Deep 3D Geometry Computer Vision - Selected Recent Research

Richard Xu

January 14, 2021

Content

Selected topics on cutting-edge research using **Deep Learning** for solving 3D computer vision problems since 2015:

- Single image $\rightarrow$ Camera Model estimation
- Multi-Person 3D pose estimation using multi-view
- GAN-based 3D pose estimation
- Deep Learning based Structure from Motion
- Deep Learning based Depth Estimation

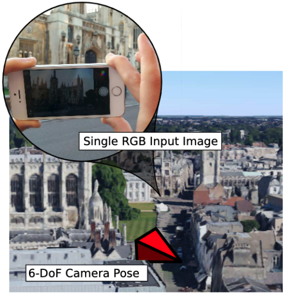
Take home message: Deep Learning is seemingly able to learn how to transform every forms of data to a supervised target

- but of course it's not true - in 3D, let's get the Geometry and CNN to work together!
- more on Deep Learning and Machine Learning can be found on

`http://richardxu.com`

PoseNet [1]

- **PoseNet** is a deep learning framework
- regress: "single image" → "6-DoF camera pose" using a convolutional neural network



[1] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DoF camera relocalization. In: ICCV2015

Regress $I \rightarrow [\mathbf{t}, \mathbf{q}]$

- **Input**: image I:
- **Output**: a pose vector $[\mathbf{t}, \mathbf{q}]$ note we did not use Euler angle $\mathbf{r}$ here
  1. translation $\mathbf{t}$
  2. rotation $\mathbf{q}$ (but in quaternion form) 4DoF
- in interpolation: Cartesian coordinates are independent of one another, but Euler angles are not, and Gimbal Lock problem
- Quaternion solves it

Intuitively, loss function is:

$$\mathcal{L}(\hat{\mathbf{t}}, \hat{\mathbf{q}}) = \|\hat{\mathbf{t}} - \mathbf{t}\|_2 + \beta \left\| \hat{\mathbf{q}} - \frac{\mathbf{q}}{\|\mathbf{q}\|} \right\|_2$$

- β is the weight between **t** and **q**

Drawback of PoseNet

$$\mathcal{L}(\hat{\mathbf{t}}, \hat{\mathbf{q}}) = \|\hat{\mathbf{t}} - \mathbf{t}\|_2 + \beta \left\| \hat{\mathbf{q}} - \frac{\mathbf{q}}{\|\mathbf{q}\|} \right\|_2$$

- Experiments show jointly regress **t** and **q** is better than training them separately
- hyperparameter β required significant tuning to get reasonable results. (two different scale)
- measuring error between parameters are strange, measuring re-projection errors are more natural.

Geometric Reprojection Error [2]

- instead of measure loss in $[\mathbf{t}, \mathbf{q}]$ space which is unnatural and difficult to determine the relative weights
- we measure loss in terms of projection error:

$$\mathcal{L}_g(I) = \frac{1}{|\mathcal{G}'|} \sum_{g_i \in \mathcal{G}'} \left\| \text{Proj}\left(\mathbf{t}, \mathbf{q}, \mathbf{g_i}\right) - \text{Proj}\left(\hat{\mathbf{t}}, \hat{\mathbf{q}}, \mathbf{g_i}\right) \right\|_\gamma$$

- Function Proj(.) just a regular $3D \to 2D$ projection, for some test point $\mathbf{g}$:

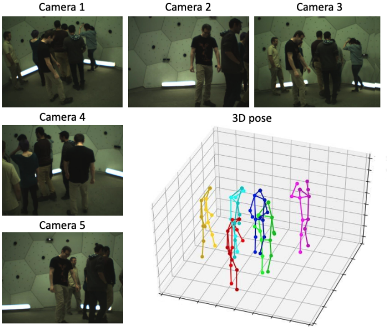
$$\text{Proj}(\mathbf{t}, \mathbf{q}, \mathbf{g}) = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u'/w' \\ v'/w' \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \mathbf{K}(\mathbf{R}\,\mathbf{g} + \mathbf{t})$$

- instead of regress image $\to$ parameter directly
- now it regress image $\to$ image location $\begin{bmatrix} u & v \end{bmatrix}^\top$ (as the result of the parameter)

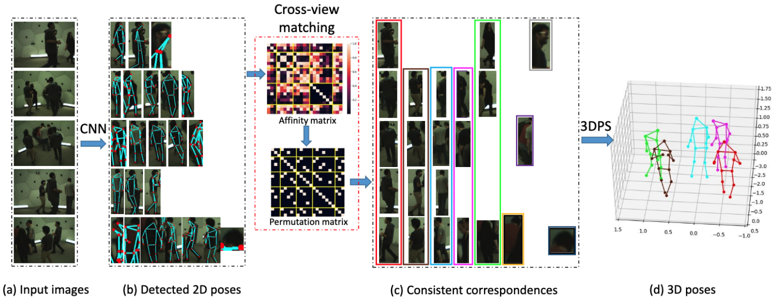
[2] Kendall, Alex and Cipolla, and Roberto. Geometric loss functions for camera pose regression with deep learning. In: CVPR2017

Multi-person 3D pose estimation from multiple views

- Reconstruct 3D human pose for multiple persons in a few calibrated camera views
- **Challenge**: Find cross-view correspondences among noisy and incomplete 2D pose predictions



Overview of the approach. [3]



[3] Dong J, Jiang W, Huang Q, et al. Fast and robust multi-person 3d pose estimation from multiple views. In: CVPR2019

Affinity Matrix

Use Cascaded Pyramid Network (CPN) trained on MSCOCO dataset for 2D pose detection.

- V cameras view
- affinity matrix $\boldsymbol{A}_{ij} \in \mathbb{R}^{p_i \times p_j}$: whose elements represent affinity scores between two sets of bounding boxes in view i and view j of all its bounding boxes
- how to calculate affinity matrix between two bounding boxes:
  - Appearance similarity: pre-trained person re-ID network.
  - Geometric compatibility: satisfy the epipolar constraint.

$$D_g\left(\mathbf{x}_i, \mathbf{x}_j\right) = \frac{1}{2N} \sum_{n=1}^{N} d_g\left(\mathbf{x}_i^n, \boldsymbol{L}_{ij}\left(\mathbf{x}_j^n\right)\right) + d_g\left(\mathbf{x}_j^n, \boldsymbol{L}_{ji}\left(\mathbf{x}_i^n\right)\right)$$

  - $\boldsymbol{L}_{ij}\left(\mathbf{x}_j^n\right)$ returns the epipolar line associated with x_j^n from the other view using Fundamental matrix: $\boldsymbol{L}_{ij}\left(\mathbf{x}_j^n\right) \equiv \mathbf{F}_{ij}\mathbf{x}_j^n$
  - $d_g(\cdot, \boldsymbol{l})$ is the point-to-line distance for $\boldsymbol{l}$.

Multi-view Correspondences

- $\boldsymbol{P}_{i \to j} \in \{0,1\}^{n_i \times n_j}$, i.e., a matrix, but no need to be square
- $\boldsymbol{P}_{i \to j}$ is a partial permutation matrix, means each of the n_i persons in view i, whom is the most similar person $j \in n_j$ in view j:
- suppose we have two views and each having 2 views and 2 persons in total:

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{1\to 1} & \mathbf{P}_{1\to 2} \\ \mathbf{P}_{2\to 1} & \mathbf{P}_{2\to 2} \end{pmatrix}$$

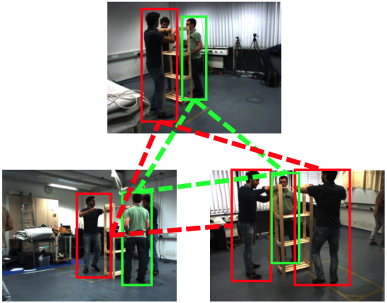
$$\underbrace{\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}}_{\text{consistent: rank}(\mathbf{P})=2} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}}_{\begin{pmatrix} \mathbf{P}_{1\to 1} \\ \mathbf{P}_{2\to 1} \end{pmatrix}} \underbrace{\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}}_{\begin{pmatrix} \mathbf{P}_{1\to 1} \\ \mathbf{P}_{2\to 1} \end{pmatrix}^\top = (\mathbf{P}_{1\to 1} \quad \mathbf{P}_{1\to 2})}$$

$$\underbrace{\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}}_{\text{in-consistent: rank}(\mathbf{P})=4}$$

- to have consistency, $\mathbf{P}_{i \to j} = \mathbf{P}_{j \to i}^\top$, so it must be symmetric and low rank
- However, we can't enforce consistency everywhere, so we encourage "lower" rank

Correspondence Matrix

- Under multiple views, just matching separately for each pair of views, and ignores the **cycle-consistency constraint** can lead to inconsistent results



- **solution**:

$$\text{rank}(\boldsymbol{P}) \leq s, \boldsymbol{P} \succeq 0$$

  where s is the underlying number of people in the scene
- Since s is unknown in advance, we should minimize $\text{rank}(\boldsymbol{P})$

Overall Objective

- real objective function is:

$$\begin{aligned} f(\boldsymbol{P}) &= -\sum_{i=1}^{n}\sum_{j=1}^{n}\langle \boldsymbol{A}_{ij}, \boldsymbol{P}_{ij}\rangle + \lambda \cdot \mathrm{rank}(\boldsymbol{P}) \\ &= -\langle \boldsymbol{A}, \boldsymbol{P}\rangle + \lambda \cdot \mathrm{rank}(\boldsymbol{P}) \end{aligned}$$

- approximated by:

$$\min_{\mathbf{p}} \left(-\langle \boldsymbol{A}, \mathbf{P}\rangle + \lambda \underbrace{\|\mathbf{P}\|_*}_{\text{nuclear norm}} \right)$$

Schatten p-norms and Nuclear Norm (1)

- Schatten p-norms, for matrix of size $m \times n$, this kind of norm "operates on vector of singular values":

$$\|A\|_p = \|\sigma_1(A), \dots, \sigma_{\min(m,n)}(A)\|_p \qquad \text{note, there is no } \max_{\|x\|=1} \text{ as the other matrix norm}$$

$$= \left(\sum_{i=1}^{\min(m,n)} \sigma_i^p(A) \right)^{\frac{1}{p}} \qquad \text{just usual definition of vector norm}$$

  singular values are non-negative, even though eigenvalues may be negative, so we can drop $|\cdot|$

- when $p = 1$:

$$\|A\|_1 = \|\sigma_1(A), \dots, \sigma_{\min(m,n)}(A)\|_1$$
$$= \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i(A) \right)$$

- we give a special name $\|A\|_* = \|A\|_1$, nuclear norm

looking at Nuclear Norm (2)

- let $K = \min(m, n)$
- just like L_1 norm:

$$\|A\|_1 = \|\sigma_1(A), \dots, \sigma_K(A)\|_1 = \left(\sum_{i=1}^{K} \sigma_i(A) \right)$$

- when minimizing $\|\sigma_1(A), \dots, \sigma_K(A)\|_1$, you obtain a sparse solution
- since SVD of A is written as:

$$\sigma_1(A) U_1 V_1^\top + \cdots + \sigma_K(A) U_K V_K^\top$$

a sparse solution delivers a low-rank matrix A

lastly: 3D Pose Generation (pictorial structure model)

$$\begin{cases} \boldsymbol{T} \equiv (T_1, \dots T_N) & \text{a person joint locations} \\ \mathbf{I} & \text{all the heatmaps} \end{cases}$$

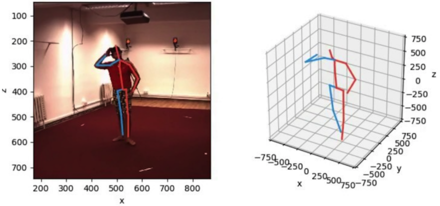
- objective function:

$$p(\boldsymbol{T}|I) \propto \underbrace{\prod_{v=1}^{V} \prod_{i=1}^{N} p\left(I_v | \pi_v\left(T_i\right)\right)}_{p(I_1,\dots I_V|\boldsymbol{T})} \underbrace{\prod_{(i,j) \in N} p\left(T_i, T_j\right)}_{p(\boldsymbol{T})}$$

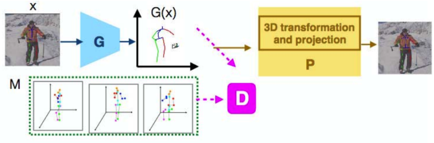
- $\pi_v\left(T_i\right)$ 2D projection of T_i on v-th view
- $p\left(I_v | \pi_v\left(T_i\right)\right)$ is likelihood of heatmap I_v
- $p\left(T_i, T_j\right)$ structural dependency between joint T_i and T_j

New topic: Monocular 3D Human Pose Estimation

- Estimate human body (pose) configuration in 3D space from single image or video
- focus on the methods combining **geometric priors + neural networks**



- Adversarial Inverse Graphics Network (AIGN) for 3D human pose estimation



- Basically a conditional GAN + reconstruction error
- GAN is genernally not reversible, i.e., $z \to \mathbf{x}$ is not possible
- however here $\mathbf{x}$ is 3D and z is 2D!
- question can you think of other application may work as well? for example, super-resolution
- answer any directional mapping function where GAN is used to learn the inverse mapping

[4] Tung H Y F, Harley A W, Seto W, et al. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In: ICCV2017

3D Pose Representation

- Decompose 3D human shape into a view-aligned 3D linear basis:

$$S_{3D} = \sum_{j=1}^{|\mathcal{B}|} w_j B_j$$

- model and a rotation matrix

$$\mathbf{x}_{3D} = \mathbf{R} \; S_{3D} \qquad \text{where} \qquad \mathbf{R} = \mathbf{R}^x(\alpha)\mathbf{R}^y(\beta)\mathbf{R}_t^z(\gamma)$$

- Shape basis $\{B_j\}$ is obtained using PCA on orientation-aligned 3D poses in priory
- in here, Generator generates weights $\mathbf{w} \equiv \{w_1, \dots w_{60}\}$ to get $\rightarrow \mathbf{x}_{3D}$, instead of to generate $\mathbf{x}_{3D}$ directly

Loss Function

- "re-projected" 2D points: $\tilde{\mathbf{x}}_{2D}^{\text{proj}}$ depends on:
  1. $\mathbf{x}_{3D}$ via "weights" $\mathbf{w} \in \mathbb{R}^{60}$
  2. focal length f
  3. Euler rotation angles α, β, γ
  4. translation $[t_x, t_y]$

$$\tilde{\mathbf{x}}_{2D}^{\text{proj}} = \mathbf{K}_f \left(\mathbf{R}\ \mathbf{x}_{3D} + \begin{bmatrix} t_x \\ t_y \\ 0 \end{bmatrix} \right)$$

Loss Function

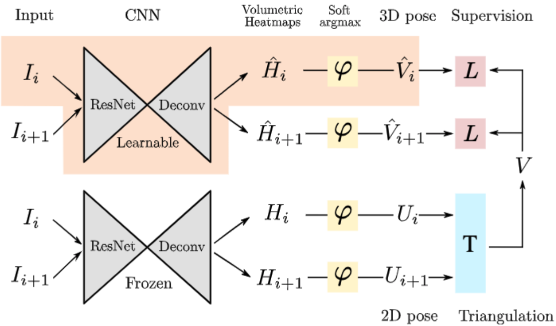
given $\mathbf{M} \in \Omega_{\mathbf{M}}$ training examples:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{\mathbf{x} \in \Omega_{\mathbf{x}}, \mathbf{M} \in \Omega_{\mathbf{M}}} \Bigg[\underbrace{\|\hat{\mathbf{x}}_{\text{2D}}^{\text{proj}} - \mathbf{x}\|_2}_{\text{reconstruction loss}} + \beta \underbrace{\log \mathcal{D}(\mathbf{M}) + \log \Big(1 - \mathcal{D}\big(\overbrace{\mathcal{G}(\mathbf{x})}^{\mathbf{x}_{\text{3D}}}\big)\Big)}_{\text{adversarial loss}} \Bigg]$$

- $\hat{\mathbf{x}}_{\text{2D}}^{\text{proj}}$: “re-projected ” 2D
- $\mathbf{x}$: “argmax of predicted 2D heatmaps”
- Discriminator network discriminates between generated keypoints $\mathbf{x}_{\text{3D}}$ and a database of 3D human skeletons $\mathbf{M}$

EpipolarPose Network [5]

- EpipolarPose network:



[5] Kocabas M, Karagoz S, Akbas E. Self-supervised learning of 3d human pose using multi-view geometry. In: CVPR2019

Overall Architecture

training

- a pair of images(I_i,I_{i+1}) from two cameras is fed into the CNN pose estimators
- 3D pose (V) generated by the lower branch using triangulation is used as a training signal for CNN in upper branch

inference

- EpipolarPose is a monocular method: it takes a single image (I_i) as input and estimates the corresponding 3D pose.

Obtain 3D Coordinate through Triangulation

- projection is use traditional stuff:

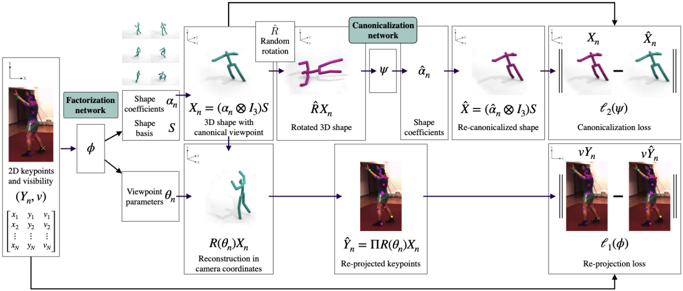
$$\begin{bmatrix} x_{i,j} \\ y_{i,j} \\ w_{i,j} \end{bmatrix} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \begin{bmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{bmatrix}, \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

- When $[\mathbf{R}|\mathbf{t}]$ are available, using triangulation to obtain 3D pose

When extrinsic parameters are **unavailable:**

- Assume first camera is at center of coordinate system: means **R** of first camera is identity matrix
- For corresponding joints in view $(i, i+1)$: U_i and U_{i+1}:
  1. estimate fundamental matrix $\mathbf{F} : U_{i,j}\mathbf{F}U_{i+1,j} = 0$ for $\forall j$ using RANSAC
  2. from **F** (acts like data), calculate Essential matrix $\mathbf{E} = \mathbf{K}^\top \mathbf{F}\mathbf{K}$
  3. by decomposing **E** (acts like data now) with SVD, obtain $\mathbf{E} = \mathbf{R}[\mathbf{t}_\times]$

- Overview architecture of the Canonical 3D Pose Networks (C3DPO)
- aims at extracting 3D models of deform-able objects from 2D keypoint annotations in unconstrained images
- notice the unique mechanism in disentangle “shape” and “rotation”
- canonicalization module is redundant, and is there to ensure robustness, useful other situations where there is “many-to-one” mappings



6

[6] Novotny D, Ravi N, Graham B, et al. C3DPO: Canonical 3D Pose Networks for Non-Rigid Structure From Motion. In: ICCV2019

- **input**

$$Y_n = \left(y_{n,1}, \ldots, y_{n,K}\right) \in \mathbb{R}^{2\times K}$$

representing N views $y_1, \ldots y_N$ of a rigid objects of K 2D keypoints each

- views are generated from a **latent** single set of 3D keypoints

$$\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_K) \in \mathbb{R}^{3\times K}$$

- N rigid motion $(\mathbf{R}_n, \mathbf{t}_n)$.
- standard projection matrix:

$$Y_n = \left[\mathbf{R}_n |\ \mathbf{t}\right] \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \underbrace{\mathbf{R}_n\mathbf{X} + \mathbf{t}_n}_{\mathbf{P}}$$

$$\implies Y_{n,k} = \mathbf{R}_n\mathbf{X}_k + \mathbf{t}_n$$

where $\mathbf{P} : \mathbb{R}^3 \to \mathbb{R}^2$ is the camera projection function.

Structure from Motion

- "for simplicity" assumes $\mathbf{K} = \begin{bmatrix} \mathbf{I}_{2\times 2} & 0 \end{bmatrix}$, i.e., orthographic camera
- yields simplified system of equations

$$y_{nk} = \mathbf{R}_n \mathbf{X}_k \quad \text{translation removed by centering in orthographic projection}$$

- Structure from Motion can be formulated as factoring the views $\mathbf{Y}$ into viewpoints $\mathbf{R}$ and structure $\mathbf{X}$.

$$\mathbf{Y} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,K} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,K} \end{bmatrix}, \mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \vdots \\ \mathbf{R}_N \end{bmatrix}, \underbrace{\mathbf{Y}}_{2N\times K} = \underbrace{\mathbf{R}}_{2N\times 3} \underbrace{\mathbf{X}}_{3\times K}$$

Non-Rigid Structure from Motion

- In Non-Rigid SFM, structure $\mathbf{X}_n$ is allowed to deform from one view to next
- Constrain deformation using simple linear model
- Express $\mathbf{X}_n$ as:

$$\mathbf{X}_n = \mathbf{X}(\alpha_n, S)$$

1. $\alpha_n \in \mathbb{R}^D$: small vector of view-specific pose parameter
2. $S \in \mathbb{R}^{3D \times K}$: view-invariant shape basis

- Non-Rigid Structure from Motion can be formulated as:

$$\underbrace{Y}_{2N \times K} = \underbrace{\bar{\mathbf{M}}}_{2N \times 3N(\text{ sparse })} \underbrace{(\overbrace{\alpha}^{N \times D} \otimes \mathbf{I}_3)}_{3N \times 3D} \underbrace{S}_{3D \times K}$$

Learning a Network for Factorization

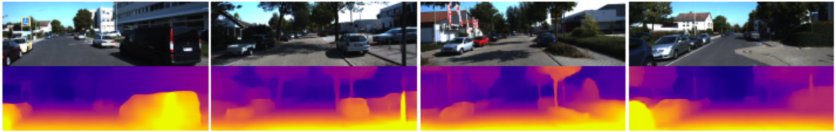
- The network is learned by minimizing the reprojection loss:

$$\ell_1(Y, v; \Phi, S) = \frac{1}{K} \sum_{k=1}^{K} v_k \cdot \left\| Y_k - M(\theta) \left(\alpha \otimes I_3 \right) S_{:,k} \right\|_\epsilon$$

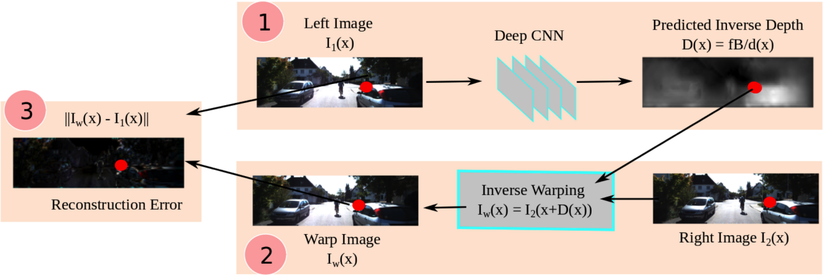
v_k: binary indicating whether k^{th} keypoint is visible in that particular view

Depth Estimation

- Depth estimation: designed to estimate depth from 2D images
- some works used supervised method: drawback: large amount of manually labelled data (image and correpodning depth)



Overall architecture of a stereopsis based auto-encoder. [7]



- **input**
  1. pair of rectified stereo images
  2. known baseline Z between the two
- $d(\mathbf{x})$ is depth and $D(\mathbf{x})$ is disparity
- naive question why not learn $D(\mathbf{x})$?

[7] Garg R, BG V K, Carneiro G, et al. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In: ECCV2016

Overall Architecture

- **encoder** 1 CNN map left image ($\mathbf{I}_1$) to depth map $d(\mathbf{x})$
- then $d(\mathbf{x})$ allows to estimate disparity $D(\mathbf{x}) = \frac{\mathbf{B}f}{Z}$
- **decoder** 2 knowing disparity $D(\mathbf{x})$, one obtains synthesized version of $\mathbf{I}_1$, i.e., $\mathbf{I}_1^2$ by backward warp $\mathbf{I}_1^w(\mathbf{x}) = \mathbf{I}_2(\mathbf{x} + D(\mathbf{x}))$
- reconstructed loss between $\mathbf{I}_1$ and $\mathbf{I}_1^w$ 3 via a simple loss.

It's an intersting auto-encoder

1. **Encoder**: $\mathbf{I}_1 \rightarrow d(\mathbf{x})$, simple! just CNN
2. **Decoder:** $d(\mathbf{x}) \rightarrow \mathbf{I}_1^w$, no neural network but geometry
   however this step needs the help of $\mathbf{I}_2$ and camera parameter $(\mathbf{B}, f, Z)$

- **key point** traditional depth estimation require disparity map (hence second image) beforehand, but this method does not!
- works by generate depth map from one image, using neural networks, but constrianed from second image + geometry

- **Photometric error** basically just the reconnstruction error:

$$\begin{aligned}\mathcal{L}^{i}_{\text{reconstruction}} &= \int_{\Omega} \left\| \mathbf{I}^{i}_{w}(x) - \mathbf{I}^{i}_{1}(x) \right\|^{2} dx \\ &= \int_{\Omega} \left\| \mathbf{I}^{i}_{2}(x + \underbrace{D^{i}(x)}_{fB/d^{i}(x)}) - \mathbf{I}^{i}_{1}(x) \right\|^{2} dx\end{aligned}$$

- **Smooth error** Use the L2 regularization on the disparity discontinuities as our prior to deal with the aperture problem.

$$\mathcal{L}^{i}_{\text{smooth}} = \left\| \nabla D^{i}(x) \right\|^{2}$$

Unsupervised Learning of Depth and Ego-Motion from Video [8]

- Jointly estimate **monocular depth** and **camera motion** from unstructured video sequence.
- End-to-end model mapping input pixels to:
  1. ego-motion (parameterized as 6-DoF transformation matrices)
  2. underlying scene structure (parameterized as per-pixel depth maps under a reference view)

[8] Zhou T, Brown M, Snavely N, et al. Unsupervised learning of depth and ego-motion from video. In: CVPR2017

View Synthesis as Supervision

- overall loss is its reconstruction between:
  - input view of a scene $\mathbf{I}_t$
  - a synthesize scene $\hat{\mathbf{I}}_s$ seen from a different camera pose $\mathbf{I}_{s \neq t}$

$$\mathcal{L}_{vs} = \sum_s \sum_p \left| \mathbf{I}_t(p) - \hat{\mathbf{I}}_s(p) \right|$$

- in order to obtain a $\hat{\mathbf{I}}_s$ seen from a different camera pose $\mathbf{I}_{s \neq t}$, it trains
  1. **monocular depth** generation and
  2. **camera pose** between views

  Together, they learned the relationship of how a corresponding pixel transform from one view to another

How a corresponding pixel transform from one view to another

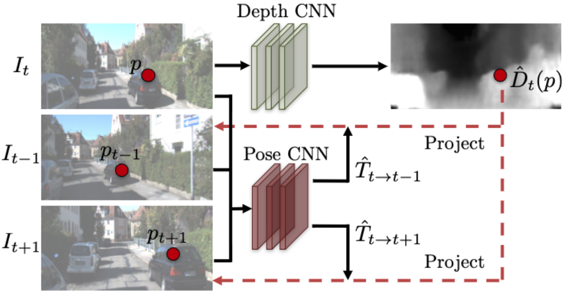
- reconstruct target view $\mathbf{I}_t$ by sampling pixels from a source view $\mathbf{I}_s$ based on the predicted depth map $\hat{D}_t$ and relative pose $\hat{T}_{t \to s}$.

$$p_s \propto \mathbf{K} \hat{T}_{t \to s} \underbrace{\hat{D}_t \left(p_t \right) \mathbf{K}^{-1} p_t}_{\mathbf{X}_{3\mathrm{D}} \text{ in } t^{\text{th}} \text{ view's coordinate}}$$

- learned both $\hat{D}_t$ and $\hat{T}_{t \to s}$ help to obtain the reverse mapping: $s \to t$

Overall Architecture

Overview of the supervision pipeline based on view synthesis.



Overall Architecture

- **Depth CNN** takes only target view as input, outputs per-pixel depth map $\hat{D}_t(p_t)$
- **Pose CNN** takes both the target view (I_t) and the nearby/source views (e.g., $I_{t-1}$and I_{t+1})as input, and outputs the relative camera poses ($\hat{T}_{t\to t-1}$, $\hat{T}_{t\to t+1}$).

inverse warp the source views to reconstruct the target view $\hat{\mathbf{I}}(s)$, and the photometric reconstruction loss is used:

$$\mathcal{L}_{vs} = \sum_s \sum_p \left| \mathbf{I}_t(p) - \hat{\mathbf{I}}_s(p) \right|$$