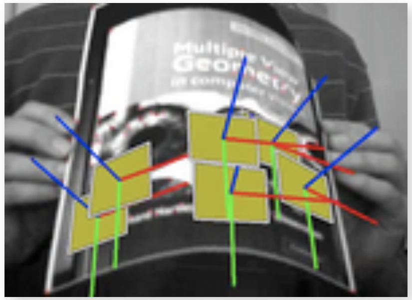
Computer Vision for Augmented Reality

Vincent Lepetit

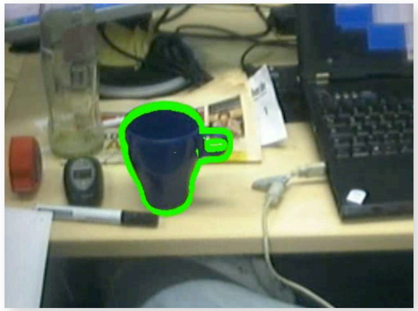
CVLab – Ecole Polytechnique Fédérale de Lausanne

Switzerland

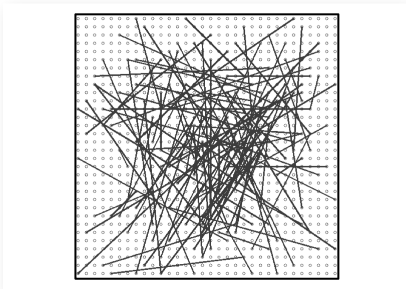
Ferns [CVPR'07]
classification for fast keypoint recognition



Gepard [CVPR'09]
real-time learning of patch rectification



DOT [CVPR'10]
dense descriptor for objet detection



BRIEF [ECCV'10]
very fast feature point descriptor

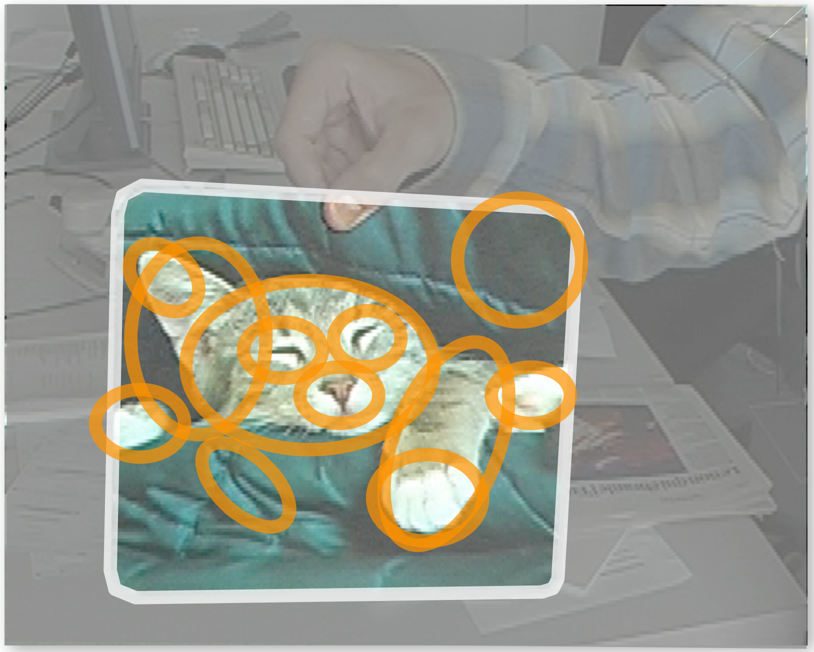
Ferns [CVPR'07]
classification for fast keypoint recognition

3D Object Detection

Registered image(s) of
the object to detect

3D Object Detection

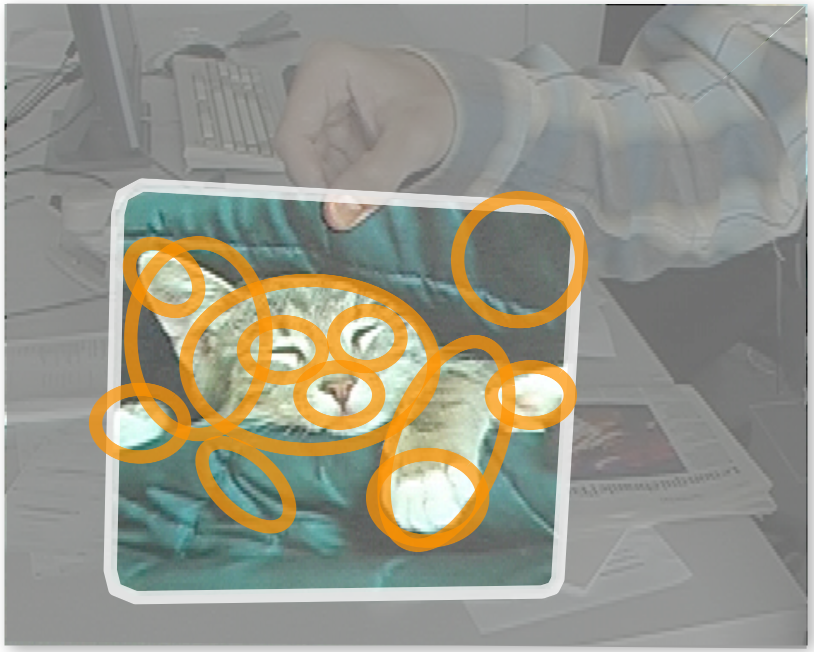
Keypoint detection (Harris, extrema of Laplacian, affine regions,...);



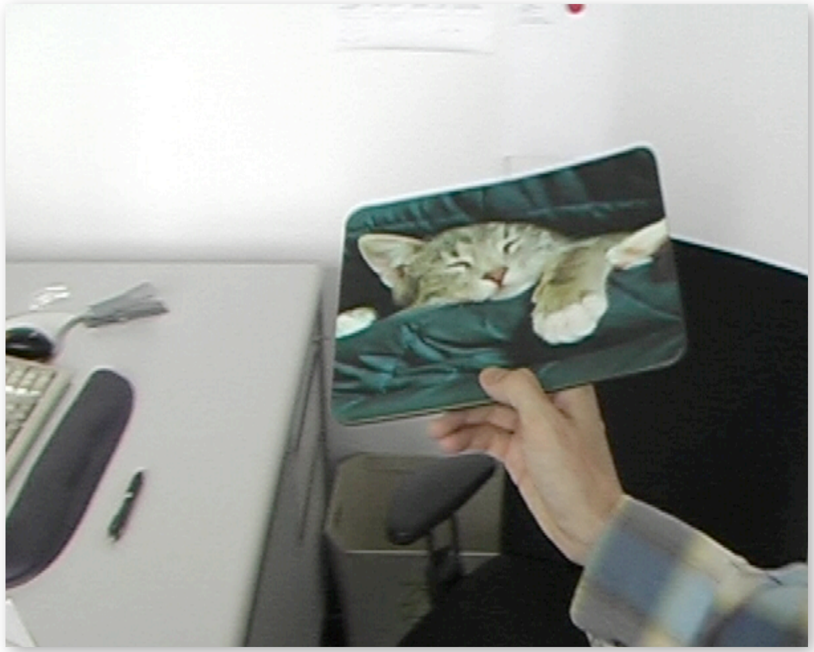
Registered image(s) of the object to detect

3D Object Detection

Keypoint detection (Harris, extrema of Laplacian, affine regions,...);



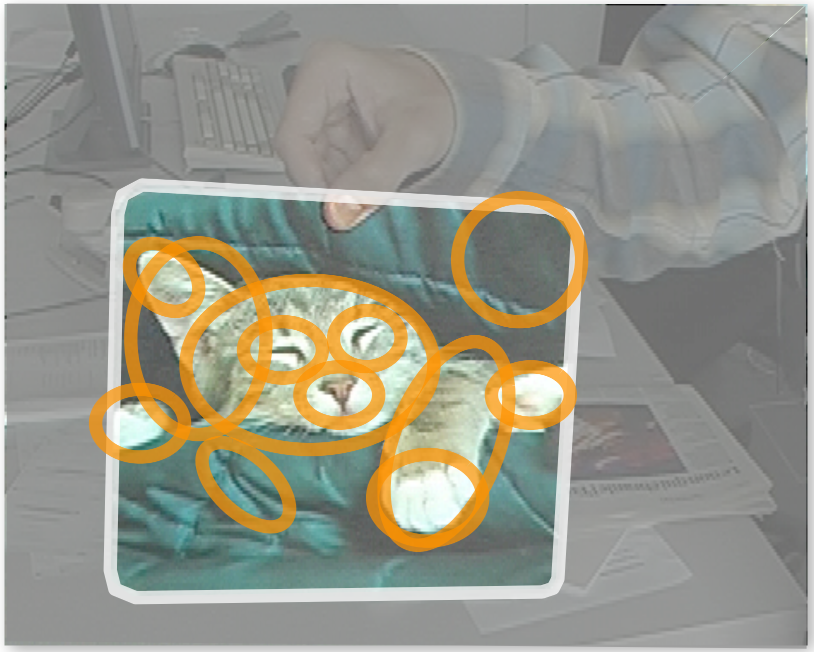
Registered image(s) of the object to detect



Input image

3D Object Detection

Keypoint detection (Harris, extrema of Laplacian, affine regions,...);



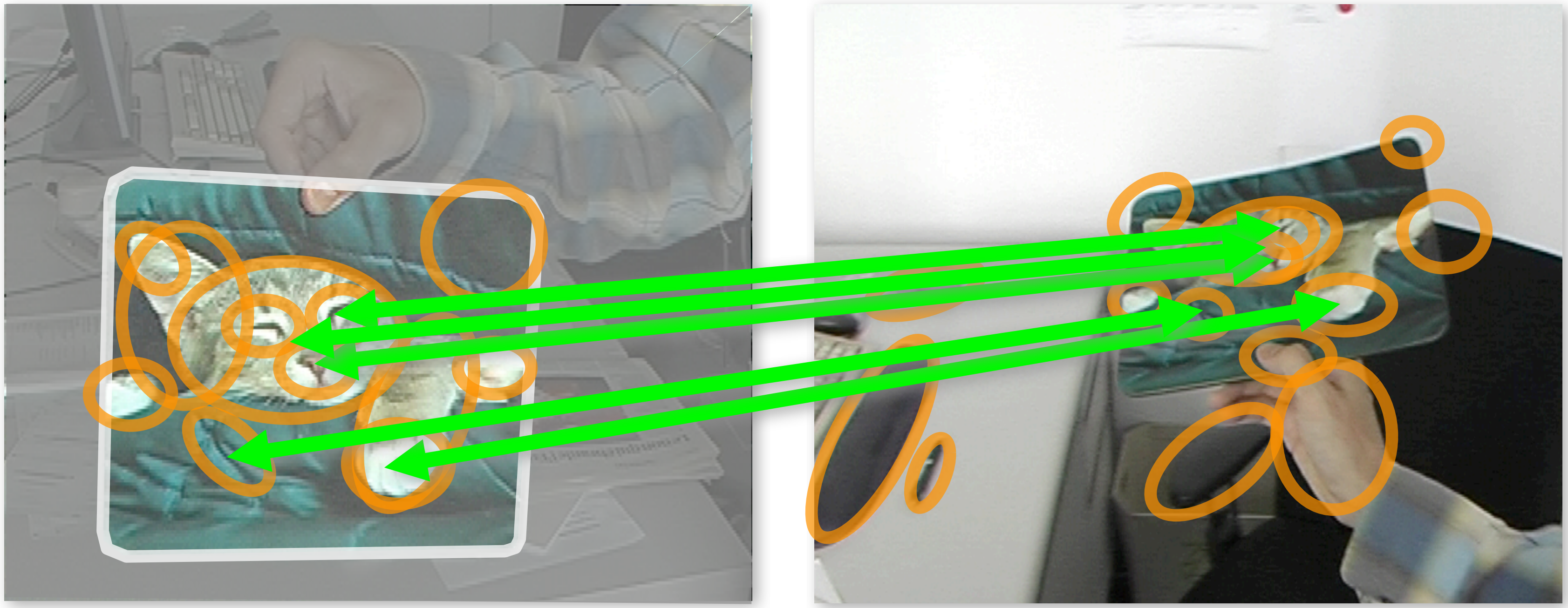
Registered image(s) of the object to detect

Input image

3D Object Detection

Keypoint detection (Harris, extrema of Laplacian, affine regions,...);

Keypoint recognition (descriptor matching or classification);



Registered image(s) of the object to detect

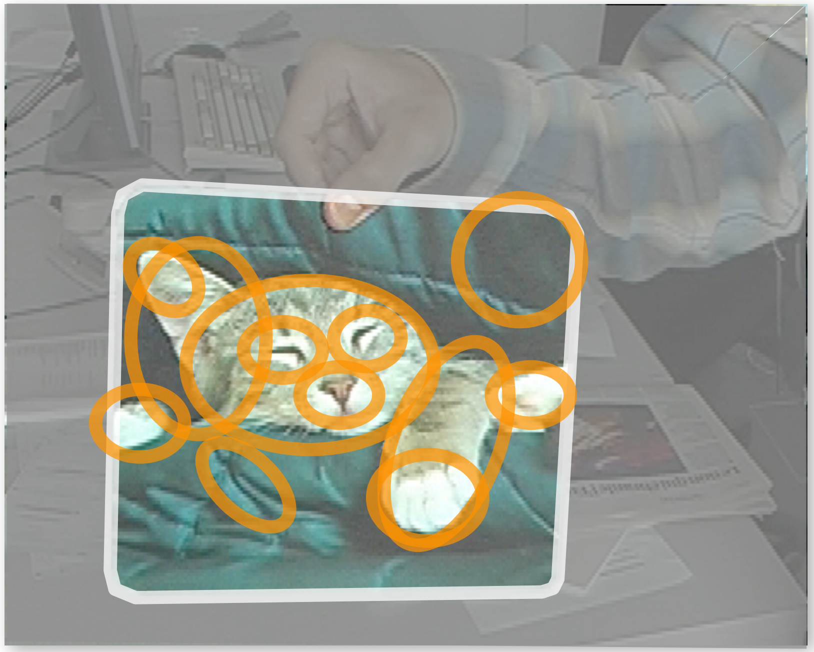
Input image

3D Object Detection

Keypoint detection (Harris, extrema of Laplacian, affine regions,...);

Keypoint recognition (descriptor matching or classification);

Robust pose estimation (RANSAC+P3P, ...).



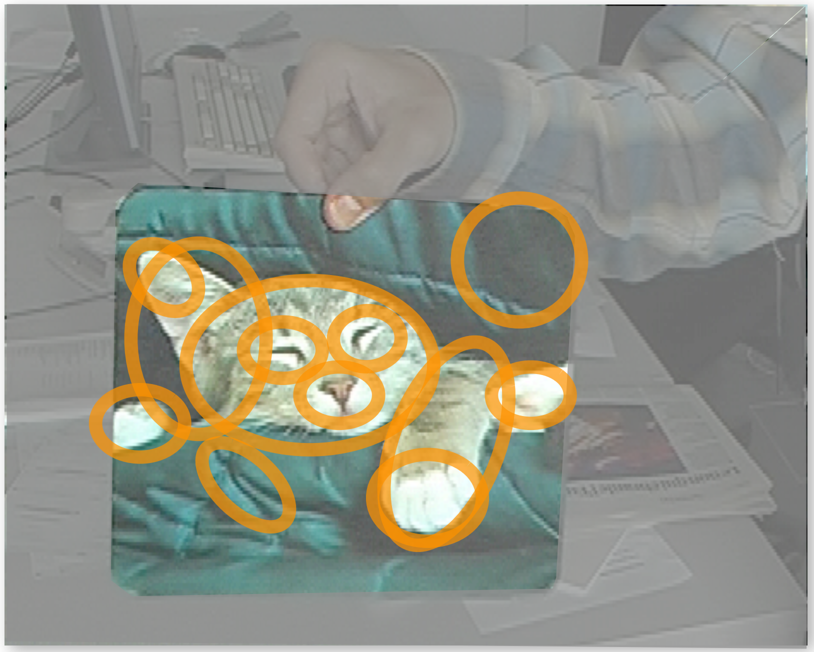
Registered image(s) of the object to detect

Input image

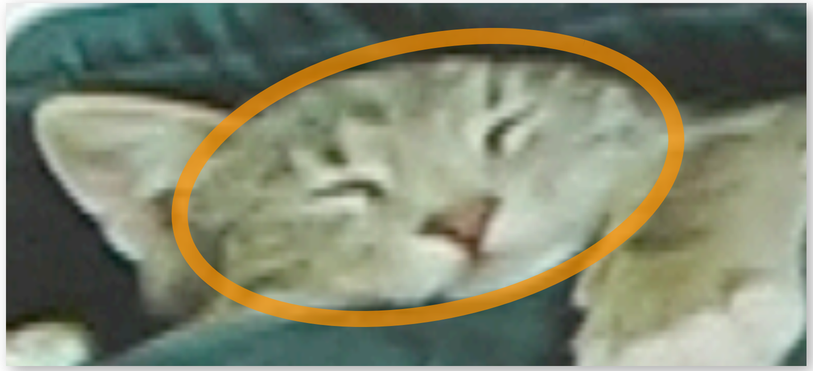
Standard Approach to Keypoint-Based Object Detection

Standard Approach

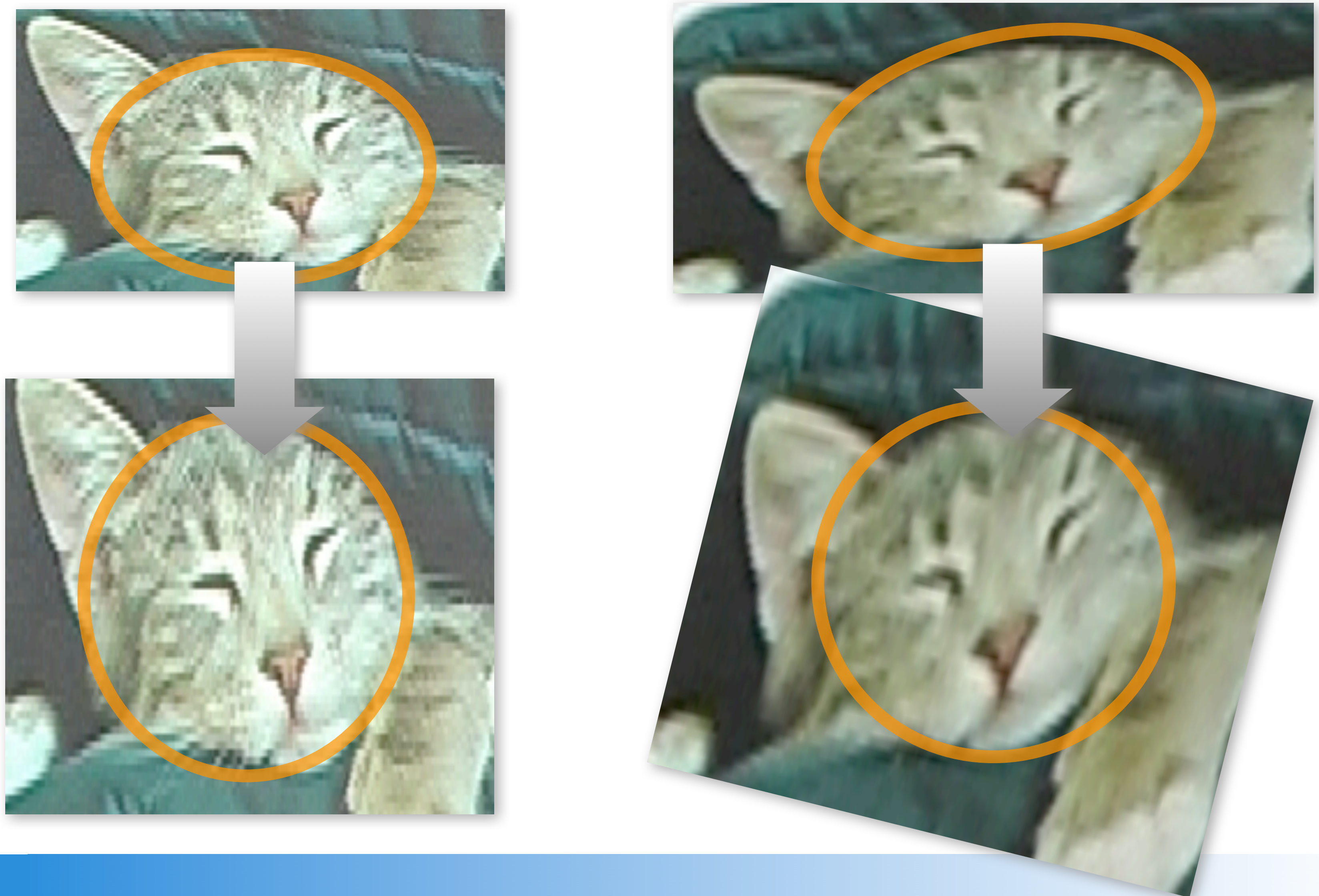
Step 1: Detection invariant to scale and rotation, or perspective transformation



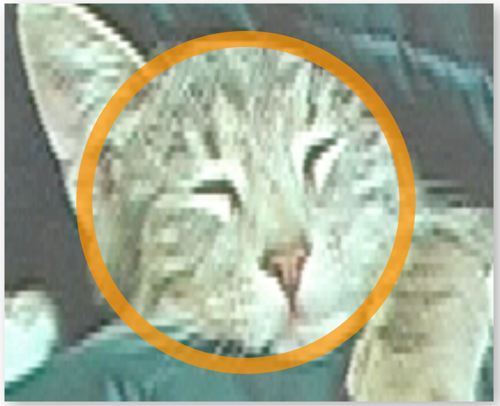
Standard Approach
Step 2: Patch rectification



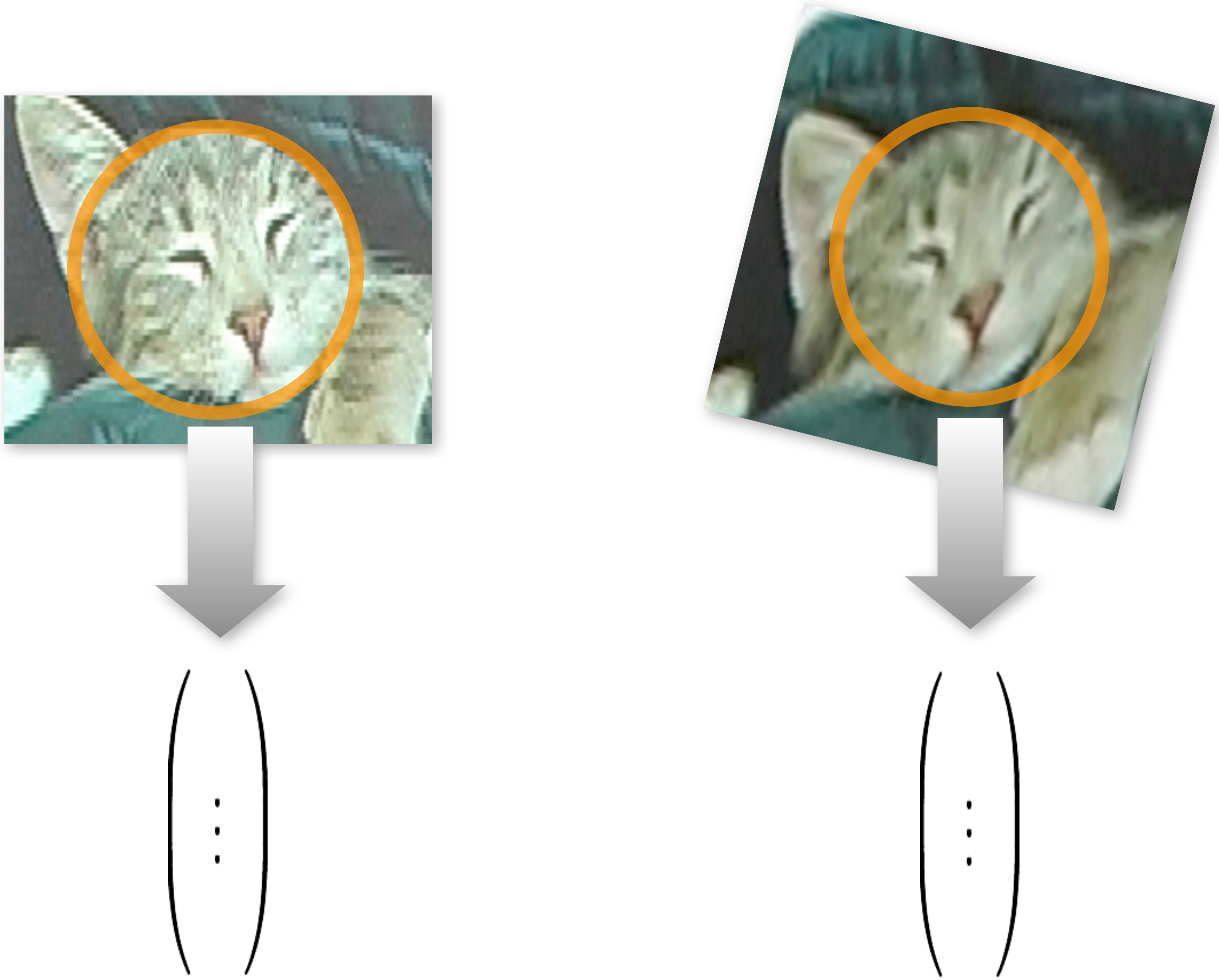
Standard Approach
Step 2: Patch rectification



Standard Approach
Step 3: Build description vector

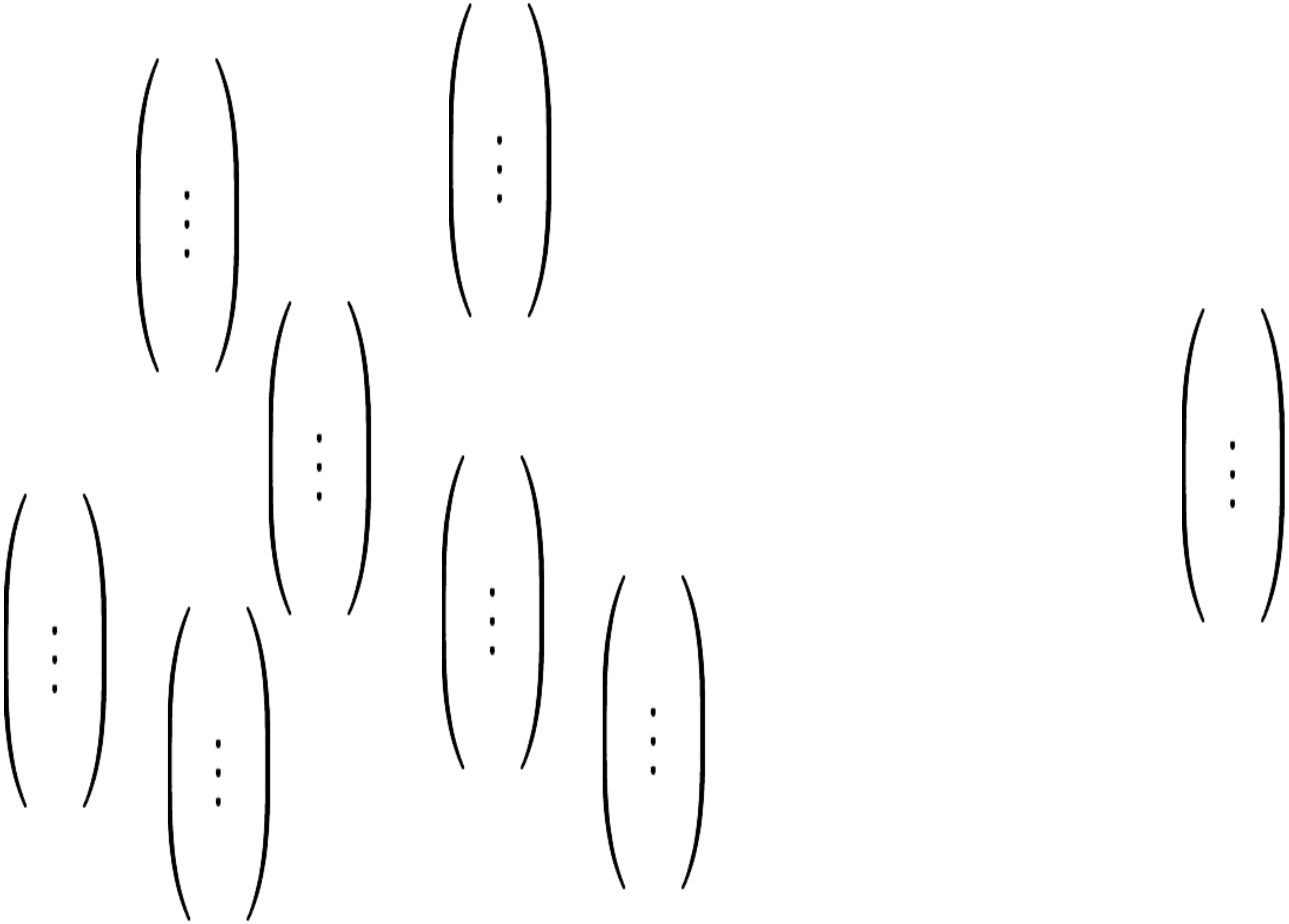


Standard Approach
Step 3: Build description vector

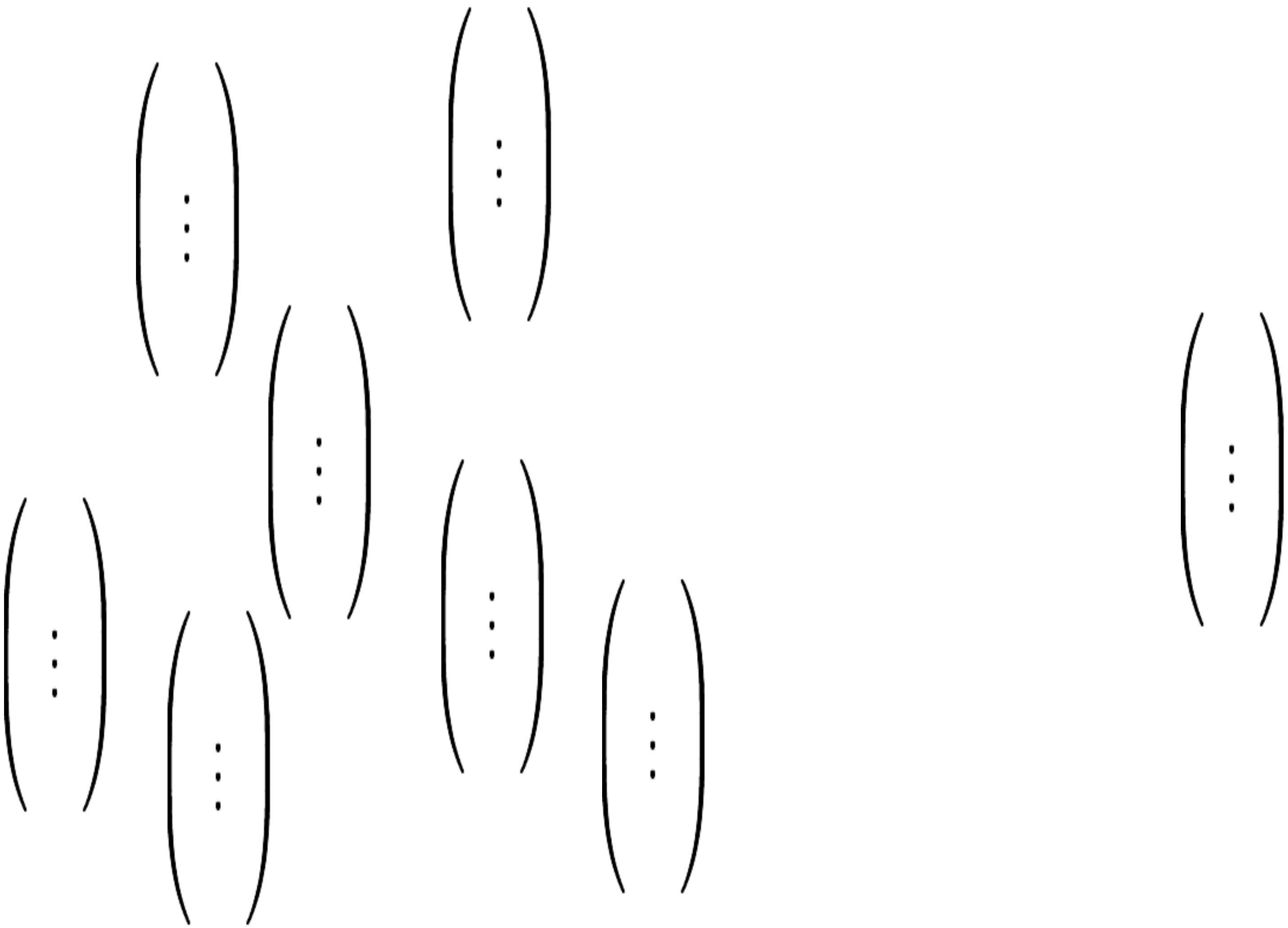


Standard Approach

Step 4: Match description vectors

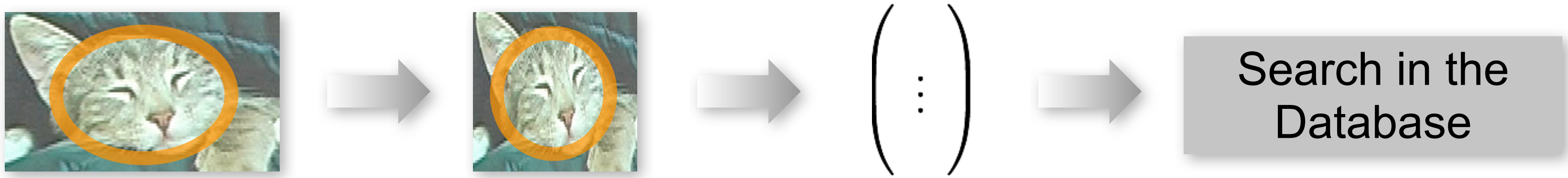


Standard Approach
Step 4: Match description vectors



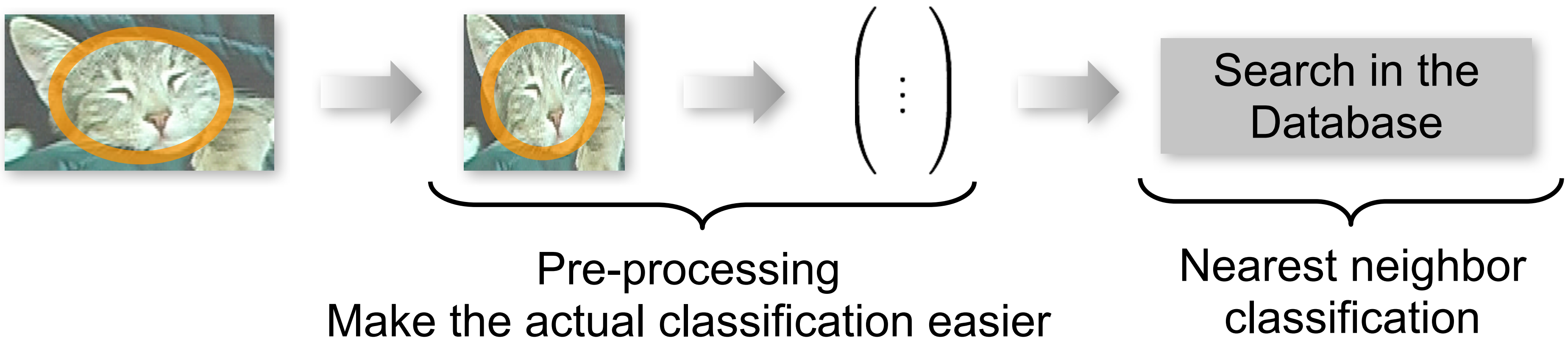
Keypoint Recognition

The standard approach is a particular case of classification:



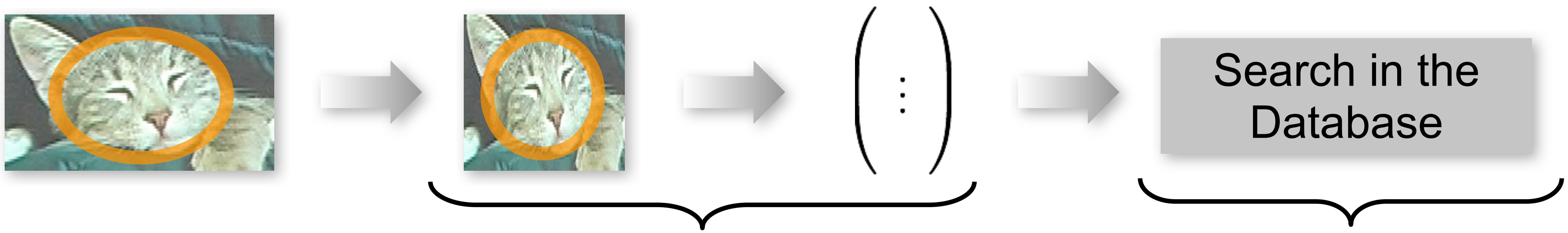
Keypoint Recognition

The standard approach is a particular case of classification:



Keypoint Recognition

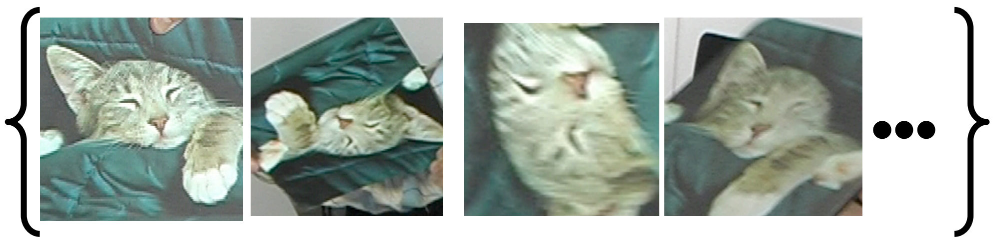
The standard approach is a particular case of classification:



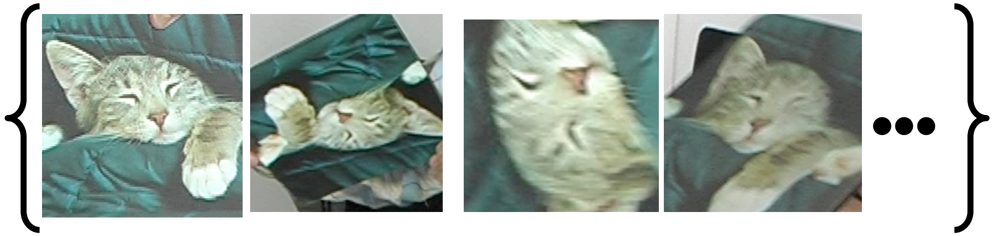
Pre-processing
Make the actual classification easier

Nearest neighbor
classification

One class per keypoint: the set of the keypoint's possible appearances under various perspective, lighting, noise...

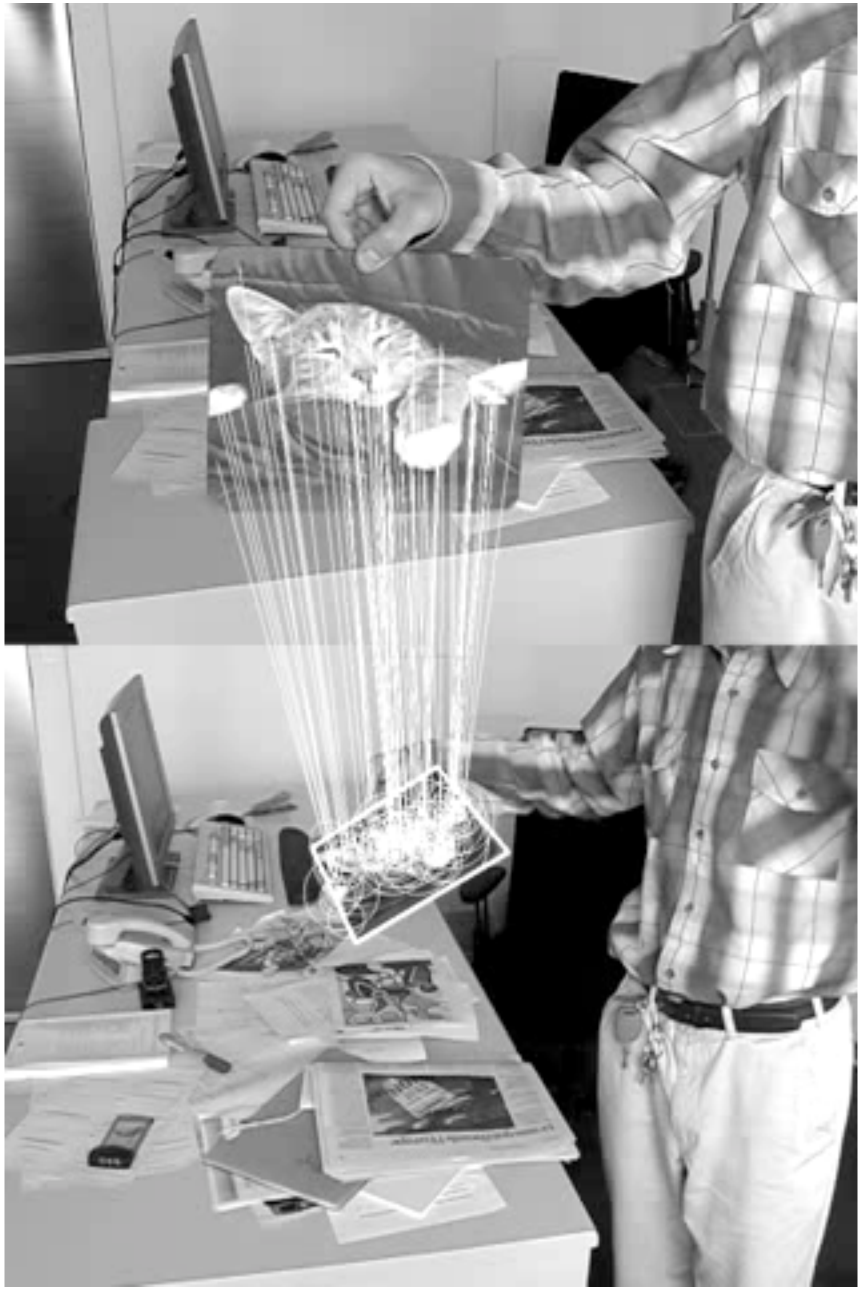


Training phase



Classifier

Used at run-time to recognize the keypoints




LE MONDE DES MONTAGNES
Camille Scherrer - ECAL / University of art and design Lausanne
Diplome Project - Media&interaction design / 2008

Patch Classification with Ferns

We are looking for $\arg\max_i P(C = c_i \mid \mathbf{patch})$

We are looking for $\arg\max_i P(C = c_i \mid \mathbf{patch})$

If $\mathbf{patch}$ can be represented by a set of image features $\{f_i\}$:

We are looking for $\arg\max_i P(C = c_i \mid \mathbf{patch})$

If **patch** can be represented by a set of image features $\{f_i\}$:

$$P(C = c_i \mid \mathbf{patch}) = P(C = c_i \mid f_1, f_2, \ldots f_n, f_{n+1}, \ldots \ldots f_N)$$

We are looking for $\arg\max_i P(C = c_i \mid \textbf{patch})$

If **patch** can be represented by a set of image features $\{f_i\}$:

$$P(C = c_i \mid \textbf{patch}) = P(C = c_i \mid f_1, f_2, \ldots f_n, f_{n+1}, \ldots \ldots f_N)$$

which is proportional to

$$P(f_1, f_2, \cdots f_n,\ f_{n+1}, \cdots \cdots f_N \mid C = c_i)$$

but complete representation of the joint distribution infeasible.

We are looking for $\arg\max_i P(C = c_i \mid \textbf{patch})$

If **patch** can be represented by a set of image features $\{f_i\}$:

$$P(C = c_i \mid \textbf{patch}) = P(C = c_i \mid f_1, f_2, \ldots f_n, f_{n+1}, \ldots \ldots f_N)$$

which is proportional to

$$P(f_1, f_2, \cdots f_n,\ f_{n+1}, \cdots \cdots f_N \mid C = c_i)$$

but complete representation of the joint distribution infeasible.

Naive Bayesian ignores the correlation:

$$\approx \prod_j P(f_j \mid C = c_i)$$

We are looking for $\arg\max_i P(C = c_i \mid \mathbf{patch})$

If **patch** can be represented by a set of image features $\{f_i\}$:

$$P(C = c_i \mid \mathbf{patch}) = P(C = c_i \mid f_1, f_2, \ldots f_n, f_{n+1}, \ldots \ldots f_N)$$

which is proportional to

$$P(f_1, f_2, \cdots f_n,\ f_{n+1}, \cdots \cdots f_N \mid C = c_i)$$

but complete representation of the joint distribution infeasible.

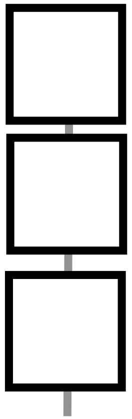
Naive Bayesian ignores the correlation:

$$\approx \prod_j P(f_j \mid C = c_i)$$

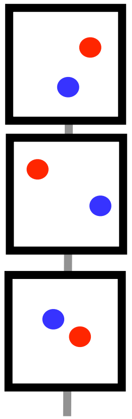
Compromise:

$$\approx P(f_1, f_2, \cdots f_n \mid C = c_i) \times P(f_{n+1}, \cdots f_{2n} \mid C = c_i) \times \cdots$$

Training



Training

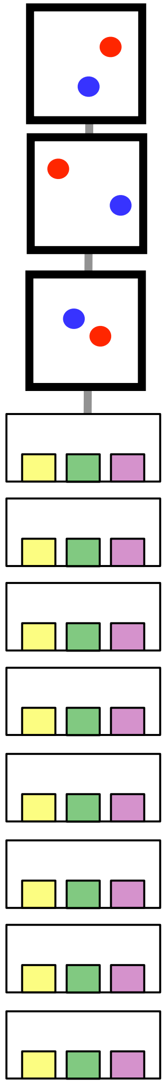


The tests compare the intensities of two pixels around the keypoint:

$$f_i = \begin{cases} 1 & \text{if } I(m_{i,1}) \leq I(m_{i,2}) \\ 0 & \text{otherwise} \end{cases}$$

Invariant to light change by any raising function.

Training



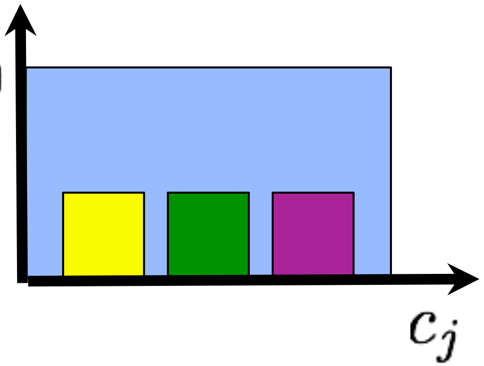
The tests compare the intensities of two pixels around the keypoint:

$$f_i = \begin{cases} 1 & \text{if } I(m_{i,1}) \leq I(m_{i,2}) \\ 0 & \text{otherwise} \end{cases}$$

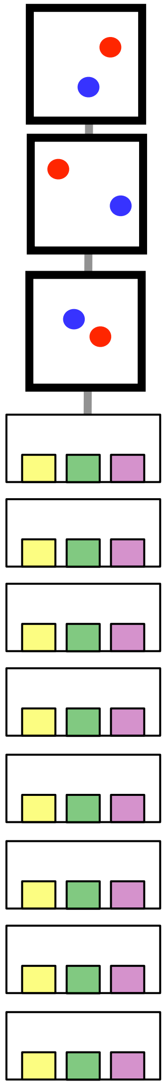
Invariant to light change by any raising function.

Posterior probabilities:

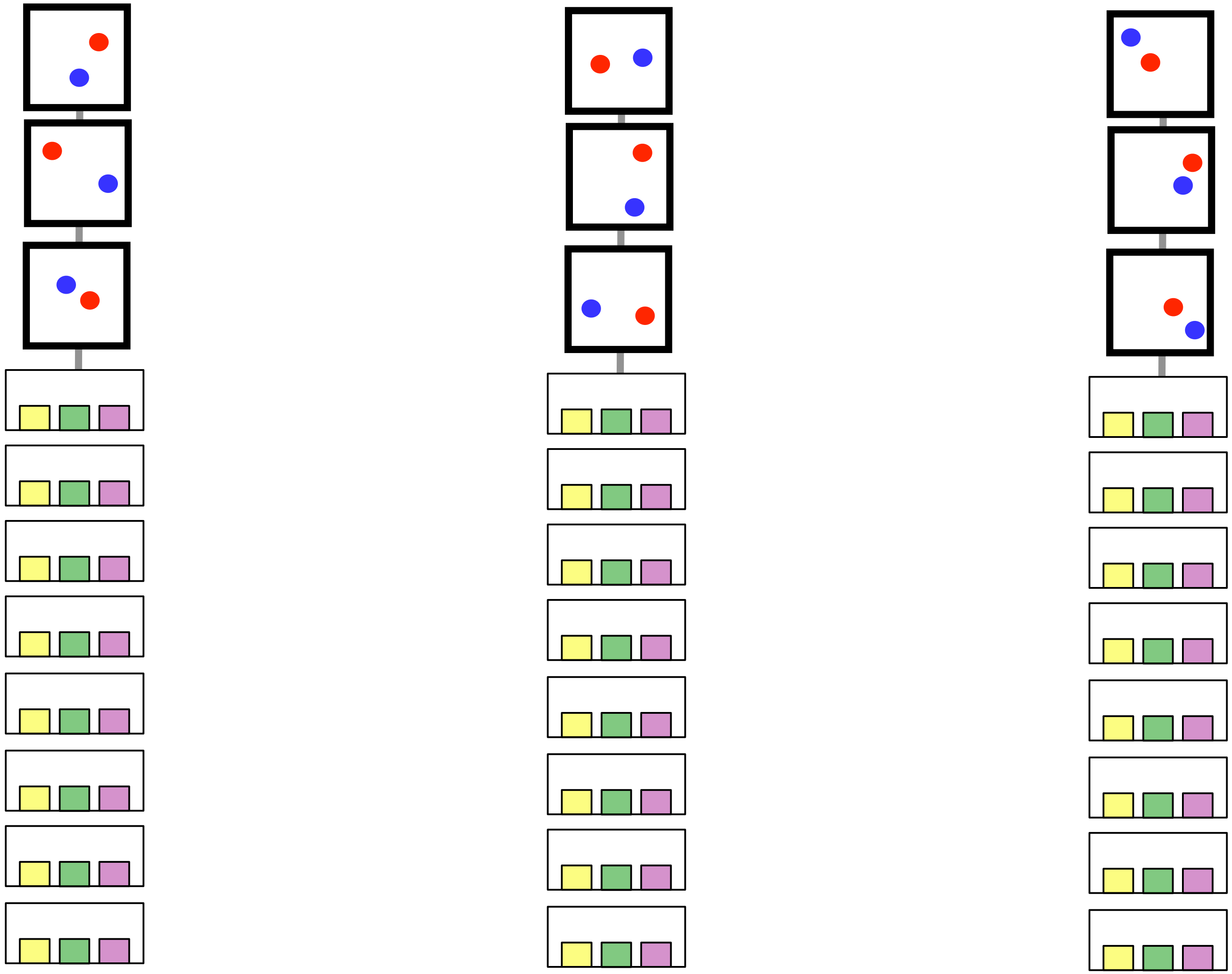
$$P(f_1, f_2, \cdots f_n \mid C = c_j)$$



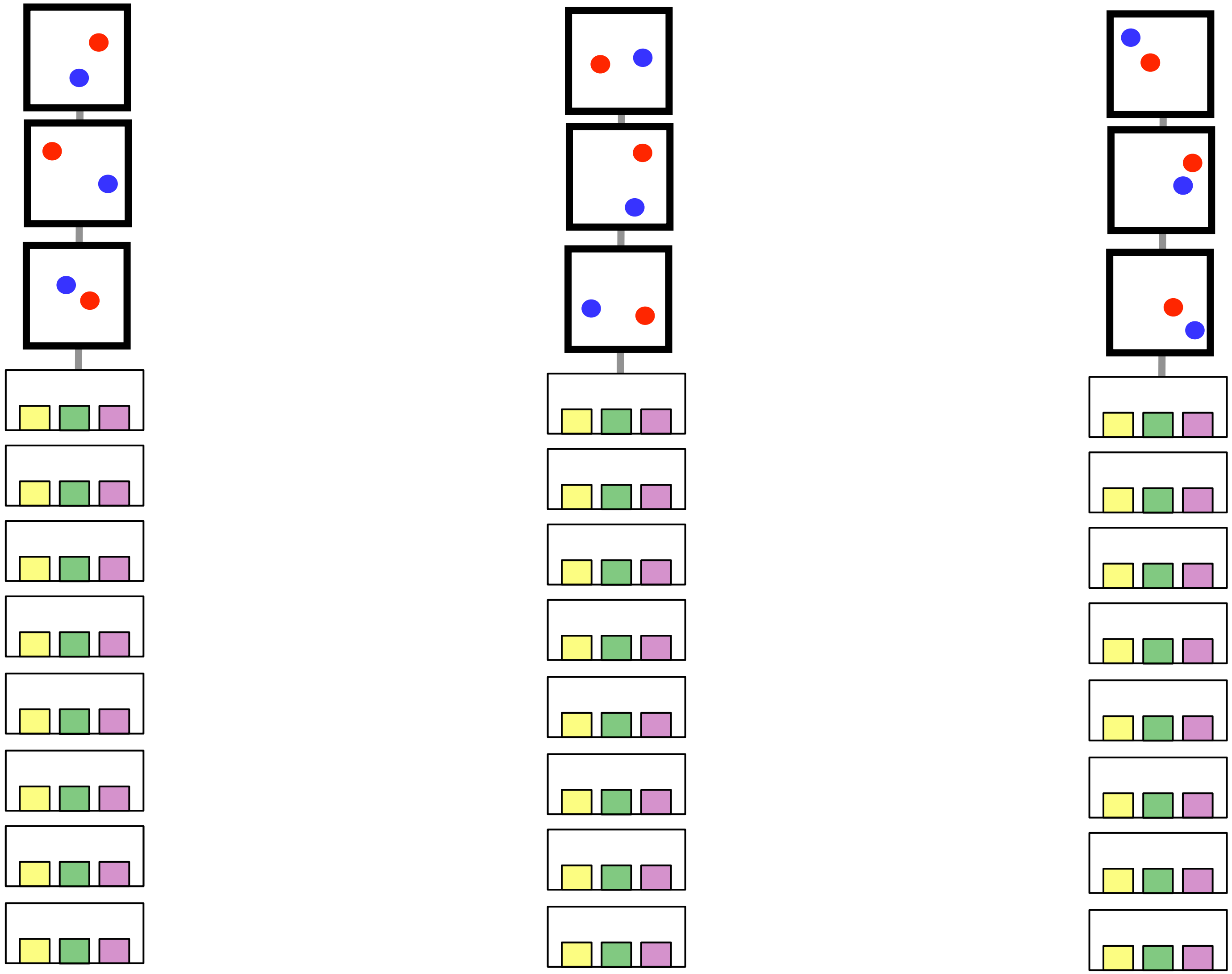
Training



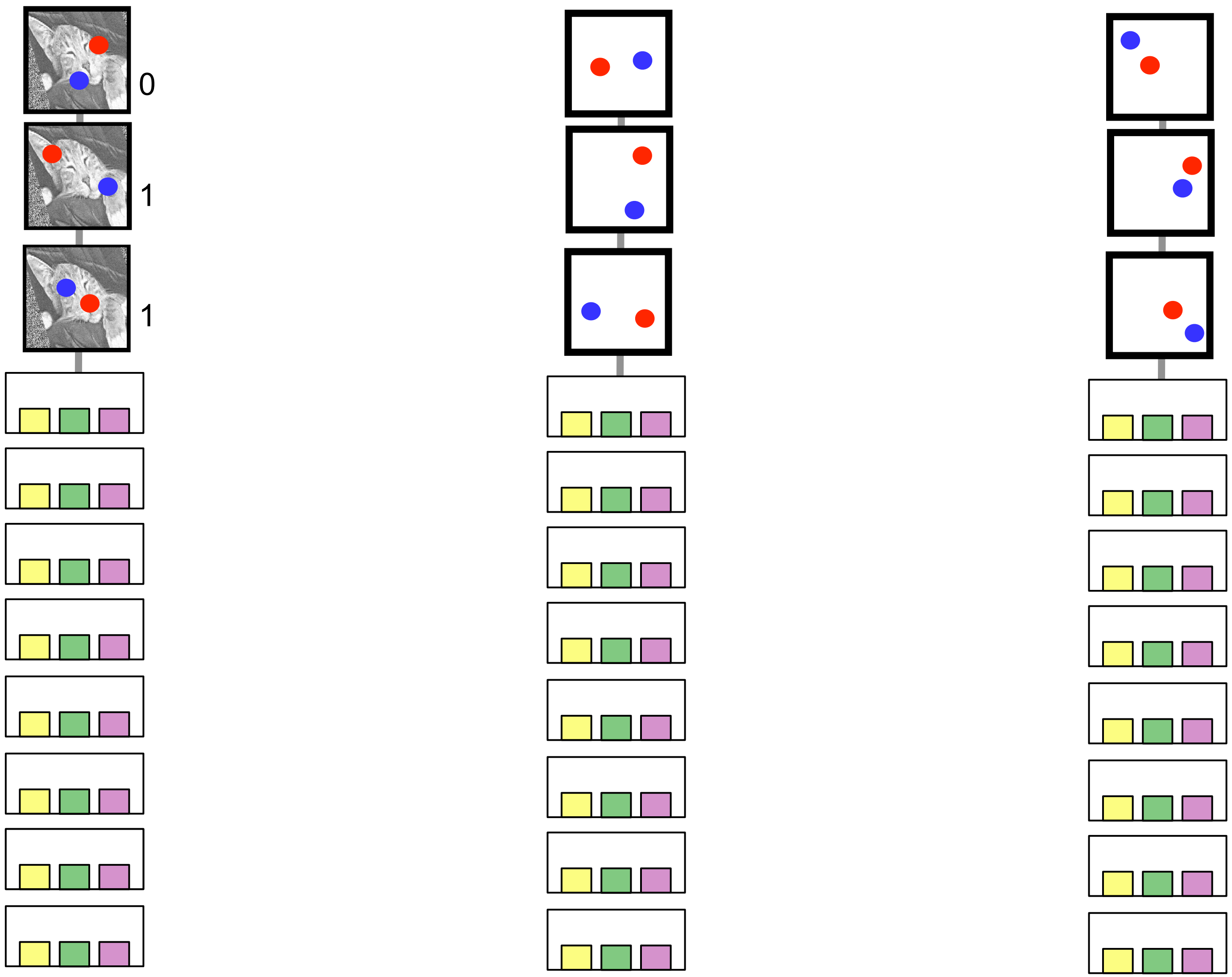
Training



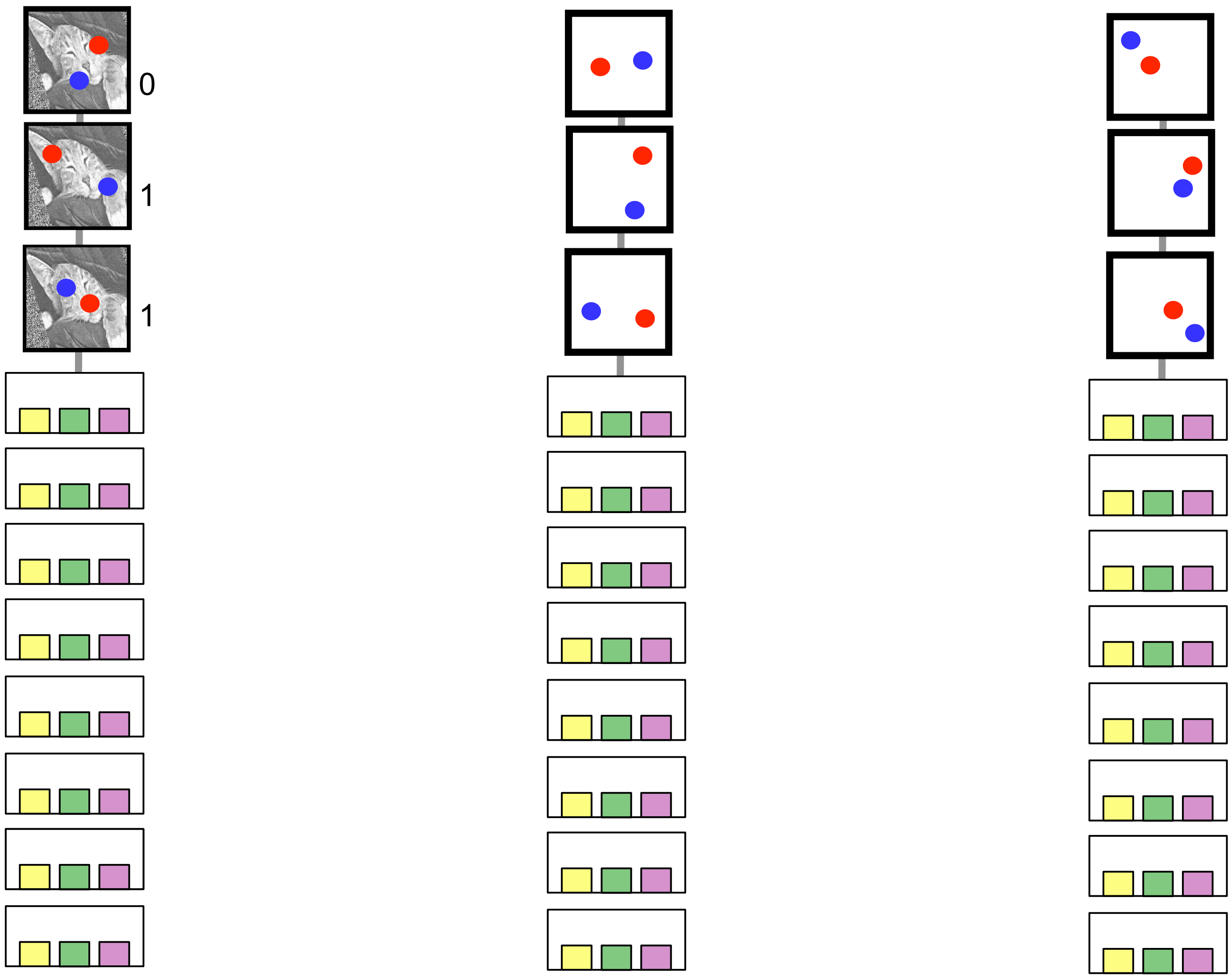
Training



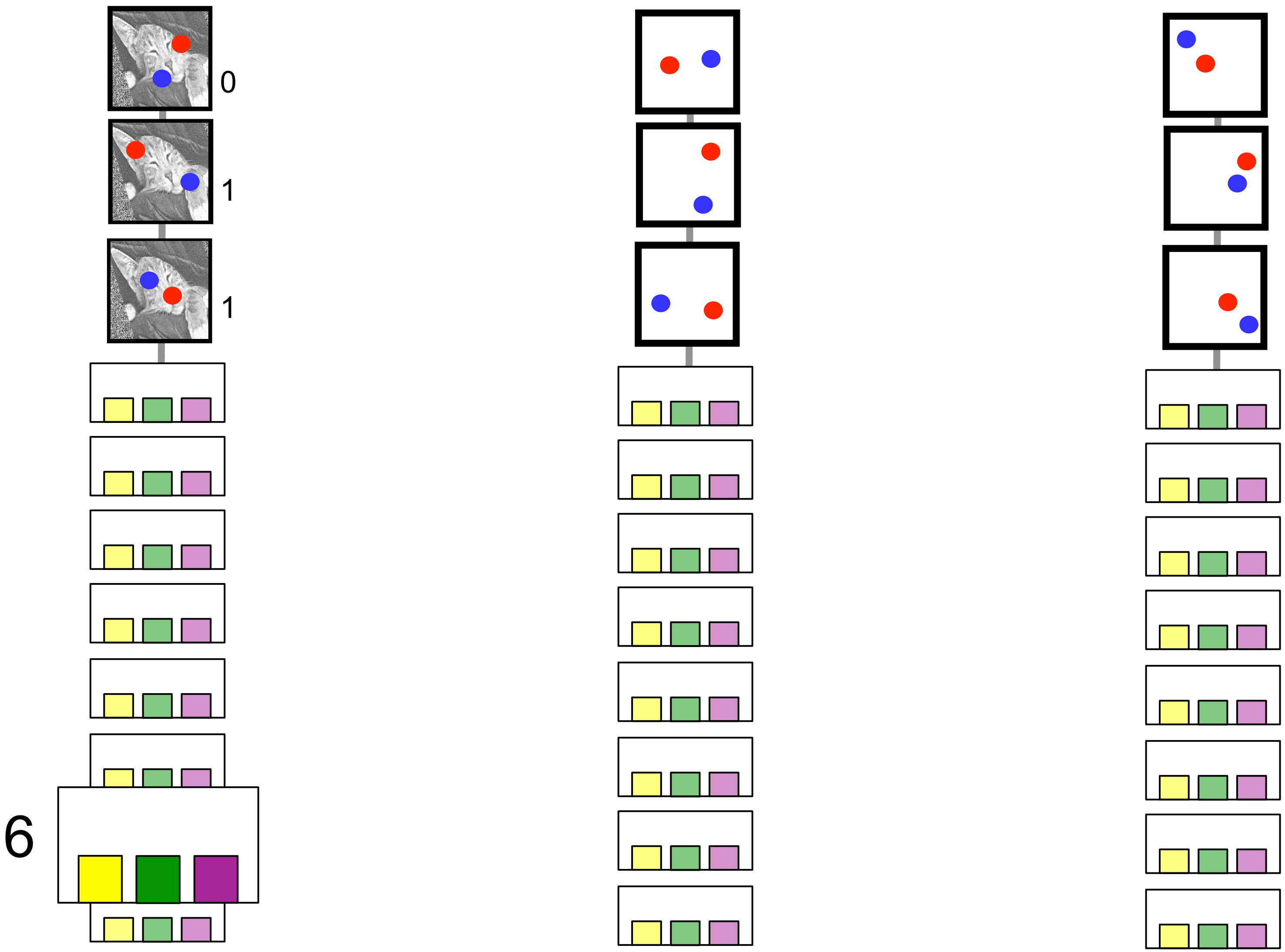
Training



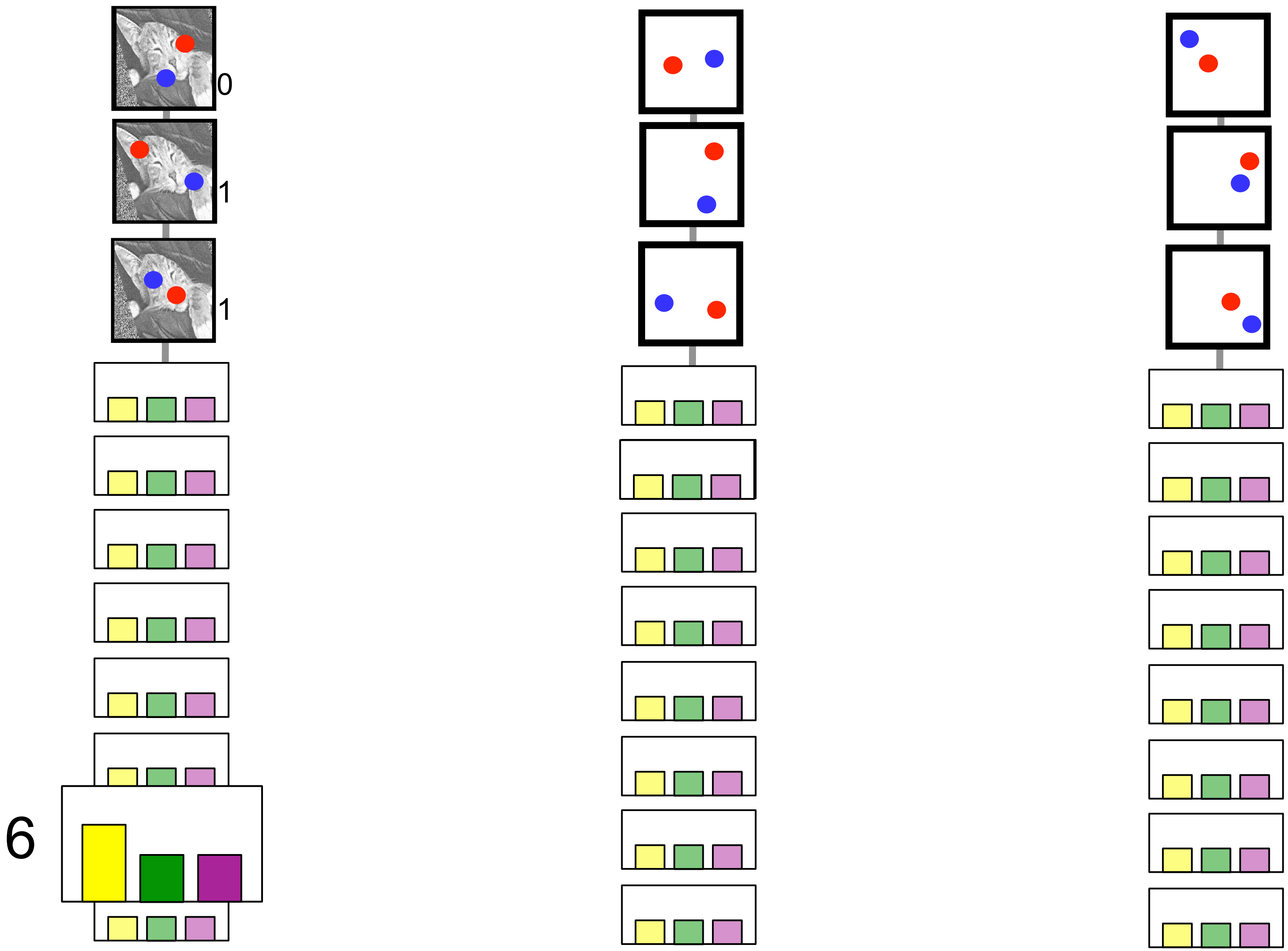
Training



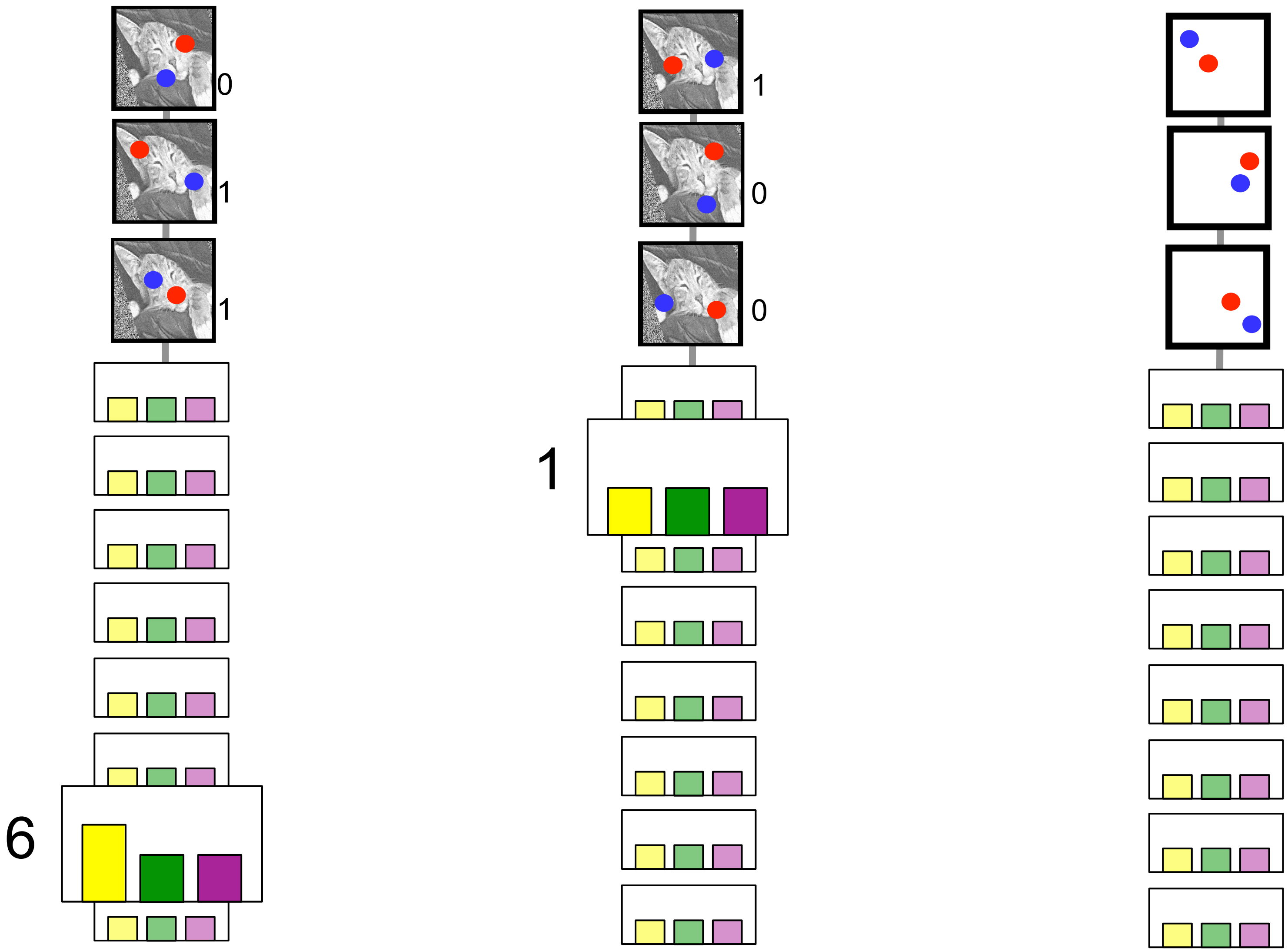
Training



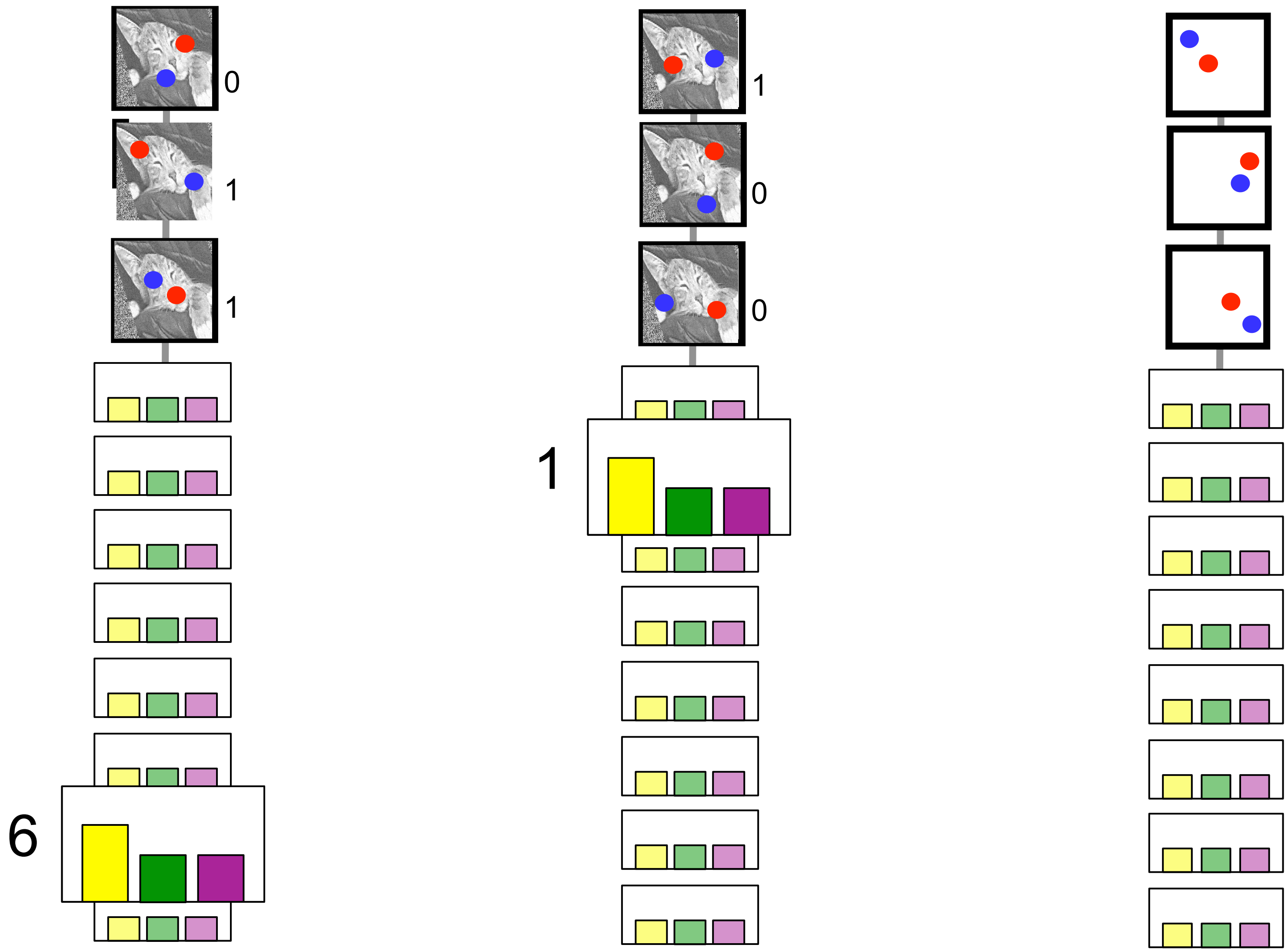
Training



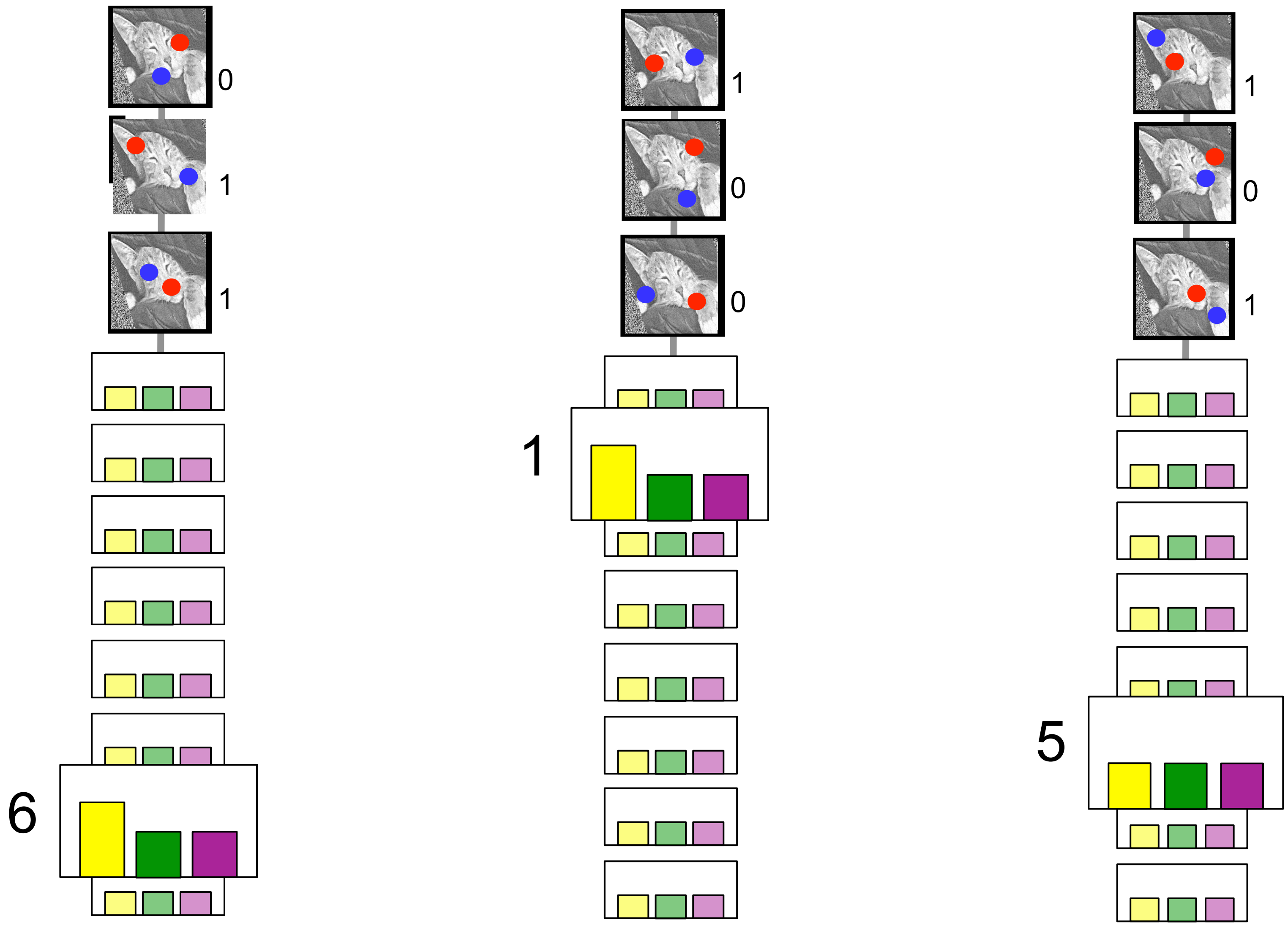
Training



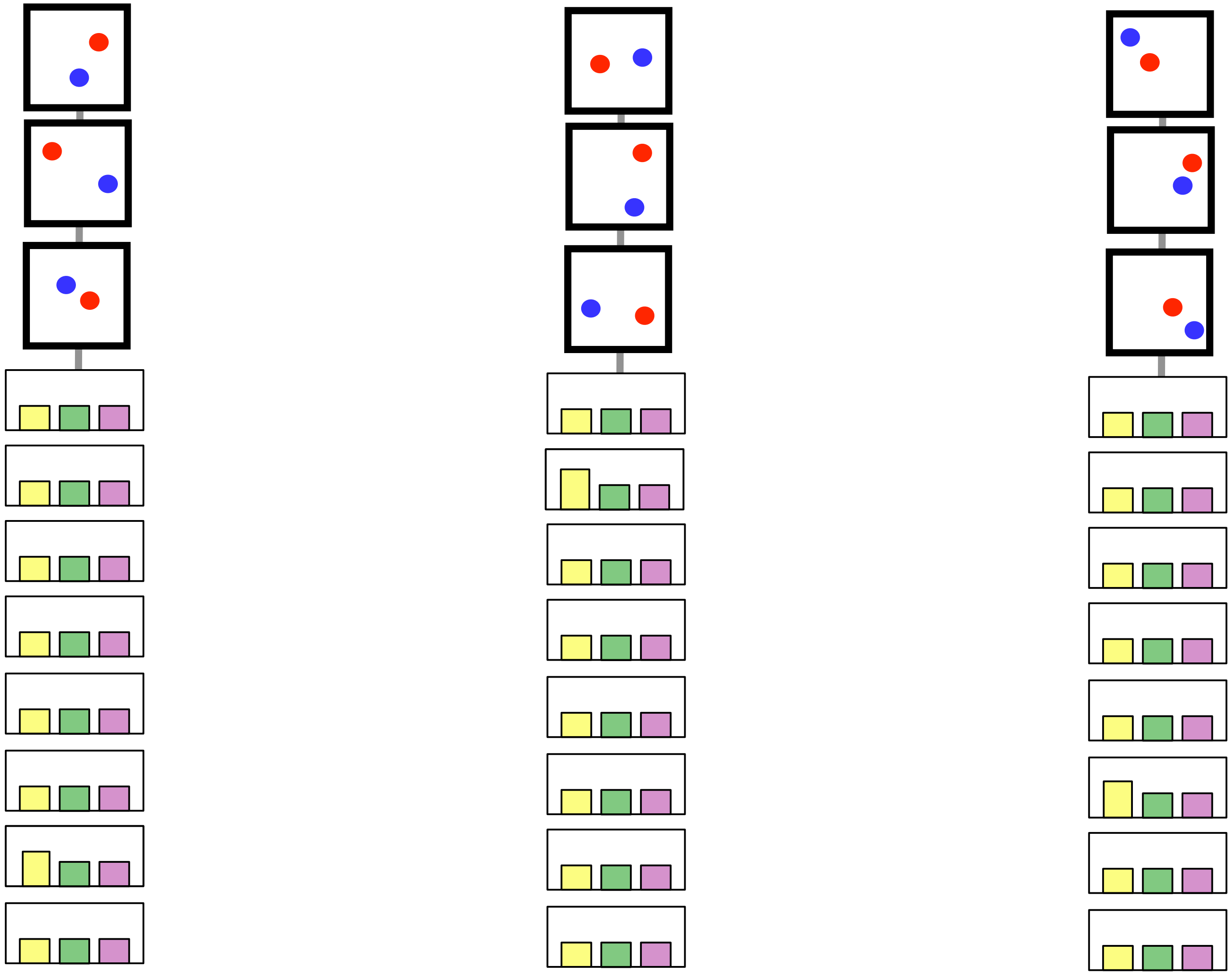
Training



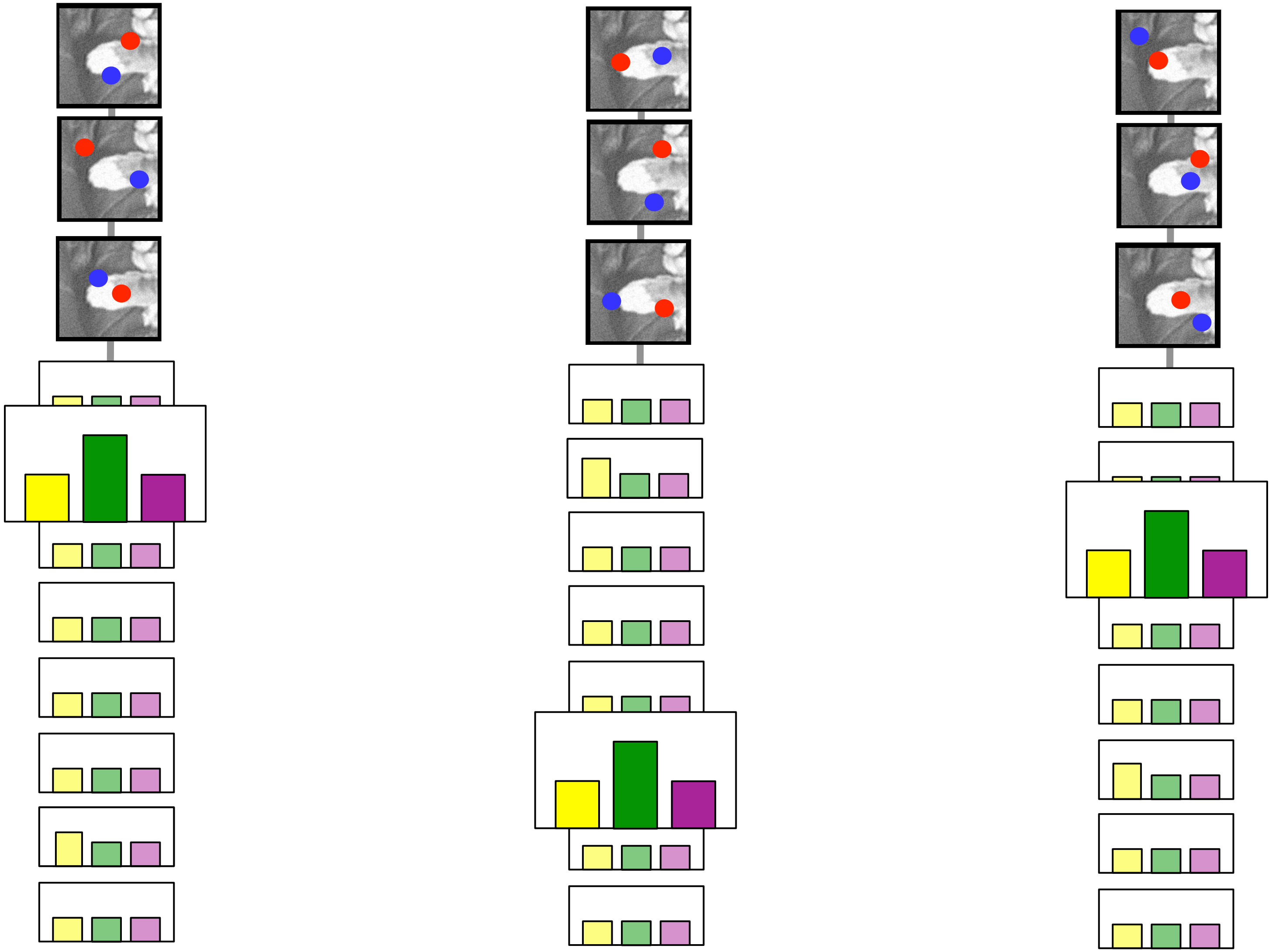
Training



Training

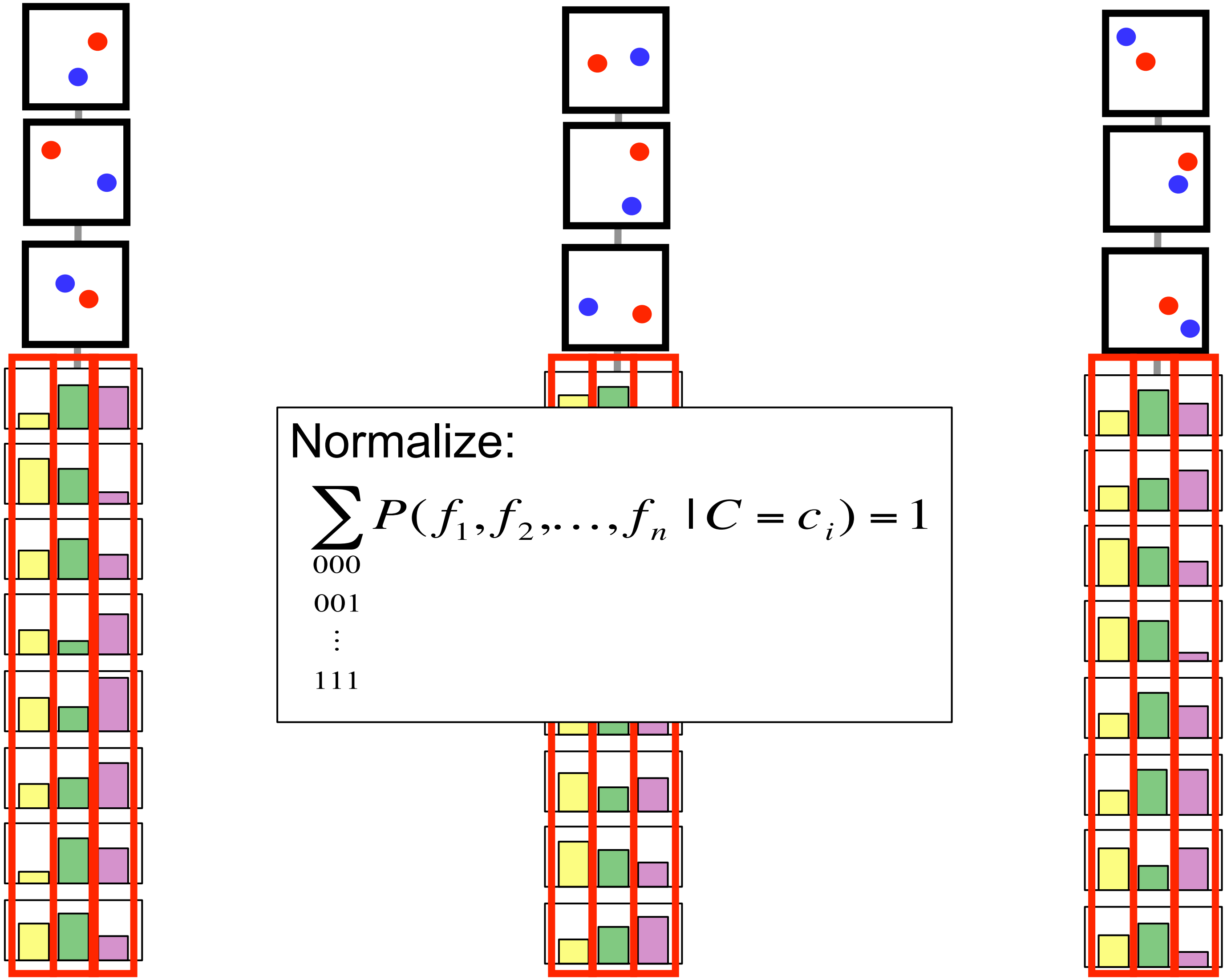


Training



Training Results

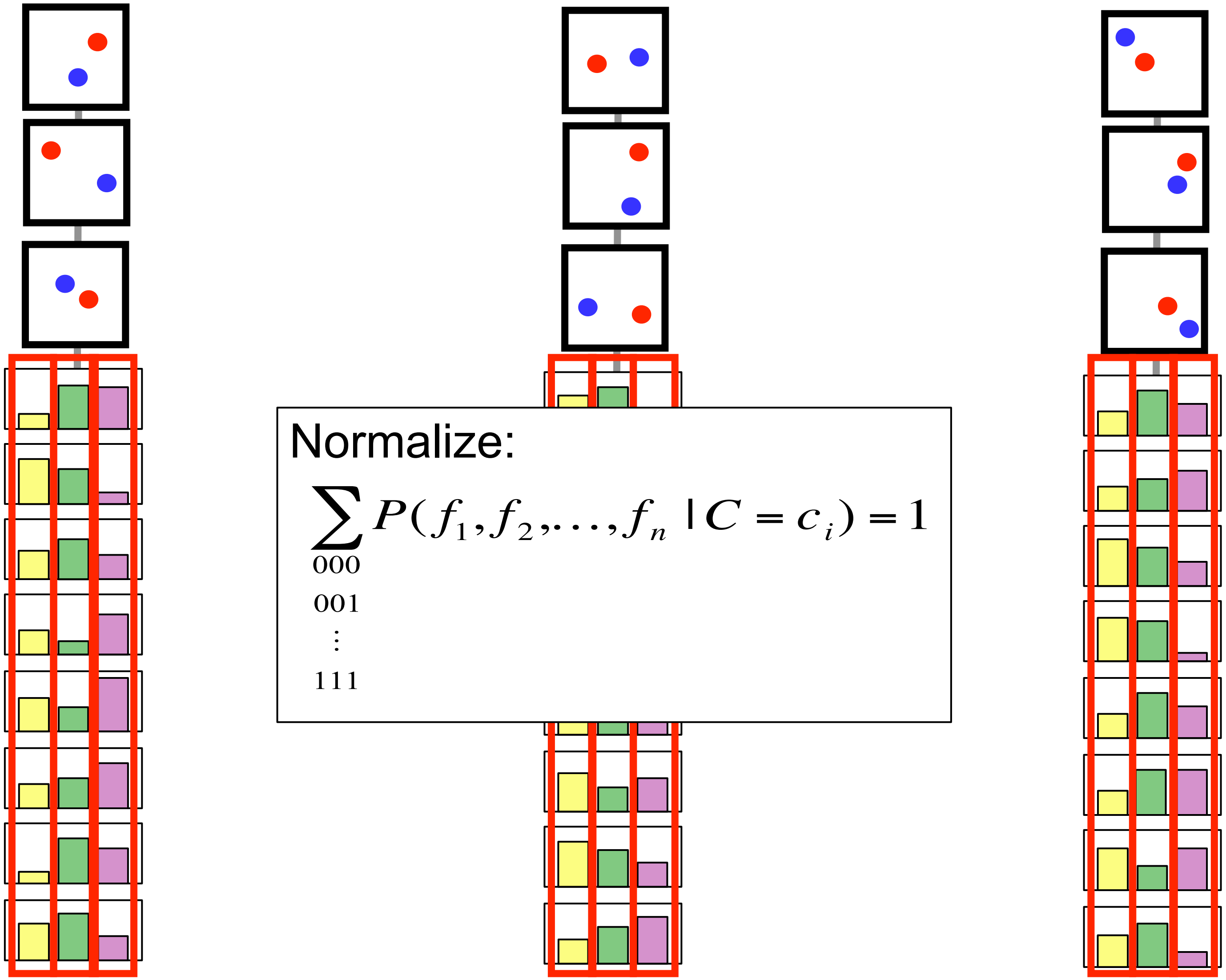
Training Results



Normalize:

$$\sum_{\substack{000\\001\\\vdots\\111}} P(f_1, f_2, \ldots, f_n \mid C = c_i) = 1$$

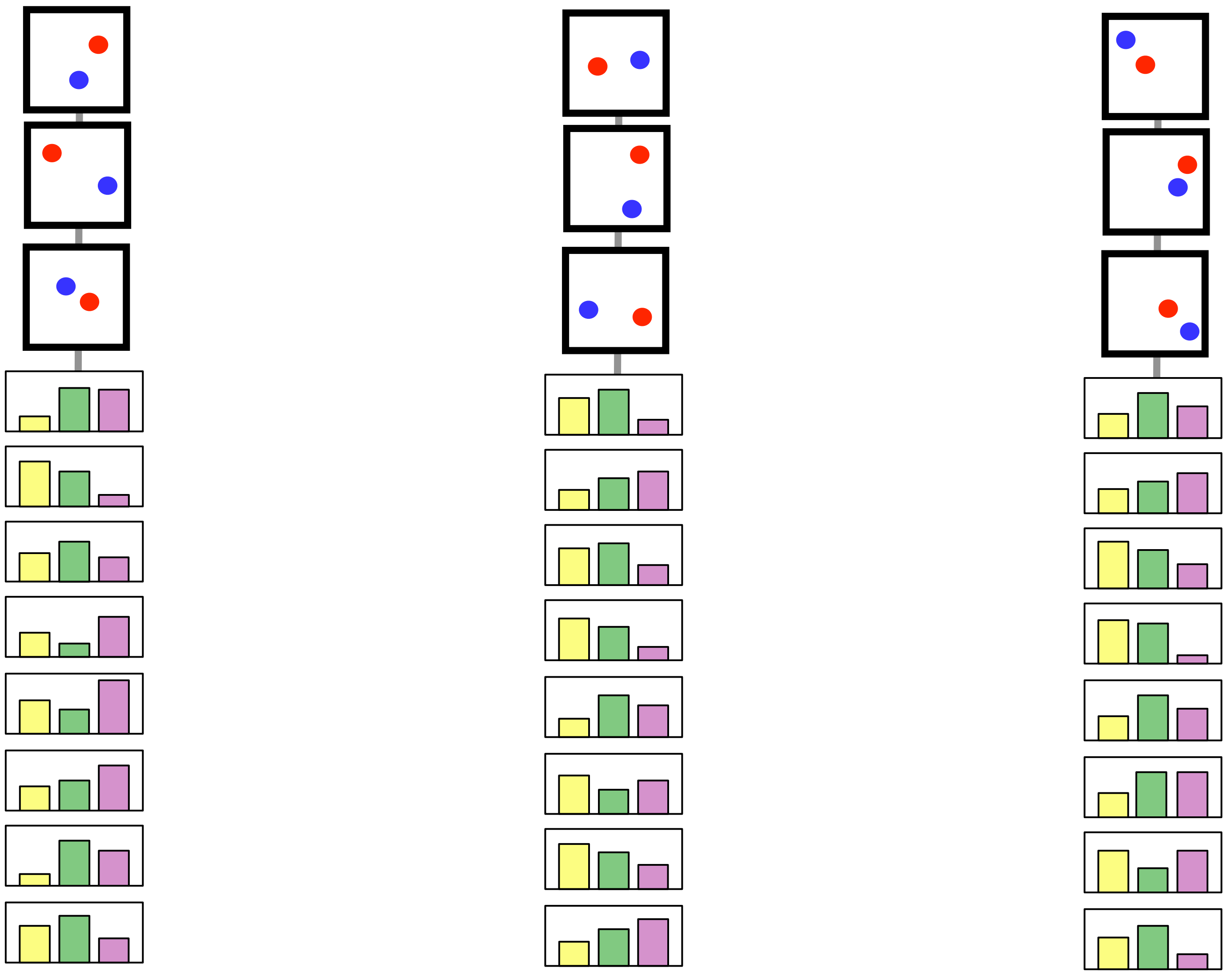
Training Results



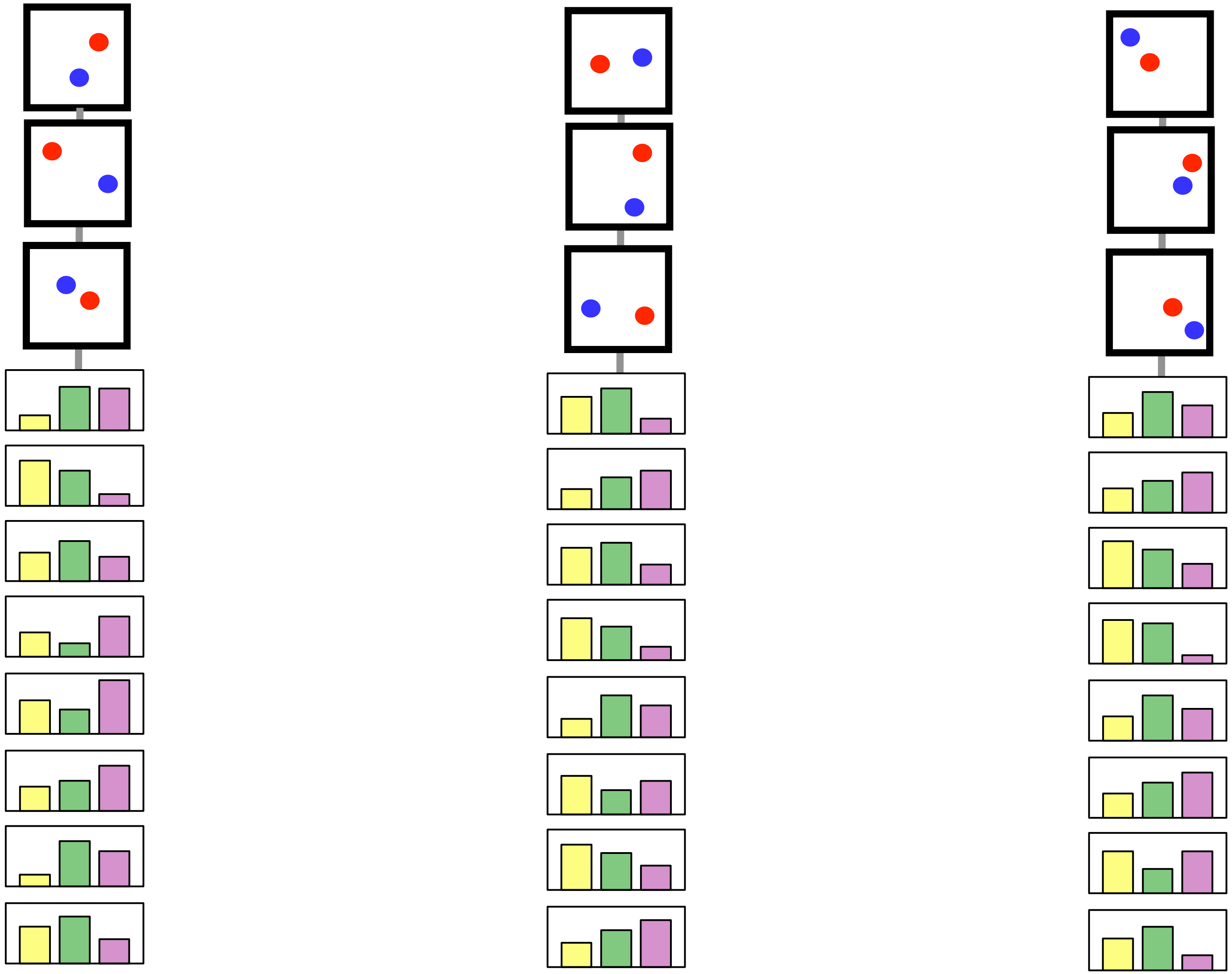
Normalize:

$$\sum_{\substack{000\\001\\\vdots\\111}} P(f_1, f_2, \ldots, f_n \mid C = c_i) = 1$$

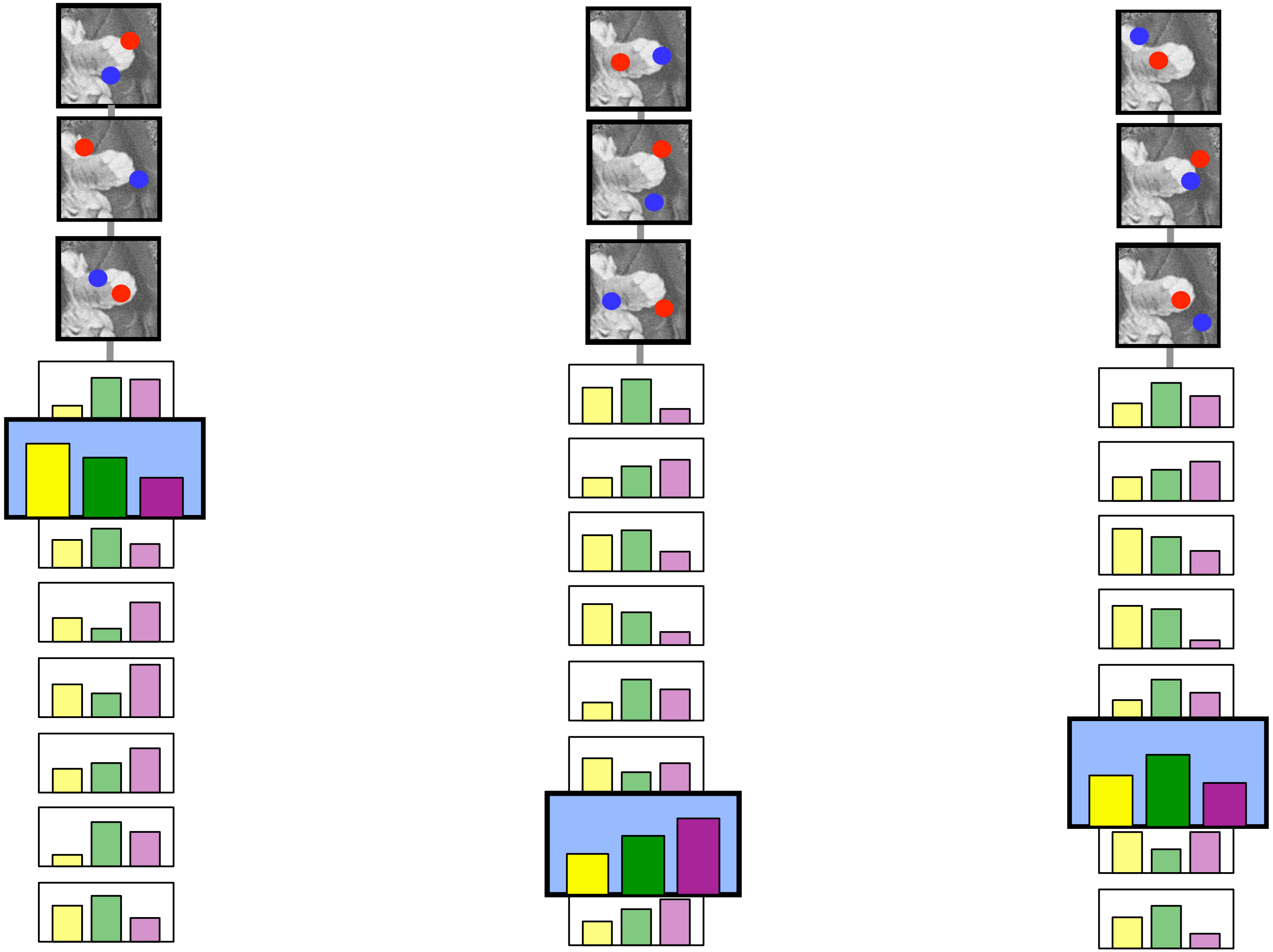
Training Results



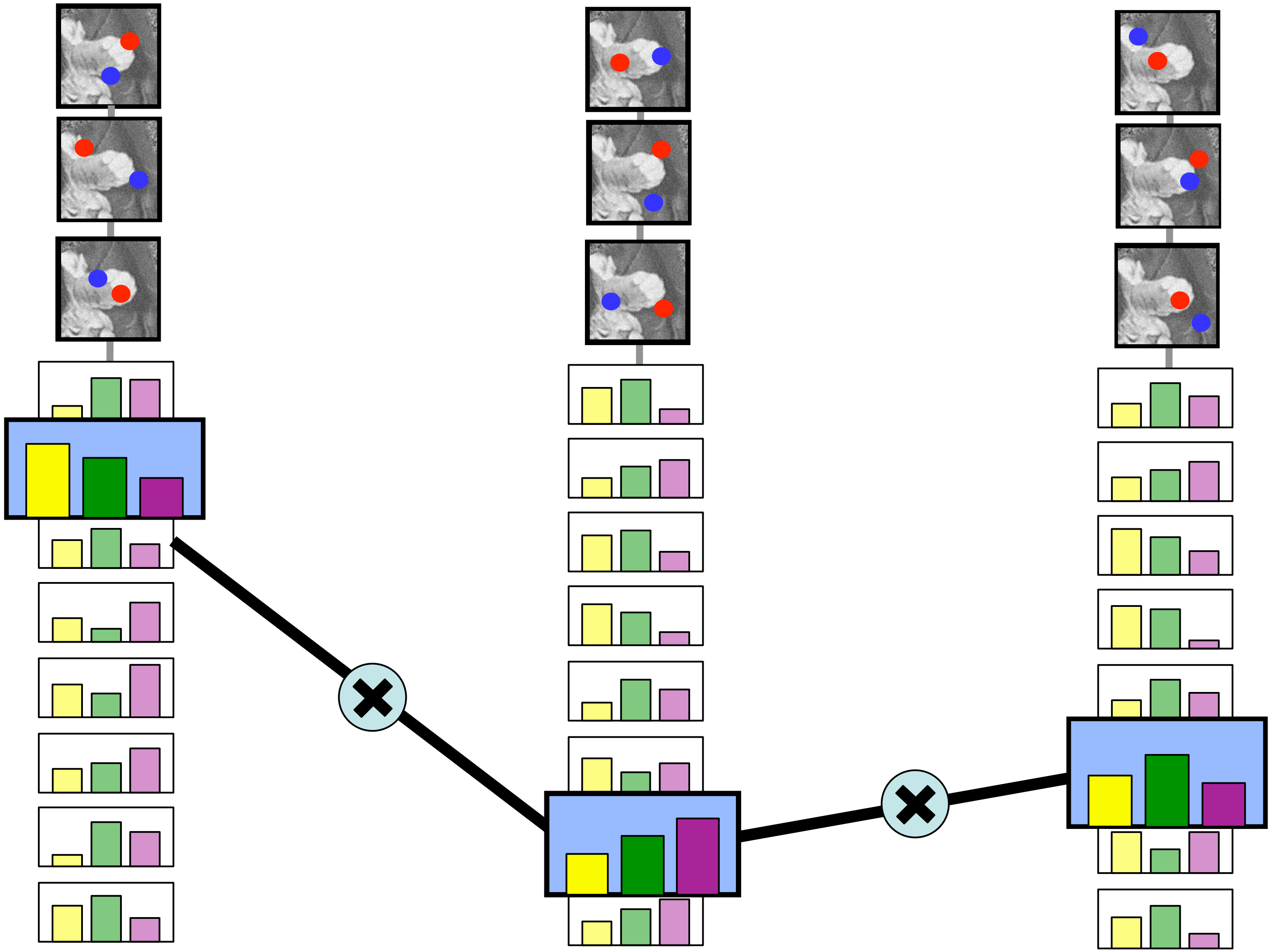
Recognition



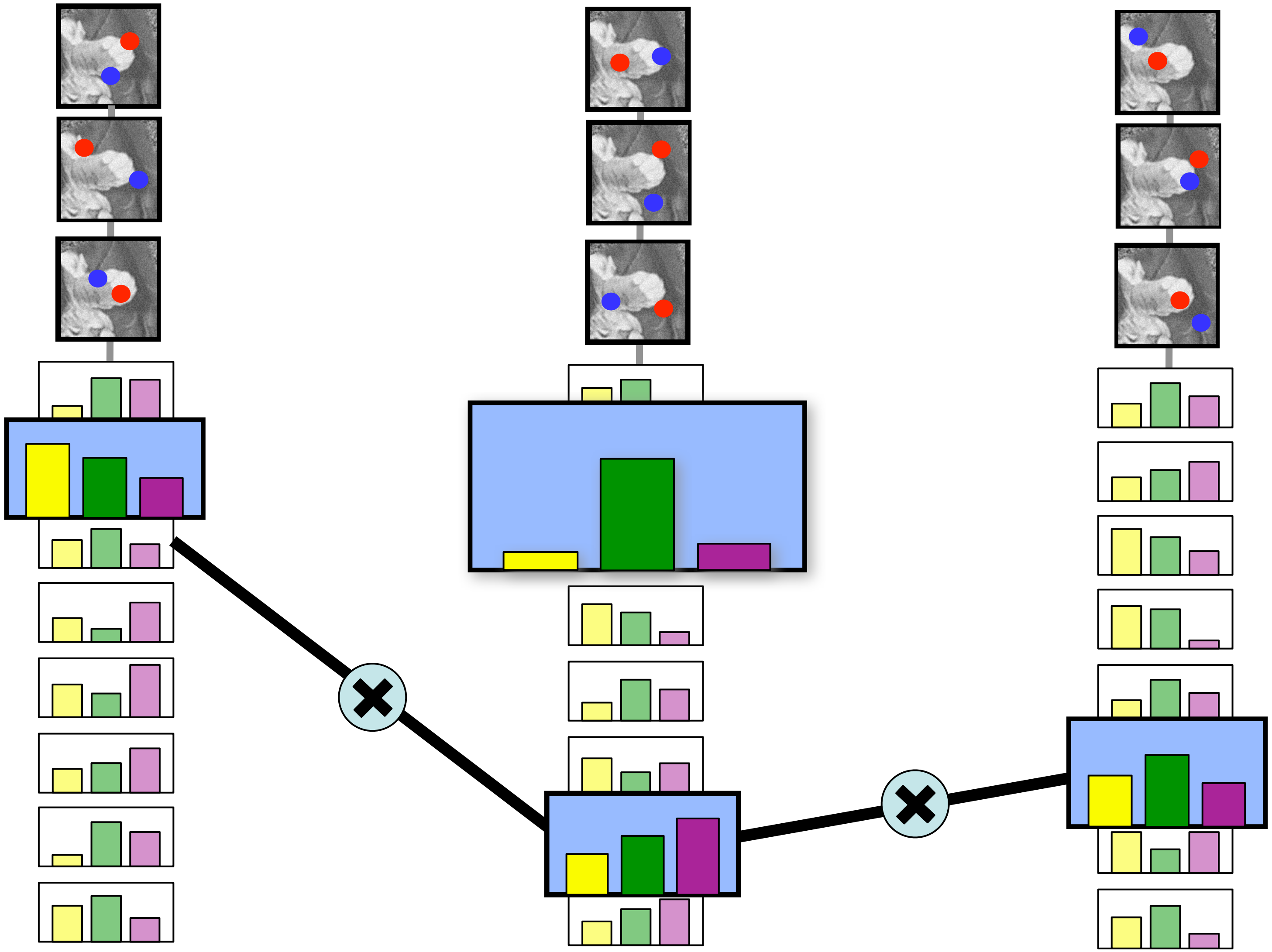
Recognition

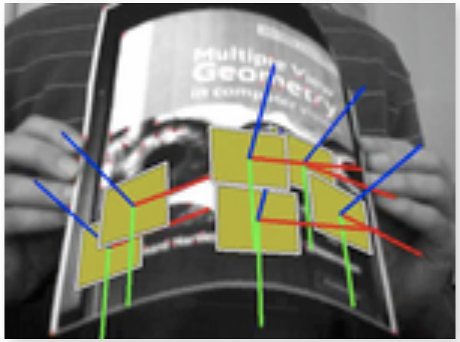


Recognition



Recognition





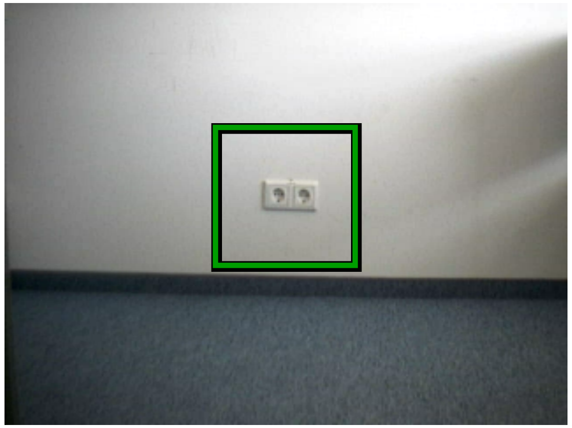
Gepard

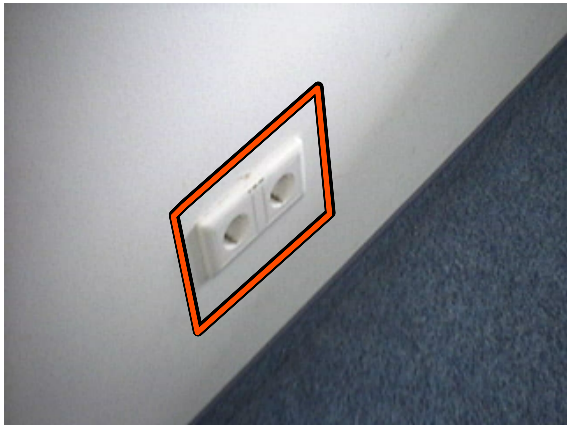
real-time learning of patch rectification

[CVPR'09]

Joint Work with Stefan Hinterstoisser

Gepard:

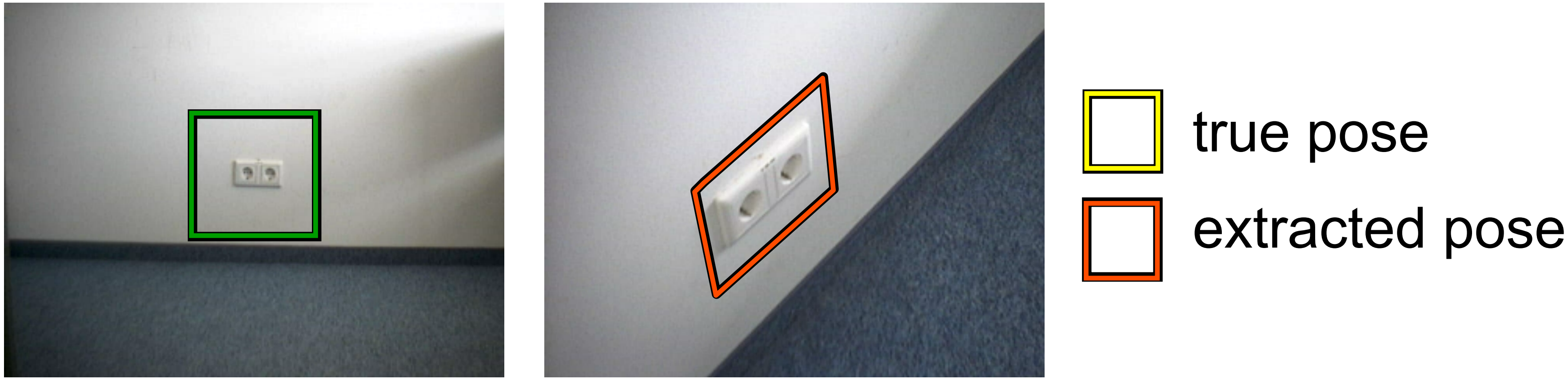




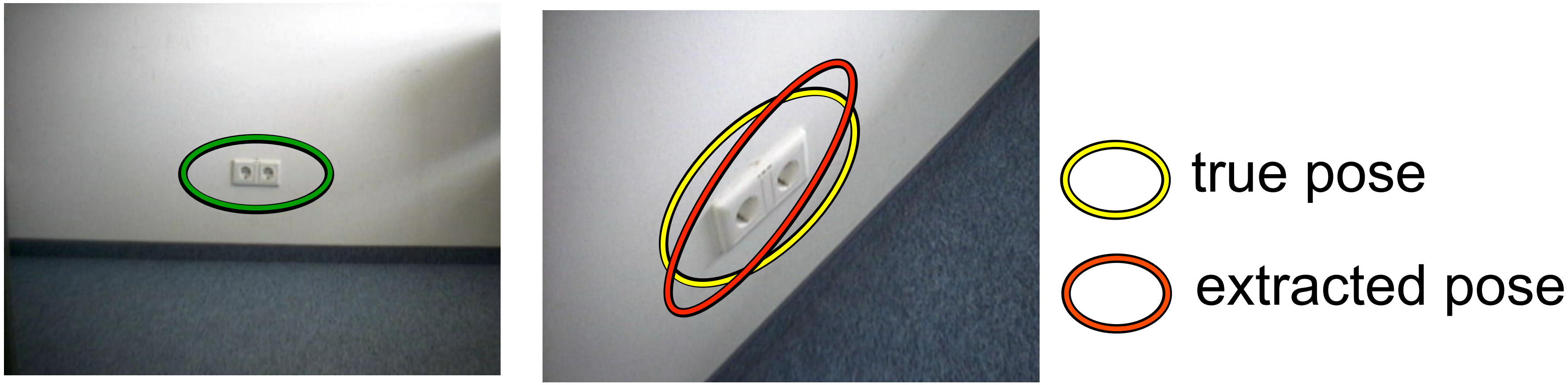
true pose

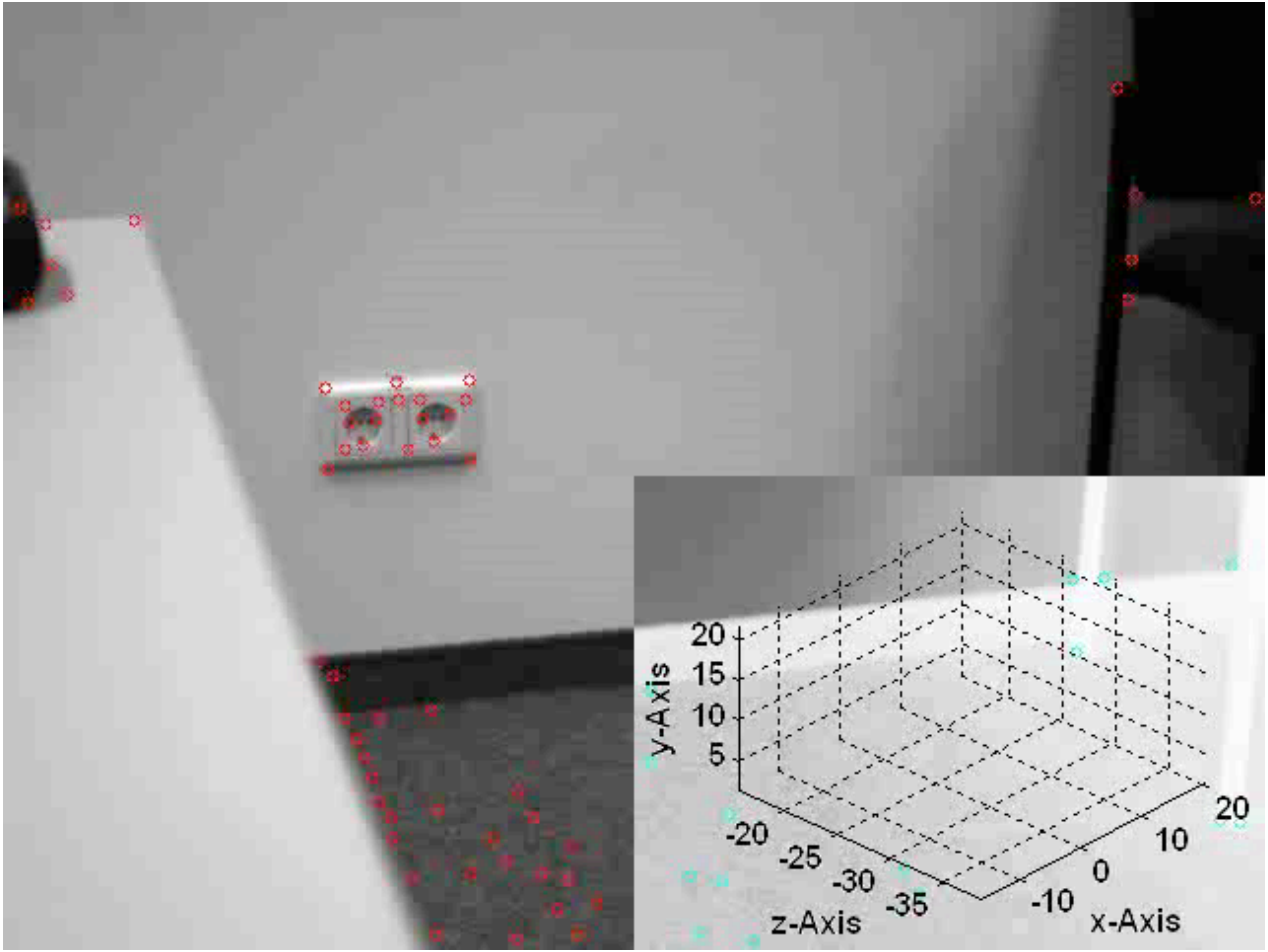
extracted pose

Gepard:



Affine region detectors:



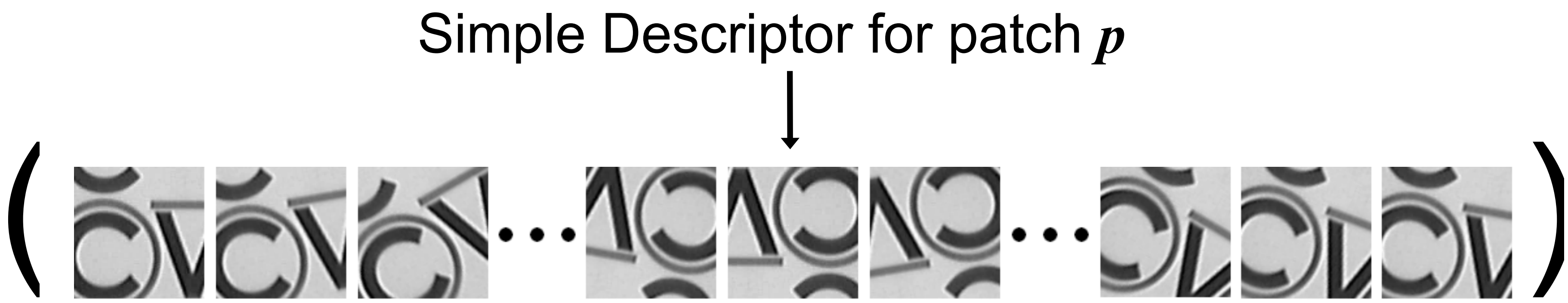

20
15
10
5
y-Axis
-20
-25
-30
-35
z-Axis
20
10
0
-10
x-Axis

Keypoint Recognition & Coarse Pose Estimation

Simple Solution:

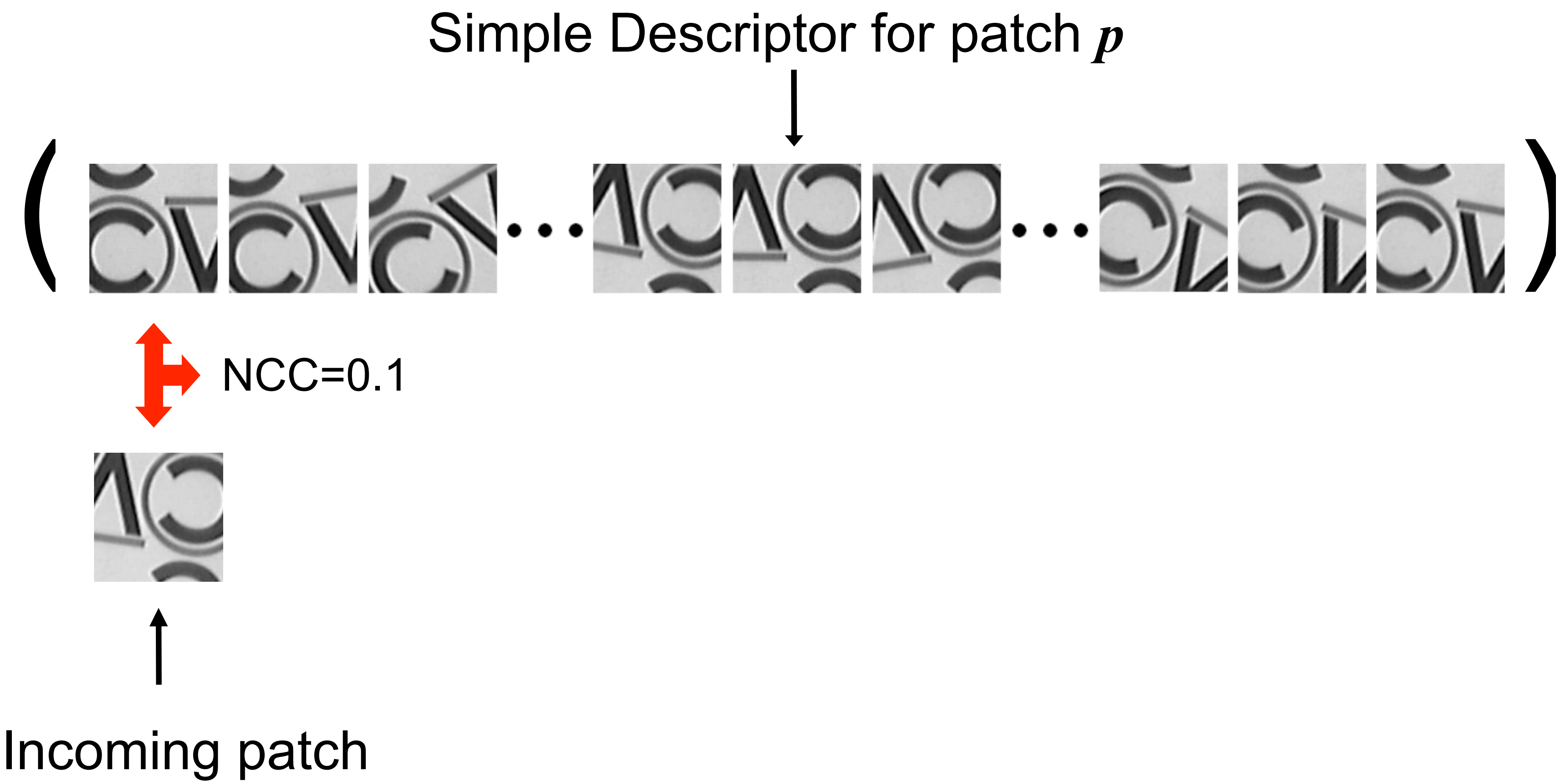
Keypoint Recognition & Coarse Pose Estimation

Simple Solution:



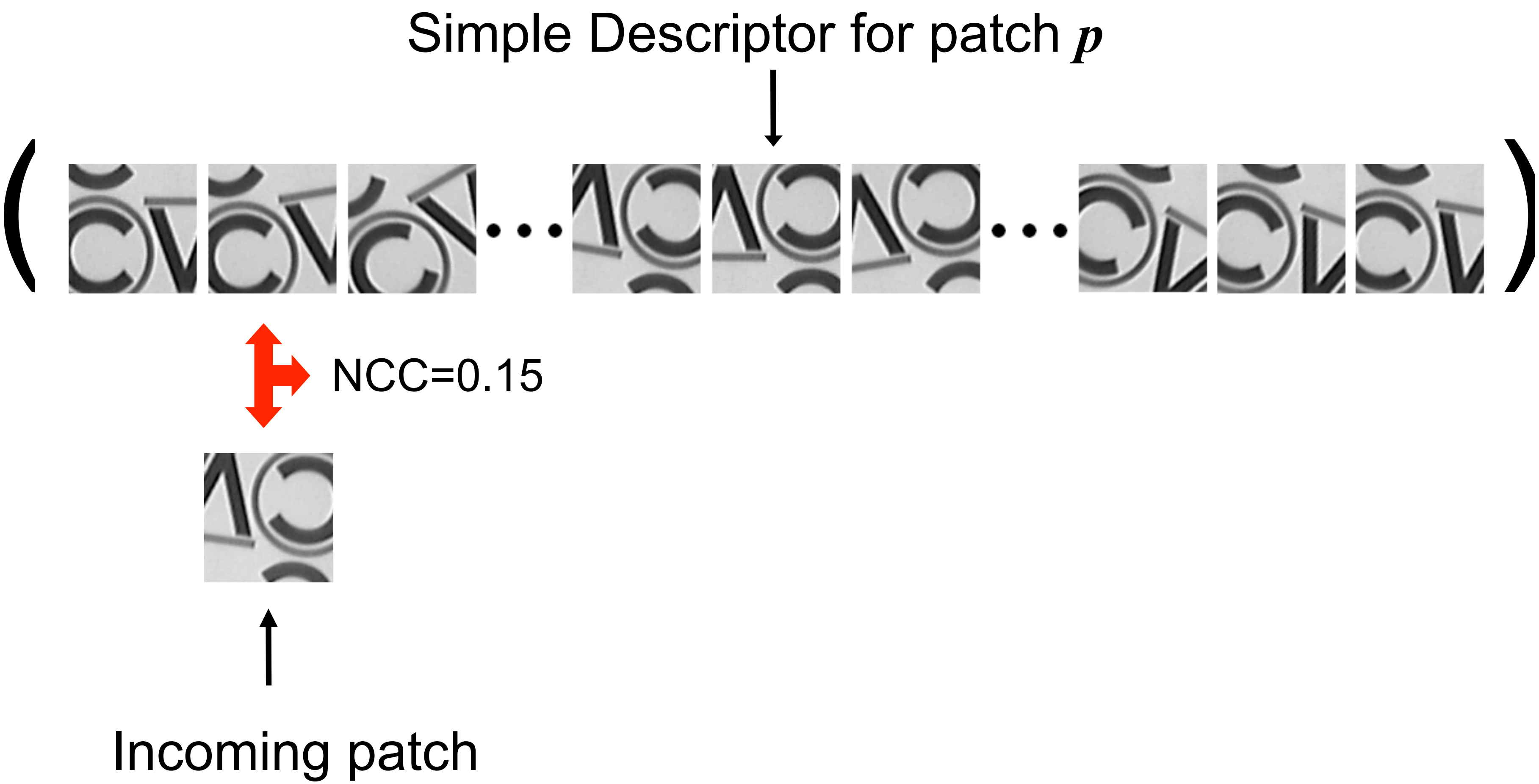
Keypoint Recognition & Coarse Pose Estimation

Simple Solution:



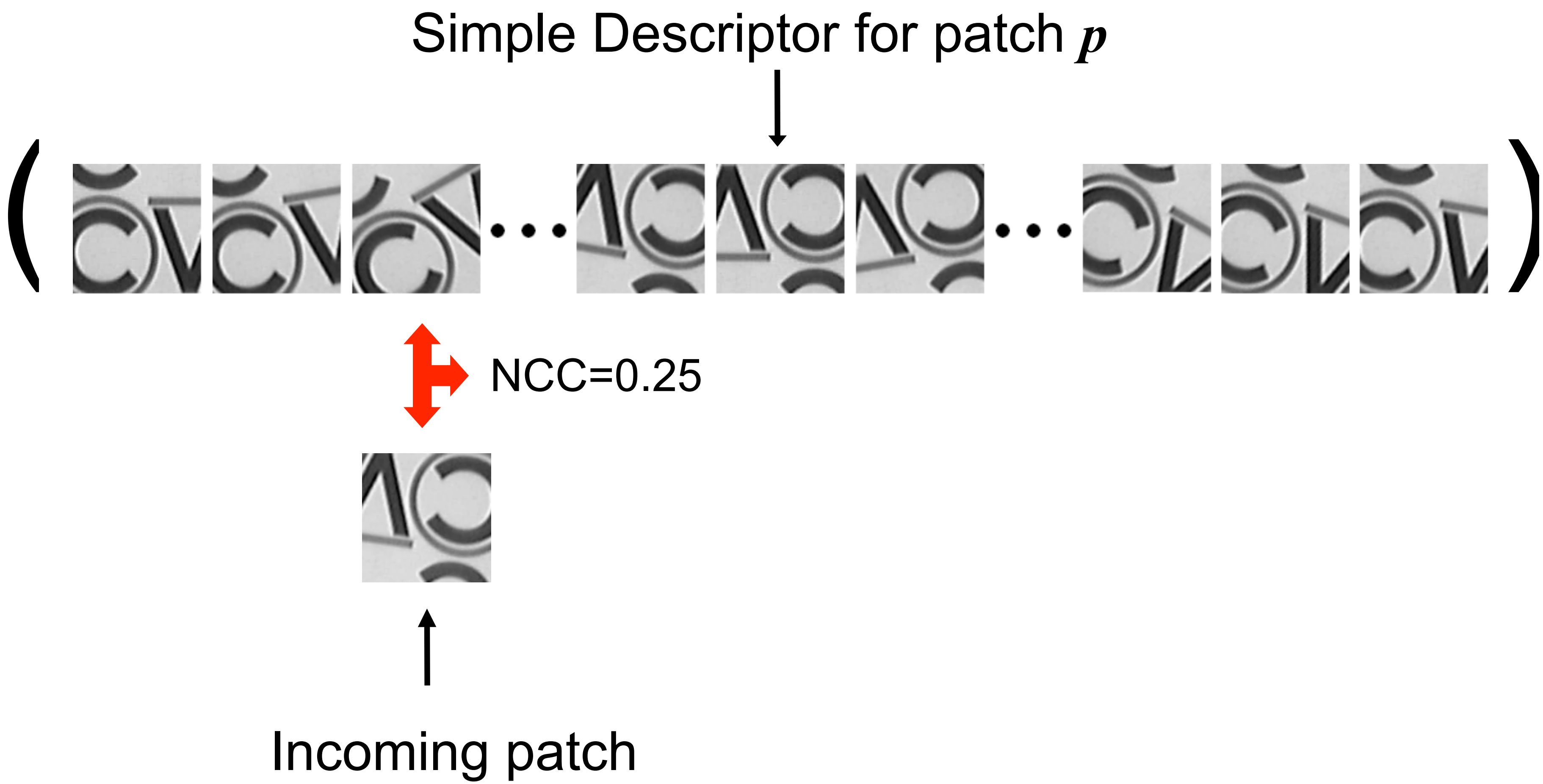
Keypoint Recognition & Coarse Pose Estimation

Simple Solution:



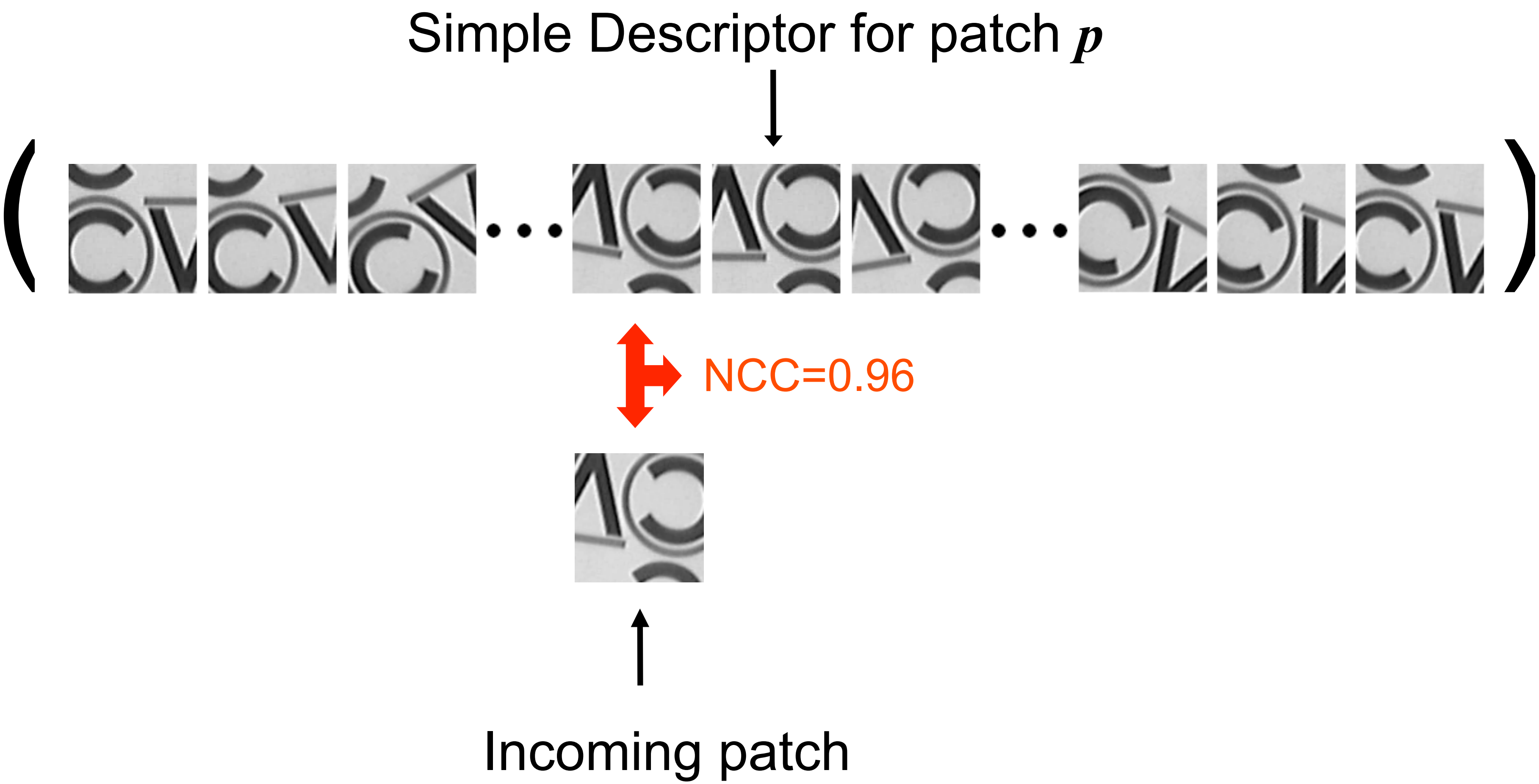
Keypoint Recognition & Coarse Pose Estimation

Simple Solution:



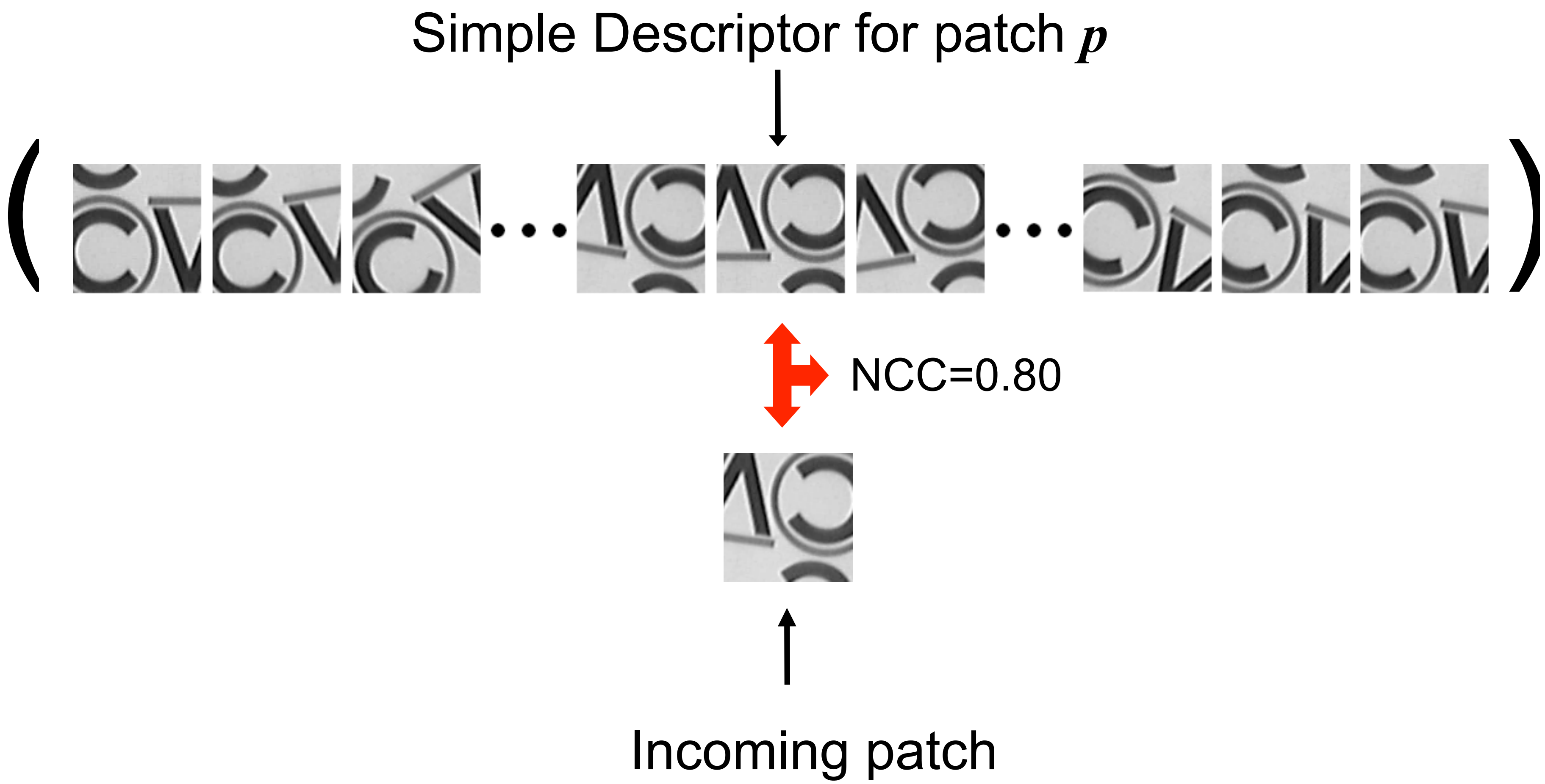
Keypoint Recognition & Coarse Pose Estimation

Simple Solution:



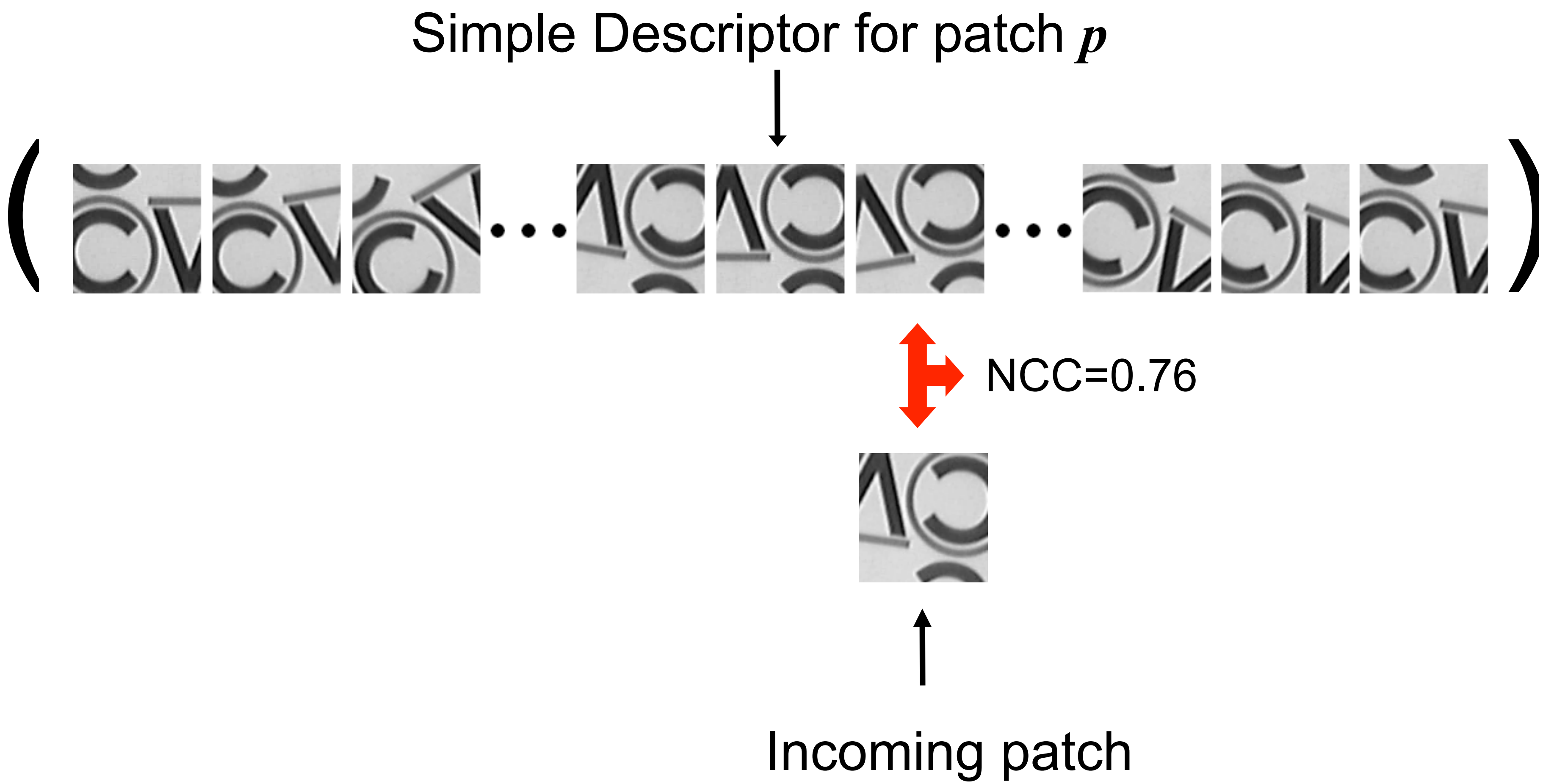
Keypoint Recognition & Coarse Pose Estimation

Simple Solution:



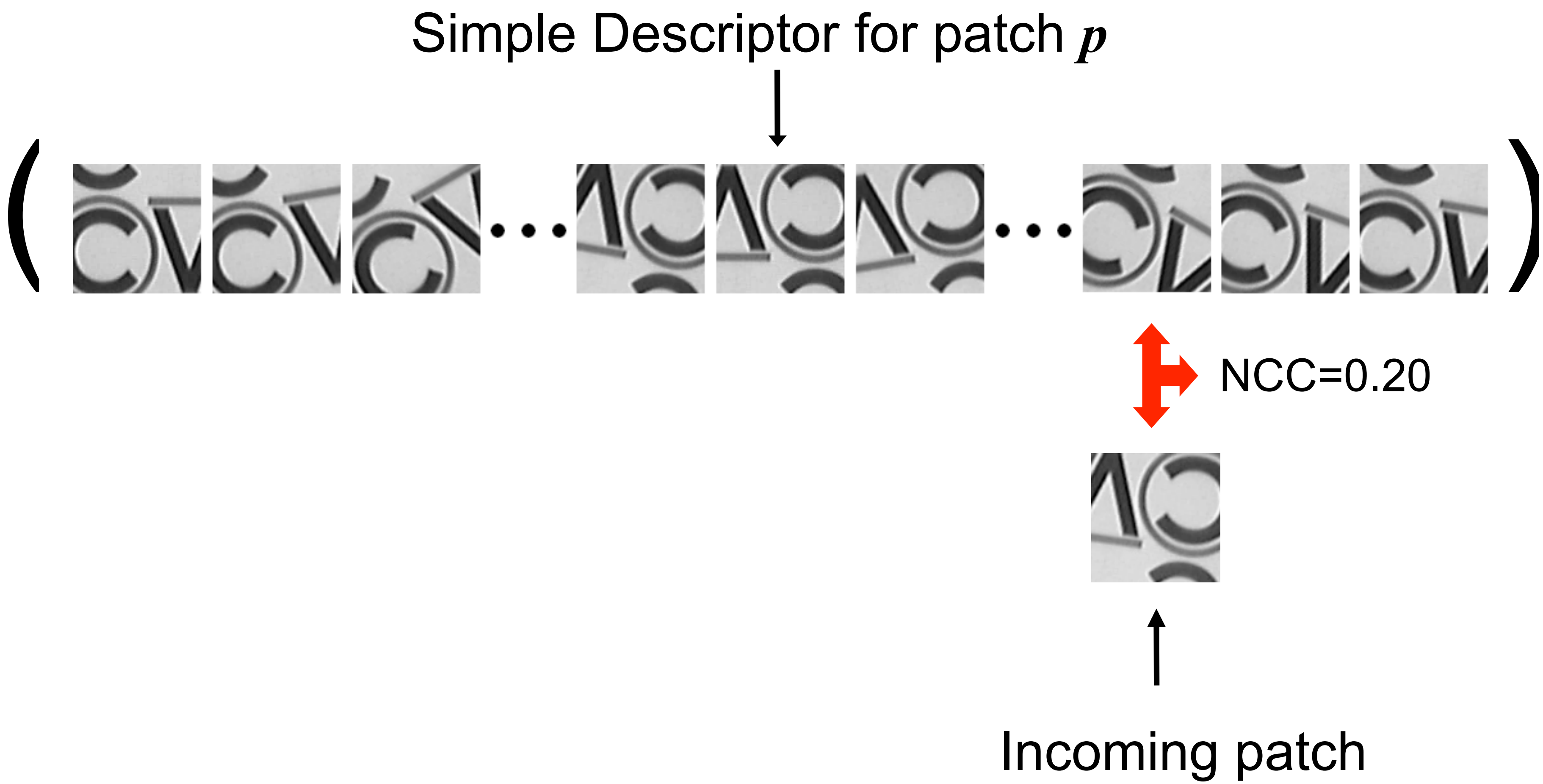
Keypoint Recognition & Coarse Pose Estimation

Simple Solution:



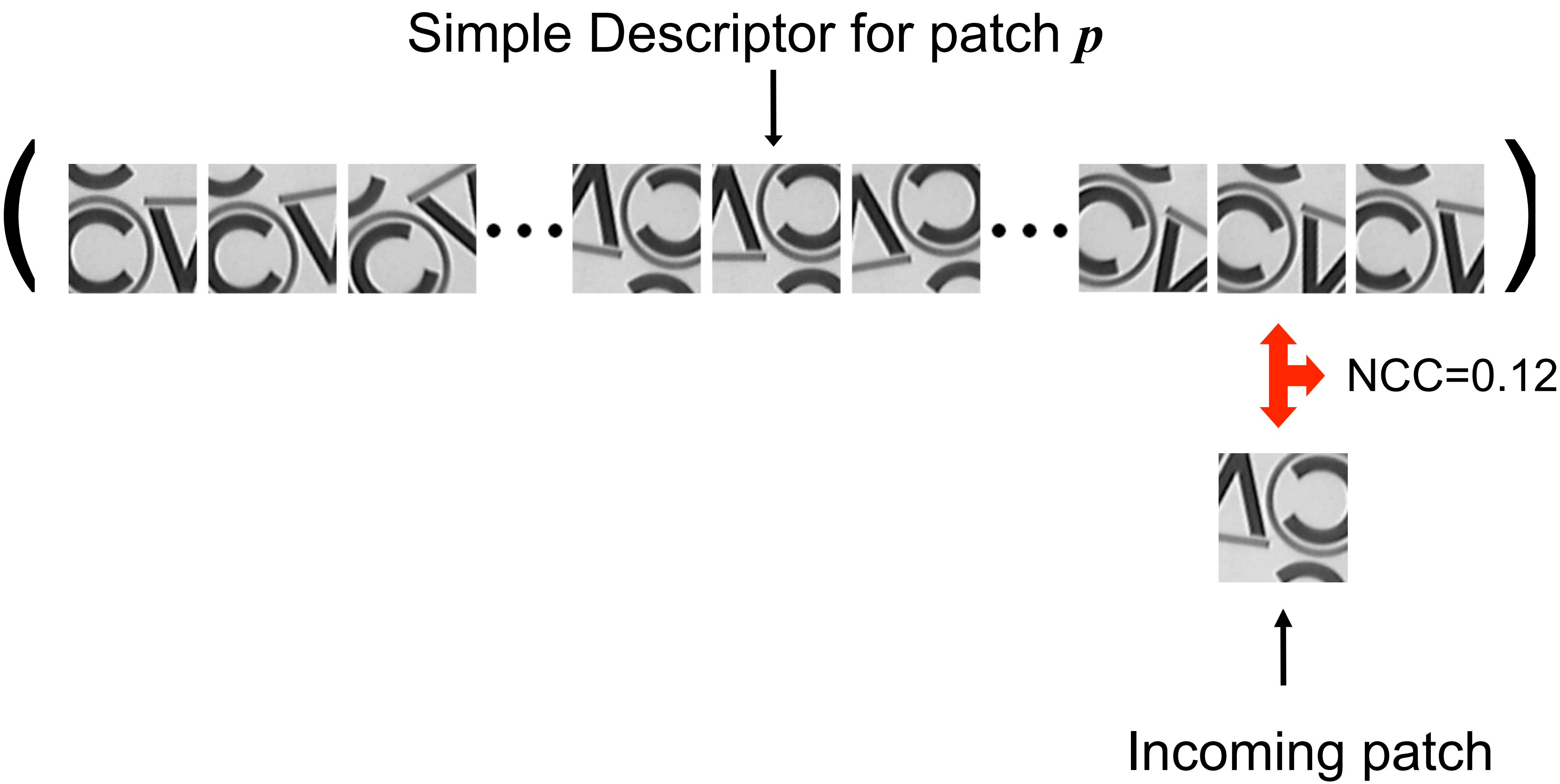
Keypoint Recognition & Coarse Pose Estimation

Simple Solution:



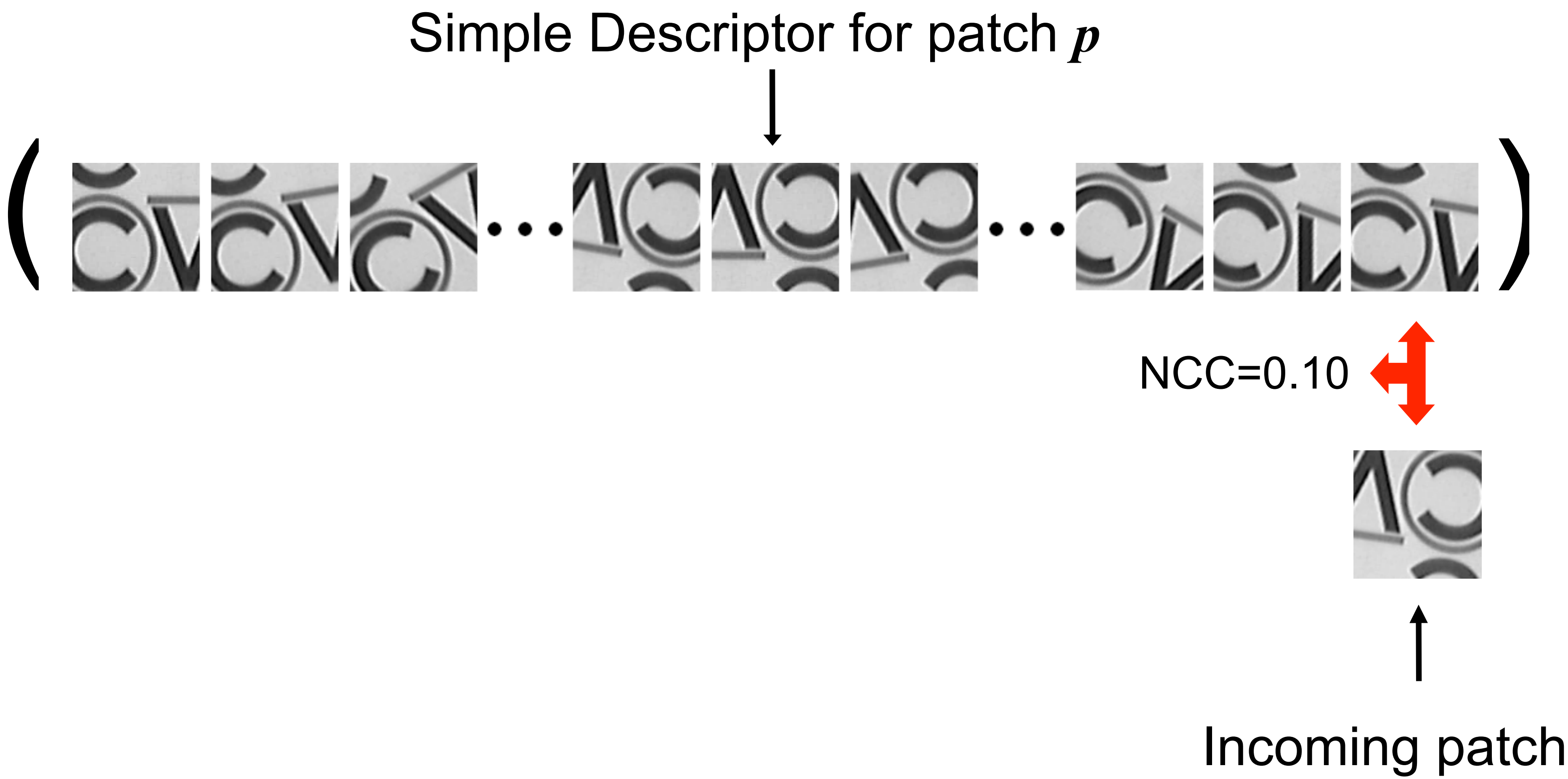
Keypoint Recognition & Coarse Pose Estimation

Simple Solution:



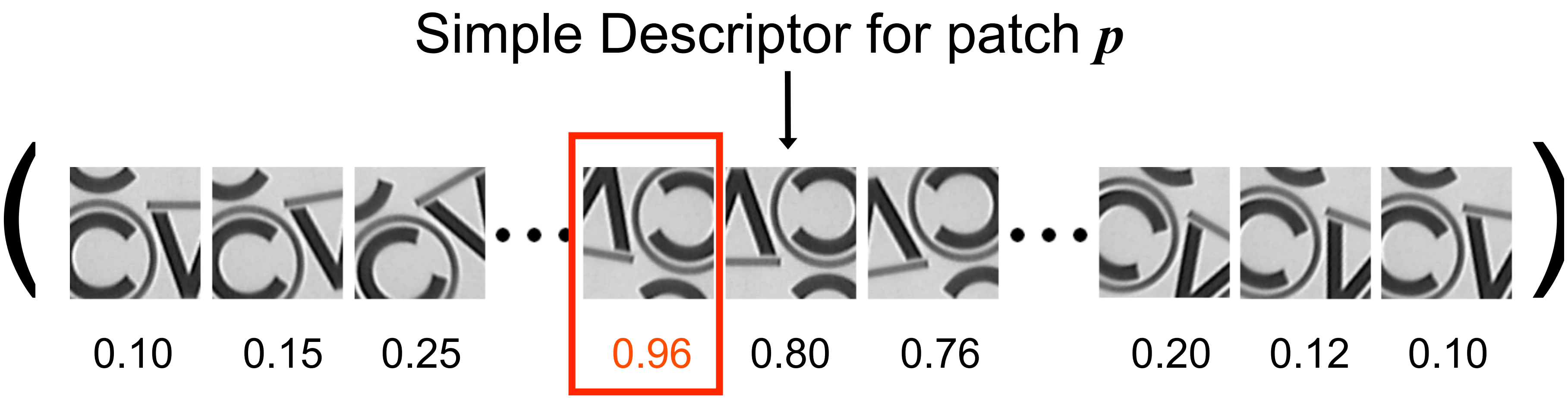
Keypoint Recognition & Coarse Pose Estimation

Simple Solution:



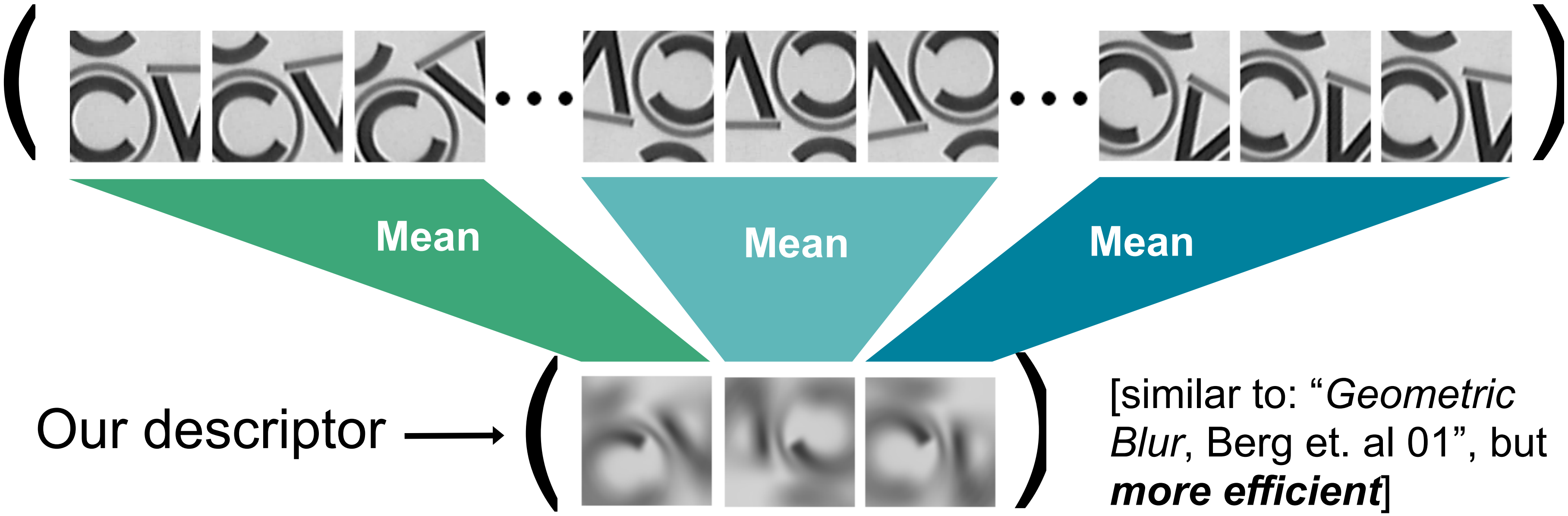
Keypoint Recognition & Coarse Pose Estimation

Simple Solution:



computationally very expensive

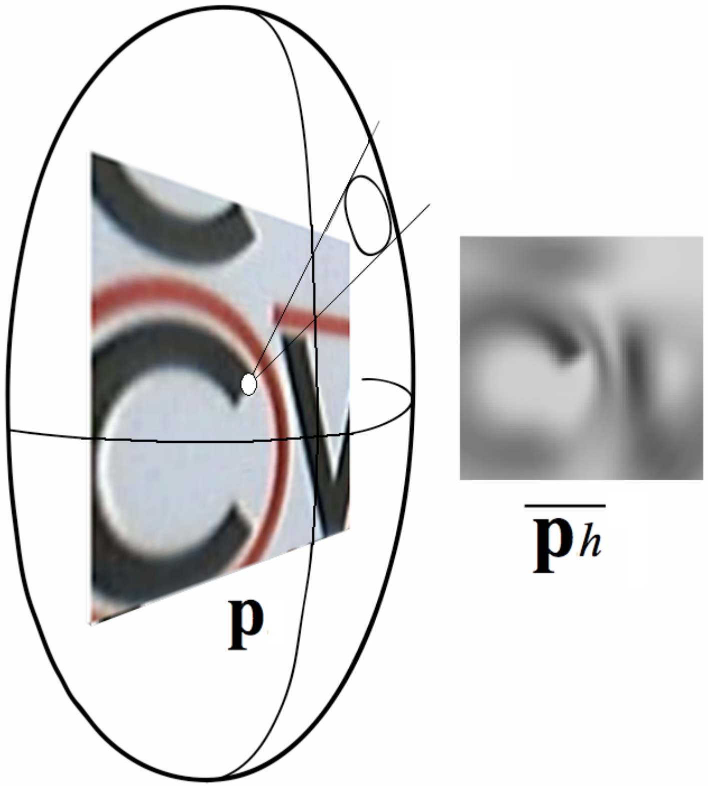
Keypoint Recognition & Coarse Pose Estimation



[similar to: "*Geometric Blur*, Berg et. al 01", but ***more efficient***]

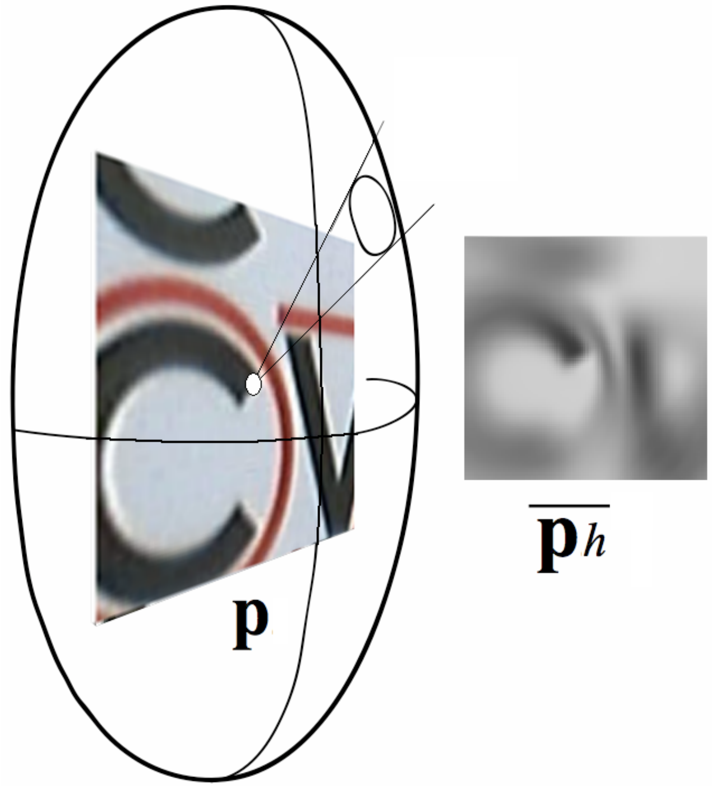
Incoming patch

Fast Computation of Mean Patches



$$\overline{\mathbf{p}_h} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{w}(\mathbf{p}, H_{h,j} \quad)$$

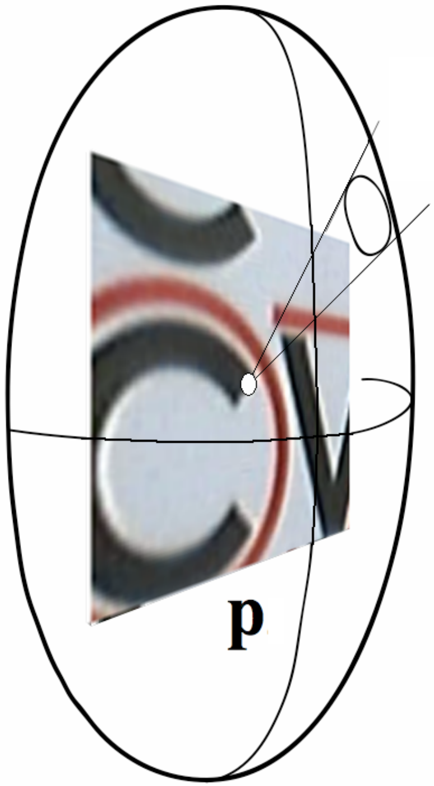
Fast Computation of Mean Patches

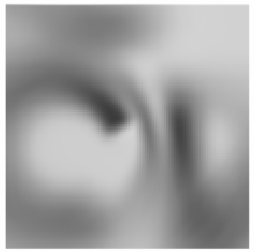


original patch

$$\overline{\mathbf{p}_h} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{w}(\mathbf{p}, H_{h,j})$$

Fast Computation of Mean Patches



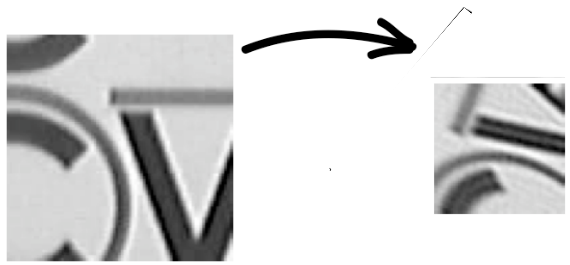


$\overline{\mathbf{p}_h}$

original patch

$$\overline{\mathbf{p}_h} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{w}(\mathbf{p}, H_{h,j} \ )$$

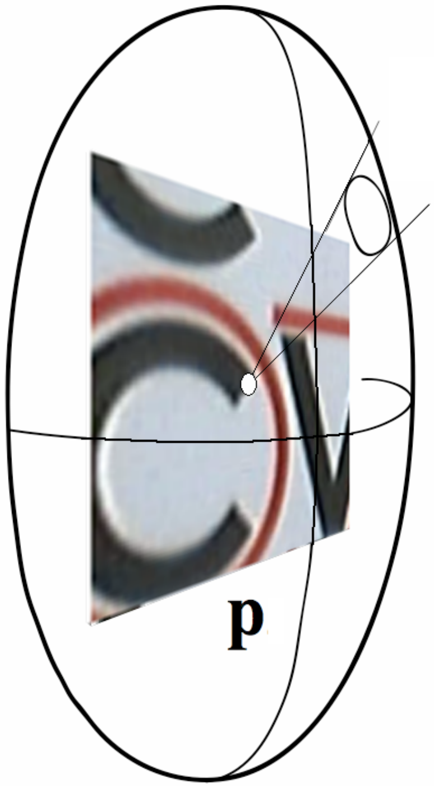
warping function:



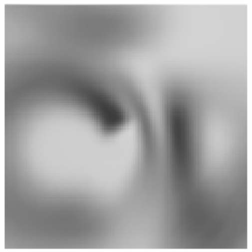
$\mathbf{p}$

$\mathbf{w}(\mathbf{p}, H)$

Fast Computation of Mean Patches



$\mathbf{p}$



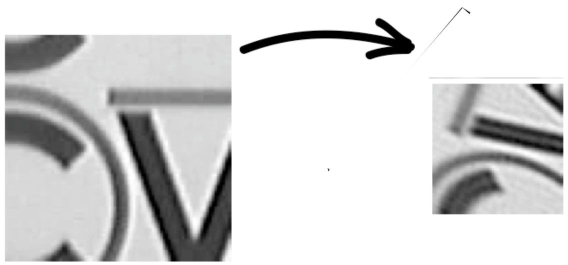
$\overline{\mathbf{p}_h}$

original patch

$$\overline{\mathbf{p}_h} = \frac{1}{N}\sum_{j=1}^{N} \mathbf{w}(\mathbf{p}, H_{h,j})$$

pose

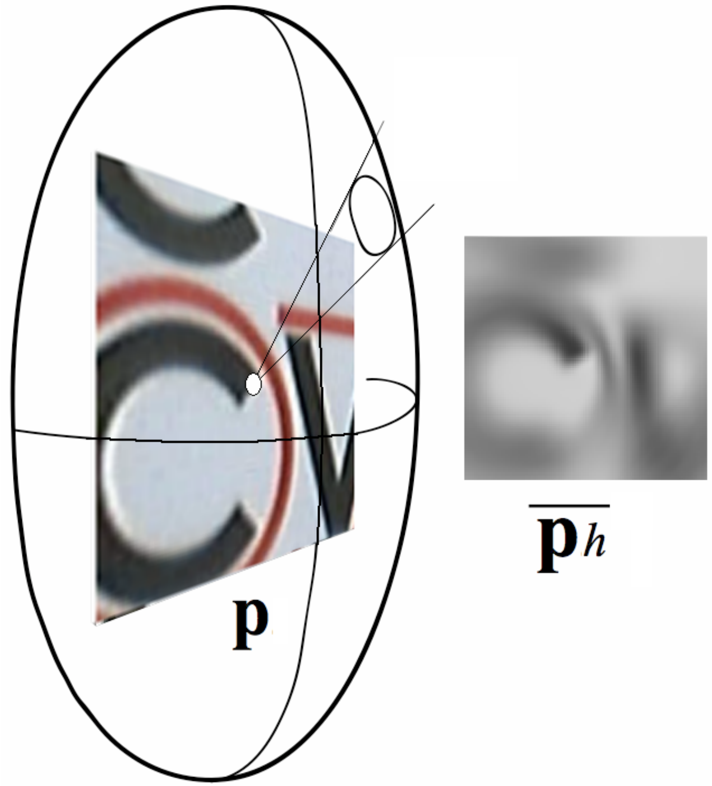
warping function:



$\mathbf{p}$

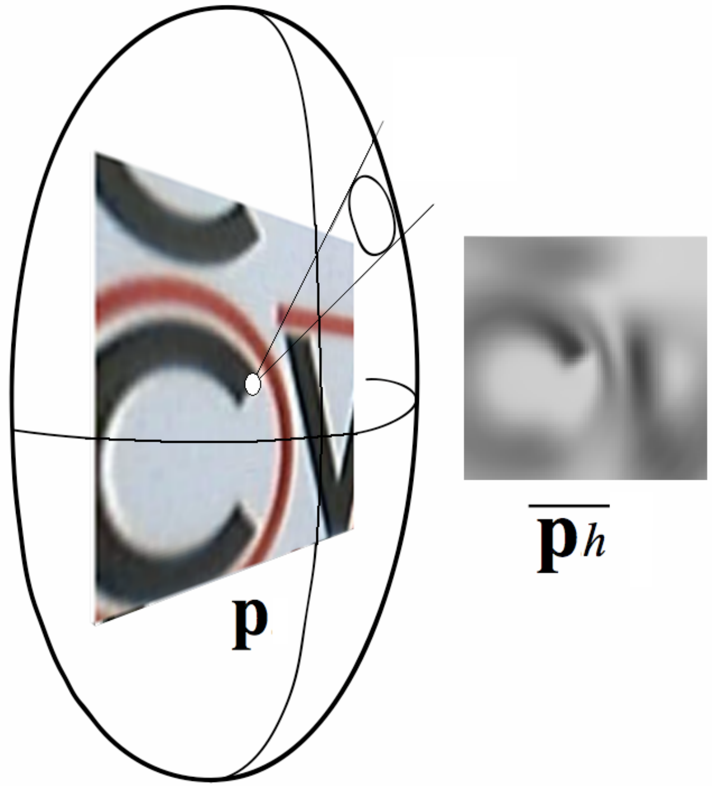
$\mathbf{w}(\mathbf{p}, H)$

Fast Computation of Mean Patches



$$\overline{\mathbf{p}_h} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{w}(\mathbf{p}, H_{h,j} \quad)$$

Fast Computation of Mean Patches

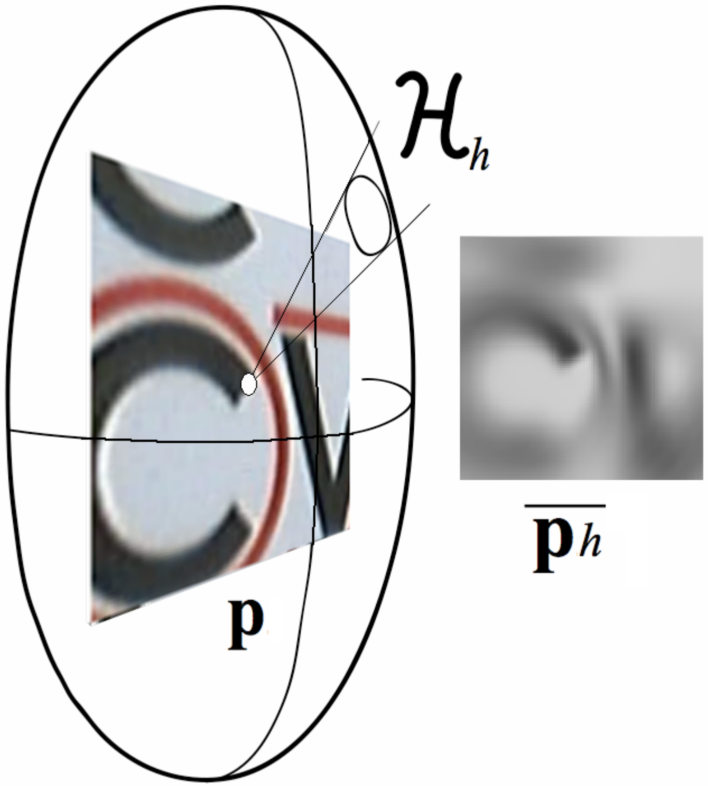


$$\overline{\mathbf{p}_h} = \frac{1}{N}\sum_{j=1}^{N} \mathbf{w}(\mathbf{p}, H_{h,j})$$

$$= \frac{1}{N}\sum_{j} \mathbf{w}(\underbrace{\sum_{l=1}^{L} \alpha_l \mathbf{v}_l}, H_{j,h})$$

PCA decomposition of the original patch

Fast Computation of Mean Patches



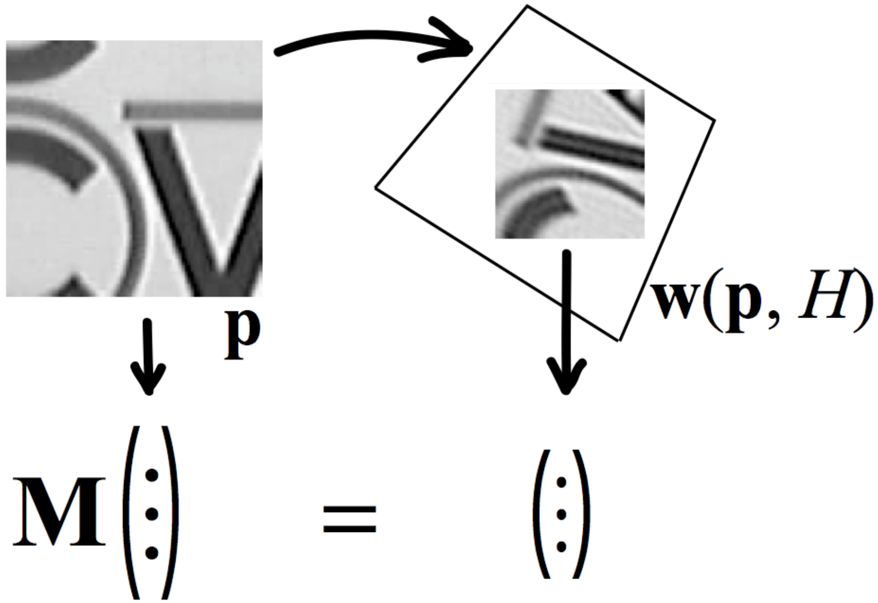
$$\overline{\mathbf{p}_h} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{w}(\mathbf{p}, H_{h,j})$$

$$\overline{\mathbf{p}_h} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{w}(\underbrace{\sum_{l=1}^{L} \alpha_l \mathbf{v}_l}, H_{j,h})$$

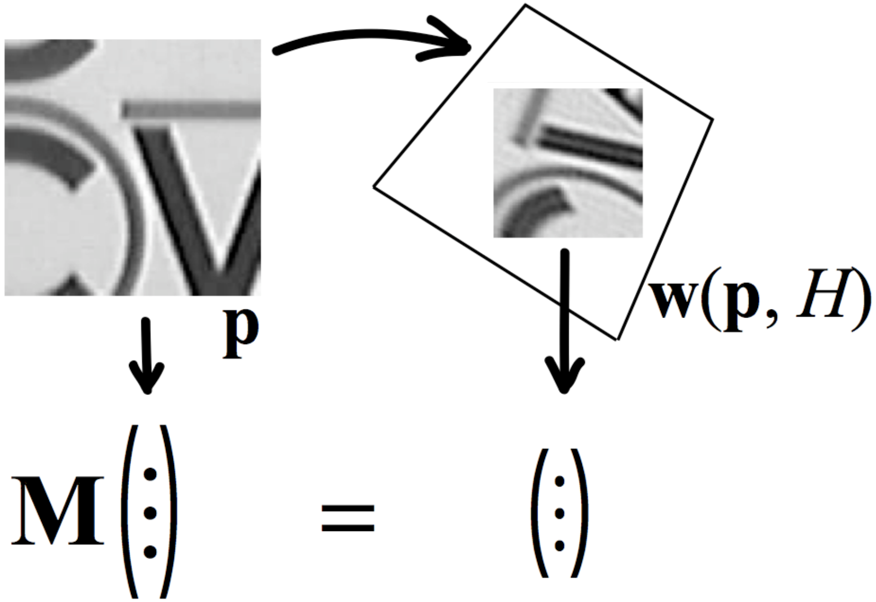
PCA decomposition of the original patch

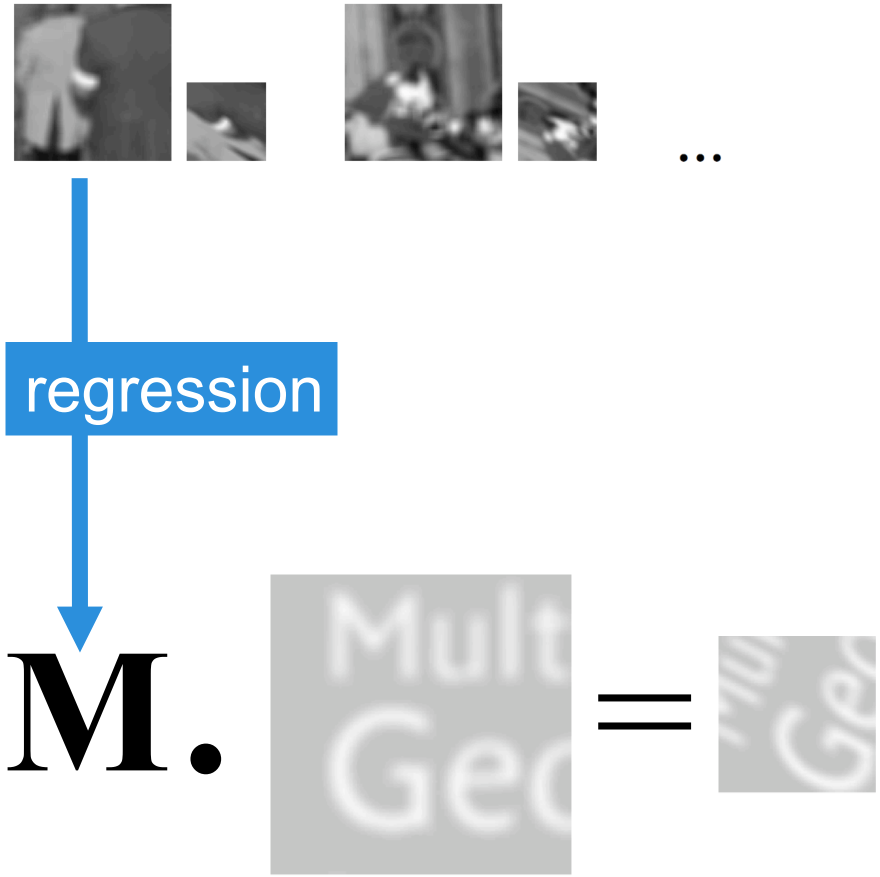
$$\overline{\mathbf{p}_h} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{w}(\sum_{l=1}^{L} \alpha_l \mathbf{v}_l, H_{j,h})$$

$\mathbf{w}(\ .\ ,\ H)$ is a linear function



$\mathbf{w}(\,.\, , H)$ is a linear function





$$\overline{\mathbf{p}_h} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{w}(\sum_{l=1}^{L}\alpha_l\mathbf{v}_l, H_{j,h})$$

$$\overline{\mathbf{p}_h} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{w}\left(\sum_{l=1}^{L}\alpha_l\mathbf{v}_l, H_{j,h}\right)$$

$$= \frac{1}{N}\sum_{j=1}^{N}\left(\sum_{l=1}^{L}\alpha_l\mathbf{w}(\mathbf{v}_l, H_{j,h})\right)$$

$$= \sum_{l=1}^{L}\frac{\alpha_l}{N}\sum_{j=1}^{N}\mathbf{w}(\mathbf{v}_l, H_{j,h})$$

$$= \sum_{l=1}^{L}\alpha_l\overline{\mathbf{v}_{l,h}}$$

computation time does not depend on the number of samples

precomputed:

$$\overline{\mathbf{v}_{l,h}} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{w}(\mathbf{v}_l, H_{j,h})$$

Matrix Form

$$(0) \qquad \boldsymbol{\alpha} = \mathbf{P}_{pca}\mathbf{p}$$

$$(1) \quad \overline{\mathbf{p}_{h=1}} = \mathbf{V}_{h=1}\boldsymbol{\alpha}$$

$$(2) \quad \overline{\mathbf{p}_{h=2}} = \mathbf{V}_{h=2}\boldsymbol{\alpha}$$

$$(i) \quad \overline{\mathbf{p}_{h=i}} = \mathbf{V}_{h=i}\boldsymbol{\alpha}$$

$$dim(\mathbf{p}) >> dim(\boldsymbol{\alpha})$$

Matrix Form

$$(0) \quad \boldsymbol{\alpha} = \mathbf{P}_{pca}\mathbf{p}$$

$$(1) \quad \overline{\mathbf{p}_{h=1}} = \mathbf{V}_{h=1}\boldsymbol{\alpha}$$

$$(2) \quad \overline{\mathbf{p}_{h=2}} = \mathbf{V}_{h=2}\boldsymbol{\alpha}$$

$$(i) \quad \overline{\mathbf{p}_{h=i}} = \mathbf{V}_{h=i}\boldsymbol{\alpha}$$

$$dim(\mathbf{p}) >> dim(\boldsymbol{\alpha})$$

Matrix Form

$$(0) \qquad \boldsymbol{\alpha} = \mathbf{P}_{pca}\mathbf{p}$$

$$(1) \quad \overline{\mathbf{p}_{h=1}} = \mathbf{V}_{h=1}\boldsymbol{\alpha}$$

$$(2) \quad \overline{\mathbf{p}_{h=2}} = \mathbf{V}_{h=2}\boldsymbol{\alpha}$$

$$(\mathrm{i}) \quad \overline{\mathbf{p}_{h=i}} = \mathbf{V}_{h=i}\boldsymbol{\alpha}$$

$$dim(\mathbf{p}) >> dim(\boldsymbol{\alpha})$$

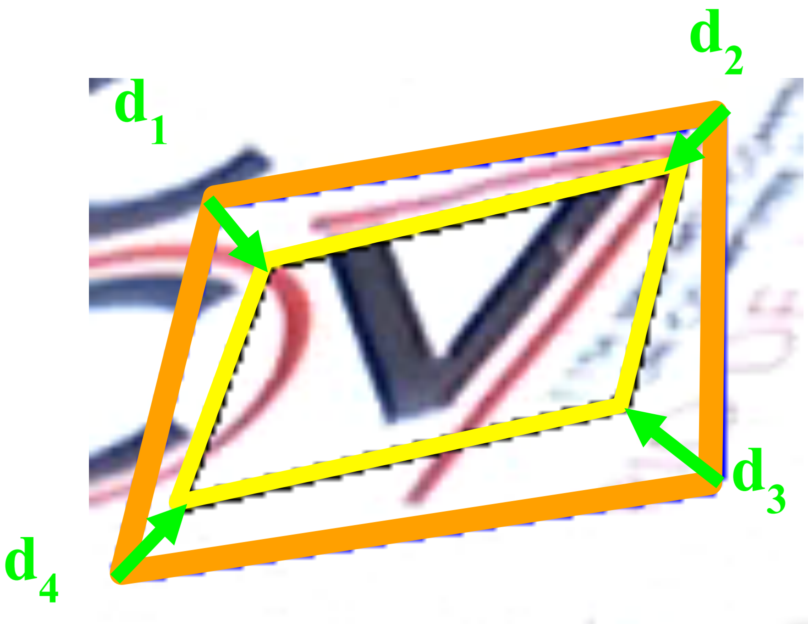
→ Online computation time only depends on the number of principal components

→ Matrix/vector products can be efficiently implemented on the CPU/GPU

Naïve	~ 1 s
PCA CPU	15 ms
PCA GPU	5 ms

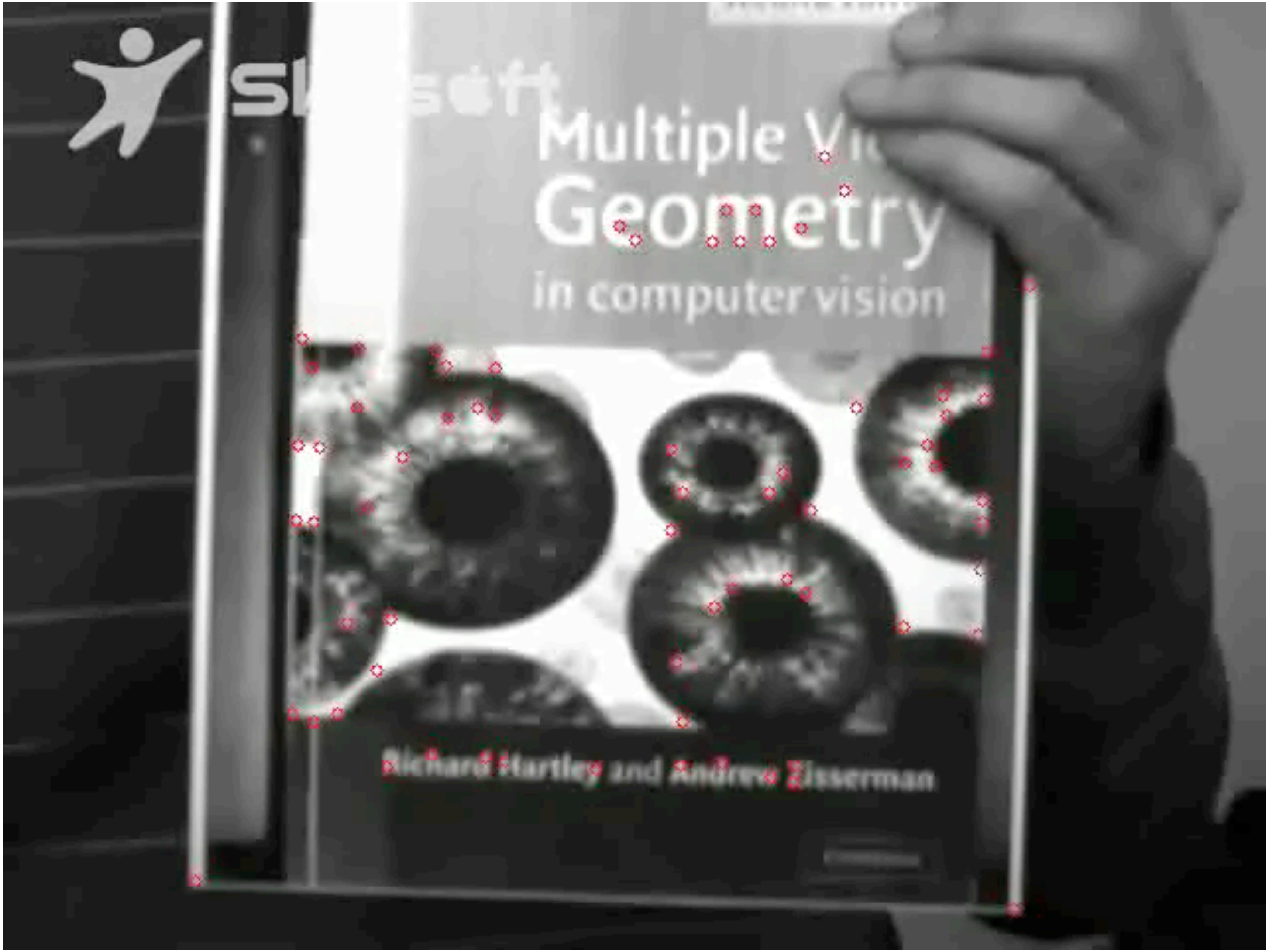
Speed improvement about the factor 200!

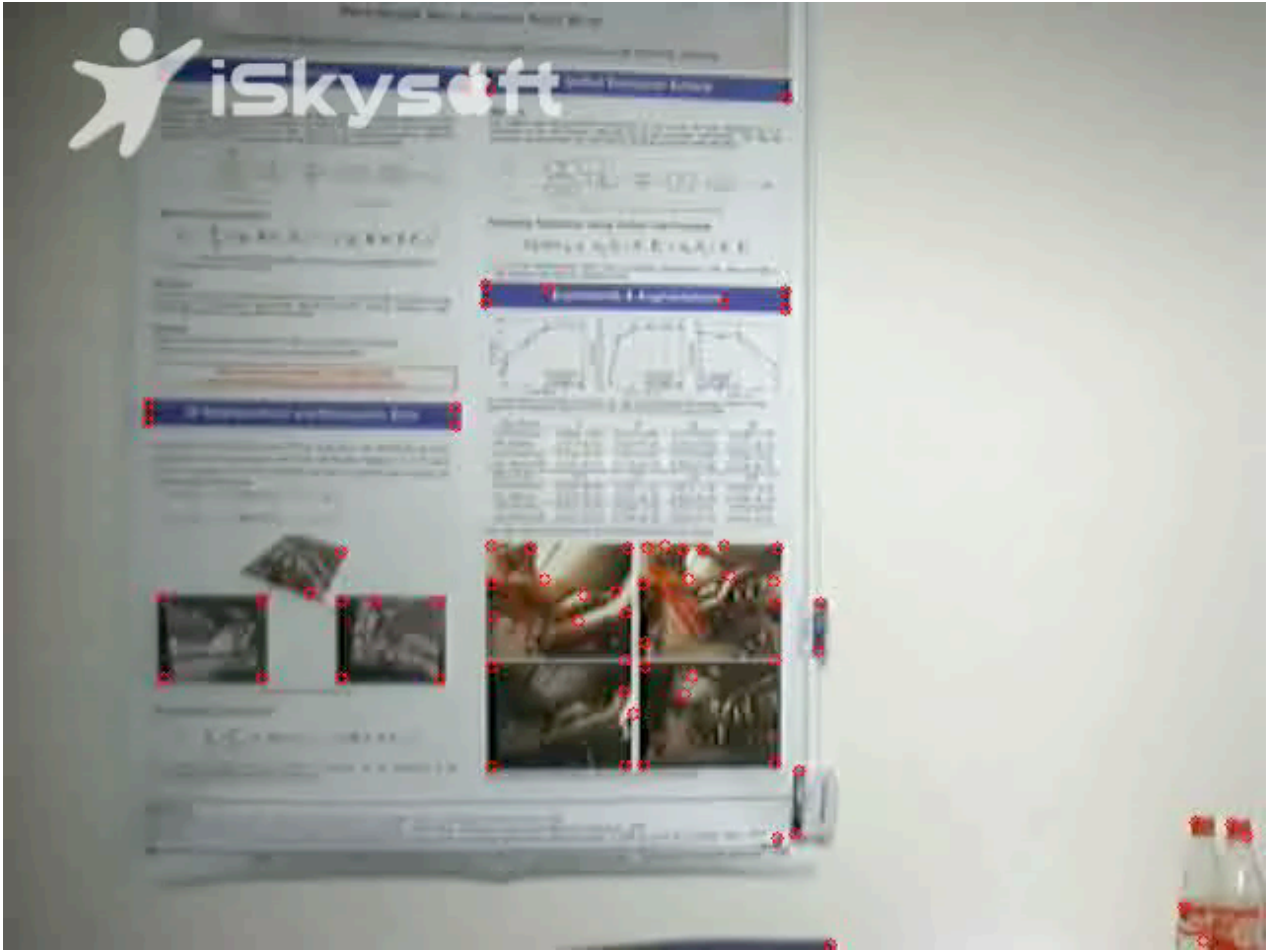
Second Step: Pose Refinement



$$\begin{pmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \mathbf{d}_3 \\ \mathbf{d}_4 \end{pmatrix} = \mathbf{A}_i(\mathbf{p} - \mathbf{p}_i)$$

Matrices $\mathbf{A}_i$ learned by regression [Jurie & Dhome 01]


Multiple
Geometry
in computer vision
Richard Hartley and Andrew Zisserman



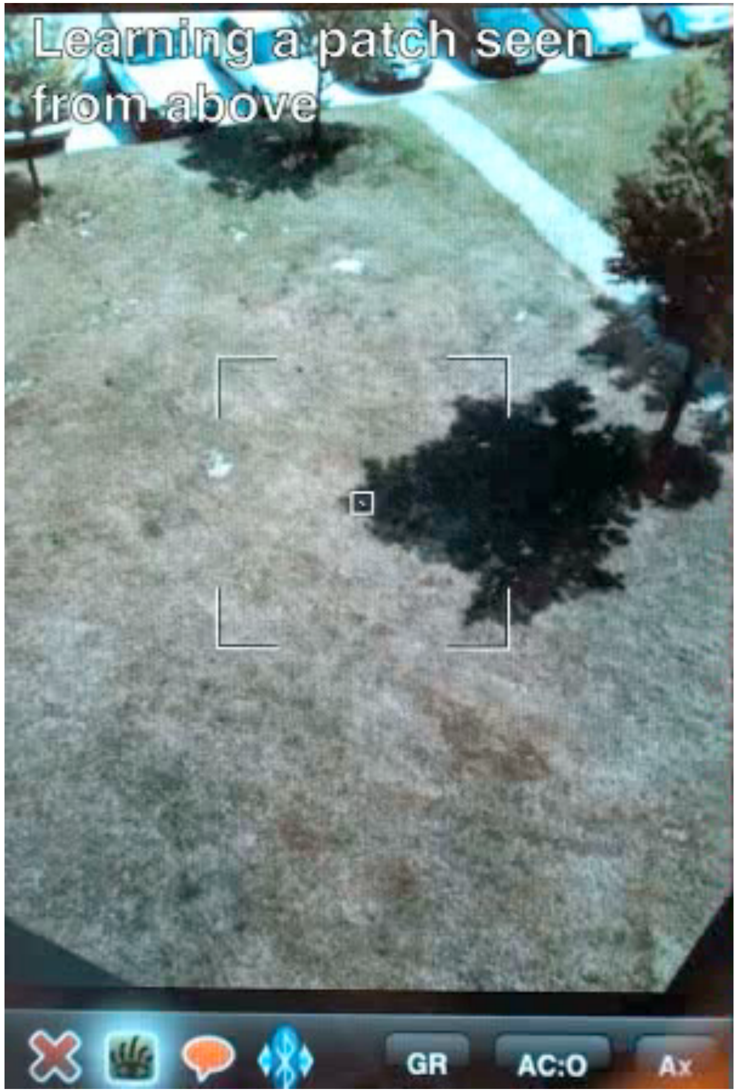
Application to point-and-shoot Augmented Reality on iPhones (with GIST - South Korea)

No need for a 3D model nor a 3D reconstruction.

Guesses the surface orientation (horizontal or vertical) based on the iPhone's accelerometers.

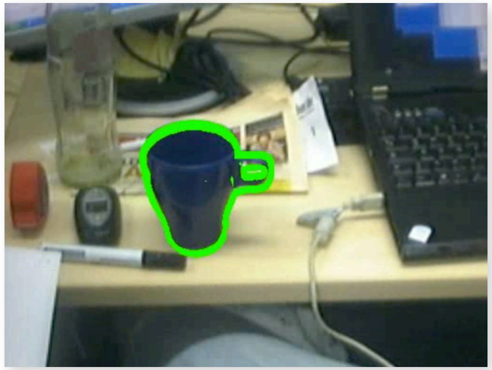
No need for a 3D model nor a 3D reconstruction.

Guesses the surface orientation (horizontal or vertical) based on the iPhone's accelerometers.



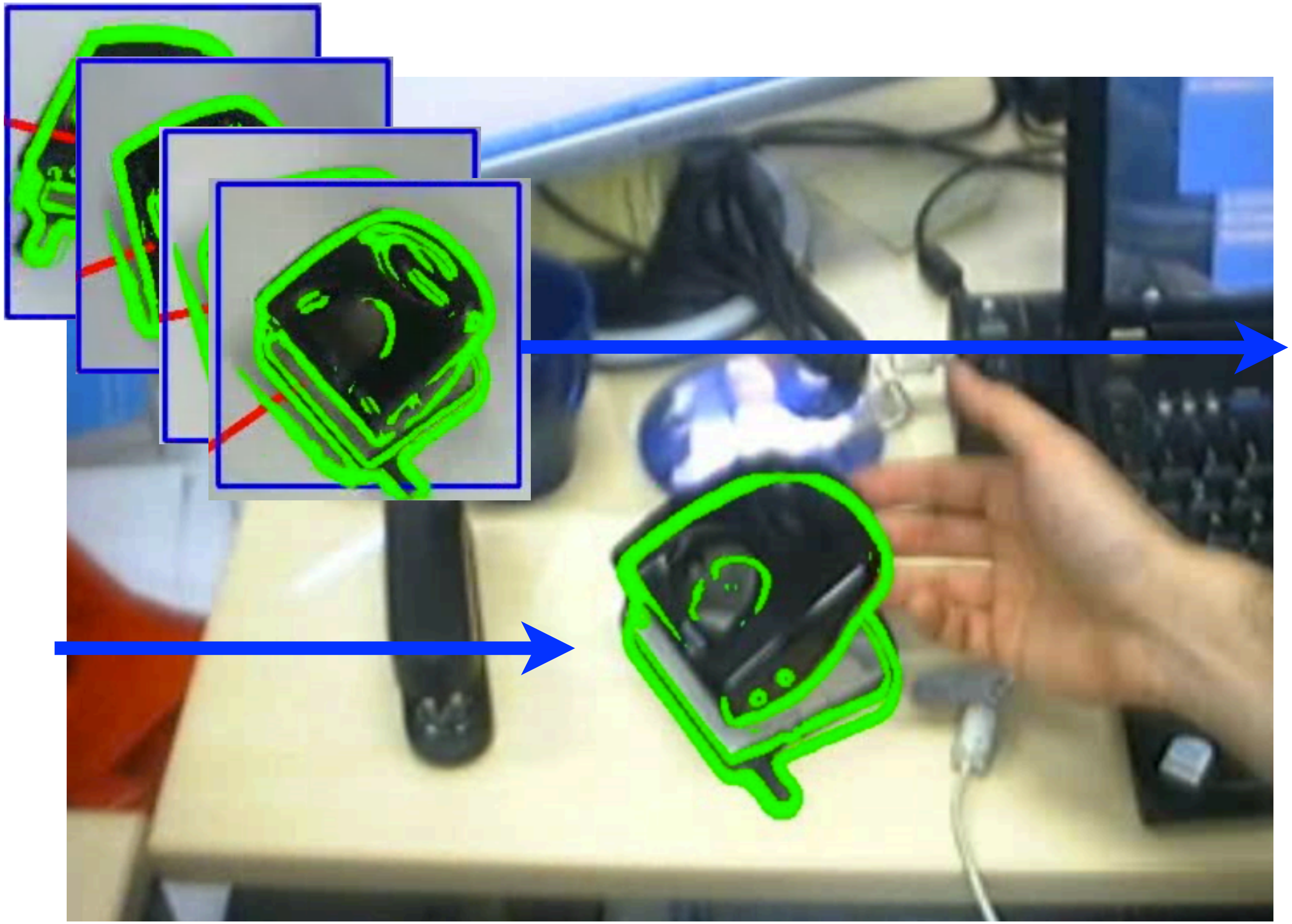
No need for a 3D model nor a 3D reconstruction.

Guesses the surface orientation (horizontal or vertical) based on the iPhone's accelerometers.

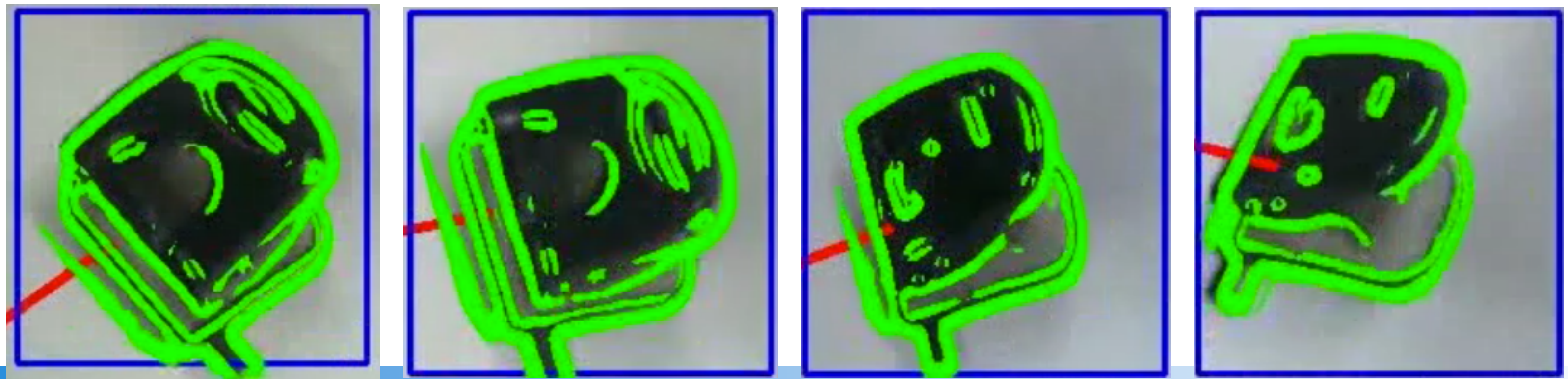


DOT [CVPR'10]
dense descriptor for object detection

Joint work with Stefan Hinterstoisser

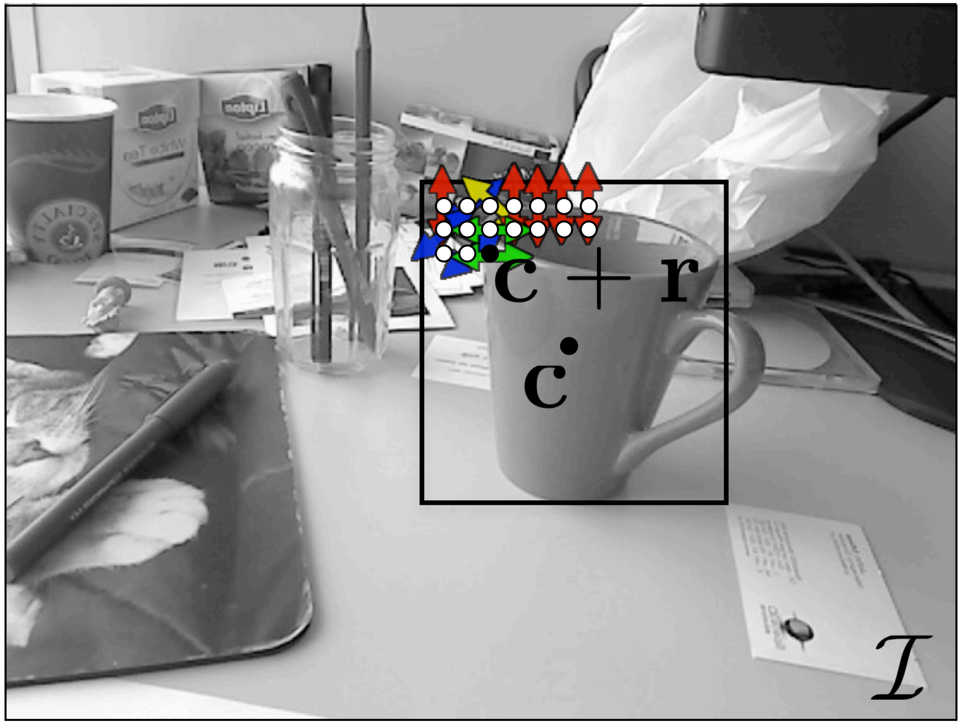


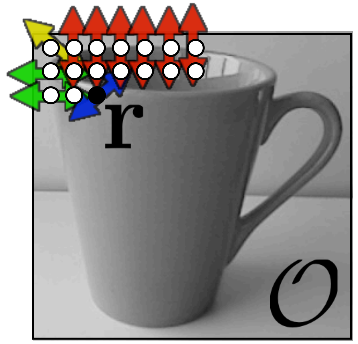
Template matching with an efficient representation of the images and the templates.



Initial Similarity Measure

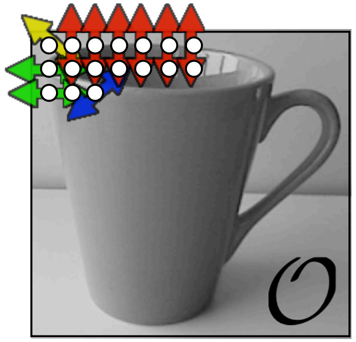
Initial Similarity Measure





$$\mathcal{E}_1(\mathcal{I},\mathcal{O},\mathbf{c}) = \sum_{\mathbf{r}} \mathbb{1}[\text{orientation}(\mathcal{I},\mathbf{c}+\mathbf{r}) = \text{orientation}(\mathcal{O},\mathbf{r})]$$

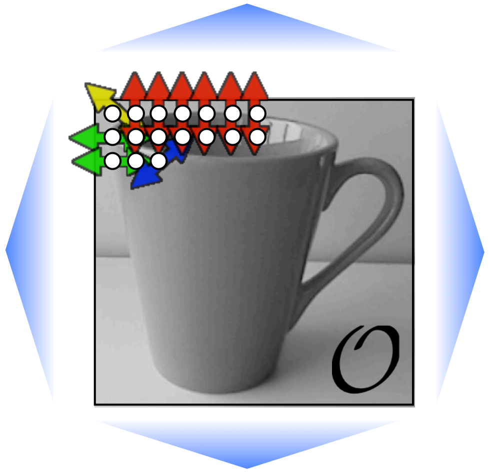
Making the Similarity Measure Robust to Small Motions



$$\mathcal{E}_1(\mathcal{I}, \mathcal{O}, \mathbf{c}) = \sum_{\mathbf{r}} \mathbb{1}[\text{orientation}(\mathcal{I}, \mathbf{c} + \mathbf{r}) = \text{orientation}(\mathcal{O}, \mathbf{r})]$$

$$\mathcal{E}_2(\mathcal{I}, \mathcal{O}, \mathbf{c}) = \max_{\text{motion}} \mathcal{E}_1(\mathcal{I}, \mathbf{w}(\mathcal{O}, \text{motion}), \mathbf{c})$$

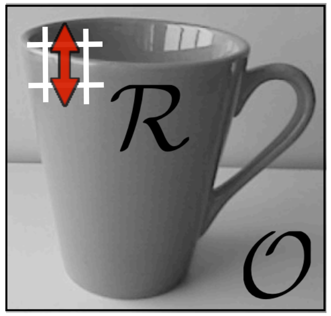
Making the Similarity Measure Robust to Small Motions



$$\mathcal{E}_1(\mathcal{I}, \mathcal{O}, \mathbf{c}) = \sum_{\mathbf{r}} \mathbb{1}[\text{orientation}(\mathcal{I}, \mathbf{c} + \mathbf{r}) = \text{orientation}(\mathcal{O}, \mathbf{r})]$$

$$\mathcal{E}_2(\mathcal{I}, \mathcal{O}, \mathbf{c}) = \max_{\text{motion}} \mathcal{E}_1(\mathcal{I}, \mathbf{w}(\mathcal{O}, \text{motion}), \mathbf{c})$$

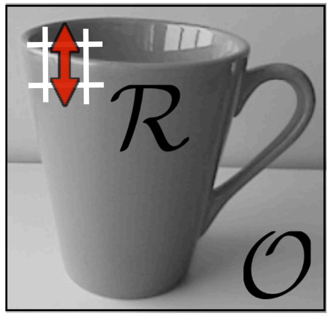
Downsampling



$$\mathcal{E}_2(\mathcal{I}, \mathcal{O}, \mathbf{c}) = \max_{\text{motion}} \sum_{\mathbf{r}} \mathbb{1}[\text{orientation}(\mathcal{I}, \mathbf{c} + \mathbf{r}) = \text{orientation}(\mathbf{w}(\mathcal{O}, \text{motion}), \mathbf{r})]$$

$$\mathcal{E}_3(\mathcal{I}, \mathcal{O}, \mathbf{c}) = \max_{\text{motion}} \sum_{\mathcal{R}} \mathbb{1}[\begin{smallmatrix}\text{dominant}\\\text{orientation}\end{smallmatrix}(\mathcal{I}, \mathbf{c} + \mathcal{R}) = \begin{smallmatrix}\text{dominant}\\\text{orientation}\end{smallmatrix}(\mathbf{w}(\mathcal{O}, \text{motion}), \mathcal{R})]$$

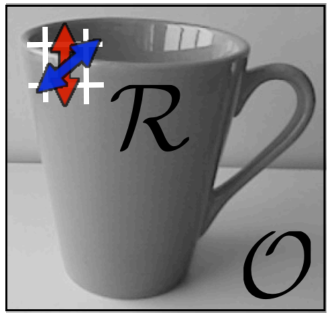
Ignoring the Dependencies between the Regions...



$$\mathcal{E}_3(\mathcal{I}, \mathcal{O}, \mathbf{c}) = \max_{\text{motion}} \sum_{\mathcal{R}} \mathbb{1}\left[\underset{\text{orientation}}{\text{dominant}}(\mathcal{I}, \mathbf{c} + \mathcal{R}) = \underset{\text{orientation}}{\text{dominant}}(\mathbf{w}(\mathcal{O}, \text{motion}), \mathcal{R})\right]$$

$$\mathcal{E}_4(\mathcal{I}, \mathcal{O}, \mathbf{c}) = \sum_{\mathcal{R}} \max_{\text{motion}} \mathbb{1}\left[\underset{\text{orientation}}{\text{dominant}}(\mathcal{I}, \mathbf{c} + \mathcal{R}) = \underset{\text{orientation}}{\text{dominant}}(\mathbf{w}(\mathcal{O}, \text{motion}), \mathcal{R})\right]$$

Lists of Dominant Orientations

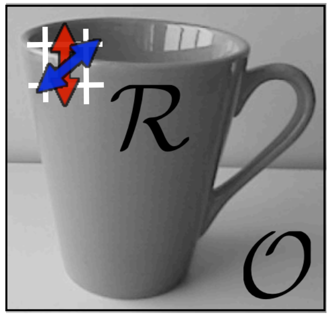


$$\mathcal{E}_4(\mathcal{I},\mathcal{O},\mathbf{c}) = \sum_{\mathcal{R}} \max_{\text{motion}} \mathbb{1}[\ \underset{\text{orientation}}{\text{dominant}}(\mathcal{I},\mathbf{c}+\mathcal{R}) = \underset{\text{orientation}}{\text{dominant}}(\mathbf{w}(\mathcal{O},\text{motion}),\mathcal{R})]$$

$$\mathcal{E}_4(\mathcal{I},\mathcal{O},\mathbf{c}) = \sum_{\mathcal{R}} \mathbb{1}[\ \underset{\text{orientation}}{\text{dominant}}(\mathcal{I},\mathbf{c}+\mathcal{R}) \in \begin{matrix}\text{dominant}\\\text{orientations}\\\text{over all}\\\text{motions}\end{matrix}(\mathbf{w}(\mathcal{O},\text{motion}),\mathcal{R})]$$

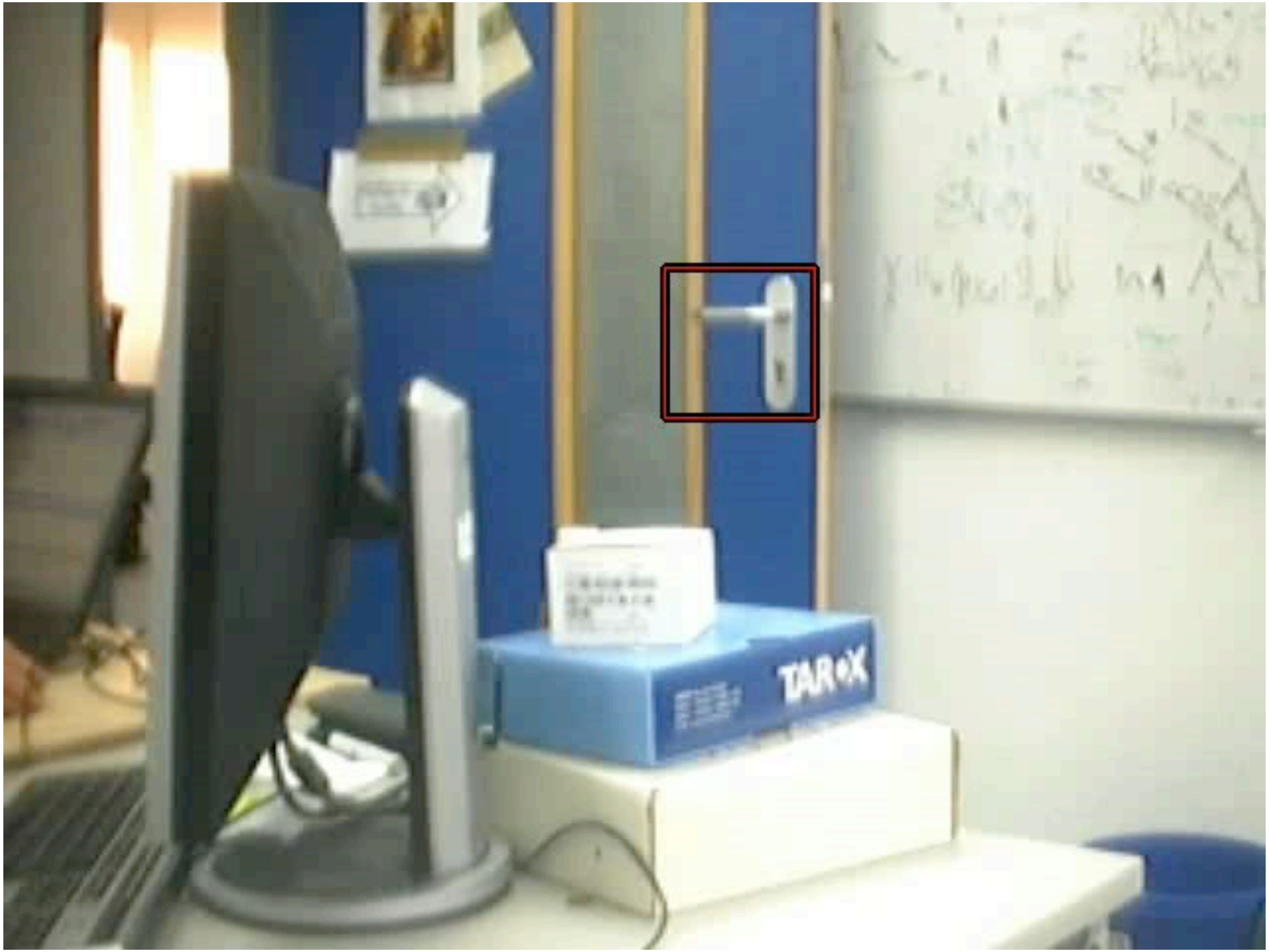
Fast Computation with Bitwise Operations

00001100



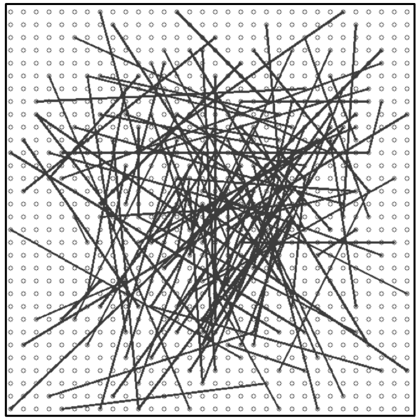
$$\mathcal{E}_4(\mathcal{I}, \mathcal{O}, \mathbf{c}) = \sum_{\mathcal{R}} \mathbb{1}[\ \underset{\text{orientation}}{\text{dominant}}(\mathcal{I}, \mathbf{c} + \mathcal{R}) \in \begin{matrix}\text{dominant}\\ \text{orientations}\\ \text{over all}\\ \text{motions}\end{matrix}(\mathbf{w}(\mathcal{O}, \text{motion}), \mathcal{R})]$$

$$\mathcal{E}_{\text{final}}(\mathcal{I}, \mathcal{O}, \mathbf{c}) = \sum_{\mathcal{R}} \mathbb{1}[\mathrm{I}_{\mathbf{c}+\mathcal{R}} \otimes \mathrm{O}_{\mathcal{R}} \neq 0]$$



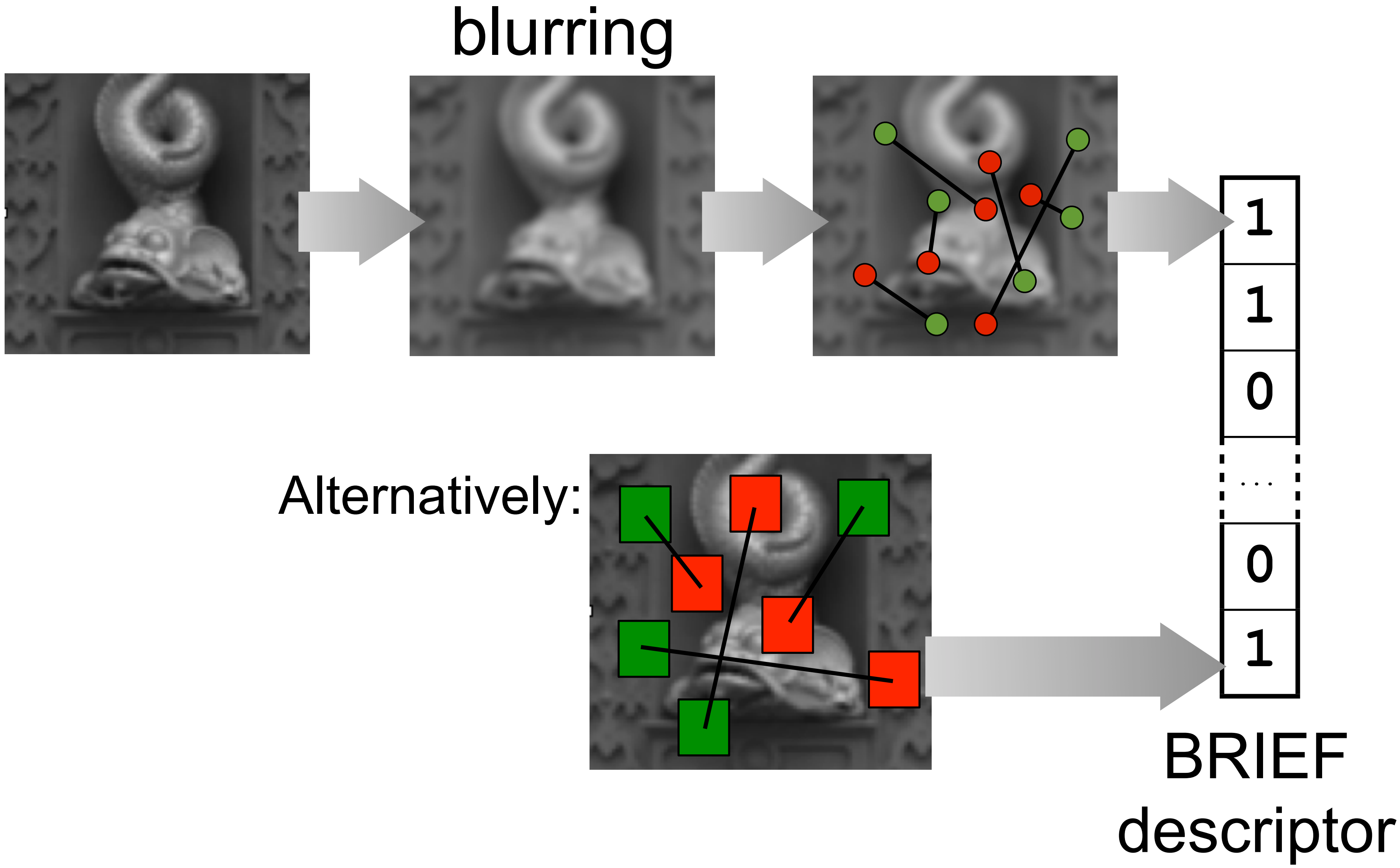
Code available under LGPL license at

http://campar.in.tum.de/personal/hinterst/index/



BRIEF [ECCV'10]
very fast feature point descriptor

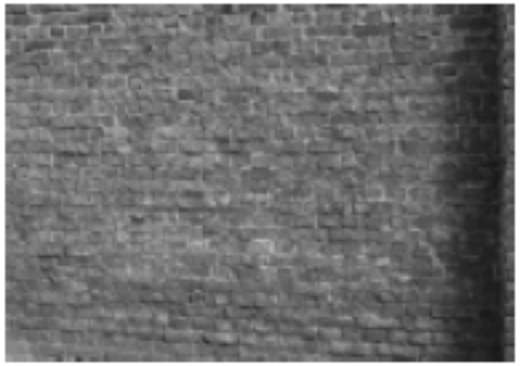
Joint work with Michael Calonder



blurring
Alternatively:
1
1
0
...
0
1
BRIEF descriptor

Evaluation

Wall



Graffiti

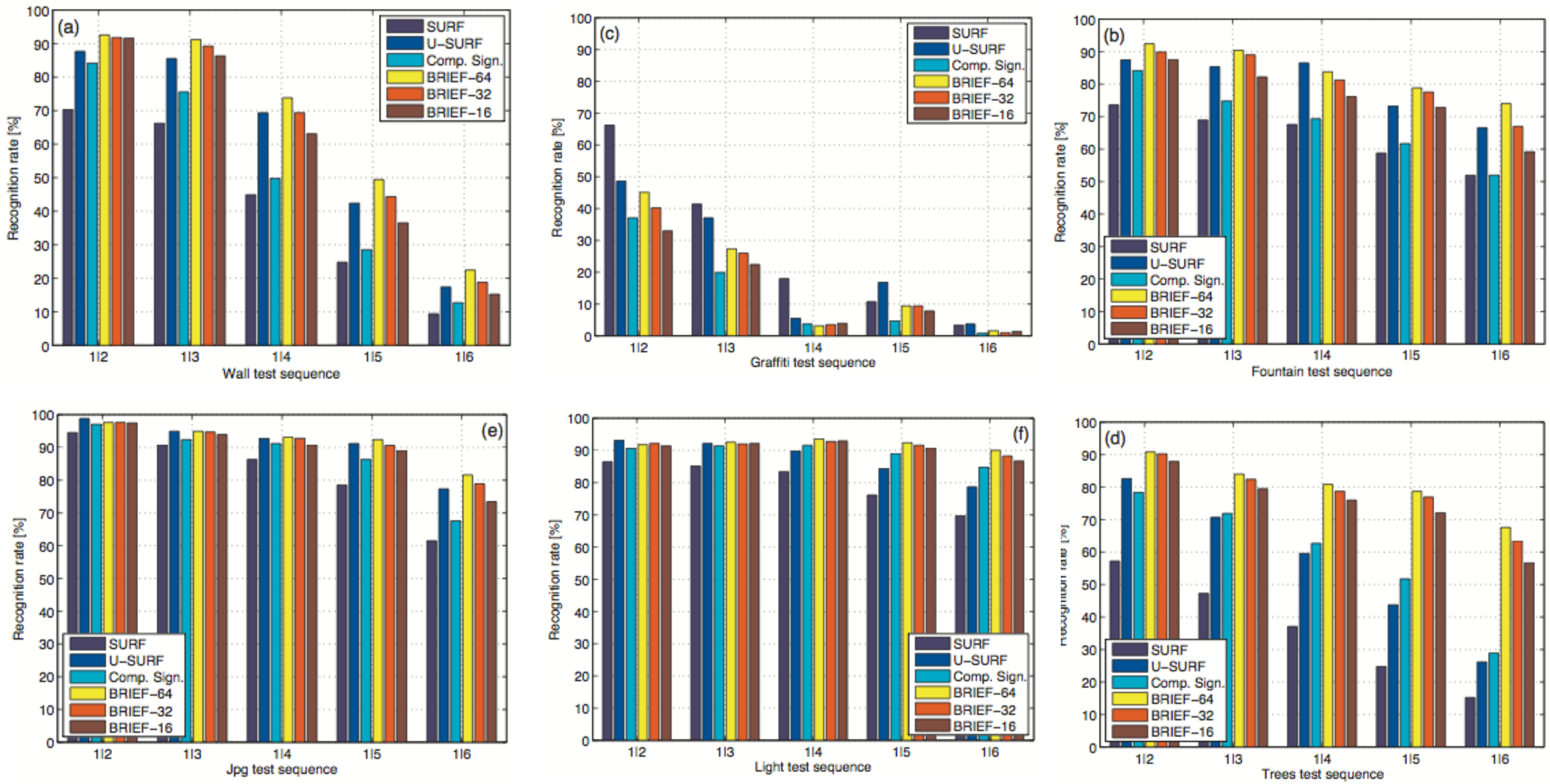
Fountain

Jpg

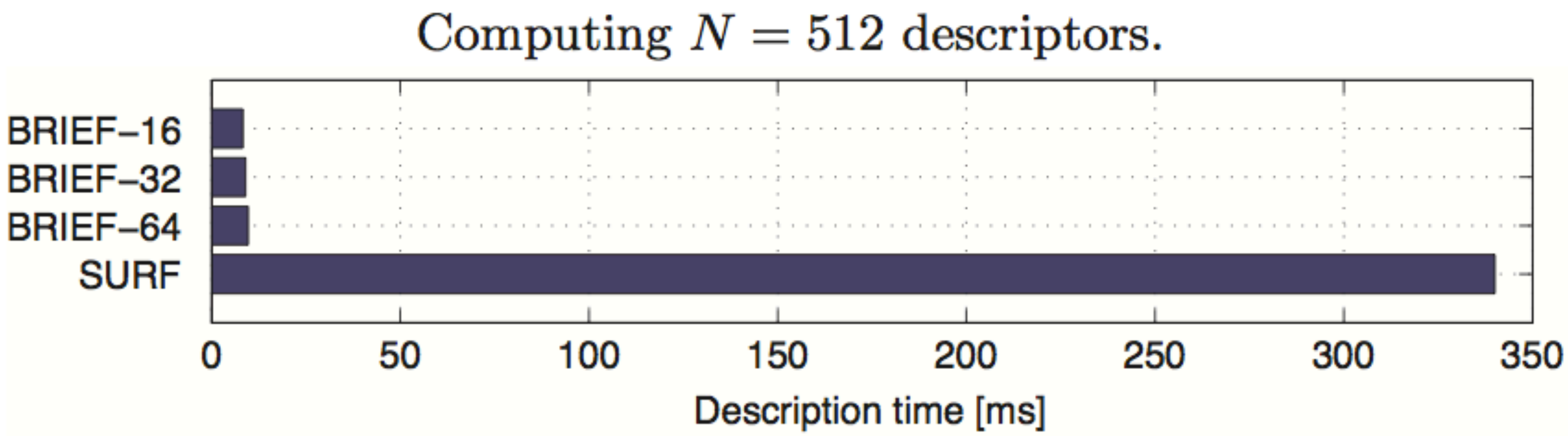
Light

Trees

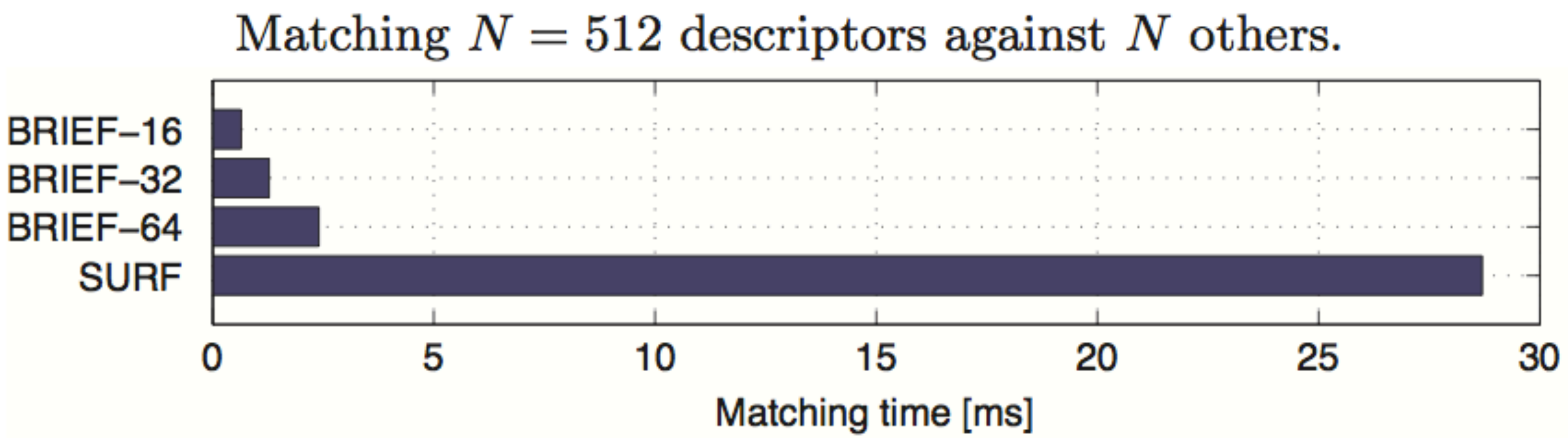
Evaluation


(a)
SURF
U–SURF
Comp. Sign.
BRIEF–64
BRIEF–32
BRIEF–16
Recognition rate [%]
1|2
1|3
1|4
1|5
1|6
Wall test sequence
(c)
Graffiti test sequence
(b)
Fountain test sequence
(e)
Jpg test sequence
(f)
Light test sequence
(d)
Trees test sequence

Computation Speed



For BRIEF, most of the time is spent in blurring the patches.

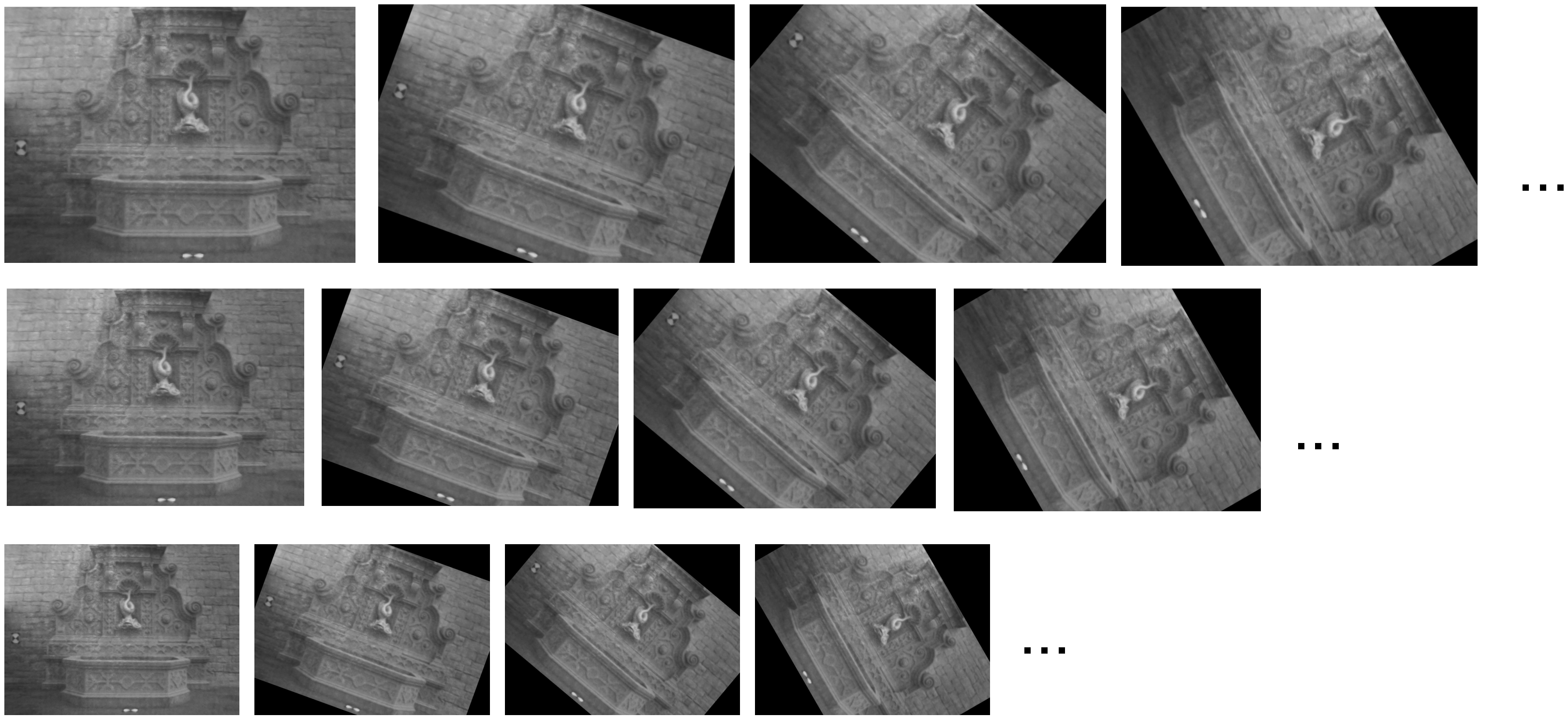


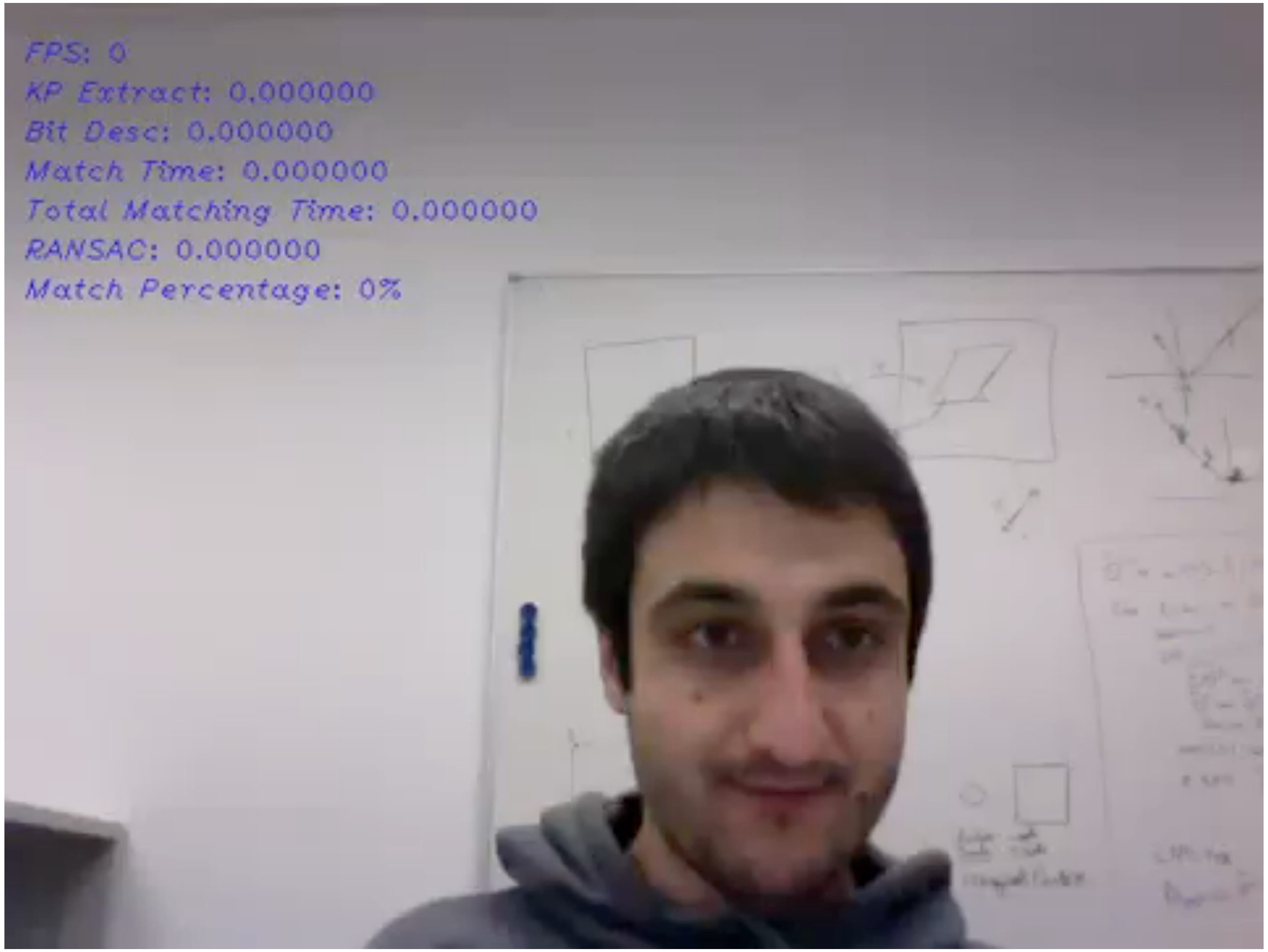
Matching BRIEF descriptors is done using the Hamming distance, which is very fast to compute using the `popcount` instruction on recent Intel CPUs.

Rotation and Scale Invariance

Duplicate the Descriptors:

18 rotations x 3 scales





code released in GPL on CVLab website

thanks !