Lec12 Optical Flow

Motion field: aka action motion. It's the projection of the 3D scene motion into the 2D image.

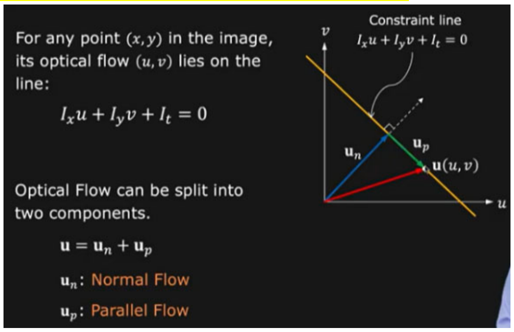
Optical flow: The motion of brightness patterns in the image.

Optical flow estimation: In image, all we have are brightness pattern, so the motion field cannot be measured directly, so we need to use optical flow. Optical flow can be estimated by tracking the displacement of brightness patten, and we hope this patten will corresponds to the motion field.

Q1: Does optical flow always corresponds to motion field? → No, there can be a case that motion field exists but not optical flow(a uniform rotating sphere under fixed lighting), and optical flow exist but not motion field (a change of light source circling around object, e.g., **Sundial**). There is also a special case where both motion field and optical flow exist, but the optical doesn't correspond to the motion field, e.g., **Barber Pole illusion**.

Assumption 1: Brightness constancy constraint. the brightness of the points remains constant, which can be formulated as: $I(x, y, t) = I(x + \delta_x, y + \delta_y, t + \delta_t)$, where $\delta_x, \delta_y = u\delta t, v\delta t$, the displacement of point--- (1)

Assumption 2: Small motion. Points don't move very far, or the displacement (δ_x, δ_y) and time step δ_t are small. → This allows us to use only the first term of Taylor series expansion to estimate the derivative, as shown below: $I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + I_x\delta x + I_y\delta y + I_t\delta t$ --- (2)

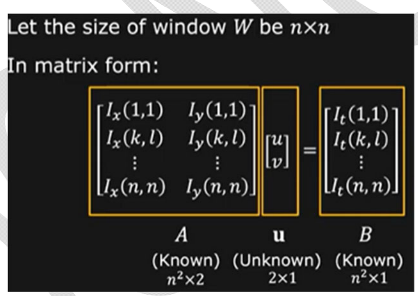


By combing above two equations, we have the: $I_x u + I_y v + I_t = 0$ (Or $\Delta I \cdot [u\ v]^T = -I_t$) --- (3) , where (u, v) is optical flow that we want to estimate.

Q2(Aperture problem) which direction in image along optical flow can't be reliably estimate? → Only normal flow can be estimated/recovered, but not parallel flow or the edges. →Because, we have 1 equation but 2 unknowns, which is an under-constrained problem. → Also, as we see equation (3) involves dot product between the image gradient ∇I and the optical flow **u**, ∇I · **u** + It = 0, which gives no equation on u when u and ∇I are orthogonal.

Assumption 3: Spatial coherence constraint – points move like their neighbors, so the pixel's neighbors have same (u, v). →With this constraint, we can use n neighboring pixels to set up a linear least squares system, which can be solve with SVD or QR decomposition. → The solution for (u, v) is $\mathbf{u} = (A^TA)^{-1}A^TB$, where A is the gradient, ΔI, and B is the time derivative, I_t

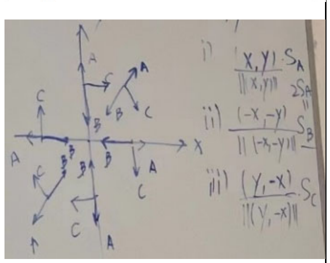
Q3: When is the system solvable? → 1) A^TA is invertible, that is $det(A^TA) \neq 0$; 2)A^TA must be well-conditioned, where the eigenvalues (λ_1, λ_2) are large and fall into the "corner region".



Q4: Where optical flow can fail: 1) Brightness constancy constrain doesn't hold → Use feature matching(e.g., SIFT) or tracking(e.g., Shi-Tomasi feature tracker) to find good matching pairs to estimate optical flow. 2) When motion is large → The higher order terms of Taylor series expansion can no longer be discarded/neglected. → One solution is to use **multi-resolution with iterative refinement**. Iteratively estimate the optical flow at lower resolution and warp the image at the next higher resolution using the estimated optical flow → Alternative is **template matching** but can be computationally expensive. 3) A point doesn't move like its neighbors → **Motion segmentation**: When doing consistency checking, only consider the pixels in the same region.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{-U+xW}{Z} + Axy - B(x^2+1) + Cy \\ \frac{-V+yW}{Z} - Bxy + A(y^2+1) - Cx \end{pmatrix}$$

Q5: Give an expression of (u, v) for three cases that camera moves forward/ backward/counter-clockwise, and sketch flow field includes the flow vectors for $(x, y) = (0, 0)$, $(0, 1)$, $(1, 0)$…



Lec 13: Camera calibration

Camera calibration problem: Given n points of 3D locations $X_i, i \in [1, n]$, and n corresponding image projection x_i, and we want to estimate the projection parameter P. ==> Two approach: **1) Linear method**: use svd to solve linear least squares system → the projection has 11 dof, and each 2D-3D correspondence gives two ind equs, so we need 6 matching pairs (FYI: The linear approach does not work if all 3D points are coplanar). **2) Preferred approach**: In practise, we first initialize the camera calibration matrix with linear method, and refine the solution with non-linear method(e.g., RANSAC) that minimize the error between measured 2D points and esitmated projection of 3D points by triangulating two images, $\sum_i |\mathrm{proj}(K[\,R \mid t\,]X_i) - x_i|_2^2$.

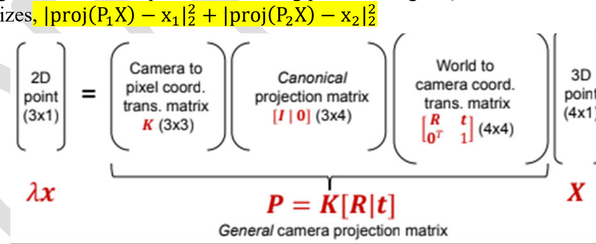
$$\begin{bmatrix} \mathbf{0}^T & X_1^T & -y_1X_1^T \\ X_1^T & \mathbf{0}^T & -x_1X_1^T \\ \dots & \dots & \dots \\ \mathbf{0}^T & X_n^T & -y_nX_n^T \\ X_n^T & \mathbf{0}^T & -x_nX_n^T \end{bmatrix} \begin{pmatrix} \boldsymbol{P}_1 \\ \boldsymbol{P}_2 \\ \boldsymbol{P}_3 \end{pmatrix} = 0$$

Q1: How to get the camera center in world coord from projection matrix? ==> The camera center in world coordinates is in null space of projection matrix P=K[R|t] (use SVD to decompose P and take last row of V).

Q2: Calibrate a new camera: you have a new camera on a robot to perform Visual Odometry, and you want to calibrate it to have more accurate estimation, describe a procedure to do it: → The underlying idea is to take a video on a scene with known distances to create 2D-3D correspondences. For example, you can print out a square grid that segmented with 1' inch black and white blocks. After you identified the location of grid, you can use it to perform calibration.

Triangulation: Given known camera projection matrix P of the same 3D point in two or more images (x1, x2), compute the 3D coordinate of that point ==> Three approaches: 1) **Geometric approach**: Find the shortest segment connecting two viewing rays, and take the midpoint of that segment; 2) **Linear**: Similar to how we estimate the projection matrix. For each matching pair of 2D points, solve the linear least square system to triangulate the 3D position of that point. → The 3D points has 3 unknown, and each matching pairs can give us two ind equs, so 1 matching pair is enough; 3) **Non-linear:** refine X that minimizes, $|\mathrm{proj}(P_1X) - x_1|_2^2 + |\mathrm{proj}(P_2X) - x_2|_2^2$

Lec 14: Single view measurement -- Vanishing Lines



Vanishing line: the line that connects two finite vanishing points is called the vanishing line → How to compute vanishing line from two vanishing points? Take cross product of two vanishing points, e.g., l = a x b

Camera calibration using vanishing points:

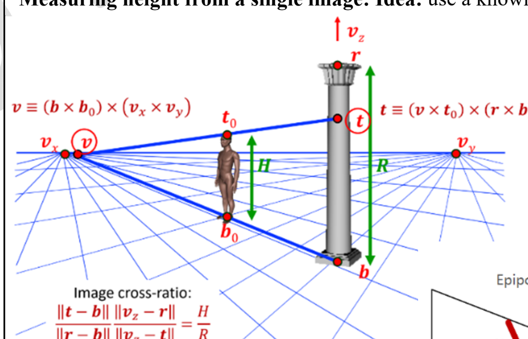
- Can be solved with three orthogonal vanishing points (at least two are finite).
- Good: No need for calibration chart, and 2D-3D correspondences; 2) Could be completely automatic.
- Bad: 1) Only applies to certain kinds of scenes with at least two finite vanishing points; 2) can be tricky to accurately localize vanishing points.

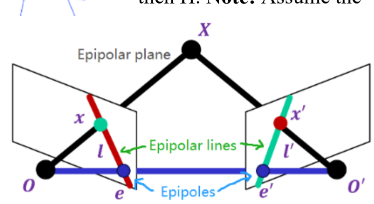
$K^{-1}v_i \equiv Re_i$

Notice: $Re_1 = [r_1\ \ r_2\ \ r_3]\begin{pmatrix}1\\0\\0\end{pmatrix} = r_1$

Thus, $r_i \equiv K^{-1}v_i$

Measuring height from a single image: Idea: use a known height of object (R) as reference and estimate height of other object (H) by using their relative ratio of height in the image. → We first compute all vanishing points (v_x, v_y, v_z), and then we compute v, then t, then H. **Note:** Assume the cross ratio of height invariant to 1) the perspective projection from 3D to 2D; 2) permutation of points





Chap 15: Epipolar Line

Epipolar geometry setup: Baseline is line connecting two camera origins; **Epipoles**(e and e') are where the baseline intersects with two image planes; **Epipolar plane** is the region formed by X, O, O'. Different X will form different epipolar planes. There is a family of planes passing through O and O'.

Different configurations of epipolar setup:

- **Converging cameras:** Epipoles are visible in the image when two images formed an angle.
- **Motion parallel to image plane**: Epipoles are not visible in the image, when two images taken at a parallel line, epipolar lines parallel
- **Motion perpendicular to image plane**: Epipoles coincides at the principal point of the camera, and epipolar lines algo go through the center of image.

Epipolar Constraints: 1) For a point x in an image, the corresponding point x' in another image must lie along the epipolar line; 2)Potential matches for x must lie on the epipolar line l', and matches for x' also must lie on l'; 3) Whenever two points x and x' lie on the matching epipolar lies, the visual rays corresponding to them meet in space, i.e., x and x' are the projections of same 3D point X.

Math of epipolar constraint:

In calibrated case, we know intrinsic and extrinsic, so we can premultiply K to get normalized image coordinates, where $x'^T[t_\times]Rx = 0 \rightarrow x'^TEx = 0$, where E is $[t_\times]R$.

Last Update: 05/16/2022, Zhengqi Dong

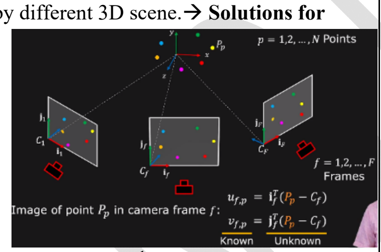
Property of E: 1) Ex is epipolar line associated with x; 2) $E^T x$ is the epipolar lines associated with x'; 3) $Ee = 0, E^T e' = 0$; 4)E is singular(rank 2) and has 5 DOF;
In uncalibrated case, we have to compute F, which is $K'^{-T} E K^{-1}$. Property of F: First 3 is same as E, and for last one, F is singular(rank 2) with 7 DOF. → How to estimate F? We have $x'^T F x = 0$.
Eight-point algo: Solve least square equation, enforceing singularity by taking SVD and drop the smallest singular value. Normalized eight-point algo. Why? → x, y, x', y' are pixel coordinates, they have different magnitude and might cause numerical instability. Solution: transform images to a new coordinate system, the new coordinate system has its origin at centroid of points. After translation, scale the coord so that the mean squared distance btw origin and points is 2 pixels. Then run eight-point algo. Enforce rank 2. Transform F back to its original unit. Suppose T and T' are being normalized, and $F = T'^T F T$.
Q1: Is it possible to find the depth of 3D point Q from affine camera(aka weak perspective or orthographic camera)? What about perspective camera? → No for affine, because all 3D objects transform to 2D image with parallel projection, so we can't find the depth; Yes for perspective camera, because we can do triangulation, and we have depth(x) = Bf/(x – x).
Q2: What is an epipolar line? How is it determined? How can the concept be used to simplify the correspondence problem in stereo vision? → **Epipolar lines** is the line connecting epipoles to the projection of X in other image; All epipolar lines intersect at the epipole, so a point x in one image generates a line in the other on which its corresponding point x' must lie. → Thus, it can help to reduce the search space for correspondence in stereo matching.

Lec 16: SFM

Problem: Given corresponding image point(2D), x_i,j, ith point in jth frame, and we want to find its corresponding 3D point X_j, $x_{ij} = A_i X_j + t_i, i = 1, \dots, m, j = 1, \dots, n$ → In practice, we can assume there is a transformation Q (a full rank 4x4 matrix), and we can apply some constraint on it, and the observation/data/measurement matrix(D) remain unchanged.

$$x \equiv PX = (PQ^{-1})(QX)$$

SFM ambiguity: SFM is not uniquely solvable, and there are many SFM ambiguity we need to account for: 1) **Projective ambiguity**(No constrain on Q): points in two images looks the same, but they were projected by different shape of object. → still preferred in practice, bez it makes the least assumption about the world; 2) **Affine ambiguity**: Imposed parallelism constraint on Q. It consists of a full rank matrix and a translation vector $[[A, t], [0^T, 1]]$. 3) **Similarity ambiguity**(one with the least ambiguity): Enforced orthogonality constraint, but still has scaling ambiguity, e.g., scaled the entire scene by k and scale the camera matrices by 1/k, the two images look same but were projected by different 3D scene.→ **Solutions for SFM: Affine SFM**:



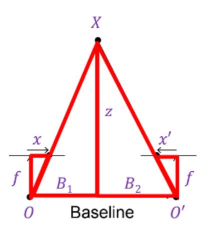
- Assume orthographic projection
- Input: 2D image points D → Output: Camera parameters(M) and 3D scene points matrix (S).
- Use **centering trick** to eliminate one unknown C_f → Basically we assume the origin of world is at the centroid of scene points, and we will shift(subtract) each image point by the centroid in each view, that's $\widehat{x_{ij}} = x_{ij} - \frac{1}{n}\sum_{k=1}^{n} x_{ik}$ → then we have: $\widehat{x_{ij}} = A_i \widehat{X_j}$, where each 2D points is being normalized.
- Q1: How many knowns and unknowns for m images and n points? → 2mn knowns and 8m + 3n unknowns, and we must have 2mn ≥8m + 3n – 12, e.g., given 10 images(m=10), we need at least six points(n>=6)

$$\underset{\text{data matrix }(2m \times n)}{D = \begin{bmatrix} \hat{x}_{11} & \hat{x}_{12} & \cdots & \hat{x}_{1n} \\ \hat{x}_{21} & \hat{x}_{22} & \cdots & \hat{x}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{x}_{m1} & \hat{x}_{m2} & \cdots & \hat{x}_{mn} \end{bmatrix}} = \underset{\substack{\textbf{M}\\ \text{cameras } (2m \times 3)}}{\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{bmatrix}} \underset{\substack{\textbf{S}\\ \text{points } (3 \times n)}}{\begin{bmatrix} X_1 & X_2 & \cdots & X_n \end{bmatrix}}$$

Q2: What must be the rank of data matrix(D)? → Rank(D) = Rank(MS) ≤ $min(2m, 3, n)$ = 3
Q3: How to deal with missing data in the data matrix(D)? → 1) Blocking method: Decompose matrix into dense sub-blocks, factorize each sub-block, and fuse the results; 2) Incremental bilinear refinement: iterate through each dense sub-block, that factorization, triangulation at two known cameras, and calibrate for new frame sees at least three known 3D point.

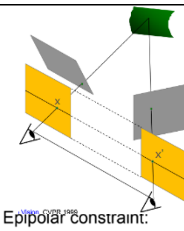
Lec 17: Two-view stereo and depth prediction

Basic stereo matching algo: If necessary, rectify two stereo images to transform epipolar lines into scanlines. For each pixel in the first image, find corresponding epipolar scanline in the right image, examine all pixels on the scanline and pick the best match, triangulate the matches in first and second image to get depth information, by computing disparity(x-x'), and depth(x) = Bf/(x – x) → Larger baseline: smaller triangulation error, but matching is more difficult, and vice versa.



Local Stereo matching algorithm: Instead of matching pixels(noisy and time consuming), we match a small window in each image. Slide a window along the scanline and compare contents between the window of two images, with matching criteria SSD or Cross Correlation. → **Effect of window size:** smaller window give us more detail but more noise; Larger window result smoother disparity depth map but less detail. →**Where basic window search will fail:** 1) Textureless surfaces (无纹理表面); 2) Occlusion, repetition; 3) n on-lambertian surfaces, e.g., with specular highlight

Non-local constraints of stereo matching: 1) Uniqueness: Each point in one image should match at most one point in the other image, but its uniqueness doesn't always hold, can be affected by the depth of point in real world; 2) Ordering: corresponding points should appear in the same order, but the order can be affected by depth as well; 3) Smoothness: the disparity value(x-x') should change slowly.



$x'^T E x = 0, \quad E = [t_\times]R$

$R = I \quad t = (t, 0, 0)$

$E = [t_\times]R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t \\ 0 & t & 0 \end{bmatrix}$

$(u' \ v' \ 1)\begin{pmatrix} 0 \\ -t \\ tv \end{pmatrix} = 0 \qquad -tv + tv' = 0 \qquad v = v'$

$(u' \ v' \ 1)\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t \\ 0 & t & 0 \end{bmatrix}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = 0$

Stereograms: aka 3D lens, humans use fuse pairs of images to get a sensation of depth.
Parallel images: image planes of camera are parallel to each other and to the baseline; Camera centers are at the same height; Focal length are the same; Eipolar line fall along horizontal scanlines of the images.
Stereo image rectification: If the image planes are not parallel, we can find a homography to project each view onto a common plane parallel to the baseline.

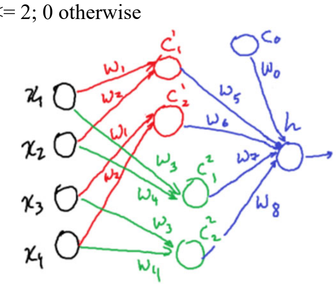
Lec18: Multi-view stereo

Goal: Given arbitrary number of images(>= 2) of same object or scene, we want to reconstruct a representation of its 3D model (e.g., depth maps, meshes, point cloud, patch clouds, volumetric model, …). The camera position is arbitrary, and camera calibration can be known or unknown. → Useful for reconstructing 3D disaster scene for inspection or interaction by drone photos.
Why? → 1) more reference view for some points being occluded or some surfaces are foreshortened in certain view; 2) high-res closeups of some regions; 3) Reduce errors.
Plane sweep stereo(Fast version): Sweep plane across a range of depths w.r.t. a reference camera. For each depth plane, project each input image onto that plane using homography, and for each pixel in the composite image stack, compute the variance. The depth plane with the lowest variance is selected. (In practice, we will merge multiple depth maps that was computed for each view into a volume or a mesh.
Patch-based multi-view stereo: 1)Detect keypoints; 2)Triangulate a sparse set of initial matches; 3) Iteratively expand matches to nearby locations; 4) Use visibility constraint to filter out false matches;5)Perform surface reconstruction

Part5: Recognition, CNNs

- (A XOR B) can be expressed as (A OR B) AND NOT(A AND B))
- z = $x1 * w1 + x2 * w2 + b$
- Activation func: f(z) = 1, if x>=2; or -1 if x <= 2; 0 otherwise

Q1: Draw a conv net with $x \in R^4$, one hidden layer with 2x1 filters and 2 channels with stride of 2, a fully connected layer with one neuron as output, how many params does it have? → 9 params in total. Note: c0 is bias, and only fully connected layer has bias.



Perceptron Learning rule: Cycle through training examples; Update the weight and biased if perceptron didn't correctly classify the input data, w(i+1) = w(i) + lr *y(i)*x(i) → Strengthen it if it fails to fire, weaken it if it misfired.
Learning types: Unsupervised(no labels, e.g., clustering, dimensionality reduction, manifold learning), Semi-supervised (labels for a small portion of training data); Weakly supervised (noisy labels, labels not exactly for the task of interest); Supervised (clean, complete training labels for the task of interest).
Gradient descent(Update upon whole batch): cycle through the entire training set; Start with some initial estimate of w; At each step, find ∇L(w), the gradient of the loss w.r.t. w, and take a small step in the opposite direction: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha\nabla\hat{L}(\boldsymbol{w})$; **SGD** → Perform parameter update for a single data point.
K-fold cross-validation: Partition the data into K groups; In each run, select one of the groups as the validation set.
hyperparameter: "complexity" of model controlling its generalization ability, e.g., number of layers, number neurons/layer, regular terms? lr? epoch? …
underfitting: training and test error are both high. Caused by high bias, incapable of capturing the import features of training data; **overfitting**:low training error, but high testing error. Caused by high variance, fitting noise and unimportant charas of training data, and perform poorly in testing data.
GAN: learn to sample from the distribution represented by the training set; 1) **Generator**: learns to generate samples, to fool discriminator; 2) Discriminator: learns to distinguish between generated and real samples
Spatial pyramids: Orderless pooling of local features over a coarse grid.
Reinforcement learning: Learn from (possibly sparse) rewards in a sequential environment.
Active learning: The learning algorithm can choose its own training examples, or ask a "teacher" for an answer on selected inputs.
ROI pooling: "crop and resample" fixed size representing ROI out of output of last conv layer(use NN interpolation or max pooling)
RPN(region proposal n etwork): put an "anchor box" of fixed size over each position in the feature map and try to predict whether this box is likely to contain an object (can be multi-scale).
Transpose conv vs normal conv: Transpose conv is also known as deconvolution. Transpose conv perform up-sampling to undo the previous operation, and normal conv can only do down-sampling. Another unsampling method is max unpooling.
"Shallow" pipeline: hand-crafted feature representation followed by trainable classifier, e.g., bad of visual words, texton models, and etc.