
  4. Warp one image to the other
- Stitching only works if:
  - The scene is (almost) planar, or
  - The camera only (mostly) rotates



Homographies in the Stitching Process

- A homography* is:
  - A linear map between two planes (or views of a plane)
  - A 3 x 3 matrix, up to a scale
- Two images of a scene are related by a homography
  - If the scene is planar, or
  - If the camera only rotates





* For our purposes – this is one case of a more general concept

Homography

$$\begin{bmatrix} u_i' \\ v_i' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

- The feature location in the second image
- Is equivalent to (equal up to a scale)
- Some homography (a 3 X 3matrix)
- Applied to...
- The feature location in the first image

RANSAC for Homographies

- Threshold for acceptance
  - Can often be interpreted in terms of the task
  - E.g. for how well does $\mathbf{H}$ align the corresponding points?
  - Set too low -> smaller consensus sets
  - Set too high -> outliers give less accurate models
- Number of trials
  - Resource limits
    - E.g. tracking in 25 fps video, do as many trials as you can in 0.04s
  - Can also be interpreted in a probabilistic manner
    - Replace number of trials with probability of success

Feathering (Alpha Blending)

- Weighted averaging across overlap
- Pros: simple, fast
- Cons: ghosting if misalignment or parallax

Quiz

3D Geometry

Translation and Scaling in 3D

- Translation by $(\Delta x,\ \Delta y,\ \Delta z)$
- Scaling by Scaling by a factor s

$$\mathbf{x}' = \mathbf{T}\mathbf{x} \qquad \mathbf{x}' = \mathbf{S}\mathbf{x}$$

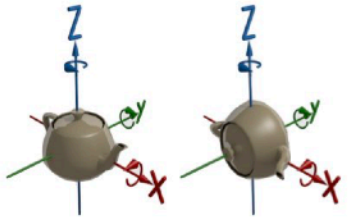
$$\mathrm{T} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathrm{S} = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

13

Rotation about the X Axis

- From Y to Z
  - X co-ordinate is unchanged

$$R_X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

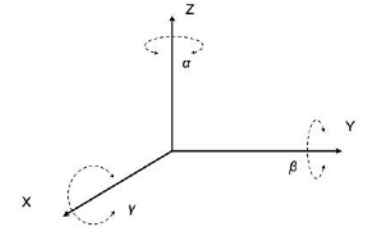


Rotation by 90° about the Z axis – from X to Y

18

Euler Angles

- Rotation defined by angles of rotation around the X-, Y-, and Z- axes
- Different conventions
  - Example Blender Rotation Mode:



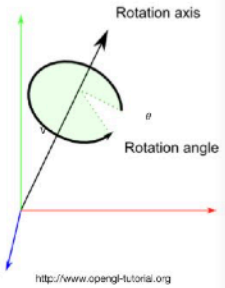
$$\mathbf{R} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

$$\mathbf{R} = \mathbf{R}_Z\mathbf{R}_Y\mathbf{R}_X$$

21

Quaternion

- Another way of presenting rotations (avoiding Gimbal Lock)
  - Can be expressed as a scalar and a 3- vector: (a, v)
  - A rotation about the unit vector v by an angle θ can be represented by the unit quaternion



angle in radians, unit-length axis

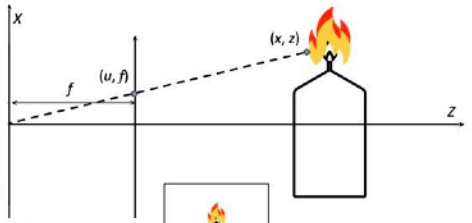
$$q(\theta, v) = \left(\cos\frac{\theta}{2}, v_x\sin\frac{\theta}{2}, v_y\sin\frac{\theta}{2}, v_z\sin\frac{\theta}{2}\right)$$

scalar = angle; vector = axis

30

The Pinhole Camera Model and Projective Geometry

- Removing the sign change
- We can put the image plane in front of the pinhole
  - Removes the sign change
  - Not practical for real cameras
  - The maths works out just fine



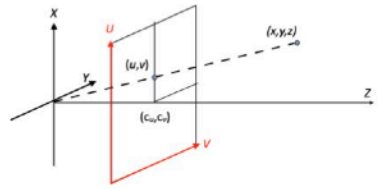
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

61

Camera Parameters - Intrinsics and Extrinsics

- Often break this down into

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \equiv \underbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Extrinsic}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



- Most simple case: $\mathbf{u} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]\mathbf{x}$
- $\mathbf{K}$: camera calibration or **Intrinsic parameters**
- $[\mathbf{I} \mid \mathbf{0}]$: camera pose or **Extrinsic parameters**

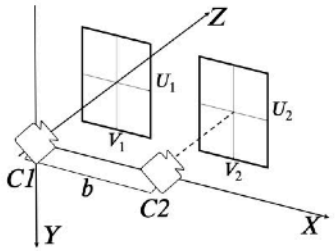
68

Stereo Vision/ Epipolar Geometry

A Simple Stereo System

- Assume a set of 2 pinhole cameras
  $\mathbf{u} = \mathrm{K}[\mathrm{R} \quad \mathbf{t}]\mathbf{x}$
  - Simplify for now: $\mathrm{K} = \mathrm{I}$
- Camera 1 (C1): $\mathrm{R} = \mathrm{I}, \mathbf{t} = \mathbf{0}$
  - No rotation, looks along Z
- Camera 2 (C2): shifted along X

$$\mathrm{R} = \mathrm{I}, \; t = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}$$



33

Project a Point into the Cameras

- Camera 1:

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x \\ y \\ z \end{bmatrix} \equiv \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$

- Camera 2:

$$\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 0 & 0 & -b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x - b \\ y \\ z \end{bmatrix} \equiv \begin{bmatrix} (x-b)/z \\ y/z \\ 1 \end{bmatrix}$$

35

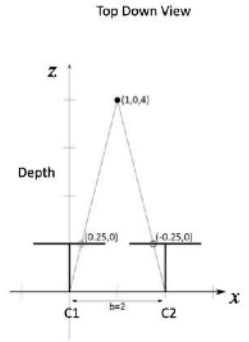
Depth and Disparity

- Difference between the views is the disparity
  - In this case it is in u:

$$u_1 - u_2 = \frac{x}{z} - \frac{x-b}{z} = \frac{b}{z}$$

  - The v values are the same
- If we know b, can find z:

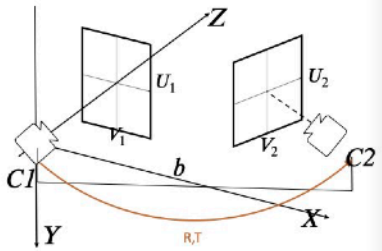
$$z = \frac{b}{u_1 - u_2}$$



36

More General Stereo

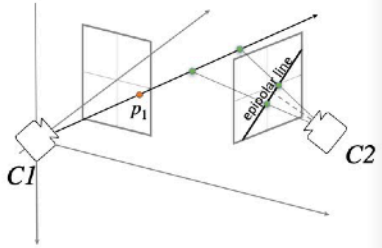
- In the more general case:
  - The cameras have different calibration parameters
  - There is a rotation between the cameras
  - There is translation along all three axes
- We can still align the XYZ coordinate system with camera C1



41

Epipolar Geometry

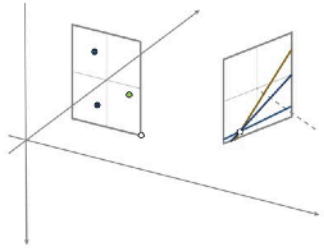
- Consider a point p_1 in the first camera
  - This gives a 3D line, along which the point must lie
  - The 3D line projects to a 2D line in the second image – this is called an **epipolar line**



44

Epipoles and Epipolar line

- Each point in one image makes an epipolar line
  - These all meet at the epipole in the other image
- For simple (parallel) case
  - Line through camera centres is the X axis
  - Epipoles are at infinity
  - Epipolar lines are the rows

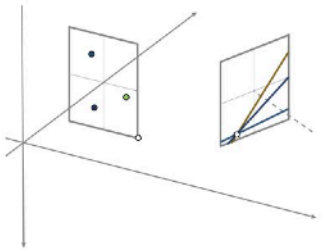


49

Stereo Vision: Fundamental Matrix

The Fundamental Matrix

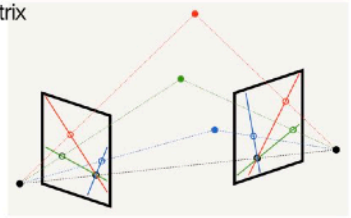
- In general: a point, $\mathbf{p}$, lies on a line, $\boldsymbol{l}$, if $p^T l = 0$
- In our case: $p_2^T l_2 = 0$
- If we now use our epipolar constraint:
  - $l_2 = \mathrm{F} p_1$
- Then $\mathbf{p}_2^\top \mathbf{F} \mathbf{p}_1 = 0$



15

The Essential Matrix

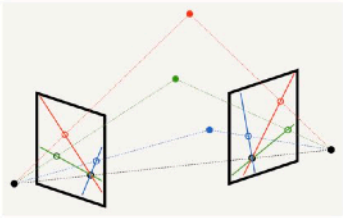
- For calibrated cameras
  - We can factor out K_1 and K_2 in $\mathrm{F} = \mathrm{K}_2^{-T}[\mathbf{t}]_\times \mathrm{R}\mathrm{K}_1^{-1}$
    - This gives us the essential matrix $\mathrm{E} = \mathrm{K}_2^T \mathrm{F} \mathrm{K}_1 = [\mathbf{t}]_\times \mathrm{R}$
  - E has only five DoF:
    - 3 for 3D rotation
    - 3 for 3D translation
    - Less 1 for unknown scale



18

The 8-Point Algorithm

- Estimate F from matches
  - F is a 3×3 matrix – 9 values
  - Defined up to a scale – 8 DoF
  - Need 8 constraints
- F turns out to be rank 2
  - Only 7 degrees of freedom
  - There is a 7-point algorithm



23

The Homogeneous Linear System

$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ u_3u_3' & v_3u_3' & u_3' & u_3v_3' & v_3v_3' & v_3' & u_3 & v_3 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

A (Known) $\quad f$ (Unknown)

$$Af = 0$$

26

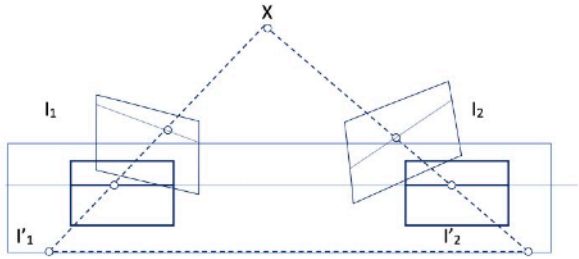
The 7-Point Algorithm

- Exploiting the geometric constraint that the determinant of F must be zero
- The "Two-Solution" Problem: With only 7 points, the system is underdetermined, resulting in two initial 'basis' solutions, let's call them F_1 and F_2
- Finding the Right Mix: The true solution F is a specific blend of these two basis solutions: $F = \alpha F_1 + (1 - \alpha) F_2$
- The Cubic Equation: By enforcing the constraint that det(F)=0, we get a cubic polynomial for the mixing value α

- **Pro**: Requires fewer points
- **Con**: More complex due to solving a cubic polynomial; can have multiple solutions; less robust to noise than the 8-point algorithm with normalisation and RANSAC

30

Stereo Rectification

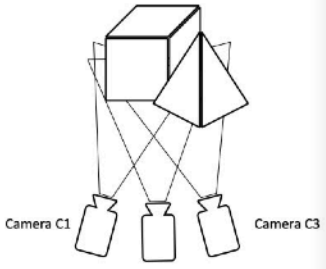


39

Depth Sensing

Passive Range Scanning

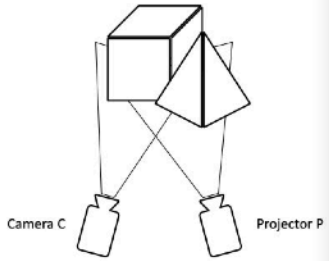
- Two or more cameras
  - Stereo camera
  - Two cameras
  - One camera takes several photos from different perspectives
  - Multiple cameras
- Which one is the easiest and why?
- What is the main challenge?



15

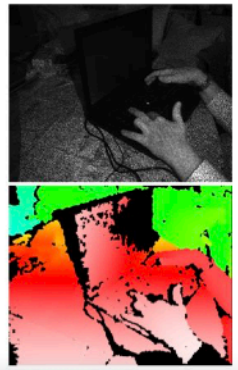
Structured Light Projection

- Many structured light techniques exist that follow the two objectives:
  - Determine projector-camera correspondences with as few images as possible
  - Be robust against surface modulation and noise
- Codes include:
  - Binary codes, such as the Gray code (most common): <100 images, quite robust
  - Intensity codes that apply cosine patterns and phase shifts: 6 images, but are not robust against strong absorptions
  - Color coding: <3 images, but very fragile (does not work on coloured surfaces)



25

Structured Light Projection



MS Kinect 1 (PrimeSense sensor, not MS Kinect 2 uses ToF discussed later)

42

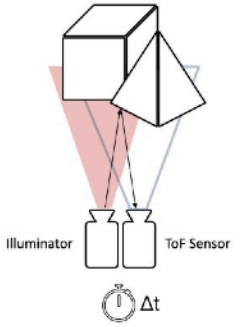
Structured Light Projection

Raskat et al. "Shaderlamps"

33

Optical Time-of-Flight (ToF)

- General idea:
  - Compute depth by round-trip estimation of a light wave emitted and its reflection back to the sensor
  - $D = C \cdot \Delta t/2$
  - $\Delta t = 2D/C$
  - D=1m $\Delta t = 6.6*10^{-09}$ s
  - D=0.01m $\Delta t = 6.6*10^{-11}$ s
  - D=0.001m $\Delta t = 6.6*10^{-12}$ s (pico seconds)
  - 3.3 picoseconds for light to travel 1 millimeter



55

Depth from Mono / Depth Estimation using ML

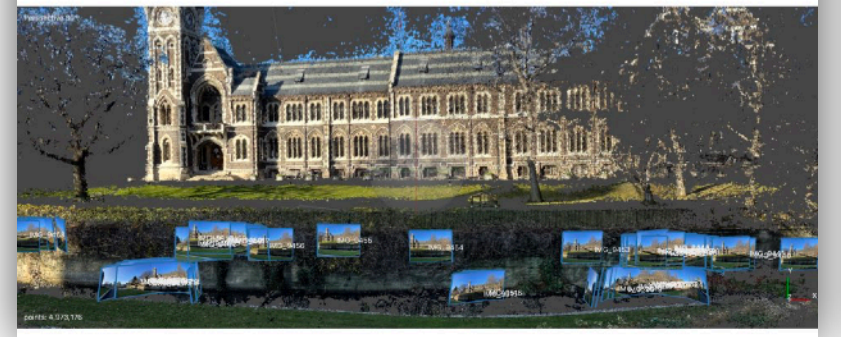
- Convolutional neural networks (CNNs) have been applied to monocular depth estimation
- Common architectures include U-Net, ResNet, and DenseNet
- During training:
  - Network is given pairs of images and their corresponding ground-truth depth maps
  - Minimize the difference between predicted depth and ground truth depth

https://github.com/yuhsuanyeh/BiFuse

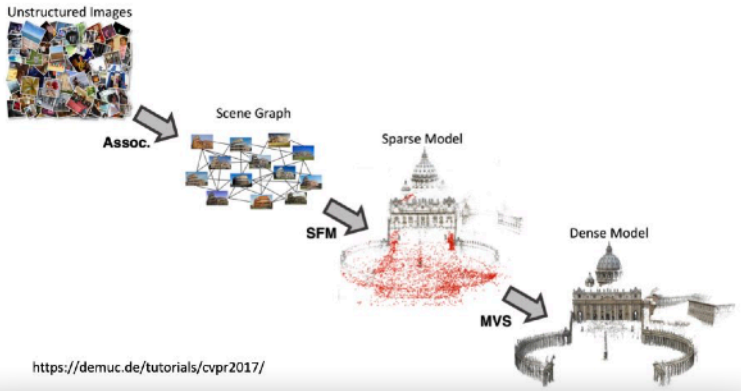
71

Multiview Geometry

Goal

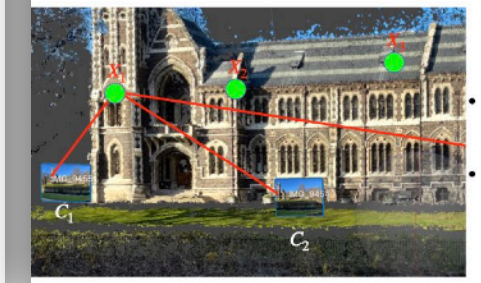


Workflow



https://demuc.de/tutorials/cvpr2017/

Structure-from-Motion

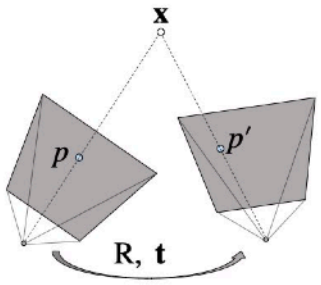


- Estimation of 3D points X_i and camera poses C_j
- Based on motion (camera views from different viewpoints)

43

Decomposing the Essential Matrix

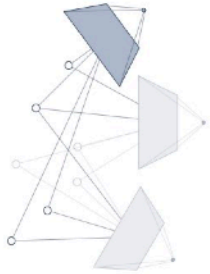
- To recover R and t
  - Compute SVD of E
  - SVD: gives us three matrices U S and V transpose: $\mathbf{E} = \mathbf{USV}^T$
  - From this decomposition, we get:
    - Two possible rotations (R)
    - Two possible translations (t)
  - That means there are four candidate solutions in total.
  - To find the correct one: check which solution places the 3D points in front of both cameras



48

Multi-View Stereo Pipeline

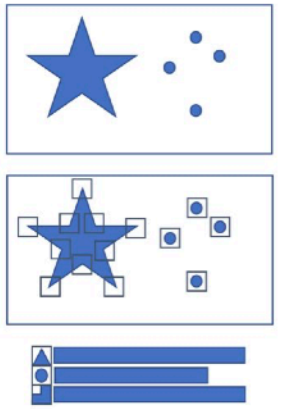
- Identify best matching pair
  - Two-view stereo for relative pose
  - Reconstruct 3D points
  - Minimise reprojection error
- While still images to add
  - Select next best view
  - Determine absolute pose
  - Reconstruct more 3D points
  - Minimise reprojection error



63

Bags of Words (BoW)

- Need equivalent of words
- Cluster features – can pre-train
  - Choose some number, k, of 'words'
  - Make k clusters – e.g. k-means
  - Cluster centres become words
- Bag of words for images
  - Detect and describe feature
  - Count features closest to each word



69

Quiz

The end!