Deep Learning: Image Registration

Steven Chen and Ty Nguyen

Lecture Outline

1. Brief Introduction to Deep Learning

2. Case Study 1: Unsupervised Deep Homography

3. Case Study 2: Deep Lucas-Kanade

What is Deep Learning?

Machine Learning with a “deep” neural network

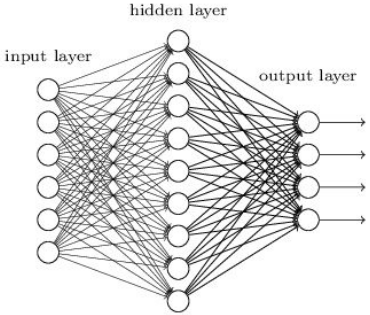
- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- … and more variants

Define:

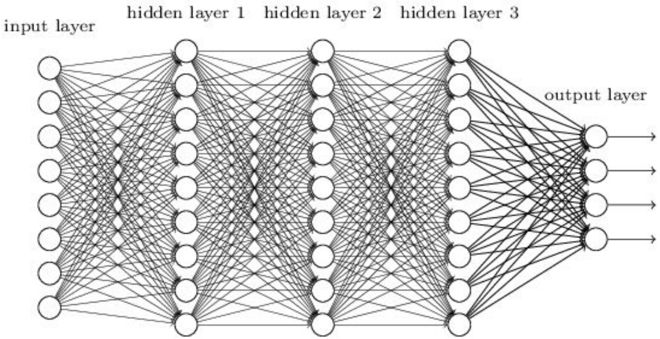
1. Inputs
2. Architecture
3. Output
4. Loss Function

Optimize by performing gradient descent on network parameters

"Non-deep" feedforward neural network

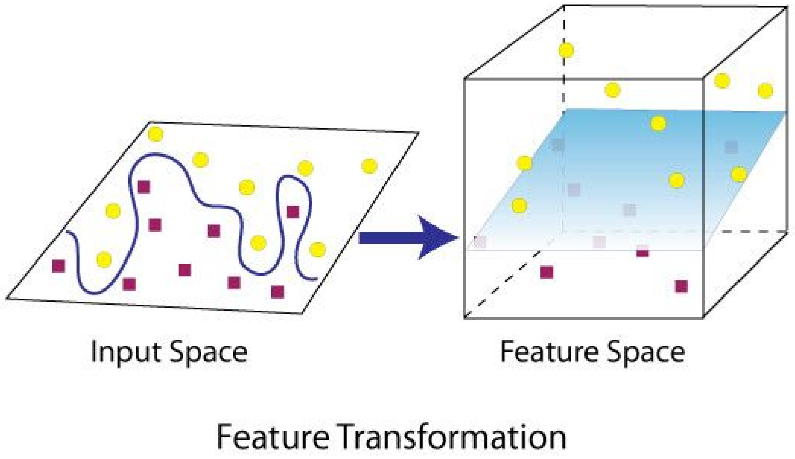


Deep neural network

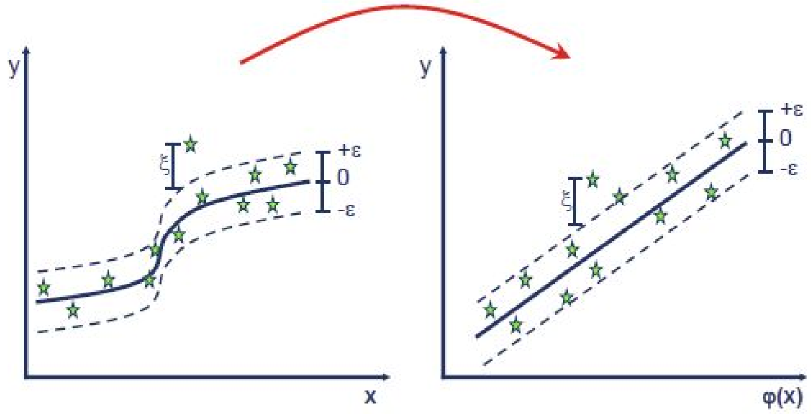


Deep Learning = Learning Features

Classification

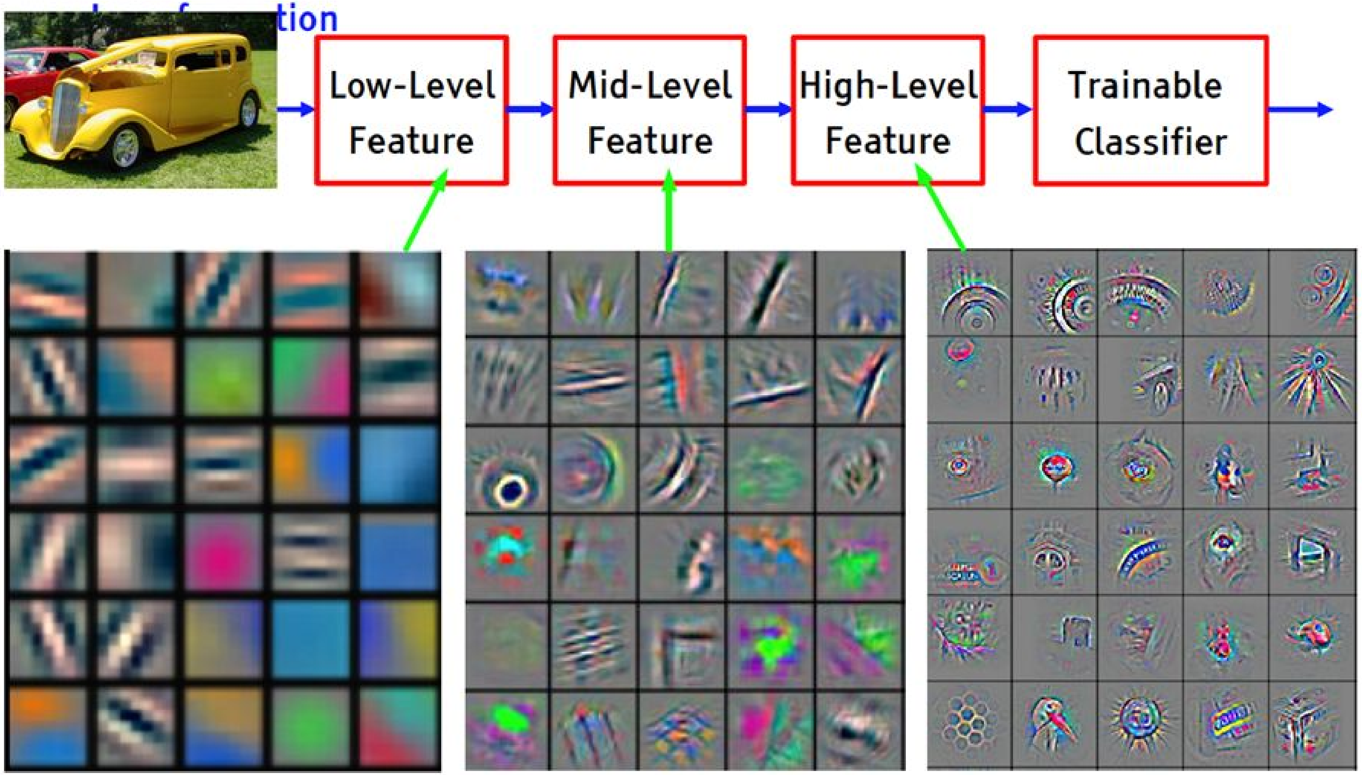


Regression



1) Linear Methods: Linear in **Input** Space
2) Kernel Methods: Linear in **Pre-Defined Feature** Space
3) Neural Network Methods: Linear in **Learned Feature** Space

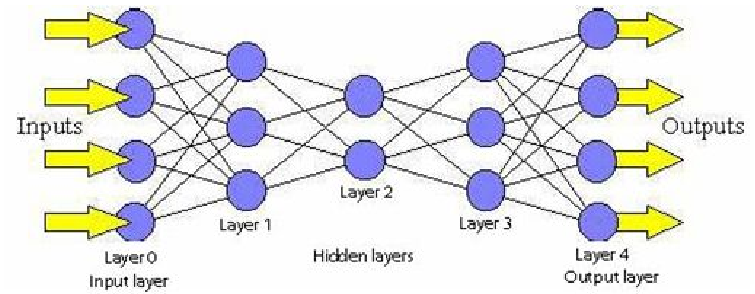
Deep Learning = Learning Hierarchical Representations

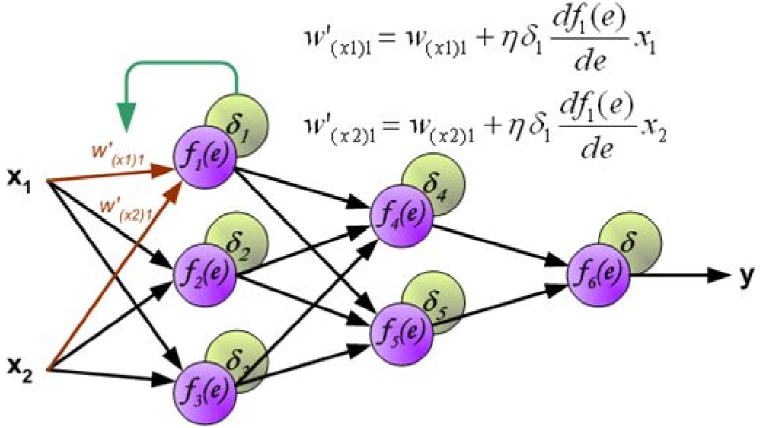


Low-Level Feature → Mid-Level Feature → High-Level Feature → Trainable Classifier →

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Training Neural Networks





$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1$$

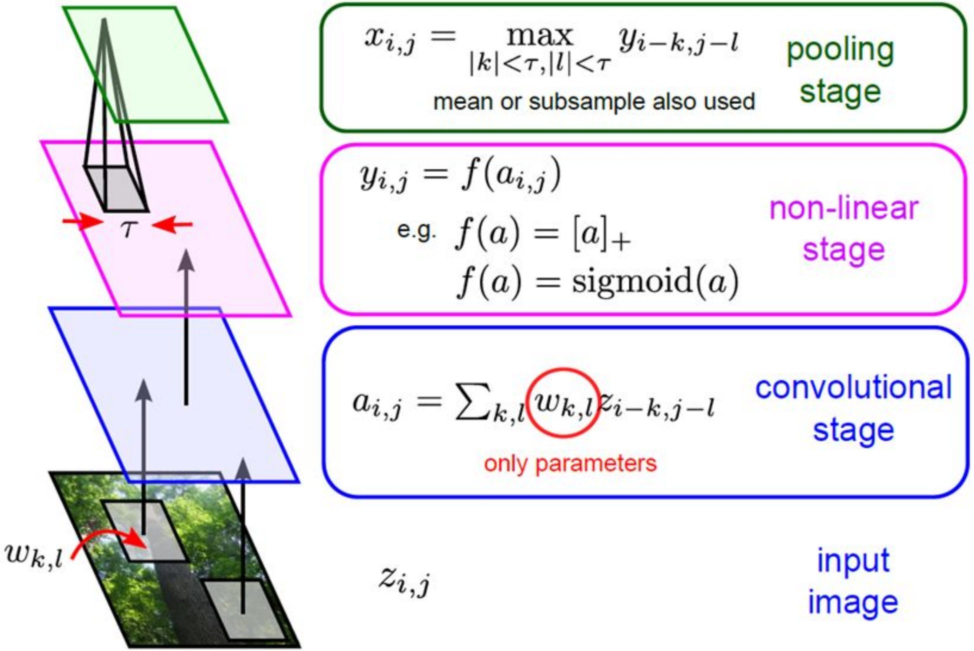
$$w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$

- **Feed Forward Network:**
  - Connections between neurons do *not* form a cycle
  - Examples: Fully-Connected, Convolutional Neural Network
  - Not feedforward: Recurrent Neural Network

- **Backpropagation / Gradient Descent:**
  - Randomly initialize the network parameters
  - Calculate loss error at the output
  - Calculate contributions to error at each step going backwards
  - Essentially Chain Rule

Convolutional Neural Network (CNN)

- Used for images
  - Parameter reduction by exploiting spatial locality
- Building Blocks for CNN:
  - Convolutional Layer
  - Non-linear Activation Function
  - Max-Pooling Layer
- Convolution Layer
  - Convolution instead of Matrix Multiplication
  - Usually implemented as cross-correlation/filtering (kernel not flip)
    - Tensorflow
  - Learn the weights of the kernel/filter

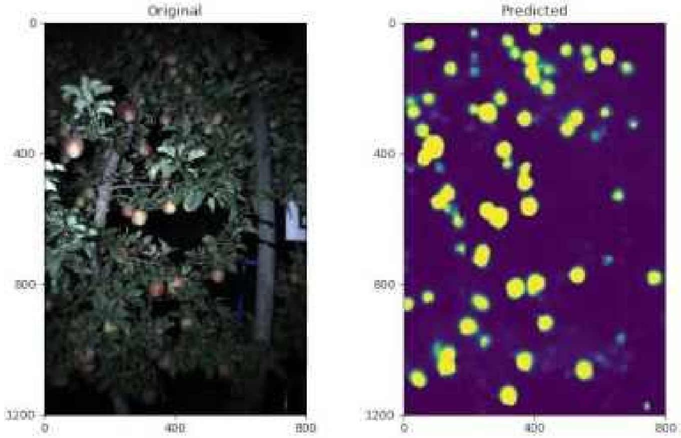

$x_{i,j} = \max_{|k|<\tau, |l|<\tau} y_{i-k,j-l}$
mean or subsample also used
pooling stage
$y_{i,j} = f(a_{i,j})$
e.g. $f(a) = [a]_+$
$f(a) = \mathrm{sigmoid}(a)$
non-linear stage
τ
$a_{i,j} = \sum_{k,l} w_{k,l} z_{i-k,j-l}$
only parameters
convolutional stage
$w_{k,l}$
$z_{i,j}$
input image

Building-blocks for CNN’s

Deep Learning in Computer Vision

Applications: (1) Classification, (2) Segmentation; (3) Image Registration; and more...

[redacted]



[redacted]

Example: Fruit Segmentation

1. Does not utilize prior knowledge about the problem (besides labels)
2. Standard “template” for any deep learning problem

Standard Deep Learning Template:

1) Collect image data and ground truth labels
2) Design network architecture
3) Train via supervised learning by minimizing a loss function against Ground Truth

Works well… but potential drawbacks:

1. Requires ground truth (not always available)
2. Limited generalization ability (need a lot of diverse data)
3. Long training times

Chen at al. Counting Apples and Oranges With Deep Learning: A Data-Driven Approach. 2017

Deep Learning in Image Registration

Classification and Segmentation have a lot of semantic problem structure

Image Registration is interesting because it has a lot of semantic and ***geometric*** structure

Key Theme of Lecture:

Incorporating problem structure and utilizing insights from traditional techniques can lead to more powerful/efficient deep learning algorithms

Applications:

1. Estimating Homographies (*supervised -> unsupervised:* improves performance and data efficiency)
2. Object Tracking (*offline -> hybrid offline/online:* improves generalization)
3. Stereo Vision
4. Visual Odometry
5. Image Mosaicing
6. … much more!

Case #1: Unsupervised Deep Homography

Estimating Homographies

Case #1 Outline

1) Deep Homography (Supervised)
2) Deep Homography (Unsupervised)
3) Traditional Feature-Based Methods

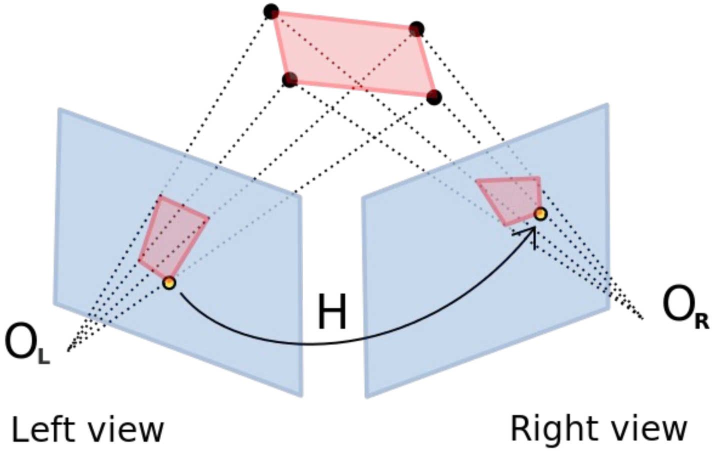
Homography Definition

A homography that maps $\mathbf{x} \leftrightarrow \mathbf{x}'$ is a linear transformation represented by a non-singular 3×3 matrix $\mathbf{H}$ such that:

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad \text{Or} \quad \mathbf{x}' = \mathbf{Hx}$$

- Project points on one plane to points on another plane.

- 8 free parameters

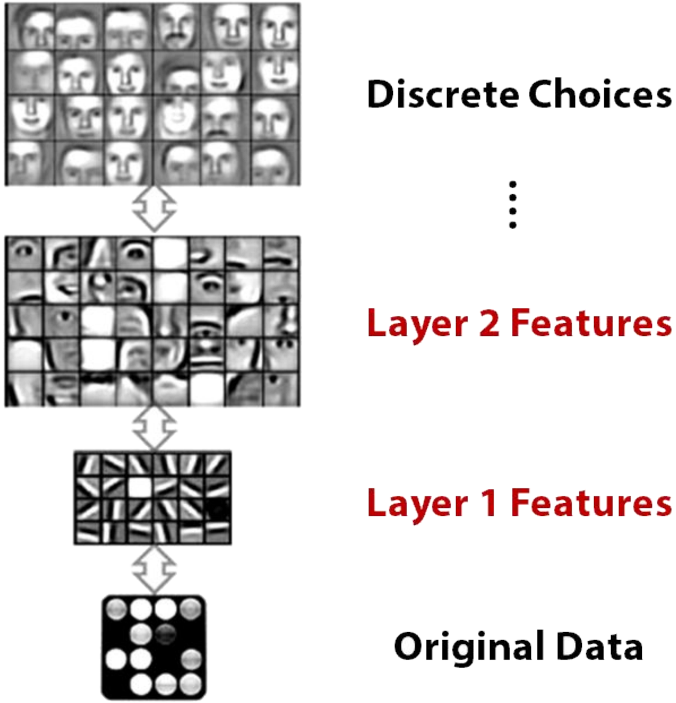


Deep Learning Based Method

- Recap: **Deep Learning = Learning Hierarchical Representations**
- Use these features to compute homography

$$\mathbf{H} = \boldsymbol{f}\left(\textit{features on image 1, features on image 2}\right)$$

- Network
  - **Input**: Pair of images
  - **Output**: Vector of 8 parameters
- Training Approaches
  - Supervised
  - Unsupervised



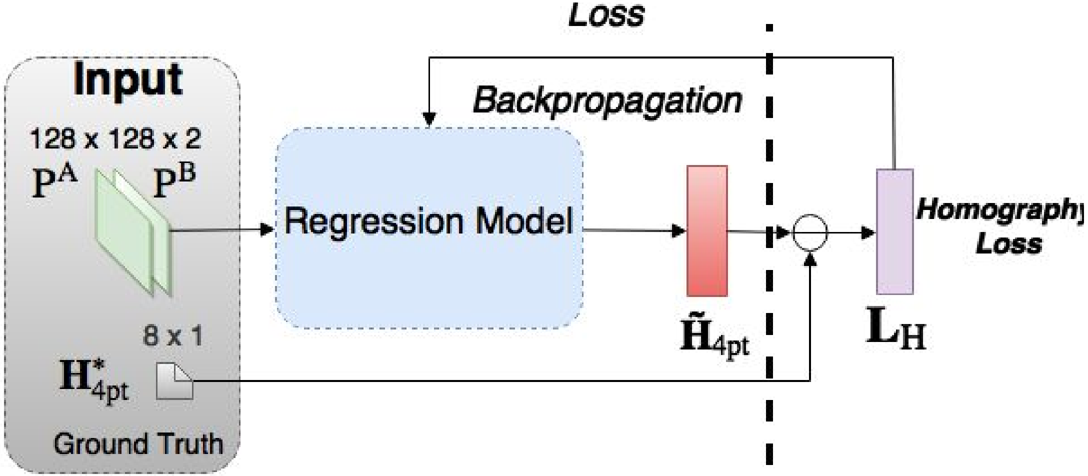
Hierarchical Representations of a Deep CNN

Supervised Deep Homography

- Needs ground truth, which are homographies parameters in the training data
- **Expensive to obtain homography ground truth on real data**
- Loss function:

$$\mathbf{L}_H = \frac{1}{2}||\tilde{\mathbf{H}}_{4pt} - \mathbf{H}^*_{4pt}||_2^2$$

Diagram of Supervised Approach



Loss
Input
128 x 128 x 2
$\mathbf{P}^{\mathrm{A}}$
$\mathbf{P}^{\mathrm{B}}$
Backpropagation
Regression Model
$\tilde{\mathbf{H}}_{\mathrm{4pt}}$
Homography Loss
$\mathbf{L}_{\mathrm{H}}$
8 x 1
$\mathbf{H}^*_{\mathrm{4pt}}$
Ground Truth

(a) Supervised Approach

Regression Model (a Deep CNN model)

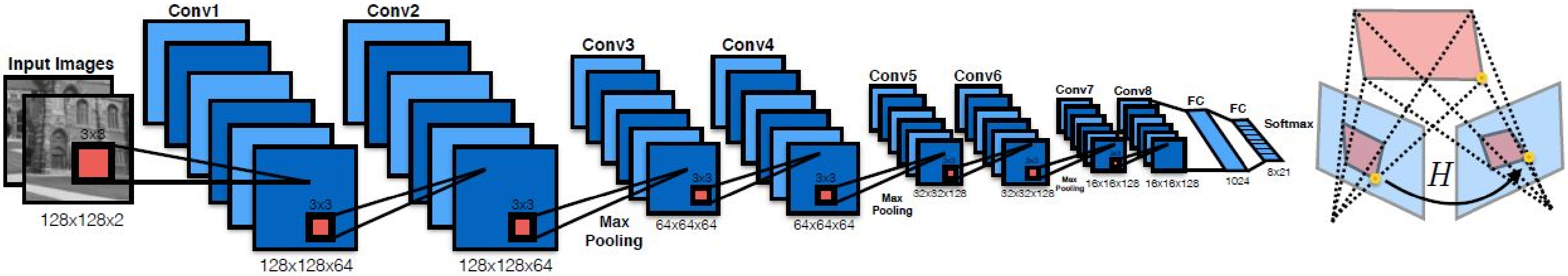


Fig. 1: **Deep Image Homography Estimation.** HomographyNet is a Deep Convolutional Neural Network which directly produces the Homography relating two images. Our method does net require separate corner detection and homography estimation steps and all parameters are trained in an end-to-end fashion using a large dataset of labeled images.

Geometric Loss vs Supervised Loss

- Supervised Loss:
  - Sum of square error in homography parameters
  - **Requires ground truth**: not always available
  - Does not capture any properties of the transformation
- Geometric Loss:
  - Sum of square error in pixel intensity values
  - Same objective as Direct Methods of Homography Estimation
  - Requires **no ground truth** => Great!
  - Captures the consistency property of the transformation => Great!

$$\mathbf{L}_H = \frac{1}{2}||\tilde{\mathbf{H}}_{4pt} - \mathbf{H}^*_{4pt}||_2^2$$

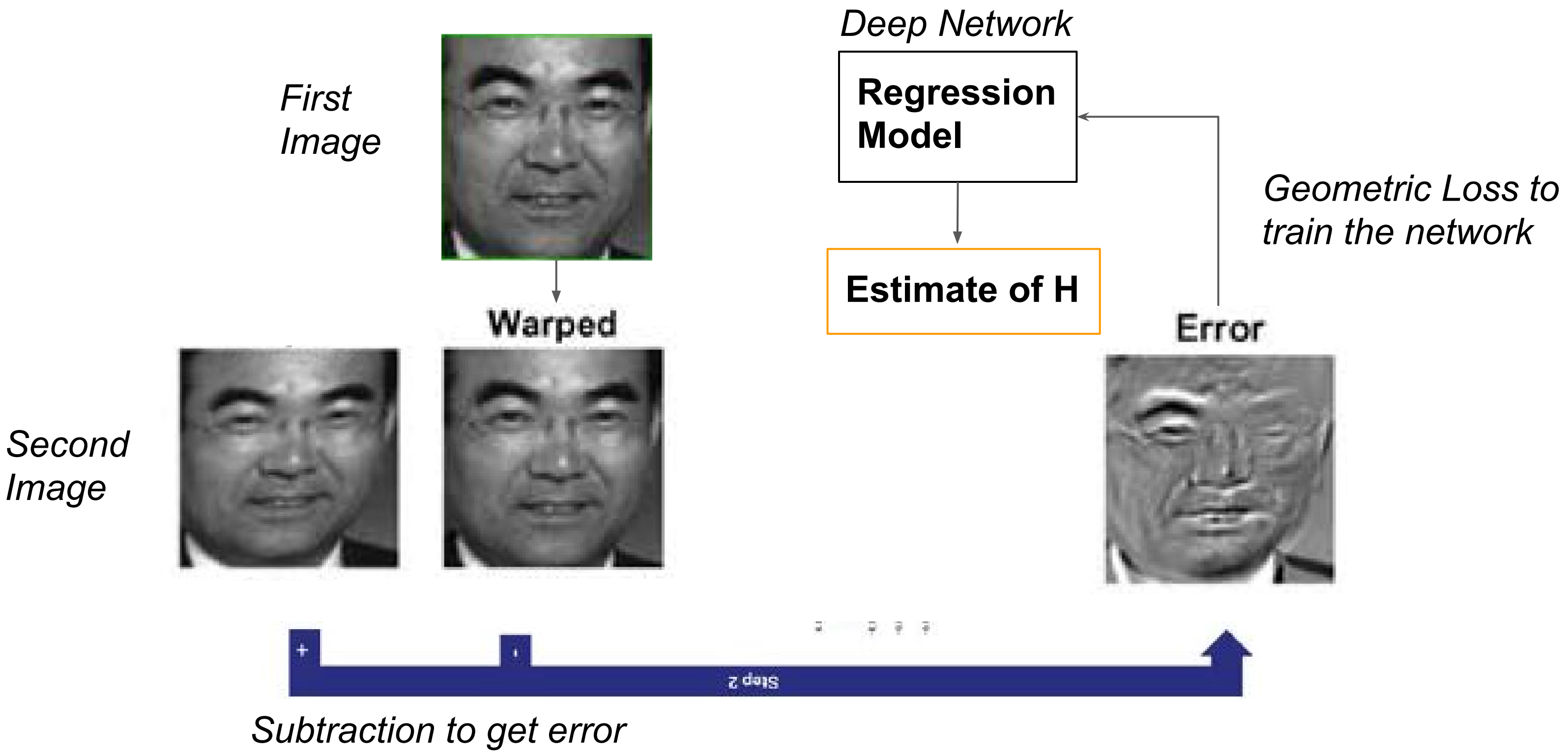
Supervised Loss

$$\mathbf{L}_{PW} = \frac{1}{2|\mathbf{x}_i|}\sqrt{\sum_{\mathbf{x}_i}\left[I^A(\mathscr{H}(\mathbf{x}_i)) - I^B(\mathbf{x_i})\right]^2}$$

Geometric Loss

Idea: make use of the geometric loss => A Better Unsupervised Approach

Unsupervised Deep Homography: Idea



First Image

Second Image

Deep Network

Regression Model

Estimate of H

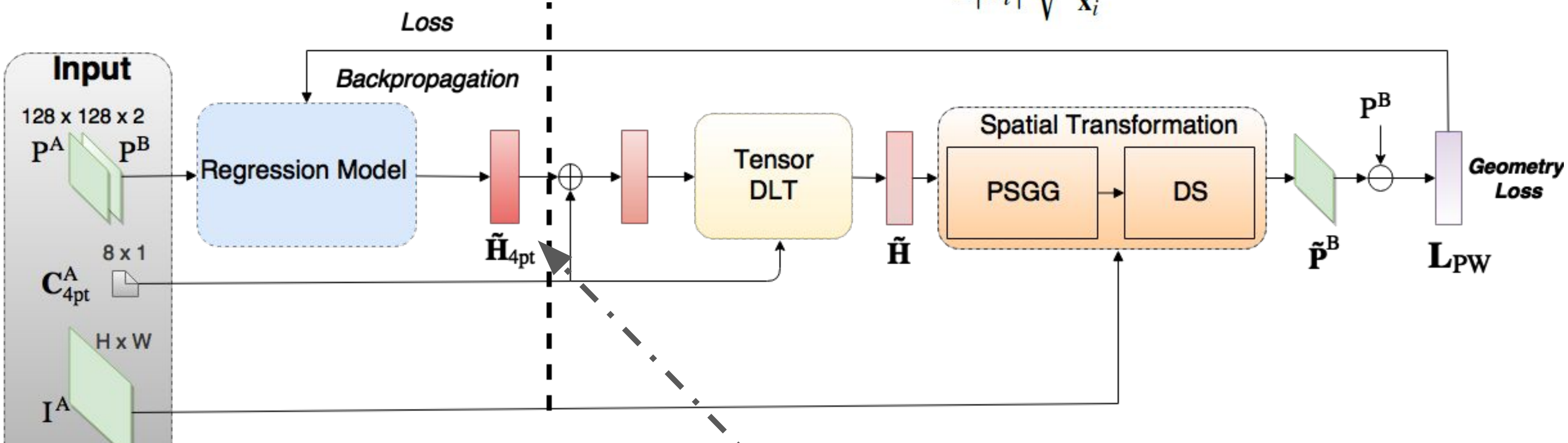
Geometric Loss to train the network

Subtraction to get error

Unsupervised Deep Homography: Diagram

- Use “unsupervised” loss function
- **Does not use ground truth!**

$$\mathbf{L}_{PW} = \frac{1}{2|\mathbf{x}_i|}\sqrt{\sum_{\mathbf{x}_i}\left[I^A(\mathscr{H}(\mathbf{x}_i)) - I^B(\mathbf{x_i})\right]^2}$$



(c) Unsupervised Approach

Output of the Deep CNN: *Estimate of parameters of homography*

2 new network structures:

1. Tensor Direct Linear Transform (DLT)
2. Spatial Transformation Layer

Direct Linear Transform (DLT)

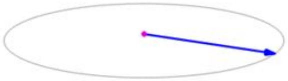
- Recap: $\mathbf{x}' = \mathbf{H}\mathbf{x}$
  - Given 4 pairs of correspondences → **H**
- Consider 1 pair of correspondences

  x$_i$ **=** (u,v, 1) and **x**$_i$’ = (u’, v’, 1’)
- Use cross-product to get rid of scale

$$\mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = 0$$

$$\mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = \begin{bmatrix} v_i'\mathbf{x}_i^T\mathbf{h}^3 - \mathbf{x}_i^T\mathbf{h}^2 \\ \mathbf{x}_i^T\mathbf{h}^1 - u_i'\mathbf{x}_i^T\mathbf{h}^3 \\ u_i'\mathbf{x}_i^T\mathbf{h}^2 - v_i'\mathbf{x}_i^T\mathbf{h}^1 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0_{3\times1}^T & -\mathbf{x}_i^T & v_i'\mathbf{x}_i^T \\ \mathbf{x}_i^T & 0_{3\times1}^T & -u_i'\mathbf{x}_i^T \\ -v_i'\mathbf{x}_i^T & u_i'\mathbf{x}_i^T & 0_{3\times1}^T \end{bmatrix} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix} = 0$$



$$\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \sin(\theta) \, \mathbf{n}$$

One pair, 2 independent equations

$$\mathbf{A}_i^{(3)}\mathbf{h} = \mathbf{0}$$

Direct Linear Transform (DLT)

- **H** can be determined with 4 pairs of correspondences (8 unknown parameters)
- Solving for **h** equivalent to finding the null space of **A** (8x9 matrix) ⟶ SVD ?

- Alternative to SVD, write equation $\mathbf{A}\mathbf{h} = \mathbf{0}$ to the form $\hat{\mathbf{A}}\hat{\mathbf{h}} = \hat{\mathbf{b}}$ by moving the last column Of **A** to the right (since $\mathbf{h}_{33}$ = 1)

One pair, 2 independent equations

$$\longrightarrow \mathbf{A_i h = 0}$$

4 pairs, 8 independent equations

$$\mathbf{A}\mathbf{h} = \mathbf{0}$$

$$\begin{bmatrix} 0^T_{3\times1} & -\mathbf{x}_i^T & v_i'\mathbf{x}_i^T \\ \mathbf{x}_i^T & 0^T_{3\times1} & -u_i'\mathbf{x}_i^T \\ -v_i'\mathbf{x}_i^T & u_i'\mathbf{x}_i^T & 0^T_{3\times1} \end{bmatrix} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix} = 0$$

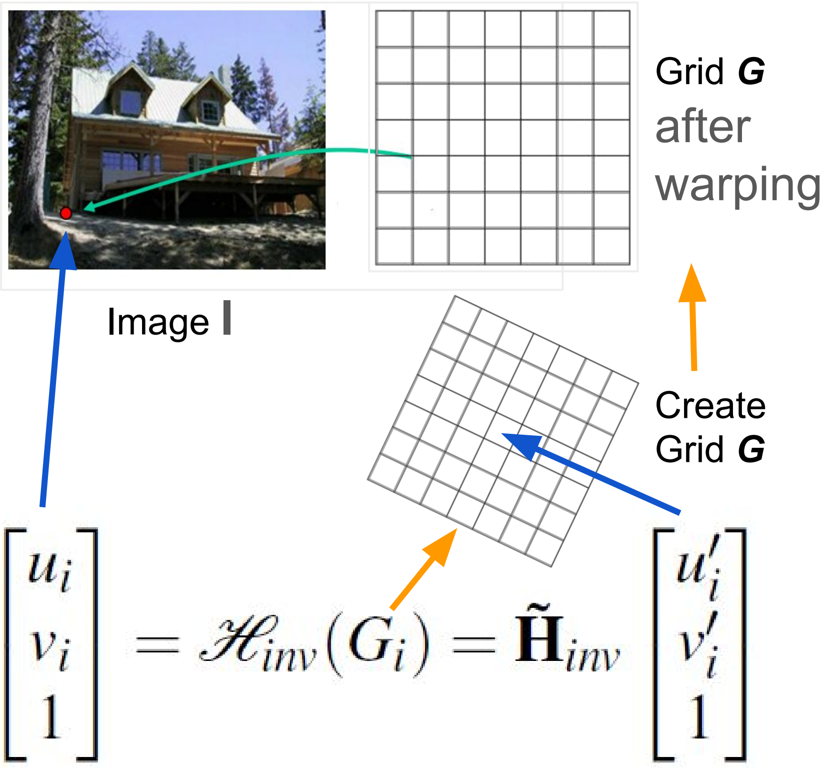
- This is straight-forward to solve and easy to compute gradients w.r.t elements of **H**

Spatial Transformer

- A warping algorithm that transforms input image **I** to a new image **I'** given a transformation matrix **H** (**H** can be homography, affine transform …)
- Differentiable w.r.t elements of **H**
- Two steps: grid generator & differentiable sampling
  - Grid generator: $G = \{G_i\}$

    $G_i = (u'_i, v'_i)$ Is a pixel in the image **I'**

    - Applying inverse of **H** to ***G***
    - Differentiable sampling to pain ***G*** **,** to obtain image **I'**

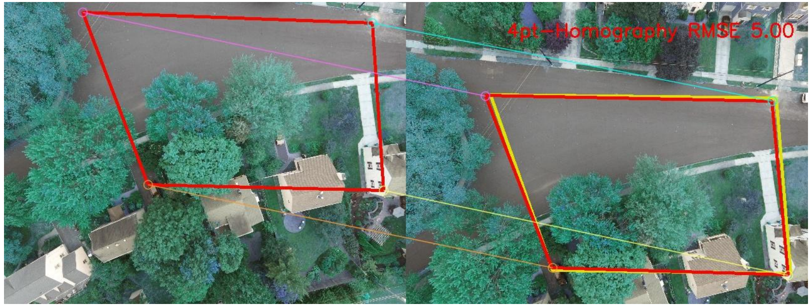


$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathscr{H}_{inv}(G_i) = \tilde{\mathbf{H}}_{inv} \begin{bmatrix} u'_i \\ v'_i \\ 1 \end{bmatrix}$$

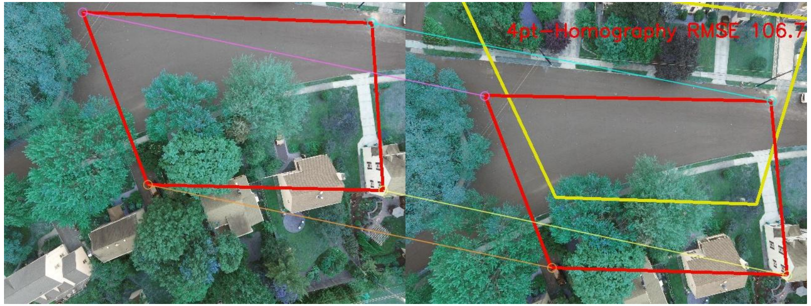
Results: Performance on Real Images

- Evaluate accuracy of homography estimation
- Use 4 pairs of correspondences. Red color: ground truth, Yellow color: estimation.
  Expectation: yellow polygon overlaps red polygon

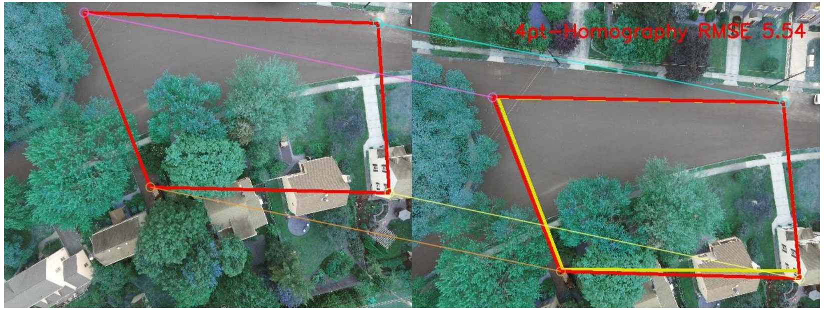
Unsupervised



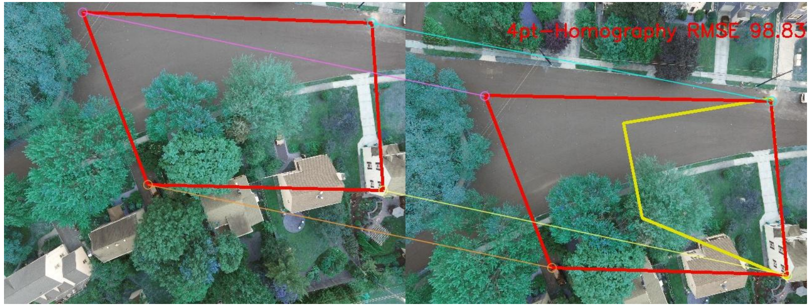
Supervised



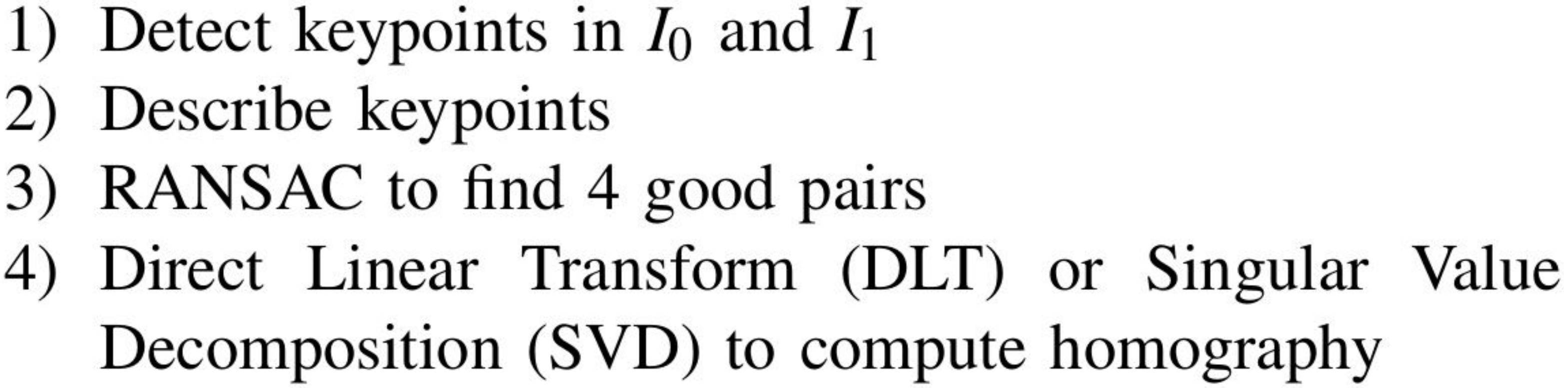
SIFT



ORB



Feature-Based Approach

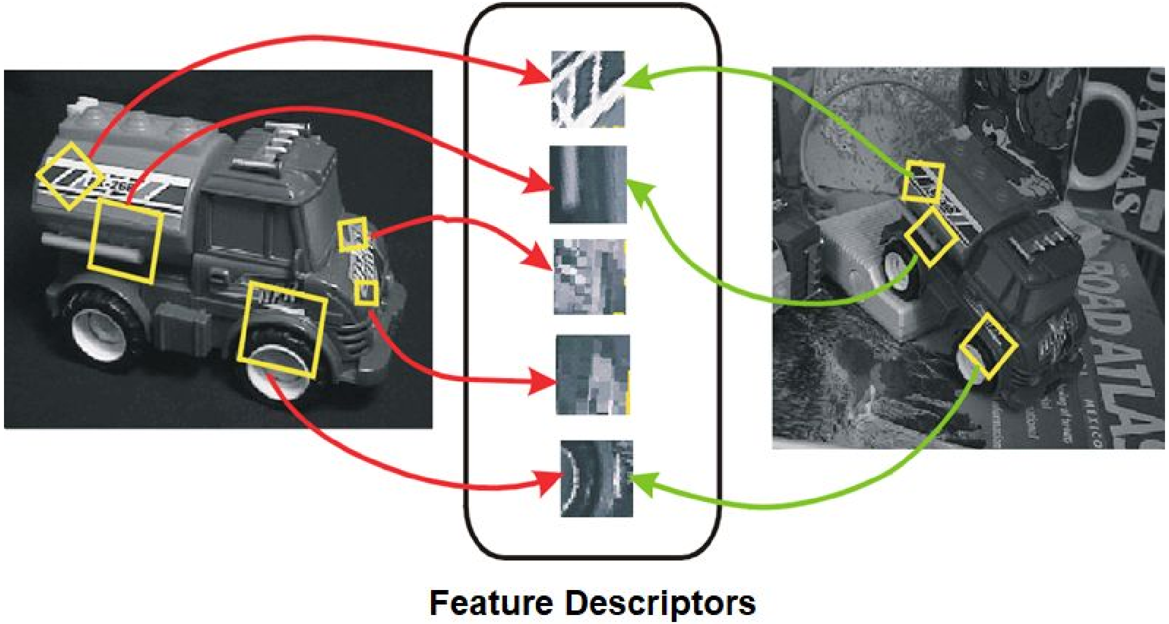


1) Detect keypoints in I_0 and I_1
2) Describe keypoints
3) RANSAC to find 4 good pairs
4) Direct Linear Transform (DLT) or Singular Value Decomposition (SVD) to compute homography

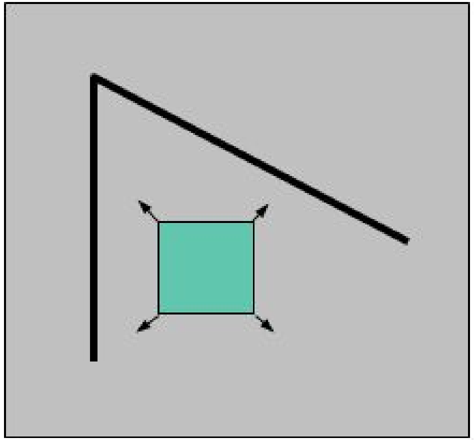
Note: these are steps that you may refer to when doing your upcoming project!

Detecting and Describing Local Features

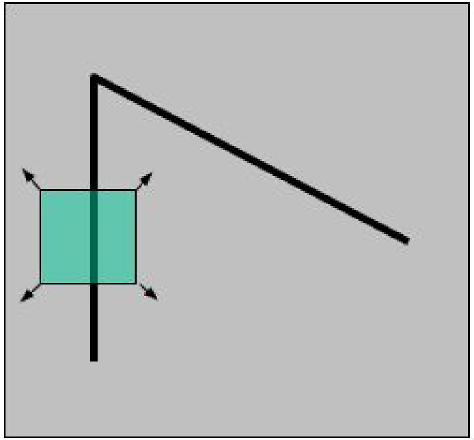
- How to find a shared pattern of the two images?
- Pixel intensity has a short range, from 0 to 255
- Idea: use a patch other than a single pixel



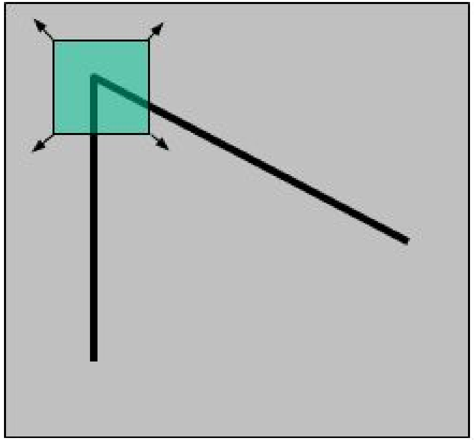
Simple Local Features (examples)



"flat" region:
no change in all directions



"edge":
no change along the edge direction

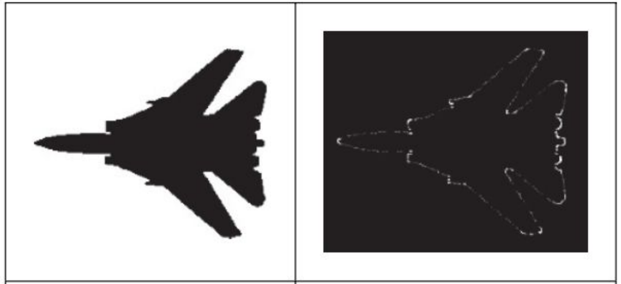


"corner":
significant change in all directions

Simple Local Features

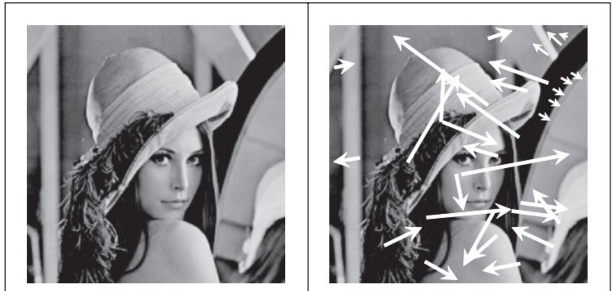
- **Corner (curvature) extraction:** points where the edge direction changes rapidly are corners. I.e. Harris corner, Shi-Tomasi features
  Pros: *fast*
  Cons: *sensitive to changes in image scale and as such is unsuited to matching images of differing size*
- **Scale invariant feature transform (SIFT):** involves two stages: feature extraction and description.
  Pros: *invariant to image scale and rotation, and with partial invariance to change in illumination.*
  Cons: *Computationally expensive*

There are numerous studies in the literature!



(a) Image

(b) Detected corners

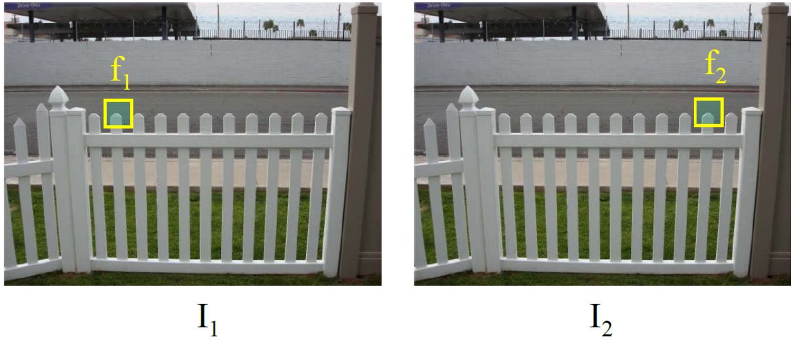


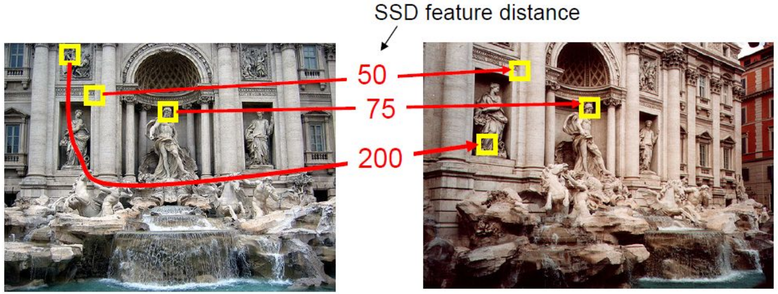
(a) Original image

(b) Output points with magnitude and direction

Feature Matching

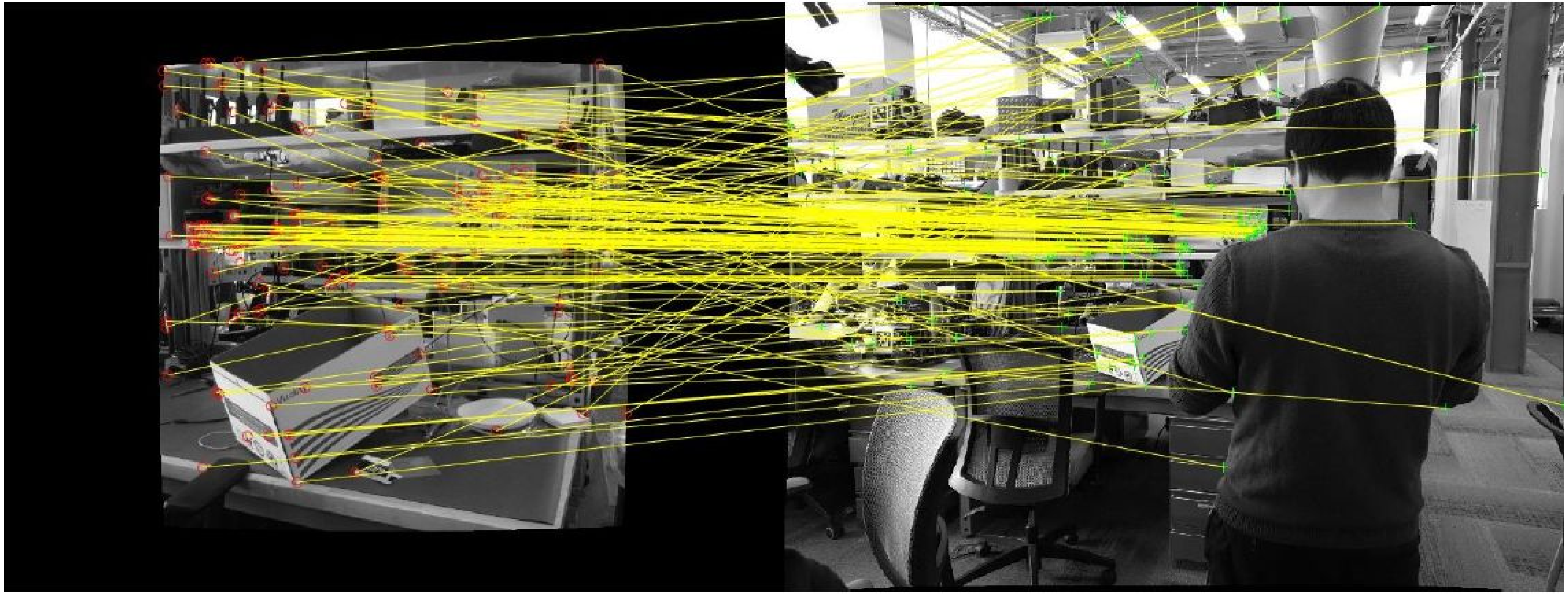
- How to define the similarity between two features f_1, f_2 ?
  - Simple approach is SSD(f_1, f_2): sum of square differences between entries of two descriptors
    Minimize $\sum_i (f(i) - g(j))^2$
  - Does not provide a way to discard ambiguous (bad) matches
    - Better approach: minimize
      - ratio distance = SSD(f_1, f_2) / SSD(f_1, f'_2)
      - Decision rule: accept a match if SSD < T





RANSAC: Motivation Example

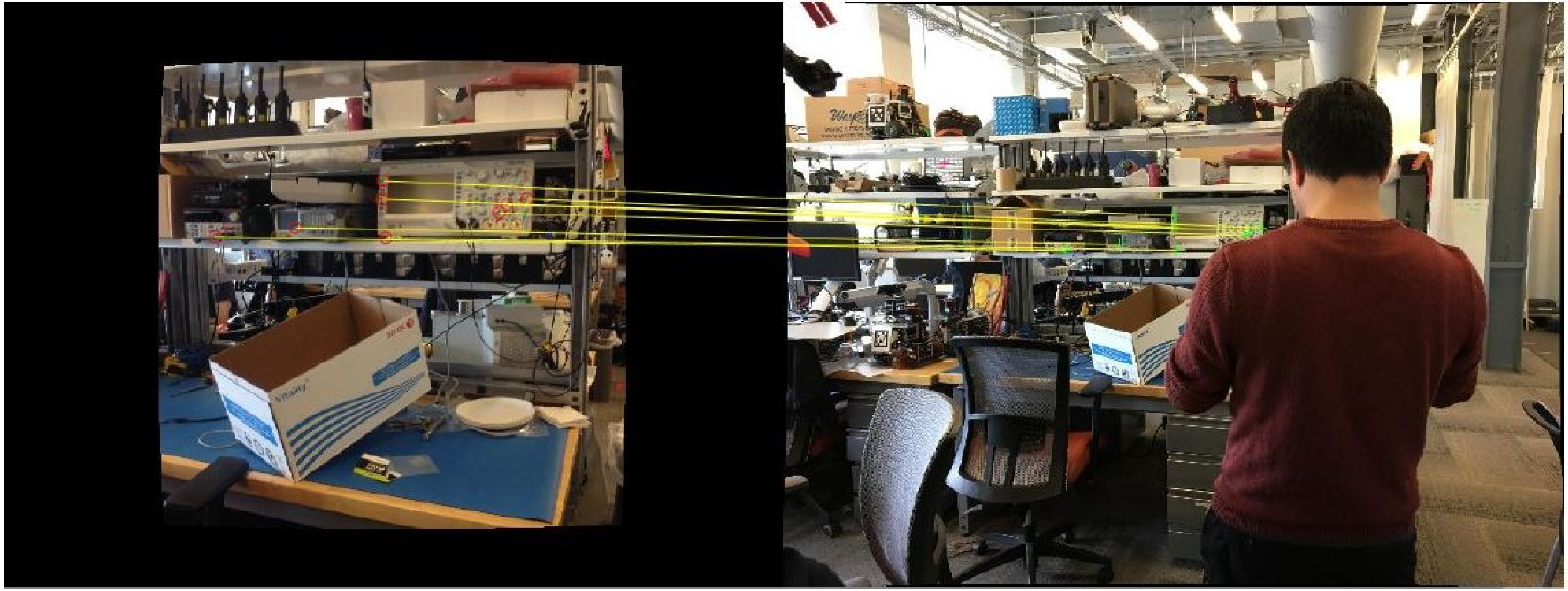
- There are outliers which represent wrong matches
- How to find good correspondences?



(a) Matches before RANSAC

RANSAC: Motivation Example

- Outliers disappear!



(b) Matches after RANSAC

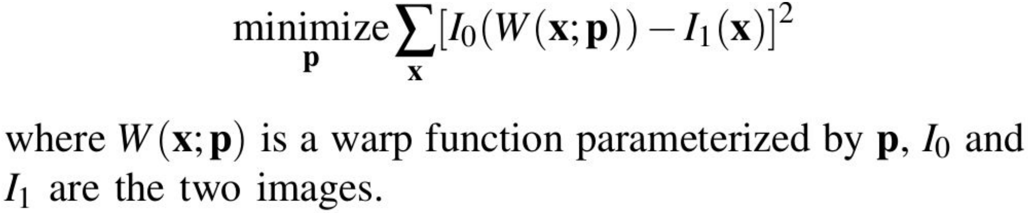
Case #2: Deep-LK

Tracking Objects

Case #2 Outline

1) Traditional Lucas-Kanade Algorithm
    a) Forward-Additive
    b) Inverse-Compositional
2) GOTURN (Offline Deep Learning Tracker w/o LK)
3) Deep-LK (Hybrid Online/Offline Deep Learning Tracker w/ LK)

Unsupervised Homography uses LK Objective



$$\underset{\mathbf{p}}{\text{minimize}} \sum_{\mathbf{x}} [I_0(W(\mathbf{x};\mathbf{p})) - I_1(\mathbf{x})]^2$$

where $W(\mathbf{x};\mathbf{p})$ is a warp function parameterized by $\mathbf{p}$, I_0 and I_1 are the two images.

E.g. affine warp

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Lucas-Kanade (Direct Method) applies to general differentiable warps and is used in:

1. Tracking
2. Optical Flow
3. Mosaicing

We will introduce the Deep-LK network which is used in object tracking.

Overview of Lucas-Kanade Algorithm

Original minimization

$$\underset{\mathbf{p}}{\text{minimize}} \sum_{\mathbf{x}} [I(W(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]^2$$

Iteratively solve:

$$\underset{\triangle\mathbf{p}}{\text{minimize}} \sum_{\mathbf{x}} [I(W(\mathbf{x};\mathbf{p}+\triangle\mathbf{p})) - T(\mathbf{x})]^2$$

and update $\mathbf{p} \leftarrow \mathbf{p} + \triangle\mathbf{p}$.

However $I(W(\mathbf{x};\mathbf{p}+\triangle\mathbf{p}))$ is non-linear...

First-Order Taylor Expansion (Gauss-Newton)

$$\underset{\triangle\mathbf{p}}{\text{minimize}}\sum_{\mathbf{x}}[I(W(\mathbf{x};\mathbf{p})+\nabla I\frac{\partial W}{\partial \mathbf{p}}\triangle\mathbf{p}-T(\mathbf{x})]^2$$

where $\nabla I = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]$ is *image gradient* evaluated at $W(\mathbf{x};\mathbf{p})$ and $\frac{\partial W}{\partial \mathbf{p}}$ is *warp Jacobian*.The partial derivative is:

$$2\sum_{\mathbf{x}}\left[\nabla I\frac{\partial W}{\partial \mathbf{p}}\right]^T\left[I(W(\mathbf{x};\mathbf{p}))+\nabla I\frac{\partial W}{\partial \mathbf{p}}\triangle\mathbf{p}-T(\mathbf{x})\right]$$

The closed-form solution is:

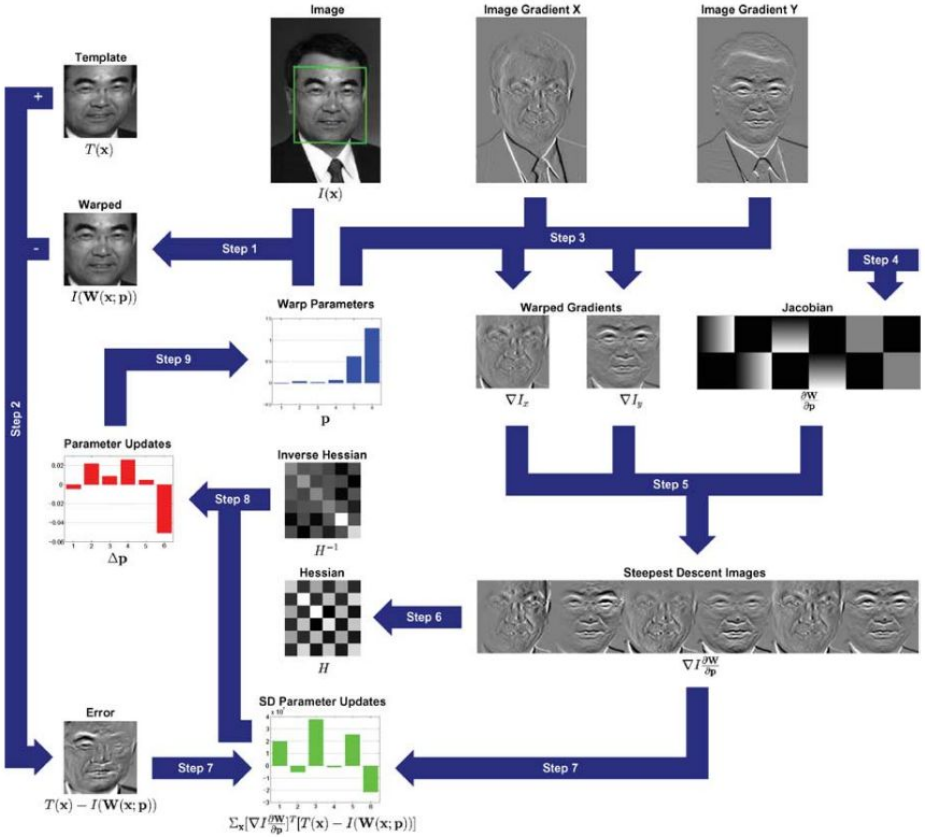
$$\triangle\mathbf{p} = H^{-1}\sum_{x}\left[\nabla I\frac{\partial W}{\partial \mathbf{p}}\right]^T[T(\mathbf{x})-I(W(\mathbf{x};\mathbf{p}))]$$

where H is Gauss-Newton approximation to *Hessian* matrix:

$$H = \sum_{x}\left[\nabla I\frac{\partial W}{\partial \mathbf{p}}\right]^T\left[\nabla I\frac{\partial W}{\partial \mathbf{p}}\right]$$

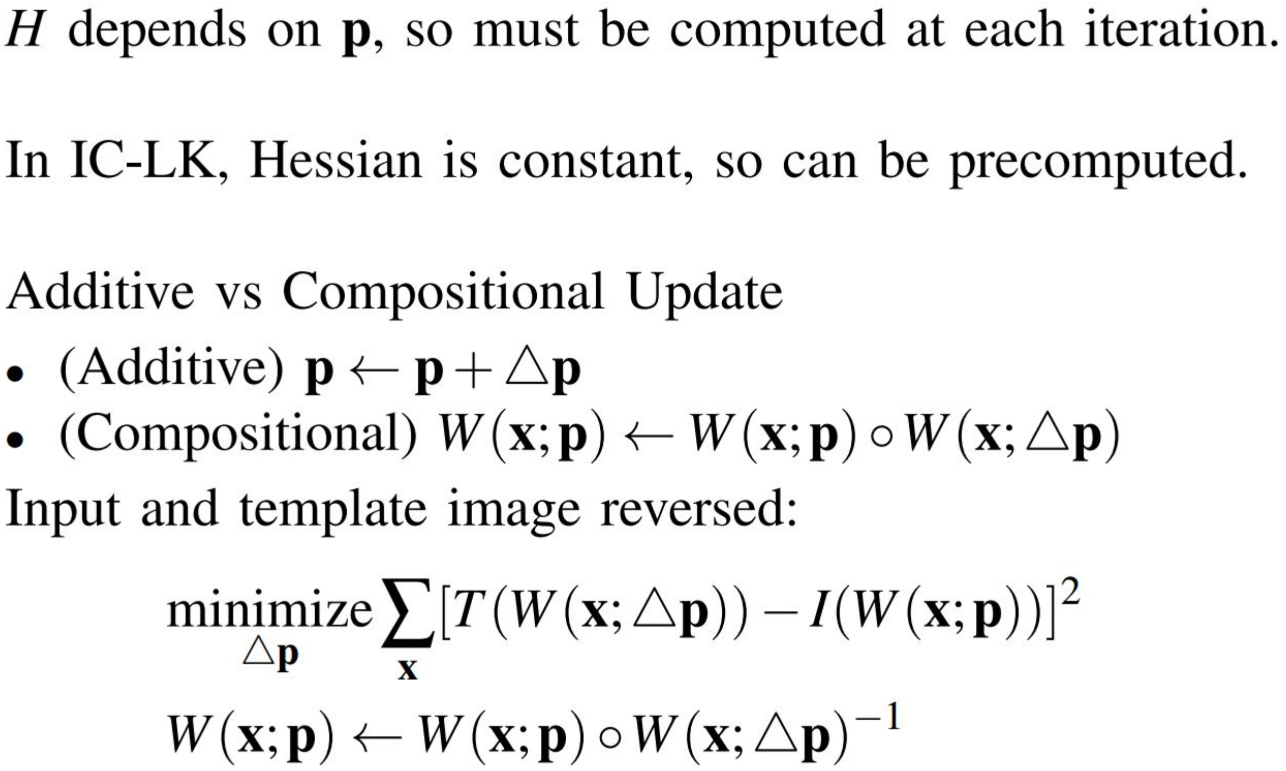
E.g. affine warp Jacobian

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix}$$



Baker and Matthews. Lucas-Kanade 20 Years On: A Unifying Framework (2004)

Inverse-Compositional LK Algorithm



H depends on $\mathbf{p}$, so must be computed at each iteration.

In IC-LK, Hessian is constant, so can be precomputed.

Additive vs Compositional Update

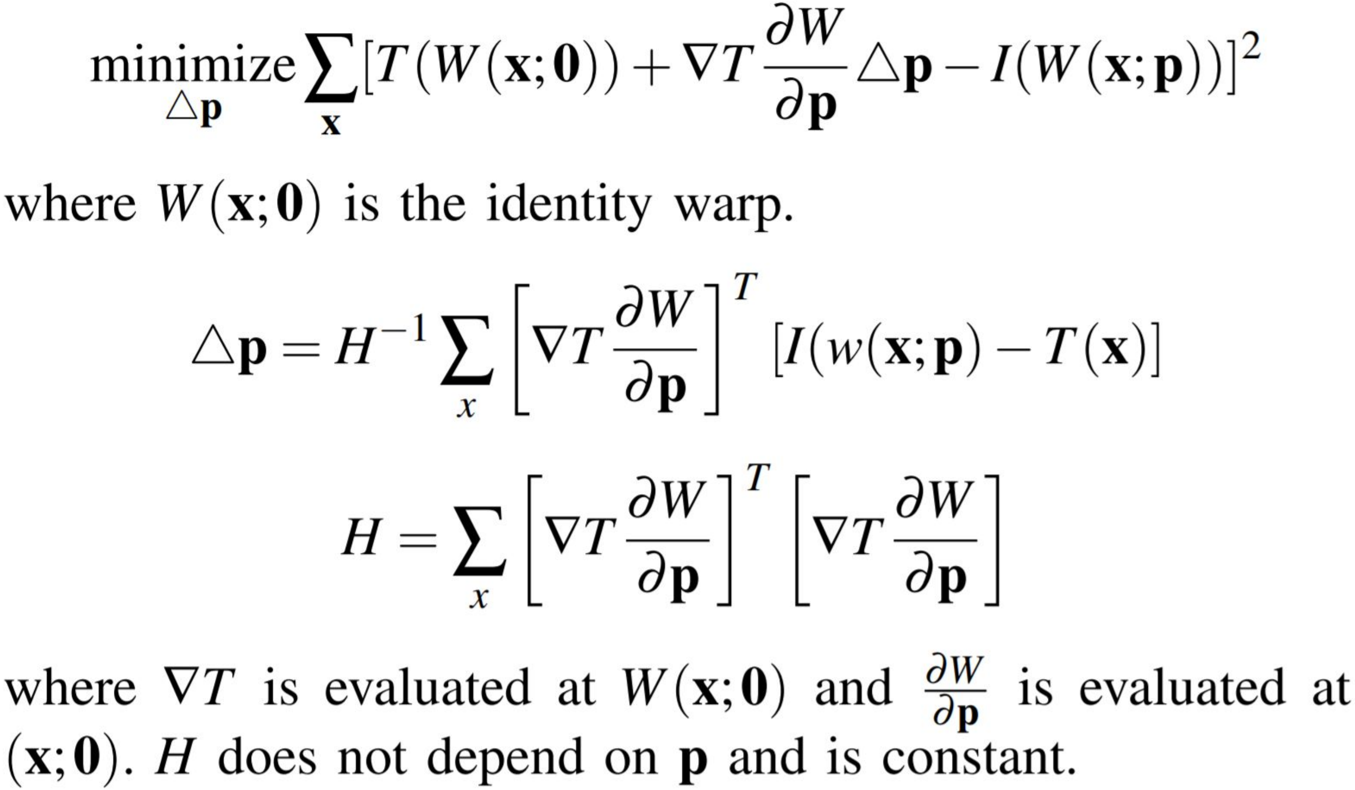
- (Additive) $\mathbf{p} \leftarrow \mathbf{p} + \triangle\mathbf{p}$
- (Compositional) $W(\mathbf{x};\mathbf{p}) \leftarrow W(\mathbf{x};\mathbf{p}) \circ W(\mathbf{x};\triangle\mathbf{p})$

Input and template image reversed:

$$\underset{\triangle\mathbf{p}}{\text{minimize}} \sum_{\mathbf{x}} [T(W(\mathbf{x};\triangle\mathbf{p})) - I(W(\mathbf{x};\mathbf{p}))]^2$$

$$W(\mathbf{x};\mathbf{p}) \leftarrow W(\mathbf{x};\mathbf{p}) \circ W(\mathbf{x};\triangle\mathbf{p})^{-1}$$

Precomputing Jacobian and Hessian



$$\underset{\triangle\mathbf{p}}{\text{minimize}}\sum_{\mathbf{x}}[T(W(\mathbf{x};\mathbf{0}))+\nabla T\frac{\partial W}{\partial \mathbf{p}}\triangle\mathbf{p}-I(W(\mathbf{x};\mathbf{p}))]^2$$

where $W(\mathbf{x};\mathbf{0})$ is the identity warp.

$$\triangle\mathbf{p}=H^{-1}\sum_{x}\left[\nabla T\frac{\partial W}{\partial \mathbf{p}}\right]^T[I(w(\mathbf{x};\mathbf{p})-T(\mathbf{x})]$$

$$H=\sum_{x}\left[\nabla T\frac{\partial W}{\partial \mathbf{p}}\right]^T\left[\nabla T\frac{\partial W}{\partial \mathbf{p}}\right]$$

where ∇T is evaluated at $W(\mathbf{x};\mathbf{0})$ and $\frac{\partial W}{\partial \mathbf{p}}$ is evaluated at $(\mathbf{x};\mathbf{0})$. H does not depend on $\mathbf{p}$ and is constant.

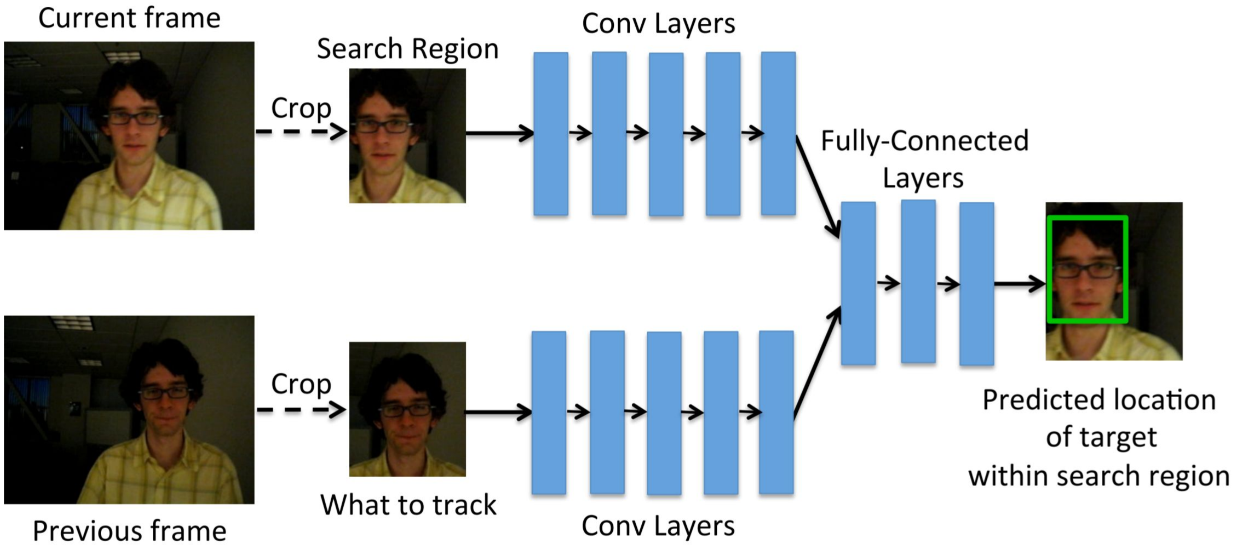
GOTURN [1]

Regression based

Fast (100 FPS)

No LK

GOTURN has poor generalization



[2] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. 2016

Deep-LK

Minimize feature distance instead of intensity distance

Deep-LK uses IC-LK algorithm with update rule:

$$\triangle \mathbf{p} = M^{\dagger}\left[\phi(I(W(\mathbf{x};\mathbf{p})) - \phi(T(\mathbf{x}))\right]$$

$$M^{\dagger} = \sum_{x}\left(S_{\phi}^{T} S_{\phi}\right)^{-1} S_{\phi}^{T}$$

$M^{\dagger}$ is the Moore-Penrose pseudoinverse
$\phi(\cdot)$ is a feature extraction function (convnet).

Before $S = \nabla T \frac{\partial W}{\partial \mathbf{p}} = \frac{\partial T(W(\mathbf{x};\mathbf{p}))}{\partial W(\mathbf{x};\mathbf{p})} \frac{\partial W(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}}\Big|_{\mathbf{p}=0}$

Now $S_{\phi} = \frac{\partial \phi(T(W(\mathbf{x};\mathbf{p})))}{\partial T(W(\mathbf{x};\mathbf{p}))} \frac{\partial T(W(\mathbf{x};\mathbf{p}))}{\partial W(\mathbf{x};\mathbf{p})} \frac{\partial W(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}}\Big|_{\mathbf{p}=0}$

Can view as regression:

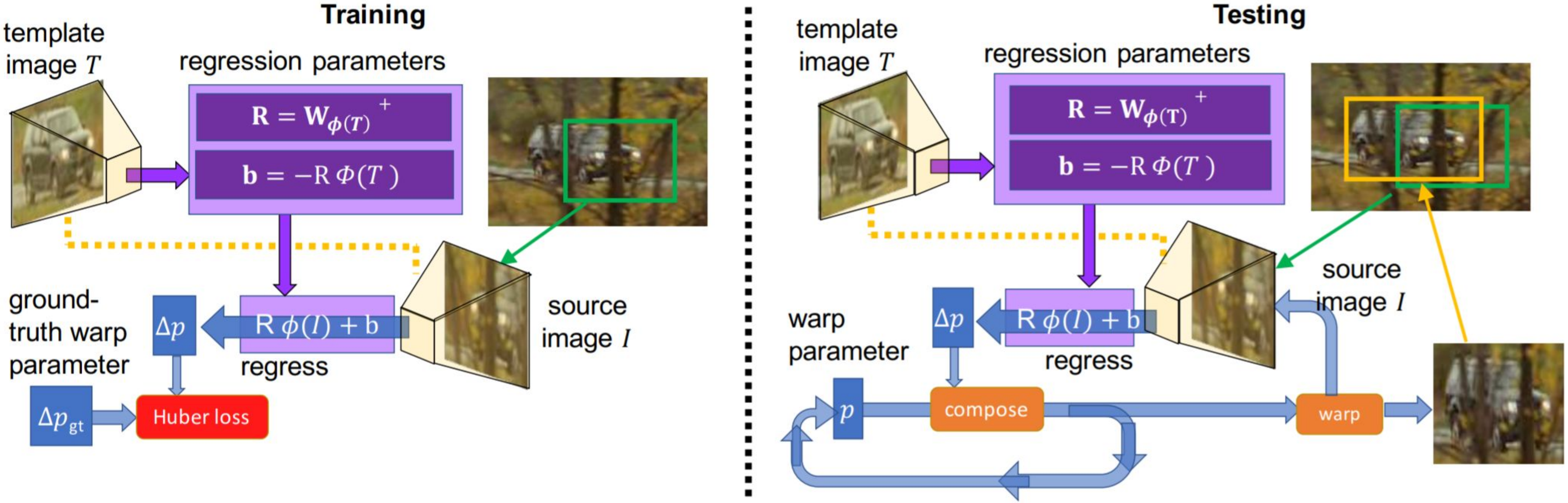
$$R \cdot \phi(I(W(\mathbf{x};\mathbf{p})) + b$$

$$R = M^{\dagger}$$

$$b = -M^{\dagger} \cdot \phi(T(\mathbf{x}))$$

[1] Wang et al. Deep-LK for Efficient Adaptive Object Tracking. 2017

Deep-LK



Training
template image T
regression parameters
$\mathbf{R} = \mathbf{W}_{\phi(T)}^{+}$
$\mathbf{b} = -\mathrm{R}\,\Phi(T)$
ground-truth warp parameter
Δp
$\mathrm{R}\,\phi(I) + \mathrm{b}$
regress
source image I
Δp_{gt}
Huber loss
Testing
template image T
regression parameters
$\mathbf{R} = \mathbf{W}_{\phi(\mathrm{T})}^{+}$
$\mathbf{b} = -\mathrm{R}\,\Phi(T)$
source image I
warp parameter
Δp
$\mathrm{R}\,\phi(I) + \mathrm{b}$
regress
p
compose
warp

Wang et al. Deep-LK for Efficient Adaptive Object Tracking. 2017

Results

Wang et al. Deep-LK for Efficient Adaptive Object Tracking. 2017

Thank you!