NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

EE6222 Machine Vision

Topics 13/14
Vision Beyond Image 2: 3D Machine Vision (2) and Revision/Recap

Dr. Xu Yuecong

15 Nov 2023

Topics for Week 12 and 13

Topics Outline:

- Fundamentals of camera models. (12)
- Fundamentals of single-view geometry and stereo/epipolar geometry. (12)
- **(Deep) Learning-based 3D vision analysis with multi-modal information. (13)**
- **Latest developments in 3D vision analysis (mostly with deep learning) (13)**

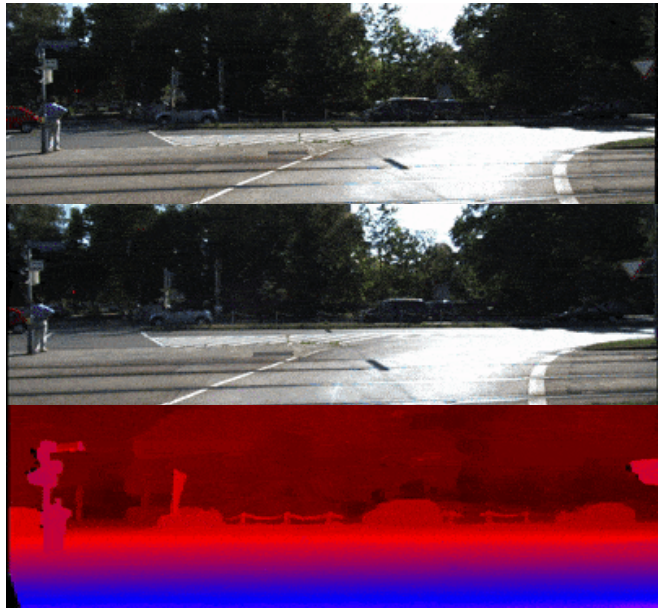
Reference (Textbook):

- Hartley, & Zisserman, A. (2004). Multiple View Geometry in Computer Vision. Cambridge University Press. https://doi.org/10.1017/CBO9780511811685 (Available in NTU Library)

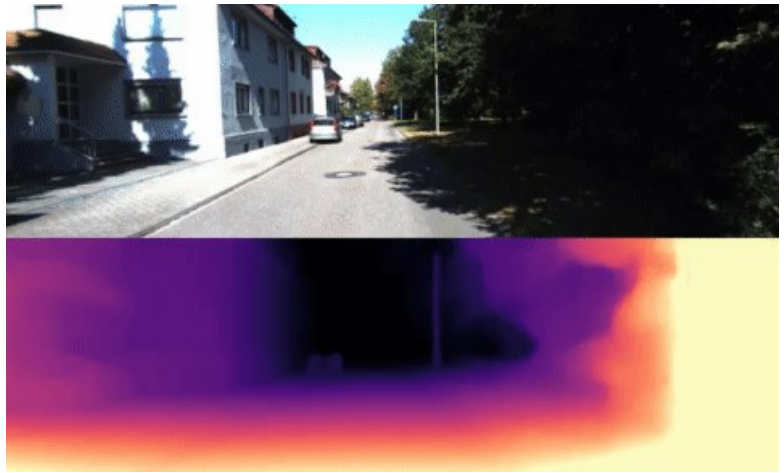
Some Admin Issues on CA

- ✓ Please submit your assignment (PDF only) by Monday Week 14 (20 Nov).
- ✓ Penalty mark will be given for the following conditions:
  - ✓ Wrong naming format of PDF file or other file formats (2 mark deduction);
  - ✓ Late submission (5 mark deduction);
  - ✓ No code (5 mark deduction) – Provide all you codes as screenshot in the Appendix;
  - ✓ No submission (15 mark deduction);
  - ✓ Plagiarism (15 mark deduction + school report).
- ✓ If you have any special circumstance (e.g., serious illness) that may require extension of the assignment deadline, please email me and cc Prof. Jiang before 17 Nov. Late request may NOT be entertained.
- ✓ Please read through the explanatory notes given on Week 11 as well as looking at the corresponding recording for what I look for in this CA.

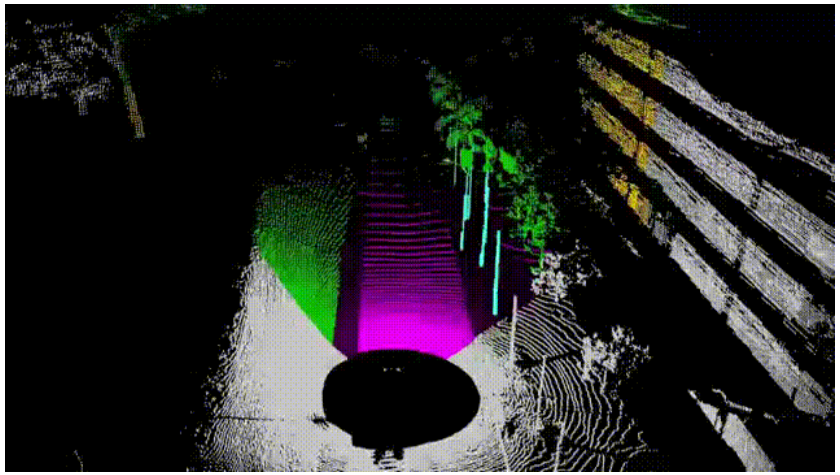
Learning-based 3D Vision: A Brief Outline through Tasks



Depth Estimation: Binocular



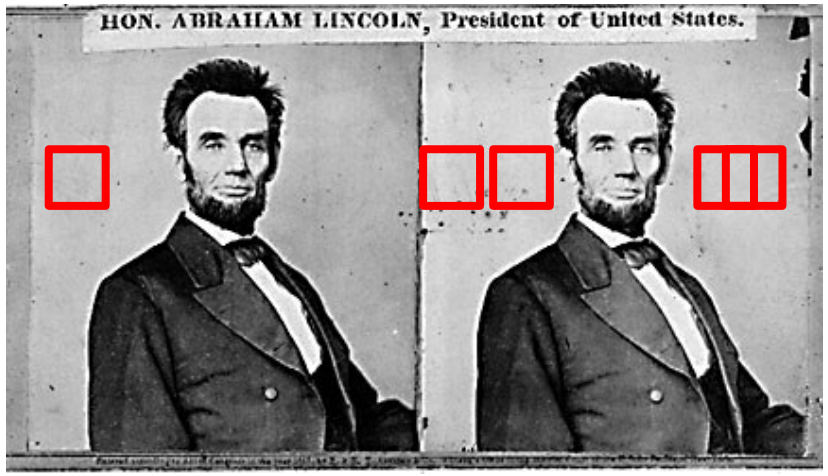
Depth Estimation: Monocular

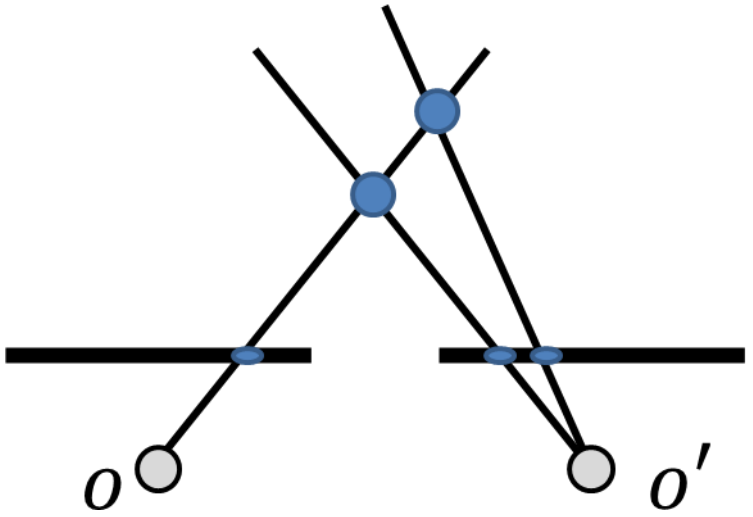


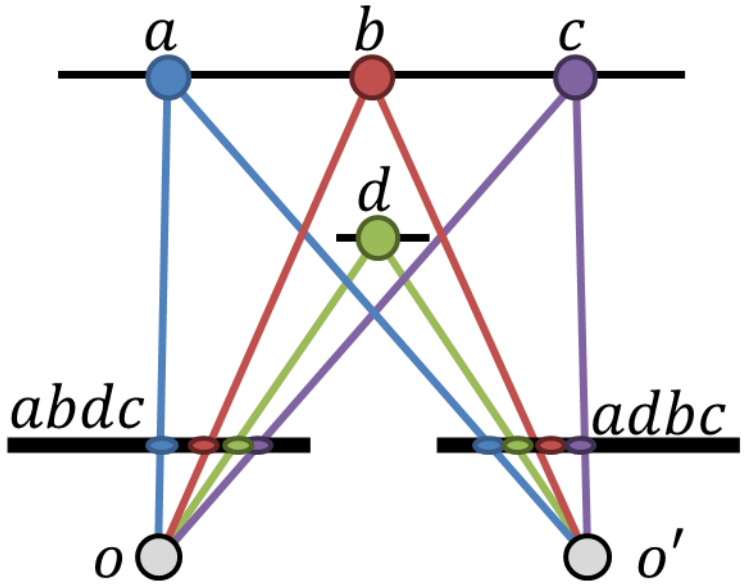
3D Segmentation
(with Point Cloud)

Why Learning-based? (Especially for Depth Estimation)

- Most 3D vision tasks start from 2D images. (Will talk about point cloud later)
- The key towards depth-estimation and most 3D vision tasks: find correspondence across multiple views of images.
- Is finding correspondence an easy task? Maybe, but in general applications, NO.
- Focus of learning-based method: Any workaround to estimate depth without correspondence?
  - Disparity Map (finding disparity ≈ finding depth)

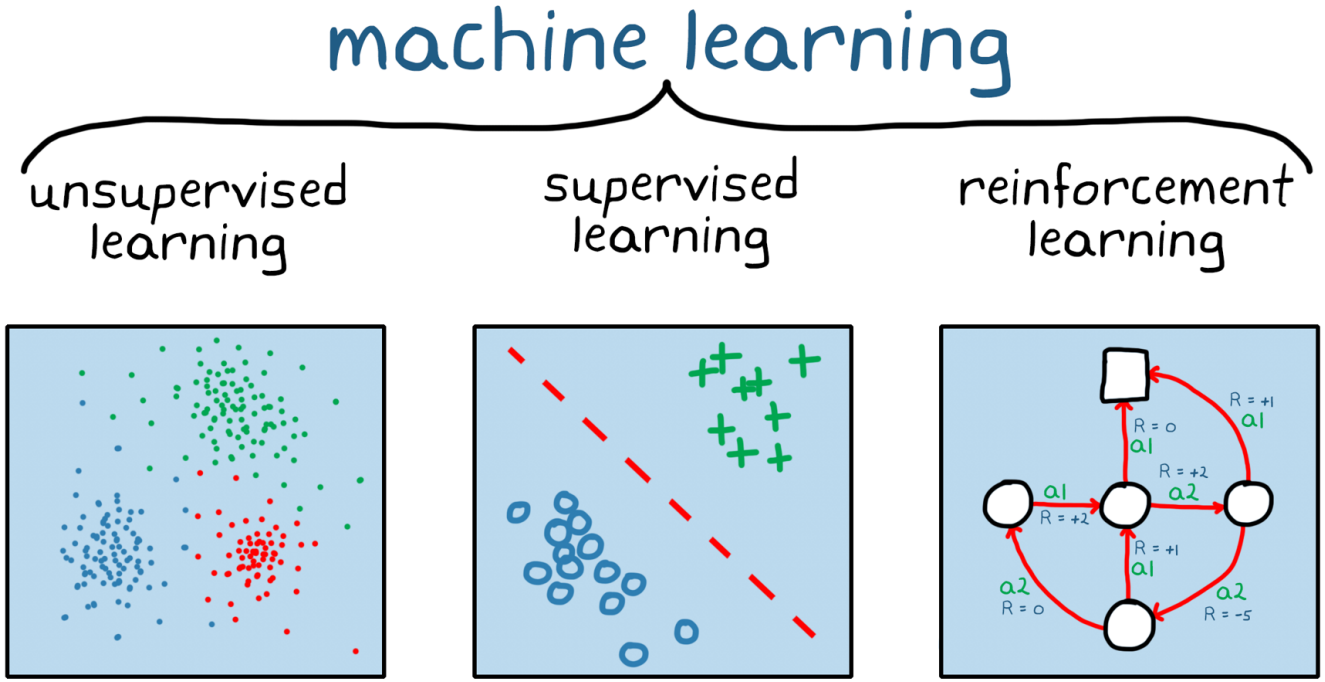






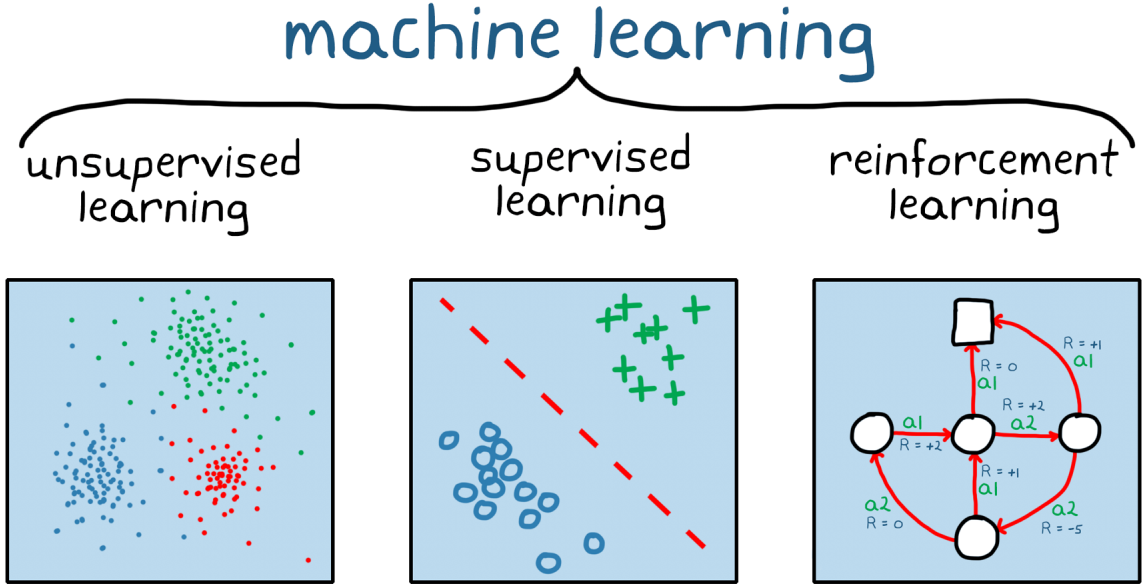
Supervised Learning? Unsupervised? Reinforcement/Adversarial?

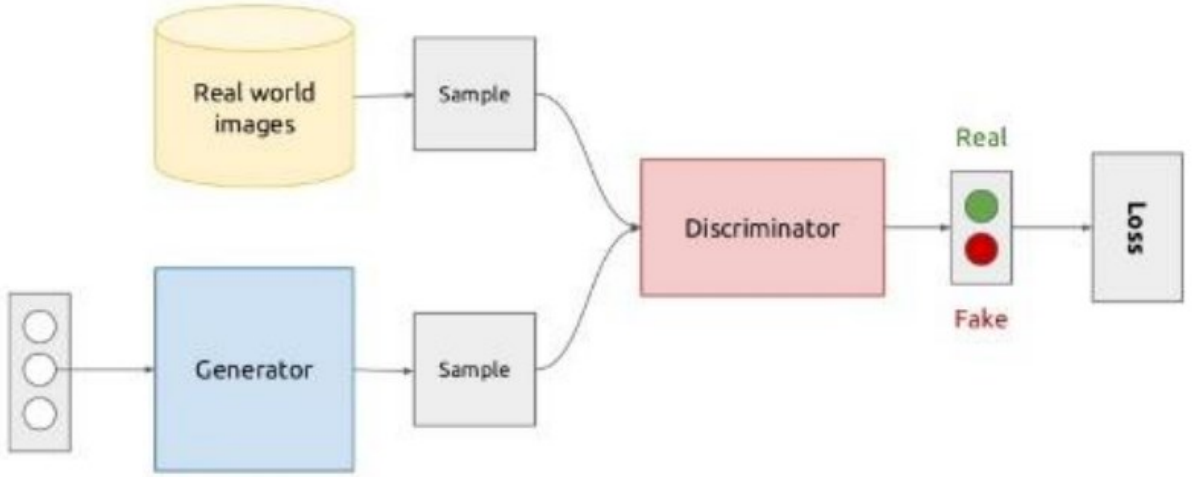
- Most action recognition methods we introduced are supervised learning, labels are provided together with data.
- Problem with supervised learning: labels needed!
- Self-supervised learning methods usually adopts an unsupervised learning strategy.
- Reinforcement learning: learning with feedback from the environment.



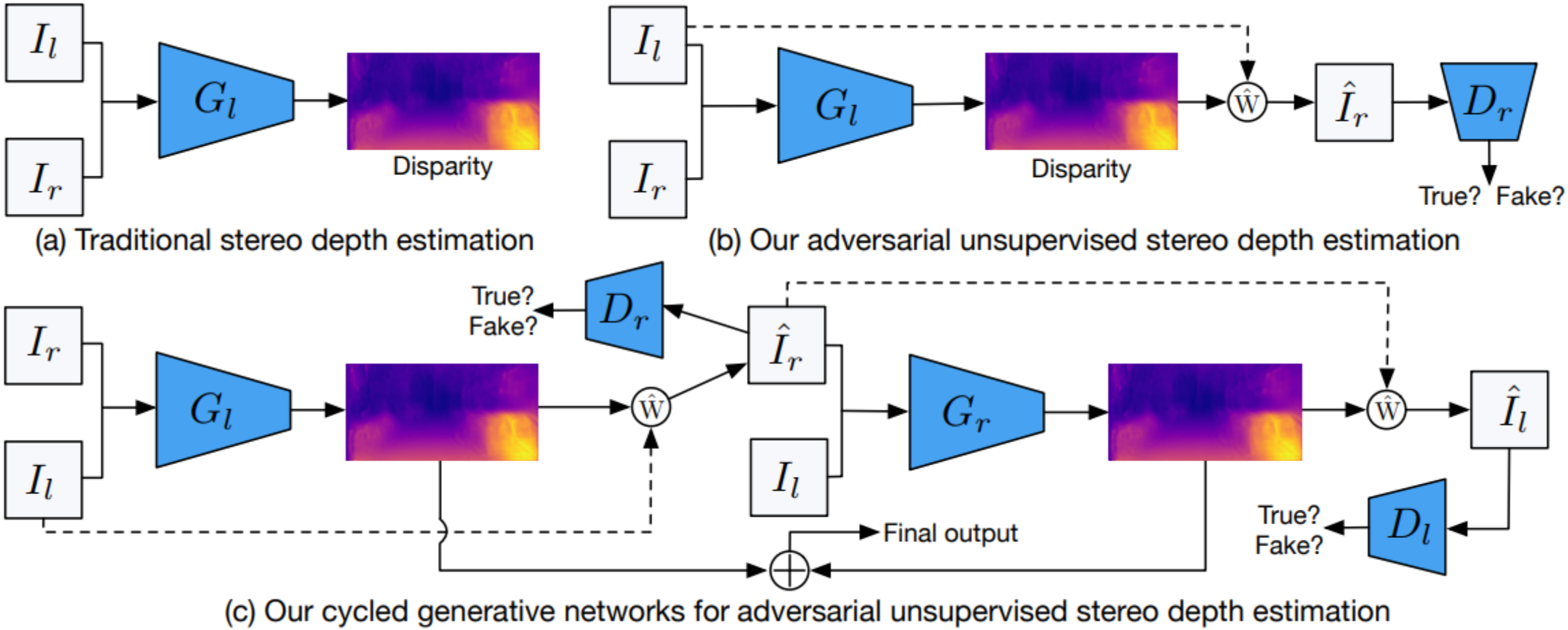
Supervised, Unsupervised, Reinforcement, Adversarial Learning?

- Most action recognition methods we introduced are supervised learning.
- Problem with supervised learning: labels needed! – Unsupervised learning then!
- Self-supervised learning methods usually adopts an unsupervised learning strategy.
- Reinforcement learning: learning with feedback from the environment.
- Adversarial learning: obtain more generalizable features that are close to features from real data through discriminating generated/real data (an auxiliary task).





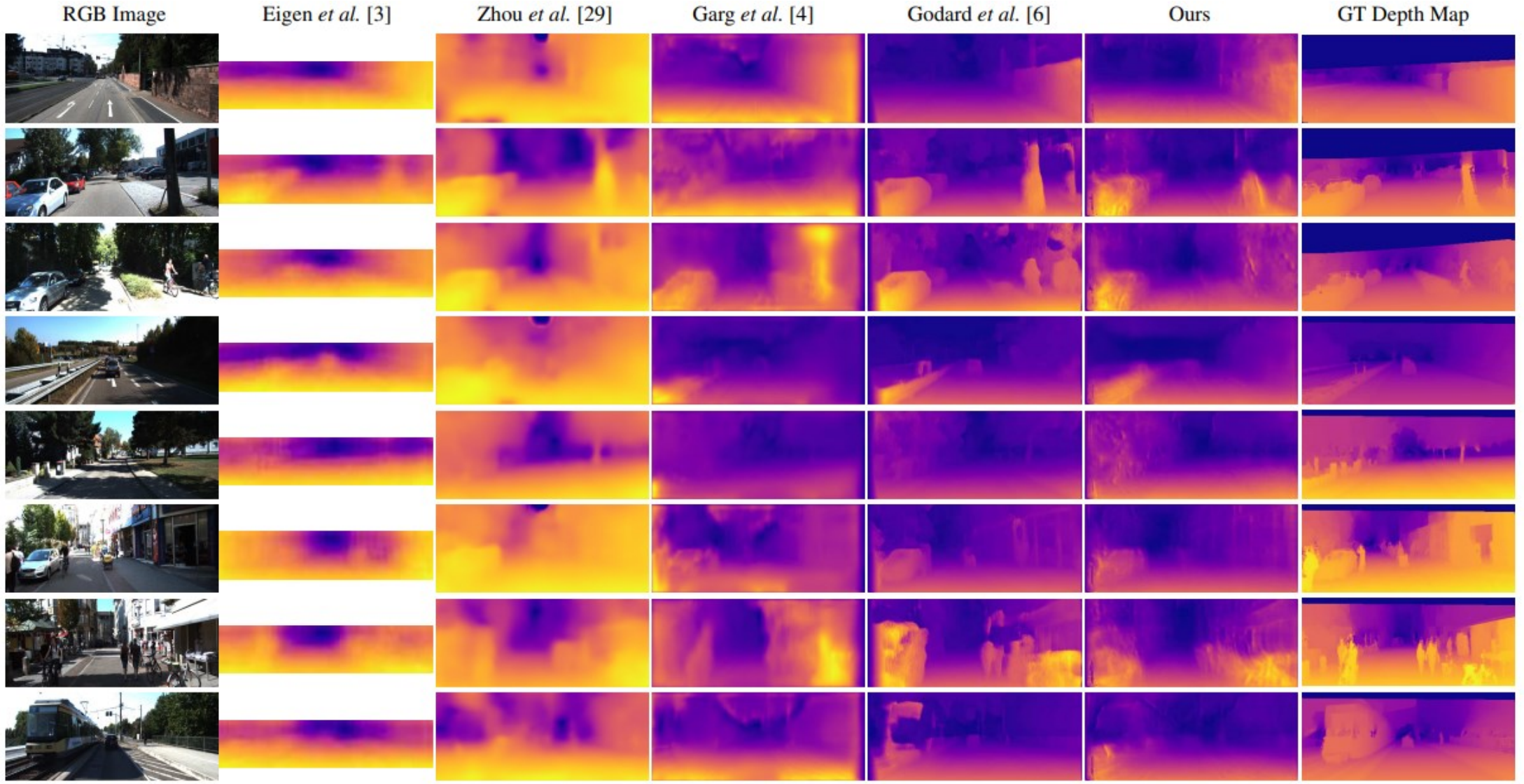
Unsupervised Adversarial Depth Estimation using Cycled Generative Networks



Unsupervised Adversarial Depth Estimation using Cycled Generative Networks

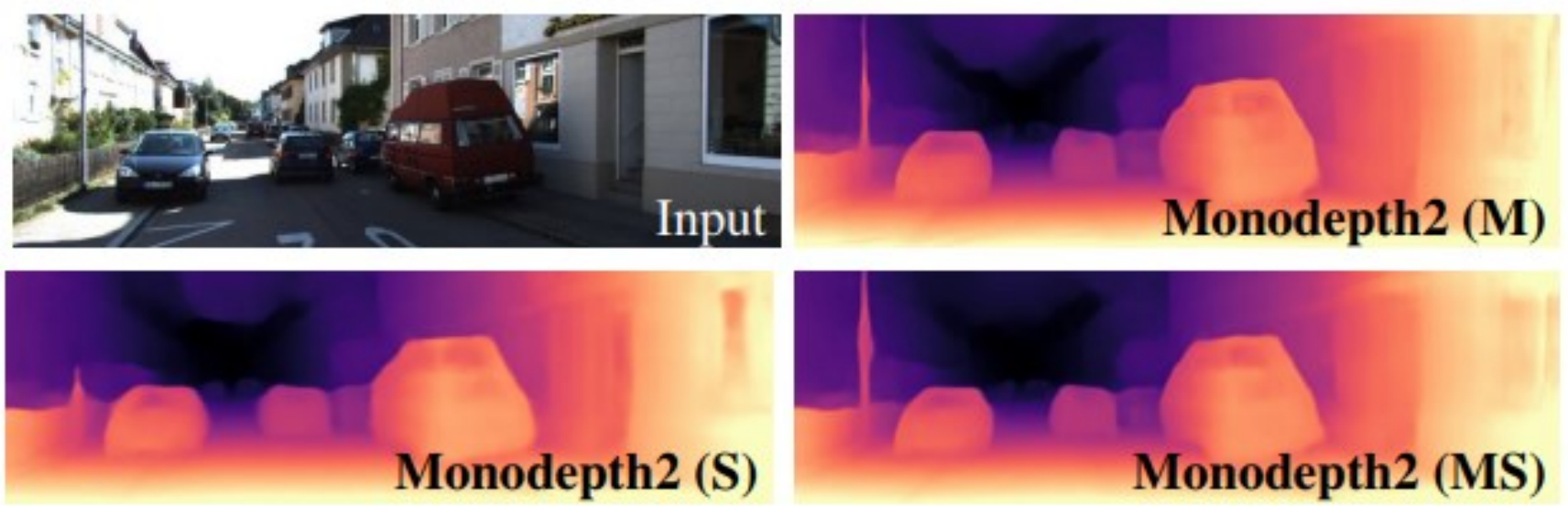
Method	Sup	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
		lower is better				higher is better		
Saxena *et al.* [18]	Y	0.280	-	8.734	-	0.601	0.820	0.926
Eigen *et al.* [3]	Y	0.190	1.515	7.156	0.270	0.692	0.899	0.967
Liu *et al.* [13]	Y	0.202	1.614	6.523	0.275	0.678	0.895	0.965
AdaDepth [9], 50m	Y	0.162	1.041	4.344	0.225	0.784	0.930	0.974
Kuznietzov *et al.* [10]	Y	-	-	4.815	0.194	0.845	0.957	0.987
Xu *et al.* [24]	Y	0.132	0.911	-	0.162	0.804	0.945	0.981
Zhou *et al.* [29]	N	0.208	1.768	6.856	0.283	0.678	0.885	0.957
Garg *et al.* [4]	N	0.169	1.08	5.104	0.273	0.740	0.904	0.962
AdaDepth [9], 50m	N	0.203	1.734	6.251	0.284	0.687	0.899	0.958
Godard *et al.* [6]	N	**0.148**	**1.344**	**5.927**	**0.247**	**0.803**	**0.922**	**0.964**
Ours, 80m	N	0.166	1.466	6.187	0.259	0.757	0.906	0.961
Ours with shared enc, 80m	N	0.152	1.388	6.016	0.247	0.789	0.918	0.965
Ours, 50m	N	0.158	1.108	4.764	0.245	0.771	0.915	0.966
Ours with shared enc, 50m	N	**0.144**	**1.007**	**4.660**	**0.240**	**0.793**	**0.923**	**0.968**

Unsupervised Adversarial Depth Estimation using Cycled Generative Networks



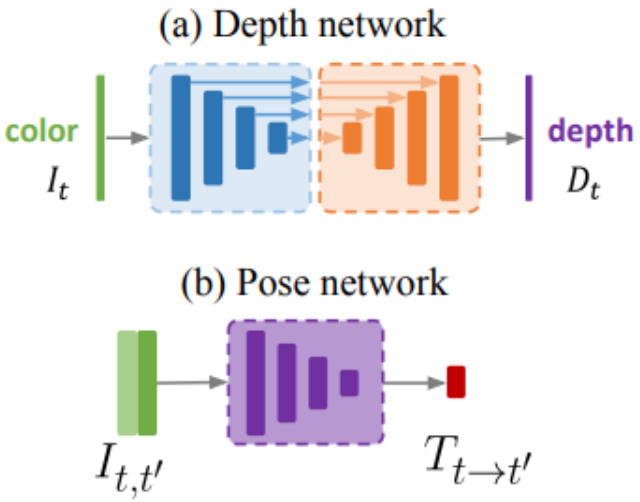
Binocular or Monocular?

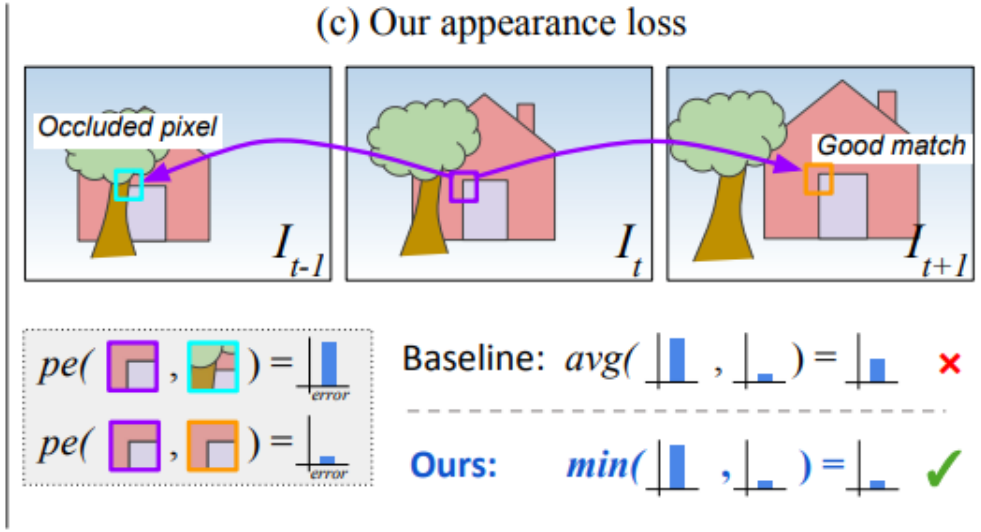
- Humans normally perceive depth information with two eyes → Binocular
- Can humans perceive depth information with one eye → Monocular?
- Actually, in most cases it is possible, and you can't observe much difference to that of using two eyes!
- How? Human observe "videos", not "still images".

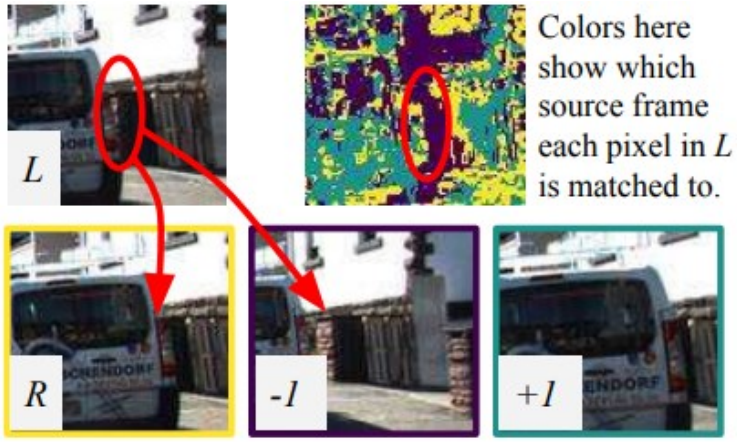


Digging Into Self-Supervised Monocular Depth Estimation

- Binocular/Multi-view depth estimation is a more developed technique → convert monocular depth estimation into a binocular/multi-view depth estimation problem!
- Different frames act as the different views → apply strategies of binocular/multi-view depth estimation.
- Not all information are useful! Drop information when needed!







Digging Into Self-Supervised Monocular Depth Estimation

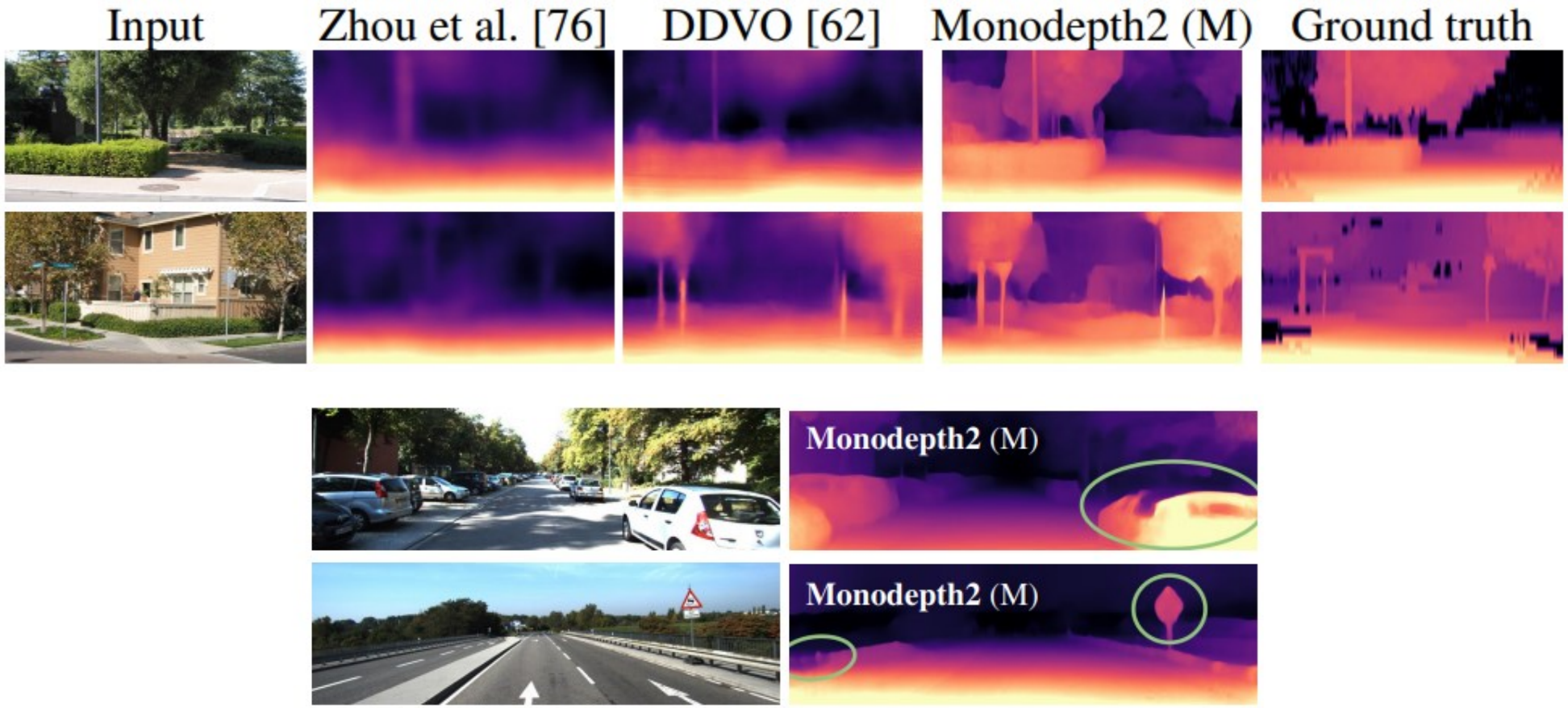
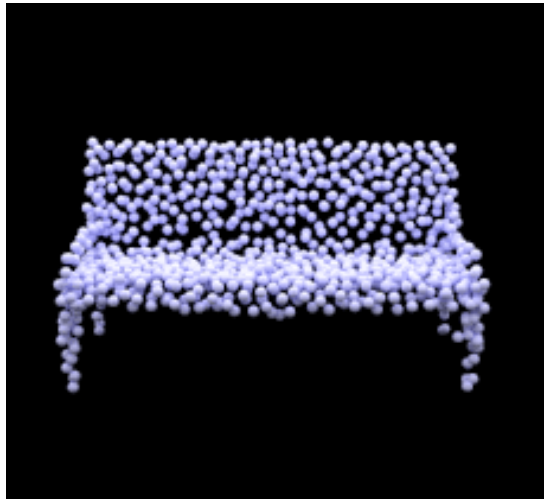
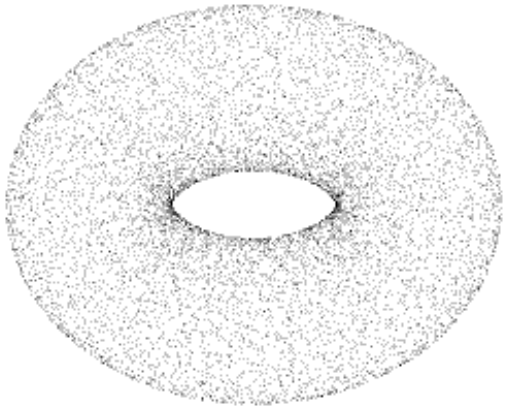


Figure 8. **Failure cases. Top:** Our self-supervised loss fails to learn good depths for distorted, reflective and color-saturated regions. **Bottom:** We can fail to accurately delineate objects where boundaries are ambiguous (left) or shapes are intricate (right).

3D Vision: Not just RGB – Point Cloud

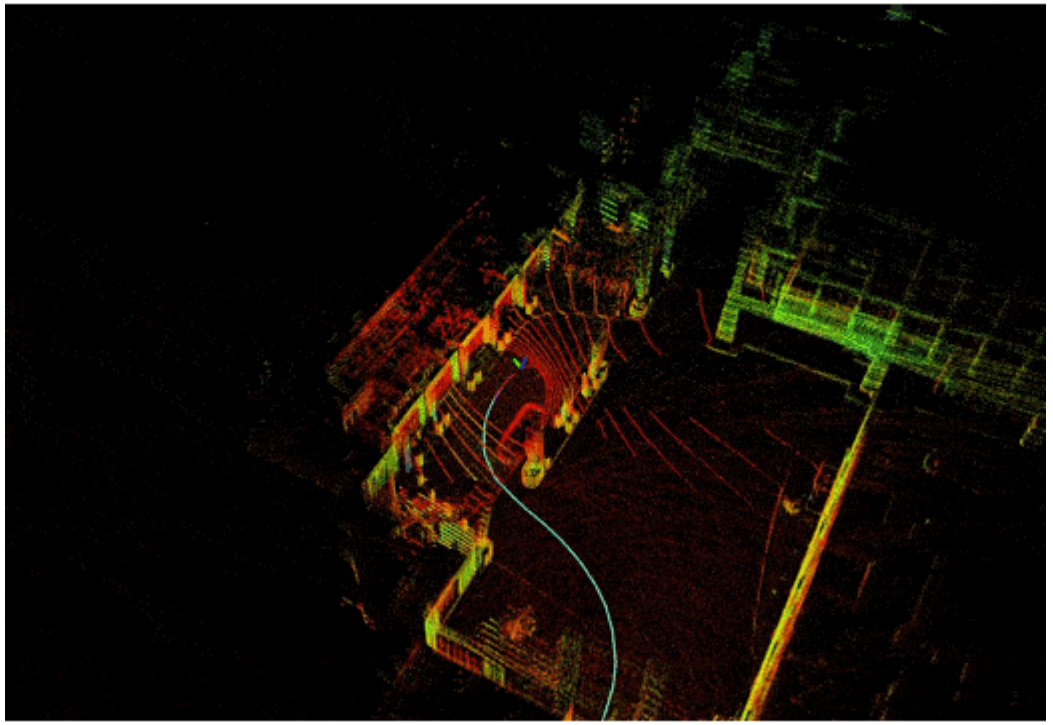
- A Point Cloud is an unordered set of 3 dimensional points in a frame of reference (Cartesian coordinate system) on the surface of objects.
- $\boldsymbol{P = \{(x_i, y_i, z_i) | i \in N\}}$
- More information can be associated with the cartesian coordinates: e.g., RGB color information associated with each point, Normal to surface at each point, Information about meshes (vertices & edges).
- Dense / Sparse, it gives the information of the observed object / environment in 3D space.





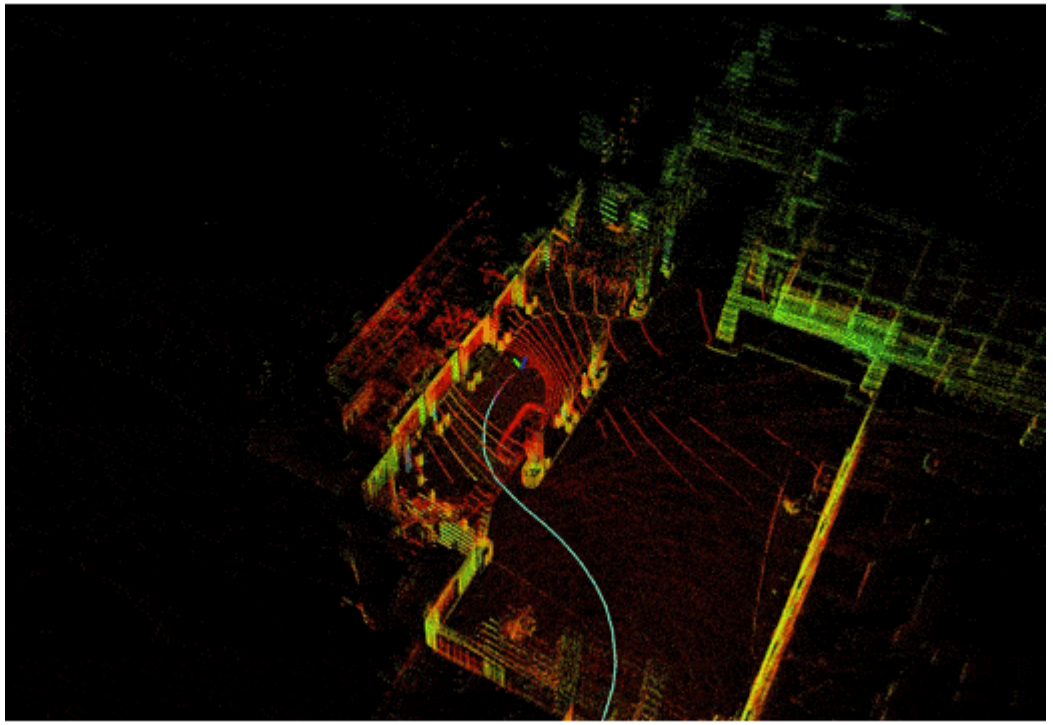
3D Vision: Point Cloud

- Applications: Reconstruction and SLAM (Simultaneous Localization And Mapping)



3D Vision: Point Cloud

- Pros:
  - Point cloud models can accurately represent relatively complex objects with a finite number of elements (points).
  - Point cloud models can be the quickest to create -- assuming you have access to 3D scanning technology or CAD conversion software.
- Cons:
  - Point-cloud models lack the precision and cannot create mathematically perfect curves.
  - Point-cloud data does not include information on surfaces, and therefore cannot be used natively for rendering or manufacturing.



PointNet: (Simple) Deep Learning on Point Sets for 3D Classification and Segmentation

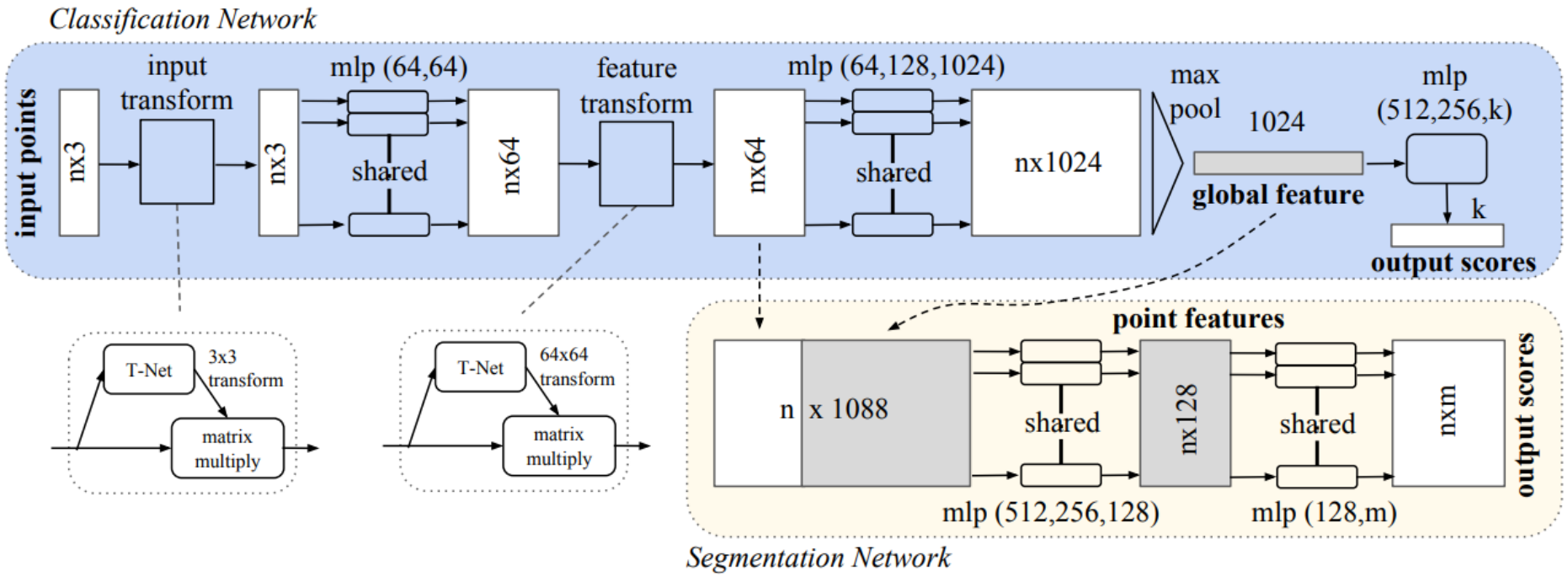


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

PointNet: (Simple) Deep Learning on Point Sets for 3D Classification and Segmentation

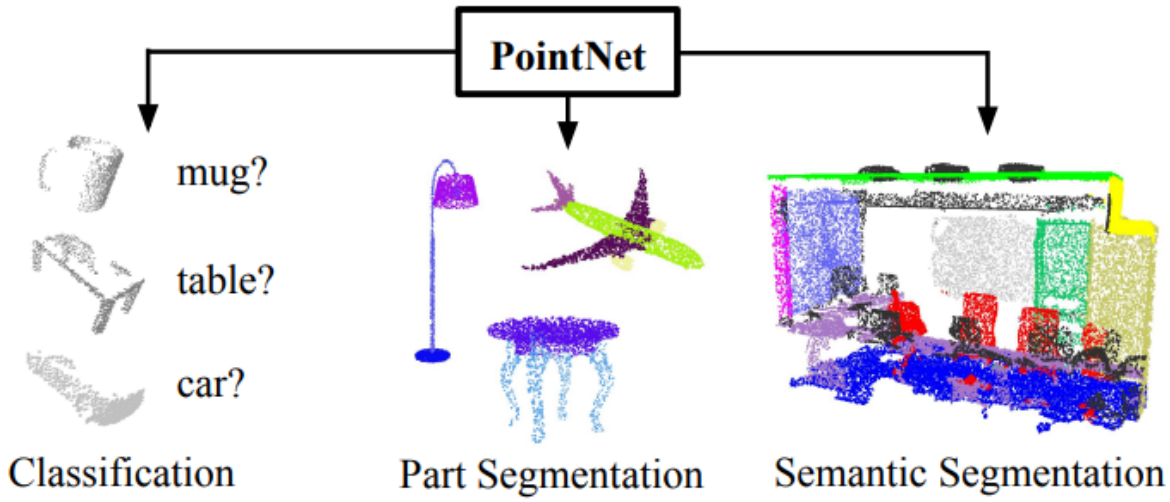
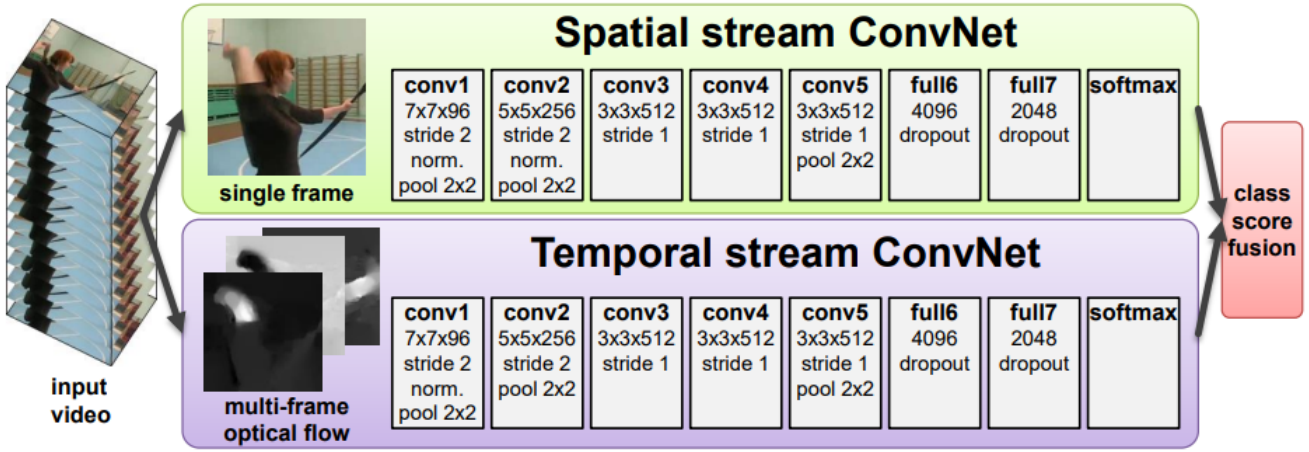


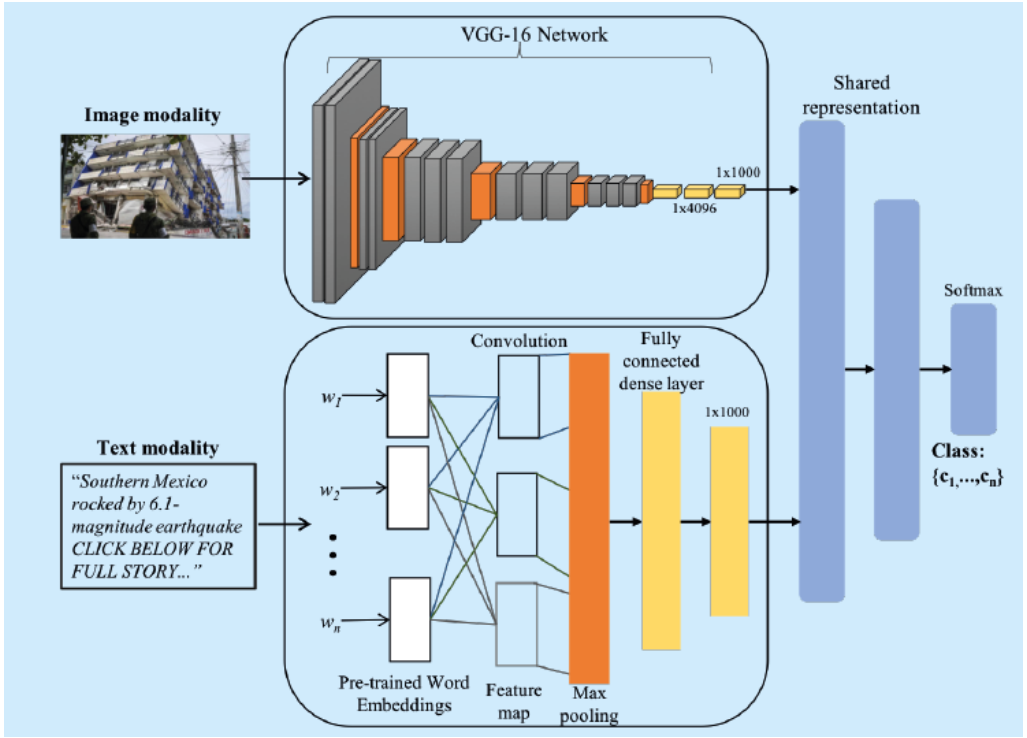
Figure 1. **Applications of PointNet.** We propose a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

Figure 4. **Qualitative results for semantic segmentation.** Top row is input point cloud with color. Bottom row is output semantic segmentation result (on points) displayed in the same camera viewpoint as input.

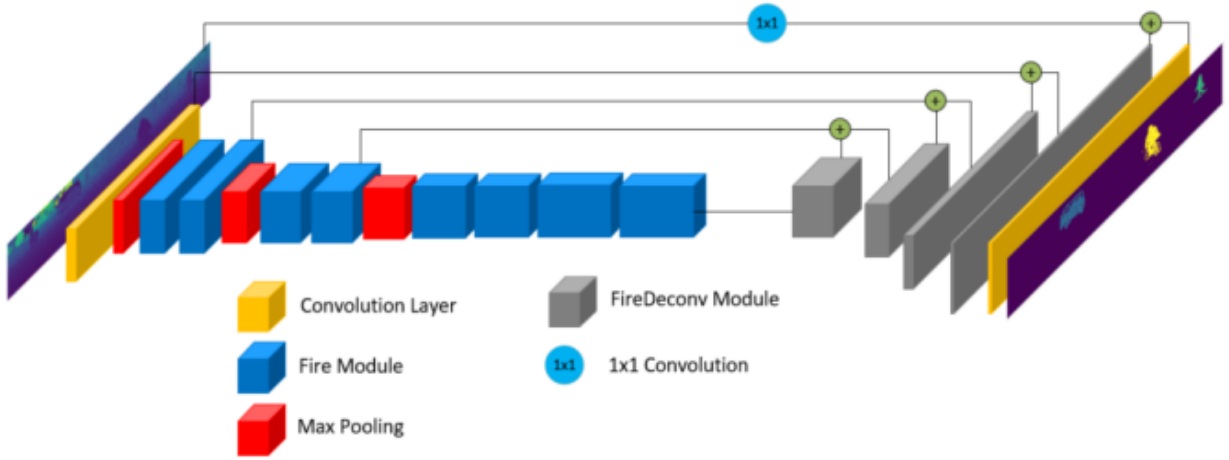
Multi-Modal Learning and 3D Vision

- Information from a **single** modality can have various constraints (e.g., the pros and cons of point cloud / RGB images).
- One intuitive but also effective way of learning: multi-modal learning!
- Action Recognition: Optical flow + RGB; 3D: Point cloud + RGB; Text + Image

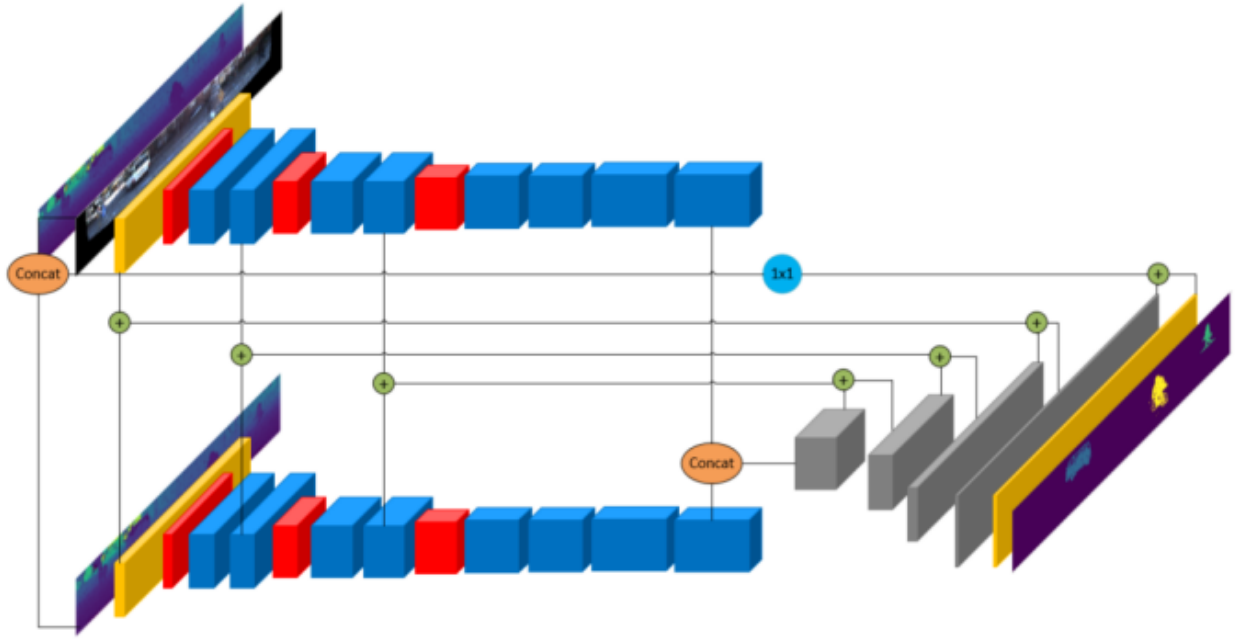




RGB and LiDAR fusion based 3D Semantic Segmentation for Autonomous Driving



(a) LiDAR baseline architecture based on SqueezeSeg [22].



(b) Proposed RGB+LiDAR mid-fusion architecture

- Similar to slow fusion!

RGB and LiDAR fusion based 3D Semantic Segmentation for Autonomous Driving

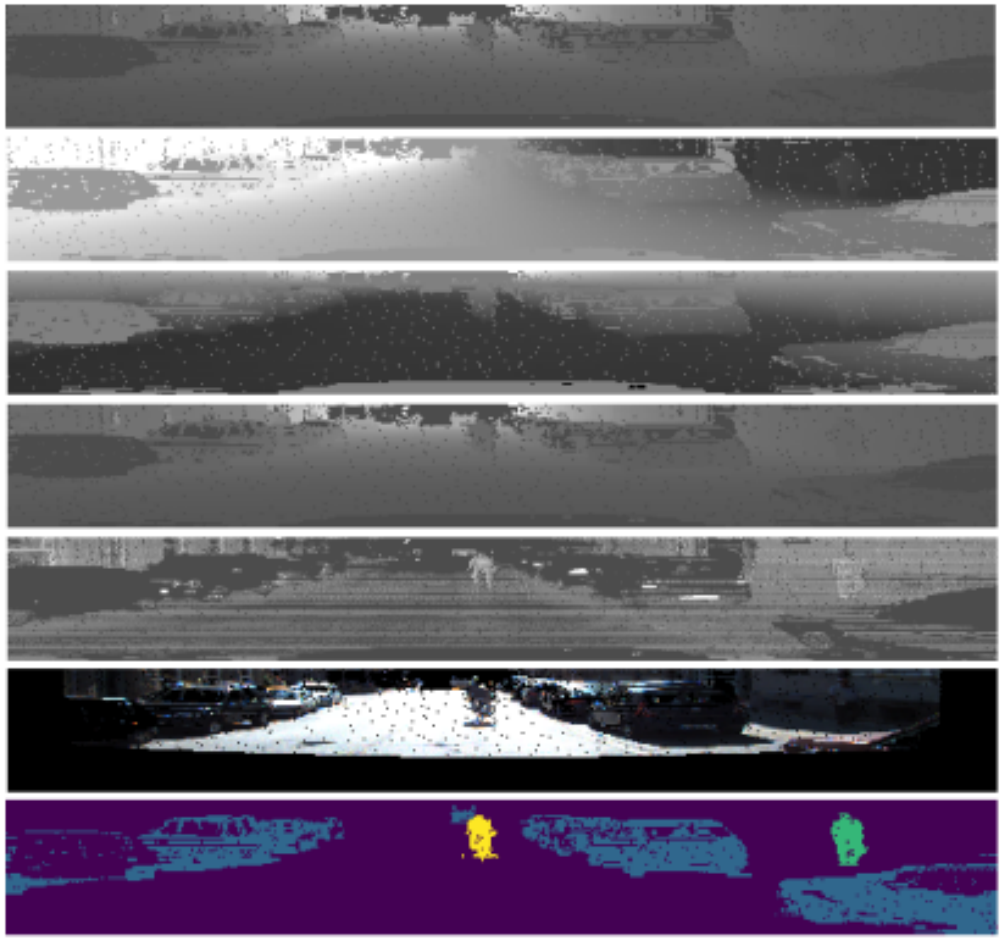


Fig. 3: Input frame and ground-truth tensor. **Top to bottom:** X, Y, Z, D, I, RGB and Ground Truth.

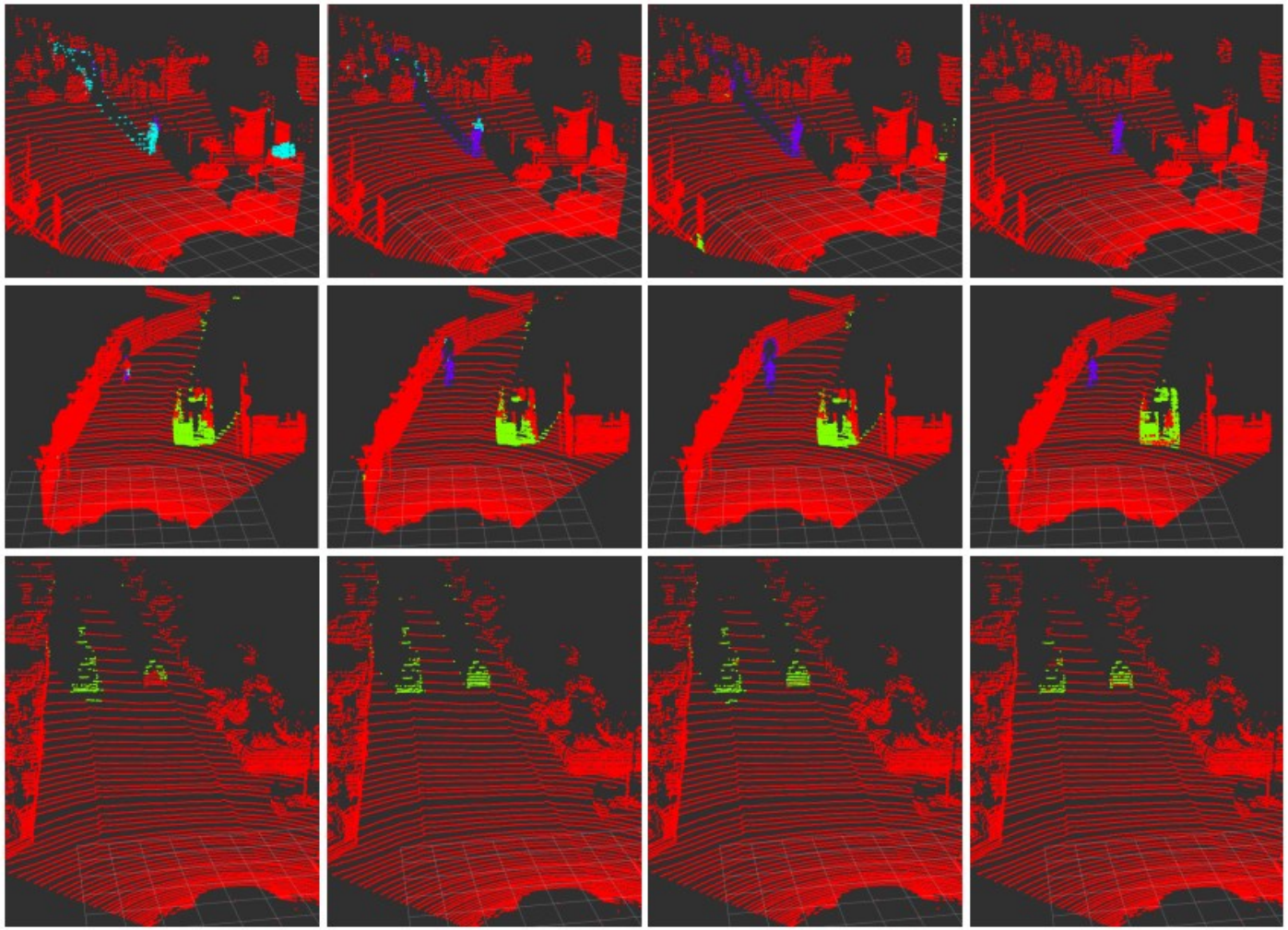


Fig. 5: Qualitative comparison of 3D semantic segmentation outputs using our approach on SqueezeSeg architecture where red color represents None, green color represents Cars, violet represents Cyclists, and light-blue color represents Pedestrians. **Left:** No-Fusion output. **Middle-Left:** Early-Fusion output. **Middle-Right:** Mid-Fusion output. **Right:** Ground Truth.

Recap: MEI and MHI

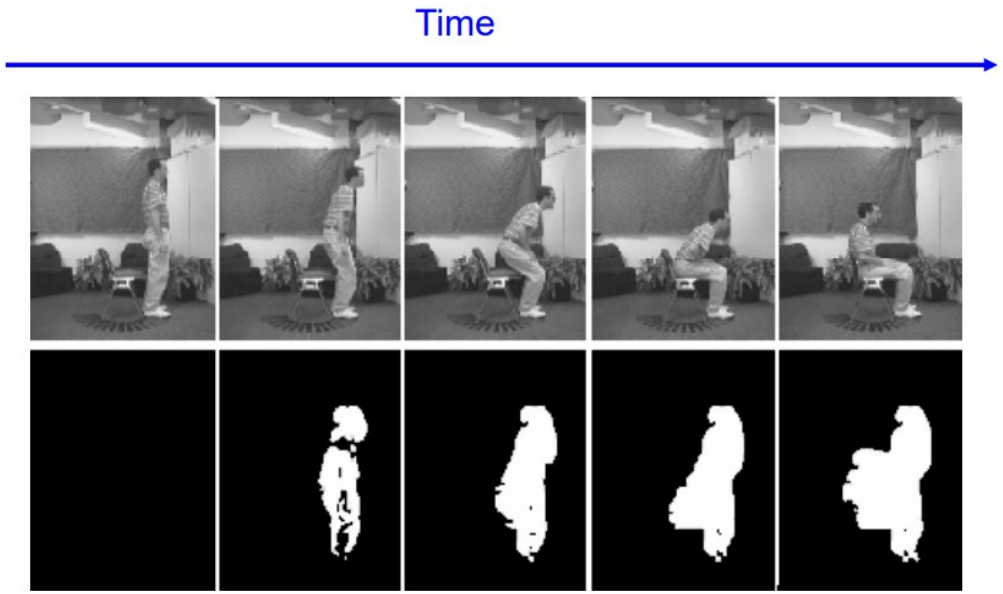


Figure credit: Bobick, A. F., & Davis, J. W. (2001). The recognition of human movement using temporal templates. IEEE Transactions on pattern analysis and machine intelligence, 23(3), 257-267.

- MEI: represents “where” the motion occurs.
- The frame difference between frame at time t and its previous adjacent frame as:

$$D(x, y, t) = \mathbb{1}_{Th}(I(x, y, t) - I(x, y, t-1))$$

- $D(x, y, t)$ is the binary image indicating regions of motion at t.
- The **binary** MEI corresponding to $I(x, y, t)$ is defined as:

$$E_\tau(x, y, t) = \bigcup_{i=0}^{\tau-1} D(x, y, t-i)$$

- τ: Duration that defines the temporal extent of the motion.
- MHI: encode “how” the motion occurs. The pixel intensity is a function of the temporal history of motion at that point:

$$H_\tau(x, y, t) = \begin{cases} \tau & if\ D(x, y, t) = 1 \\ \max(0, H_\tau(x, y, t-1) - \delta) & otherwise \end{cases}$$

- $D(x, y, t) = 1$ indicates the motion is still ongoing. δ is the decay parameter (usually default to 1)

Recap : MEI and MHI

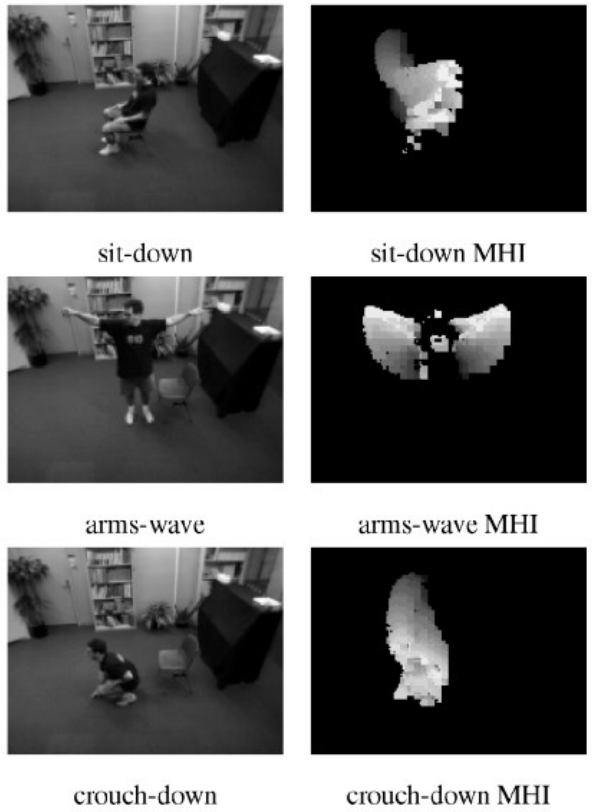


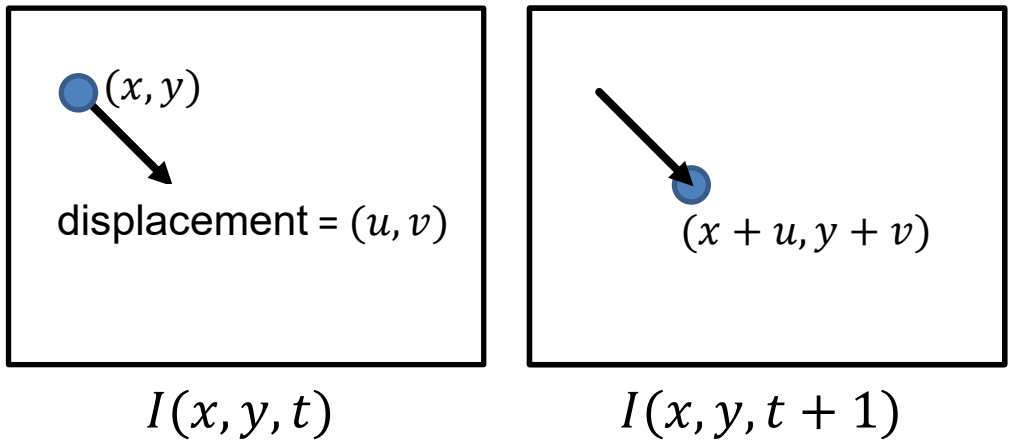
Fig. 4. Simple movements along with their MHIs used in a real-time system.

Figure credit: Bobick, A. F., & Davis, J. W. (2001). The recognition of human movement using temporal templates. IEEE Transactions on pattern analysis and machine intelligence, 23(3), 257-267.

- MEI can be generated by thresholding MHI above zero.
- MEI + MHI → Temporal template.
- Disadvantage:
  - Limit to static background.
  - Limit to simple and short motion. (Videos in datasets at this point tend to only last tens of seconds)
  - Cannot cope with fine-grained actions.
  - Not view-invariant (motion overwriting problem).
  - Sensitive to parameter selection such as the variance of motion duration and decay parameter.

Recap: Optical Flow

- Key assumptions
  - **Brightness constancy**: projection of the same point looks the same in every frame.
  - **Small motion**: points do not move very far.
  - **Spatial coherence**: points move like their neighbors.



- Brightness constancy:

$$I(x, y, t) = I(x + u, y + v, t + 1) \quad \text{Eq. B}$$

- It is impossible to find corresponding pixels in a brute force manner.
- Linearize the right using Taylor expansion with the first-order Taylor series approximation (I_x denotes gradient of I over x)

$$I(x, y, t) \approx I(x, y, t + 1) + I_x u + I_y v$$

$$\boxed{I(x, y, t + 1) - I(x, y, t)} + I_x u + I_y v = 0$$

Essentially I_t

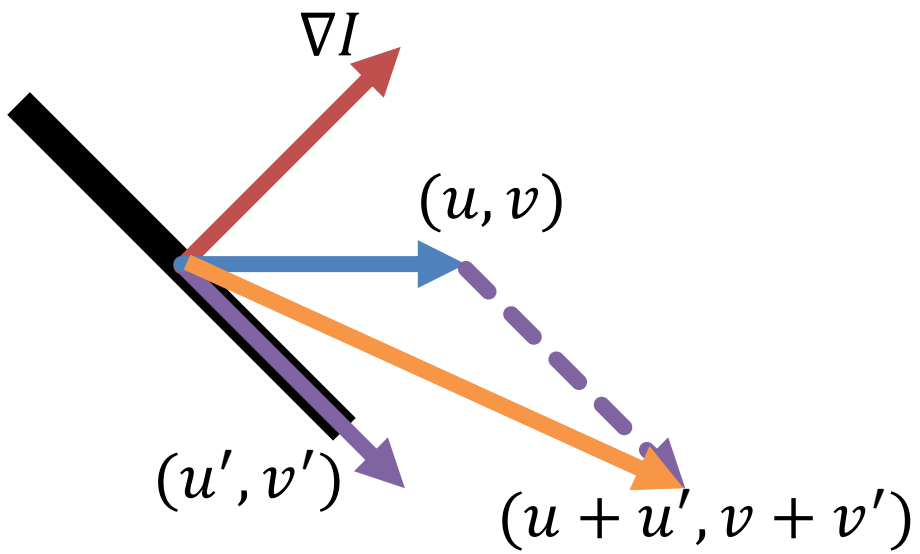
$$I_x u + I_y v + I_t = 0$$

or equivalently $\nabla I \cdot (u, v) + I_t = 0$

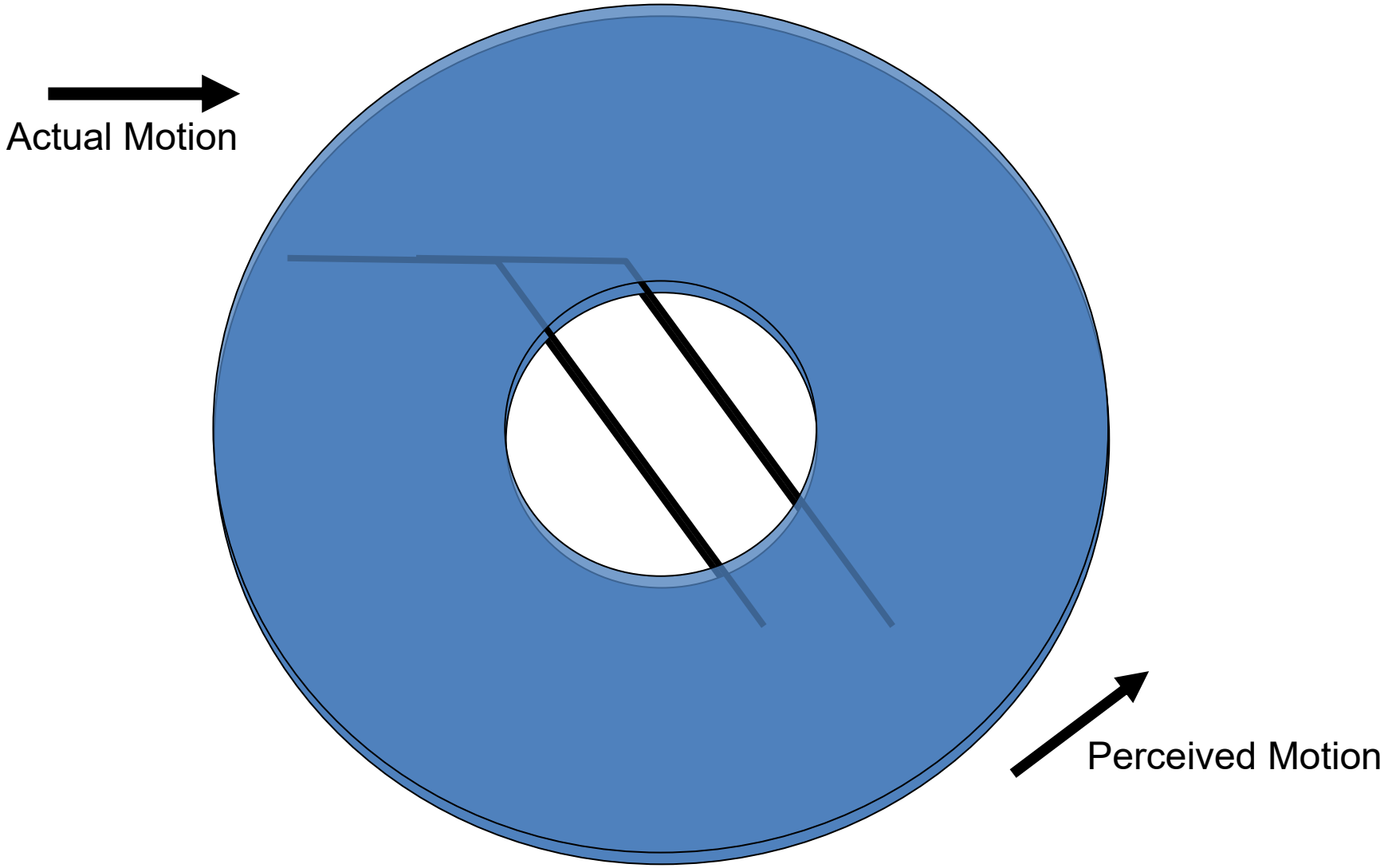
Taylor expansion: $f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \ldots$

Recap: Optical Flow – Brightness Constancy

- What does the equation $I_x u + I_y v + I_t = \nabla I \cdot (u, v) + I_t = 0$ indicates?
- A single equation but two unknowns – there are trivial solutions $\nabla I \cdot (u, v) = 0$.
- Brightness constancy constraint can only identify the motion along the gradient but **not** the motion perpendicular to the gradient.
- If (u, v) satisfies brightness constancy constraint, so does $(u + u', v + v')$ if $\nabla I \cdot (u', v') = 0$.



Recap: Optical Flow – Aperture Problem



Recap: Lucas-Kanade Optical Flow

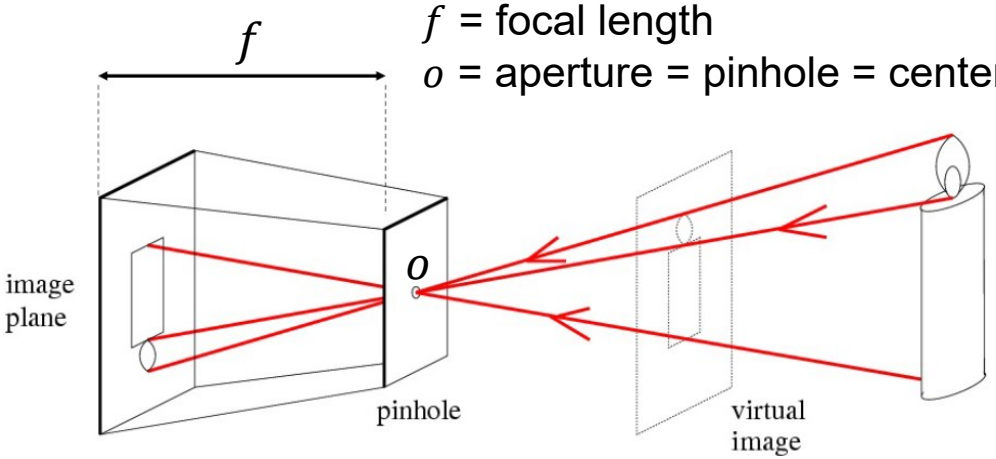
- How to solve the aperture problem?
  - Prevent trivial solutions by getting more equations for a set of unknowns.
- Spatial coherence constraint: assume the pixel's neighbors have the same (u, v) – **Lucas-Kanade** [1] optical flow.
  - E.g., 5×5 window gives 25 new equations.

$$I_t + I_x u + I_y v = 0$$

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

$$p_1 = (x_1, y_1)$$

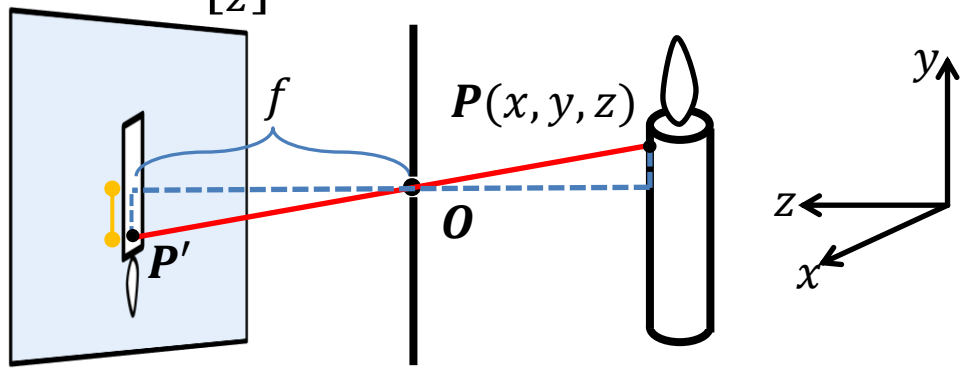
Recap: Camera Fundamentals: Projection with Pinhole and Homogeneous Coordinates



f = focal length

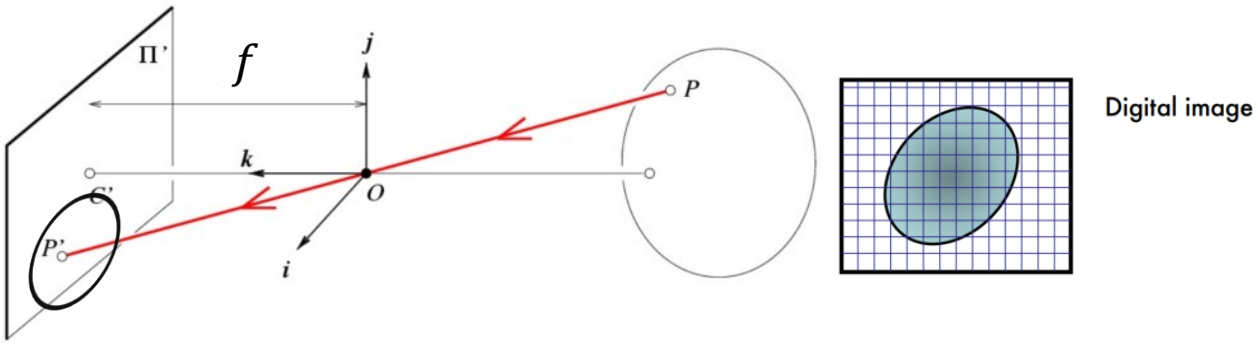
o = aperture = pinhole = center of the camera

$$\boldsymbol{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \boldsymbol{P}' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} fx/z \\ fy/z \end{bmatrix}$$

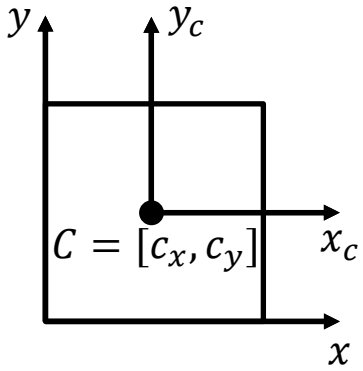


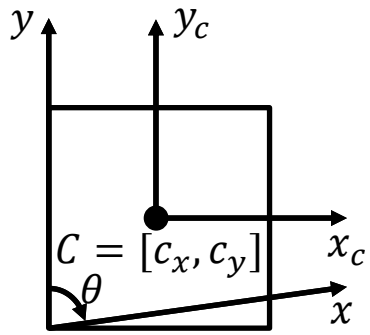
- How to mitigate the risk of zero-division in 3D→2D transformation? Homogeneous Coordinates – add a dimension: Physical → Homogeneous point → Physical Point: (Euclidean → Homogeneous → Euclidean) $\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} u \\ v \\ w \end{bmatrix} \rightarrow \begin{bmatrix} u/w \\ v/w \end{bmatrix}$: usually we assume $w = 1$.
- Why homogeneous coordinates? Lines (3D) and points (2D → 3D) are now the same dimension.
- Use the cross ($\times$) and dot product for: a) Intersection of lines l and m: $l \times m$; b) Line through two points p and q: $p \times q$; c) Point p on line l: $l^T p$.

Recap: Camera Models: Projection Matrix (Intrinsic)



Digital image

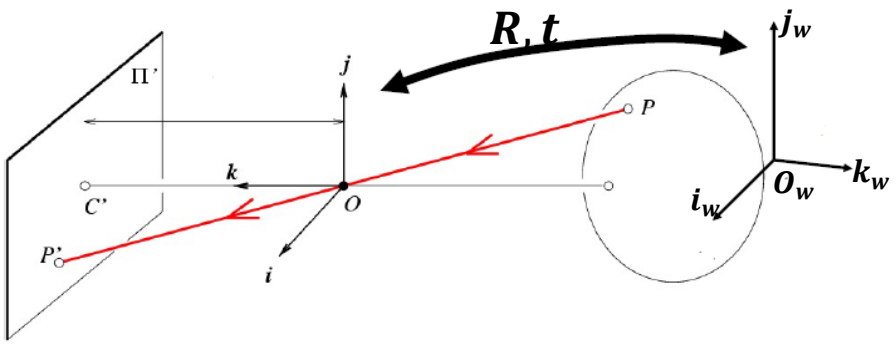


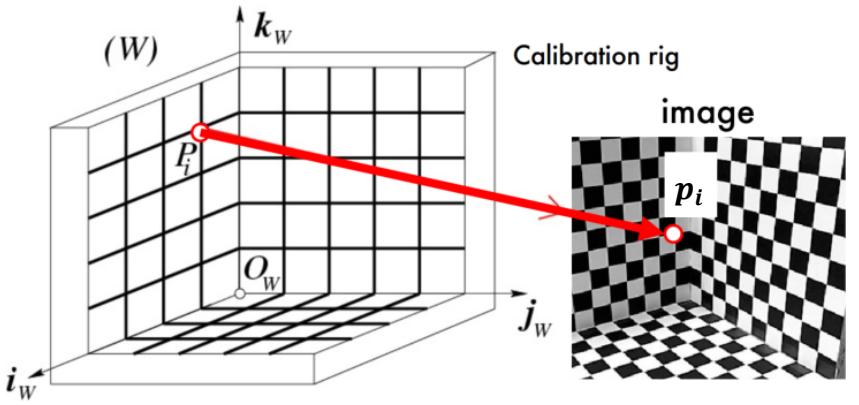


- Factor 1: offset of image plane center in the image plane coordinate. $\boldsymbol{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \boldsymbol{P'} = \begin{bmatrix} f\,x/z + c_x \\ f\,y/z + c_y \end{bmatrix}$
- Factor 2: changing from metric to pixels (f – meters, computation in image planes are pixels (non-square)). $\boldsymbol{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \boldsymbol{P'} = \begin{bmatrix} fk\frac{x}{z} + c_x \\ fl\frac{y}{z} + c_y \end{bmatrix} = \begin{bmatrix} \alpha\frac{x}{z} + c_x \\ \beta\frac{y}{z} + c_y \end{bmatrix} \rightarrow \begin{bmatrix} \alpha x + c_x z \\ \beta y + c_y z \\ z \end{bmatrix}$, $\boldsymbol{P_h}' = \begin{bmatrix} \alpha x + c_x z \\ \beta y + c_y z \\ z \end{bmatrix} = \boldsymbol{M P_h} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \boldsymbol{K}[\boldsymbol{I} \quad 0]\boldsymbol{P}$
- Factor 3: camera skewness – not perfect aligned pixels: $\boldsymbol{P'} = \boldsymbol{MP} = \begin{bmatrix} \alpha & -\alpha\cot\theta & c_x & 0 \\ 0 & \beta/\sin\theta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

Figure from CS231A Stanford University, Credit: Prof. Silvio Savarese

Recap: Camera Models: Projection Matrix (Extrinsic)



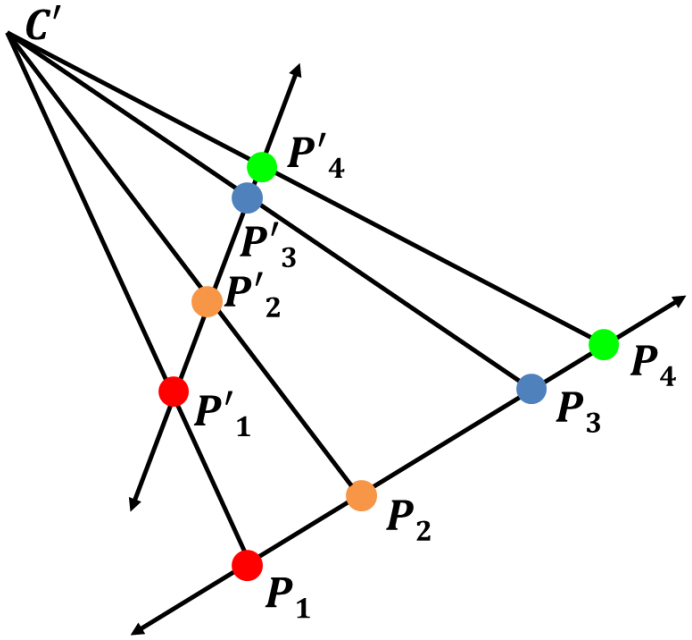


- 3D Translation $\boldsymbol{P'} \rightarrow$ ($\rightarrow$: corresponds to) $\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$.
- 3D Rotation $\boldsymbol{R} = \boldsymbol{R}_x(\alpha)\boldsymbol{R}_y(\beta)\boldsymbol{R}_z(\gamma)$, $\boldsymbol{P'} \rightarrow \begin{bmatrix} \boldsymbol{R} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$.
- Combining the translation matrix and rotation matrix:
- $\boldsymbol{P'} \rightarrow \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$.

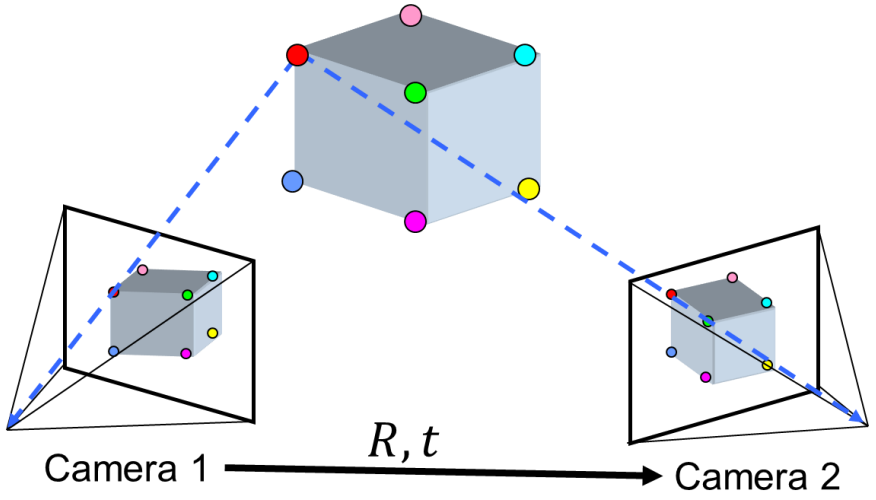
- $\boldsymbol{P'} = \boldsymbol{K}[\boldsymbol{I} \quad 0]\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{I} \quad 0] \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \boldsymbol{P_w} =$
  $\boldsymbol{K}[\boldsymbol{R} \quad \boldsymbol{t}]\boldsymbol{P_w} = \boldsymbol{K}[\boldsymbol{R}, \boldsymbol{t}]\boldsymbol{P_w} = \boldsymbol{M}\boldsymbol{P_w} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ v & b \end{bmatrix} \boldsymbol{P_w}$.
- Application: camera calibration with $2n > 11$ points using the camera model.

Recap: Single-view Geometry: Projective Invariant – Cross Ratio

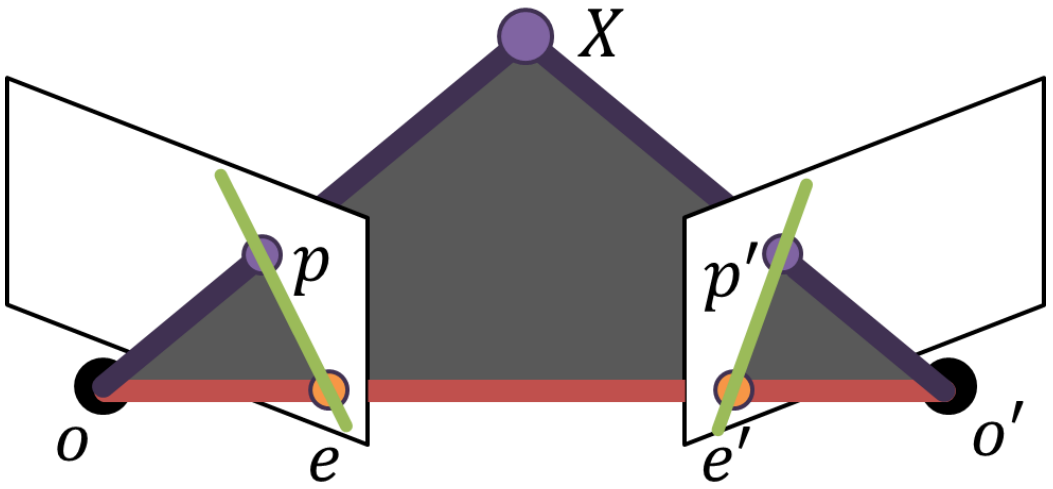
- *Projective invariant*: a quantity that does not change under projective transformations.
- ***Theorem**: we denote the cross ratio of the four points in space as* $[P_1, P_2, P_3, P_4]$, *while the correspond to a set of projected points on the image space* P'_1, P'_2, P'_3, P'_4. *One of its cross ratio is defined as* $[P_1, P_2, P_3, P_4] = \frac{||P_1P_3||||P_2P_4||}{||P_2P_3||||P_1P_4||} = \frac{||P_1P_3||}{||P_2P_3||} / \frac{||P_1P_4||}{||P_2P_4||}$. *This cross ratio is projective invariant, that is:* $[P_1, P_2, P_3, P_4] = [P'_1, P'_2, P'_3, P'_4]$.

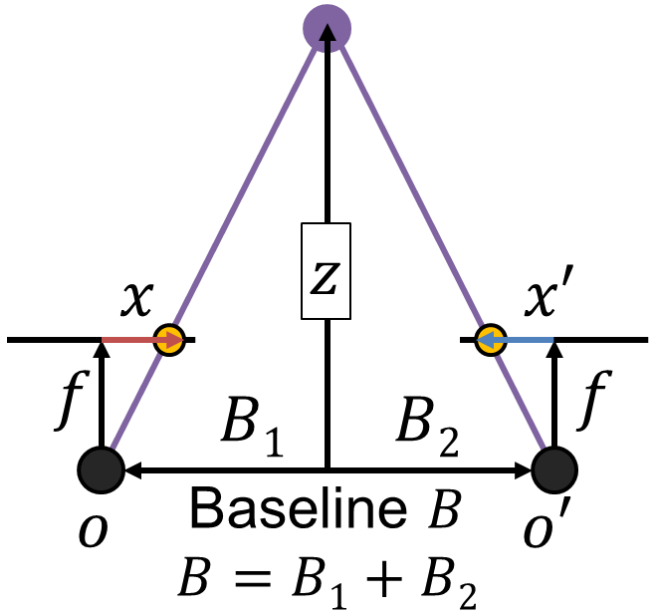


Recap: Resolving Single-view Ambiguity: Two-view Geometry

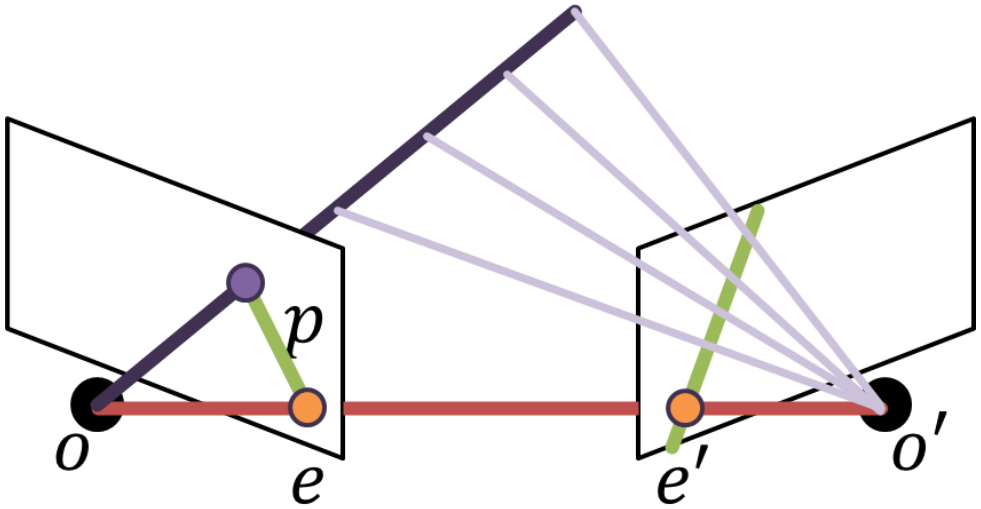


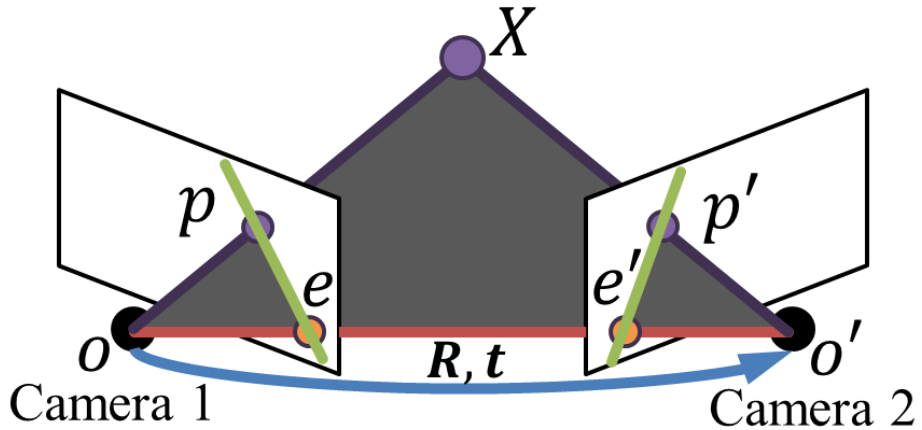
- Single-view ambiguity:
  - Only know how a 3D point in world space corresponds to each pixel, process cannot reverse.
- Two-view geometry:
  - Stereo/Epipolar geomery: given 2 cameras with the extrinsic matrix known, find where a point could be in 3D space – the key: finding corresponding points in the two views.





Recap: Epipolar Geometry: Epipolar Constraint



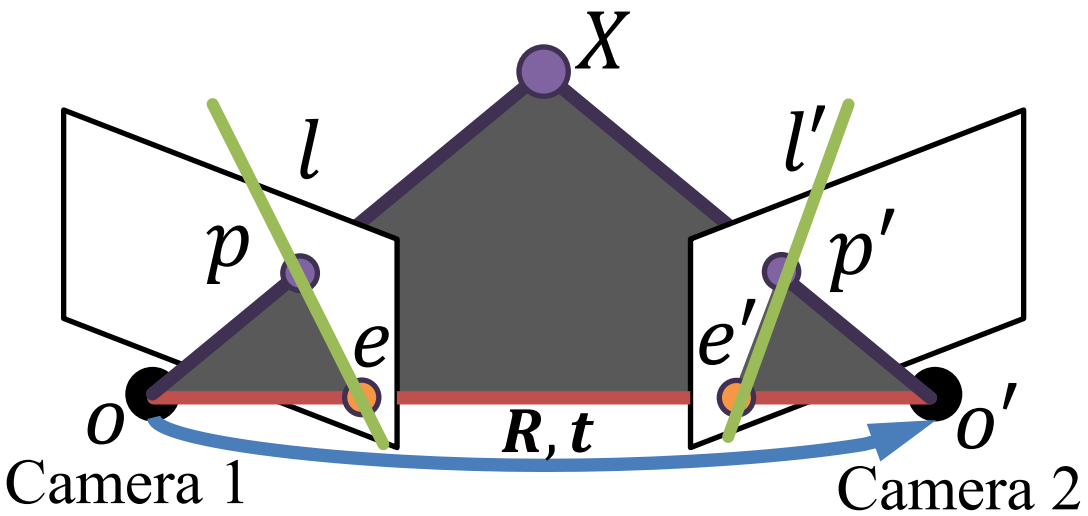


- Why analyze with epipolar geometry?
  - Naïve search: for each pixel, search every other pixel on the second view.
  - Search with epipolar geometry: for each pixel, seach ONLY along the epipolar line.

- $\hat{p}^T[t_x]\boldsymbol{R}\widehat{p'} = 0$, the essential matrix: $\boldsymbol{E} = [t_x]\boldsymbol{R}$, we have the following constraints:
- $\hat{p}^T\boldsymbol{E}\widehat{p'} = 0$; $l = \boldsymbol{E}\widehat{p'}$, $l' = \boldsymbol{E}^T\hat{p}$; $\boldsymbol{E}^T\hat{e} = 0$, $\boldsymbol{E}\widehat{e'} = 0$.
- $p^T\boldsymbol{K}^{-T}\boldsymbol{E}\boldsymbol{K'}^{-1}p' = 0$, the fundamental matrix: $\boldsymbol{F} = \boldsymbol{K}^{-T}\boldsymbol{E}\boldsymbol{K'}^{-1}$, we have the following constraints:
- $p^T\boldsymbol{F}p' = 0$; $l = \boldsymbol{F}p'$, $l' = \boldsymbol{F}^T p$; $\boldsymbol{F}^T e = 0$, $\boldsymbol{F}e' = 0$.

Epipolar Constraint: Notes on Epipolar Line

- We have $\hat{p}^T \boldsymbol{E} \widehat{p'} = 0$.
- Meanwhile, since $\hat{p}$ is on the epipolar line l, we therefore also have $l^T \hat{p} = 0 = \hat{p}^T l$.
- Thus the following equation stands:
  - $l = \boldsymbol{E} \widehat{p'}$.
- Vice versa for epipolar line l' and $\widehat{p'}$ where we have $l'^T \widehat{p'} = 0$, and therefore $l'^T = \hat{p}^T \boldsymbol{E}$, which leads to $l' = \boldsymbol{E}^T \hat{p}$.



Thank you!