Lecture 8: Dense Construction

→ Why do we need Dense Reconstruction?
- → Accurate 3D models → cultural heritage
- → Reconstruction of houses, buildings, famous sites.

→ Stereo Reconstruction →
- → Input → a left and right pair of images
- → Output → a disparity or depth map.

$$\text{disparity} \propto \frac{1}{\text{depth}}$$

★ In stereo vision, the two images, captured by the two cameras separated by a distance, can be used to get the 3d location (x, y, z) of the points of images in real world i.e. the depth.

disparity → is the difference in image location of the same 3D point when projected under perspective to two different cameras.

★ Any point in the scene that is visible in both cameras will be projected to a pair of image points in the two images, called a 'conjugate pair'. The displacement b/w the positions of the two points is called the "disparity".

depth → (the actual location of 3d point) can be calculated by using the disparity of the corresponding point

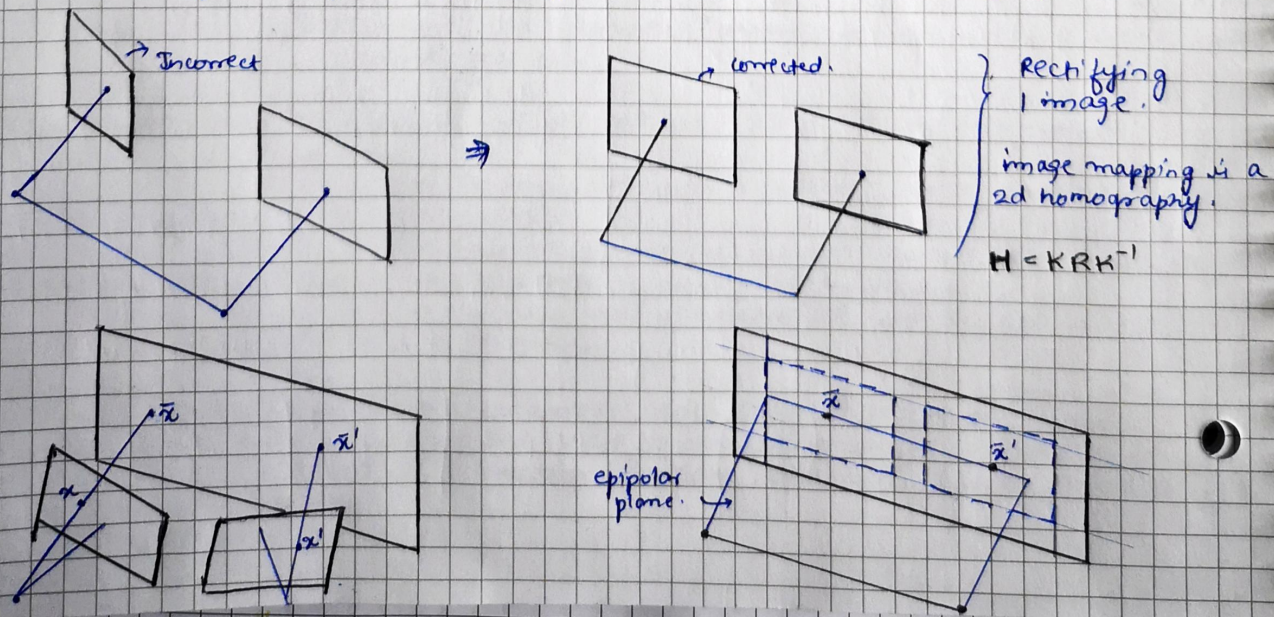
$$\text{depth} = \frac{\text{baseline} * \text{focal length}}{\text{disparity}}$$

baseline → distance b/w 2 cameras

→ How can we recover the depth?
we can recover the depth by (matching + triangulation)

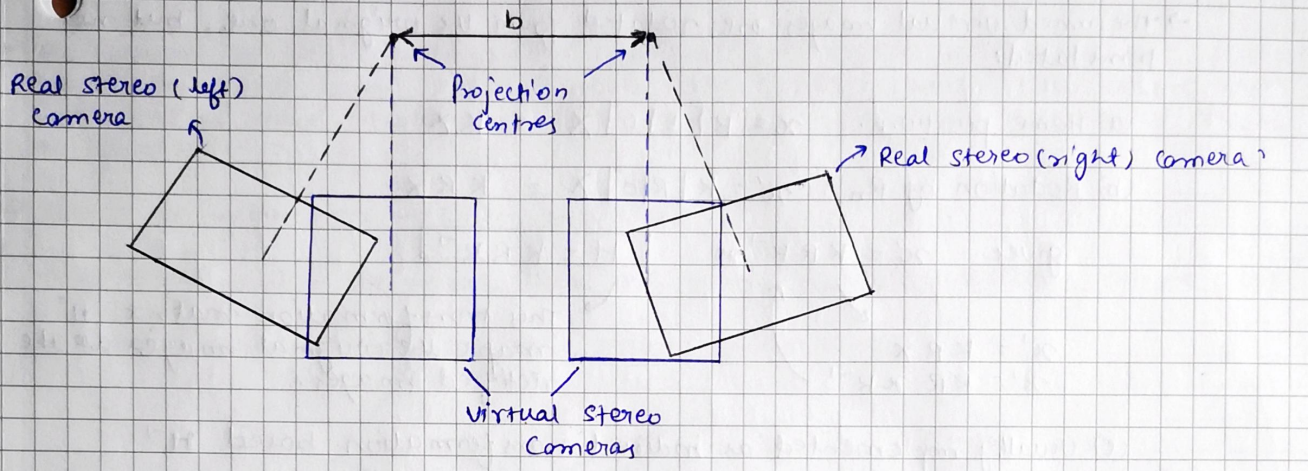
→ stereo reconstruction → Image Rectification.

→ for stereo cameras, we assume that the cameras are parallel.
When we make a stereo camera, there may be many physical inaccuracies during manufacturing leading to the camera not being parallel.



→ How do we do this?

① we have two cameras which are not perfectly aligned. we have to rotate the cameras in such a way that they are aligned with the baseline. and also to the y-axis.



② Define rotations of the two virtual views that the x-axes of the camera coordinate system are parallel to the transition of the ~~right~~ original camera positions.

→ These rotations are distributed equally to the right and the left camera to reduce the resulting distortions.

→ Additionally, the camera matrix representing the intrinsic ~~camera~~ parameters should be equal for both the virtual views.

③ Assuming calibrated stereo cameras

④ the intrinsic and extrinsic parameters are given.

⑤ Determination of Image Transformation

① The new camera matrix K_n should be equal for both virtual views which can be computed by calculating the mean of the old views.

$$K_n = \frac{K_0^{left} + K_0^{right}}{2}$$

② New rotation matrix R_n which represents the axes (x, y, z) of virtual camera system wrt world coordinates system.

→ Virtual x-axes is set parallel to the image baseline → r_1

→ Virtual y-axes is set orthogonal to the virtual x-axis and
- → orthogonal to old Z-axis of left camera. or
- → orthogonal to the mean of old Z-axes of two cameras → r_2

→ Virtual Z-axes orthogonal to the virtual x-axes (baseline) and virtual y-axis → r_3

translation of world camera on left.

$$r_1 = \frac{t^{left} - t^{right}}{\| t^{left} - t^{right} \|}$$

$$r_2 = 0.5 (r_{3ol} + r_{3or}) \times r_1$$

$$r_3 = r_1 \times r_2$$

$$R_n = \begin{bmatrix} r_1^T \\ r_2^T \\ r_3^T \end{bmatrix}$$

→ Computation of Homography → finding the mapping of old image to the new image.

① Link transformations to Homography "H"

→ The aimed virtual images are rotated from the original one, but not translated:

(a) Home position: $x = K[I|0]X = KX$

(b) Rotation by R_n: $x' = K[R|0]X = KRX$

gives $x' = KRK^{-1}x$ $\quad \boxed{H = KRK^{-1}}$

This transformation matrix "H" maps the original images to the rectified images.

$x' = KRX$

$x' = KR\,xK^{-1}$

(c) Usually implemented as indirect transformation based H^{-1}

(d) Right image is calculated in a similar way

$\boxed{H^{-1}x' = x}$ pixels in the new image projected back into the original image.

We don't go from old image to new image, but rather from new image (virtual) to the old image (real)
This is done to take care of interpolation errors.

→ Direct and indirect Geometric Transformation

Direct Transformation	Indirect Transformation
① For each pixel of the input image, the corresponding coordinate of the output image are computed.	① The inverse transformation is used to calculate (for each pixel) of the output image), the corresponding (non-integer) position of input image
② These coordinates will usually have non-integer values (decimal values). Also, some pixels might have multiple or non-assignments.	② The grey values of the surrounding pixels are then used for simple interpolation (bi-linear interpolation) → Any other interpolation method is also possible.
③ This requires additional computational effort during assignments of grey values.	
④ Not recommended.	

→ Step 2 → Triangulation with Parallel Cameras

→ Triangulation reduces to a simple geometric problem

$$K = K' = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R = I \qquad t = \begin{pmatrix} t_x \\ 0 \\ 0 \end{pmatrix}$$

→ Then $y = y'$ and depth z can be computed from disparity:

$$z = \frac{f\,t_x}{d} \qquad d(\text{disparity}) = x' - x$$

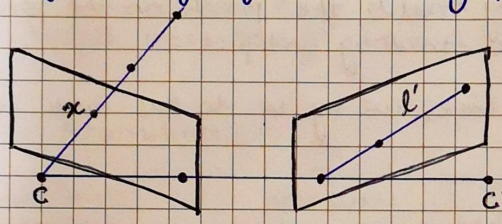
→ refer to Lecture 7 page 3

→ As opposed to the matching for structure for motion where we use feature based mapping with (Harris / lines, edges / SIFT keypoints), here we are more interested in dense reconstruction therefore, we try to match as many pixels as possible.

⇒ Stereo Reconstruction → Dense matching.

→ for every point in the image, there are many possible matches on the other image

→ there is high-ambiguity since many points look similar.



$$l' = Fx$$

$$F = K'^{-T}[t]_x R K^{-1}$$

$$\boxed{x'^T F x = 0}$$

→ we can use the camera geometry i.e. finding the correspondences becomes a 1D search problem. But this solution also yields multiple solutions for a point.

→ Therefore we use the pixel-neighbourhood information – correlation based approaches (also called window-based approaches)

→ To do this window based matching, we have 3 approaches:

(a) Normalised cross-correlation (NCC) → Calculating the correlation of 2 signals and also normalizing it.

$$\frac{\sum_{i,j \in W} I_1(i,j) \cdot I_2(x+i, y+j)}{\sqrt{\sum_{(i,j) \in W} I_1(i,j)^2 \cdot \sum_{(i,j) \in W} I_2(x+i, y+j)^2}}$$

(b) Sum of squared differences (SSD)

$$\sum_{(i,j) \in W} (I_1(I,J) - I_2(x+i, y+j))^2$$

W → window.

(c) Sum of absolute differences (SAD)

$$\sum_{(i,j) \in W} |I_1(I,J) - I_2(x+i, y+j)|$$

(d) Census transform.

→ Based on comparison of radiance values within the window

→ The effect of outliers is low.

* Now we have the steps for matching.

① search along the epipolar line for the best match.

② Best match : (a) maximum NCC
(b) minimum SSD or SAD.

→ Enhanced matching

→ Estimating the disparity solely based on matching leads to errors.

→ there are 2 approaches of optimizing correspondences jointly.
→ Single scan-line at a time with dynamic programming techniques
→ full 2D grid with graph cuts.

→ Scanline - stereo → coherently match the pixels on the entire scan-line
* Scanline stereo generates streaking artifact.

→ Instead of doing this line by line, we try to do this with 2D.

→ Energy minimization →

(a) In order to find more reliable correspondences, an energy function can be minimized.

(b) It can be adapted to the situation and becomes arbitrarily complicated.

$$\min \; \varepsilon_{data} + \lambda \varepsilon_{reg}$$

ε_{data} → data term

ε_{reg} → regularization term (smoothness using interpolation)

(c) usually consists of a data term (consistency of matching information)
→ NCC, SSD, SAD

(d) A Regularization term (smoothness (total variation regularization))

(e) Also different patch sizes can be included.

(f) can be minimized using graph cuts

→ Dense matching → challenges

Scene elements do not always look the same in the images.

(A) Camera-related problems.

(a) Image Noise (b) Lens distortion (c) Color/chromatic abberation

(B) Viewpoint related problems

(a) Perspective distortions (b) Occlusions (c) Specular reflection

(C) Scene-related problems

(a) Illumination ~~problems~~ changes (b) moving objects

(d) matching ambiguity

(a) low texture regions (b) Repititive texture patterns.

→ Assumptions in Dense matching.

① Appearance → The projections of a scene (3D) patch should have similar appearances in all images.

② Uniqueness → A point in an image will only have one match in another image

③ ordering → The order of appearance is not altered by the camera displacement.

④ Smoothness → Depth varies smoothly (objects have smooth surfaces)

→ choice of camera setup during Dense matching.

Short baseline	wide baseline.
① matching is relatively easy	① matching is hard
② Higher depth uncertainty	② lower depth uncertainty. ✓

⇒ Multi view Reconstruction.

① more constraints
② Less occlusions
③ more robust
④ Appearance constraint can be relaxed

However.

① more complex
② strong perspective distortions
③ Propagation of calibration uncertainties

⇓

Dense matching along many images is still hard.

→ Triangulation in multiple images.

→ Assuming we had found the correct matches along "n" images, we triangulate them exactly as before by stacking the equations for all camera views. i.e.

$$\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} X = 0$$

→ multi view Reconstruction - Algorithms.

* we have many different algorithms but all of them have same assumptions

① Lambertian surfaces (no specular reflections)
② static scene
③ camera parameters are available (projection matrices are known)
④ Underlying photometric consistency. (NCC, etc)

→ Scene Representation → Voxel Grid. (binary representation)

→ Divides the scene into cubes (voxels)
→ Need to sample the 3D space
→ Voxels are marked as occupied / empty according to a photometric consistency measure (NCC, SSD, etc)
→ Accuracy depends on size of voxels.
→ No assumptions on the scene required.
→ memory allocation issues.

(b) Depth map.

→ usually 1 depth map per view (by matching and triangulating features in small sets of images)
→ NO sampling of 3D space
→ simple
→ Suitable for multi-core
→ Depth maps have to be merged.

(c) Point cloud.

→ local planar approximation of the surface at each point
→ Usually a normal vector is associated to each point.
→ Sparse to dense reconstruction.
→ Suitable for multi-core
→ may be noisy

(d) mesh.

→ models the surface as a connected set of planar facets
→ Usually triangular meshes
→ Triangles are locally good approximations of the surface
→ Good for optimization
→ may become difficult to handle for complex surfaces.