

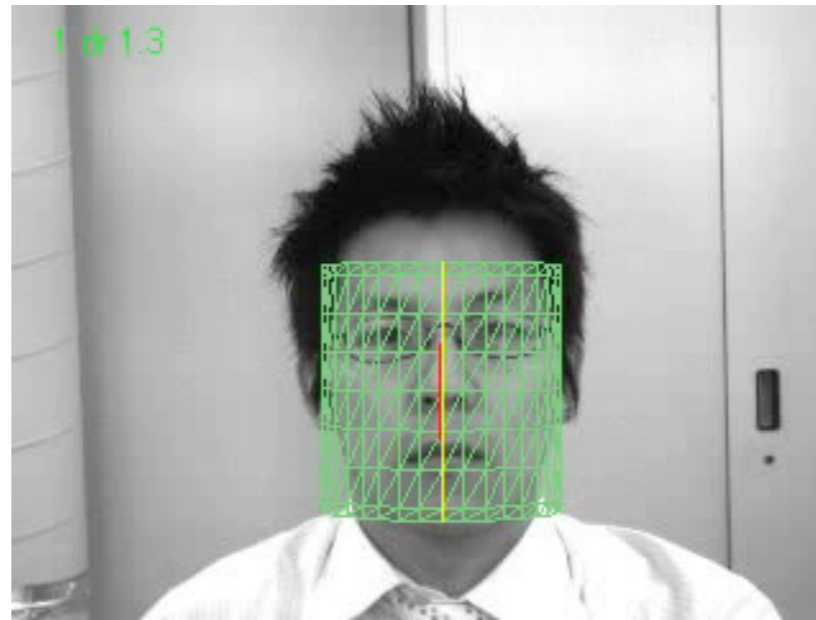
Object Registration and Tracking from a Learning Perspective (Part I)

Simon Lucey
The Robotics Institute, Carnegie Mellon University

Some Motivation for Course



“Face Recognition”



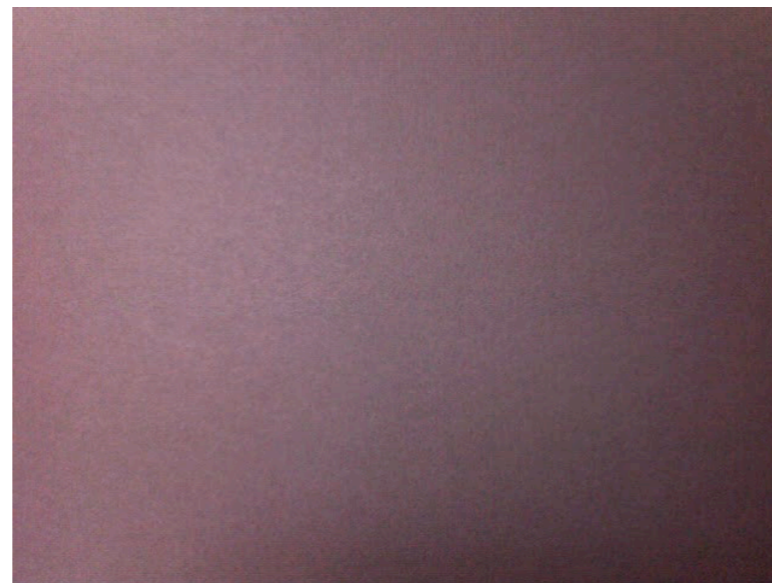
“Pose Estimation”



“Body Tracking”



“Speech Reading”



“Palm Recognition”



“Car Tracking”

Vision and Learning

“A vision problem that needs fixing.”



“Your machine learning toolbox.”

Should be pretty full after the last week and a half.

“The typical way people from learning look at computer vision”

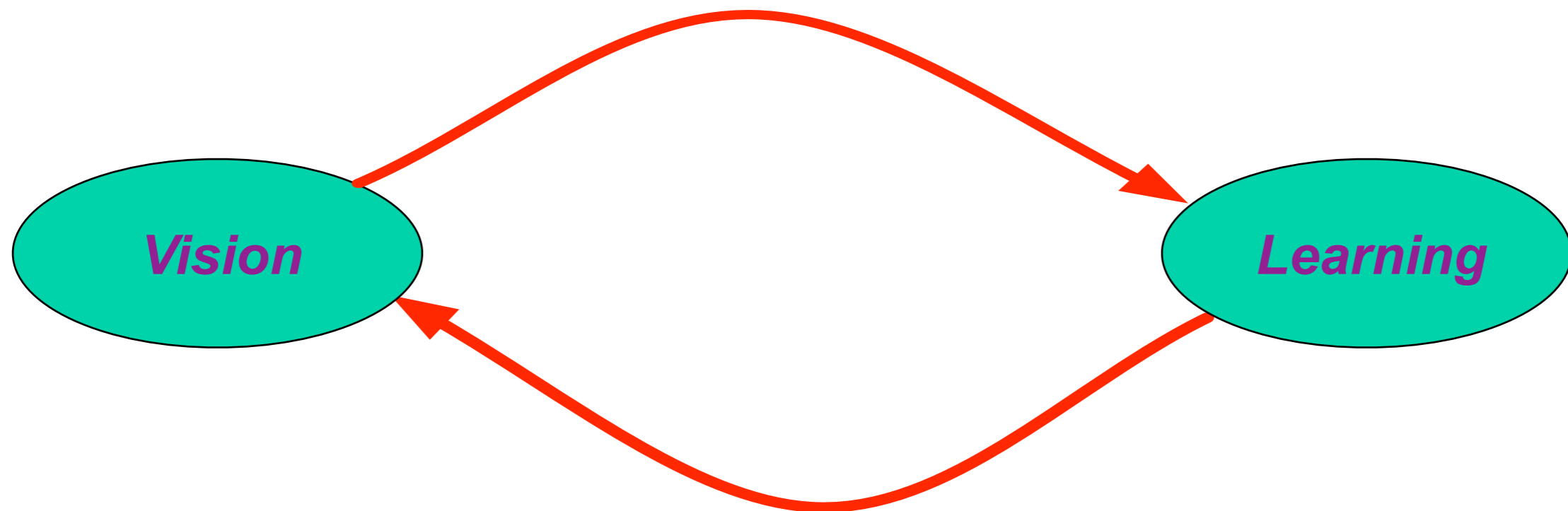
Vision and Learning



“The typical result”

Vision and Learning

“Vision specific constraints/assumptions.”



“Applied and evolved learning tools.”

“Applying learning to vision should be an evolutionary process.”

Vision and Learning



“Your result from this course”

Overall Course Outline

- Lecture 1

- Why registration and tracking is hard?
- Exhaustive Search (ES).
- Generative Models for ES Registration

- Lecture 2

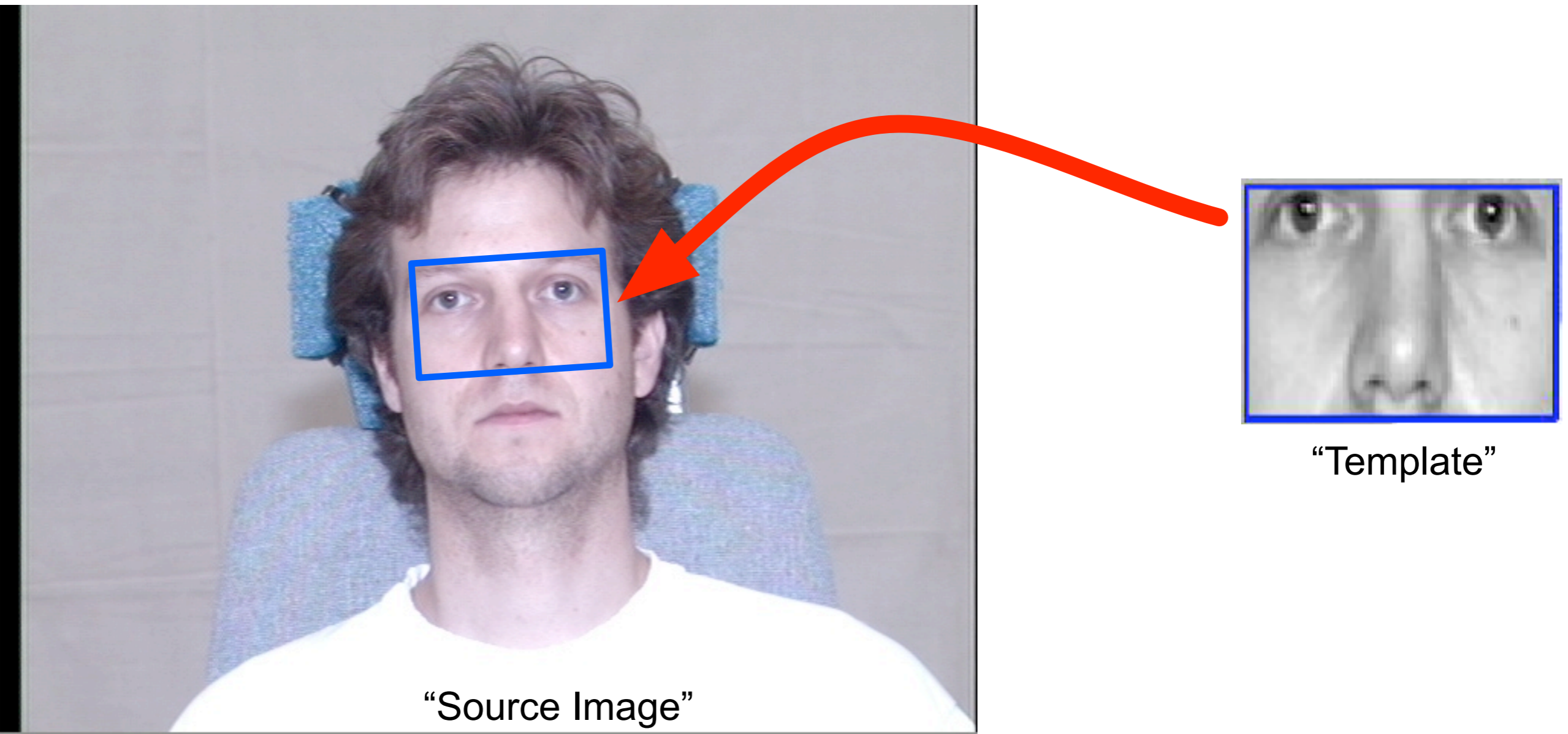
- Discriminative Models for ES Registration
- Efficient ES using FFT.
 - Correlation Filters.
 - Support Vector Machines
 - Neural Networks
- Efficient ES using integral images
 - Adaboost
 - Mutual Information
- Speed and performance comparisons.

Overall Course Outline

- Lecture 3
 - Gradient Search (GS)
 - Generative models for GS
 - Lucas-Kanade, Black-Jepson algorithms
 - Speeding up GS through Inverse Composition
- Lecture 4
 - Non-Rigid Warps
 - Discriminative models for GS
 - Support Vector Tracking (SVT)
 - Regression Search (RS)
 - Williams & Blake

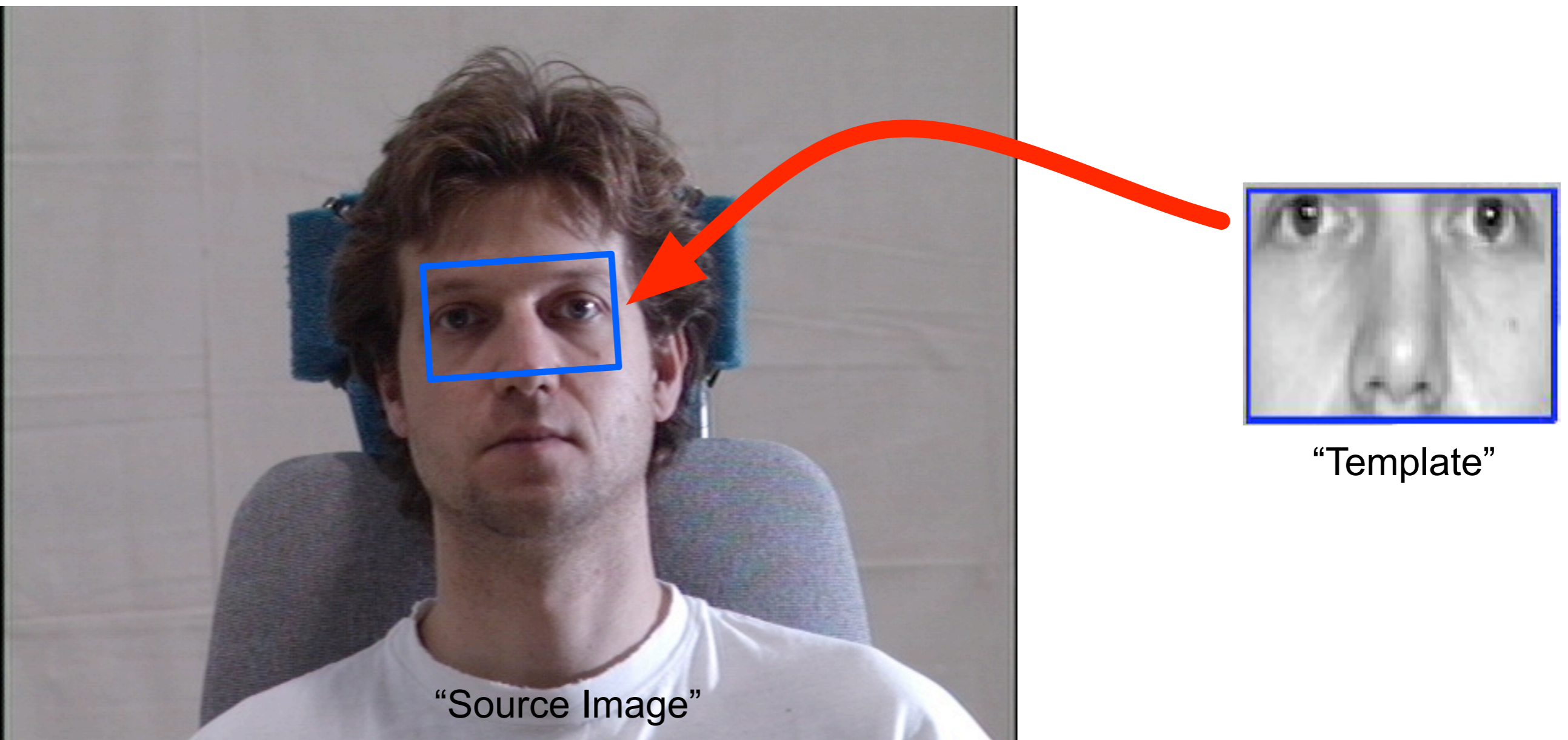
Task of Alignment/Registration

- Object registration is considered “high-level” computer vision.
- Task is to align a *template* with a given *source image*.



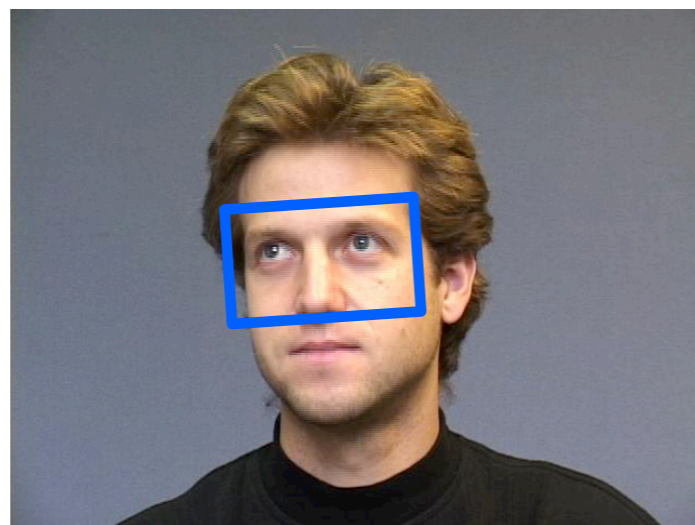
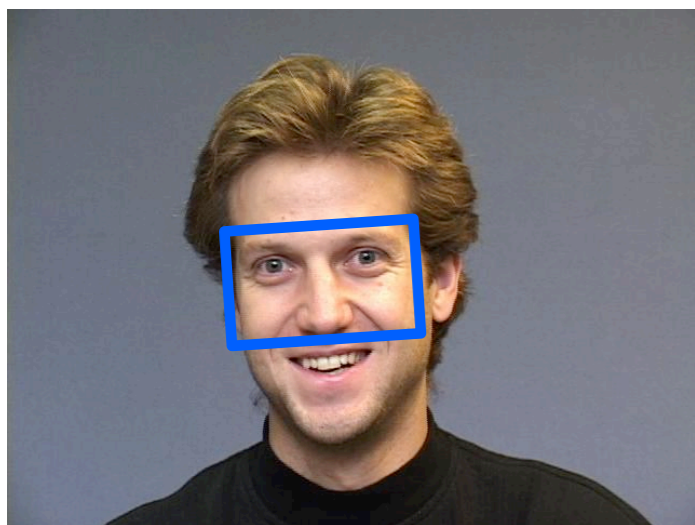
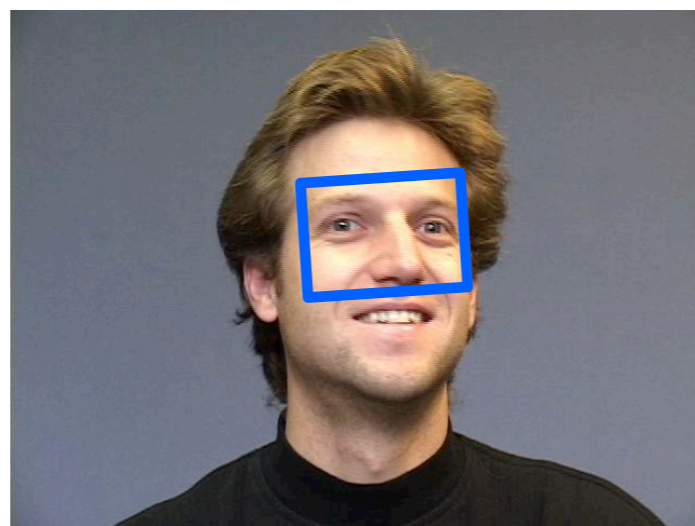
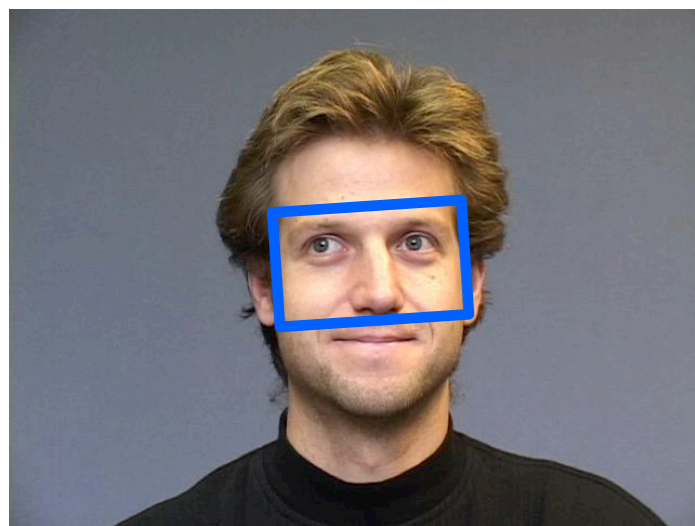
Why is it hard?

- Want to register an object even when *illumination* varies.

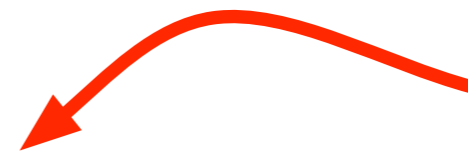


Why is it hard?

- Want to register an object even when *expression* and *pose* varies.



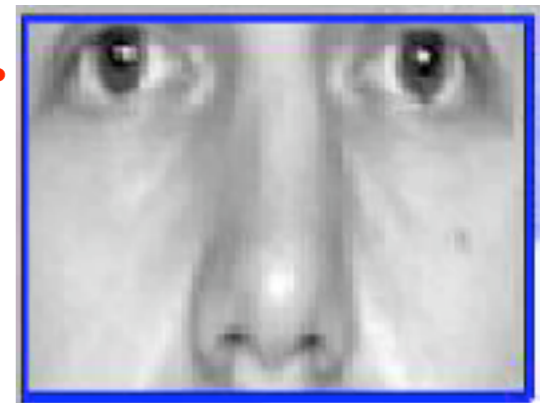
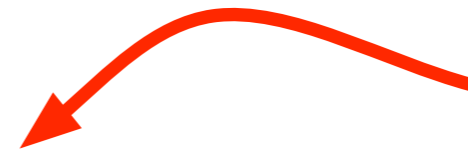
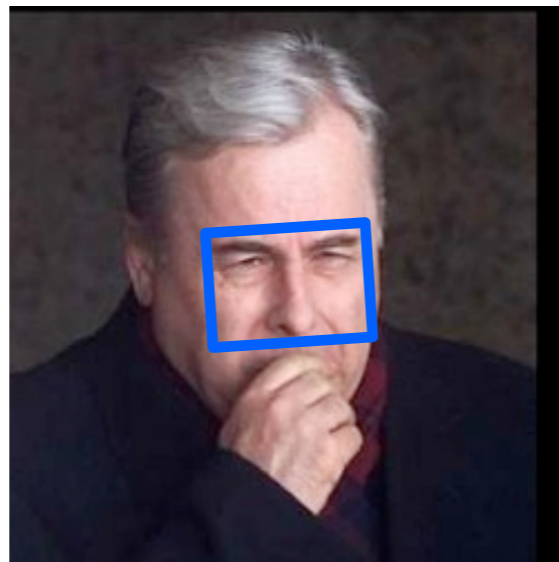
“Source Image”



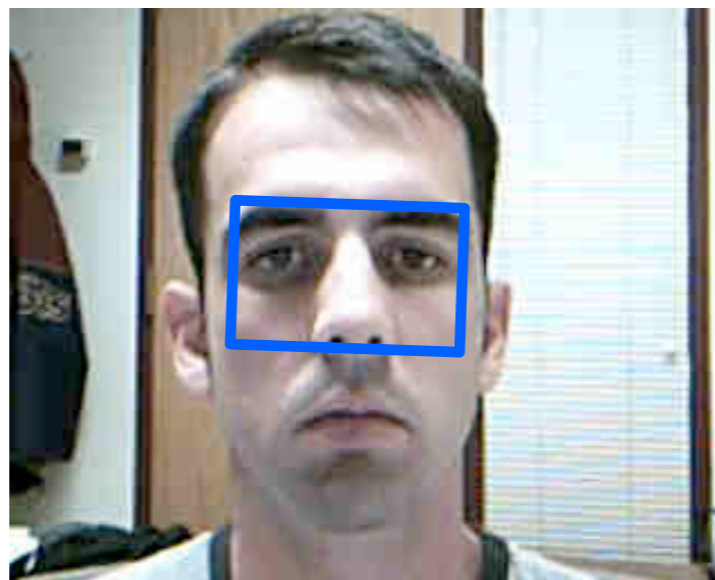
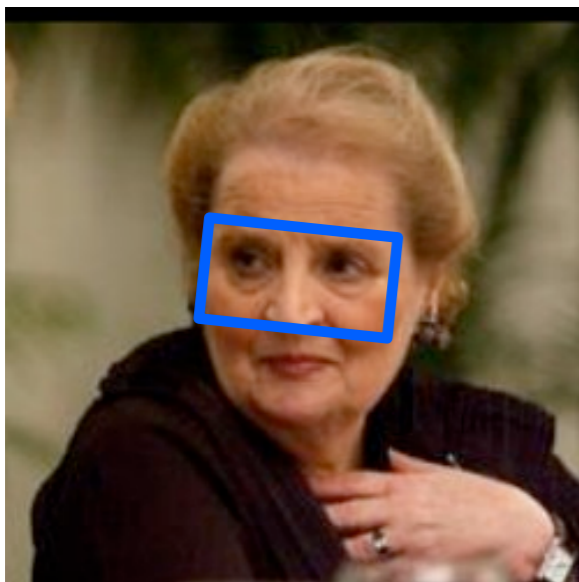
“Template”

Why is it hard?

- Want to register an object even when *identity* and *quality* varies.



“Template”



“Source Image”

Two Problems in Registration

1. *Learning*,

- How do I learn an object template/model that satisfactorily discriminates between the object and the image background?

(Will cover later....)

2. *Fitting*,

- How do I efficiently evaluate the object template/model across all possible warp values?

“As we will show through this course, both questions are linked!!!”

Challenge #1: Learning

- How can we deal with all the variations an object can undergo,

lighting



pose



occlusions



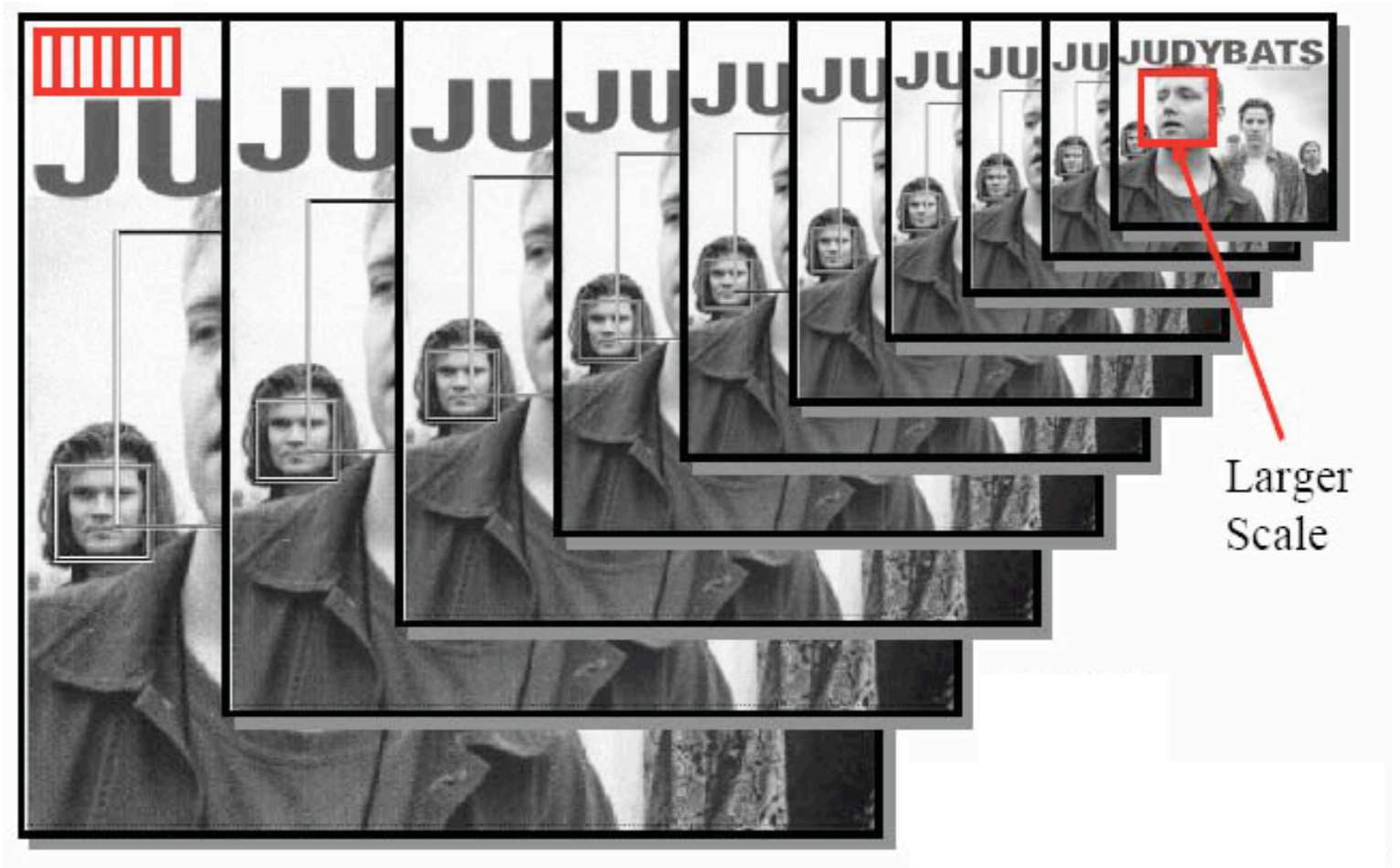
“nuisance” variables



Challenge #2: Fitting

- How can we efficiently fit the object model?

- 640×480 pixels
- scales 24, 36, 48, 60, 72
- sliding every 8 pixels



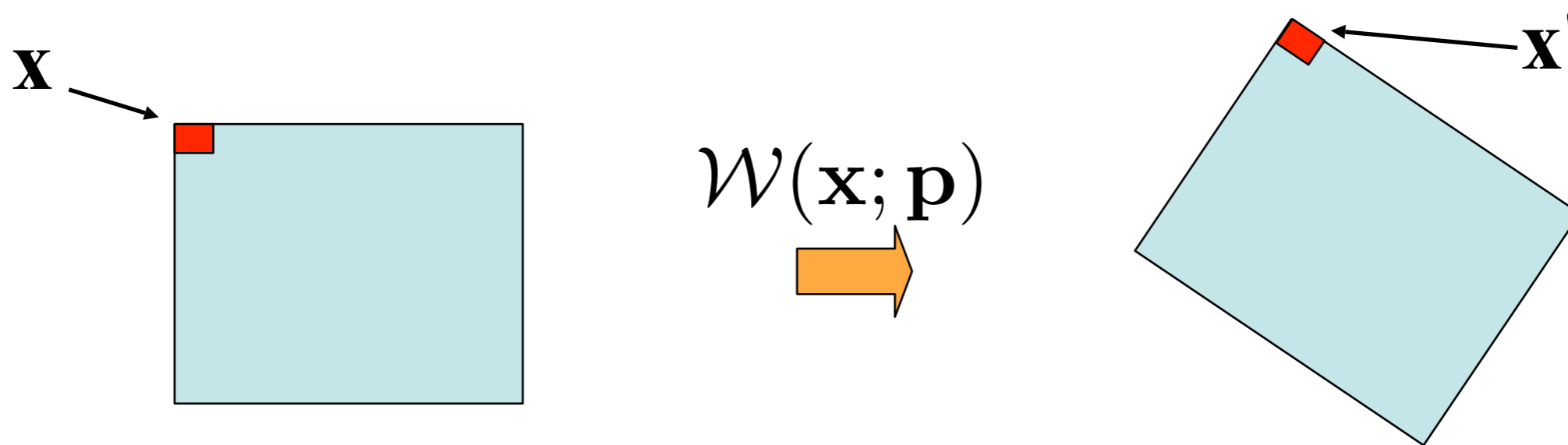
⇒ 24,000 subwindows to classify

Important Message

- The way one fits an object model/template drives the method used to learn the object model/template.
- Important to understand how fitting is performed first, before thinking about which classifier we use from our machine learning toolbox.
- A key understanding of the assumptions and constraints in vision is required before applying effective learning techniques.

Warp Functions

- To perform alignment we need a formal way of describing how the template relates to the source image.
- To do this we can employ what is known as a warp function:-



$$\mathbf{z} = [\mathbf{x}_1^T, \dots, \mathbf{x}_N^T]^T$$

where,

\mathbf{x}_i = i -th 2D coordinate

\mathbf{p} = parametric form of warp

\mathbf{z} = concatenation of all points in the template

Different Warp Functions



translation



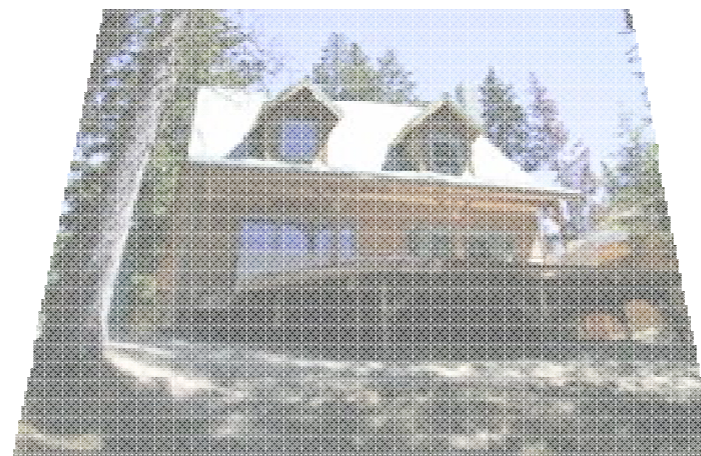
rotation



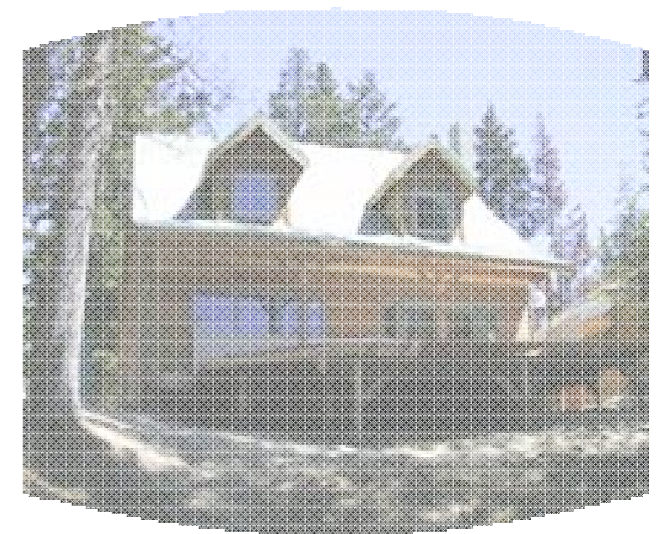
aspect



affine



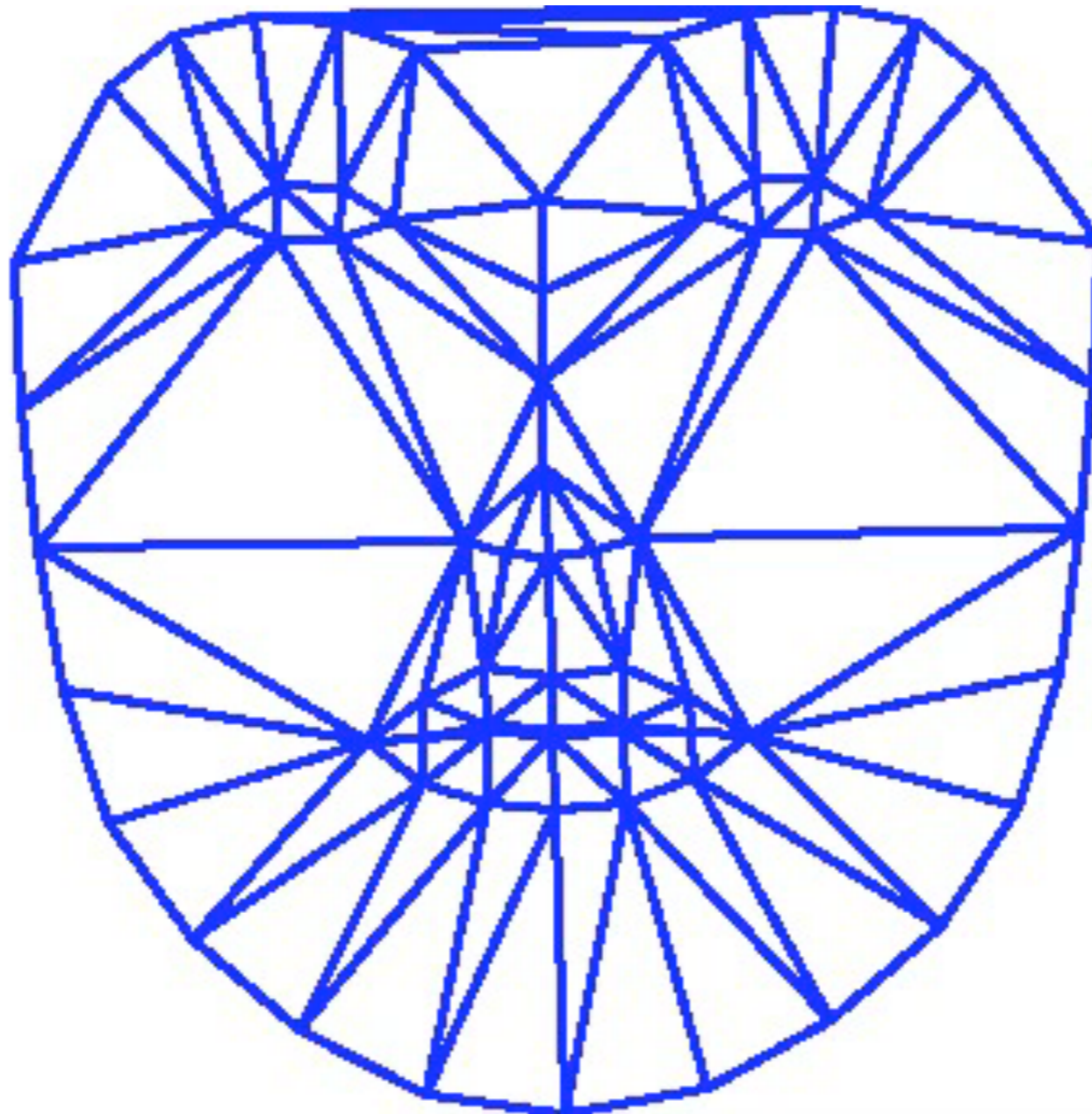
perspective



cylindrical

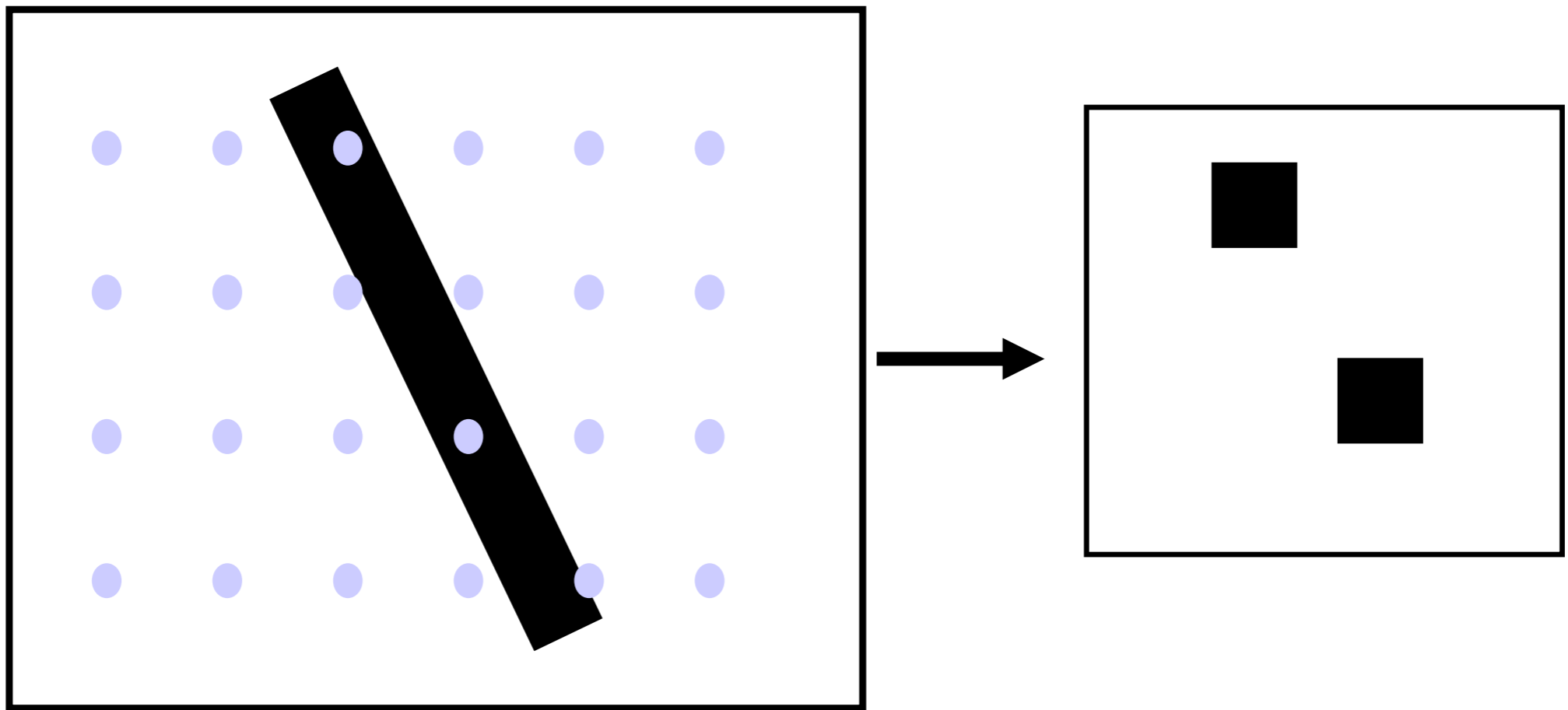
(Szeliski and Fleet)

Learnt Warps



Aliasing

- The warp function gives nearly always fractional output.
- But we can only deal in integer pixels.
- What happens if we need to subsample an image,



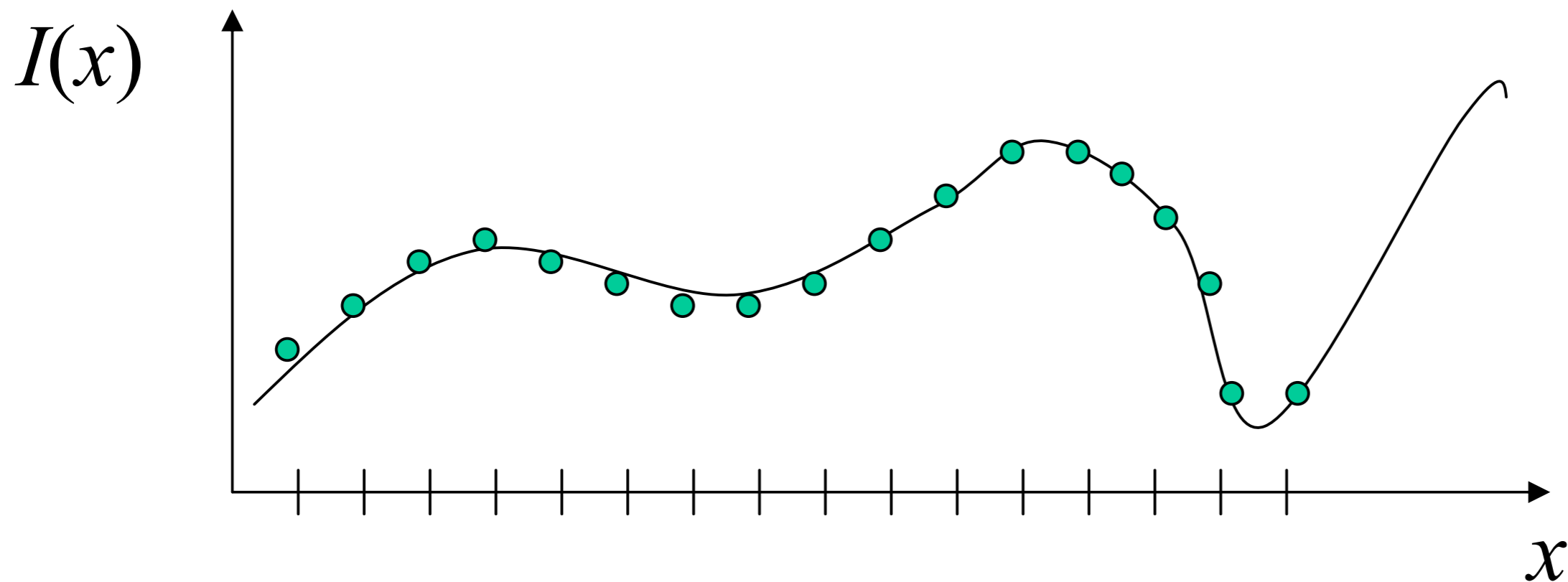
(Black)

Aliasing

- Can't shrink an image by taking every second pixel
- If we do, characteristic errors appear
- Common phenomenon
 - Wagon wheels rolling the wrong way in movies.

Image Interpolation

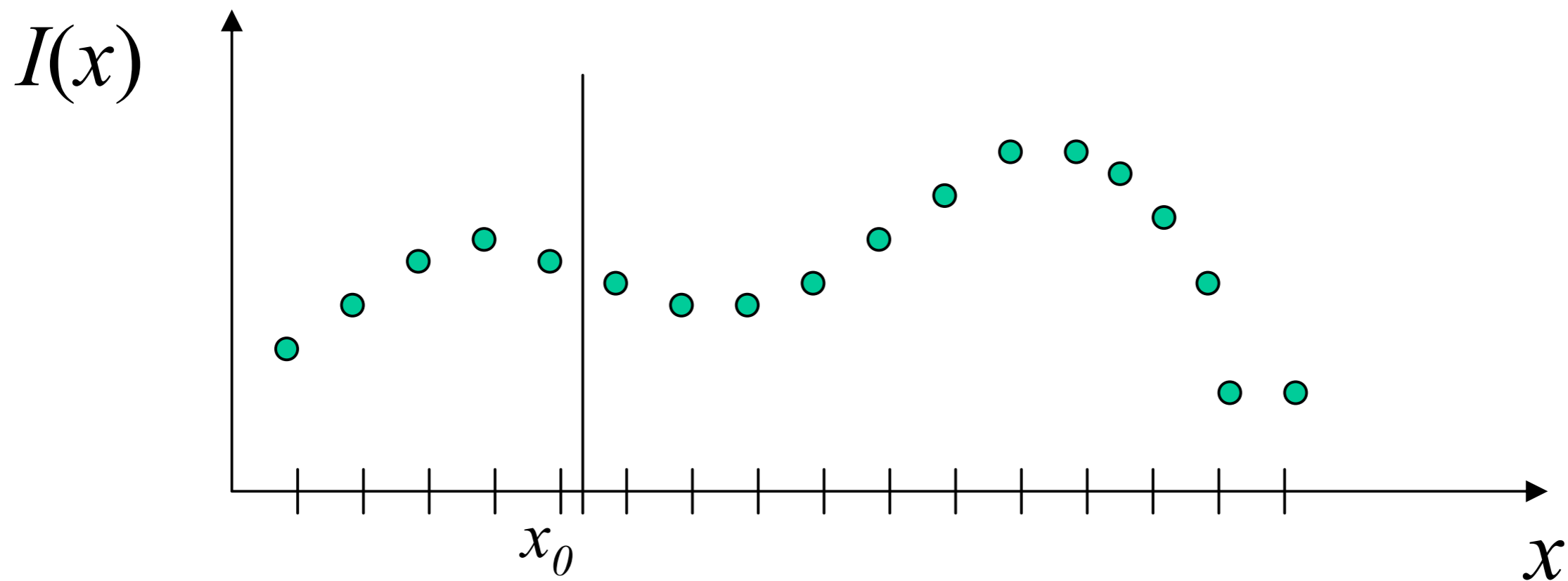
- Images are a discretely sampled representation of a continuous signal,



(Black)

Image Interpolation

- What if I want to know $I(x_0+dx)$ for small $dx < 1$?

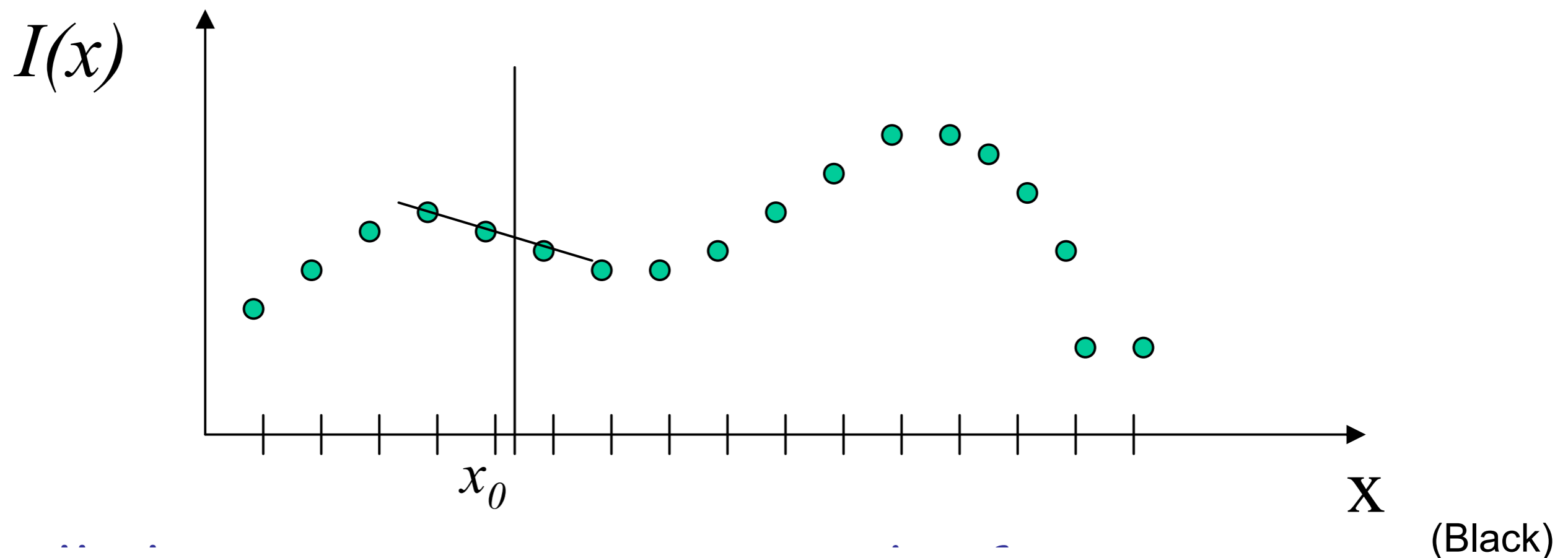


(Black)

Image Interpolation

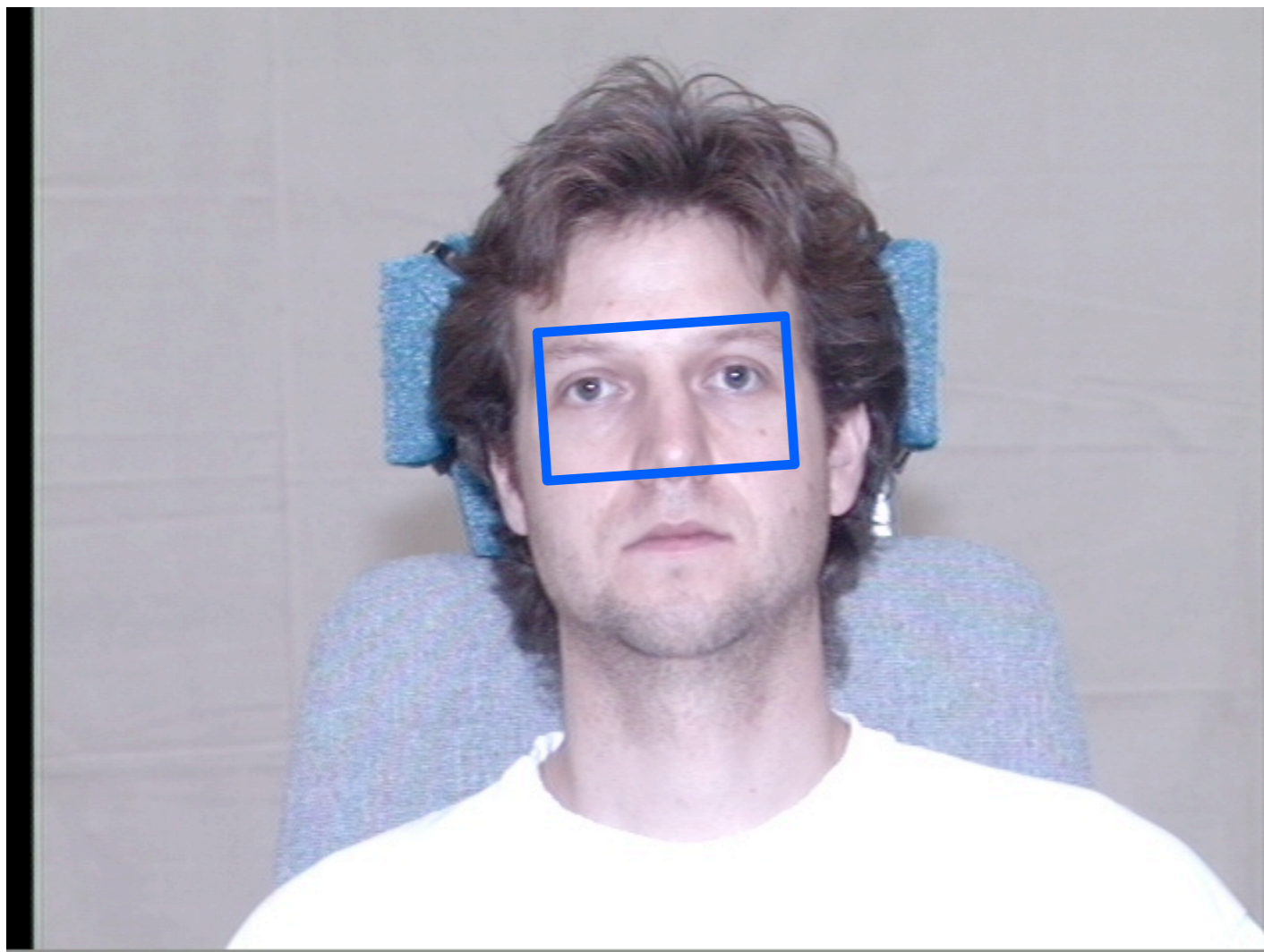
- Simply take the Taylor series approximation,

$$I(x_0 + dx) \approx I(x_0) + dx \frac{d}{dx} I(x_0) + \varepsilon$$

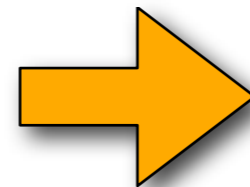


Naive Approach to Registration

- If you were a person coming straight from machine learning you might suggest,



“Images of Object at various warps”



[255,134,45,.....,34,12,124,67]
[123,244,12,.....,134,122,24,02]
[67,13,245,.....,112,51,92,181]
⋮
[65,09,67,.....,78,66,76,215]

“Vectors of pixel values at
each warp position”

Naive Approach to Registration

- If you were a person coming straight from machine learning you might suggest,

[255,134,45,.....,34,12,124,67]

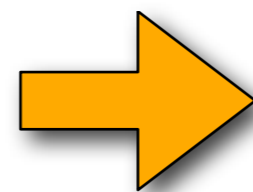
[123,244,12,.....,134,122,24,02]

[67,13,245,.....,112,51,92,181]

⋮

[65,09,67,.....,78,66,76,215]

“Vectors of pixel values at
each warp position”



$f(\text{teal box})$

“classifier”

$\begin{matrix} \text{object} \\ \geq \\ < \\ \text{background} \end{matrix} \quad Th$

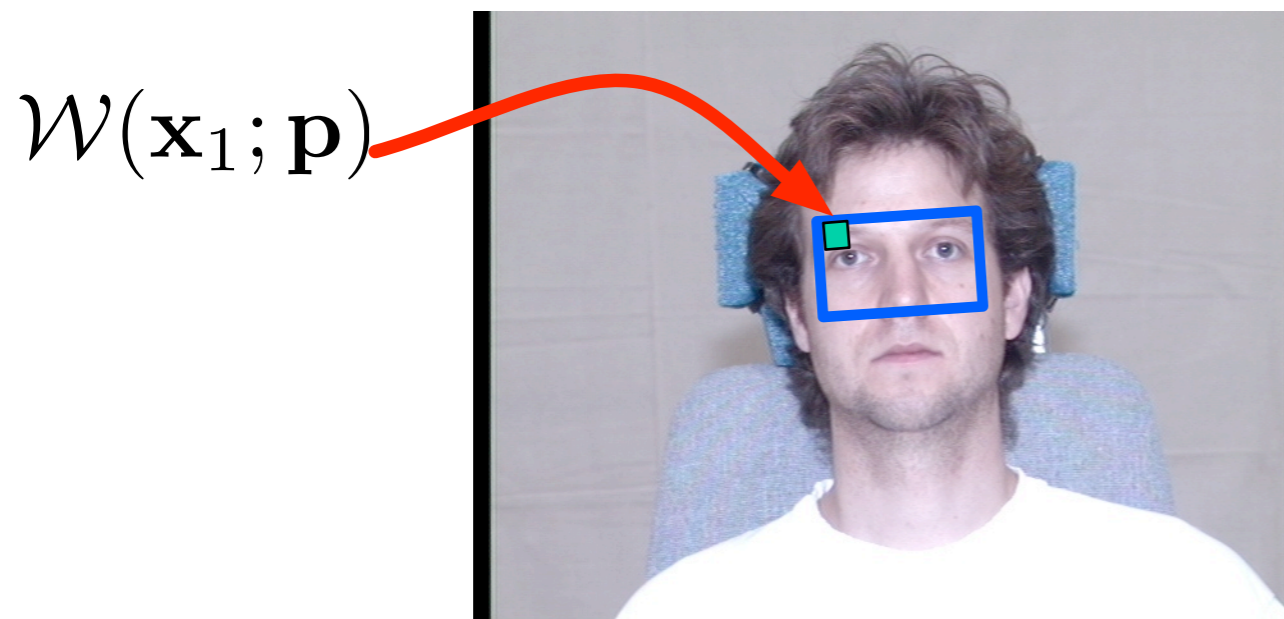
Naive Approach to Registration

- Problem,
 - Do we sample every warp parameter value?
 - Plausible if we are only doing translation?
 - If the image is high resolution, do we need to sample every pixel?
 - What happens if warps are more complicated (e.g., scale)?
 - Becomes prohibitively expensive computationally as warp dimensionality expands.

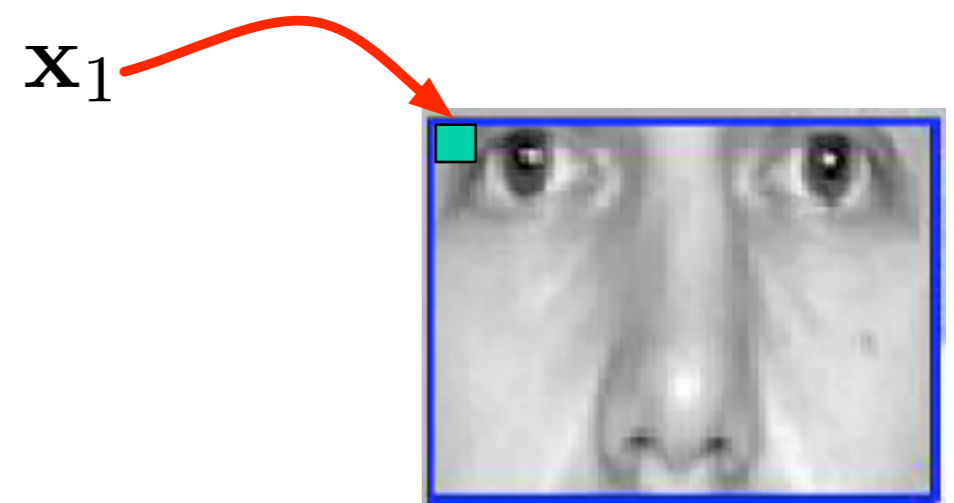
Measures of Image Similarity

- Although not always perfect, common measures of image similarity are:
 - Sum of squared differences (SSD)

$$SSD(\mathbf{p}) = \sum_{i=1}^N ||I(\mathcal{W}(\mathbf{x}_i; \mathbf{p})) - T(\mathbf{x}_i)||^2$$



I
“Source Image”

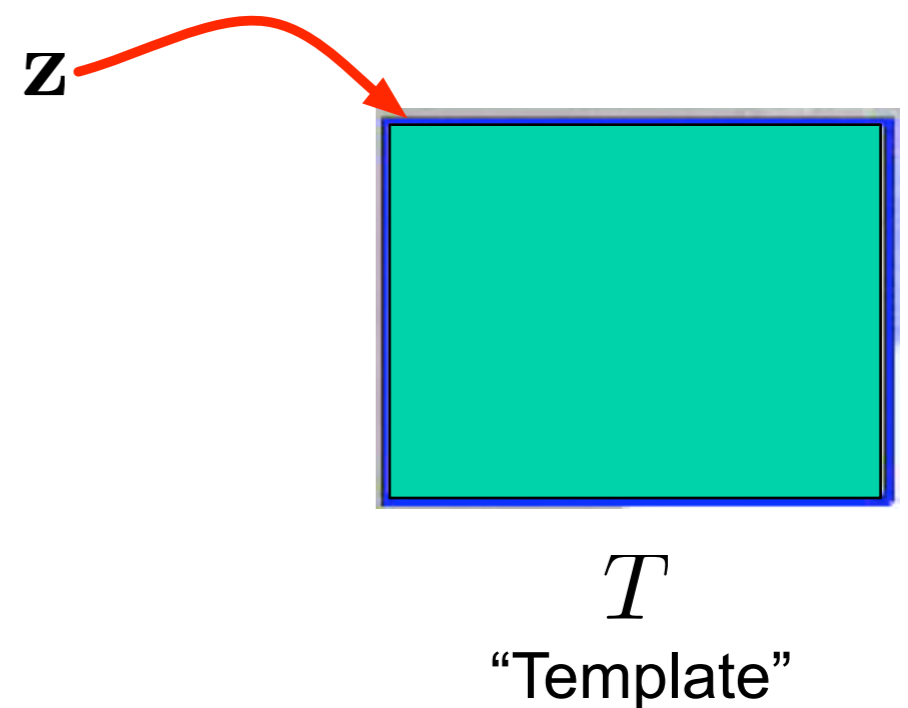
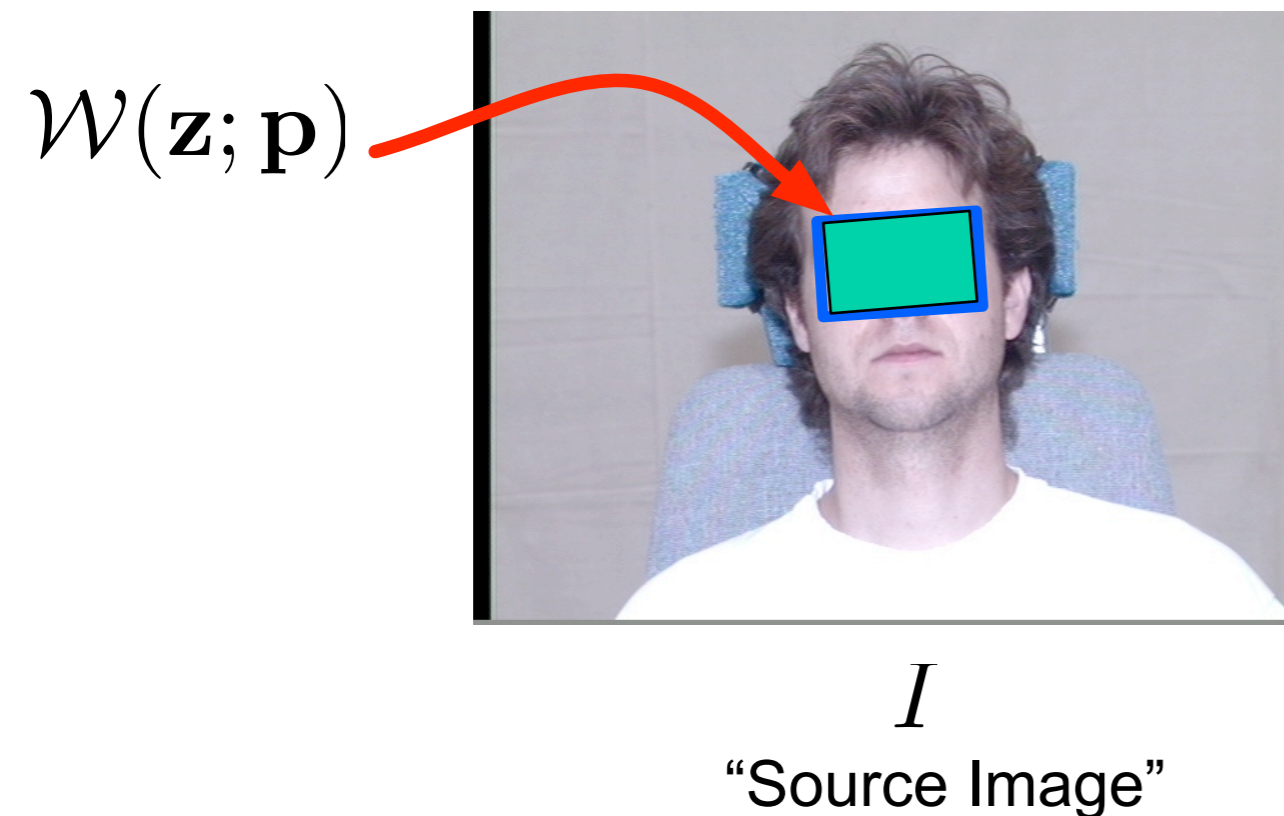


T
“Template”

Measures of Image Similarity

- Although not always perfect, common measures of image similarity are:
 - Sum of squared differences (SSD)

$$SSD(\mathbf{p}) = ||I(\mathcal{W}(\mathbf{z}; \mathbf{p})) - T(\mathbf{z})||^2 \quad \text{“Vector Form”}$$

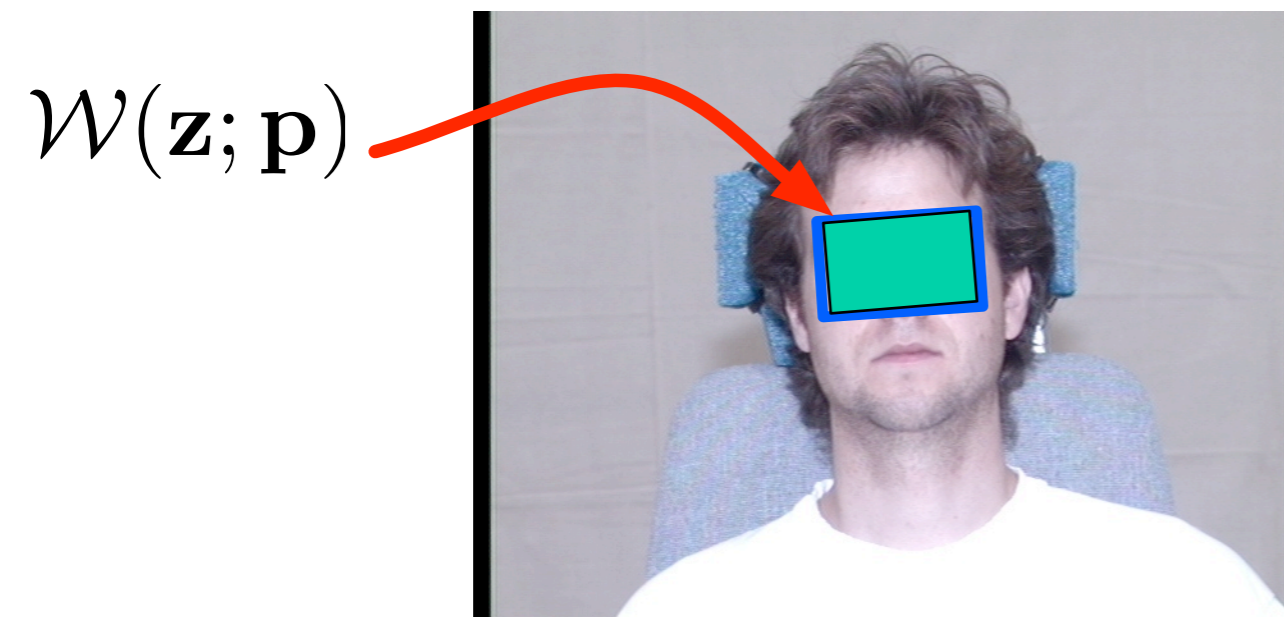


Measures of Image Similarity

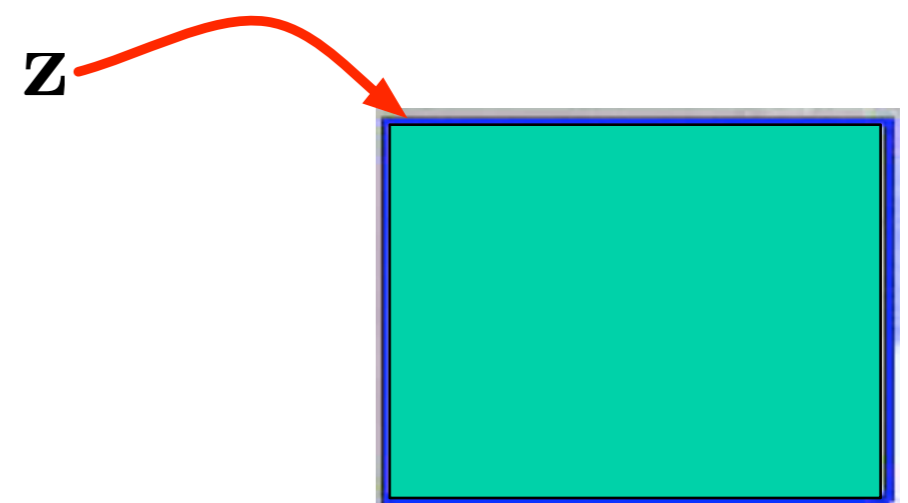
- Although not always perfect, common measures of image similarity are:
 - Dot product/Correlation (corr)

$$\text{corr}(\mathbf{p}) = I(\mathcal{W}(\mathbf{z}; \mathbf{p}))^T T(\mathbf{z})$$

“Vector Form”



I
“Source Image”



T
“Template”

Classifier Complexity

- General rule of thumb, is that for a classifier to be useful in registration/tracking it has to be **computationally** efficient.
- As a result we are generally restricted to classifiers that can be expressed in terms of correlation or SSD.

- Linear classifier,

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \quad \begin{array}{c} \text{object} \\ \geq \\ < \\ \text{background} \end{array} \quad Th$$

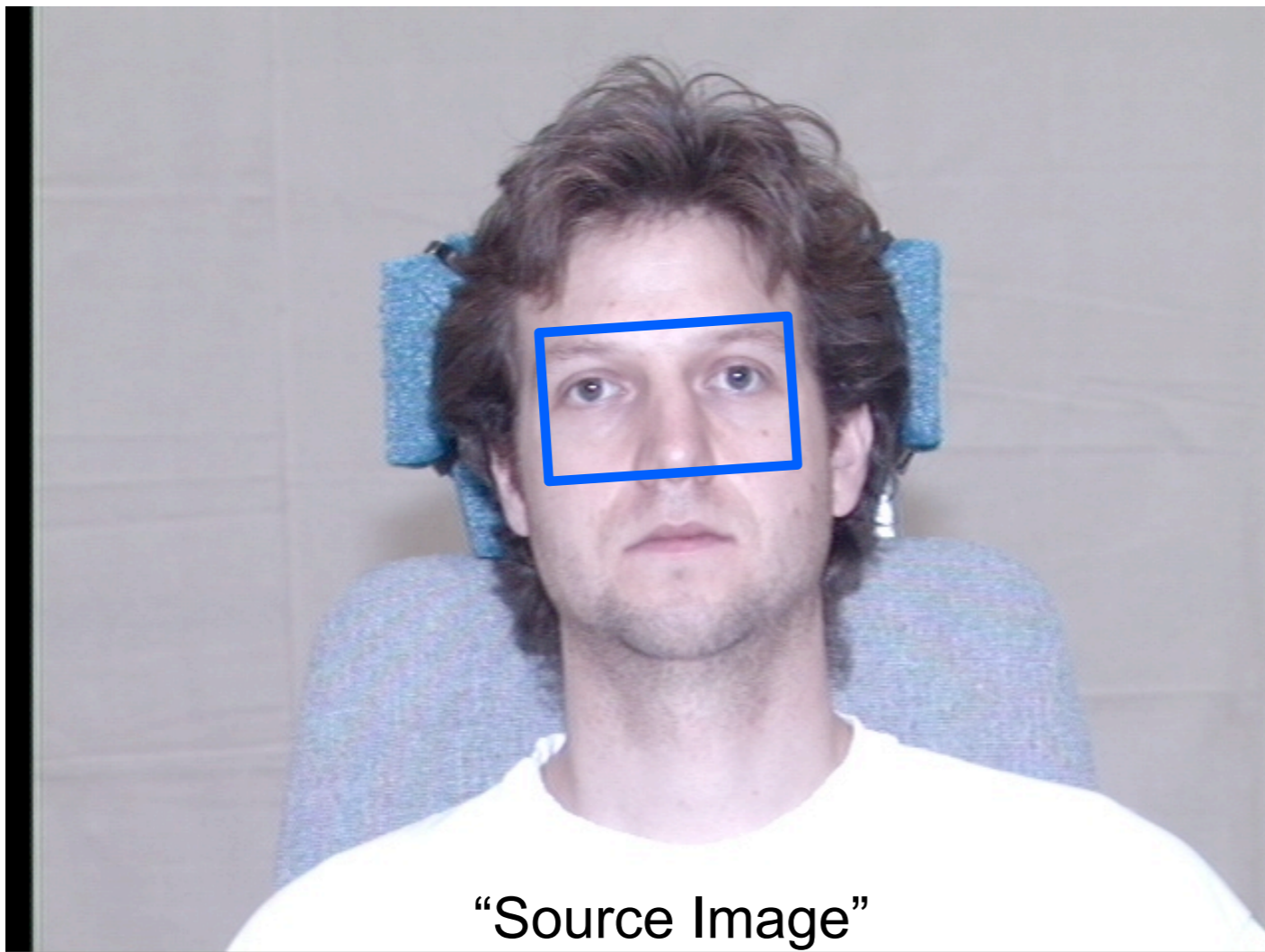
- Quadratic classifier,

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + c \quad \begin{array}{c} \text{object} \\ \geq \\ < \\ \text{background} \end{array} \quad Th$$

- Cascaded classifiers have been an exception to this rule.

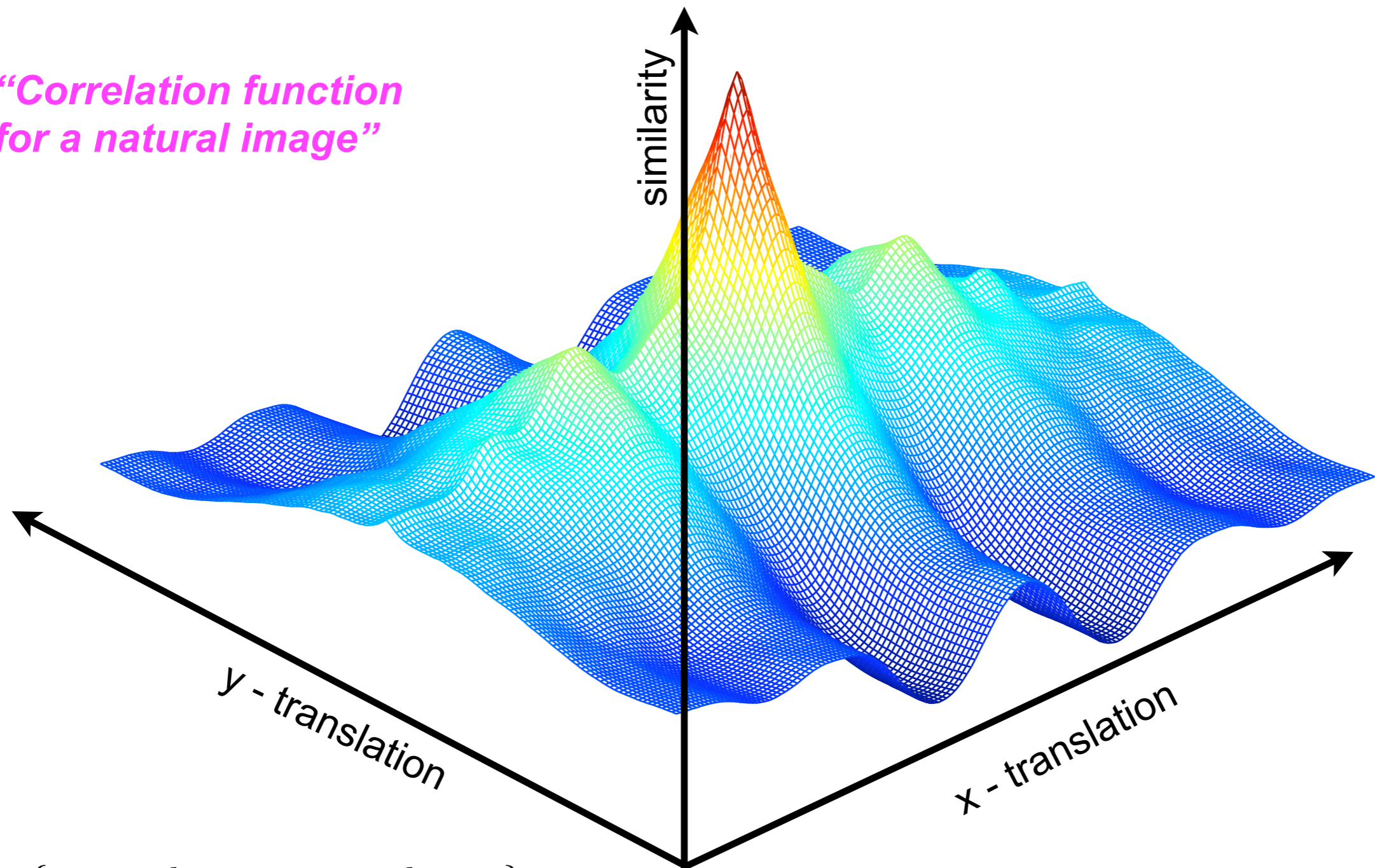
Pixel Coherence Assumption

- The pixel similarity between one warp position in an image and another warp position degrades gracefully as a function of distance.



Pixel Coherence Assumption

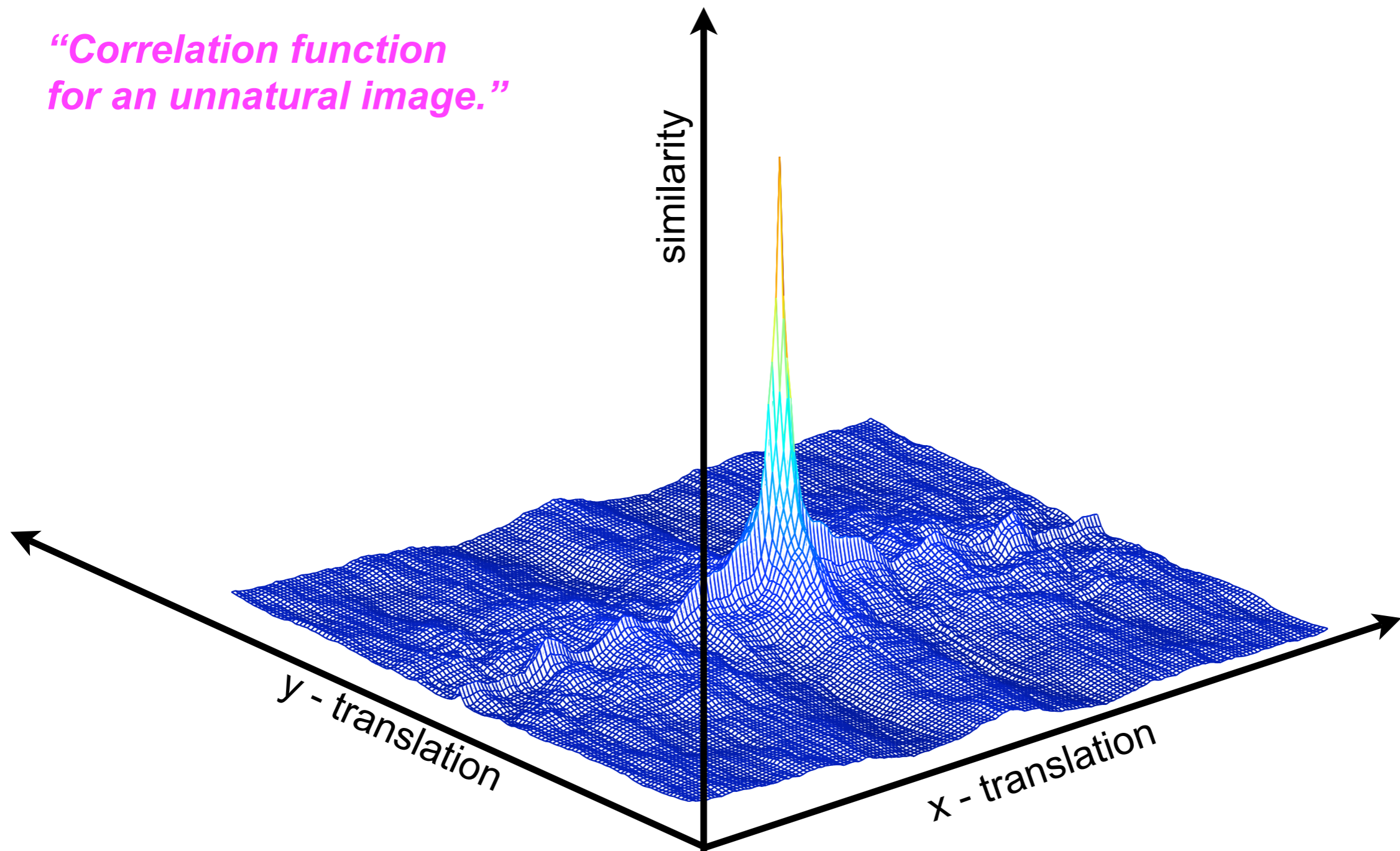
*“Correlation function
for a natural image”*



$$\mathbf{p} = \{\text{x-translation}, \text{y-translation}\}$$

Pixel Coherence Assumption

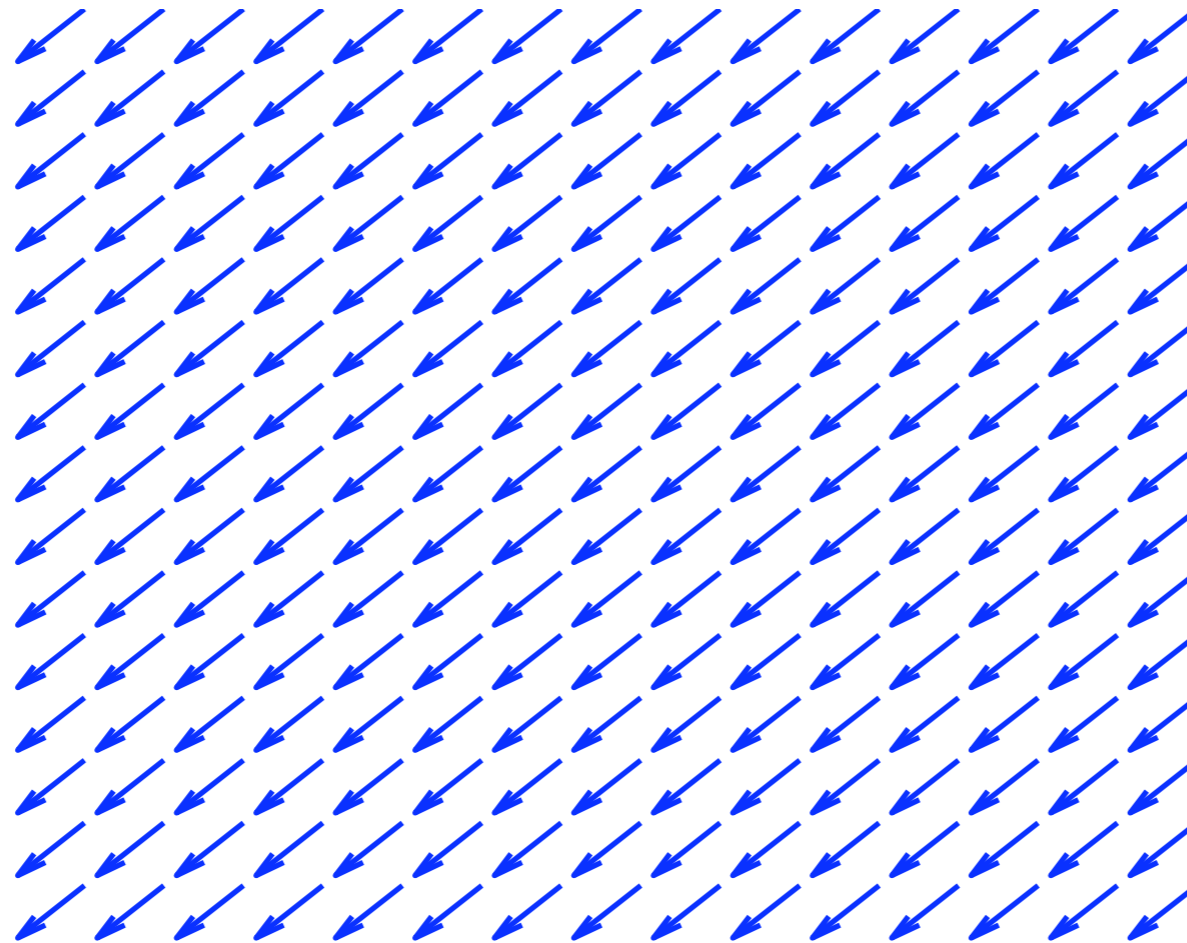
*“Correlation function
for an unnatural image.”*



$$\mathbf{p} = \{\text{x-translation}, \text{y-translation}\}$$

Pixel Coherence Assumption

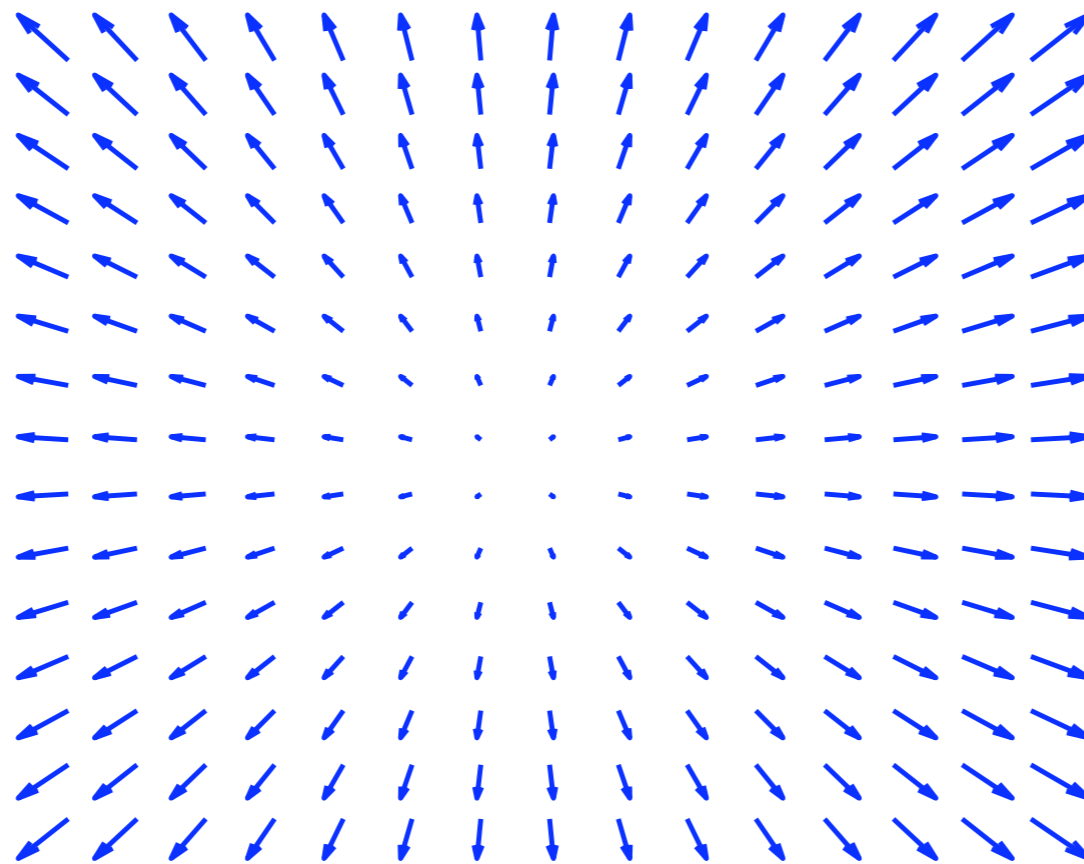
- Is this pixel coherence assumption good for only translation?



“Motion Field for Translation”

Pixel Coherence Assumption

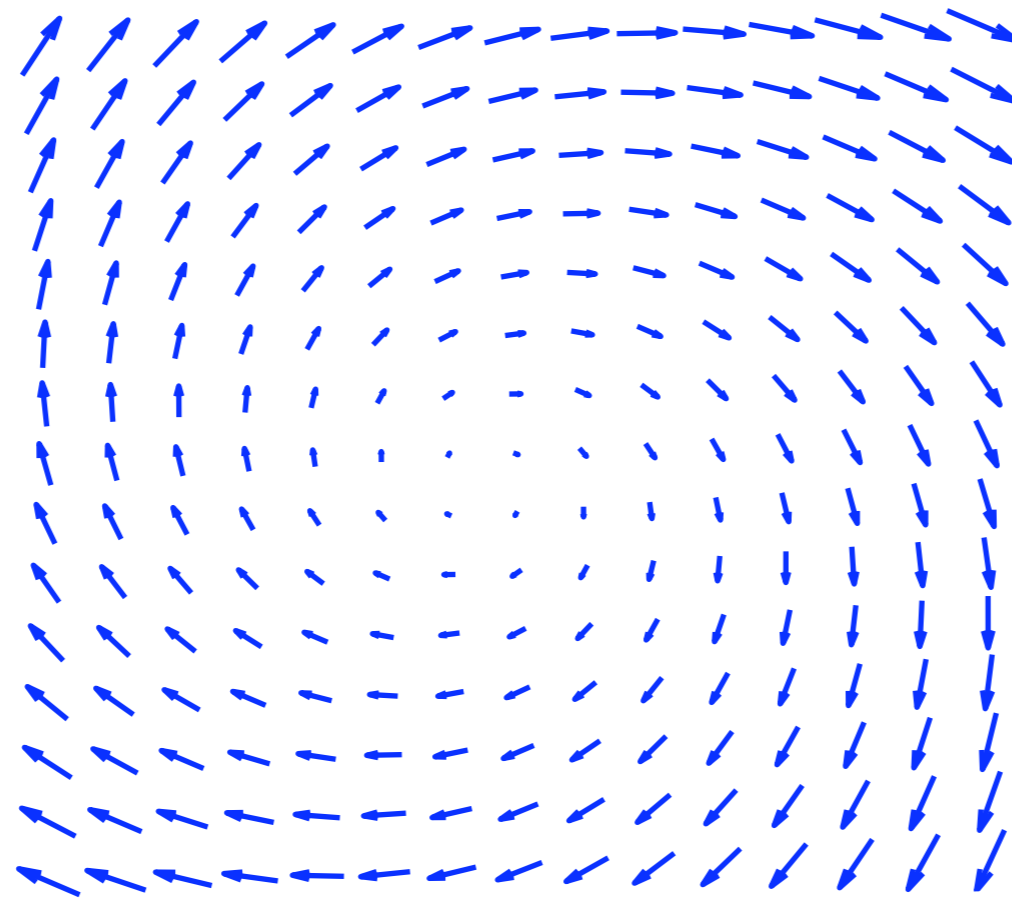
- Is this pixel coherence assumption good for only translation?
- No,



“Motion Field for Scale”

Pixel Coherence Assumption

- Is this pixel coherence assumption good for only translation?
- No,



“Motion Field for Rotation”

Pixel Coherence Assumption

- Is this pixel coherence assumption good for only translation?
- No, all warps can be interpreted as translation warps at each pixel within the template.
- Therefore, pixel coherence assumption is useful across all warps.

Pixel Coherence Assumption

- Pixels within natural images are heavily correlated within a spatially coherent region.
- Images do not have to exactly match in translation or scale to give a good similarity score.

$$\begin{array}{|c|} \hline I \\ \hline \end{array} \begin{array}{|c|} \hline I \\ \hline \end{array} - \begin{array}{|c|} \hline I \\ \hline \end{array} \begin{array}{|c|} \hline I \\ \hline \end{array}^2 \begin{array}{|c|} \hline \text{object} \\ \hline \end{array} \begin{array}{|c|} \hline \leq \\ \hline \end{array} \begin{array}{|c|} \hline > \\ \hline \end{array} \begin{array}{|c|} \hline \text{background} \\ \hline \end{array} Th$$

Pixel Coherence Assumption

- Pixels within natural images are heavily correlated within a spatially coherent region.
- Images do not have to exactly match in translation or scale to give a good similarity score.

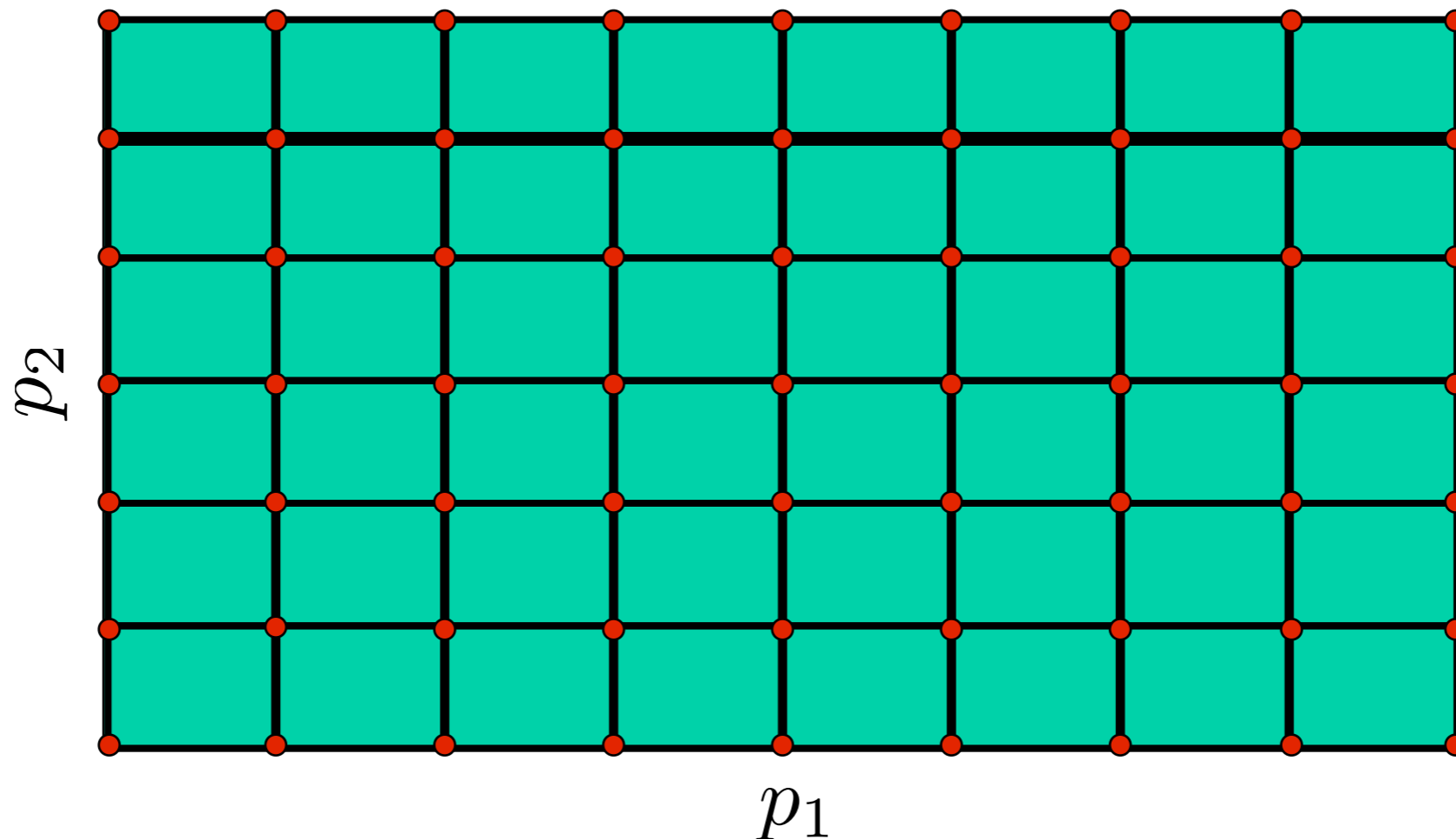
$$\left\| \begin{array}{c} \text{Image 1} \\ \text{Image 2} \end{array} \right\|_2^2 \leq Th$$

The diagram illustrates the Pixel Coherence Assumption. It shows two grayscale images, one of a nose and one of eyes, separated by a minus sign. To the right, a large '2' is shown above a less-than-or-equal-to symbol, which is followed by 'Th'. Below the 'Th' is the word 'background' circled in orange.

Can only go so far!!!

Exhaustive Search Strategy

- Based on this assumption we can do a reasonable job by discretely sampling the warp parameter space (typically translation & scale).



$$\mathbf{p} = \{p_1, p_2\}$$

“Discrete Warp Space”

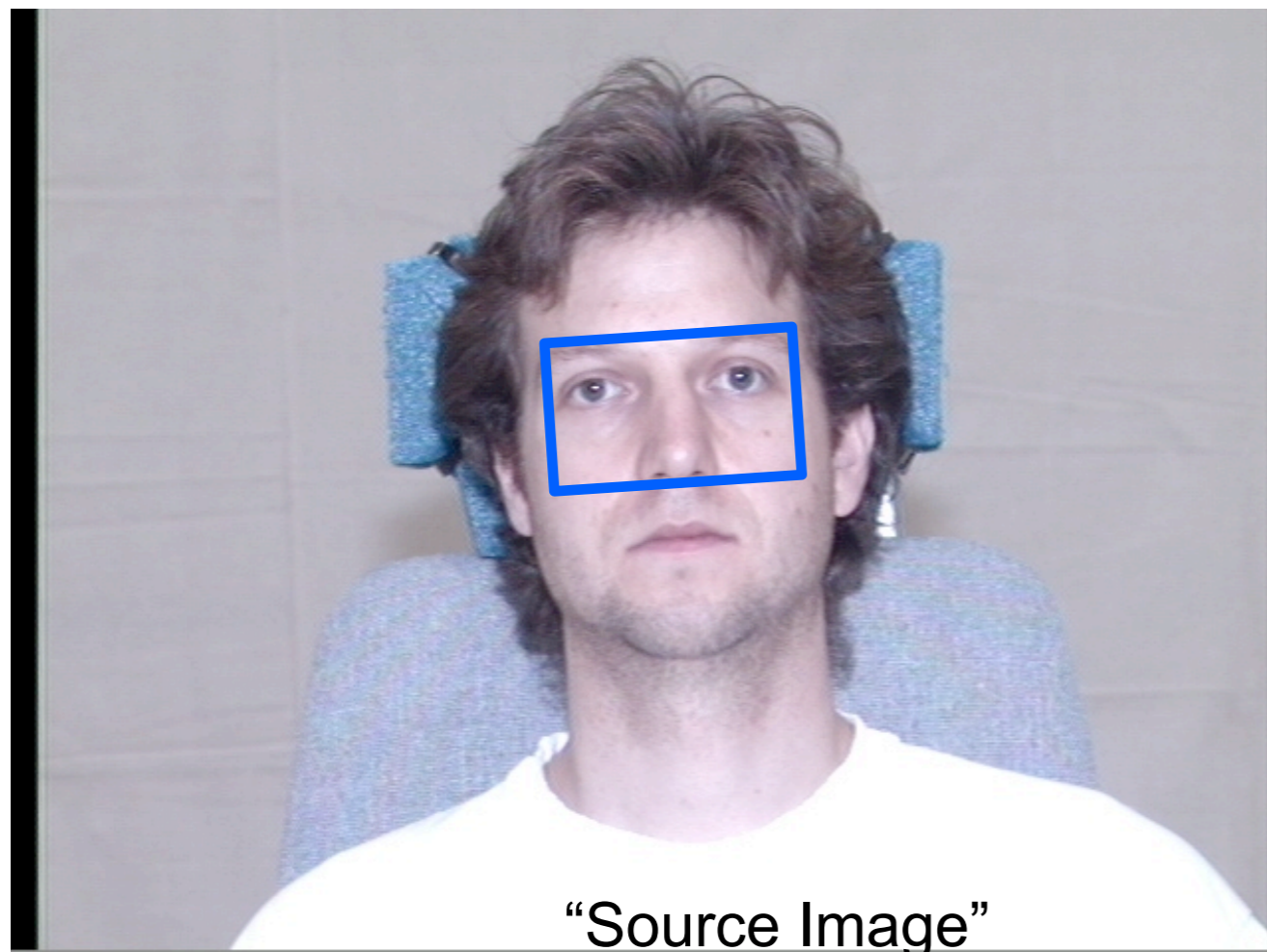
Exhaustive Search Strategy

- Based on this assumption we can do a reasonable job by discretely sampling the warp parameter space (typically translation & scale).
- Shall refer to this now on as Exhaustive Search (ES).



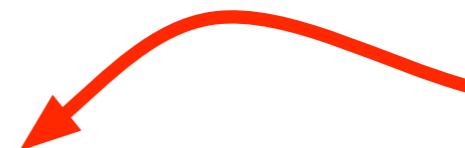
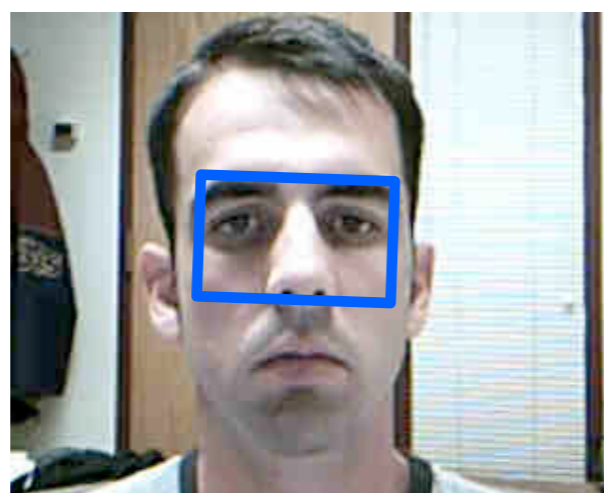
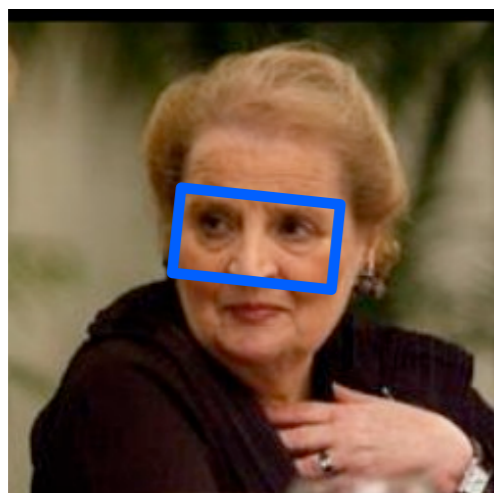
Problem....

- SSD and correlation measures are good if the template stems from the source image.



Problem....

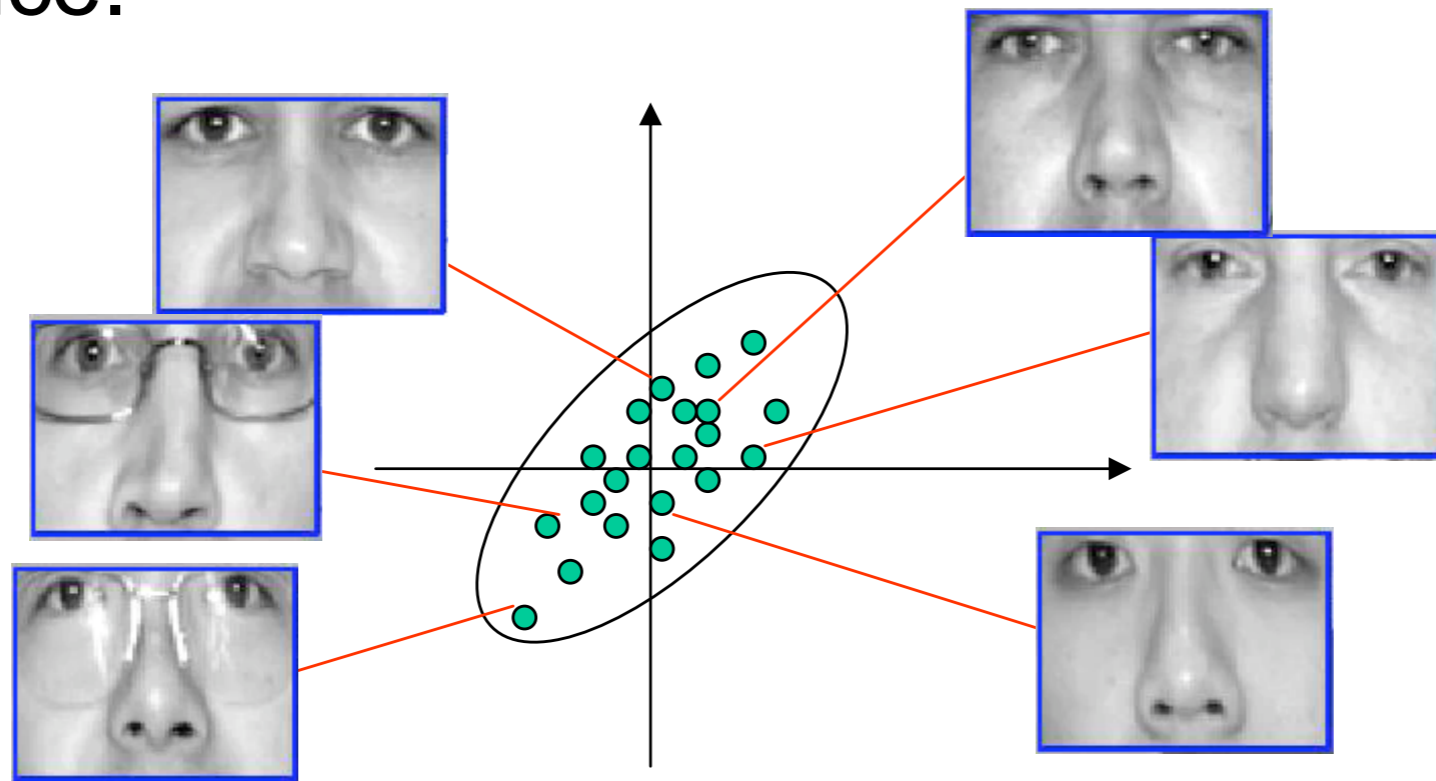
- SSD and correlation measures are good if the template stems from the source image.
- Not so good if there is noise stemming from appearance variation, e.g.:



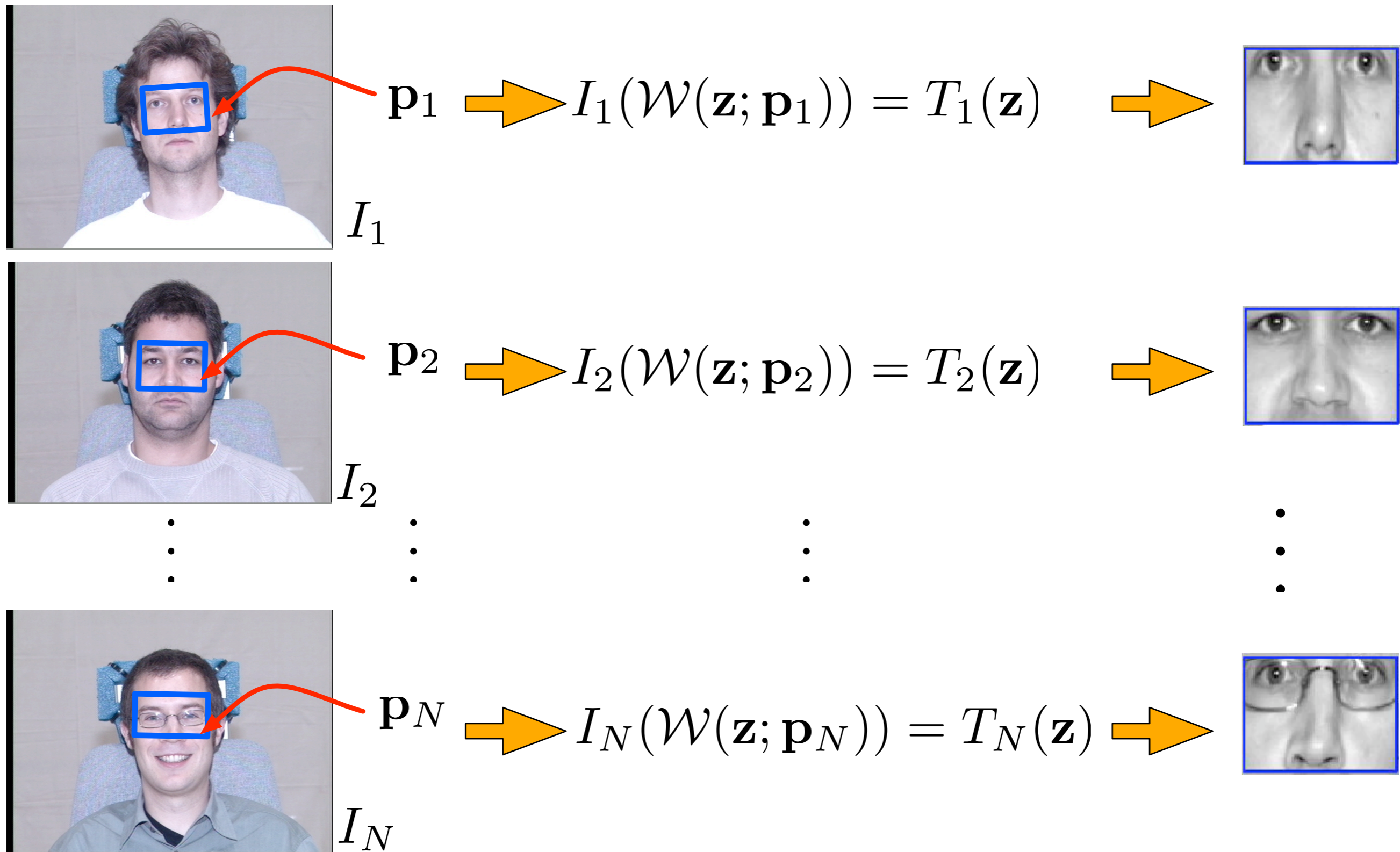
“Template”

Eigen-Objects

- Is there a way to learn an object model that can handle these appearance variations?
- Turk and Pentland (1991) proposed a method they referred to as “Eigenfaces” that could handle appearance variation.
- The technique employed principal component analysis (PCA) to model how a registered object could vary in appearance.



Training Examples



Eigen-Objects

- First, we concatenate all the training objects into a large vector.

$$\mathbf{C} = [T_1(\mathbf{z}), T_2(\mathbf{z}), \dots, T_N(\mathbf{z})]$$

$$= \begin{array}{c} \text{[face image]} \quad \text{[face image]} \quad \dots \quad \text{[face image]} \end{array}$$

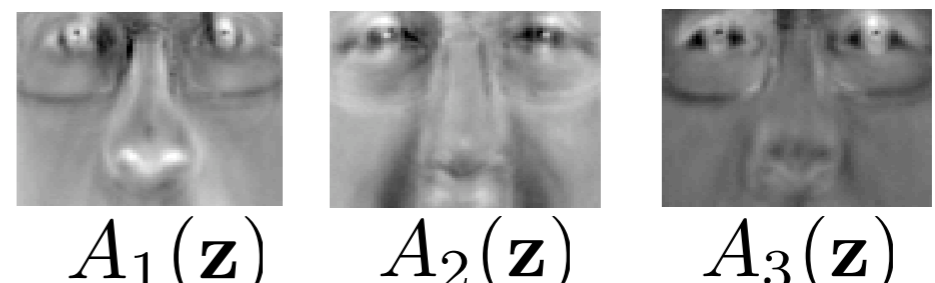
- Then we remove the mean,

$$\mathbf{C} \rightarrow \begin{array}{c} \text{[face image]} \quad \text{[face image]} \quad \dots \quad \text{[face image]} \end{array} - \begin{array}{c} \text{[mean face image]} \end{array} \quad \leftarrow \begin{array}{l} \text{Ensemble mean} \\ A_0(\mathbf{z}) \end{array}$$

- Then we apply PCA.

$$\mathbf{C}^T \mathbf{C} = \mathbf{A} \mathbf{\Lambda} \mathbf{A}^T = [\mathbf{a}_1, \dots, \mathbf{a}_M] \text{diag} [\sigma_1, \dots, \sigma_M] [\mathbf{a}_1, \dots, \mathbf{a}_M]^T$$

First three eigenvectors \rightarrow



Eigen-Objects

- We can use conventional SSD to then gain the match at a particular warp by minimizing,

$$SSD(\mathbf{p}, \boldsymbol{\lambda}) = ||I(\mathcal{W}(\mathbf{z}; \mathbf{p})) - A_0(\mathbf{z}) - \sum_{m=1}^M A_m(\mathbf{z})||^2$$

- Interestingly, we can “project out” the appearance change so that we are just minimizing,

$$DFFS(\mathbf{p}) = ||I(\mathcal{W}(\mathbf{z}; \mathbf{p})) - A_0(\mathbf{z})||_{\text{null}(\mathbf{A})}^2$$

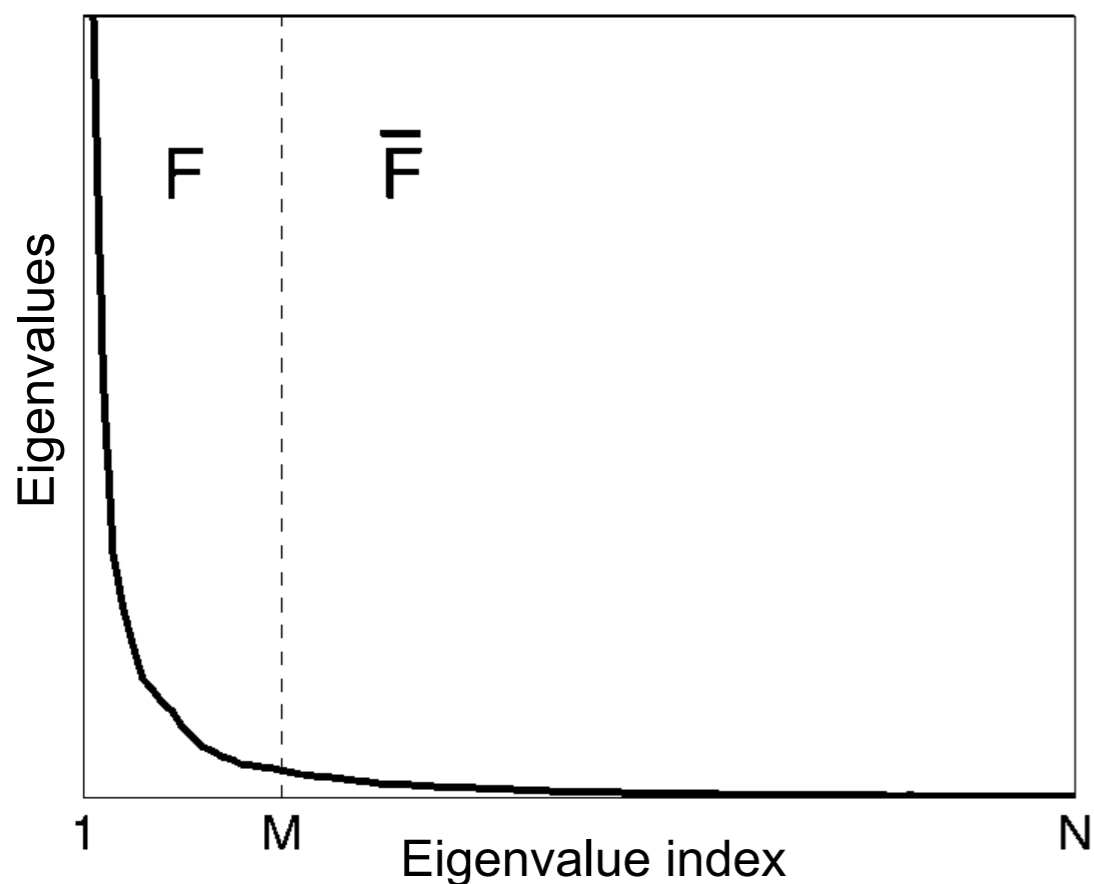
- where,

$$||\mathbf{u}||_{\text{null}(\mathbf{A})}^2 = \mathbf{u}^T \mathbf{u} - \mathbf{u}^T \mathbf{A} \mathbf{A}^T \mathbf{u}$$

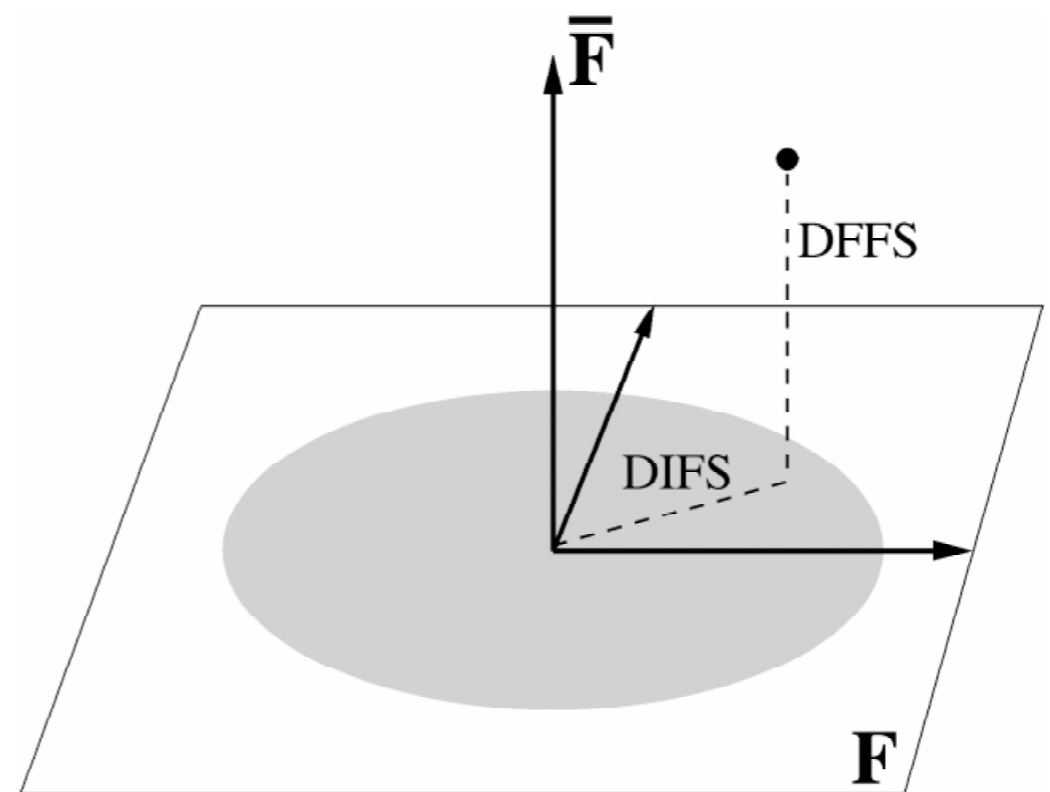
“Sometimes referred to as the distance from feature space (DFFS)”

DFFS Interpretation

- PCA is useful for assuming the object stems from a low-dimensional manifold.
- Choose $M < N$ eigenvectors for better “generalization”.
- Assume $N-M$ other eigenvectors stem from sample noise.

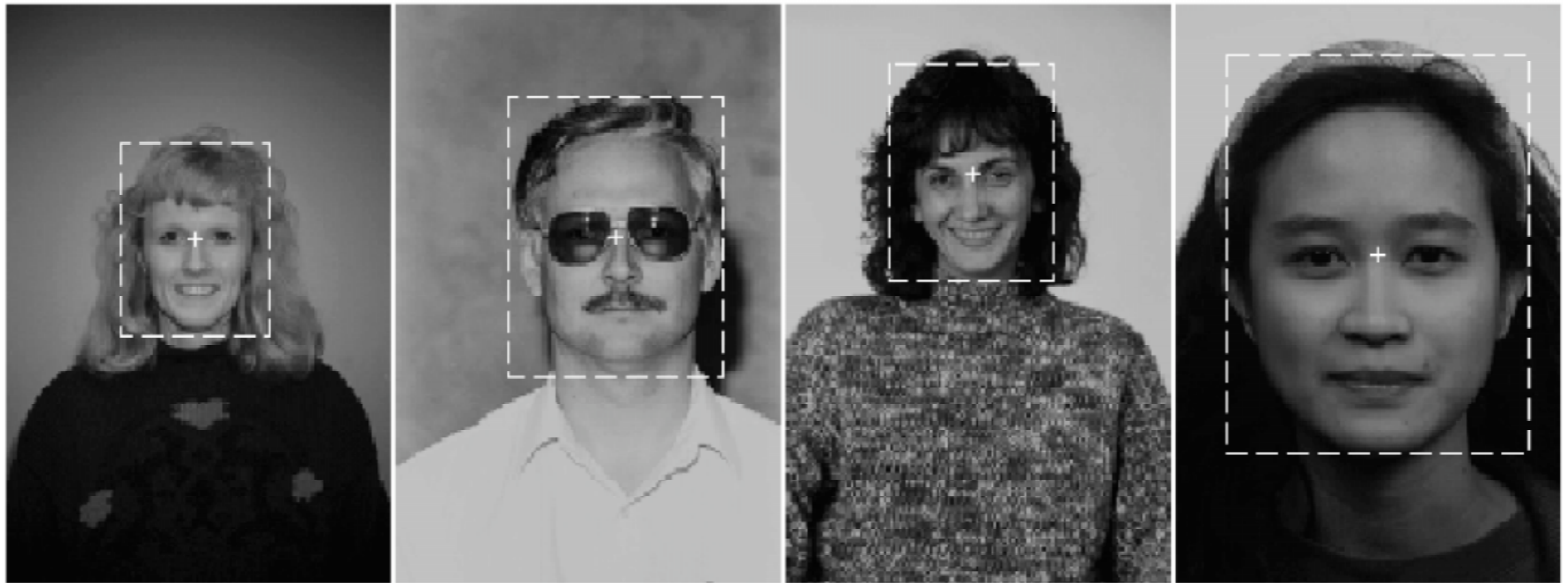


“Eigenspectrum”



“Visualization”
(Moghaddam & Pentland)

DFFS Examples of Performance



(Moghaddam and Pentland)

Eigen-Objects = Generative

- Approach is inherently a generative approach:
 - Learn how to synthesize a registered object's appearance variation using a linear model. (i.e. PCA)
 - Invert model to gain a measure of similarity. (i.e., DFFS).
- Unfortunately, like many generative models this approach has poor generalization properties.
 - What happens if my model cannot synthesize the registered object in a given source image?



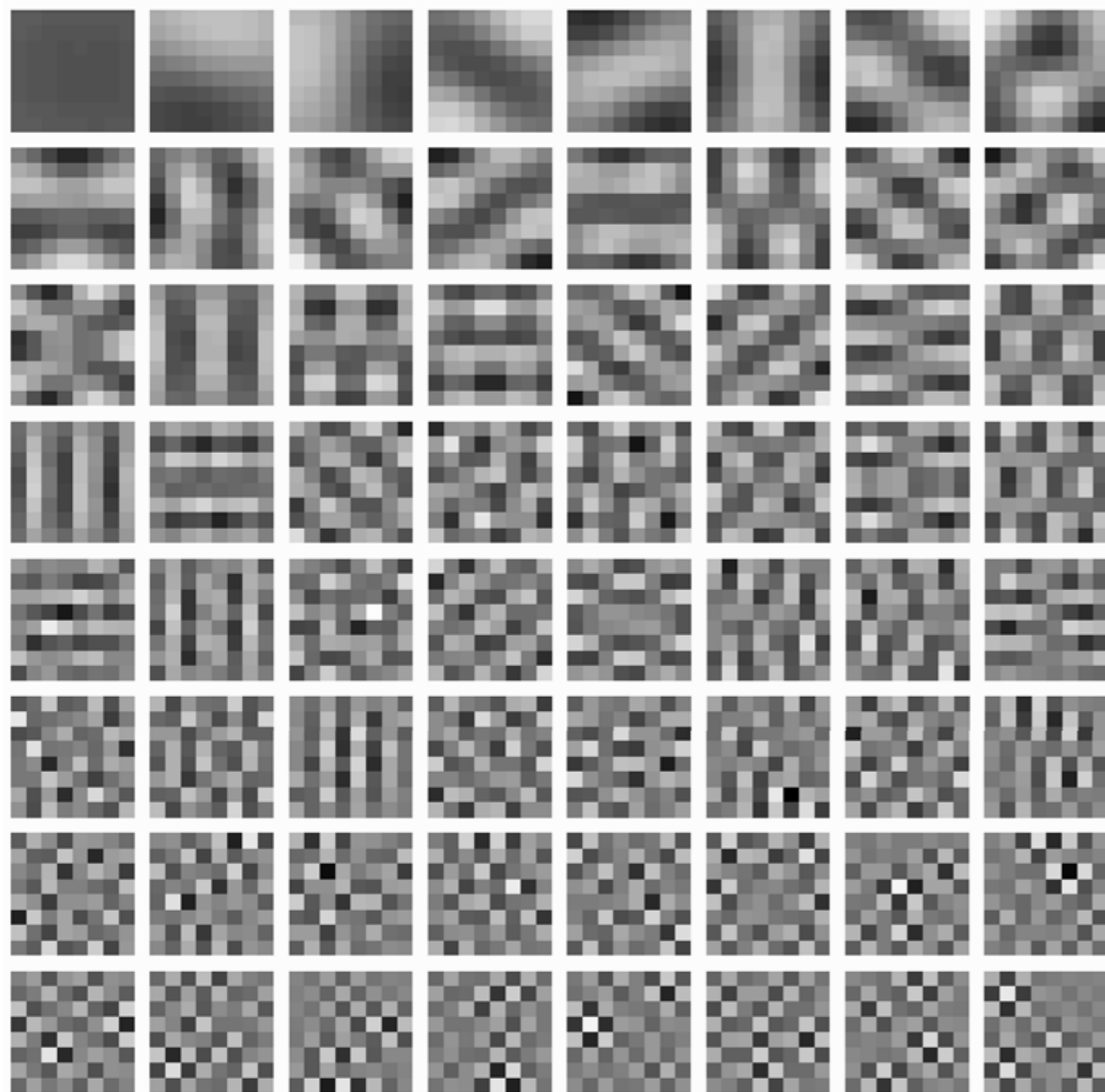
Eigen-Patches

- What do the PCA basis functions look like for local regions of natural images?



(Szeliski and Fleet)

Eigen-Patches

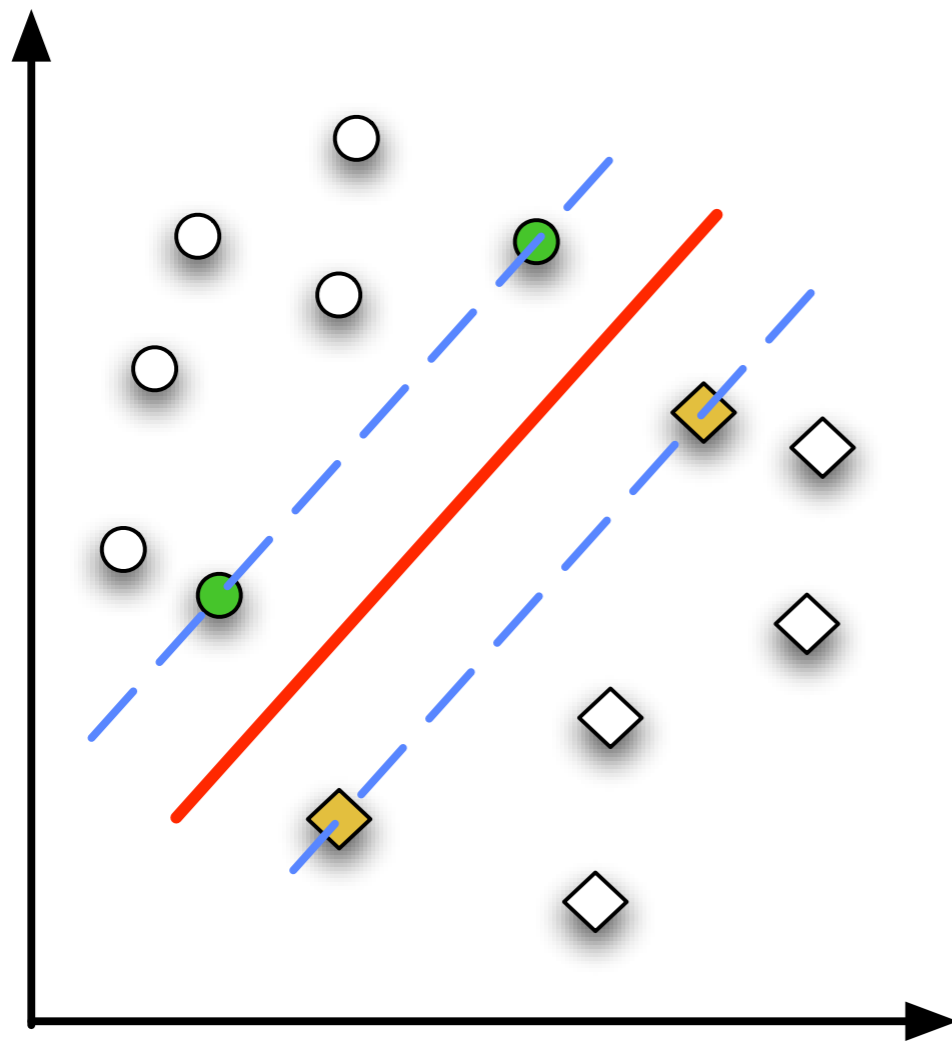


- Does this look familiar?
 - 8x8 patches
- DCT2 - basis image set.
 - Forms the basis of modern-day image and video coding.
- Pixels in natural images are naturally correlated with one another.

(Szeliski and Fleet)

Discriminative Approaches

- Better generalization performance can often be realized by learning the difference between two classes.
- We no longer get caught up with the problem of attempting to synthesize all variations of an object.



“How do we get negative examples?”

Object Registration and Tracking from a Learning Perspective (Part 2)

Simon Lucey
The Robotics Institute, Carnegie Mellon University

Overall Course Outline

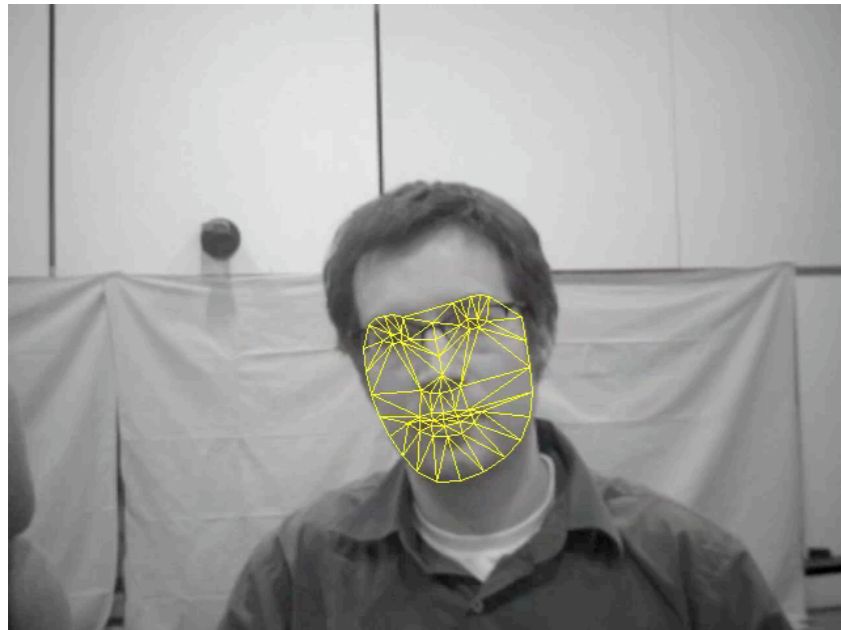
- Lecture 1

- Why registration and tracking is hard?
- Exhaustive Search (ES).
- Generative Models for ES Registration

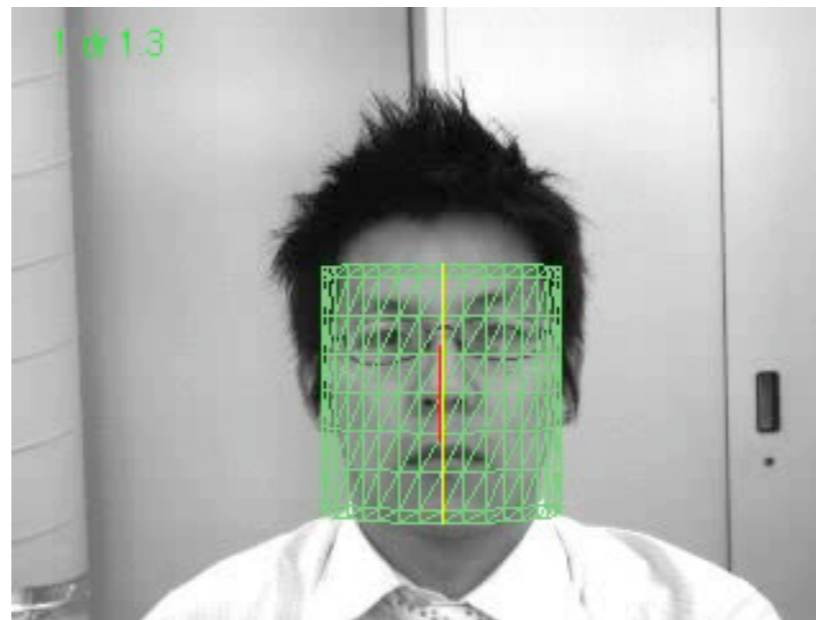
- Lecture 2

- Discriminative Models for ES Registration
- Efficient ES using FFT.
 - Correlation Filters.
 - Support Vector Machines
 - Neural Networks
- Efficient ES using integral images
 - Adaboost
 - Mutual Information
- Speed and performance comparisons.

Some Motivation for Course



“Face Recognition”



“Pose Estimation”



“Body Tracking”



“Speech Reading”



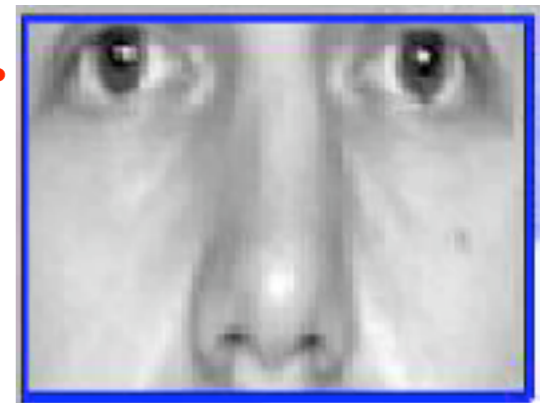
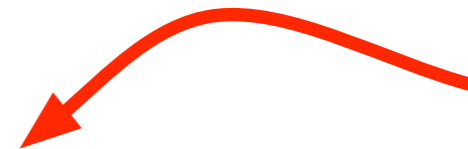
“Palm Recognition”



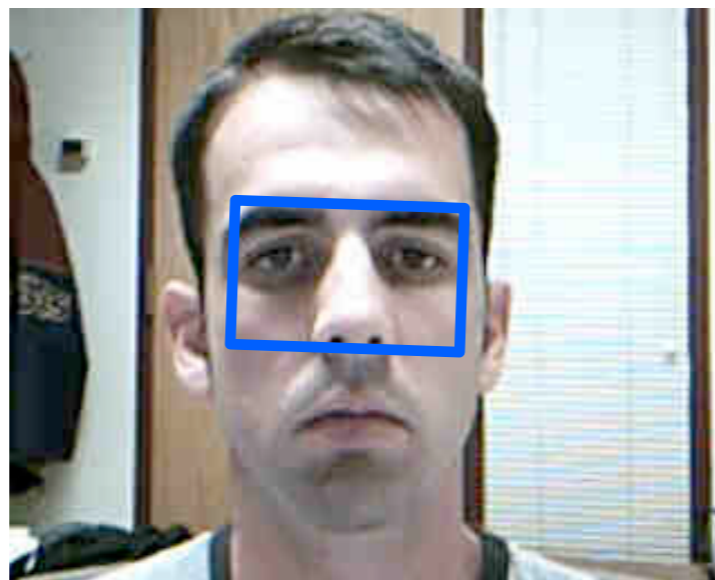
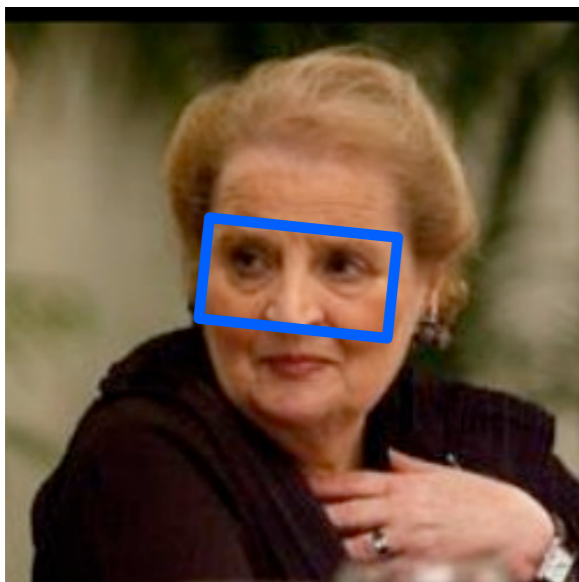
“Car Tracking”

Why is it hard?

- Want to register an object even when *identity* and *quality* varies.



“Template”



“Source Image”

Two Problems in Registration

1. *Learning*,

- How do I learn an object template/model that satisfactorily discriminates between the object and the image background?

2. *Fitting*,

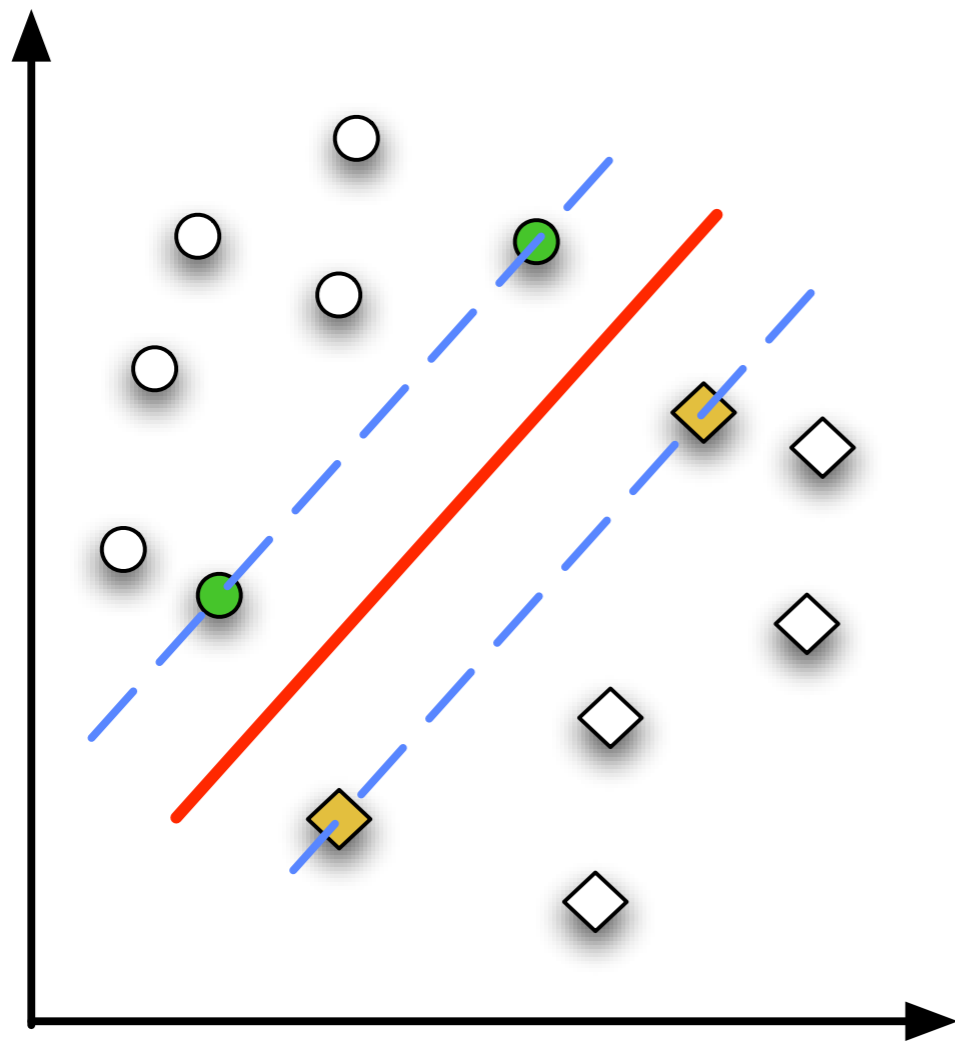
- How do I efficiently evaluate the object template/model across all possible warp values?

(Covered last lecture proposed ES)

“As we will show through this course, both questions are linked!!!”

Discriminative Approaches

- Better generalization performance can often be realized by learning the difference between two classes.
- We no longer get caught up with the problem of attempting to synthesize all variations of an object.



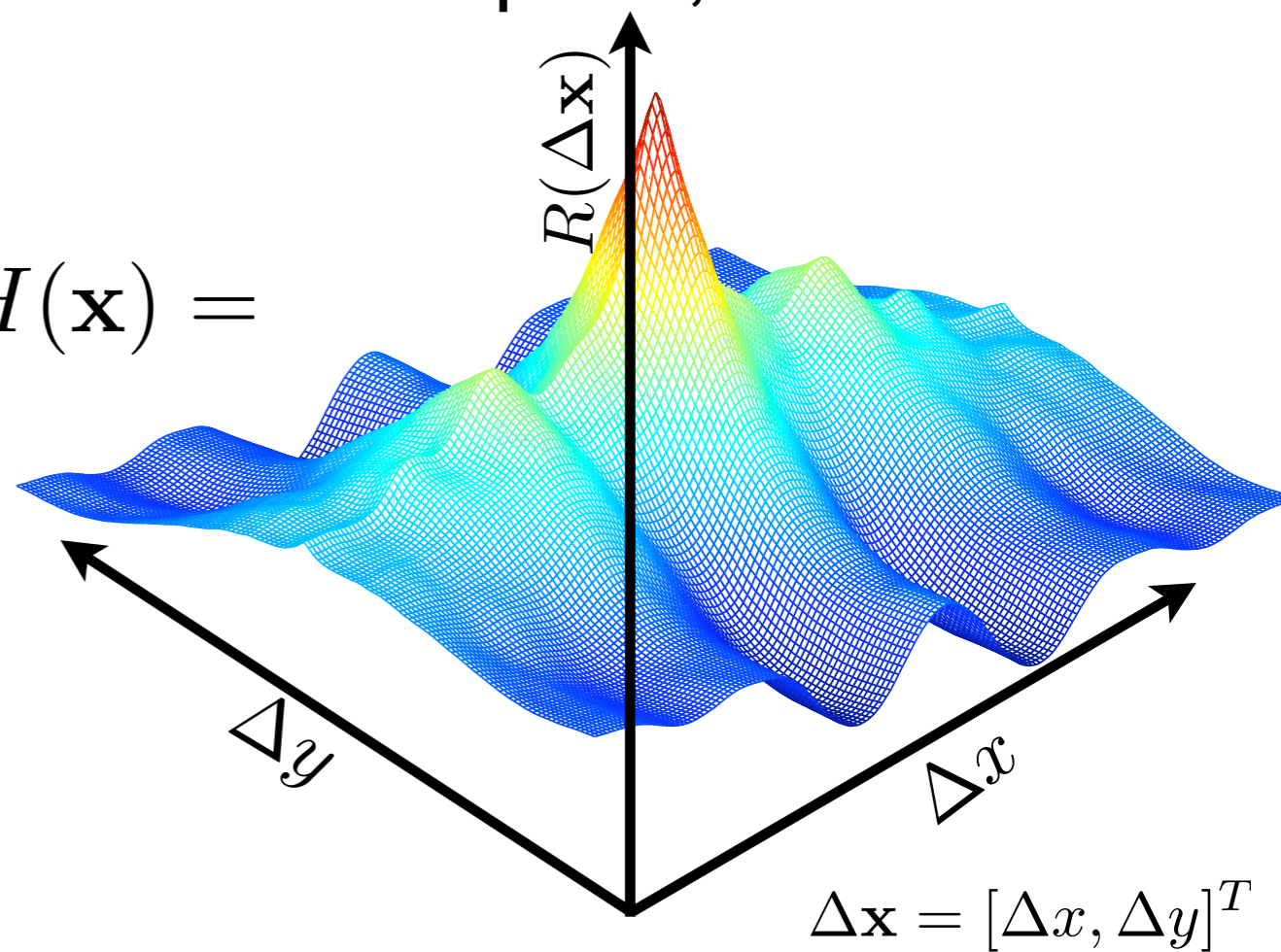
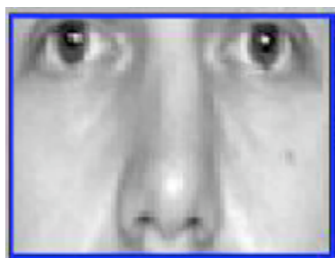
“How do we get negative examples?”

Correlation Filters

- Mahalanobis, Kumar and Casasent (1987) realized that a Fourier representation is an ideal way to learn a template using positive and negative examples, without having to generate negative examples.
- Approach takes advantage of the cross-correlation function of an annotated image and a learnt filter/template,

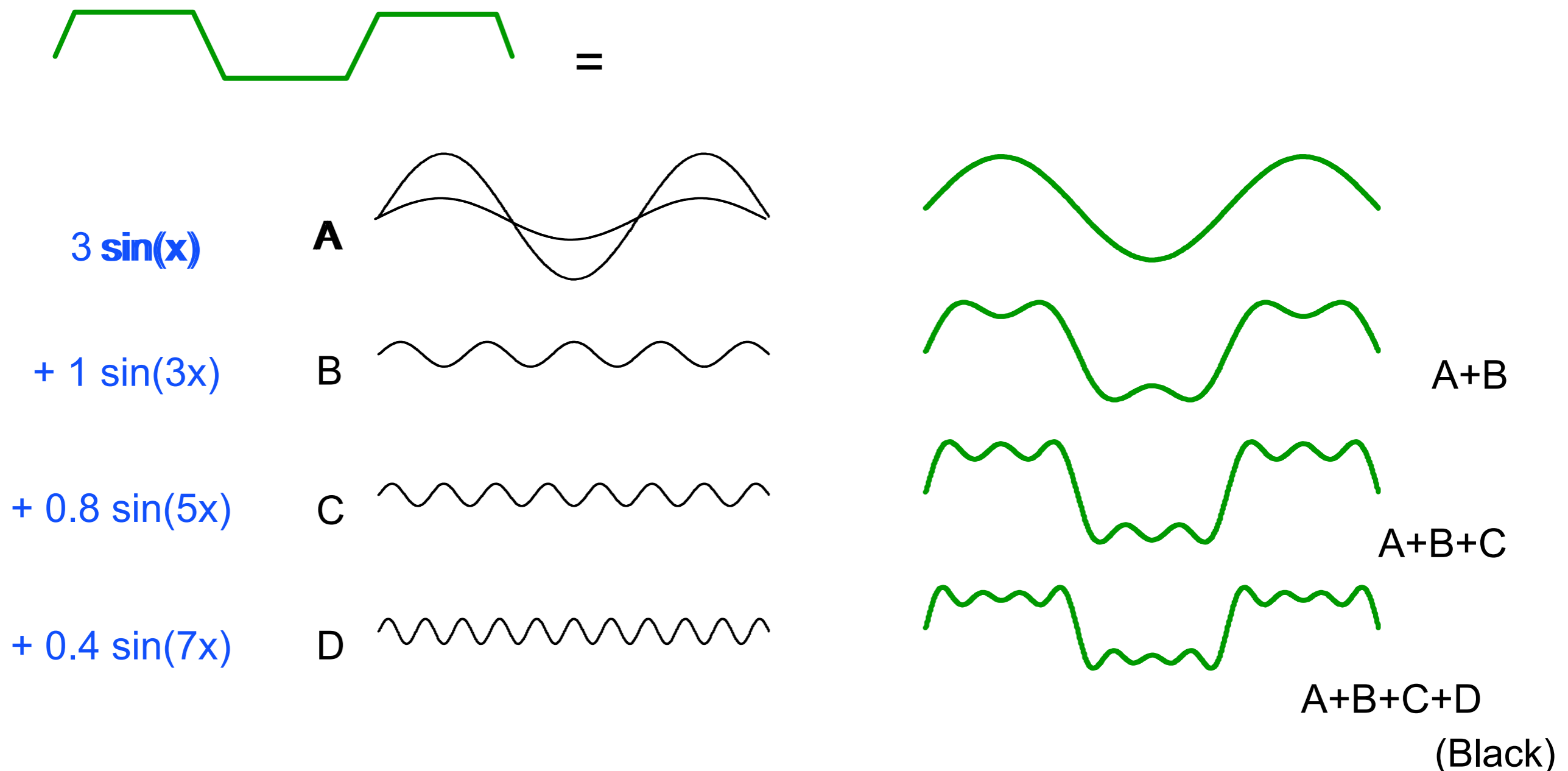
$$R(\Delta \mathbf{x}) = \sum_{\mathbf{x}} T(\mathbf{x} + \Delta \mathbf{x}) H(\mathbf{x}) =$$

where, $T =$



Fourier Transform in Pictures

- Quick revision of the Fourier transform,



Correlation Through FFT

- Can obtain a cross-correlation function, far more efficiently in the Fourier domain.
- We can employ a 2-D fast Fourier transform (FFT) $\mathcal{F}\{\}$ to take advantage of this redundancy.

$$R(\Delta \mathbf{x}) = T(\mathbf{z} + \Delta \mathbf{x})^T H(\mathbf{z}) \text{ “Linear correlation”}$$

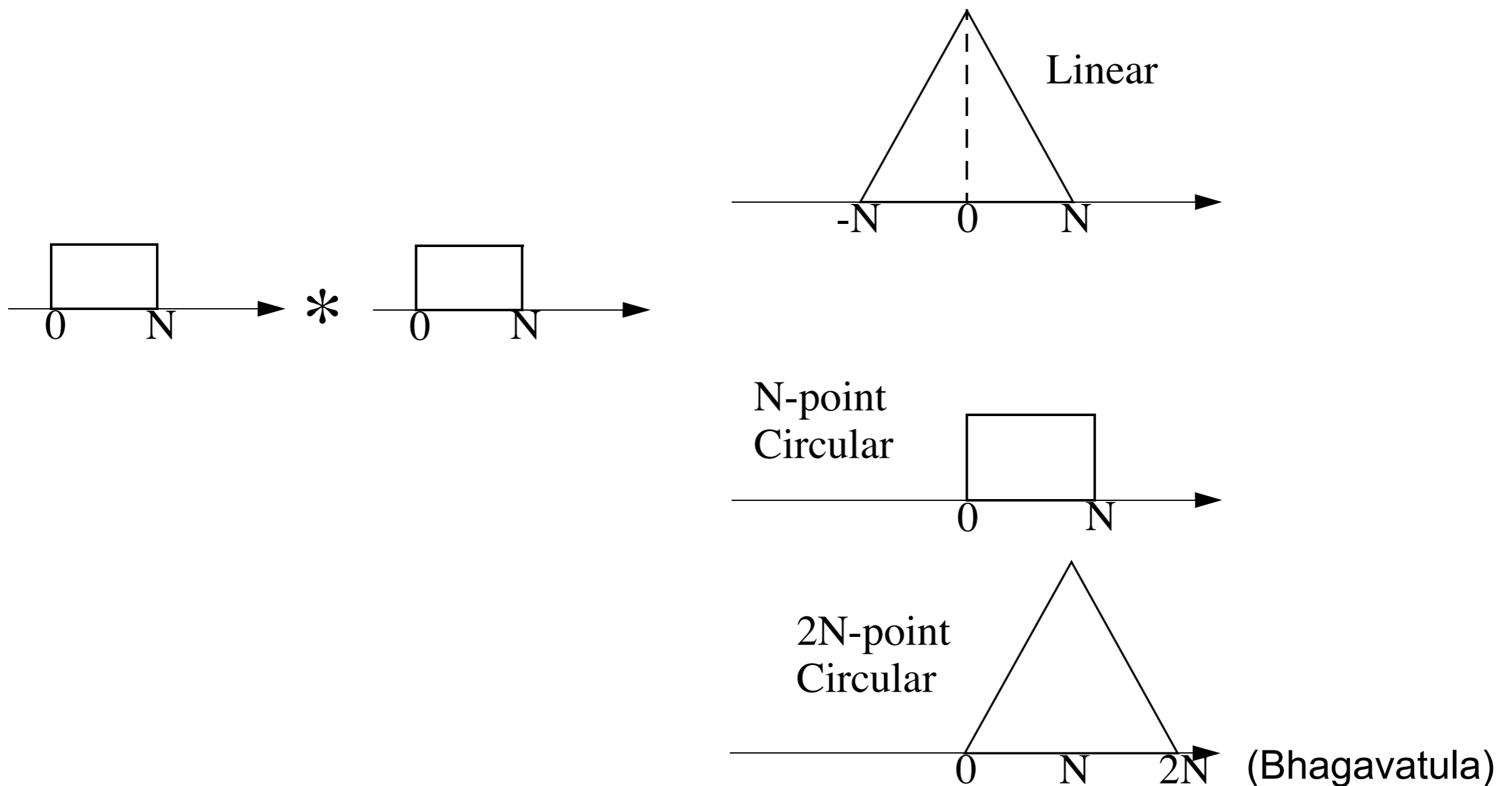
$$R(\Delta \mathbf{z}) = [R(\Delta \mathbf{x}_1), \dots, R(\Delta \mathbf{x}_D)]^T$$

$$R(\Delta \mathbf{z}) = \mathcal{F}^{-1} \{ \mathcal{F}\{T(\mathbf{z})\} \cdot \mathcal{F}^* \{H(\mathbf{z})\} \} \text{ “Circular correlation”}$$

Caution: without the complex conjugate this operation is convolution not correlation!

Linear Vs. Circular Correlation

- Using an FFT results in a circular, not linear, correlation.



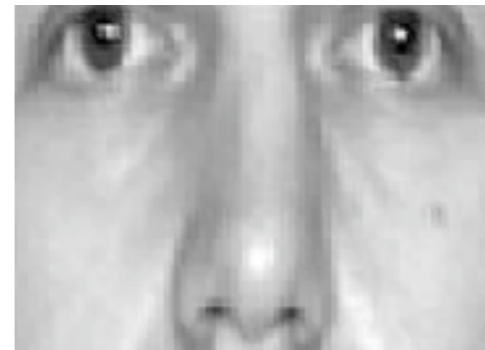
Linear Vs. Circular Correlation

- Using FFTs leads to circular correlations rather than linear correlations
- A circular correlation is as if the signals are made artificially periodic with a period N where N is the FFT size.
- If N is very large, then the circular and linear convolutions will be mostly similar; otherwise, the circular correlation obtained will be an aliased version of the linear

MACE Filter

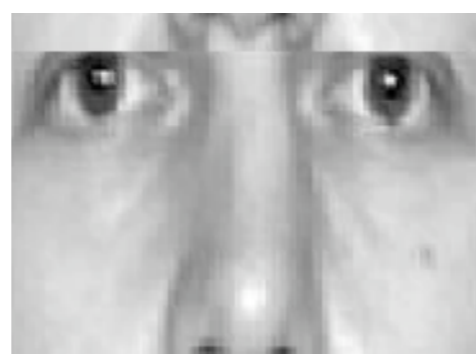
- Mahalanobis et al. wanted to ensure that the correlation of a registered image with a template/filter is unity,

$$T(\mathbf{z})^T H(\mathbf{z}) = 1$$

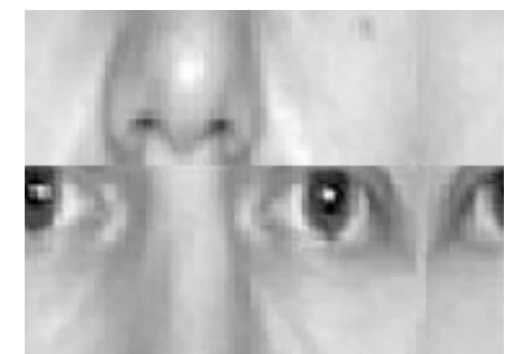


- And that the average correlation energy across all circular alignments is minimized,

$$\min_{H(\mathbf{z})} = \sum_{\Delta \mathbf{x}} ||T(\mathbf{z} + \Delta \mathbf{x})^T H(\mathbf{z})||^2$$



.....



Referred to as a “Minimum Average Correlation Energy (MACE) Filter”

MACE Filter

- In spatial domain this optimization would be difficult and messy. Fortunately, things are simplified greatly in the Fourier domain since,

$$\begin{aligned} T(\mathbf{z})^T H(\mathbf{z}) &= \frac{1}{D} \mathcal{F}\{T(\mathbf{z})\}^T \mathcal{F}\{H(\mathbf{z})\} \\ &= \frac{1}{D} T(\mathbf{f})^T H(\mathbf{f}) \quad \text{“Measure of aligned correlation”} \end{aligned}$$

- and,

$$\sum_{\Delta \mathbf{x}} ||T(\mathbf{z} + \Delta \mathbf{x})^T H(\mathbf{z})||^2 = \frac{1}{D} H(\mathbf{f})^T \text{diag}[T(\mathbf{f}) \cdot T^*(\mathbf{f})] H(\mathbf{f})$$

“Measure of average cross-correlation energy”

MACE Filter

- Problem can be formally written as,

$$\arg \min_{H(\mathbf{f})} H(\mathbf{f})^T \mathbf{S} H(\mathbf{f})$$

$$\text{given, } \mathbf{C}^T H(\mathbf{f}) = \mathbf{y}$$

“QP problem with equality constraints”

- where,

$$\mathbf{S} = \text{diag}\left\{\sum_{n=1}^N T_n(\mathbf{f}) \cdot T_n^*(\mathbf{f})\right\}$$

“Average spectral energy”

$$\mathbf{C} = [T_1(\mathbf{f}), \dots, T_N(\mathbf{f})]$$

“Concatenated training images”

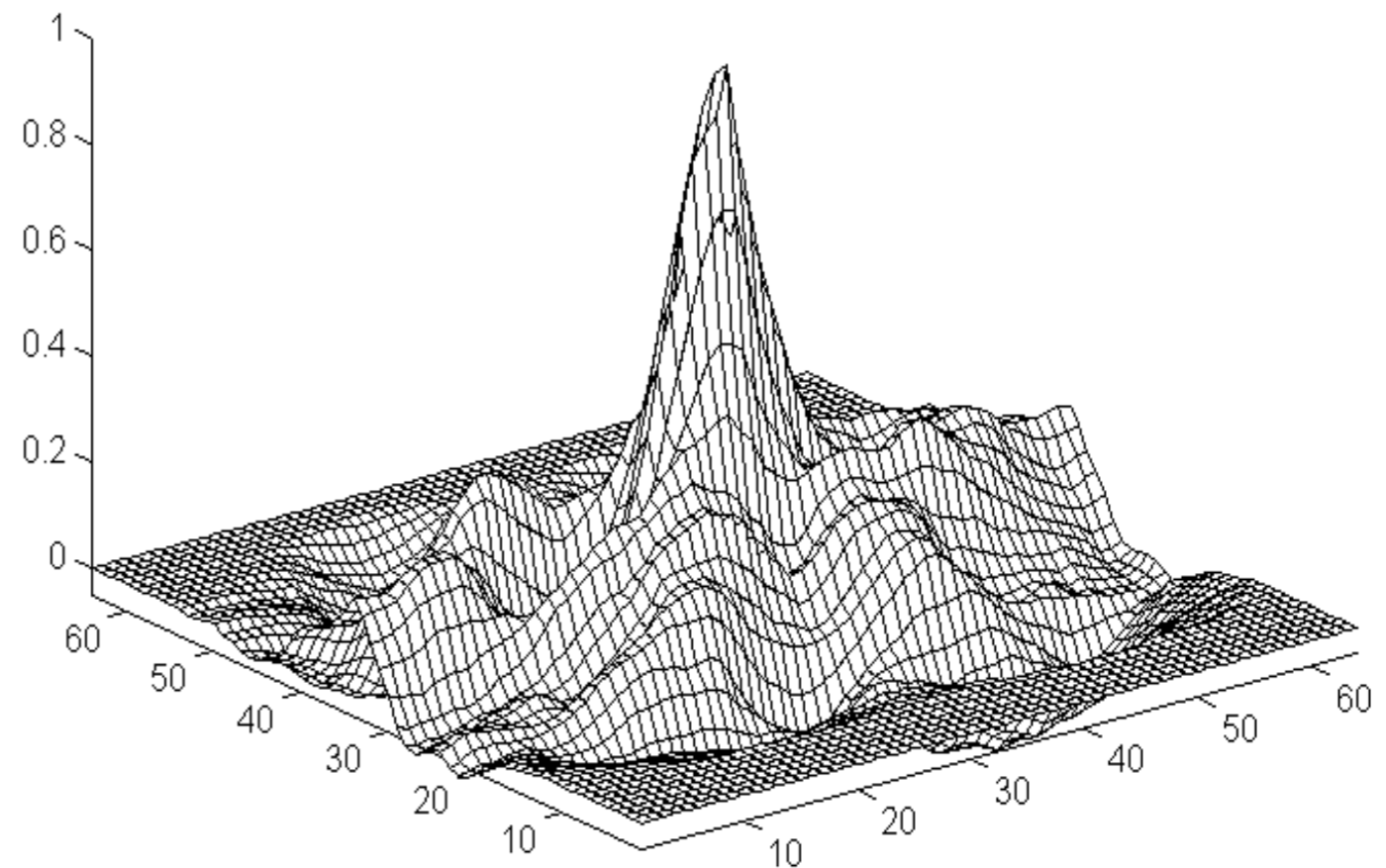
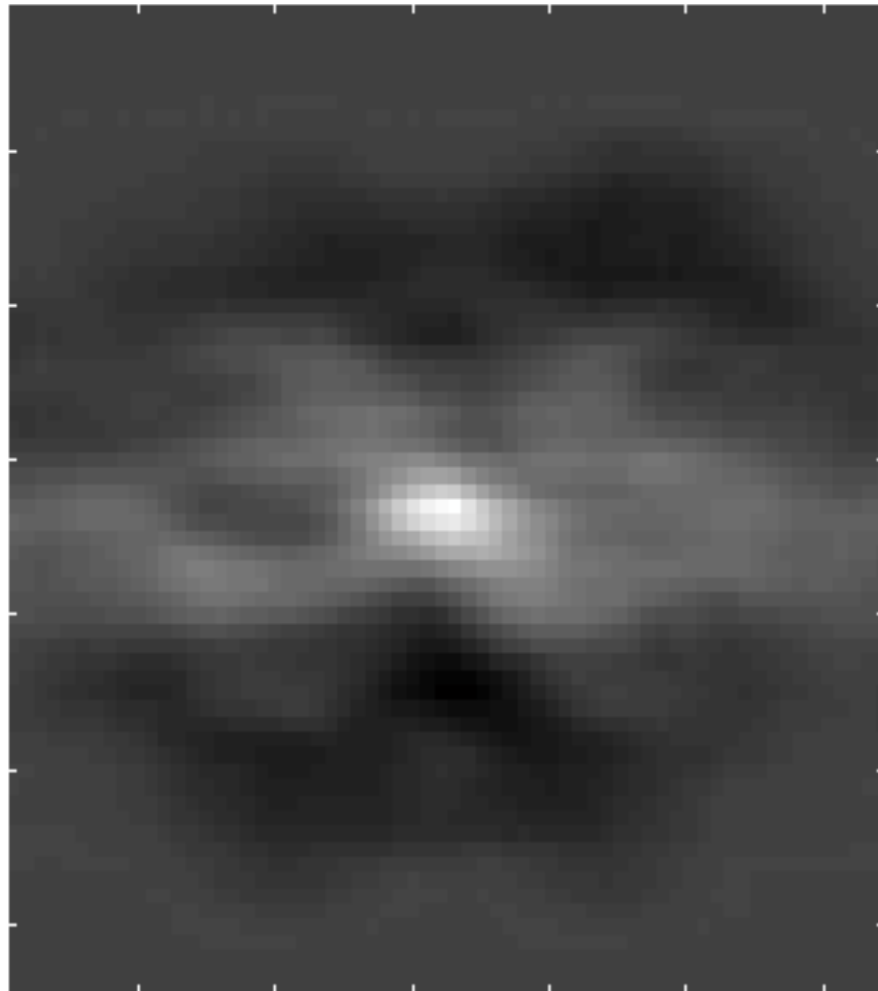
$$\mathbf{y} = [D, \dots, D]^T$$

“Desired correlation output for each training image”

- Solution can be explicitly found in the frequency domain,

$$H(\mathbf{f}) = \mathbf{S}^{-1} \mathbf{C} (\mathbf{C}^T \mathbf{S}^{-1} \mathbf{C})^{-1} \mathbf{y}$$

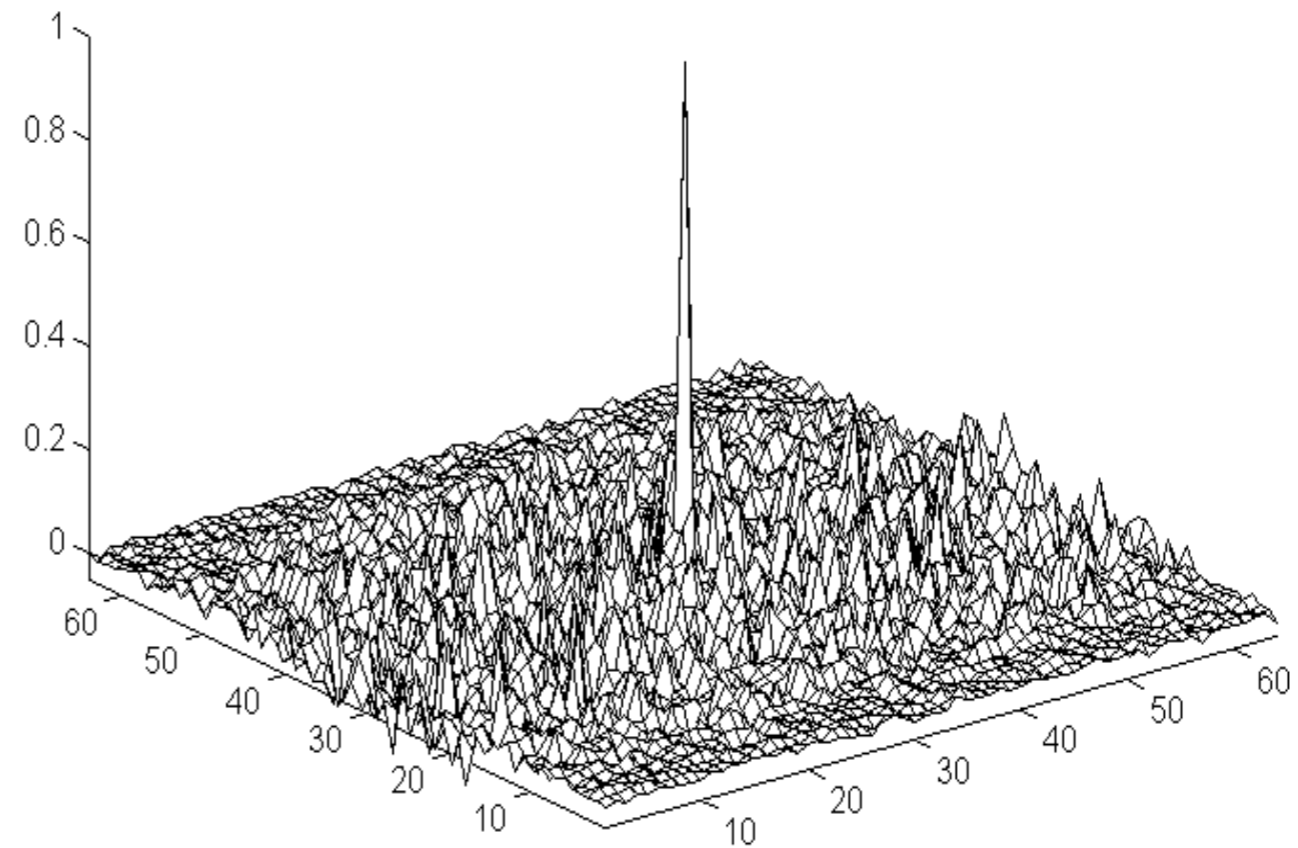
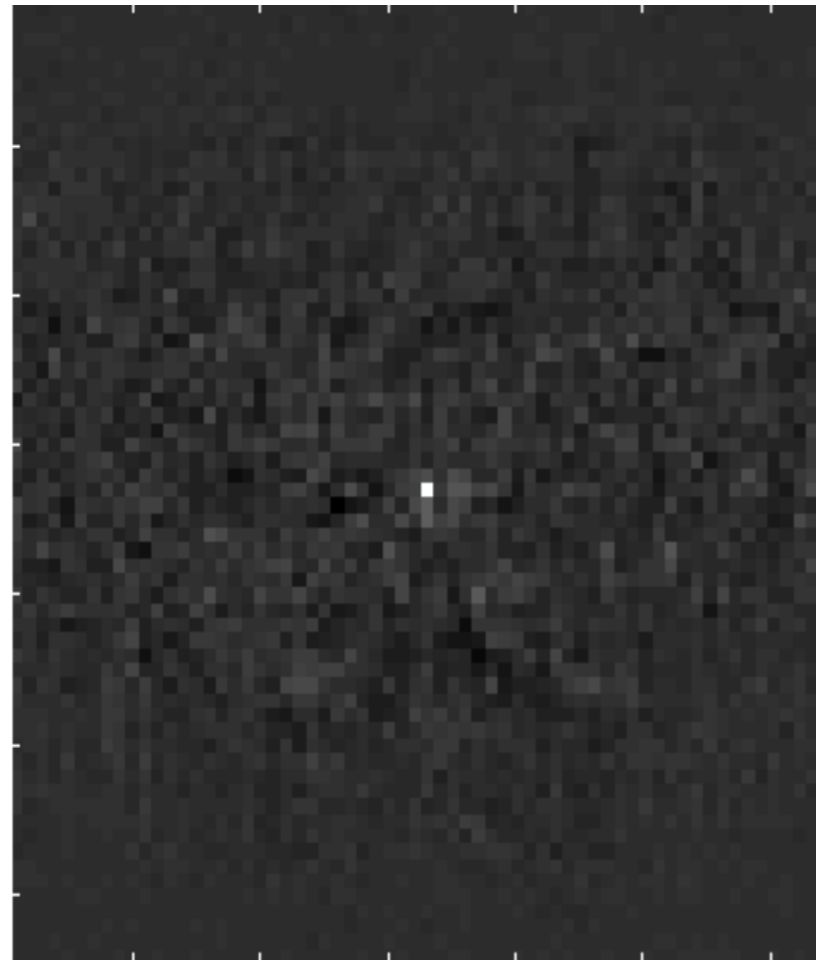
Example of Average Template Response



$$H(\mathbf{f}) = \frac{1}{N} \sum_{n=1}^N T_n(\mathbf{f}) \quad \text{“Average filter”}$$

(Bhagavatula)

Example of MACE Response

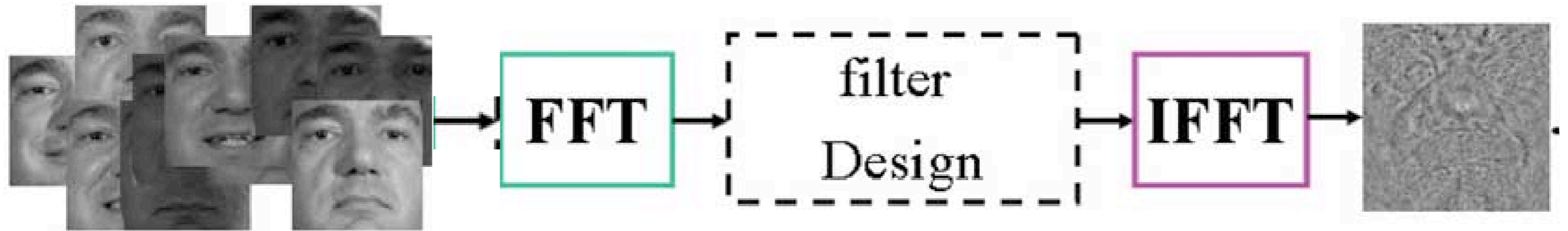


$$H(\mathbf{f}) = \mathbf{S}^{-1} \mathbf{C} (\mathbf{C}^T \mathbf{S}^{-1} \mathbf{C})^{-1} \mathbf{y} \quad \text{"MACE filter"}$$

(Bhagavatula)

Visualizing MACE Filter

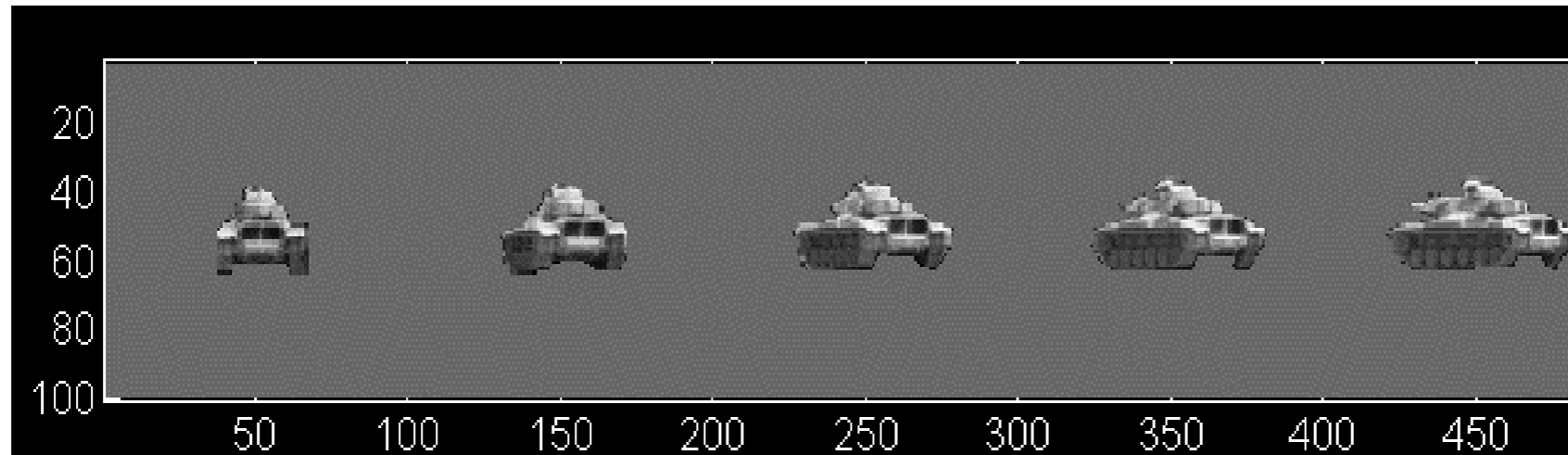
- Visualizing the MACE filter,



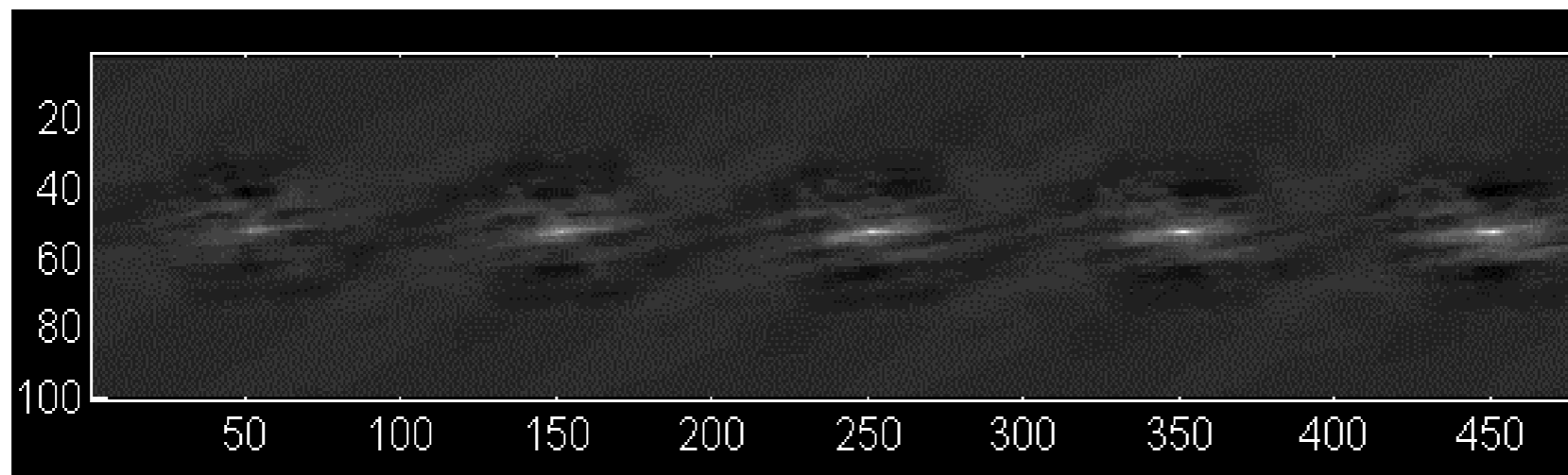
(Bhagavatula)

Correlation Filter Applications

- Found a lot of success in automatic target recognition (ATR),



Input Images

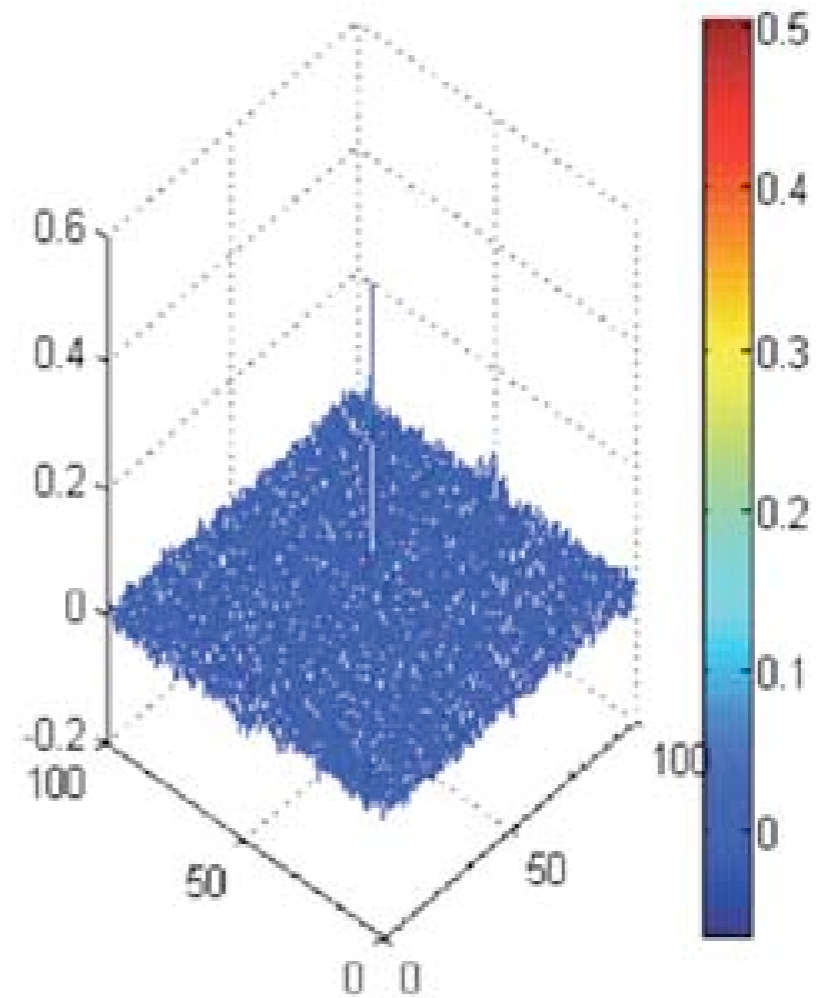
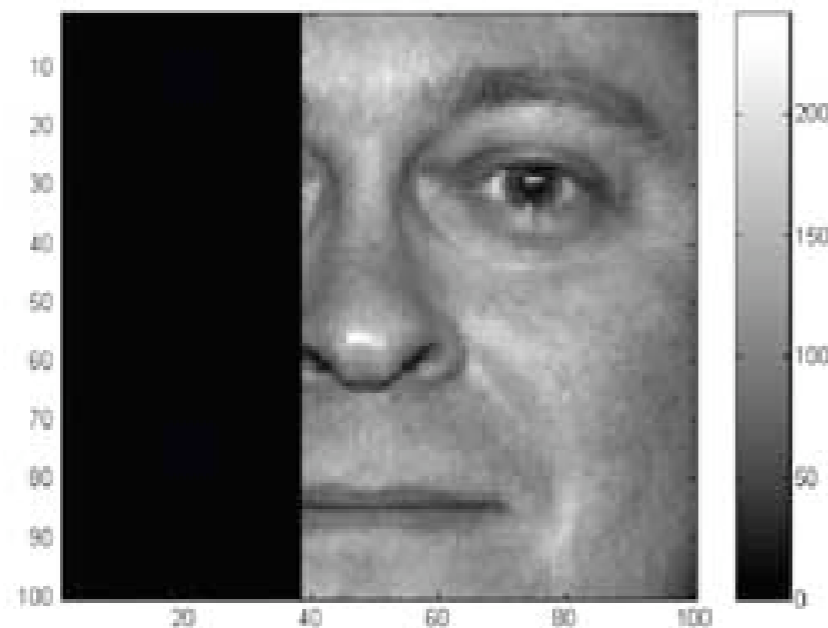


Output Correlations

(Bhagavatula)

Correlation Filters

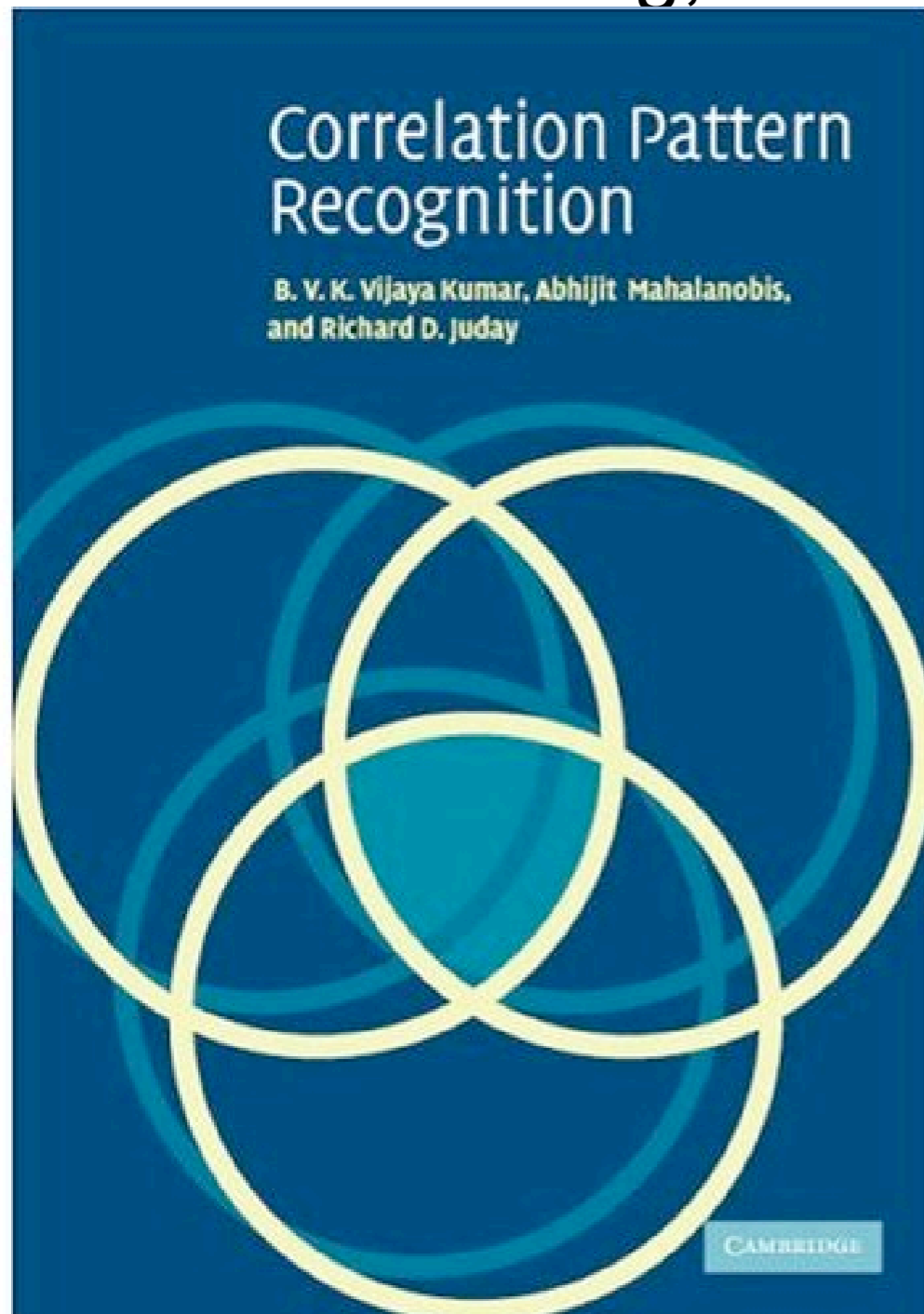
- More recently has found success in biometrics (face and iris recognition):



(Bhagavatula)

Correlation Filters

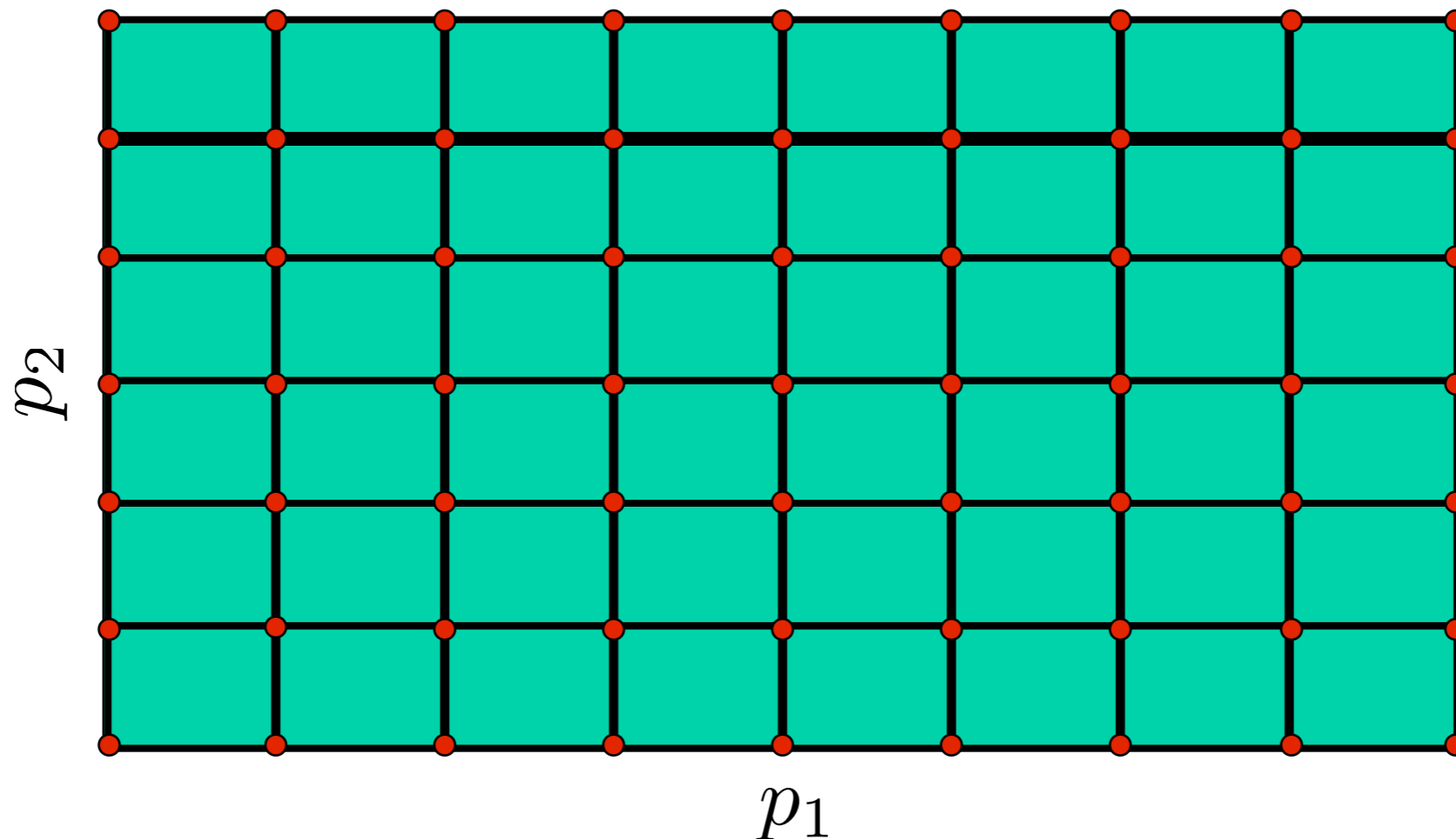
- Further reading,....



B. V. K. Vijaya Kumar, Abhijit Mahalanobis, Richard D. Juday, "Correlation Pattern Recognition", Cambridge, 2005.

Improving ES through FFT

- ES can still be awfully slow if we want our registration to run faster than real-time (i.e., 30 fps).
- Since we are playing in the Fourier domain can anything be done?



$$\mathbf{p} = \{p_1, p_2\}$$

“Discrete Warp Space”

Search through Correlation

- For the translation case, it turns out we can view the job as a 2-D cross-correlation operation ($*$),

$$corr(\mathbf{p}) =$$



$$\mathbf{p} = \{\text{x-translation, y-translation}\}$$

Computation $\Rightarrow O(n \log n) < O(n^2)$
“Frequency” *“Spatial”*

Correlation Filter Limitations

- Although elegant, correlation filters have some problems.
 - Negative images are just circular shifts of the aligned images.
 - Objects in natural images vary in more ways than that (scale/rotation).
 - Does not give graceful degradation if object is not aligned exactly.
- We are left with the problem of synthetically obtaining positive and negative examples of an object.
- Two problems,
 - Collecting positive and negative training examples.
 - Learning a computationally feasible model with large dimensionality ($D > 400$ pixels) and large number of examples ($N > 100,000$ s)
- Need to employ machine learning techniques that can deal with large amounts of data and generalize well from them.

Generating Positive Examples



- Coarsely normalize for scale, orientation and translation.
- Bad idea to normalize too much due to the nature of the discrete ES.

Generating Negative Examples



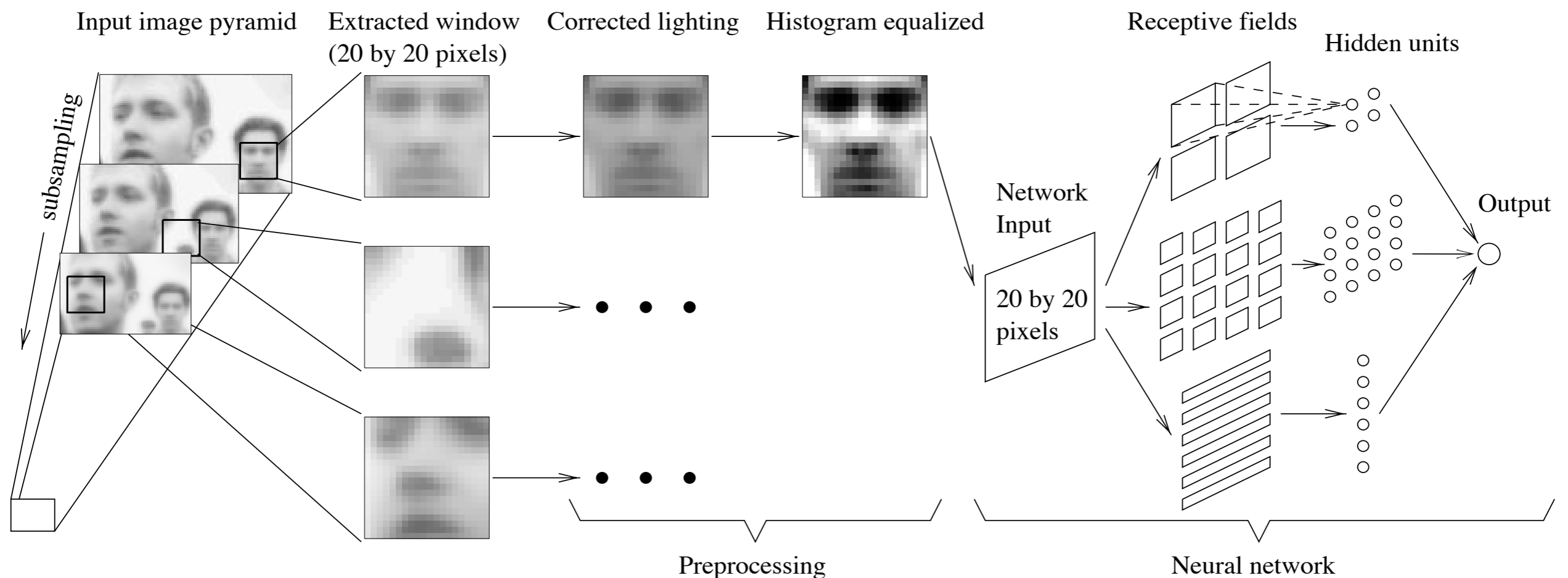
- Obtain a large number of non-object images (through the web).
- Randomly, sample through various positions within the images.

Learning a Discriminative Classifier

- If we are learning a classifier to discriminate between large training sets of positive and negative images we need,
 - An algorithm that can learn sequentially (maybe impossible to optimize over all training data in batch)
 - Good candidates:- Boosting, SVM (Platt's method), Neural networks.
 - Classifiers that are computationally inexpensive due to ELS.

ANNs for Learning

- Rowley, Baluja and Kanade (1998) had success applying Artificial Neural Networks (ANN).

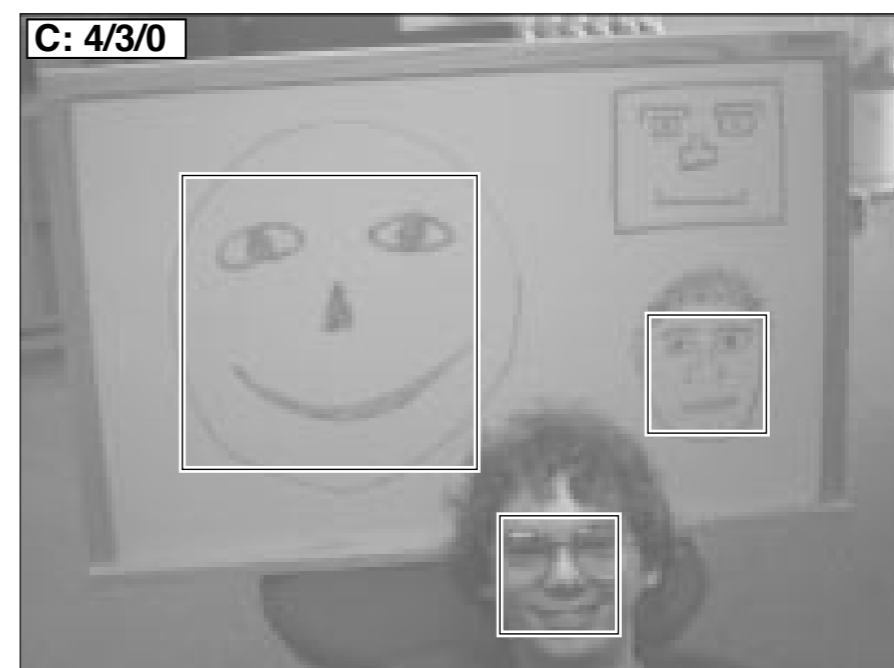


- Works well, but ANN based on many heuristics.
- Computationally costly. (1 fps)

(Rowley)

ANNs for Learning

- Examples of performance,



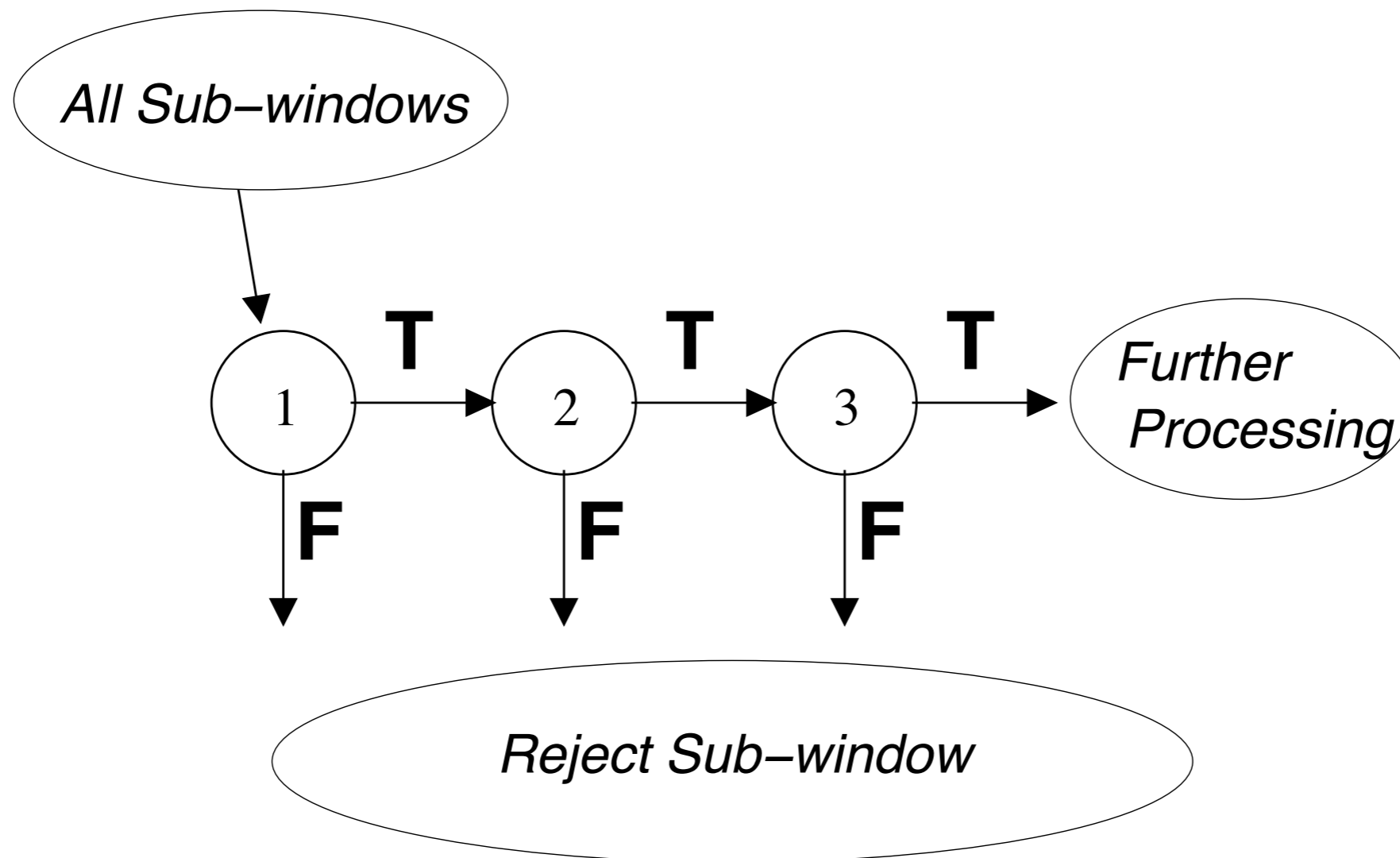
(Rowley)

SVMs for Learning

- Support vector machines (SVMs) were first employed by Osuna, Freund, and Girosi (1997) for face detection.
- More elegant solution than Rowley et al.'s
 - Better generalization properties (maximizing a margin)
 - No heuristic feature selection.
- Obtained reasonable results, but only for non-linear SVM.
 - Using RBF kernel.
- Still computationally expensive as SVM could contain 100s-1000s of support vectors.

Cascade Classifier

- Instead of searching all regions of an image with the same complexity classifier, we can use a cascade.



“Example of a Cascade Classifier”

(Viola & Jones)

Training a Cascade

- Each level in the cascade relies heavily on a ROC curve.
 - False Reject Rate (FRR) is held static for each level.
 - False Accept Rate (FAR) at every level should improve.
- For example if we are learning a cascade of linear classifiers,

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \quad \begin{array}{c} \text{object} \\ \geq \\ < \\ \text{background} \end{array} \quad Th$$

- The threshold (Th) needs to be chosen such that the FRR is static and the FAR is reduced.
- Approach, however, is prone to over-fitting, will work on only some classifiers.

Cascaded SVM

- Romdhani et al. (2001) proposed a cascade strategy for employment with an SVM.
- Employed a reduced set method which attempts to,

$$\sum_{i=1}^{N_z} \beta_i \Psi(\mathbf{z}_i) \approx \sum_{i=1}^{N_x} \alpha_i \Psi(\mathbf{x}_i)$$

- where,
 $\mathbf{z}_i \in \mathcal{X}$ $\mathbf{x}_i \in \mathcal{X}$ $\alpha_i \in \mathcal{R}$ $\beta_i \in \mathcal{R}$
 $\mathcal{X} \leftarrow$ space of training images
 $\Psi() \leftarrow$ non-linear feature expansion
- Approach attempts to find a reduced set of support vectors that approximate the full set given $N_z \ll N_x$.

Cascaded SVM

- Examples of a reduced set SVM,

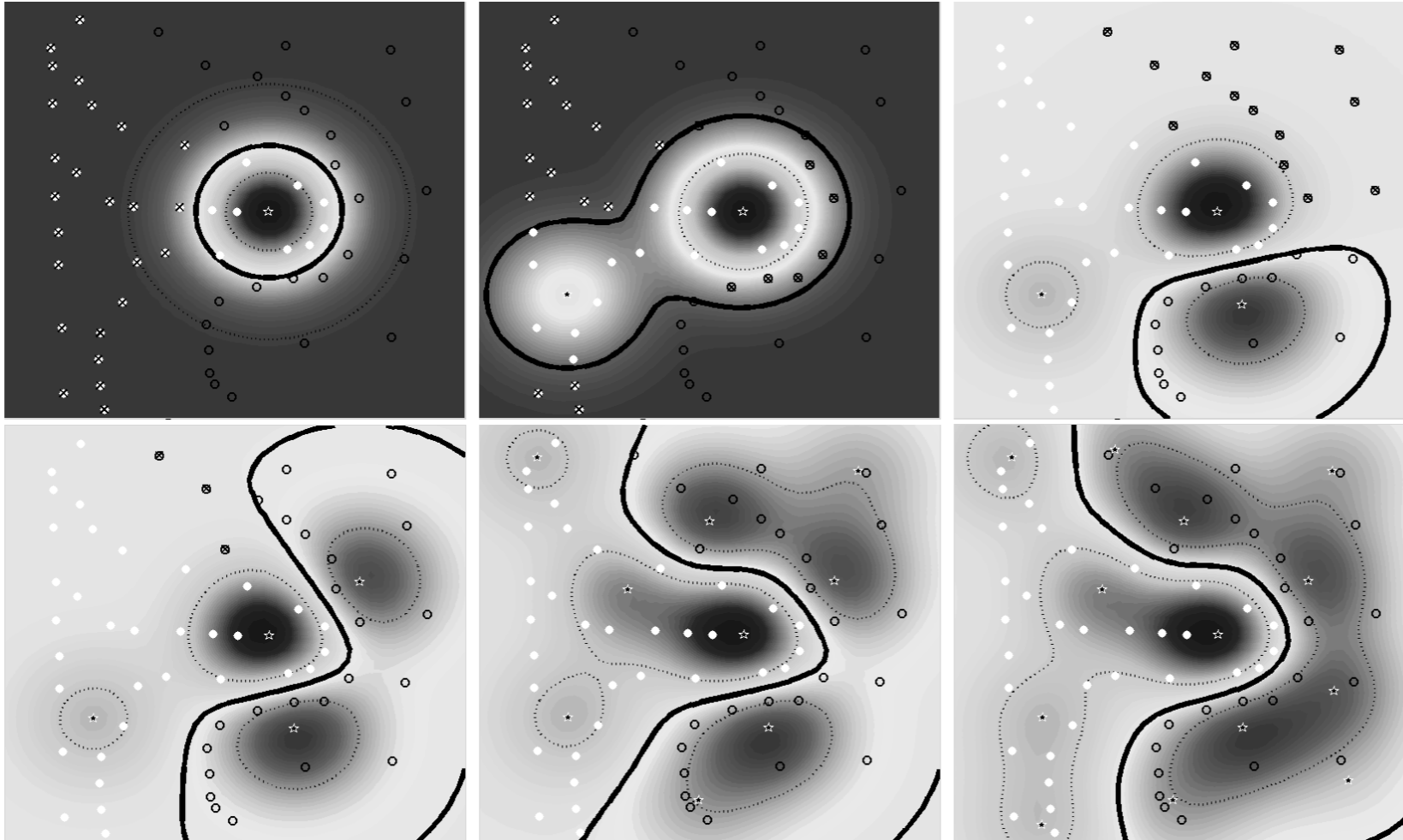


Figure 1: *The result of the sequential application of RV's (stars) to a classification problem, showing the result of using 1,2,3,4,9 and 13 RV's Darker regions indicate strong support for the classification.*

(Romdhani)

Cascaded SVM Results

- Vast majority of regions in an natural image can be rejected using small N_z .

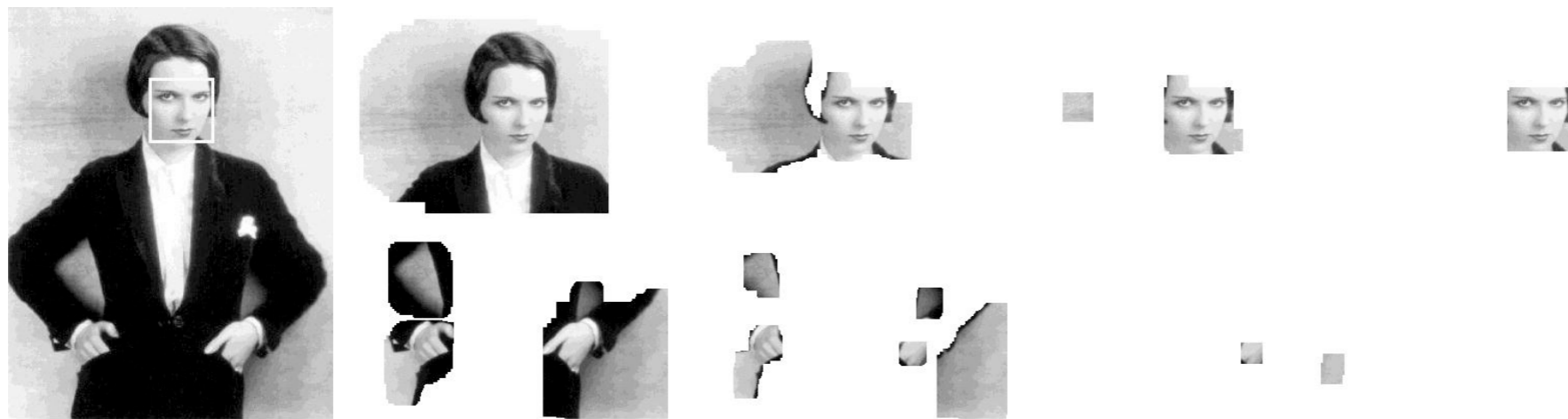
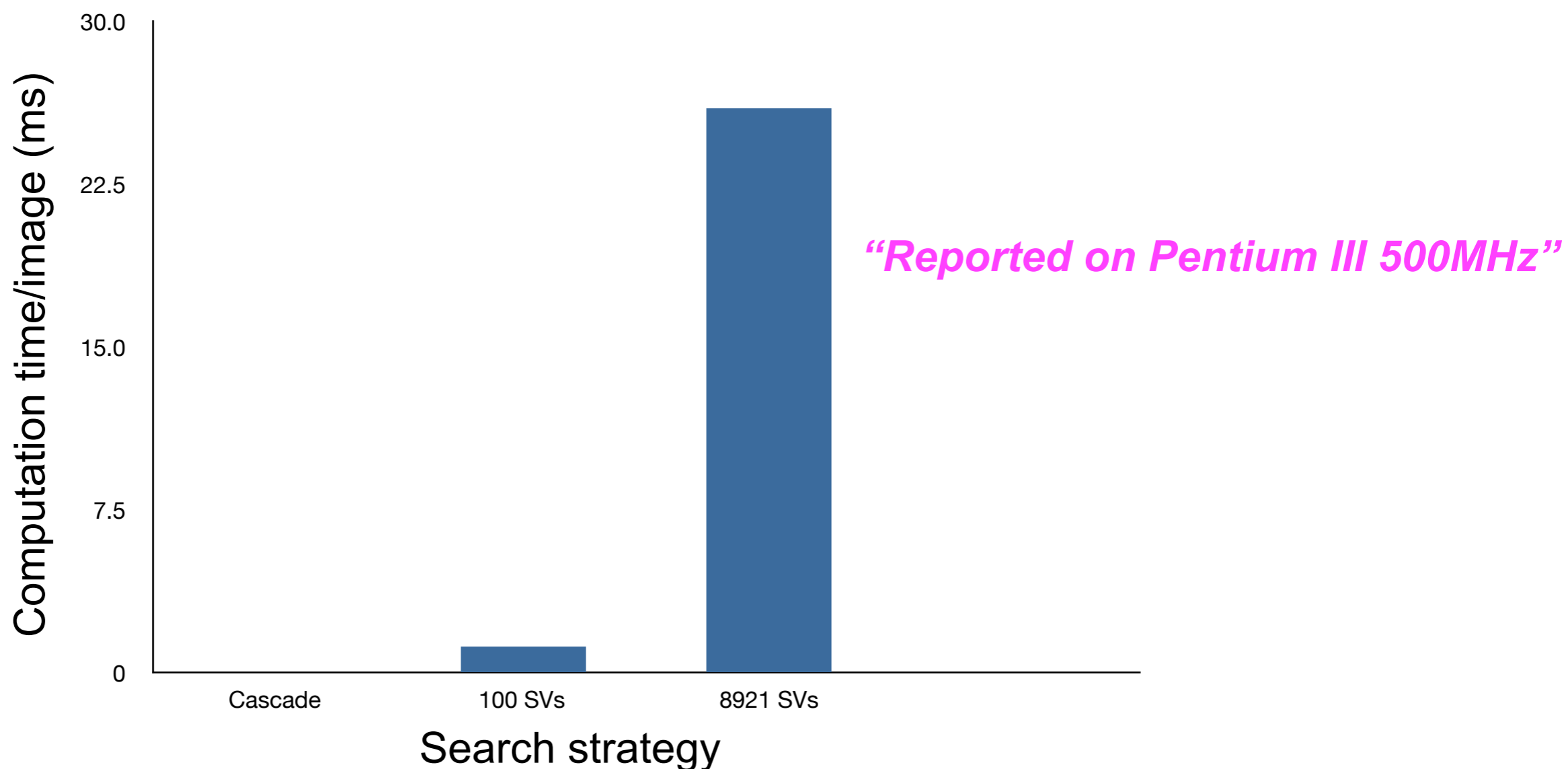


Figure 4: **From left to right:** *input image, followed by portions of the image which contain un-reject patches after the sequential evaluation of 1 (13.3% patches remaining), 10 (2.6%), 20 (0.01%) and 30 (0.002%) support vectors. Note that in these images, a pixel is displayed if it is part of any remaining un-rejected patch at any scale, orientation or position. This explains the apparent discrepancy between the above percentages and the visual impression.*

(Romdhani)

Computational Savings

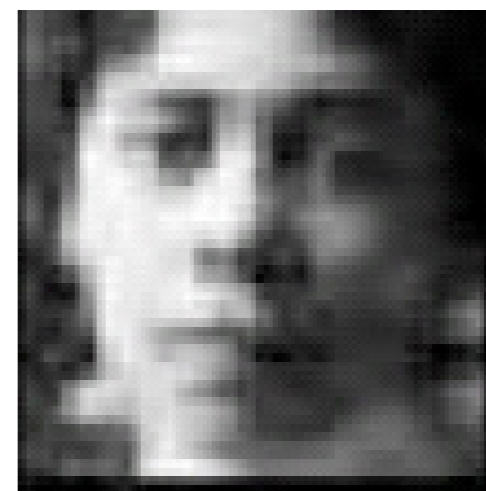
- Employing a cascade strategy can lead to sizable speedup.



Improving Speed Further

- Can we make ES even faster by moving away from pixels?
- Viola & Jones (2001) suggested using box filters.

- (Sub)image $I(x, y)$
- Two adjacent regions R_1, R_2



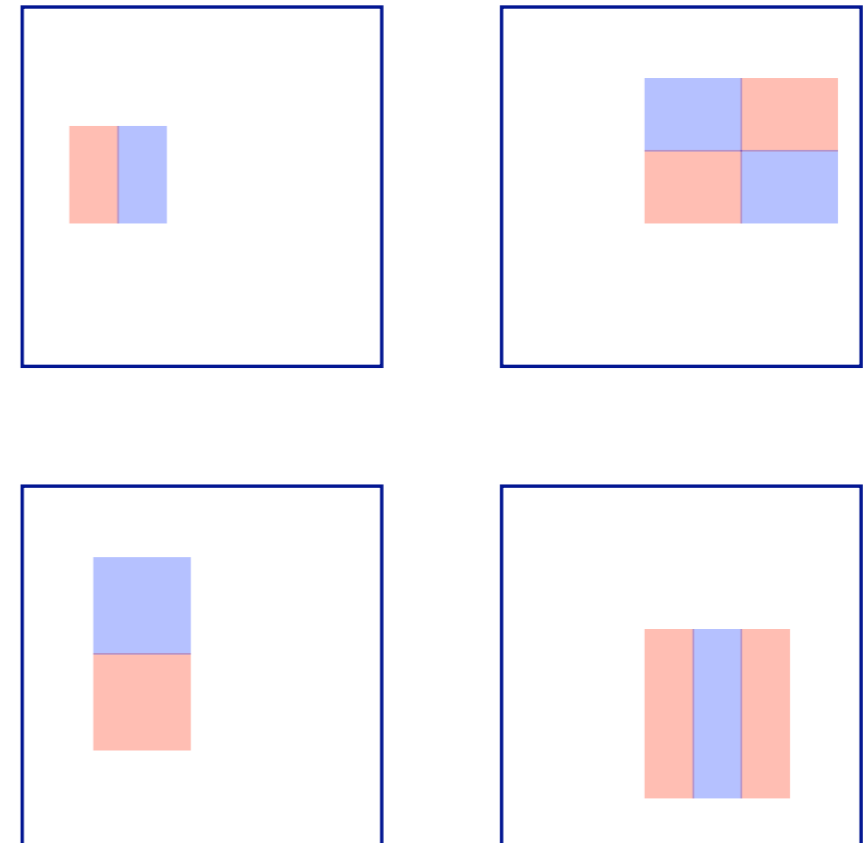
- Feature value:

$$f(I) = \sum_{(x,y) \in R_2} I(x, y) - \sum_{(x,y) \in R_1} I(x, y)$$

(Black)

Box Filters

- Three types of box filters
- Vary size, aspect ration, location, orientation
- Defined over 24×24 window
 \Rightarrow 160,000 distinct features
- Classifier using a single feature:

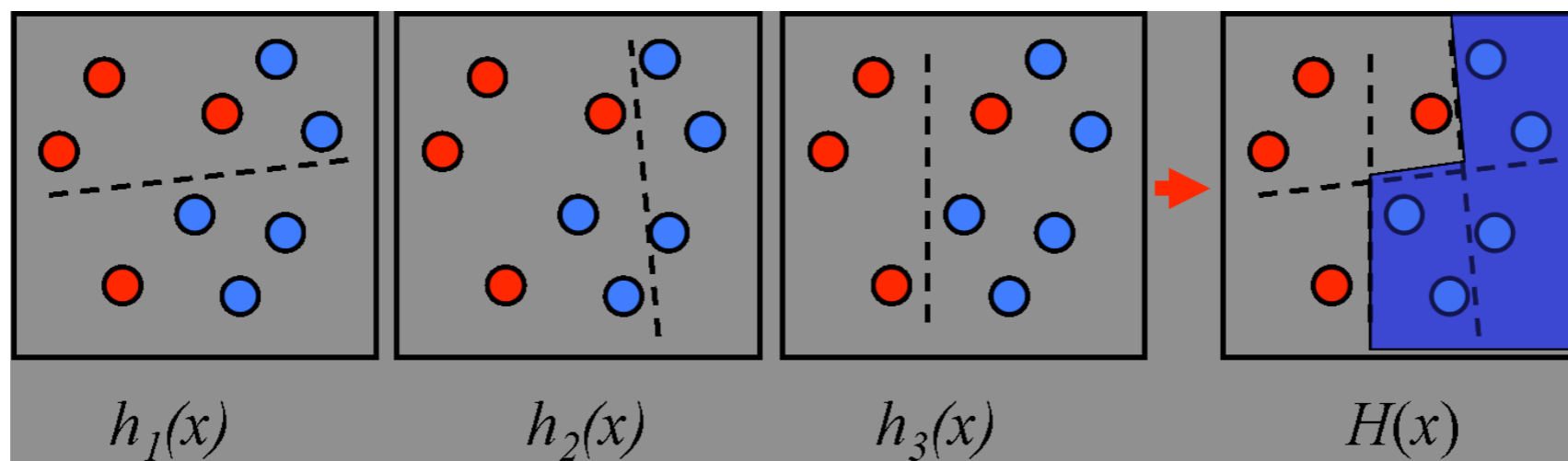


(Black)

Learning with Box Filters

- Boosting is ideally suited to be used with box filters.
- Techniques like AdaBoost, LogitBoost or GentleBoost can naturally learn a complex classifier from a cascade of weak classifiers.

$$H(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^M \alpha_m h_m(\mathbf{x}) \right)$$

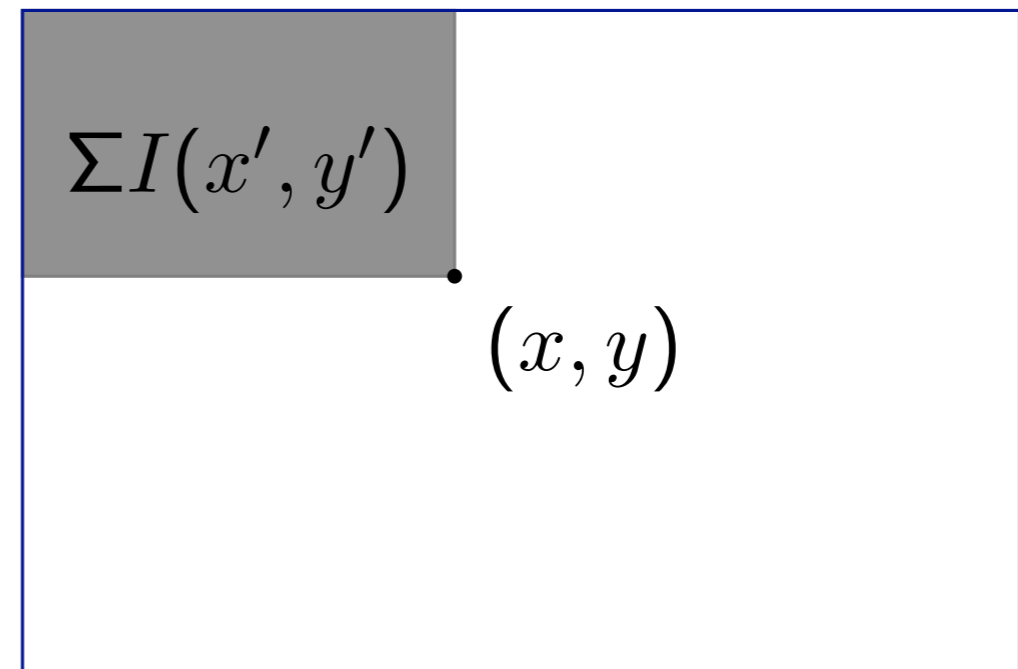


(Black)

Why do this?

- We need to compute the box filter values many, many times and we must do it very fast!

$$II(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

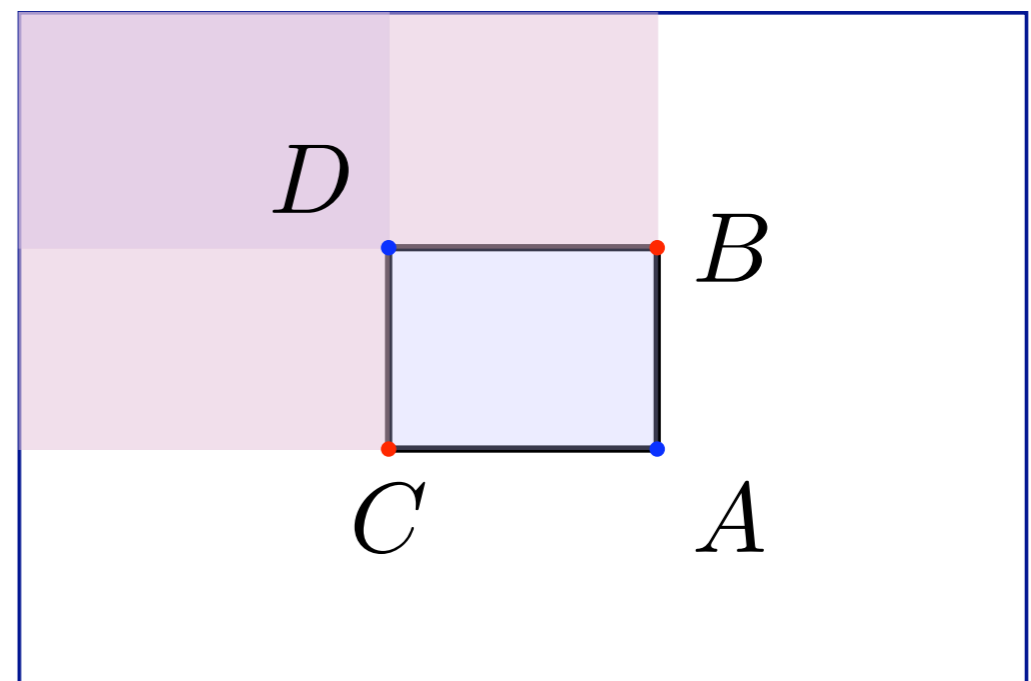


(Black)

Computing Integral Images

- Computing sum of pixels in a rectangular area:

$$f(A) = II(A) - II(B) - II(C) + II(D)$$



- A 3 box filter array  takes only 8 lookups.

(Black)

Computational Comparison

- To evaluate a 24x24 region of an image using pixels with a linear classifier takes 576 lookups and flops.
- Conversely, to evaluate a 24x24 region with a 3 box filter classifier takes 8 lookups and flops.
- Technique provides a method to do even faster ELS.

False detections Detector	10	31	50	65	78	95	110	167	422
Viola-Jones	78.3%	85.2%	88.8%	89.8%	90.1%	90.8%	91.1%	91.8%	93.7%
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	-	90.1%	89.9%

“Evaluation reported on the MIT-CMU Face Database”

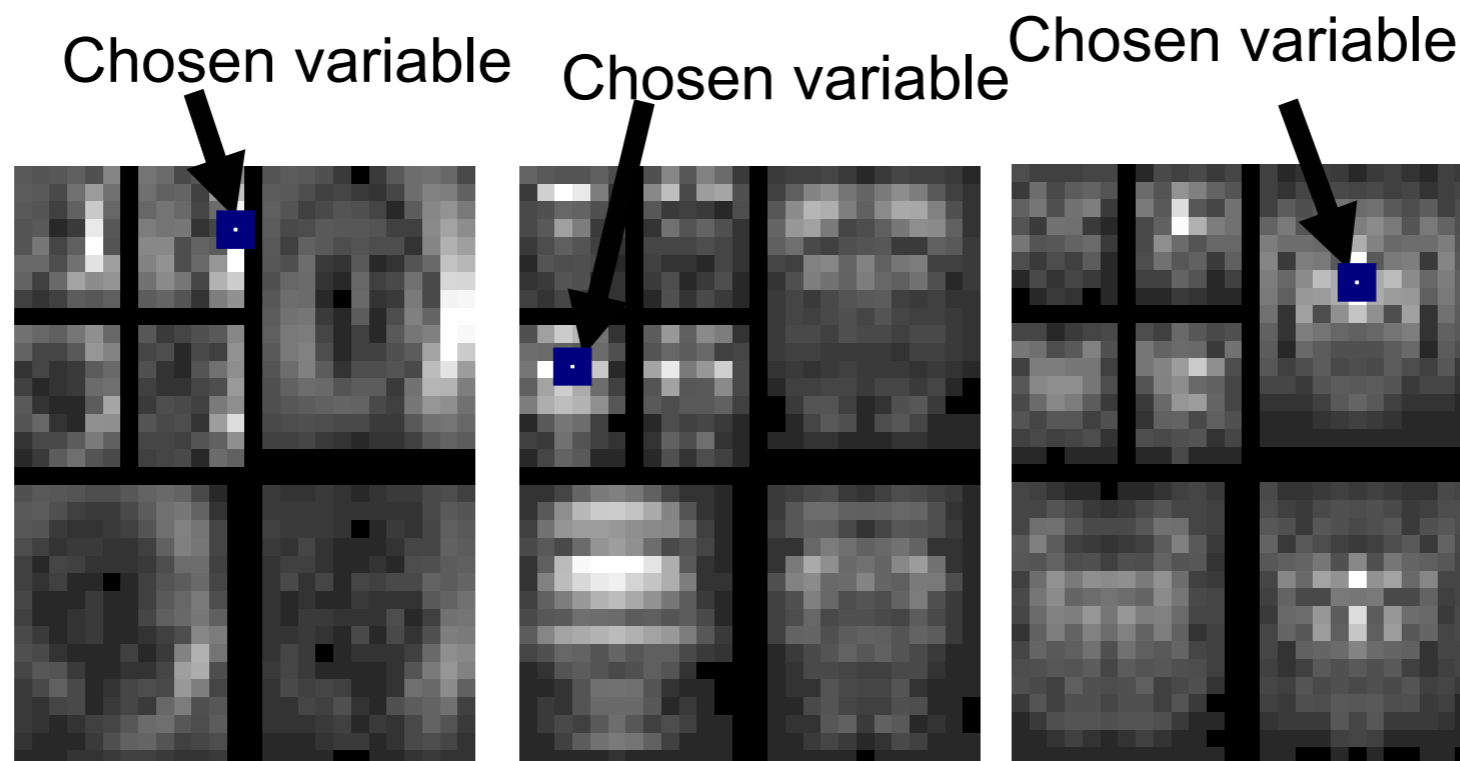
(Viola & Jones)

Examples of Detections



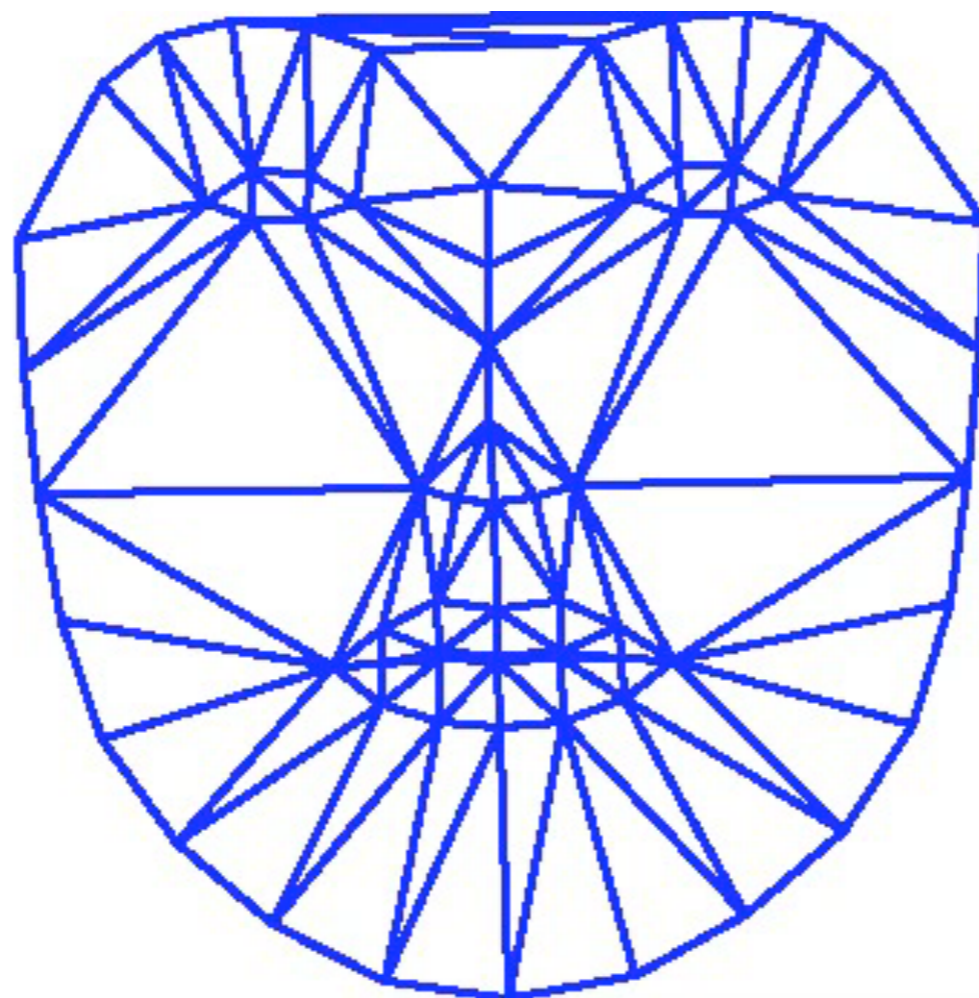
Recent Improvements

- More recently Schneiderman (2004) reported improved results using a Semi-Naive Bayesian classifier.
- Technique used a cascade of box filters, but used a mutual information (MI) to group features into independent sets.



Question?

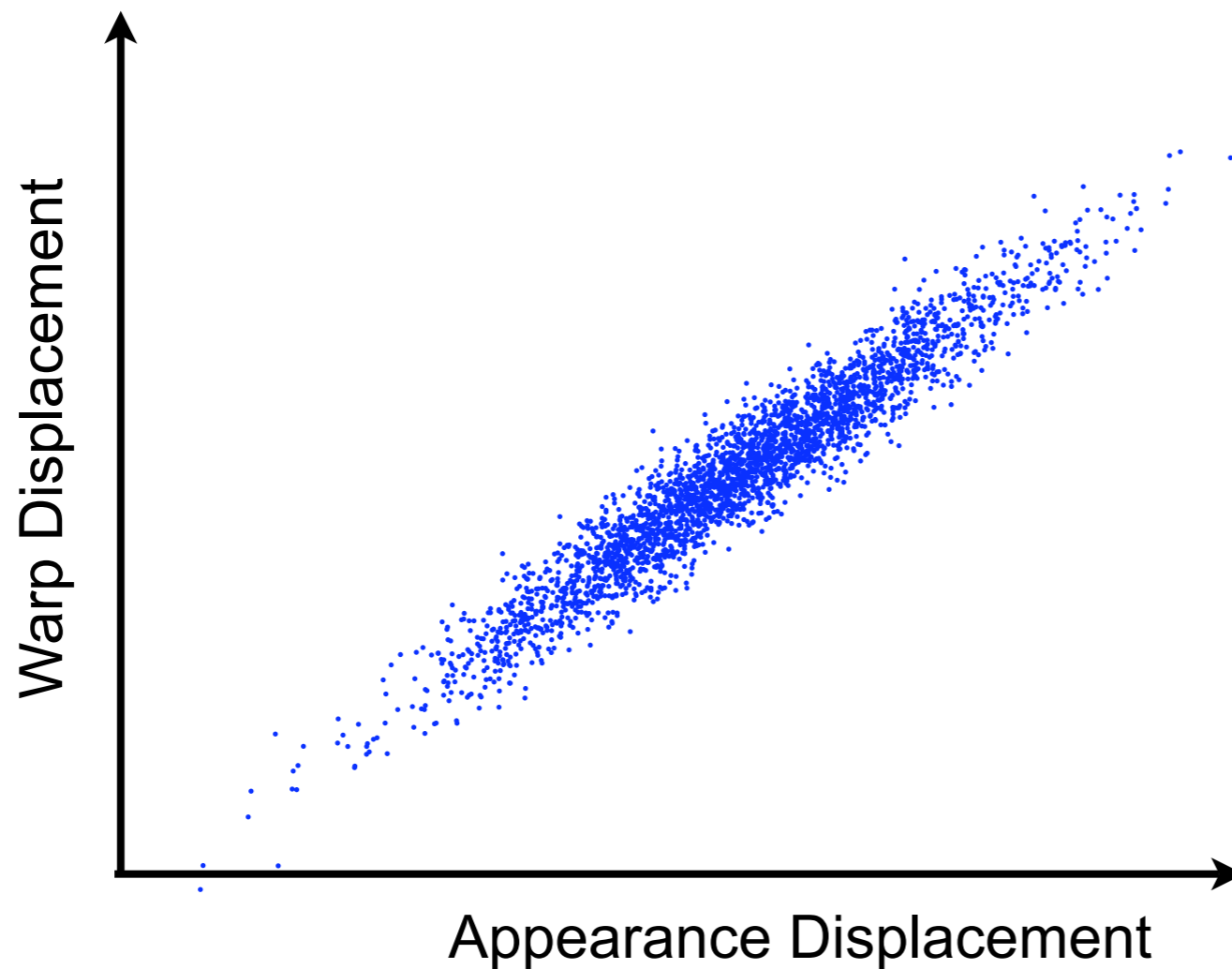
- ELS is feasible for warps in 3-4 dimensional space (e.g., translation, scale and perhaps rotation).
- What happens if we want finer alignment than this?



“Is there an alternative to ELS?”

Alternative?

- Is there perhaps some relationship between,



“Stay Tuned”

Object Registration and Tracking from a Learning Perspective (Part 3)

Simon Lucey
The Robotics Institute, Carnegie Mellon University

Overall Course Outline

- Lecture 2

- Discriminative Models for ES Registration
- Efficient ES using FFT.
 - Correlation Filters.
 - Support Vector Machines
 - Neural Networks
- Efficient ES using integral images
 - Adaboost
 - Mutual Information
- Speed and performance comparisons.

- Lecture 3

- Gradient Search (GS)
- Generative models for GS
- Lucas-Kanade, Black-Jepson algorithms
- Speeding up GS through Inverse Composition

Two Problems in Registration

1. *Learning*,

- How do I learn an object template/model that satisfactorily discriminates between the object and the image background?

(We might have to revisit this?)

2. *Fitting*,

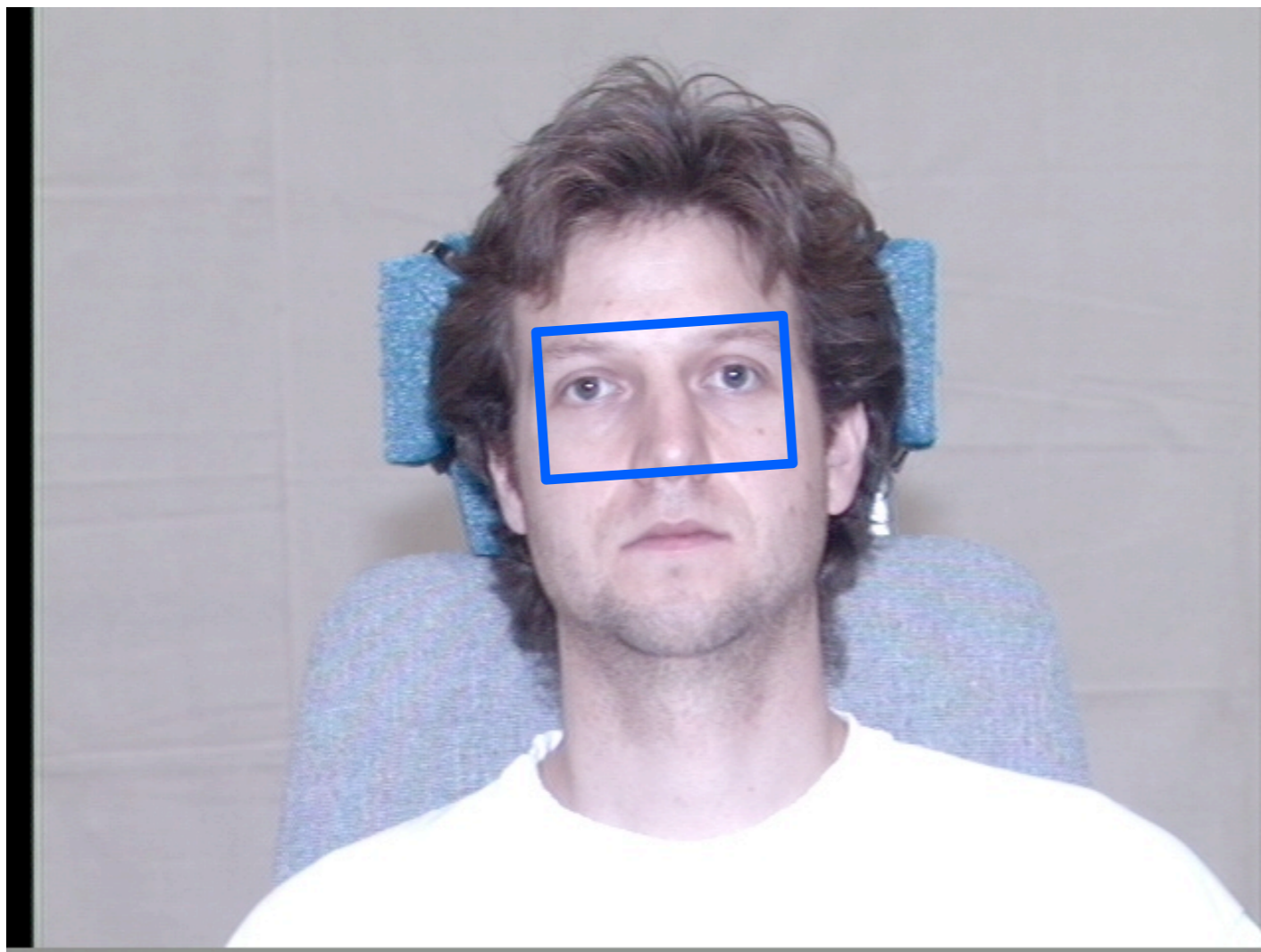
- How do I efficiently evaluate the object template/model across all possible warp values?

(Is there an alternative to ES?)

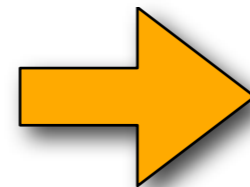
“As we will show through this course, both questions are linked!!!”

Naive Approach to Registration

- If you were a person coming straight from machine learning you might suggest,



“Images of Object at various warps”



[255,134,45,.....,34,12,124,67]
[123,244,12,.....,134,122,24,02]
[67,13,245,.....,112,51,92,181]
⋮
[65,09,67,.....,78,66,76,215]

“Vectors of pixel values at
each warp position”

Naive Approach to Registration

- If you were a person coming straight from machine learning you might suggest,

[255,134,45,.....,34,12,124,67]

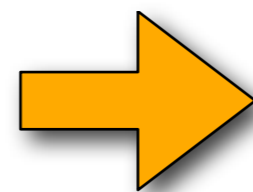
[123,244,12,.....,134,122,24,02]

[67,13,245,.....,112,51,92,181]

⋮

[65,09,67,.....,78,66,76,215]

“Vectors of pixel values at
each warp position”



$f(\text{teal bar})$

“classifier”

$\begin{matrix} \text{object} \\ \geq \\ < \\ \text{background} \end{matrix} Th$

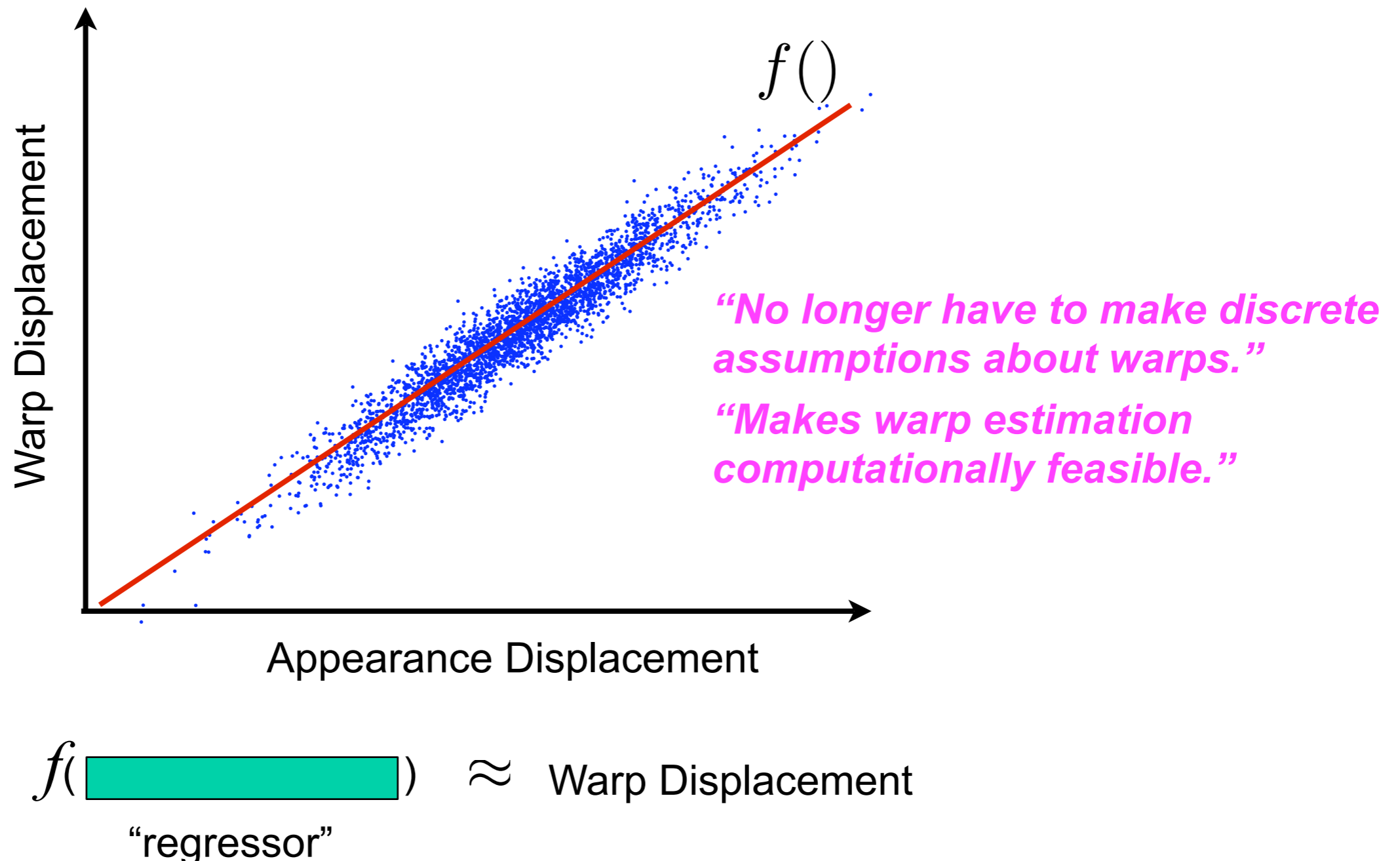
*“ES is fundamentally
computationally expensive.”*

Naive Approach to Registration



Slightly Less Naive Approach

- Instead of sampling through all possible warp positions to find the best match, let us instead learn a regression,



Problems?

- For us to learn this regression effectively we need to make a couple of assumptions.
 - What is the distribution $p(\Delta \mathbf{p})$ of warp displacements?
 - Is there a relationship between appearance displacement and warp displacement?
 - When does this relationship occur, when does it fail?



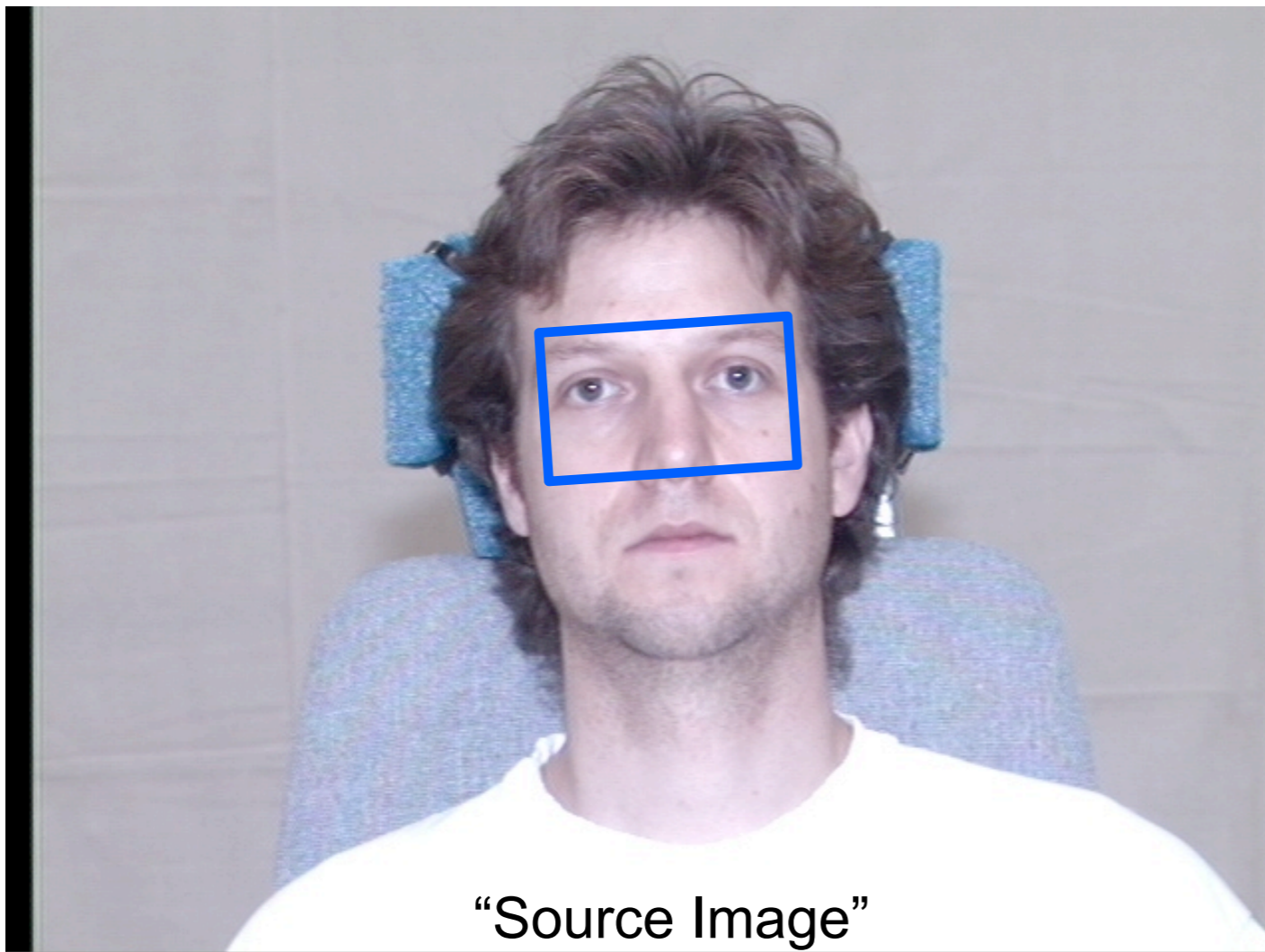
$$T(\mathcal{W}(\mathbf{z}; \mathbf{0}))$$



$$T(\mathcal{W}(\mathbf{z}; \Delta \mathbf{p}))$$

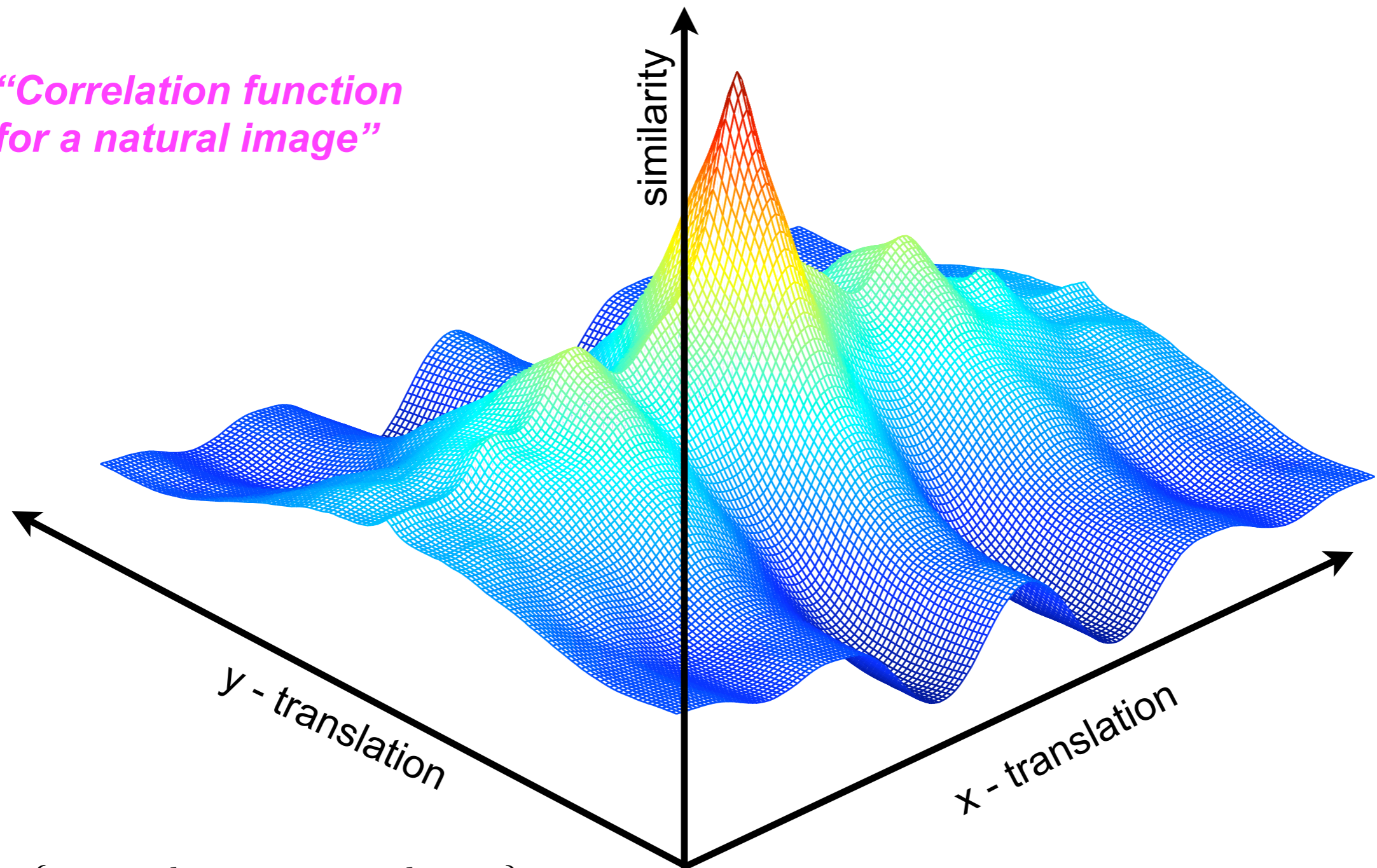
Reminder: Pixel Coherence Assumption

- The pixel similarity between one warp position in an image and another warp position degrades gracefully as a function of distance.



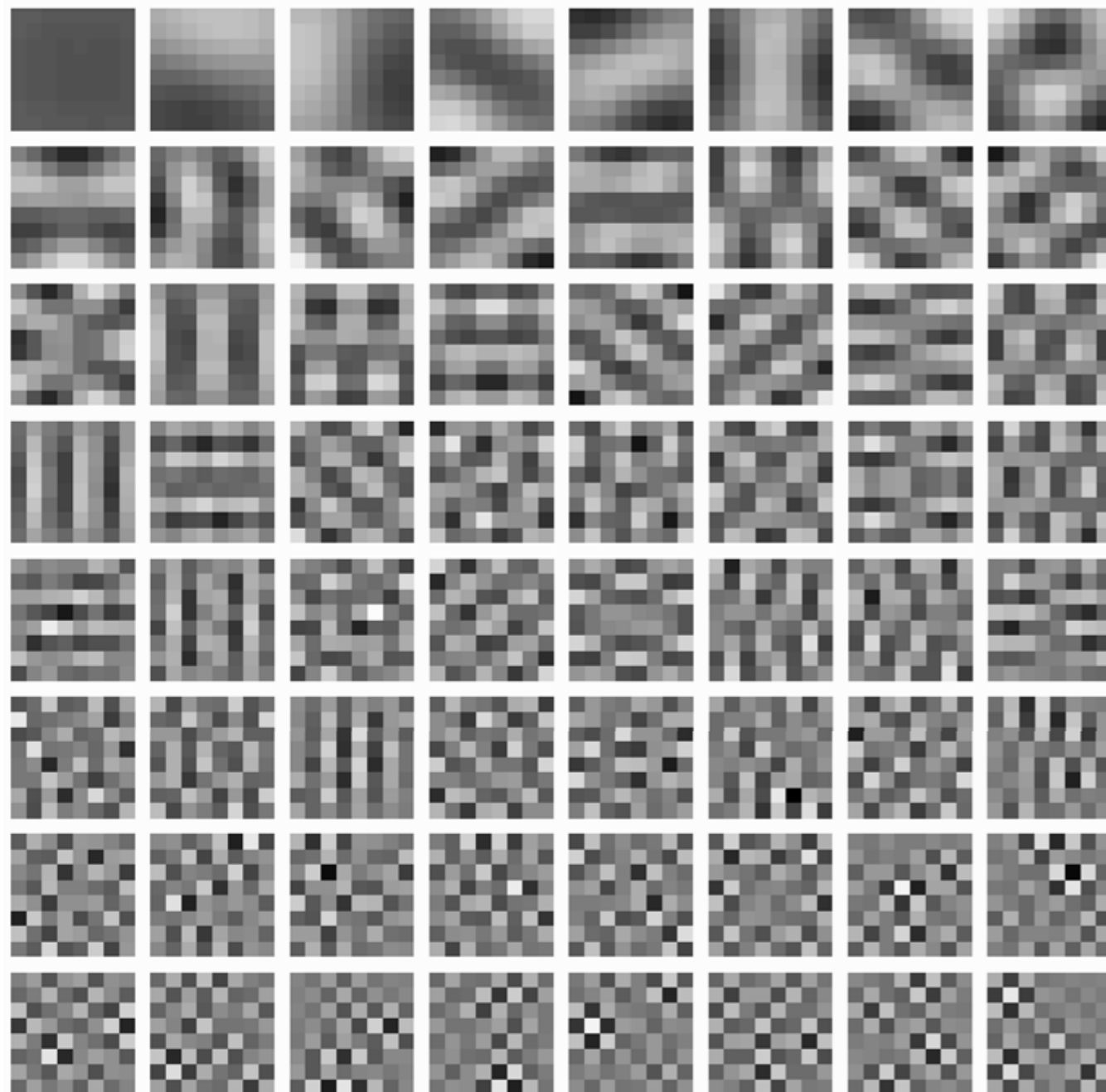
Reminder: Pixel Coherence Assumption

*“Correlation function
for a natural image”*



$$\mathbf{p} = \{\text{x-translation}, \text{y-translation}\}$$

Reminder: Eigen-Patches



- Does this look familiar?
 - 8x8 patches
- DCT2 - basis image set.
 - Forms the basis of modern-day image and video coding.
- Pixels in natural images are naturally correlated with one another.

(Szeliski and Fleet)

Pixel Coherence

- The pixel coherence assumption demonstrates that pixels within natural images are heavily correlated with one another within a local neighborhood \mathcal{N} (e.g., ± 5 pixels).
- For example,

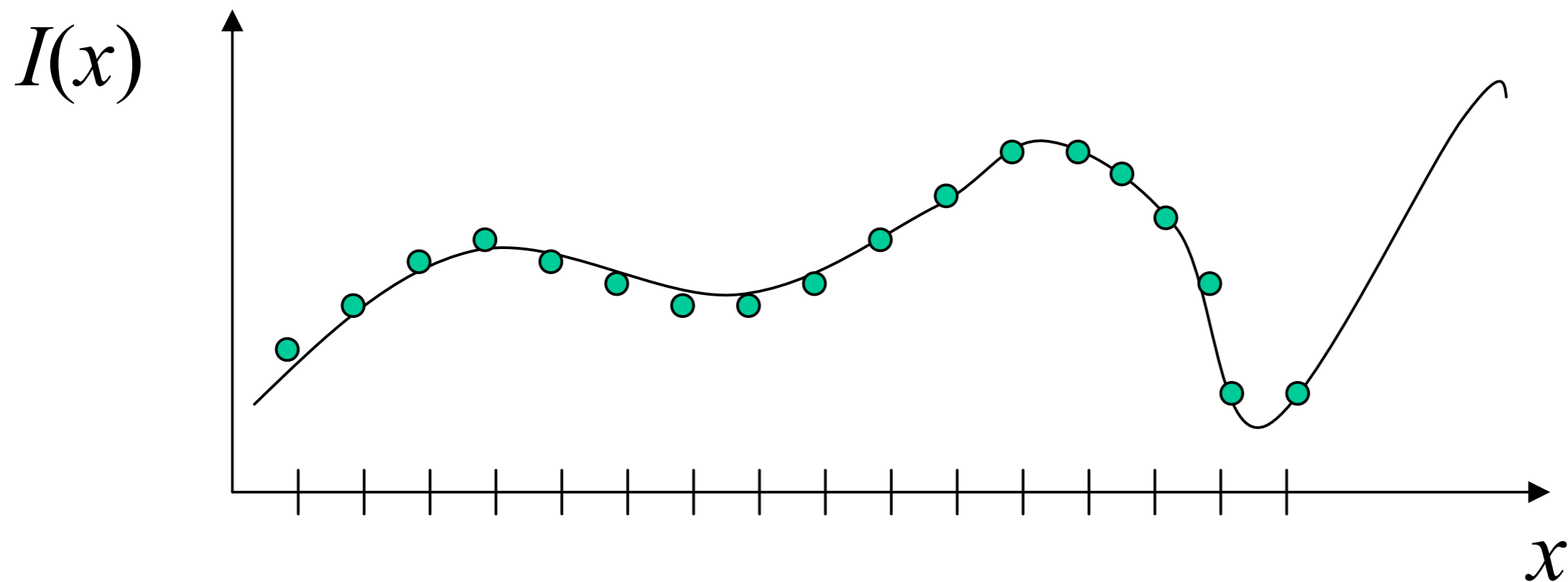
3	4	3
2	?	5
5	4	2

“Typically assume that pixels within $\pm 3, 5$ or 7 pixels are highly correlated.”

3

Estimating Gradients

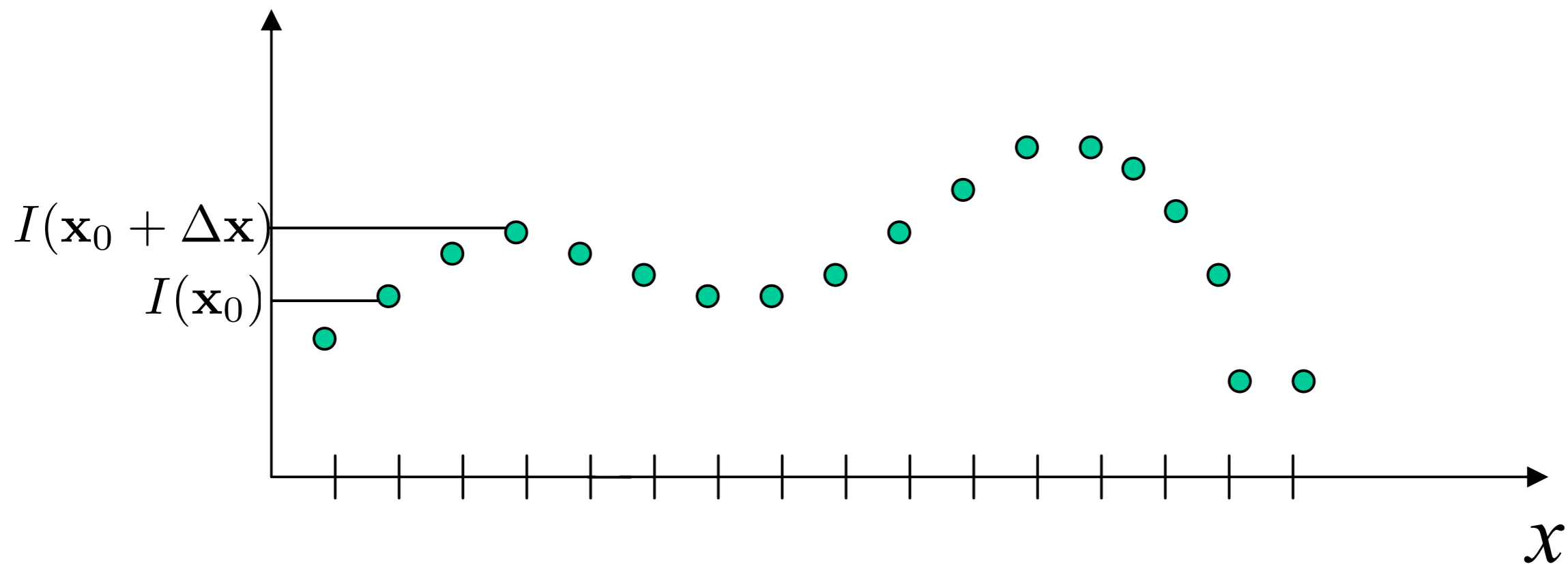
- Images are a discretely sampled representation of a continuous signal,



(Black)

Estimating Gradients

- What if I want to know Δx given that I have only the appearance at $I(x_0)$ and $I(x_0 + \Delta x)$?

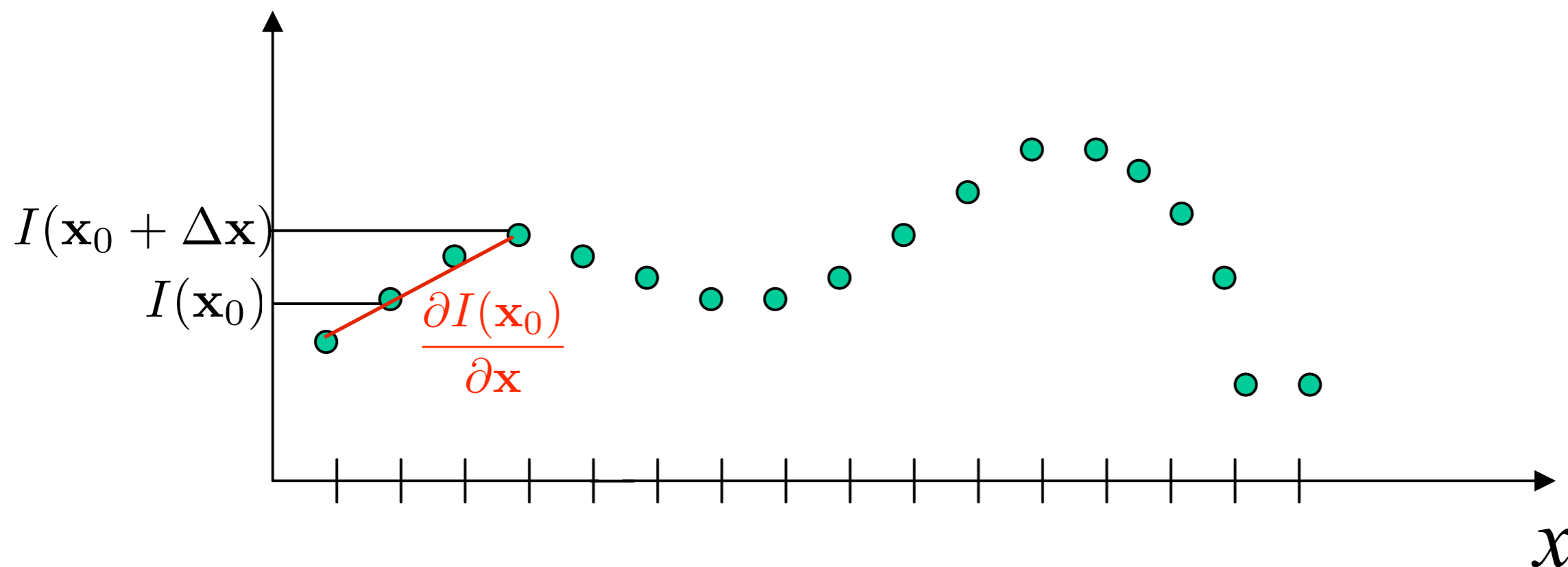


(Black)

Estimating Gradients

- Again, simply take the Taylor series approximation?

$$I(\mathbf{x}_0 + \Delta \mathbf{x}) \approx I(\mathbf{x}_0) + \frac{\partial I(\mathbf{x}_0)}{\partial \mathbf{x}}^T \Delta \mathbf{x}$$



(Black)

Estimating Gradients

- Again, simply take the Taylor series approximation,

$$I(\mathbf{x}_0 + \Delta\mathbf{x}) \approx I(\mathbf{x}_0) + \frac{\partial I(\mathbf{x}_0)}{\partial \mathbf{x}}^T \Delta\mathbf{x}$$

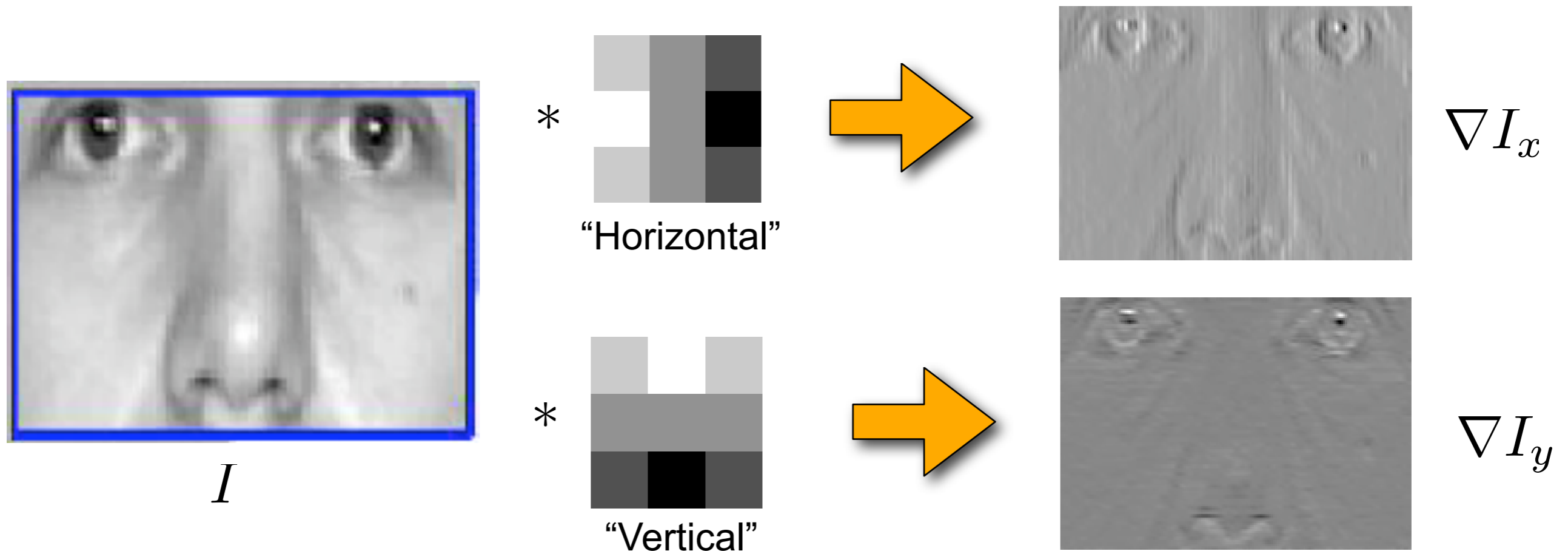
- Therefore,

$$\Delta\mathbf{x} \approx \left(\frac{\partial I(\mathbf{x}_0)}{\partial \mathbf{x}} \frac{\partial I(\mathbf{x}_0)}{\partial \mathbf{x}}^T \right)^{-1} \frac{\partial I(\mathbf{x}_0)}{\partial \mathbf{x}} [I(\mathbf{x}_0 + \Delta\mathbf{x}) - I(\mathbf{x}_0)]$$

- We refer to $\frac{\partial I(\mathbf{x}_0)}{\partial \mathbf{x}}$ as the gradient of the image at \mathbf{x}_0 .

Gradients through Filters

- Traditional method for calculating gradients in vision is through the use of edge filters. (e.g., Sobel, Prewitt).



- where,
$$\frac{\partial I(\mathbf{x})}{\partial \mathbf{x}} = [\nabla I_x(\mathbf{x}), \nabla I_y(\mathbf{x})]^T$$

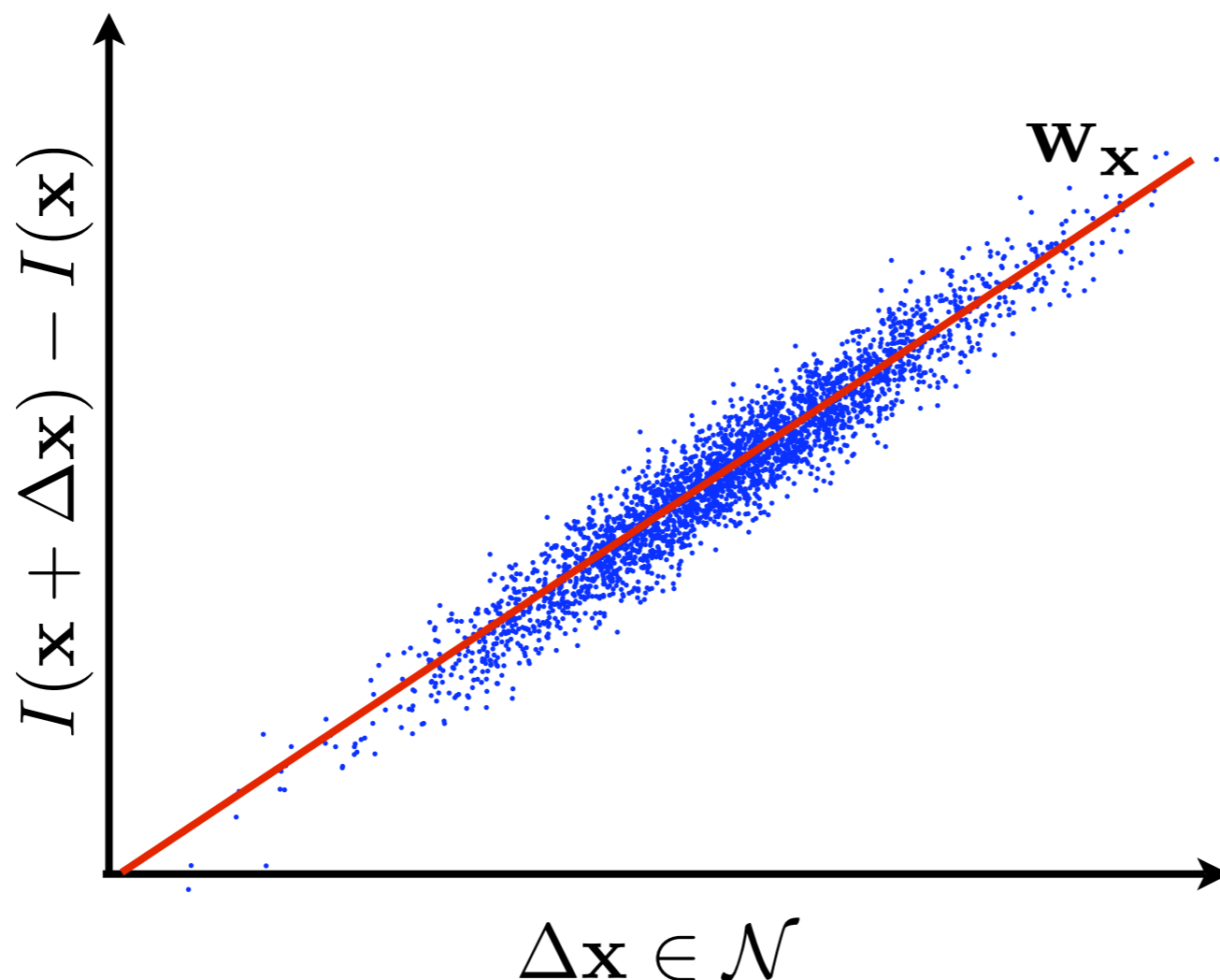
"Often have to apply a smoothing filter as well."

Gradients through Regression

- Another strategy is to learn a least-squares regression for every pixel in the source image such that,

$$I(\mathbf{x} + \Delta\mathbf{x}) \approx I(\mathbf{x}) + \mathbf{w}_{\mathbf{x}}^T \Delta\mathbf{x} + \textcircled{b}$$

Why does b have to equal zero?



Gradients through Regression

- Can be written formally as,

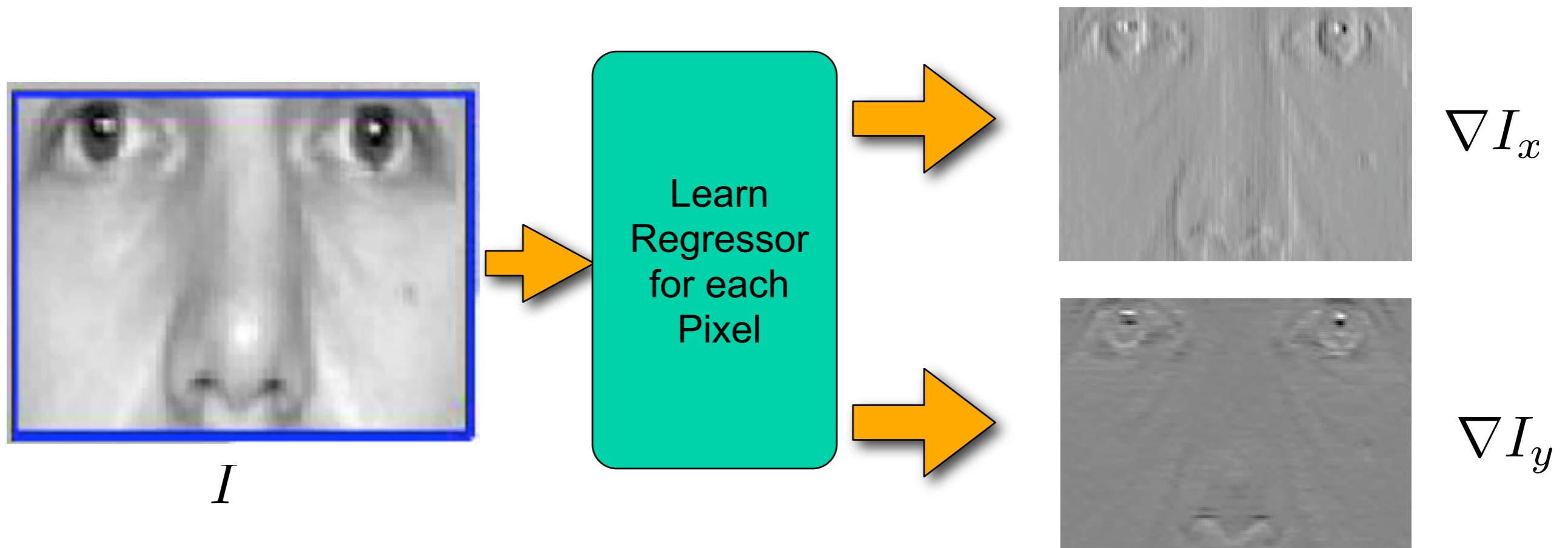
$$\arg \min_{\mathbf{w}_{\mathbf{x}}} \sum_{\Delta \mathbf{x} \in \mathcal{N}} ||I(\mathbf{x} + \Delta \mathbf{x}) - I(\mathbf{x}) - \mathbf{w}_{\mathbf{x}}^T \Delta \mathbf{x}||^2$$

- Solution is given as,

$$\mathbf{w}_{\mathbf{x}} = \left(\sum_{\Delta \mathbf{x} \in \mathcal{N}} \Delta \mathbf{x} \Delta \mathbf{x}^T \right)^{-1} \left(\sum_{\Delta \mathbf{x} \in \mathcal{N}} \Delta \mathbf{x} [I(\mathbf{x} + \Delta \mathbf{x}) - I(\mathbf{x})] \right)$$

Gradients through Regression

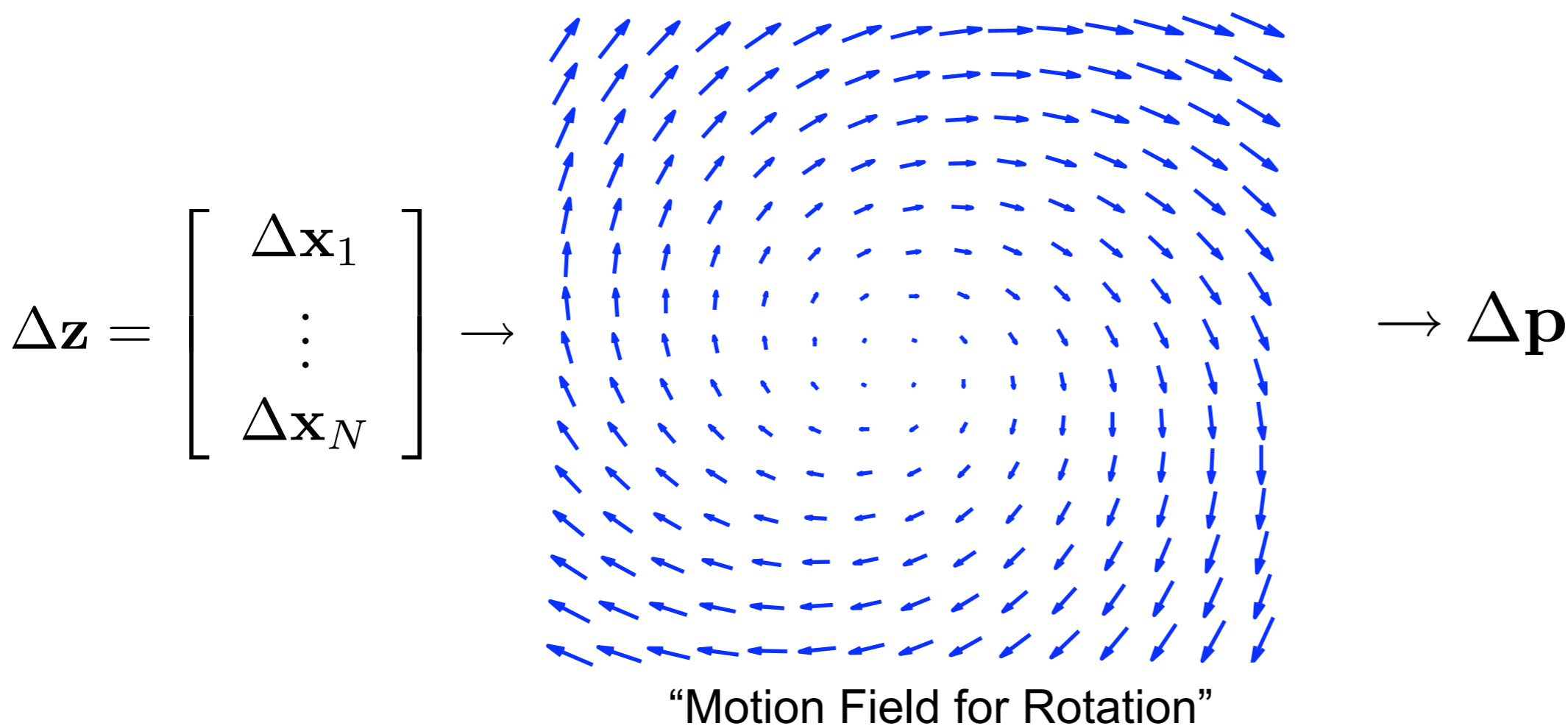
- Can solve for gradients using least-squares regression.
- Has nice properties: expand neighborhood, no heuristics.



- where,
$$\frac{\partial I(\mathbf{x})}{\partial \mathbf{x}} = [\nabla I_x(\mathbf{x}), \nabla I_y(\mathbf{x})]^T = [w_x, w_y]^T$$

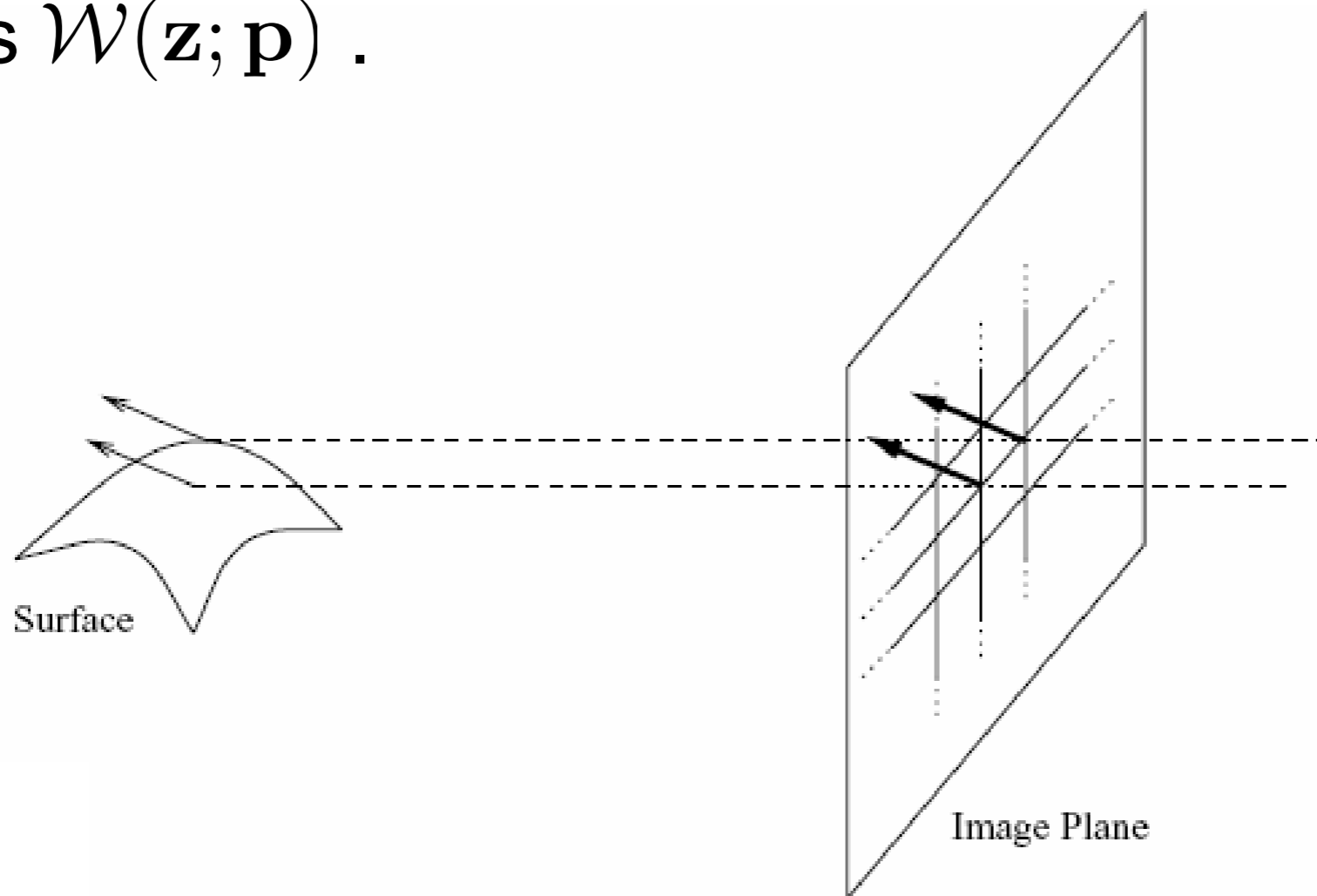
Possible Solution?

- Could we now just solve for individual pixel translation, and then estimate a complex warp from these motions?



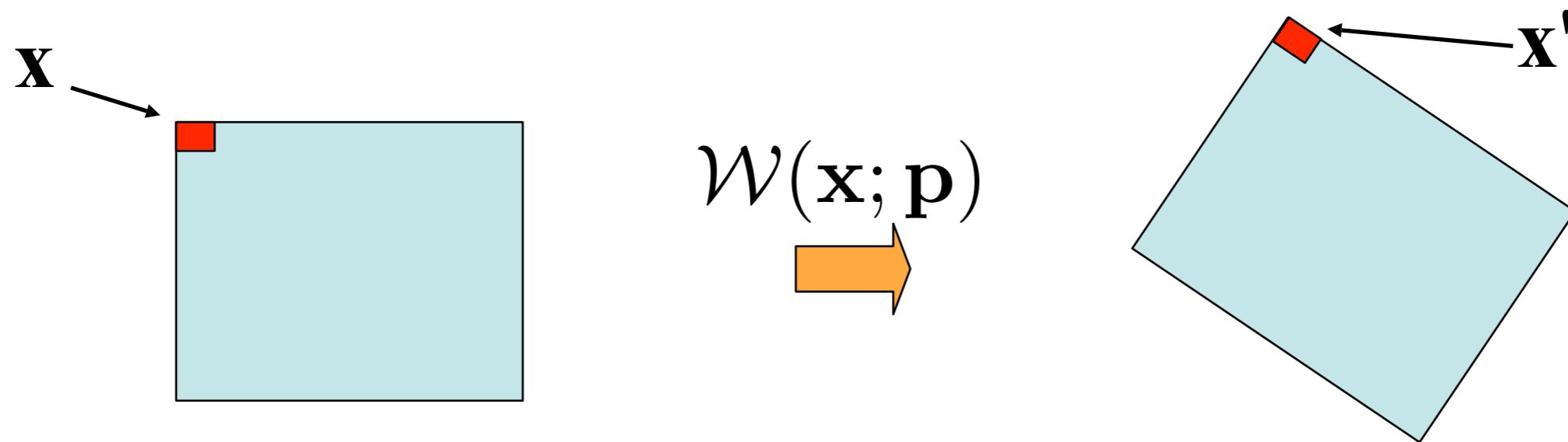
Spatial Coherence

- Problem is pixel noise. Individual pixels are too noisy to be reliable estimators of pixel movement (optical flow).
- Fortunately, neighboring points in the scene typically belong to the same surface and hence typically have similar motions $\mathcal{W}(\mathbf{z}; \mathbf{p})$.



Reminder: Warp Functions

- To perform alignment we need a formal way of describing how the template relates to the source image.
- To do this we can employ what is known as a warp function:-



$$\mathbf{z} = [\mathbf{x}_1^T, \dots, \mathbf{x}_N^T]^T$$

where,

\mathbf{x}_i = i -th 2D coordinate

\mathbf{p} = parametric form of warp

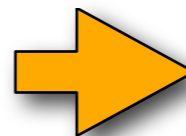
\mathbf{z} = concatenation of all points in the template

Lucas-Kanade Algorithm

- Lucas & Kanade (1980) realized this and proposed a method for estimating warp displacement using the principles of ***gradients*** and ***spatial coherence***.
- Technique applies Taylor series approximation to any spatially coherent area governed by the warp $\mathcal{W}(\mathbf{z}; \mathbf{p})$.

$$I(\mathcal{W}(\mathbf{z}; \mathbf{p} + \Delta\mathbf{p})) \approx I(\mathcal{W}(\mathbf{z}; \mathbf{p})) + \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}^T \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}{\partial \mathbf{p}}^T \Delta\mathbf{p}$$

“We consider this image to always be static....”



Lucas-Kanade Algorithm

- Lucas & Kanade (1980) realized this and proposed a method for estimating warp displacement using the principles of ***gradients*** and ***spatial coherence***.
- Technique applies Taylor series approximation to any spatially coherent area governed by the warp $\mathcal{W}(\mathbf{z}; \mathbf{p})$.

$$I(\mathcal{W}(\mathbf{z}; \mathbf{p} + \Delta\mathbf{p})) \approx \boxed{I(\mathcal{W}(\mathbf{z}; \mathbf{p}))} + \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}^T \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}{\partial \mathbf{p}}^T \Delta\mathbf{p}$$



Lucas-Kanade Algorithm

- Lucas & Kanade (1980) realized this and proposed a method for estimating warp displacement using the principles of ***gradients*** and ***spatial coherence***.
- Technique applies Taylor series approximation to any spatially coherent area governed by the warp $\mathcal{W}(\mathbf{z}; \mathbf{p})$.

$$I(\mathcal{W}(\mathbf{z}; \mathbf{p} + \Delta \mathbf{p})) \approx I(\mathcal{W}(\mathbf{z}; \mathbf{p})) + \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}^T \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}{\partial \mathbf{p}}^T \Delta \mathbf{p}$$



∇I_y

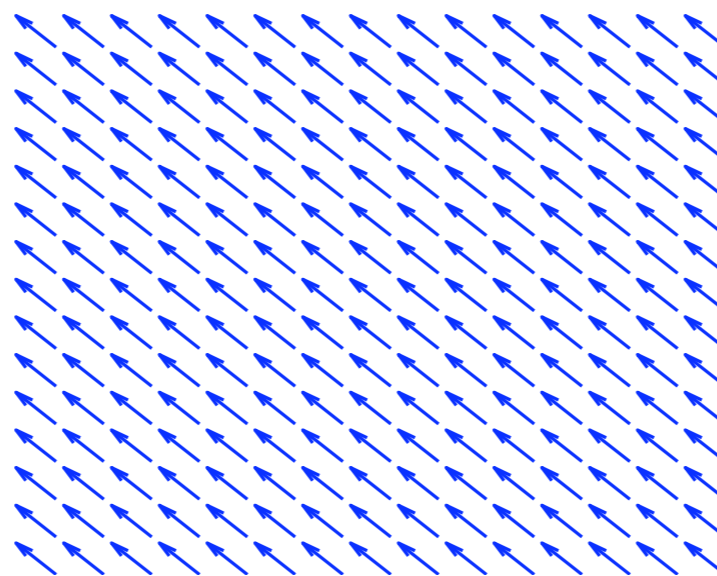


∇I_x

Lucas-Kanade Algorithm

- Lucas & Kanade (1980) realized this and proposed a method for estimating warp displacement using the principles of ***gradients*** and ***spatial coherence***.
- Technique applies Taylor series approximation to any spatially coherent area governed by the warp $\mathcal{W}(\mathbf{z}; \mathbf{p})$.

$$I(\mathcal{W}(\mathbf{z}; \mathbf{p} + \Delta\mathbf{p})) \approx I(\mathcal{W}(\mathbf{z}; \mathbf{p})) + \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}^T \boxed{\frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}{\partial \mathbf{p}}^T} \Delta\mathbf{p}$$



Lucas-Kanade Algorithm

- Lucas & Kanade (1980) realized this and proposed a method for estimating warp displacement using the principles of ***gradients*** and ***spatial coherence***.
- Technique applies Taylor series approximation to any spatially coherent area governed by the warp $\mathcal{W}(\mathbf{z}; \mathbf{p})$.

$$I(\mathcal{W}(\mathbf{z}; \mathbf{p} + \Delta \mathbf{p})) \approx I(\mathcal{W}(\mathbf{z}; \mathbf{p})) + \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathbf{p}} \Delta \mathbf{p}$$

Diagram illustrating the dimensions of the terms in the Taylor series approximation:

- $I(\mathcal{W}(\mathbf{z}; \mathbf{p} + \Delta \mathbf{p}))$ has dimension $N \times 1$.
- $I(\mathcal{W}(\mathbf{z}; \mathbf{p}))$ has dimension $N \times 1$.
- $\frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}$ has dimension $N \times 2N$.
- $\frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathbf{p}}$ has dimension $2N \times K$.

N = number of pixels

K = number of warp parameters

Lucas-Kanade Algorithm

- Often just refer to,

$$\mathbf{J} = \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}^T \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}{\partial \mathbf{p}}^T$$

as the “***Jacobian***” matrix.

- Also refer to,

$$\mathbf{H} = \mathbf{J}^T \mathbf{J}$$

as the “***pseudo-Hessian***”.

- Finally, we can refer to,

$$T(\mathbf{z}) = I(\mathcal{W}(\mathbf{z}; \mathbf{p} + \Delta \mathbf{p}))$$

as the “***template***”.

Lucas-Kanade Algorithm

- Actual algorithm is just the application of the following steps,

Step 1:

$$\Delta \mathbf{p} = \mathbf{H}^{-1} \mathbf{J}^T [I(\mathcal{W}(\mathbf{z}; \mathbf{p})) - T(\mathbf{z})]$$

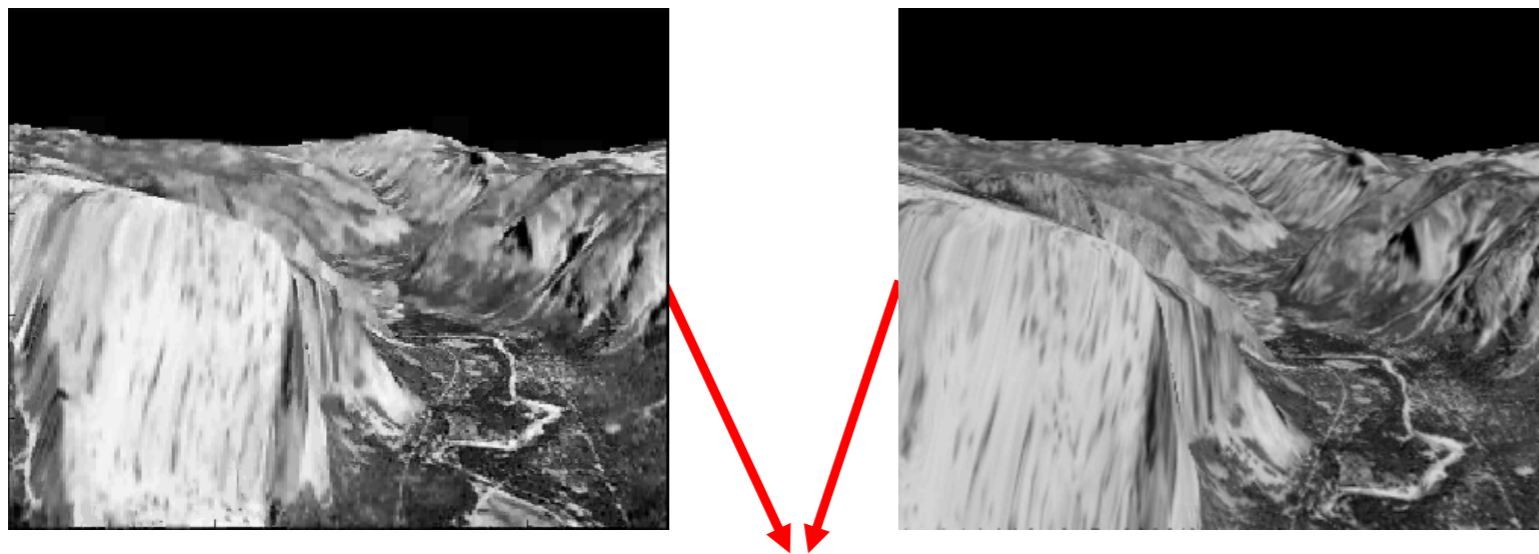
Step 2:

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$$

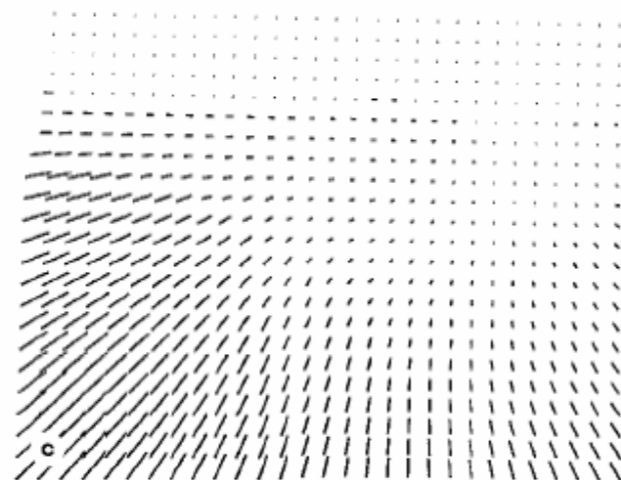
keep applying steps until $\Delta \mathbf{p}$ converges.

LK for Optical Flow

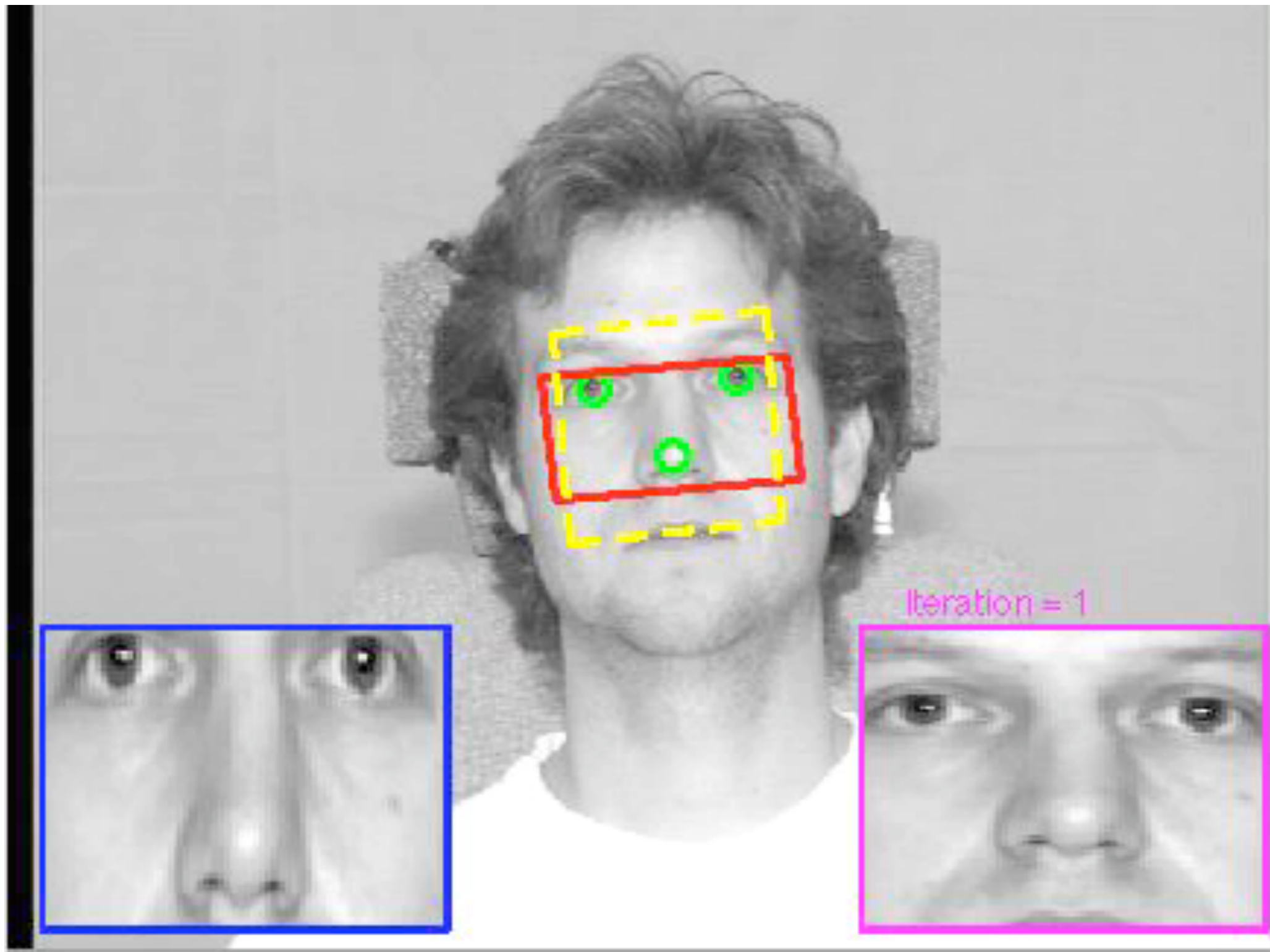
- Optical flow is typically calculated from a local region of pixels (i.e., patch) using LK.
- Warp function used is normally just translation.



“Extremely useful for gaining stereo vision.”



Examples of LK Alignment



Optimization Interpretation

- The optimization employed with the LK algorithm can be interpreted as Gauss-Newton optimization.
- Other non-linear least-squares optimization strategies have been investigated (Baker et al. 2003)
 - Levenberg-Marquadt
 - Newton
 - Steepest-Descent
- Gauss-Newton in empirical evaluations has appeared to be the most robust (Baker et al.).

Computation Concerns

- Unfortunately, the LK algorithm can still be computationally expensive.
 - Requires the re-computation of the Jacobian matrix.

$$\mathbf{J} = \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathbf{p}}$$

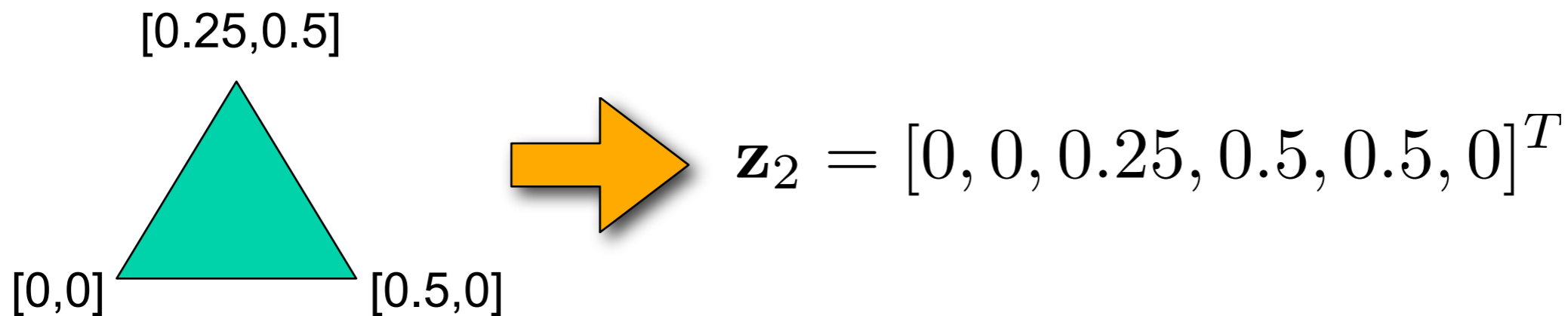
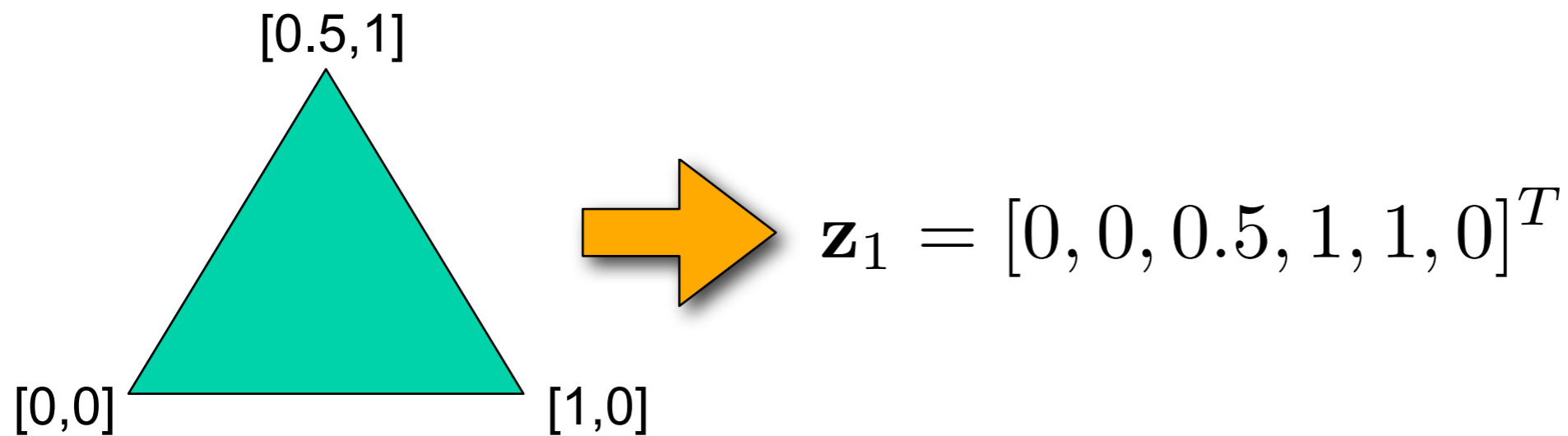
- With the additional inversion of the Hessian matrix at each iteration.

$$\mathbf{H} = \mathbf{J}^T \mathbf{J}$$

“Is there any way we can pre-compute any of this?”

Composite Warps

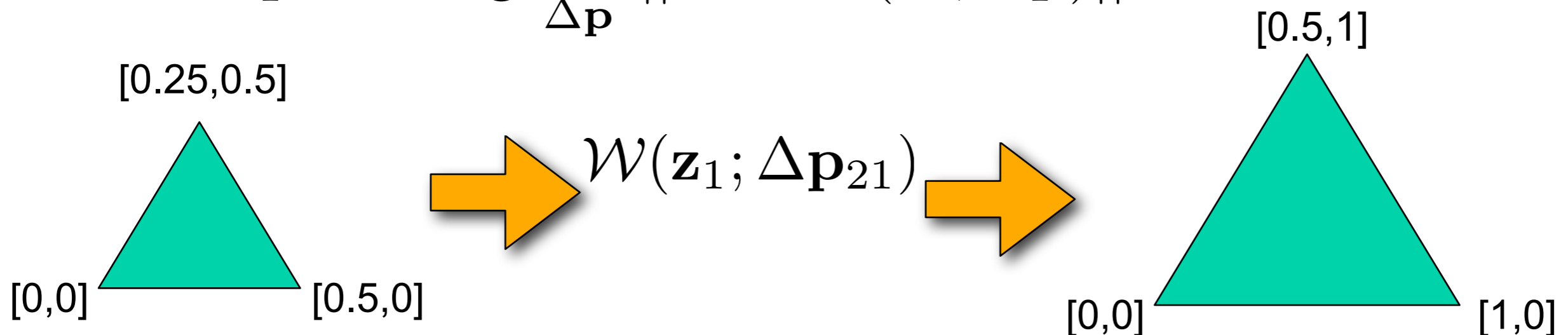
- Some warps (such as the affine warps and its subsets) can be composed.



Composite Warps

- Let us try and find the warp that maps from tri 2 to 1,

$$\Delta \mathbf{p}_{21} = \arg \min_{\Delta \mathbf{p}} ||\mathbf{z}_1 - \mathcal{W}(\mathbf{z}_2; \Delta \mathbf{p})||^2$$

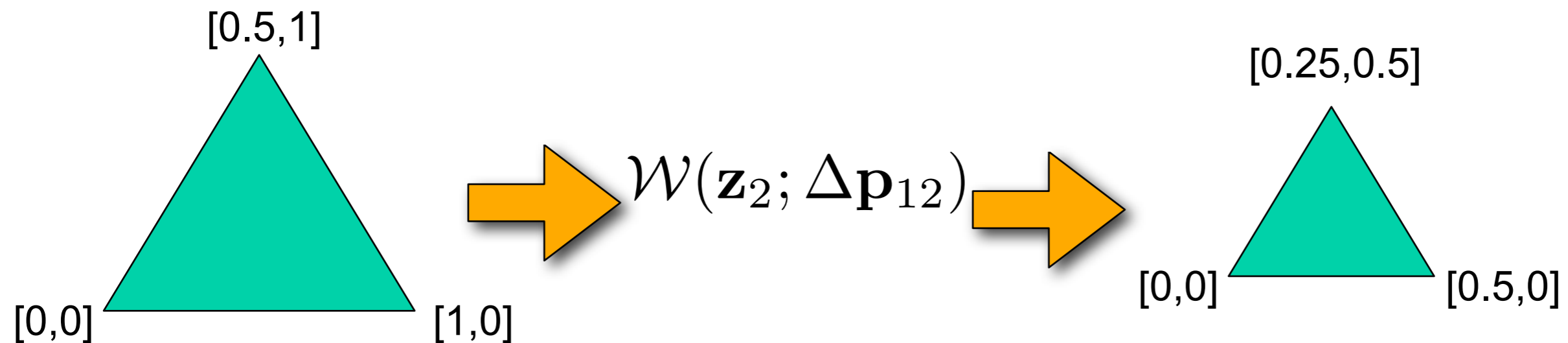


$$\begin{bmatrix} 0, & 0.25, & 0.5 \\ 0, & 0.5, & 0 \\ 1, & 1, & 1 \end{bmatrix} \mathbf{M}_{21} = \begin{bmatrix} 0, & 0.5, & 1 \\ 0, & 1, & 0 \\ 1, & 1, & 1 \end{bmatrix} \quad \text{"In matrix form."}$$

Composite Warps

- Similarly, find the warp that maps from tri 1 to 2,

$$\Delta \mathbf{p}_{12} = \arg \min_{\Delta \mathbf{p}} ||\mathbf{z}_2 - \mathcal{W}(\mathbf{z}_1; \Delta \mathbf{p})||^2$$



$$\begin{bmatrix} 0 & 0.5 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \mathbf{M}_{12} = \begin{bmatrix} 0 & 0.25 & 0.5 \\ 0 & 0.5 & 0 \\ 1 & 1 & 1 \end{bmatrix} \text{ “In matrix form.”}$$

Composite Warps

- For nearly all warps, except translation.

$$\mathbf{p}_{12} \neq -\mathbf{p}_{21}$$

- However, with an affine warp the following relationship does hold,

$$\mathbf{M}_{12} = \mathbf{M}_{21}^{-1}$$

- Or more generally,

$$\begin{aligned}\mathbf{z}_1 &= \mathcal{W}(\mathcal{W}(\mathbf{z}_1; \mathbf{p}_{12}); \mathbf{p}_{21}) \\ &= \mathcal{W}^{-1}(\mathcal{W}(\mathbf{z}_1; \mathbf{p}_{12}); \mathbf{p}_{12})\end{aligned}$$

- where $\mathcal{W}^{-1}()$ is the inverse warp.

Composite Warps

- Intuition: most warps contain scale.
- If, $b \times s = a$

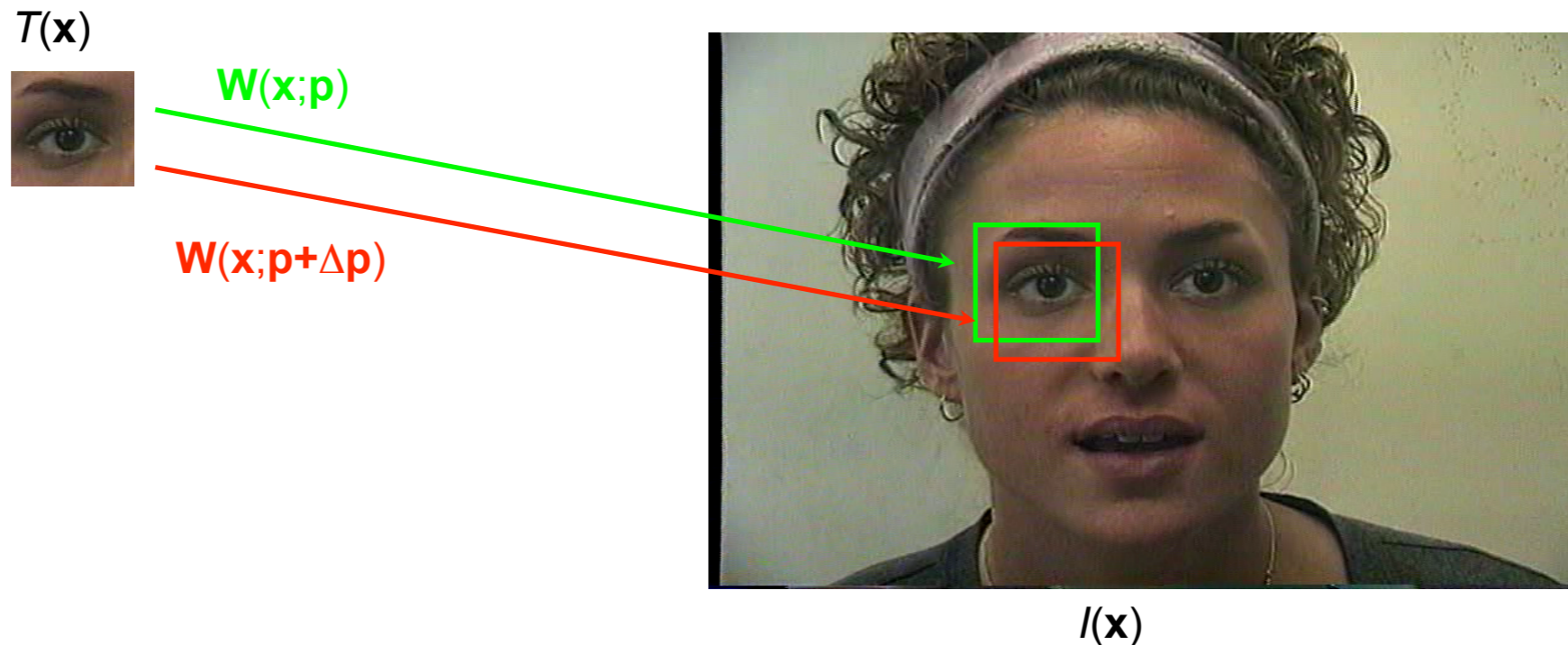
then, $a \times -s \neq b$

but,

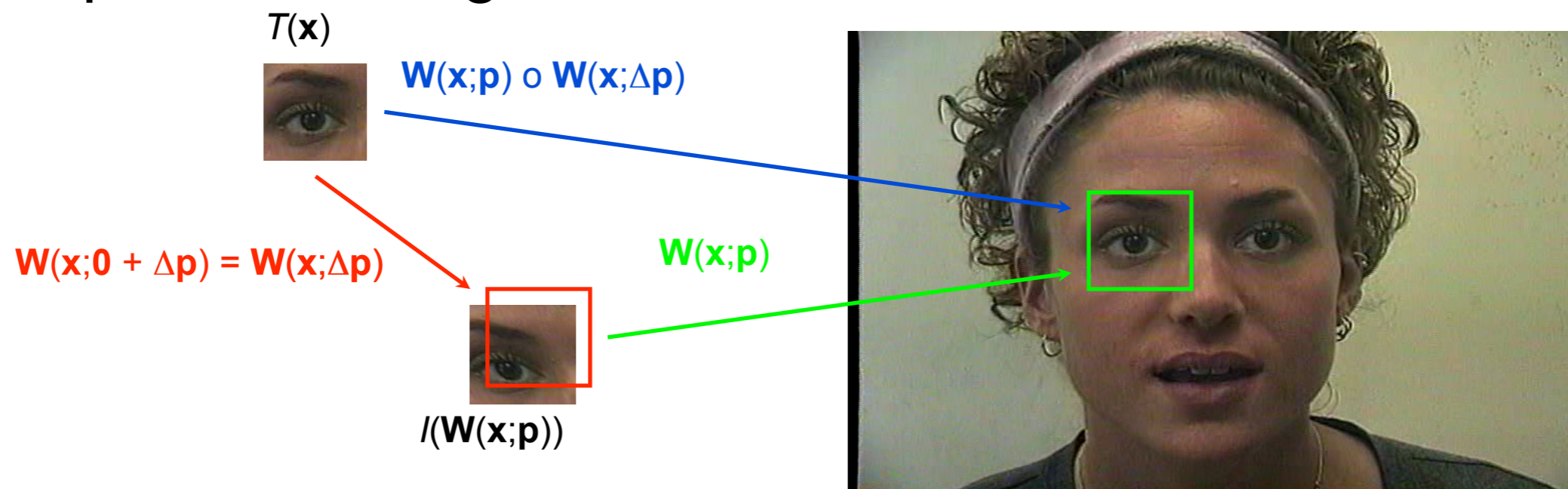
$$a \times s^{-1} = b$$

Compositional Image Alignment

- Additive Image Alignment – Lucas & Kanade

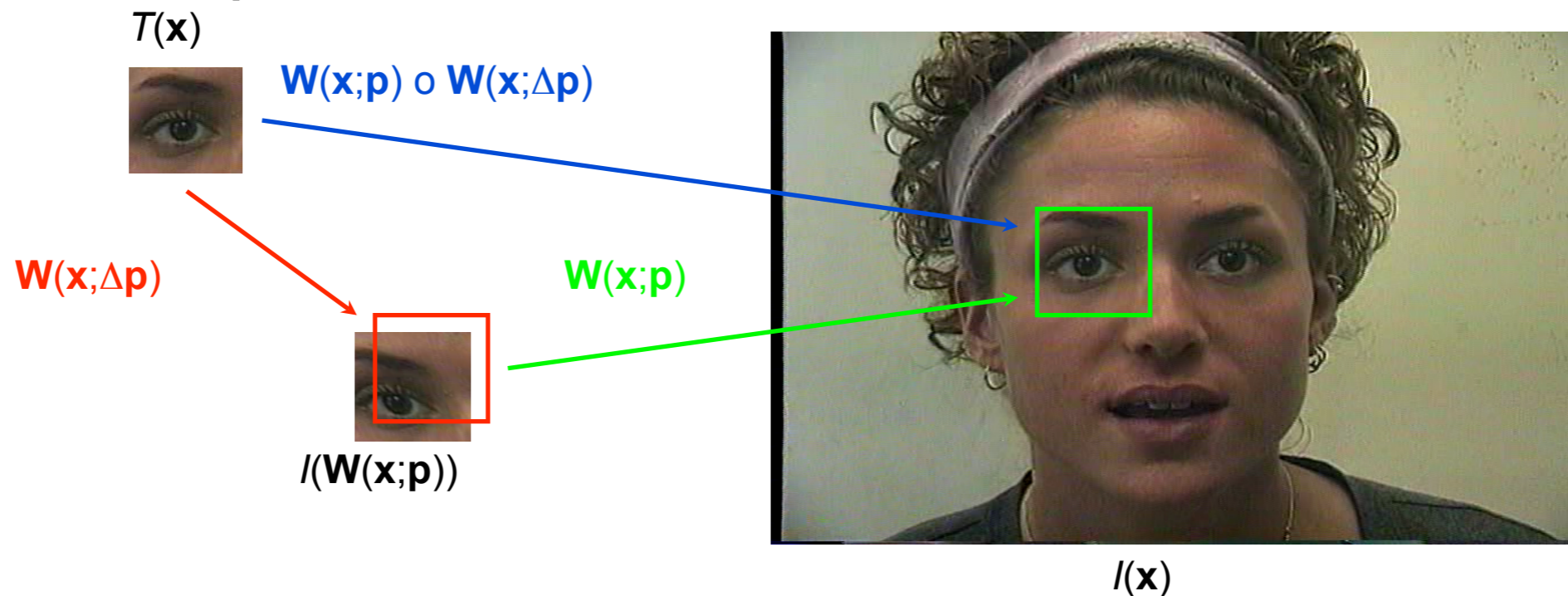


- Compositional Alignment – Shum & Szeliski

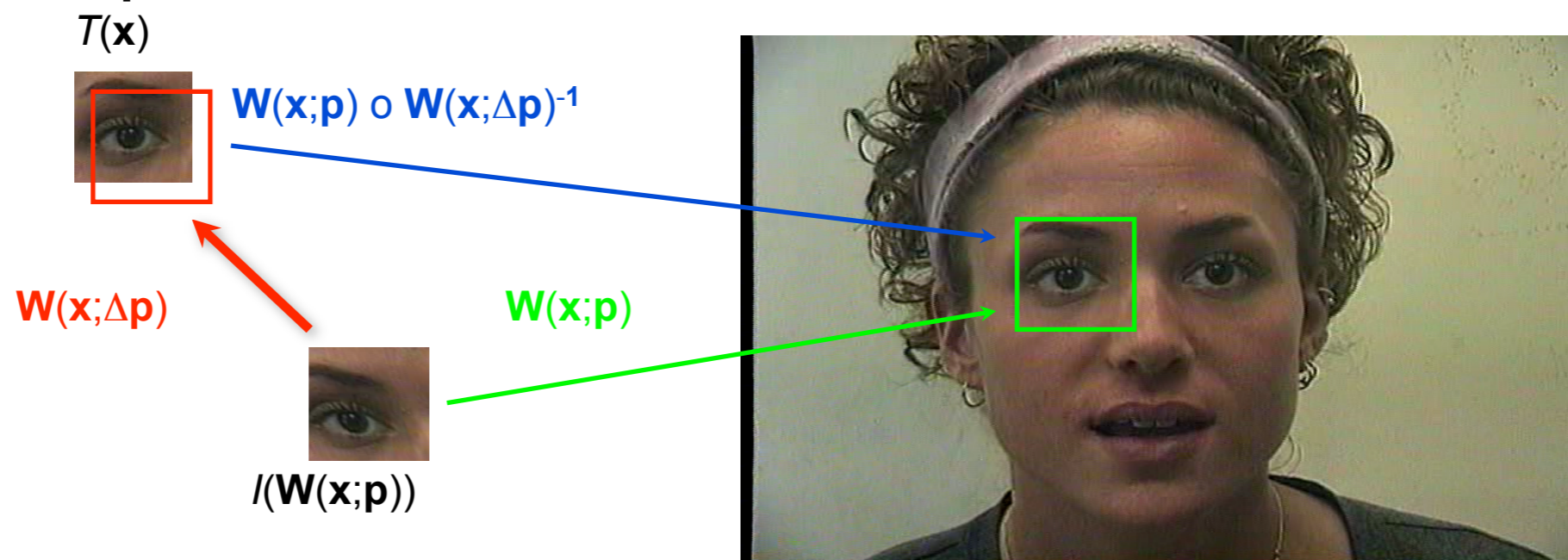


Inverse Compositional Alignment

- Forwards compositional



- Inverse compositional



Composite Warps in LK

- Shum & Szelski demonstrated that composite warps can be introduced into the LK algorithm.
- So we replace,

$$I(\mathcal{W}(\mathbf{z}; \mathbf{p} + \Delta\mathbf{p})) \approx I(\mathcal{W}(\mathbf{z}; \mathbf{p})) + \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathbf{p}} \Delta\mathbf{p}$$

“Forward Additive Approximation”

$$I(\mathcal{W}(\mathcal{W}(\mathbf{z}; \Delta\mathbf{p}); \mathbf{p})) \approx I(\mathcal{W}(\mathbf{z}; \mathbf{p})) + \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})^T}{\partial \mathbf{p}} \Delta\mathbf{p}$$

“Forward Composition Approximation”

Forward-Composition Algorithm

- Actual algorithm is just the application of the following steps,

Step 1:

$$\Delta \mathbf{p} = \mathbf{H}_{fc}^{-1} \mathbf{J}_{fc}^T [I(\mathcal{W}(\mathbf{z}; \mathbf{p})) - T(\mathbf{z})]$$

Step 2:

$$\mathcal{W}(\mathbf{z}; \mathbf{p}) \leftarrow \mathcal{W}(\mathcal{W}(\mathbf{z}; \Delta \mathbf{p}); \mathbf{p}) \quad \text{“Forward Composition”}$$

keep applying steps until $\Delta \mathbf{p}$ converges.

$$\mathbf{J}_{fc} = \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})^T}{\partial \mathbf{p}} \quad \mathbf{H}_{fc} = \mathbf{J}_{fc}^T \mathbf{J}_{fc}$$

Inverse-Composition

- Forward Composite (FC) still has the same problems as with Forward Additive (FA) (recomputing the Jacobian).
- However, if we assume the template is not static we can invert our problem using the approximation,

$$I(\mathcal{W}(\mathbf{z}; \mathbf{p})) \approx T(\mathcal{W}(\mathbf{z}; \mathbf{0})) + \frac{\partial T(\mathcal{W}(\mathbf{z}; \mathbf{0}))}{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})}^T \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})}{\partial \mathbf{p}}^T \Delta \mathbf{p}$$

“Inverse Composition Approximation”

given,

$$T(\mathcal{W}(\mathbf{z}; \mathbf{0})) = I(\mathcal{W}(\mathcal{W}(\mathbf{z}; \Delta \mathbf{p}); \mathbf{p}))$$

- but in fact apply the update to the mis-aligned source image by inverting the composite warp.

Inverse-Composition Algorithm

- Actual algorithm is just the application of the following steps,

Step 1:

$$\Delta \mathbf{p} = \boxed{\mathbf{H}_{ic}^{-1} \mathbf{J}_{ic}^T} [T(\mathbf{z}) - I(\mathcal{W}(\mathbf{z}; \mathbf{p}))]$$

“Static”

Step 2:

$$\mathcal{W}(\mathbf{z}; \mathbf{p}) \leftarrow \mathcal{W}(\mathcal{W}^{-1}(\mathbf{z}; \Delta \mathbf{p}); \mathbf{p}) \quad \text{“Inverse Composition”}$$

keep applying steps until $\Delta \mathbf{p}$ converges.

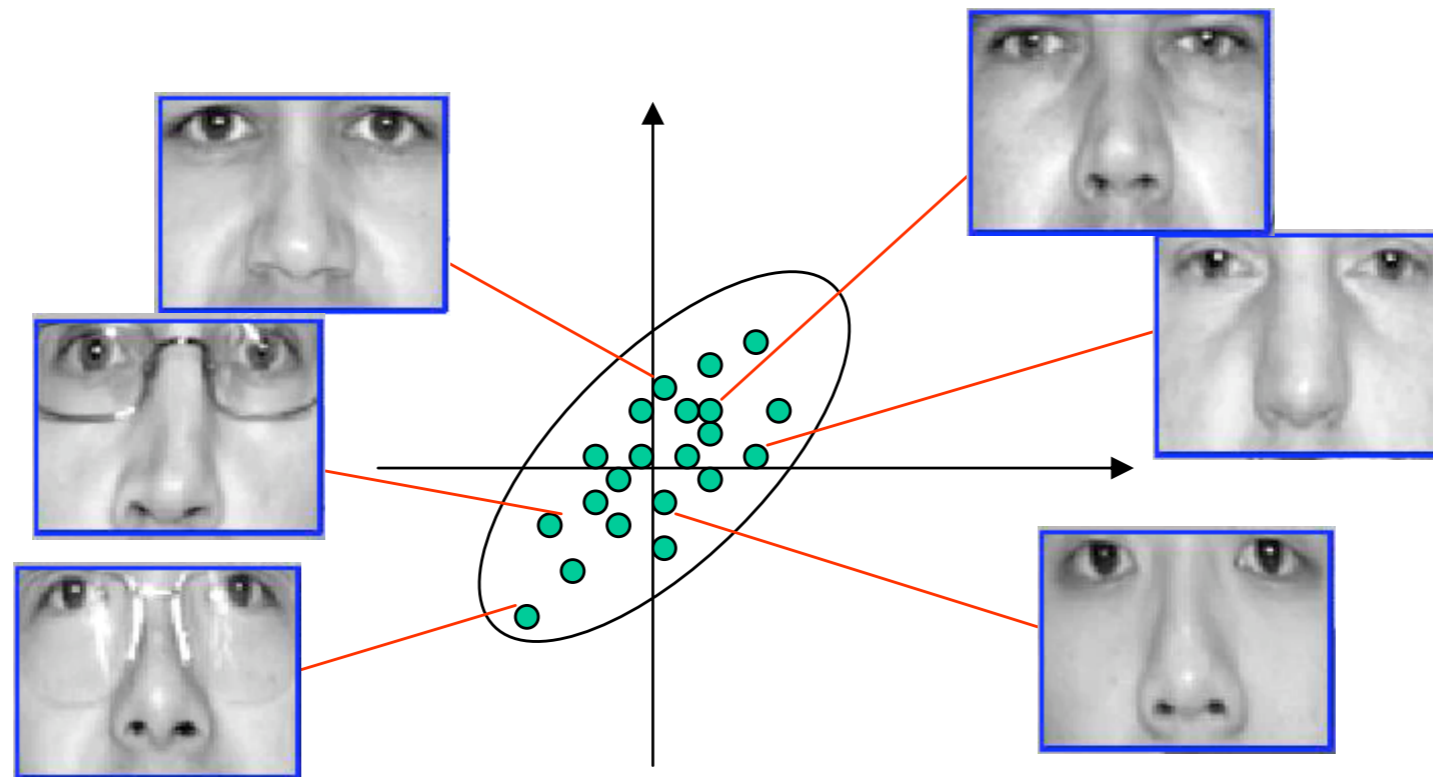
$$\mathbf{J}_{ic} = \frac{\partial T(\mathcal{W}(\mathbf{z}; \mathbf{0}))^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})^T}{\partial \mathbf{p}} \quad \mathbf{H}_{ic} = \mathbf{J}_{ic}^T \mathbf{J}_{ic}$$

Inverse Composition Algorithm

- Baker et al. (2002) demonstrated that there is no empirical difference between the FA and IC algorithms.
- A massive bonus of the IC algorithm is that the Hessian and Jacobian matrices can all be pre-computed.
- In recent evaluations (Baker et al.) on complex warps the IC approach could register and track objects at close to 200 fps.

Appearance Variation

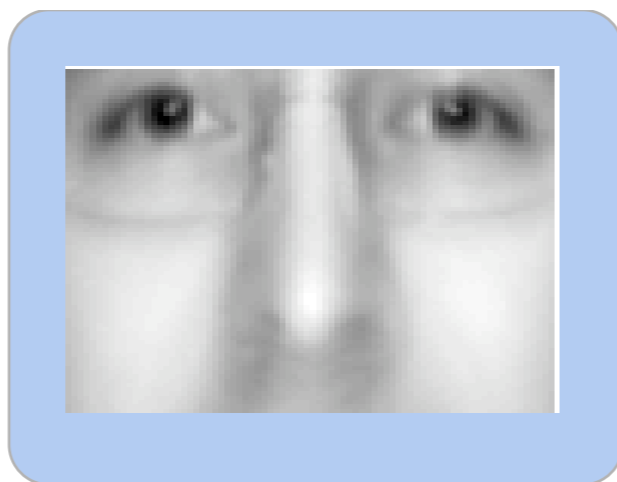
- Like for the ES methods GS approaches also have to deal with appearance variation.



Eigen-Objects

- Black & Jepson (1998) first proposed that the DFFS method of Moghaddam & Pentland employed for ES could be applied within GS.

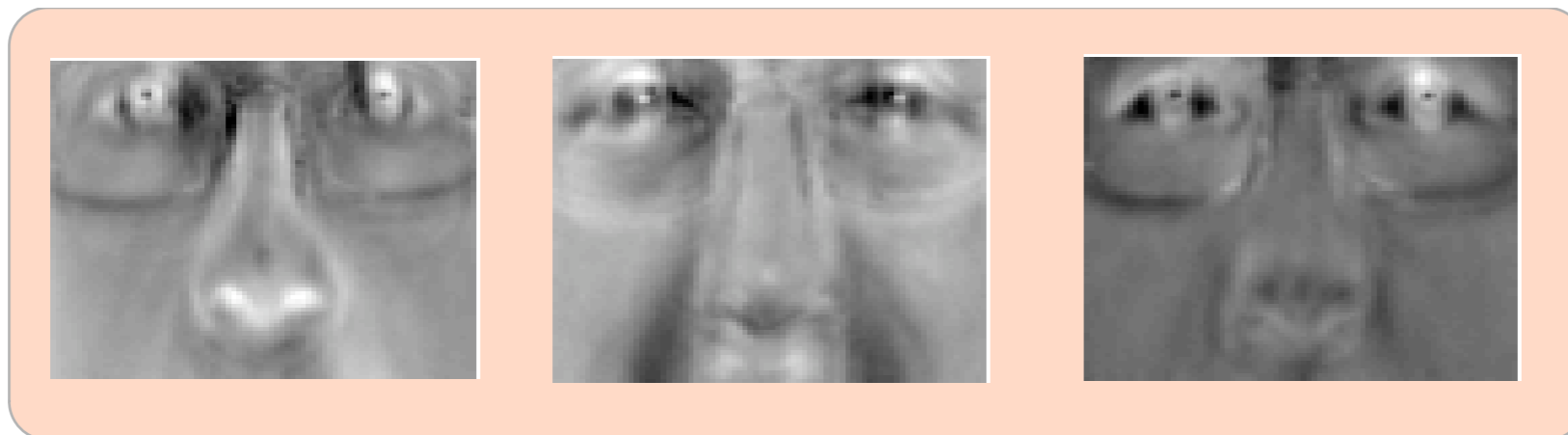
$$T(\mathbf{z}) + \mathbf{A}\Delta\lambda \approx I(\mathcal{W}(\mathbf{z}; \mathbf{p})) + \mathbf{J}\Delta\mathbf{p}$$



Eigen-Objects

- Black & Jepson (1998) first proposed that the DFFS method of Moghaddam & Pentland employed for ES could be applied within GS.

$$T(\mathbf{z}) + \mathbf{A}\Delta\lambda \approx I(\mathcal{W}(\mathbf{z}; \mathbf{p})) + \mathbf{J}\Delta\mathbf{p}$$



Eigen-Objects

- Black & Jepson (1998) first proposed that the DFFS method of Moghaddam & Pentland employed for ES could be applied within GS.

$$T(\mathbf{z}) + \mathbf{A}\Delta\lambda \approx I(\mathcal{W}(\mathbf{z}; \mathbf{p})) + \mathbf{J}\Delta\mathbf{p}$$



Eigen-Objects

- Black & Jepson (1998) first proposed that the DFFS method of Moghaddam & Pentland employed for ES could be applied within GS.

$$T(\mathbf{z}) + \mathbf{A}\Delta\lambda \approx I(\mathcal{W}(\mathbf{z}; \mathbf{p})) + \mathbf{J}\Delta\mathbf{p}$$

$$\mathbf{J} = \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathbf{p}}$$

Simultaneous Algorithm

- Simultaneous algorithm is just the application of the following steps,

Step 1:

$$\begin{bmatrix} \Delta \mathbf{p} \\ \Delta \lambda \end{bmatrix} = \mathbf{H}_{sim}^{-1} \mathbf{J}_{sim}^T [I(\mathcal{W}(\mathbf{z}; \mathbf{p})) - T(\mathbf{z})]$$

Step 2:

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p} \quad T(\mathbf{z}) \leftarrow T(\mathbf{z}) + \mathbf{A} \Delta \lambda$$

keep applying steps until $\Delta \mathbf{p}$ converges.

$$\mathbf{J}_{sim} = [\mathbf{J}, \mathbf{A}] \quad \mathbf{H}_{sim} = \mathbf{J}_{sim}^T \mathbf{J}_{sim}$$

Robust Error Functions

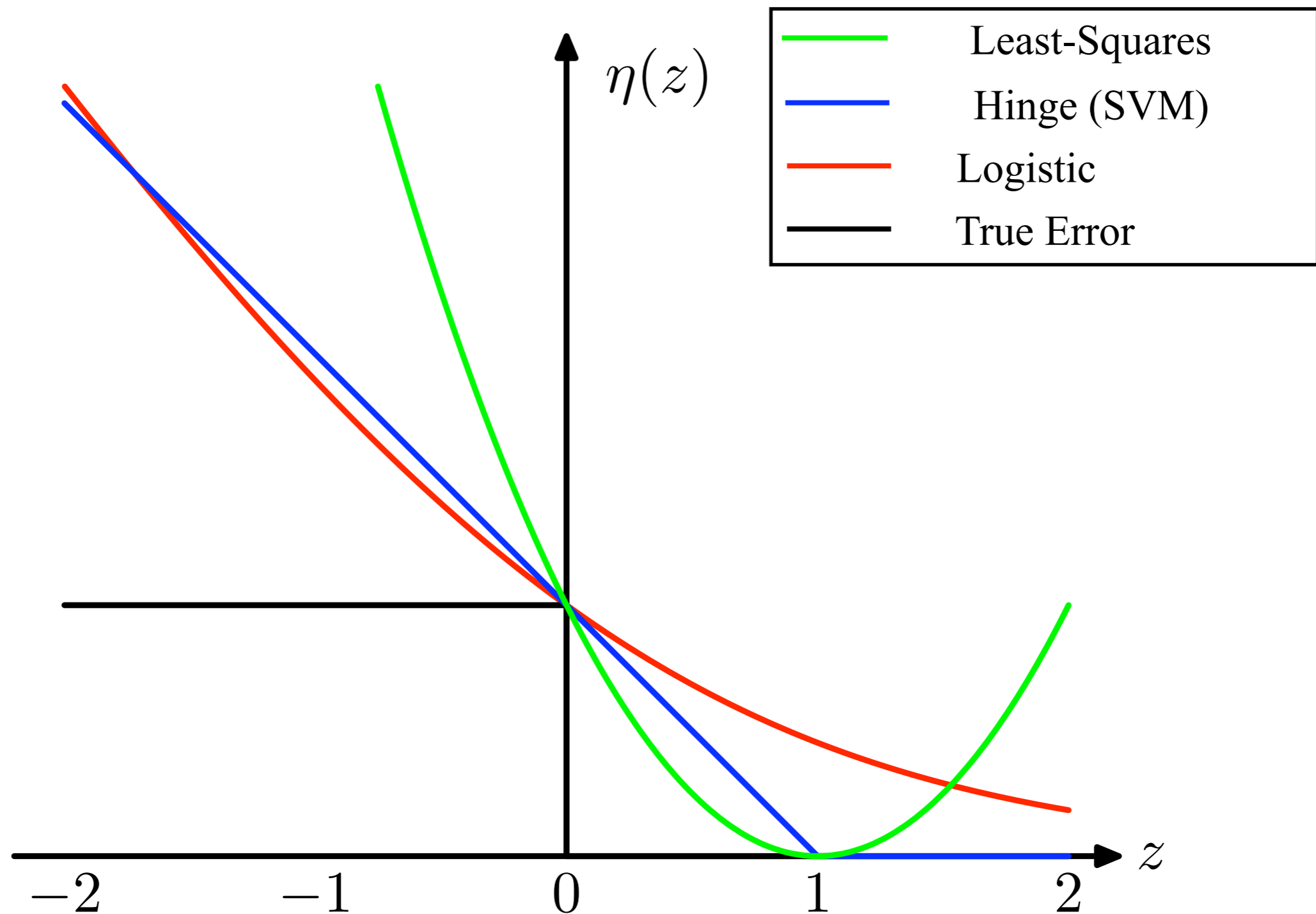
- Well known that least-squares optimization is extremely sensitive to outliers (e.g., occlusion)
- Black & Jepson proposed replacing the least-squares error function,

$$\arg \min_{\Delta \lambda, \Delta \mathbf{p}} ||T(\mathbf{z}) + \mathbf{A}\Delta \lambda - I(\mathcal{W}(\mathbf{z}; \mathbf{p})) - \mathbf{J}\Delta \mathbf{p}||^2$$

with a robust error function $\eta()$,

$$\arg \min_{\Delta \lambda, \Delta \mathbf{p}} \eta[T(\mathbf{z}) + \mathbf{A}\Delta \lambda - I(\mathcal{W}(\mathbf{z}; \mathbf{p})) - \mathbf{J}\Delta \mathbf{p}]$$

Robust Error Functions



Computation Concerns

- Baker et al. reformulated the “simultaneous” algorithm within the IC framework.
- However, no major speedup as the Jacobian matrix needs to still be re-estimated at each iteration due to the appearance change.
- Can be incorporated, however, if we simply “project out” the appearance variation from the IC linear model.

Project-Out Algorithm

- We can simply “project-out” the appearance variation,

Step 1:

$$\Delta \mathbf{p} = \mathbf{H}_{po}^{-1} \mathbf{J}_{po}^T [T(\mathbf{z}) - I(\mathcal{W}(\mathbf{z}; \mathbf{p}))]$$

“Static”

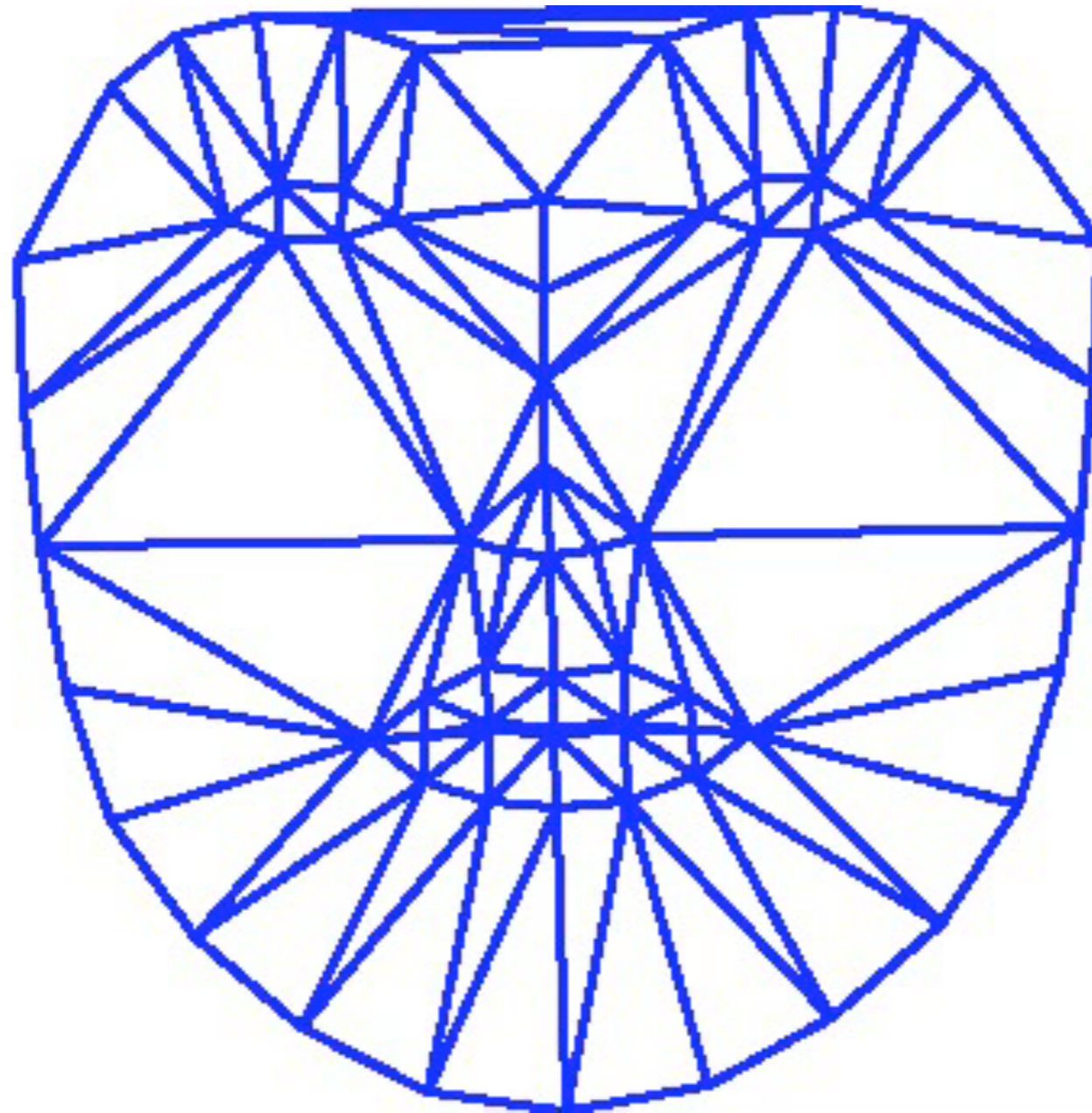
Step 2:

$$\mathcal{W}(\mathbf{z}; \mathbf{p}) \leftarrow \mathcal{W}(\mathcal{W}^{-1}(\mathbf{z}; \Delta \mathbf{p}); \mathbf{p}) \quad \text{“Inverse Composition”}$$

keep applying steps until $\Delta \mathbf{p}$ converges.

$$\mathbf{J}_{po} = \mathbf{J}_{ic} - \mathbf{A} \mathbf{A}^T \mathbf{J}_{ic} \qquad \mathbf{H}_{po} = \mathbf{J}_{po}^T \mathbf{J}_{po}$$

Next: Non-Rigid Warps



Object Registration and Tracking from a Learning Perspective (Part 4)

Simon Lucey
The Robotics Institute, Carnegie Mellon University

Overall Course Outline

- Lecture 3
 - Gradient Search (GS)
 - Generative models for GS
 - Lucas-Kanade, Black-Jepson algorithms
 - Speeding up GS through Inverse Composition
- Lecture 4
 - Non-Rigid Warps - Active Appearance Models
 - Discriminative models for GS
 - Support Vector Tracking (SVT)
 - Regression Search (RS)
 - Williams & Blake

Rigid Warps

- Up until now, we have been limited to dealing with objects that lie conveniently within a rectangle.
- These objects can only deform rigidly,



translation



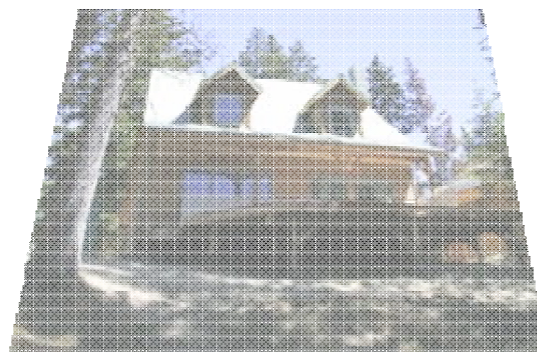
rotation



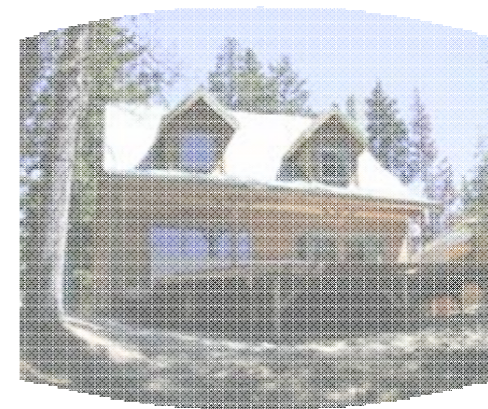
aspect



affine



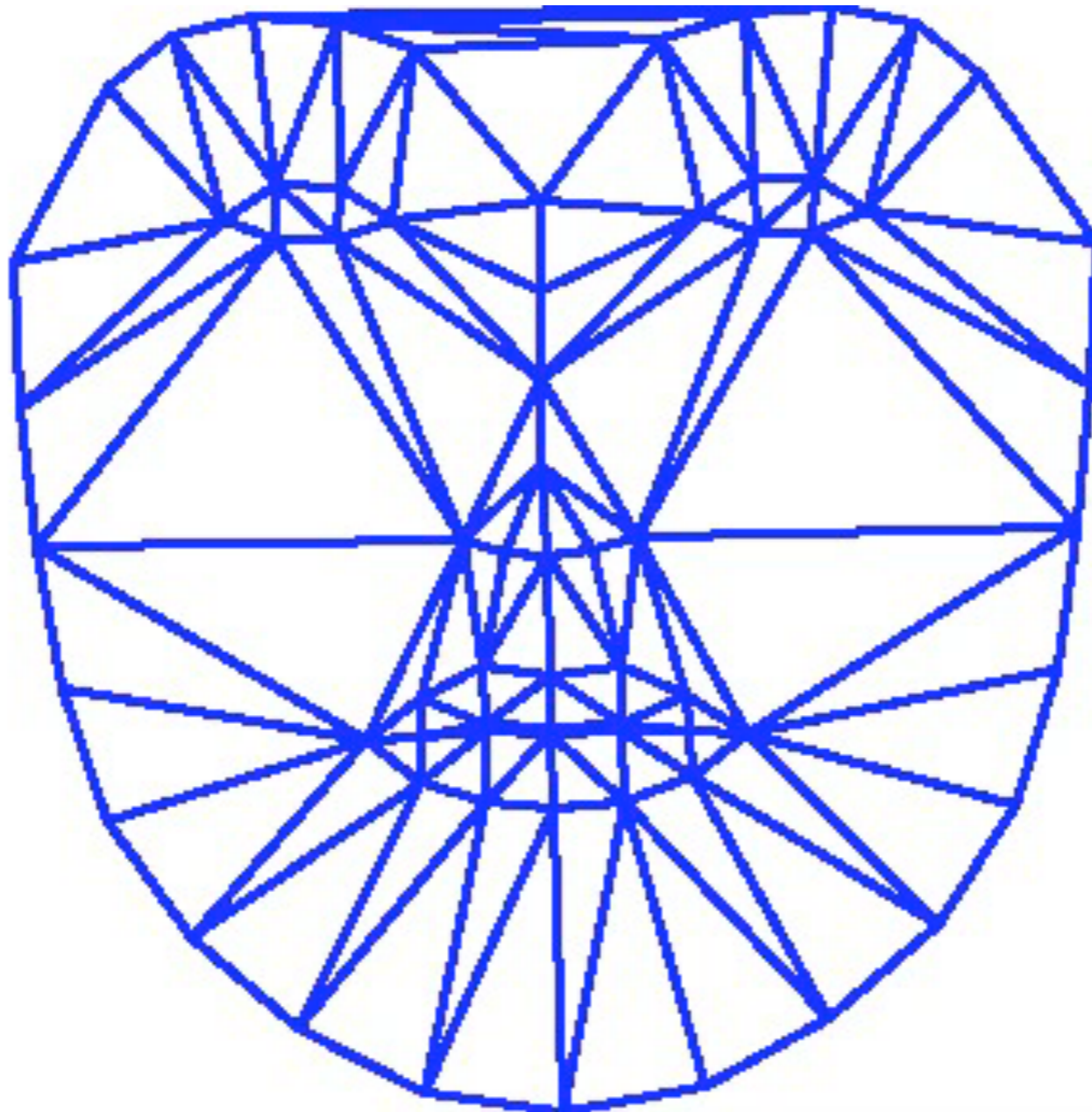
perspective



cylindrical

Non-Rigid Warps

- What happens if I want to deal with non-rigid warps,



(Szeliski and Fleet)

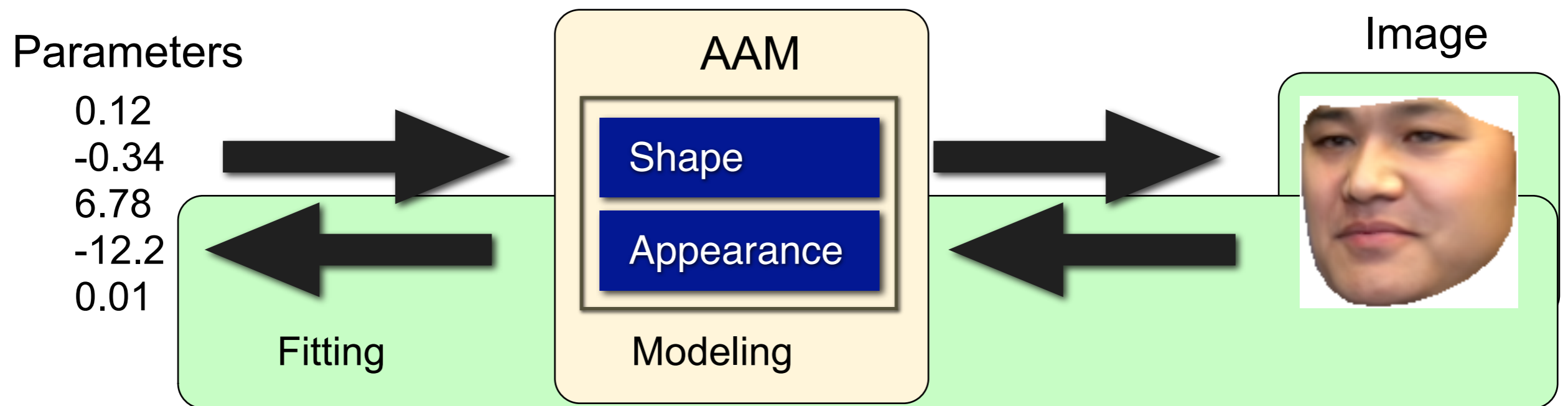
Active Appearance Models

- Introduced by Cootes and Taylor in 1998
- Example of class of deformable models which also includes Active Blobs (Sclaroff and Isidoror), Active Shape Models (Cootes and Taylor) and Morphable Models (Blanz and Vetter)
- Used successfully on faces, hands, and in medical imaging

(Matthews and Gross)

Active Appearance Models

- Separately model object shape and appearance



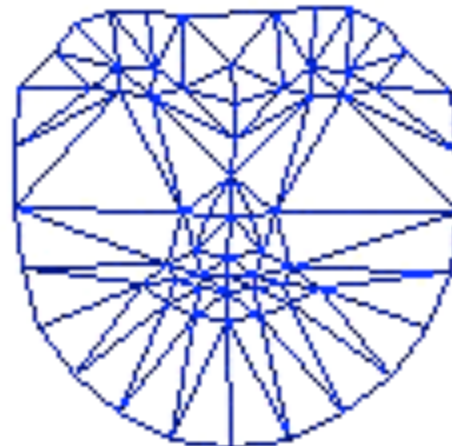
(Matthews and Gross)

Active Appearance Models

Appearance



Shape



Combined



(Matthews and Gross)

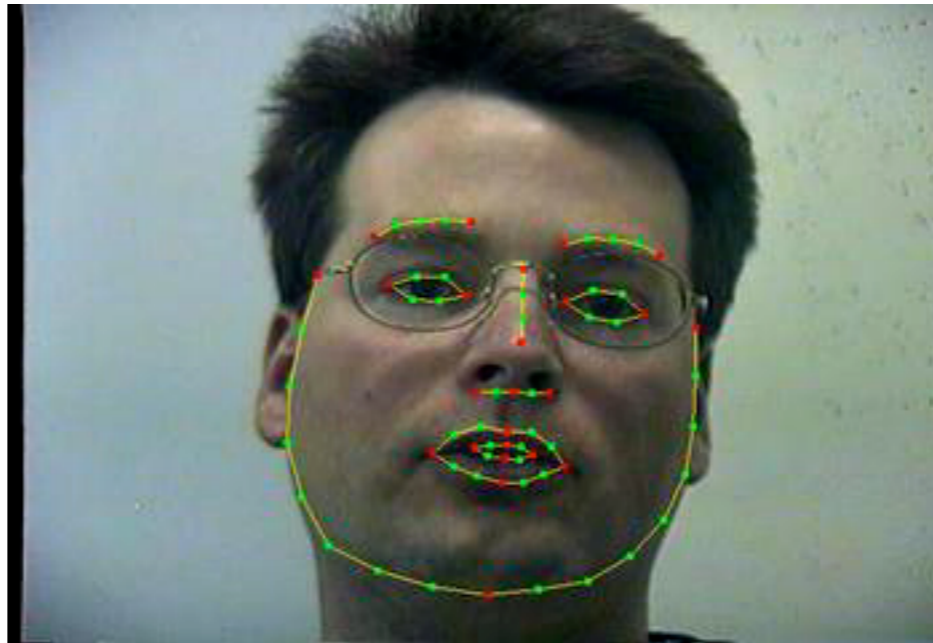
Lecture Overview

- How do you build an AAM?
- How do you fit an AAM to an image?
 - What is the general approach?
 - How can we do it **efficiently**?
- Extensions

(Matthews and Gross)

Model Building - Definition

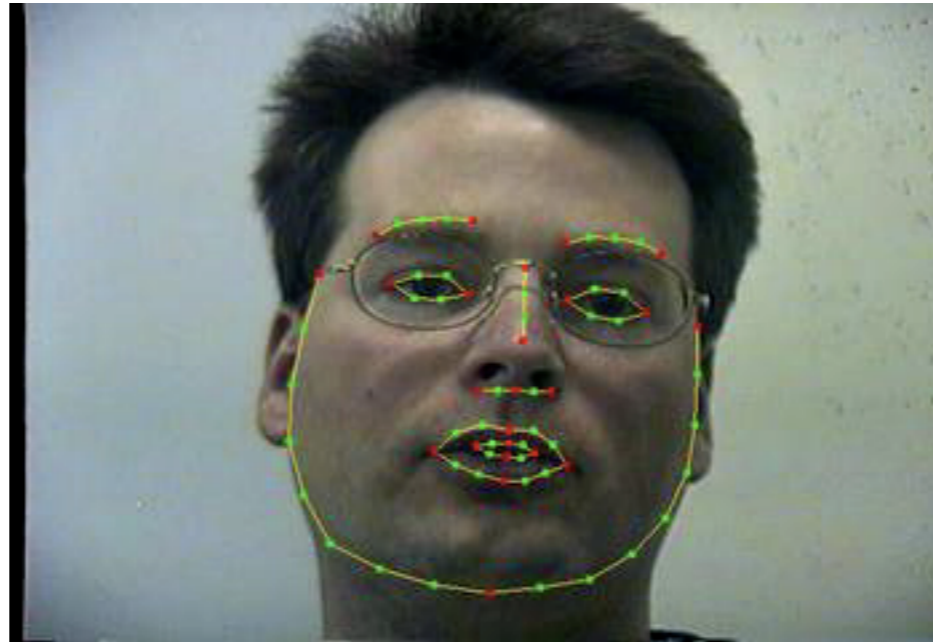
- Define a set of landmarks on a face that can be reliably located in an image



- Still open questions:
 - How many landmarks, which landmarks

(Matthews and Gross)

Model Building - Labeling



$$(x_1, y_1, x_2, y_2, \dots, x_n, y_n)$$

Manually label lots and lots of data

(Matthews and Gross)

Model Building - Shape

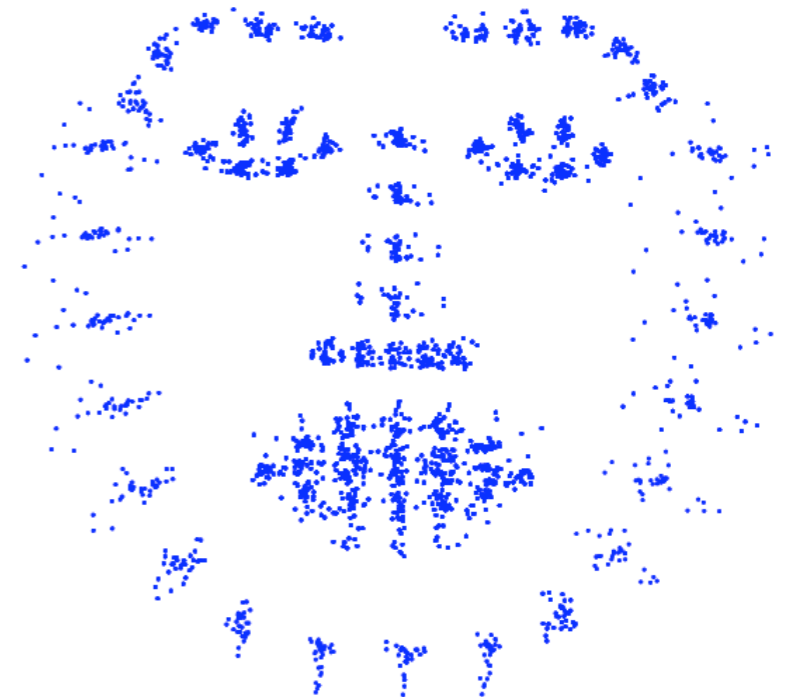
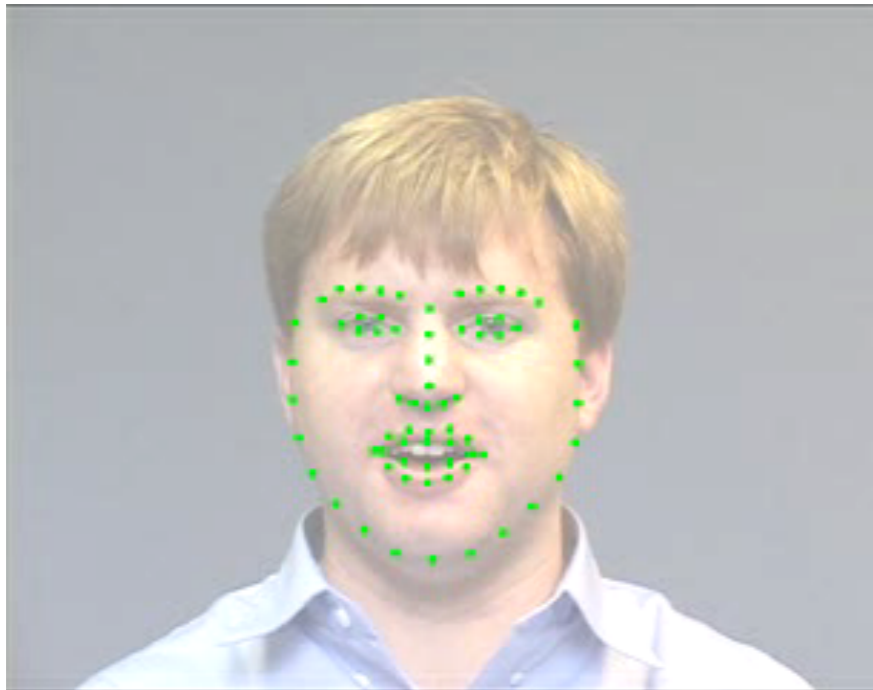
- Various sources of shape variations between individual images
 - Identity
 - Facial expression
 - Position in the image

Don't Care!

(Matthews and Gross)

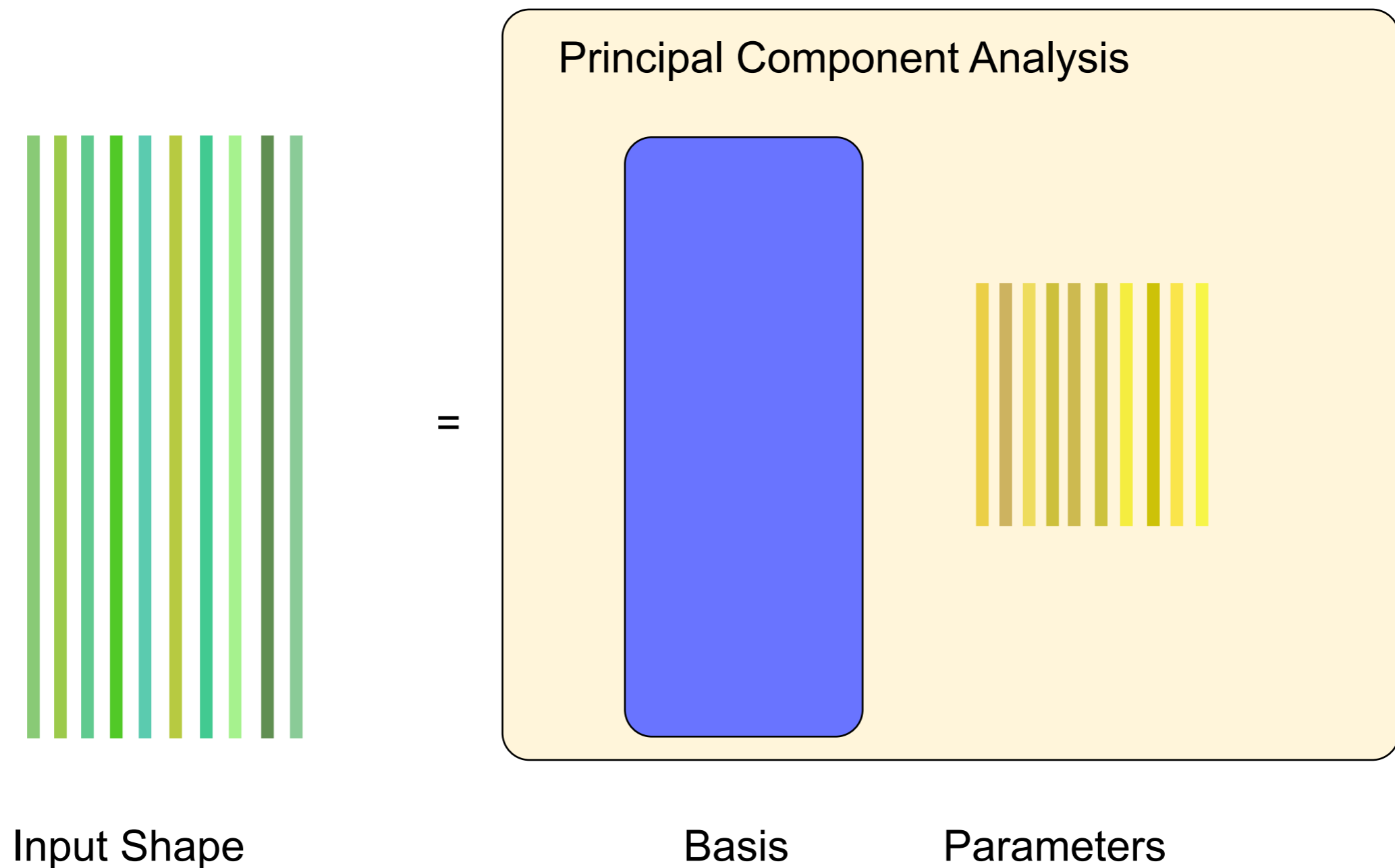
Shape Alignment

- Procrustes analysis
 - Removes variation due to predetermined shape transformation



(Matthews and Gross)

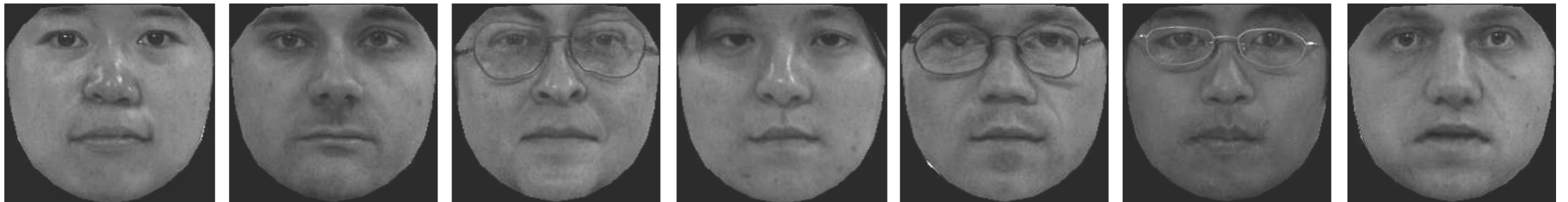
Building a Shape Model



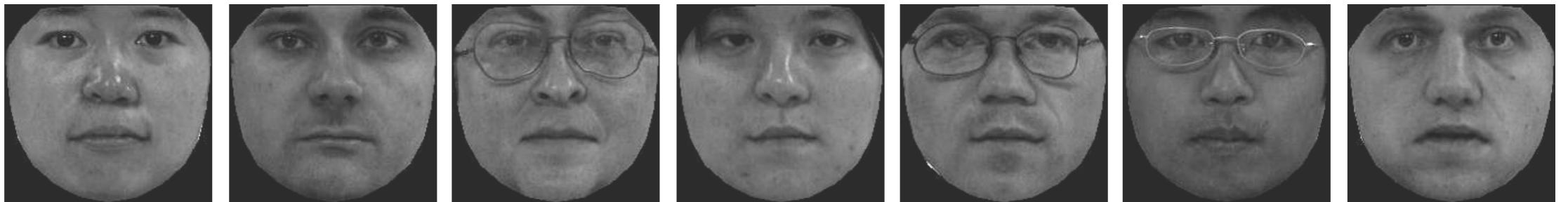
(Matthews and Gross)

PCA on Appearance

Original



Full



20 dim.



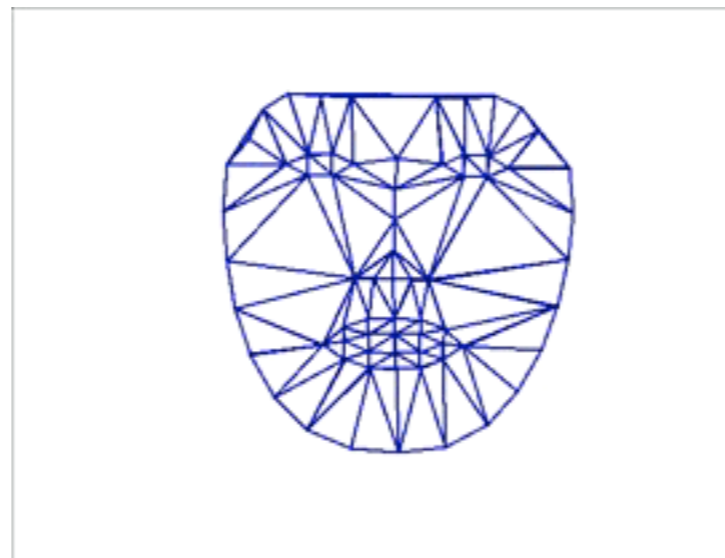
10 dim.



(Matthews and Gross)

PCA Shape Model

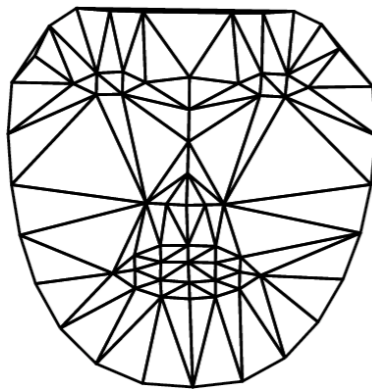
- Input:
Bunch of aligned face shapes
- Output
Mean shape and the direction of the largest variations (eigenvectors)



(Matthews and Gross)

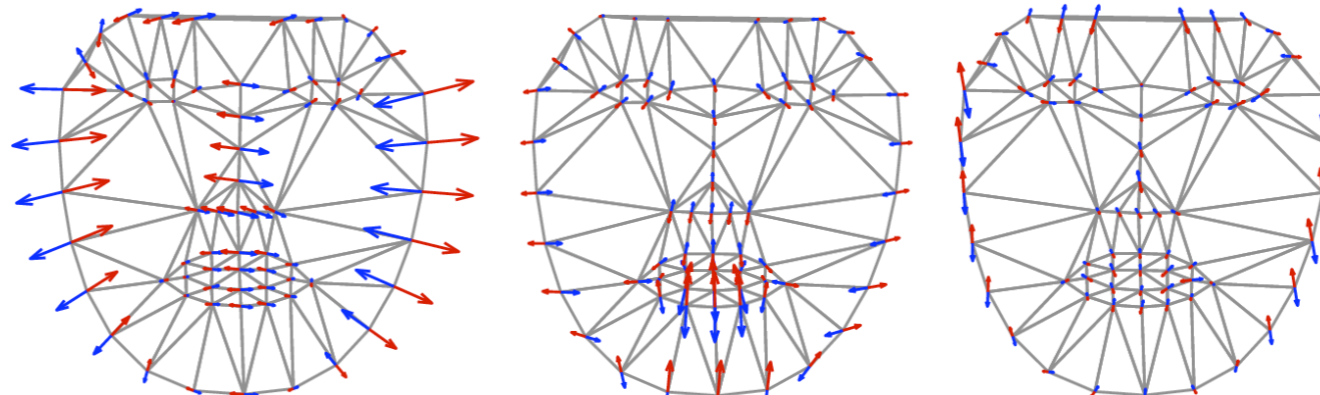
PCA Shape Model

Mean Face



s_0

First 3 Shape Modes



s_1

s_2

s_3

Shape Model

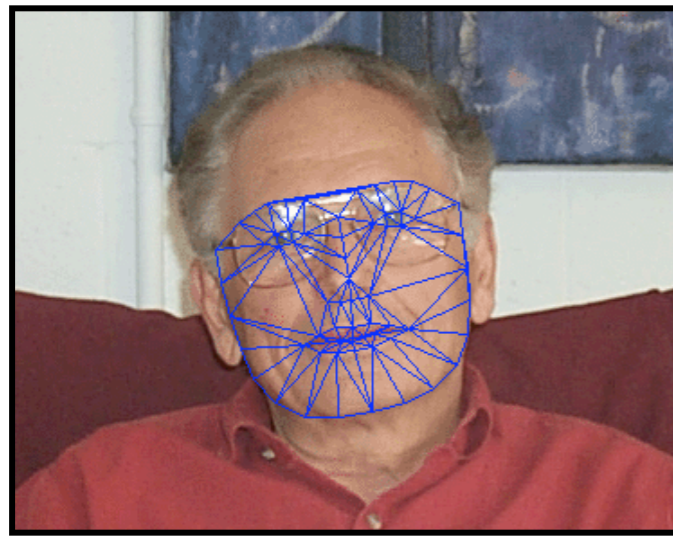
$$\mathcal{W}(\mathbf{z}; \mathbf{p}) = \mathbf{s}_0 + \sum_{k=1}^K p_k \mathbf{s}_k$$

Shape Parameters

$$\mathbf{p} = [p_1, p_2, \dots, p_K]^T$$

(Matthews and Gross)

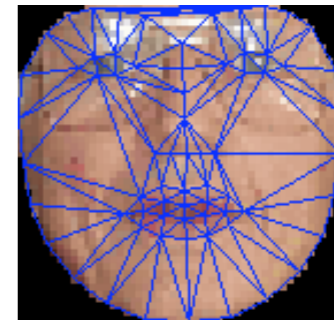
Construction of Appearance Model



Warp to mean shape



S_0



PCA on Appearance

(Matthews and Gross)

PCA Appearance Model

- Input:
Bunch of shape normalized textures
- Output:
Mean appearance and appearance eigenvectors



(Matthews and Gross)

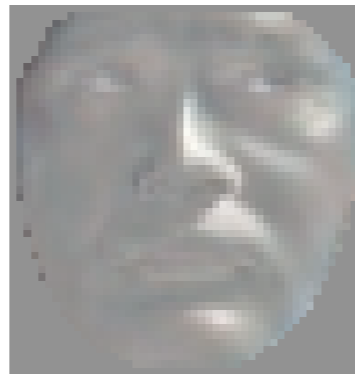
AAM Appearance Model

Mean Face

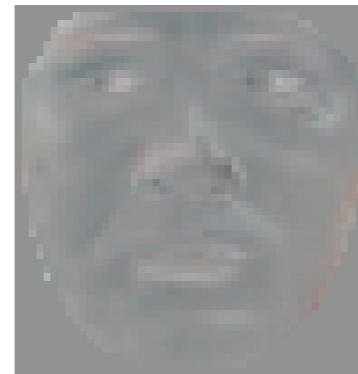


$A_0(\mathbf{z})$

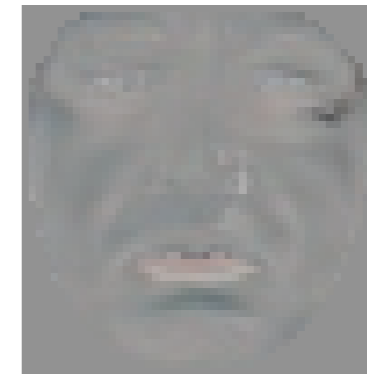
First 3 Appearance Modes



$A_1(\mathbf{z})$



$A_2(\mathbf{z})$



$A_3(\mathbf{z})$

Appearance Model

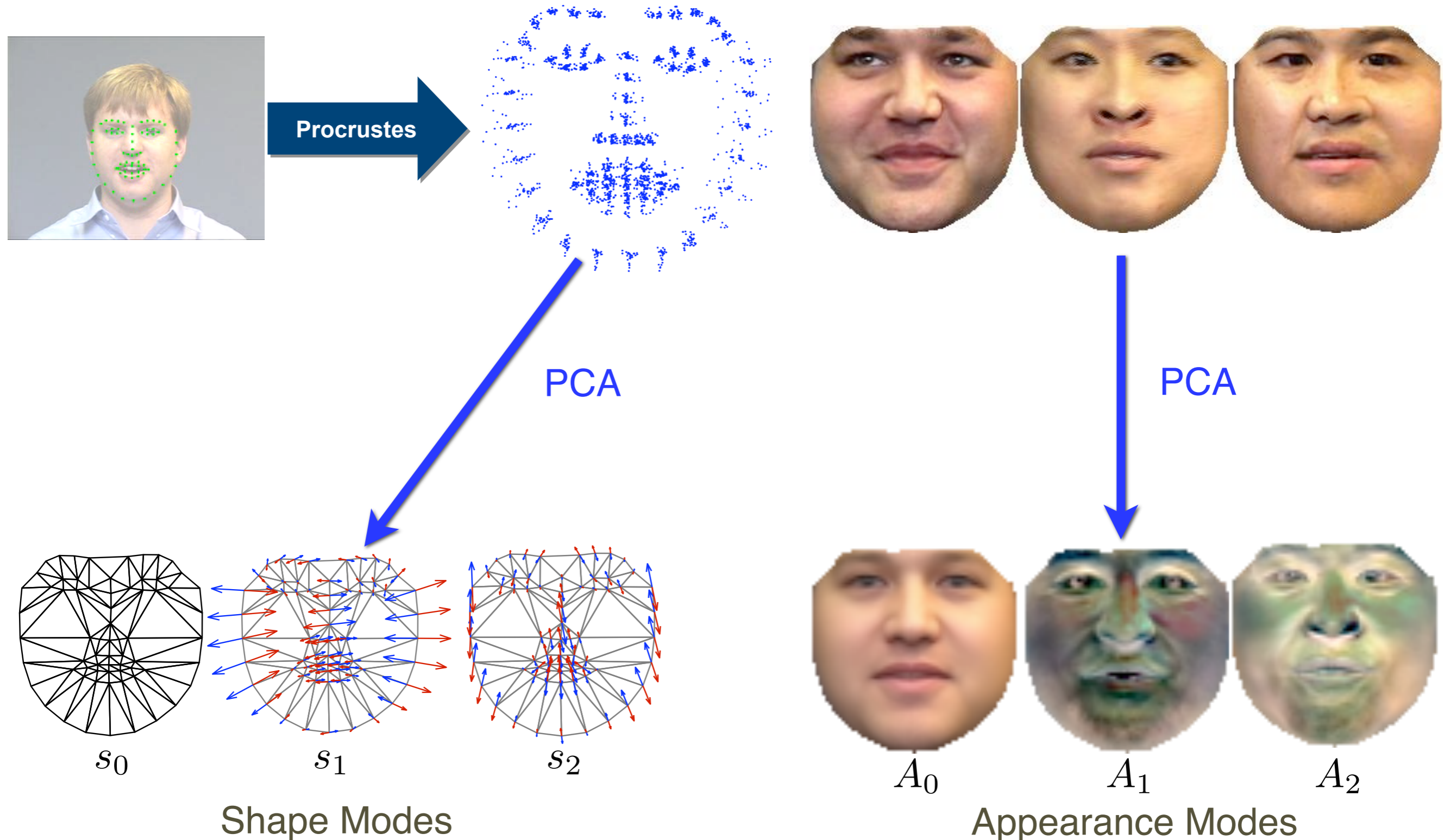
$$T(\mathbf{z}) = A_0(\mathbf{z}) + \sum_{m=1}^M \lambda_m A_m(\mathbf{z})$$

Appearance Parameters


$$(\lambda_1, \lambda_2, \dots, \lambda_n)^T$$

(Matthews and Gross)

AAM Model Building Overview



Model Instantiation



Appearance, $T(\mathbf{z}) = A_0(\mathbf{z}) + 3559A_1(\mathbf{z}) + 351A_2(\mathbf{z}) - 256A_3(\mathbf{z}) \dots$

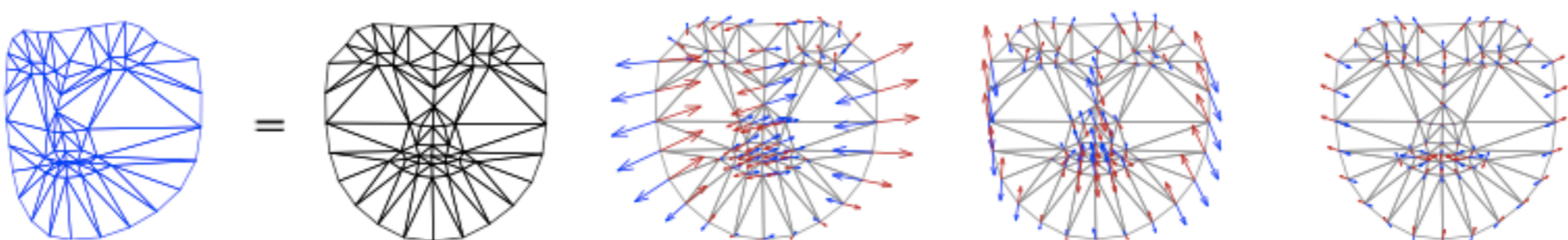


$\mathcal{W}(\mathbf{z}; \mathbf{p})$



$T(\mathcal{W}(\mathbf{z}; \mathbf{p}))$

Shape, $\mathbf{z} = \mathbf{s}_0 - 54\mathbf{s}_1 + 10\mathbf{s}_2 - 9.1\mathbf{s}_3 \dots$

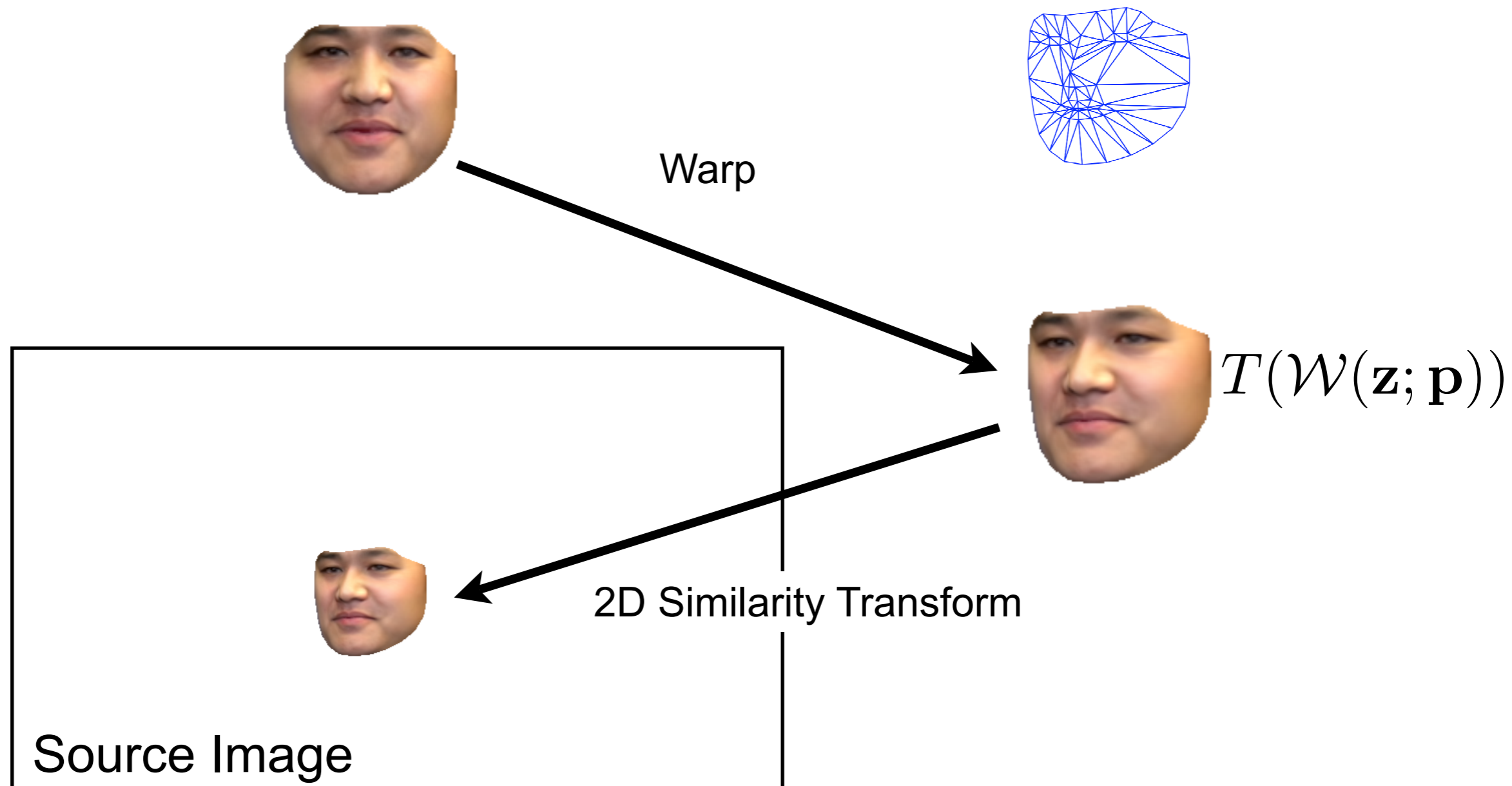


(Matthews and Gross)

Model Instantiation

$$T(\mathbf{z}) = A_0(\mathbf{z}) + \sum_{m=1}^M \lambda_i A_m(\mathbf{z})$$

$$\mathcal{W}(\mathbf{z}; \mathbf{p}) = \mathbf{s}_0 + \sum_{k=1}^K p_k \mathbf{s}_k$$



(Matthews and Gross)

A Parameterized Face



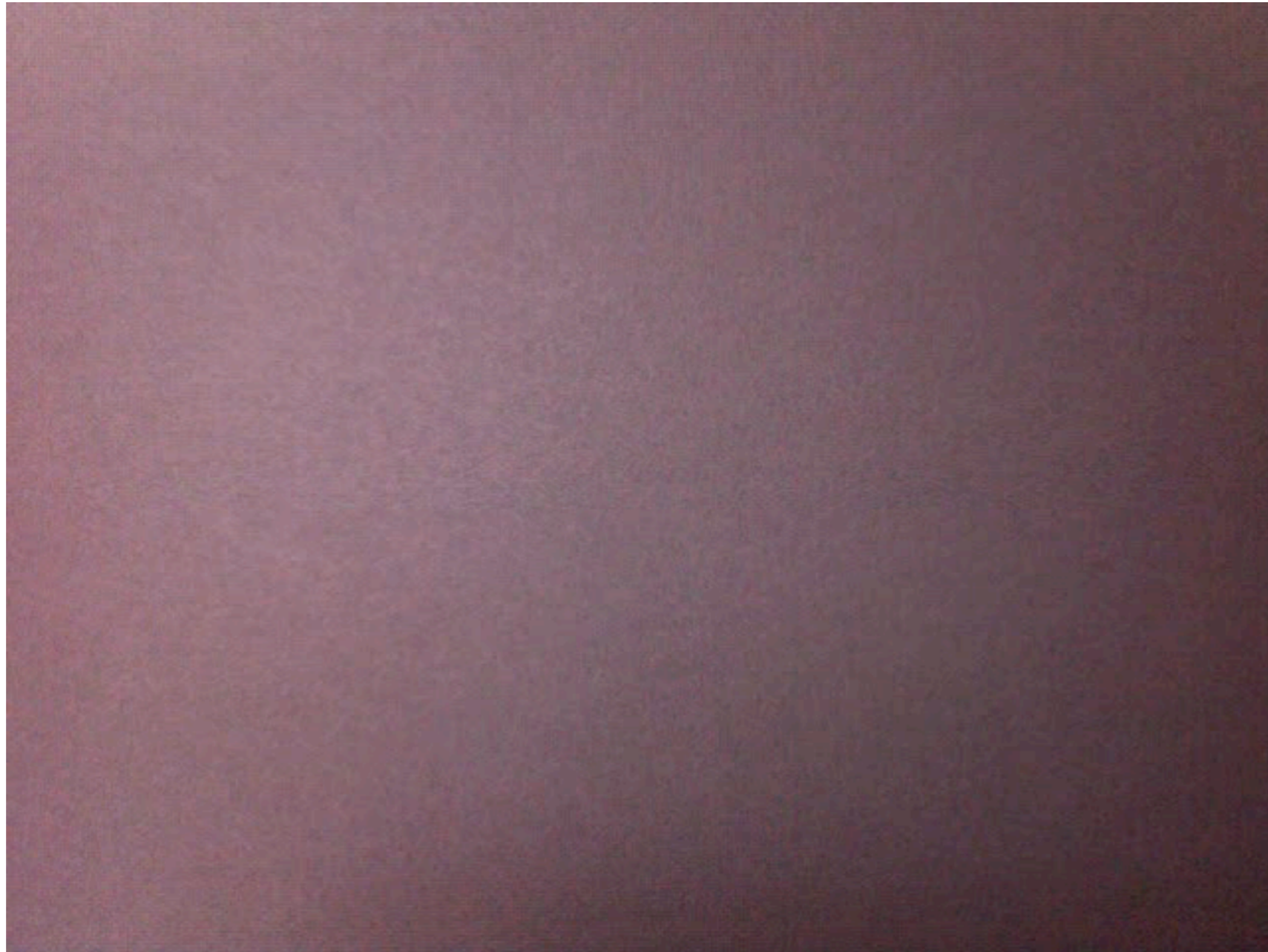
(Matthews and Gross)

Things to Note

- AAMs are **linear** models
 - Efficiently capture non-rigid face deformations
- AAMs are entirely **data-driven**
 - No “face constraints”
- We discussed independent AAMs. “Combined” AAMs add an additional PCA on the combined shape and appearance parameters.

(Matthews and Gross)

Hand AAM



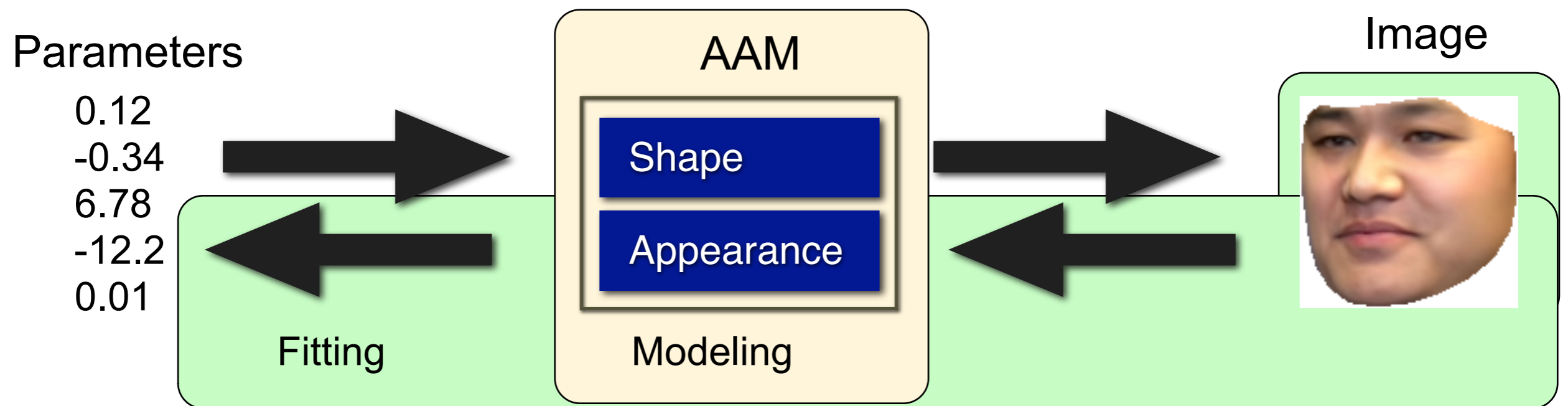
(Matthews and Gross)

Things to Note

- AAMs are **linear** models
 - Efficiently capture non-rigid face deformations
- AAMs are entirely **data-driven**
 - No “face constraints”
- We discussed independent AAMs. “Combined” AAMs add an additional PCA on the combined shape and appearance parameters.

(Matthews and Gross)

Active Appearance Models



Determine the best model parameters to reconstruct the image

(Matthews and Gross)

AAM - Fitting “Simultaneous”

- Now that we have defined an AAM, how should do the fitting?
- Turns out, we can still use the “simultaneous” extension of the LK algorithm,

$$\arg \min_{\Delta \lambda, \Delta \mathbf{p}} ||T(\mathbf{z}) + \mathbf{A} \Delta \lambda - I(\mathcal{W}(\mathbf{z}; \mathbf{p})) - \mathbf{J} \Delta \mathbf{p}||^2$$

- where,

$$\frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathbf{p}} = [\mathbf{s}_1, \dots, \mathbf{s}_K] \quad \mathbf{J} = \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})^T}{\partial \mathbf{p}}$$

AAM - Fitting “Inverse Composition”

- Even faster if we use the inverse composition (IC) extension as well as “projecting out” the appearance,

$$\arg \min_{\Delta \mathbf{p}} ||T(\mathbf{z}) + \mathbf{J}_{ic} \Delta \mathbf{p} - I(\mathcal{W}(\mathbf{z}; \mathbf{p}))||_{\text{null}(\mathbf{A})}^2$$

- where,

$$\frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})^T}{\partial \mathbf{p}} = [\mathbf{s}_1, \dots, \mathbf{s}_K] \quad \mathbf{J}_{ic} = \frac{\partial T(\mathcal{W}(\mathbf{z}; \mathbf{0}))^T}{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})} \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{0})^T}{\partial \mathbf{p}}$$

IC AAM Fitting

- Runs at 230 Hz on a 3.2GHz PC

Input



Shape
Overlaid



Overlaid
Model
Instance

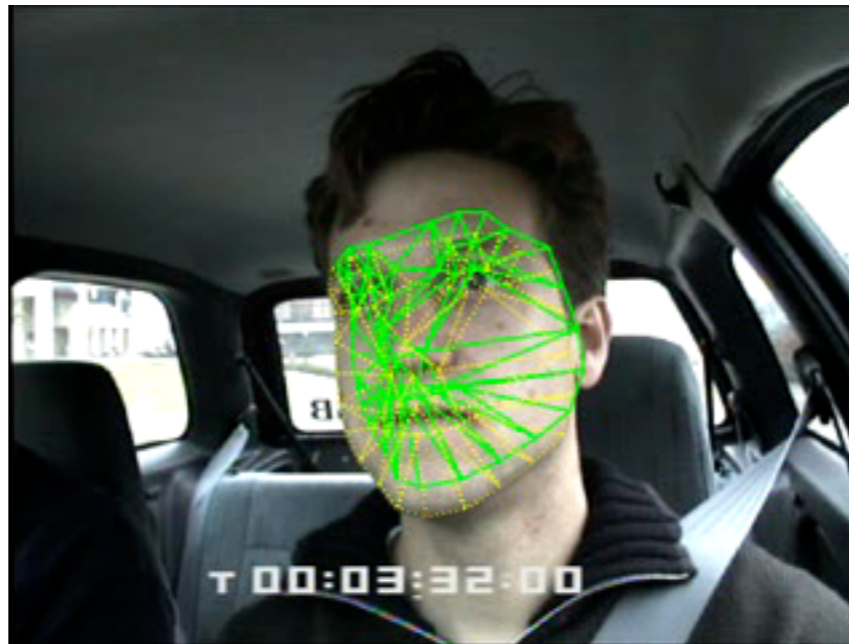


Rendered
Model
Instance



(Matthews and Gross)

Why We Need High Speed?



Re-initialize model multiple times if tracking fails and still track in real time

(Matthews and Gross)

Not All Is Peachy



Original model does not handle occlusion well

(Matthews and Gross)

AAM - Fitting “Robust Error Function”

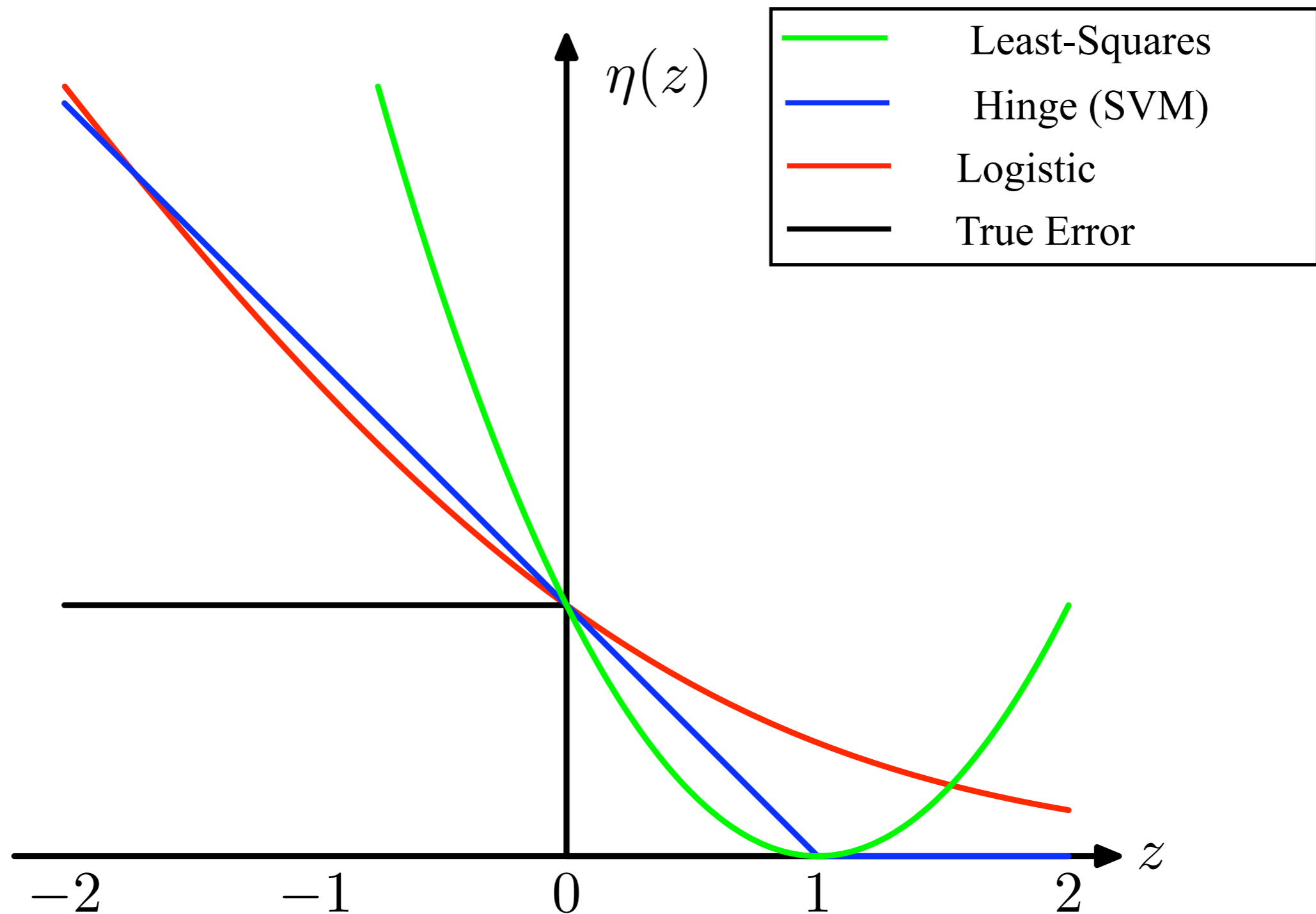
- To handle occlusions we can employ the robust error function,

$$\arg \min_{\Delta \lambda, \Delta \mathbf{p}} \eta(T(\mathbf{z}) + \mathbf{A} \Delta \lambda - I(\mathcal{W}(\mathbf{z}; \mathbf{p})) - \mathbf{J} \Delta \mathbf{p})$$

- where,

$$\frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}{\partial \mathbf{p}}^T = [\mathbf{s}_1, \dots, \mathbf{s}_K] \quad \mathbf{J} = \frac{\partial I(\mathcal{W}(\mathbf{z}; \mathbf{p}))}{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}^T \frac{\partial \mathcal{W}(\mathbf{z}; \mathbf{p})}{\partial \mathbf{p}}^T$$

Robust Error Functions



AAMs with Occlusion Modeling



(Matthews and Gross)

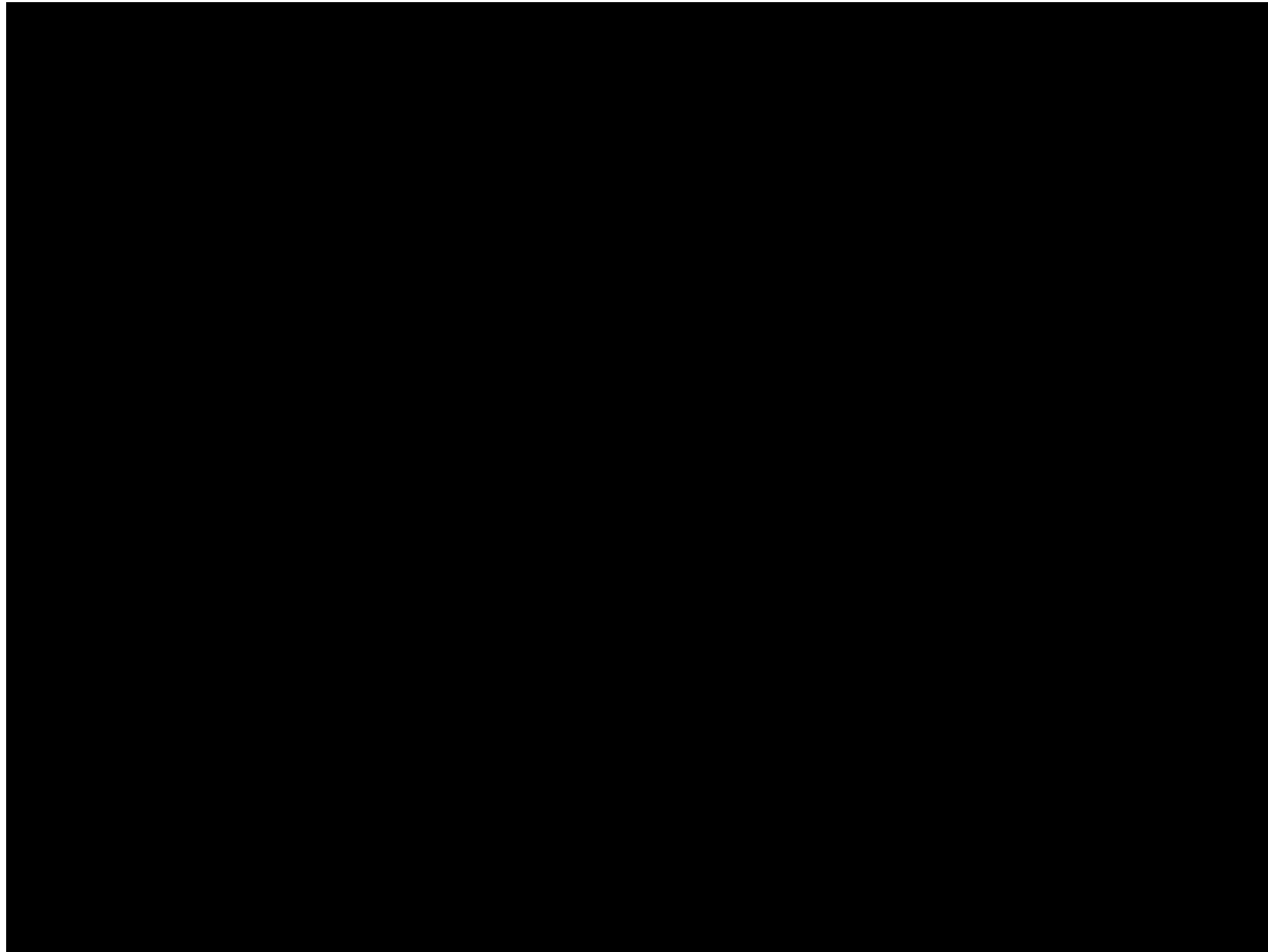
Applications

- **User Interfaces:**
 - Mouse replacement
 - Automotive: Windshield Displays, Smart Airbags
- **Face Recognition:**
 - Pose Normalization
- **Lipreading/Audio-Visual Speech Recognition**
- **Rendering and Animation:**
 - Low-Bandwidth Video Conferencing
 - Audio-Visual Speech Synthesis

(Matthews and Gross)

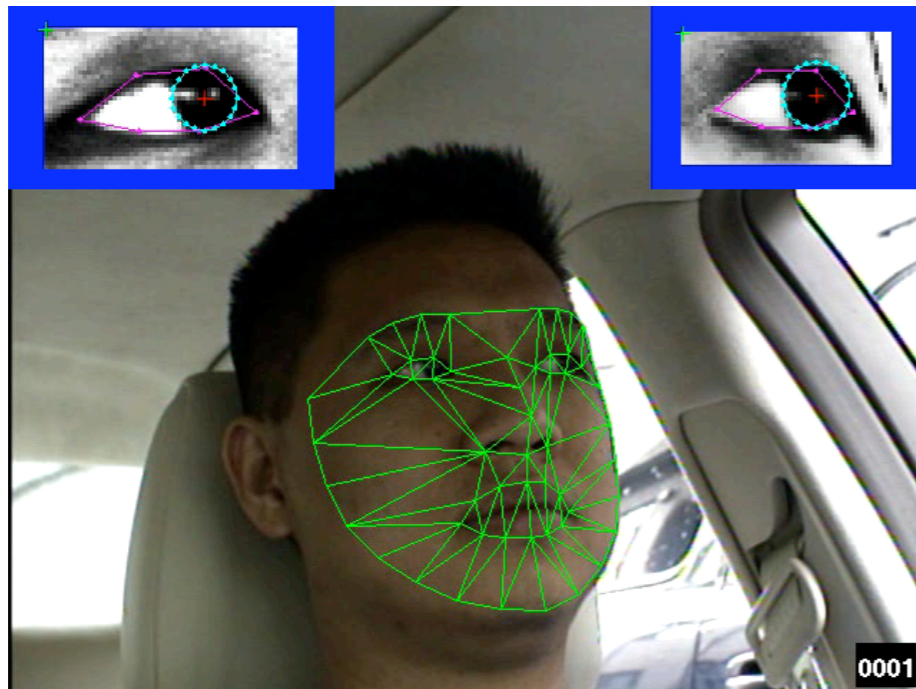
User Interfaces: Head Pose

- Mouse replacement

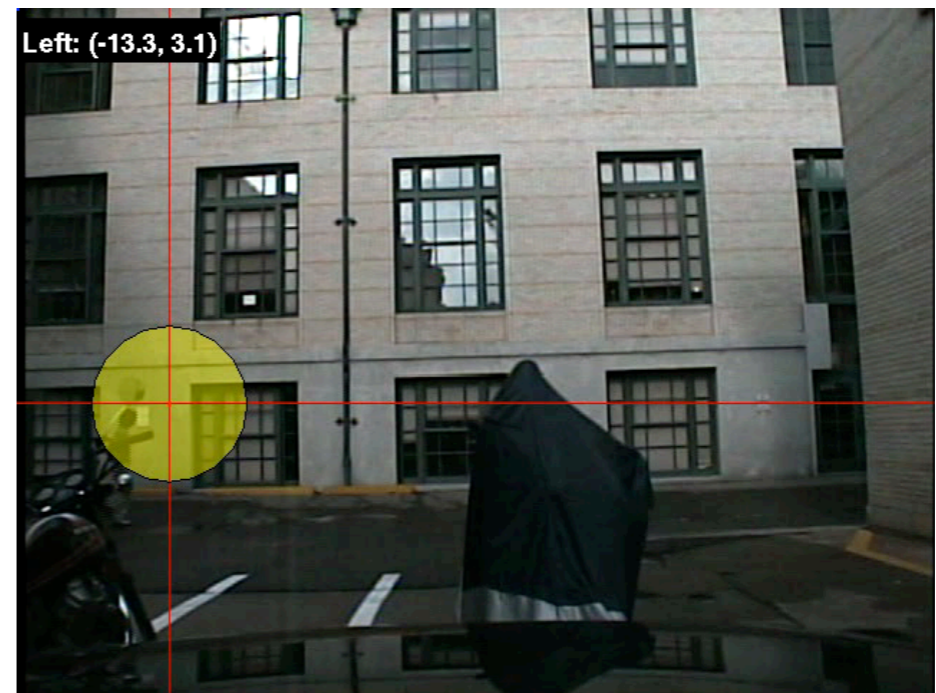


(Matthews and Gross)

User Interfaces: Gaze Tracking



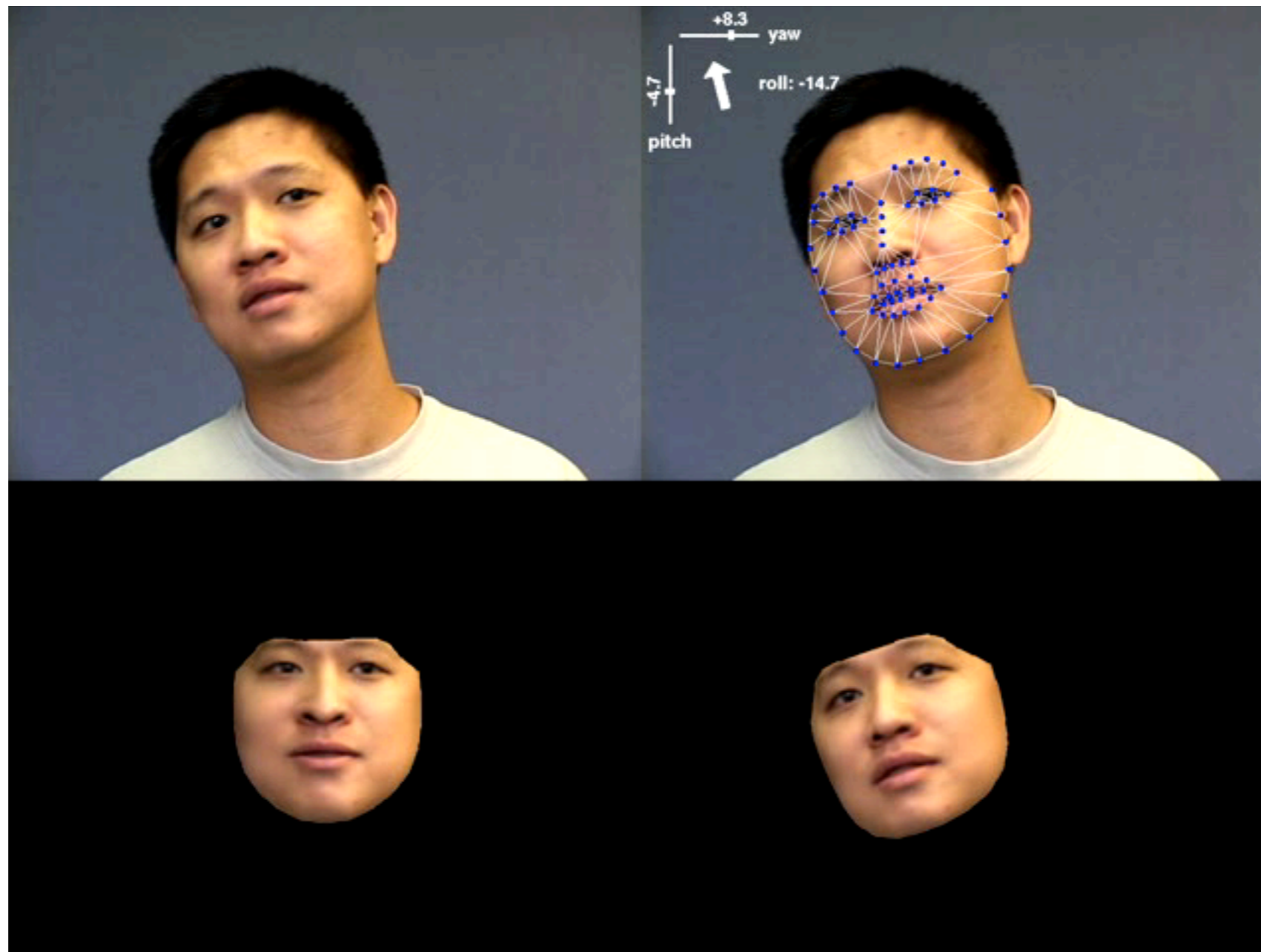
Driver Camera



Exterior View Camera

(Matthews and Gross)

Face Recognition: Pose Normalization



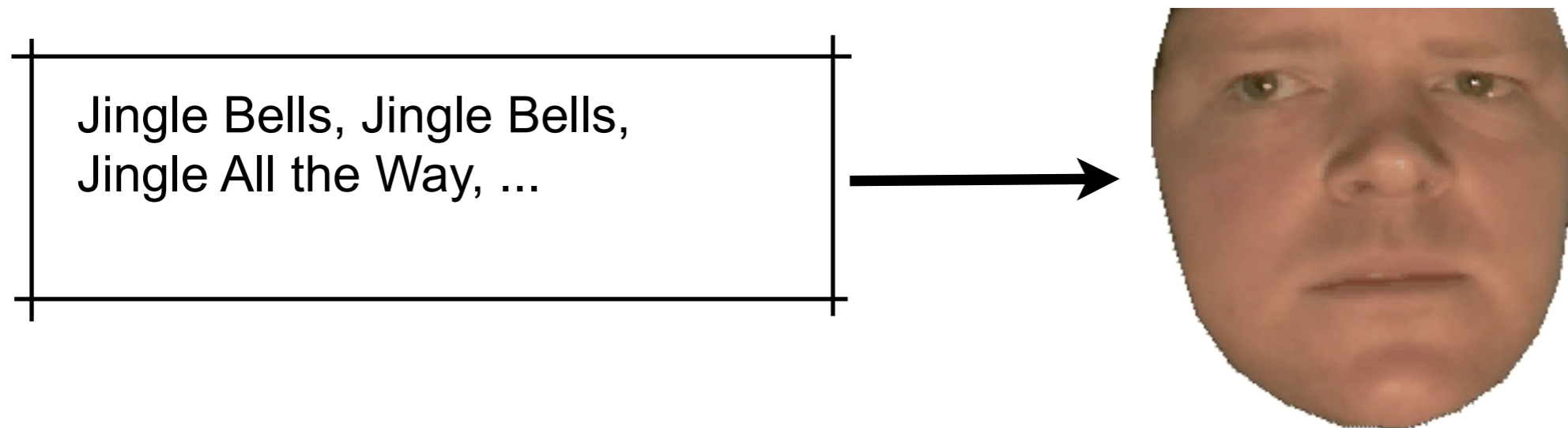
(Matthews and Gross)

Animation Generation



(Matthews and Gross)

Audio-Visual Speech Synthesis



(Matthews and Gross)

Still Open Questions

- Automatic model building without weeks of manual labeling
- Optimal model structure
- Fitting models to unseen faces
- Increase fitting robustness

(Matthews and Gross)