Low-level Vision for Resource-limited Devices

IAGO SUÁREZ

IAGO.SUAREZ@THEGRAFFTER.COM

UNIVERSIDAD POLITÉCNICA DE MADRID

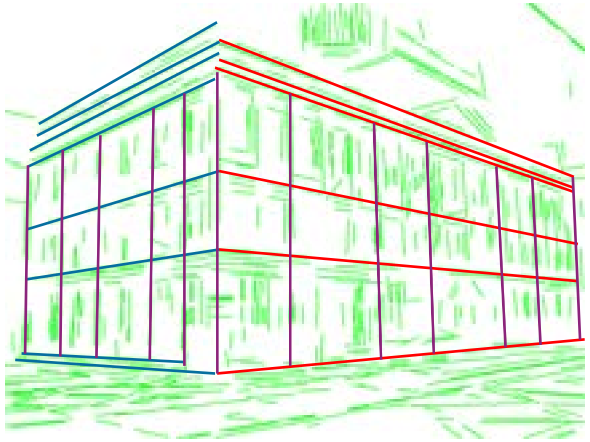
THE GRAFFTER S.L.

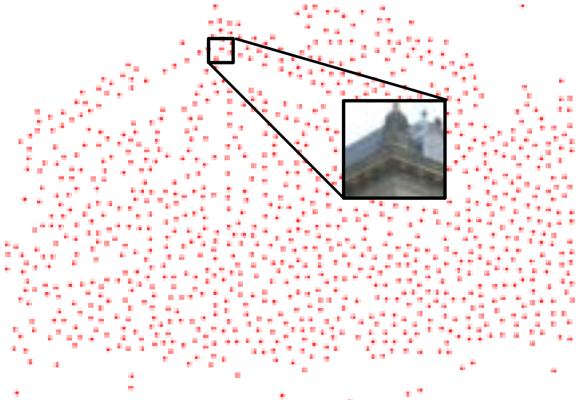
Visual abstract

1. Intro. Computer vision in resource-limited devices

2. Line segment detection

3. Full line detection and vanishing point estimation





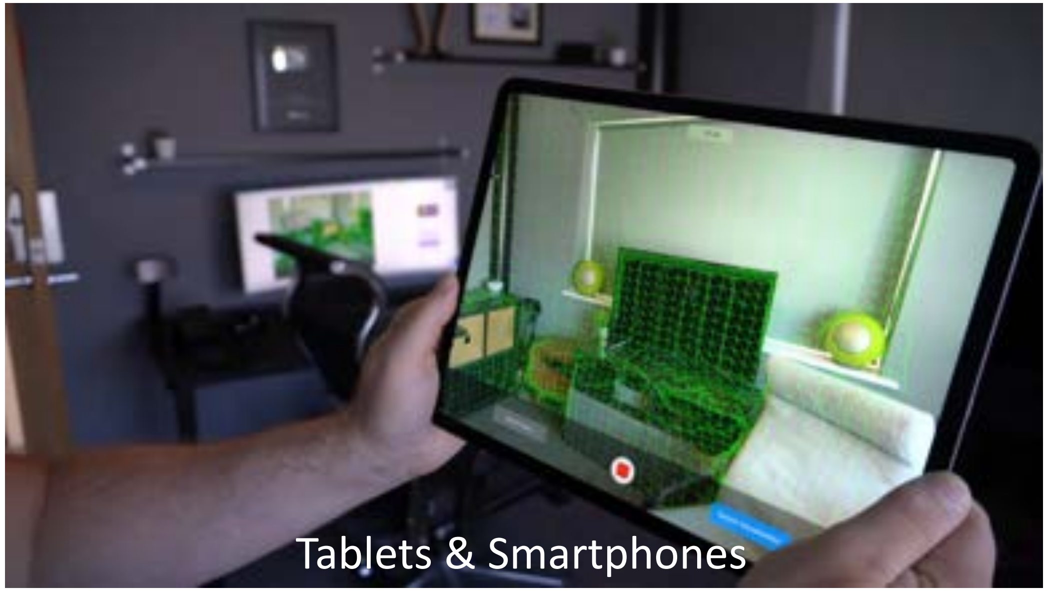
4. Local feature description

5. Results and conclusions

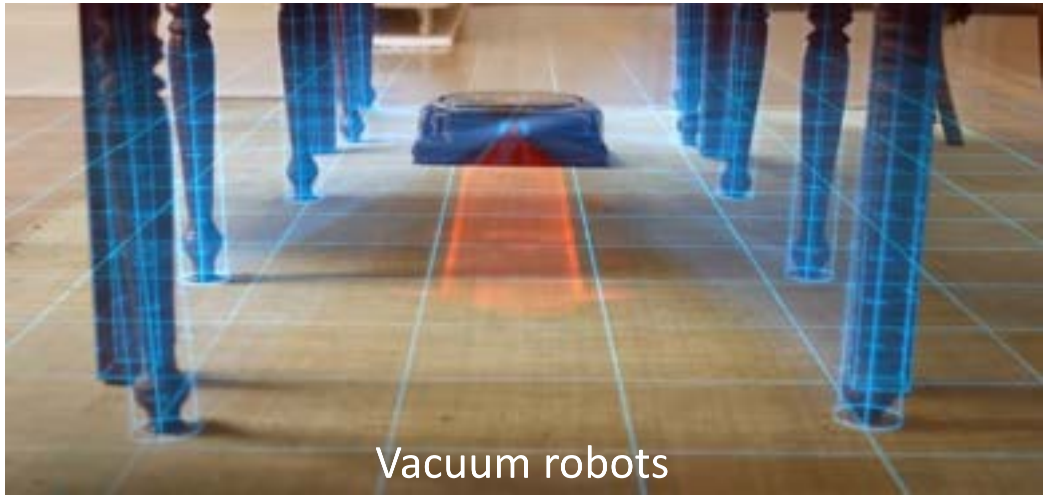
Index

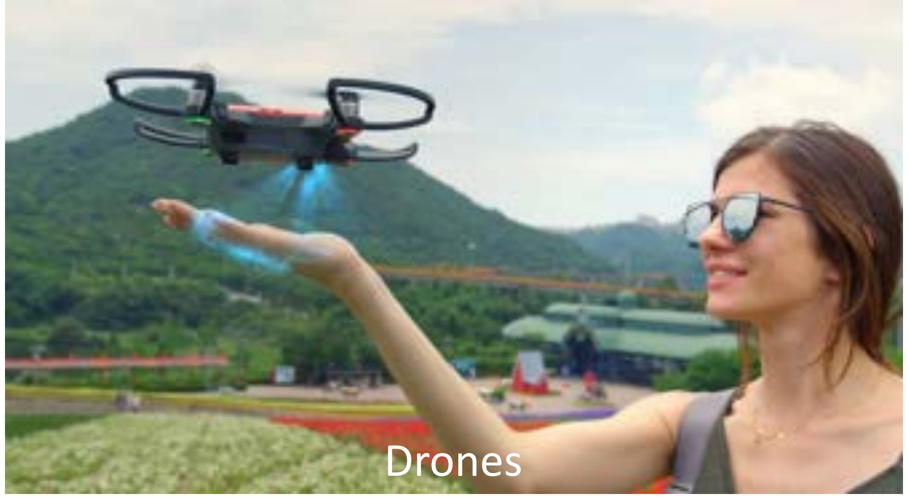
0. Abstract
1. **Introduction**
2. Line segment detection
3. Full line detection and vanishing point estimation
4. Local feature description
5. Industrial results
6. Conclusions

Computer Vision is ubiquitous

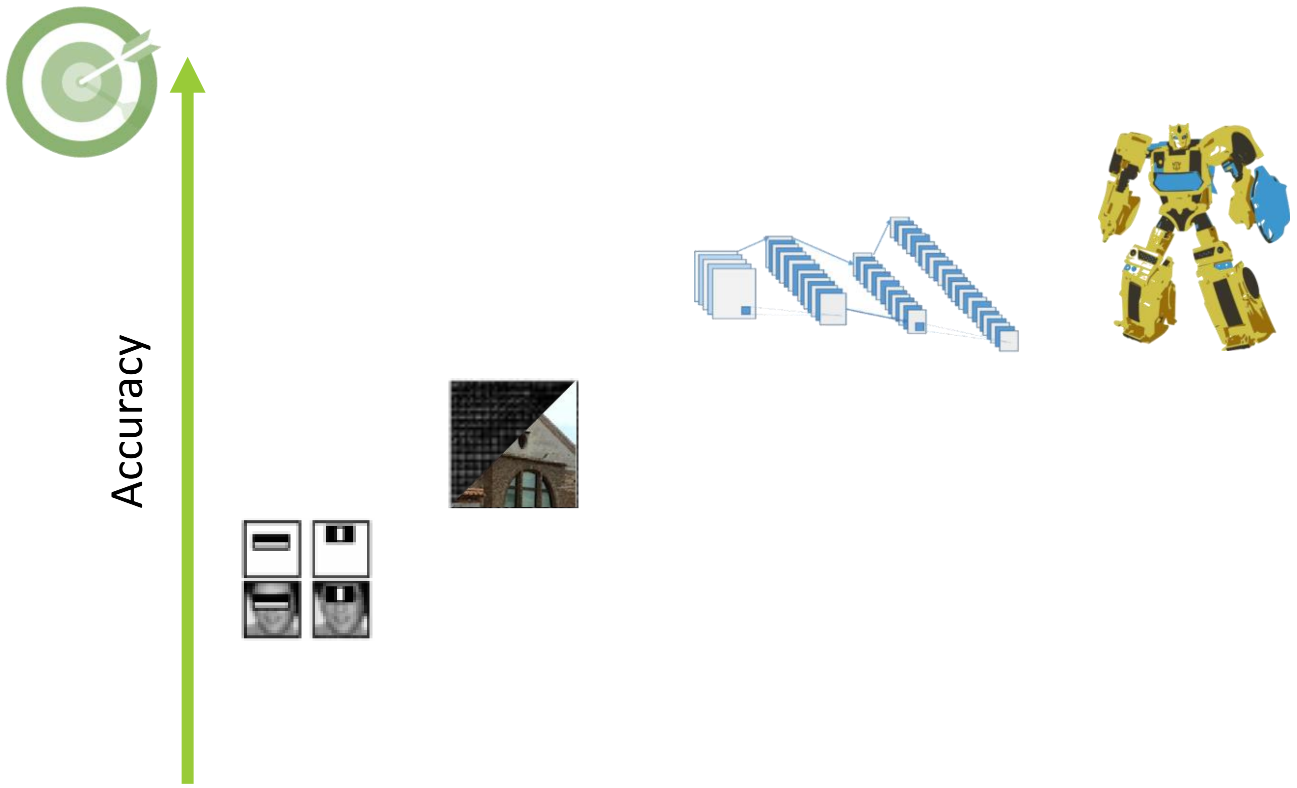

Tablets & Smartphones


Smart glasses


Vacuum robots


Drones

Accuracy vs resources curve



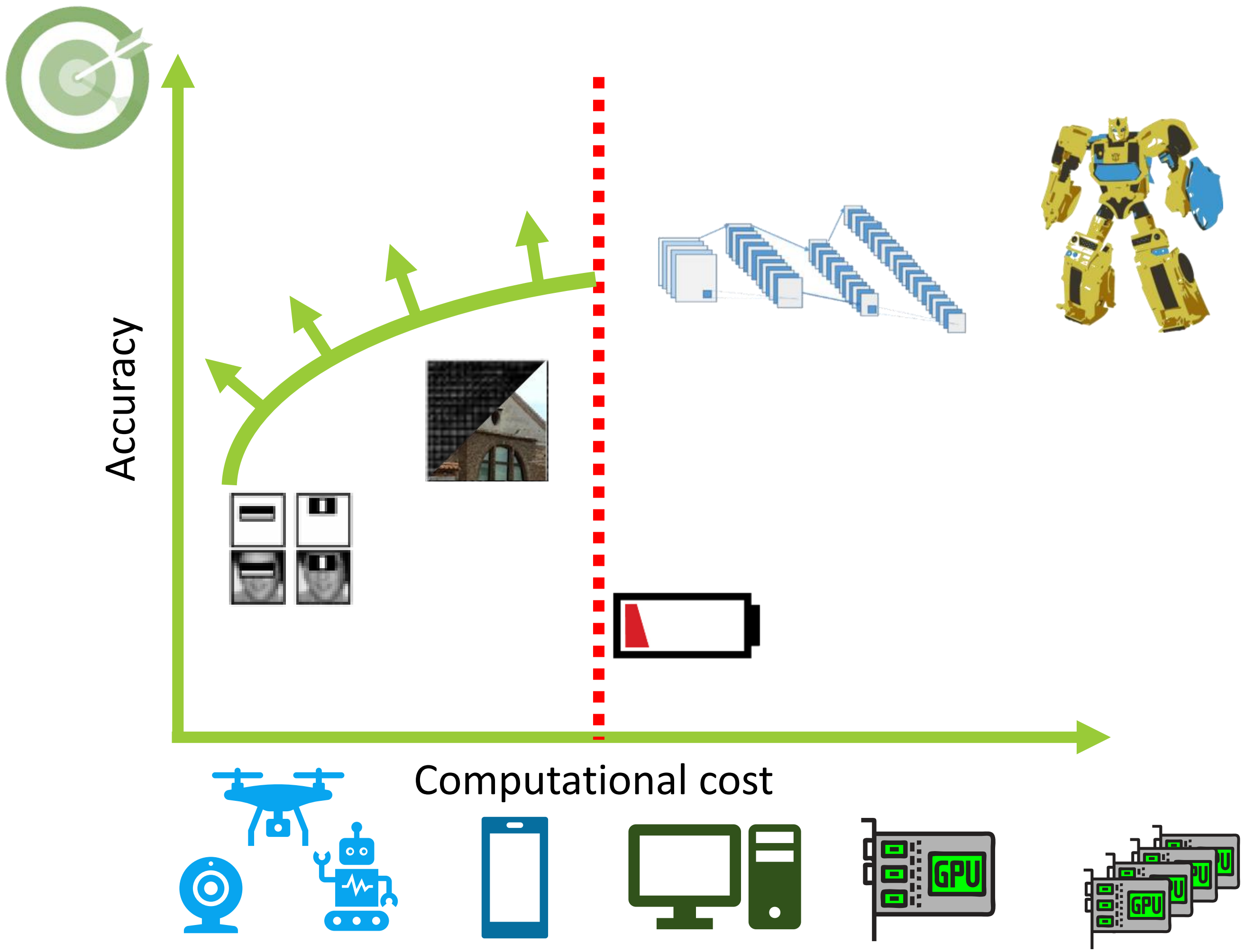
Accuracy vs resources curve

CV opportunities:

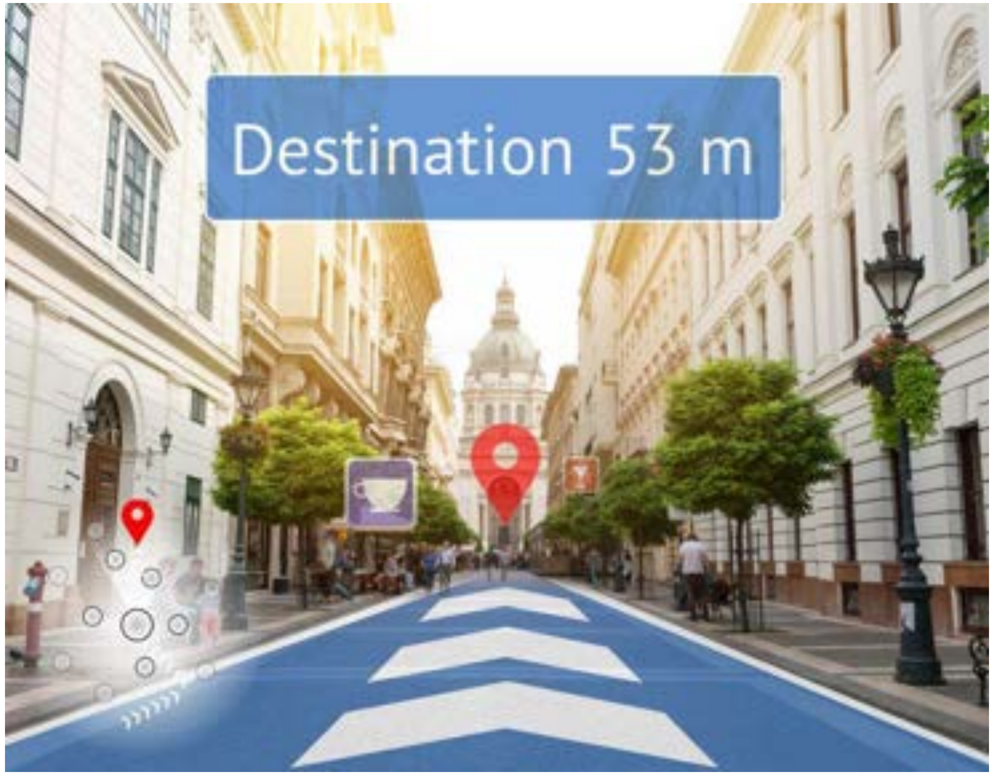
- IoT
- Drones
- Robotics
- Smartphones

Limited resources:

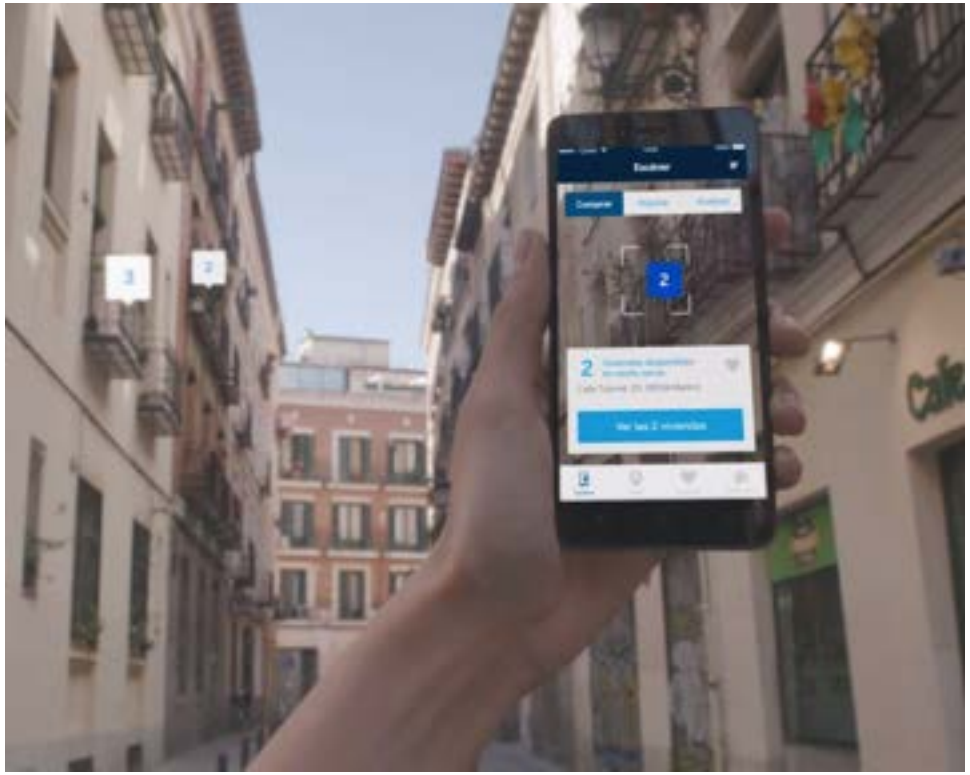
- Restricted-CPU, no-GPU
- Drones: Fly time
- Smartphones: App. consumption
- AR glasses: Time of usage and heat



Mixed Reality (MR) in the street



Visual navigation guidelines



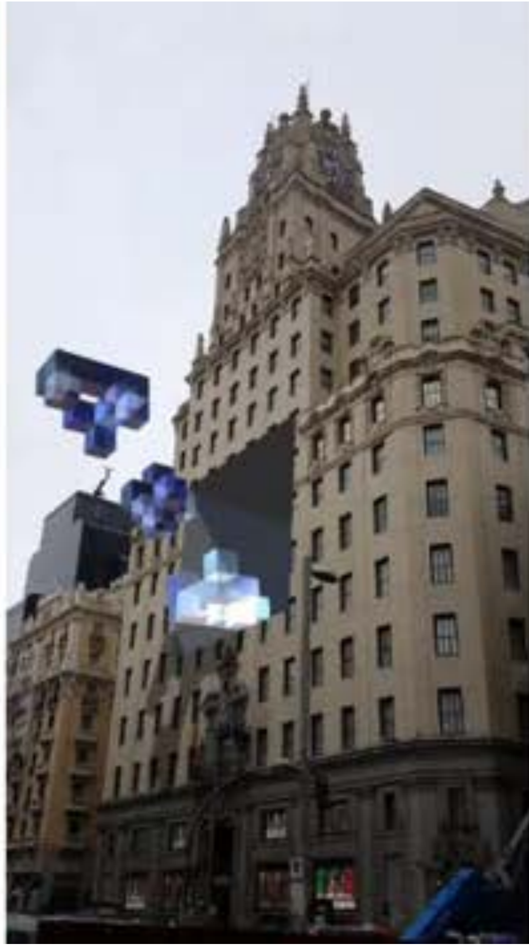
Accurate location-based information

Mixed Reality (MR) in the street

Successful examples in recent years:

Pokemon Go

Google Street View

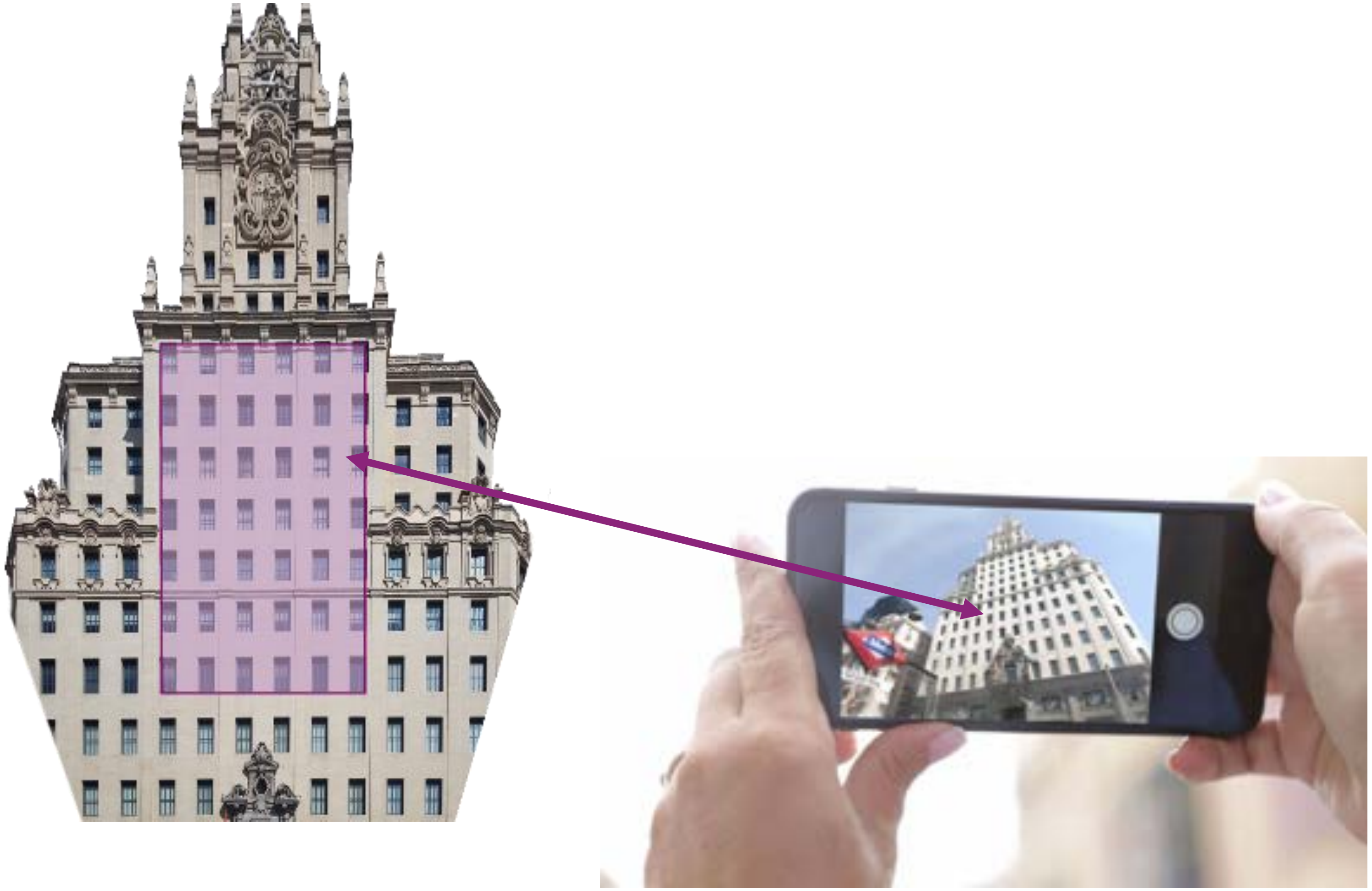


The Graffter: Urban Mixed Reality

MIXED REALITY IN THE BUILDING FAÇADES

The Graffter: Urban Mixed Reality

We face the image matching problem in mobile devices



Urban MR Challenges

It seems very easy!

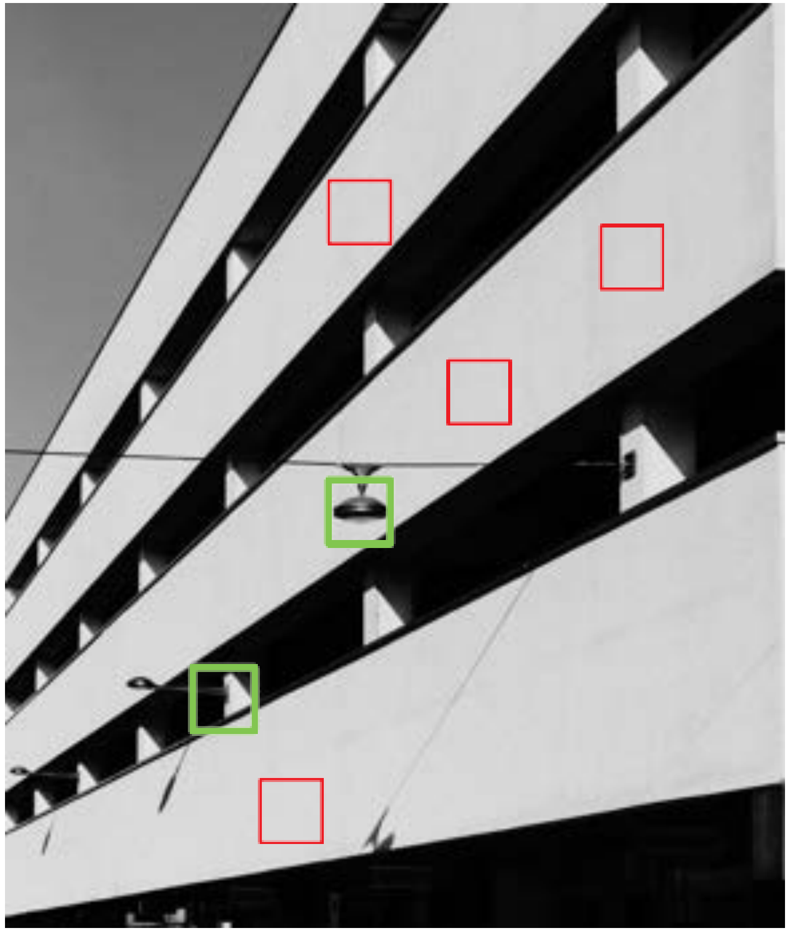
Urban MR Challenges: Building façade repeatability

Where is this window?

We need extra knowledge to locate repetitive patterns

Urban MR Challenges: Perspective and lighting

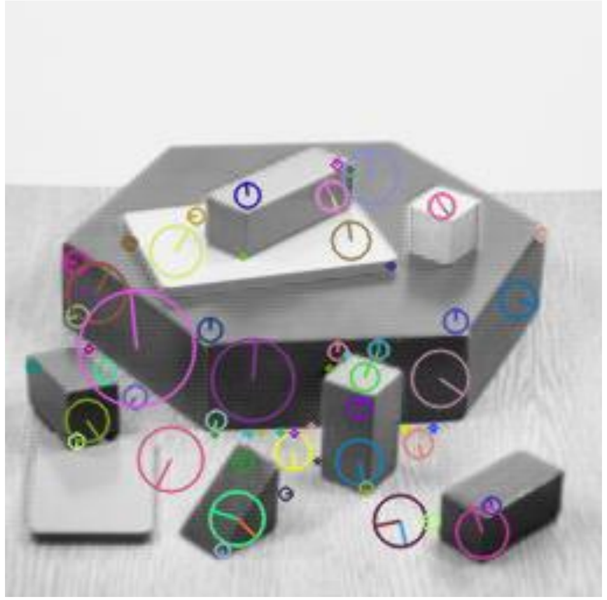
Urban MR Challenges: Lack of texture

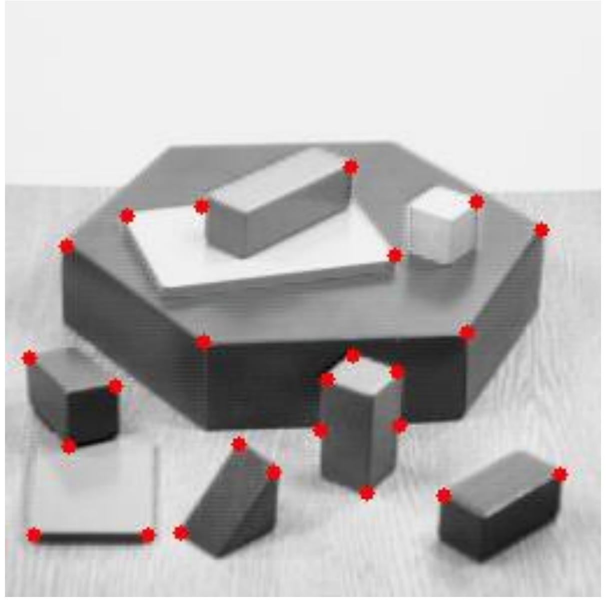


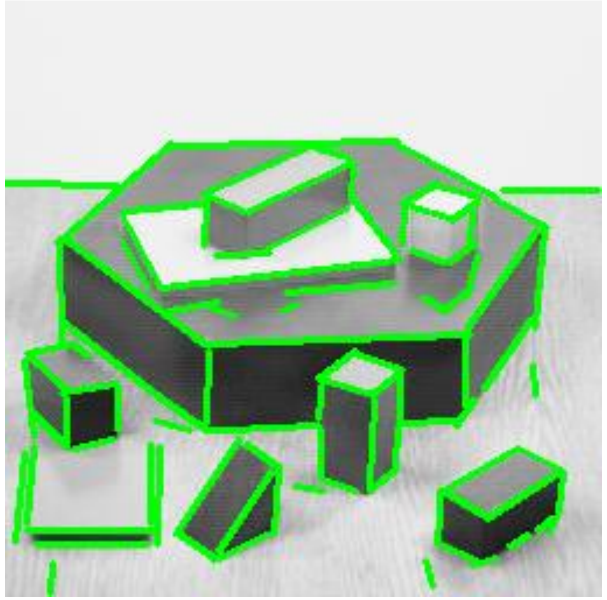
Texture-less building

Glazed surfaces

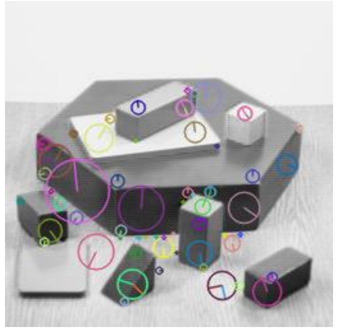
Localization pipeline: Local features


Blobs

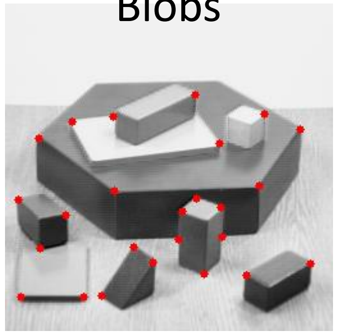

Corners


Line segments

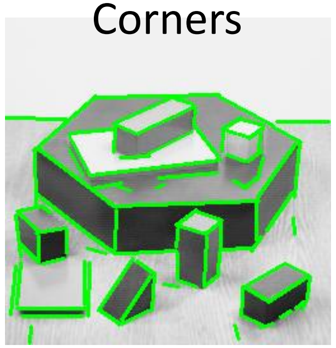
Localization pipeline: Local features



Blobs

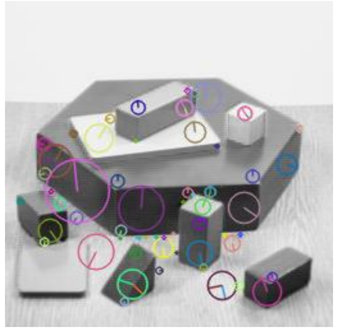


Corners

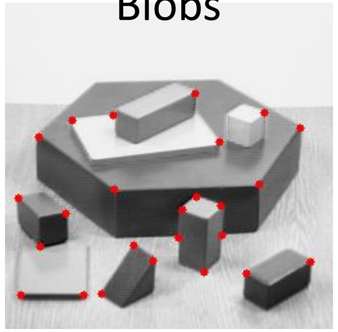


Line segments

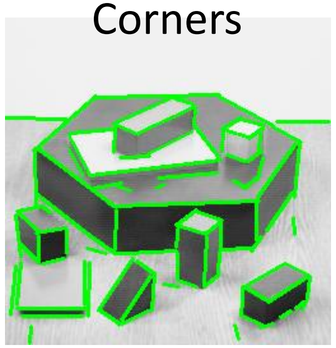
Localization pipeline: Local features



Blobs



Corners

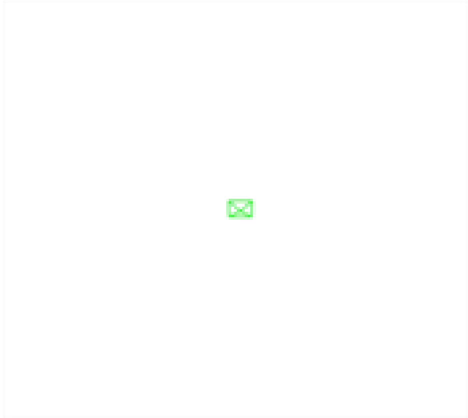


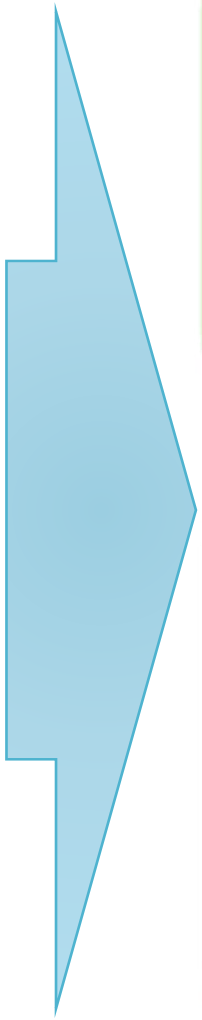
Line segments

Structure from Motion (SfM) (Agarwal, 2009)

Obj. pose estimation (Hu 2019)

Simultaneous Localization And Mapping (SLAM)

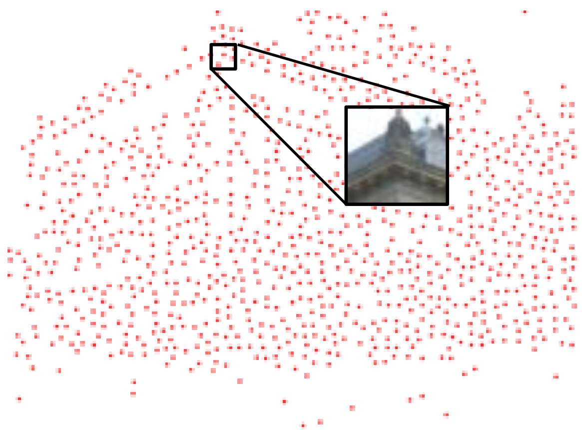




Place recognition (Lowry 2016)

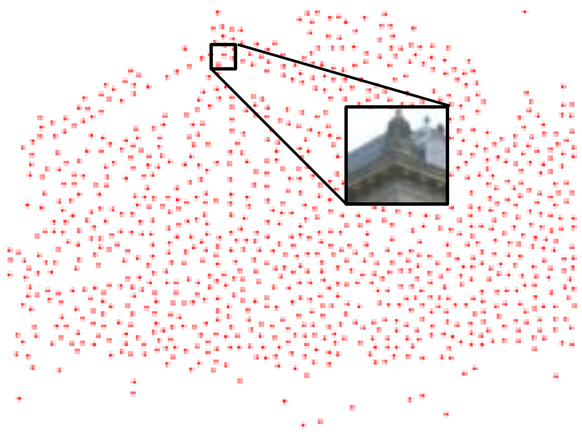
In this presentation…

1. The Graffter S.L and the industry, match corners to recognize buildings



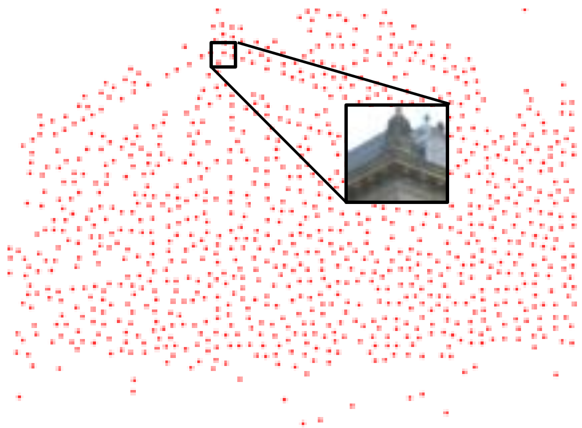
In this presentation…

1. The Graffter S.L and the industry, match corners to recognize buildings
    - There is room for improvements in the performance of this matching → **Improve the matching techniques for corner**



In this presentation…

1. The Graffter S.L and the industry, match corners to recognize buildings
   - There is room for improvements in the performance of this matching → **Improve the matching techniques for corner**



2. Some scenes have nearly no corners, only lines
   - Matching lines is harder than corners, more repetitiveness and lack of texture
   - Instead of failing with repetitiveness, we take advance of it → **We make contributions to a matching-free pipeline**:

Line segment detection → **Full line detection by segment grouping** → **Vanishing point estimation**

Index

0. Abstract
1. Introduction
2. **Line segment detection**
3. Full line detection and vanishing point estimation
4. Local feature description
5. Industrial results
6. Conclusions

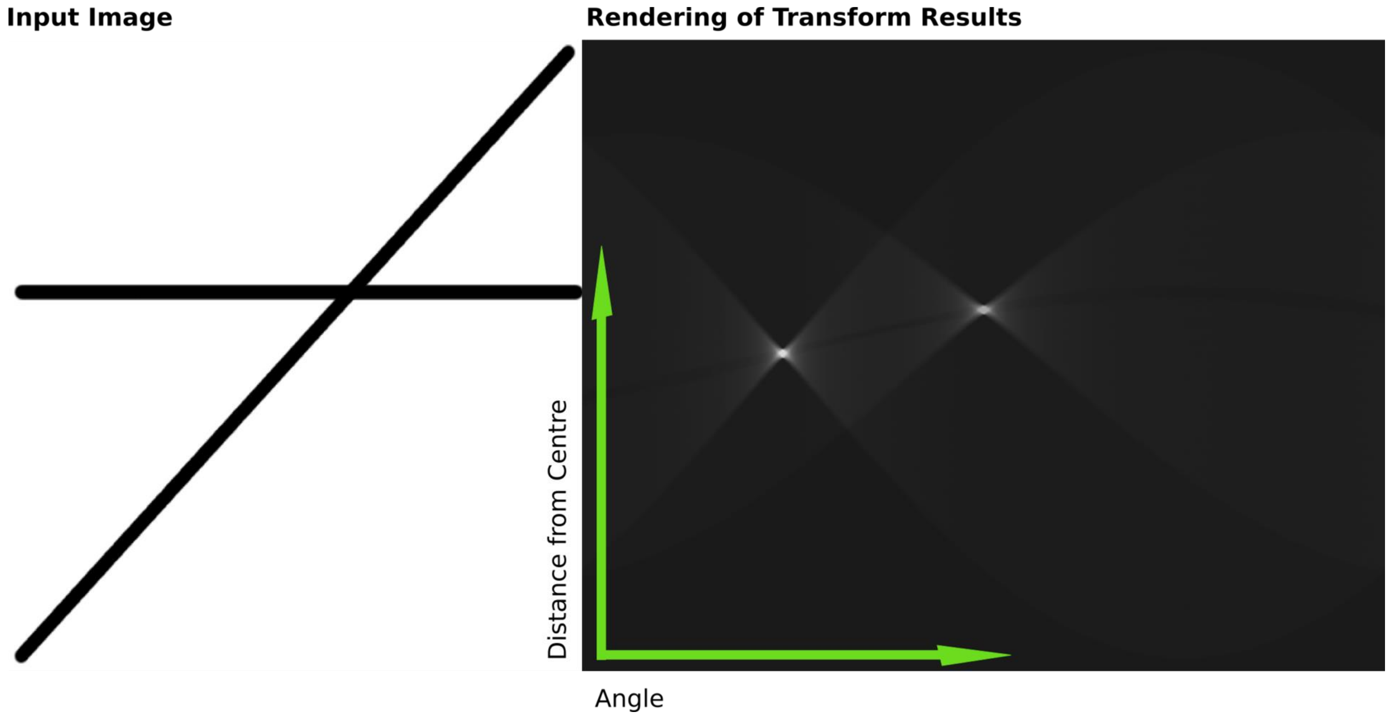
Lines and segments: definition

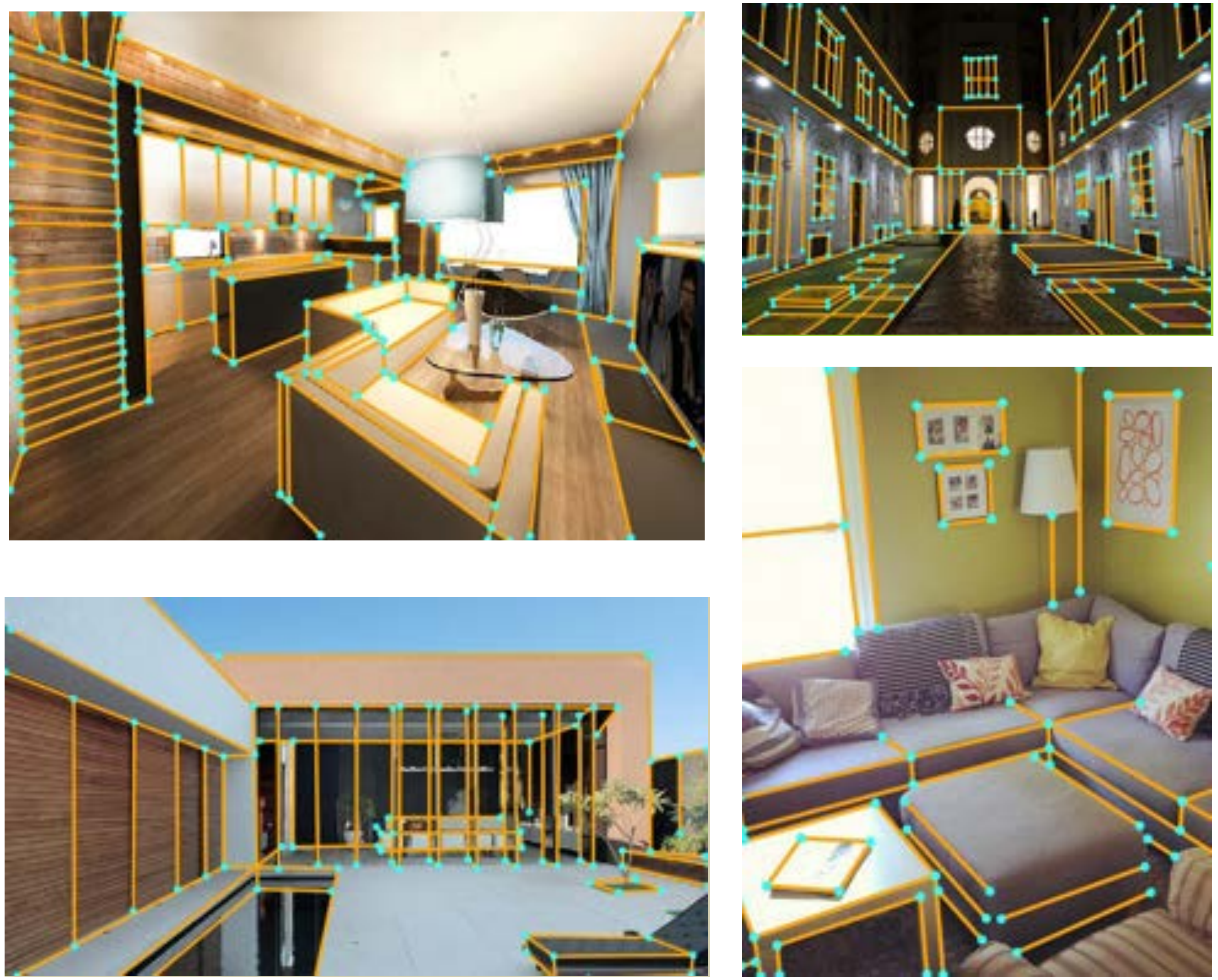
Segments

Lines

Line segment detection with Hough Transform

- **Hough Transform (HT)** (Hough, 1962)
- **Probabilistic Hough Transform** (Matas, J., 2000)
- **Kernel-Based Hough Transform** (Fernandes, 2008)
- **MCMLSD** (Almazan, 2017)





(Zhou2019)

Wireframe parsing approaches

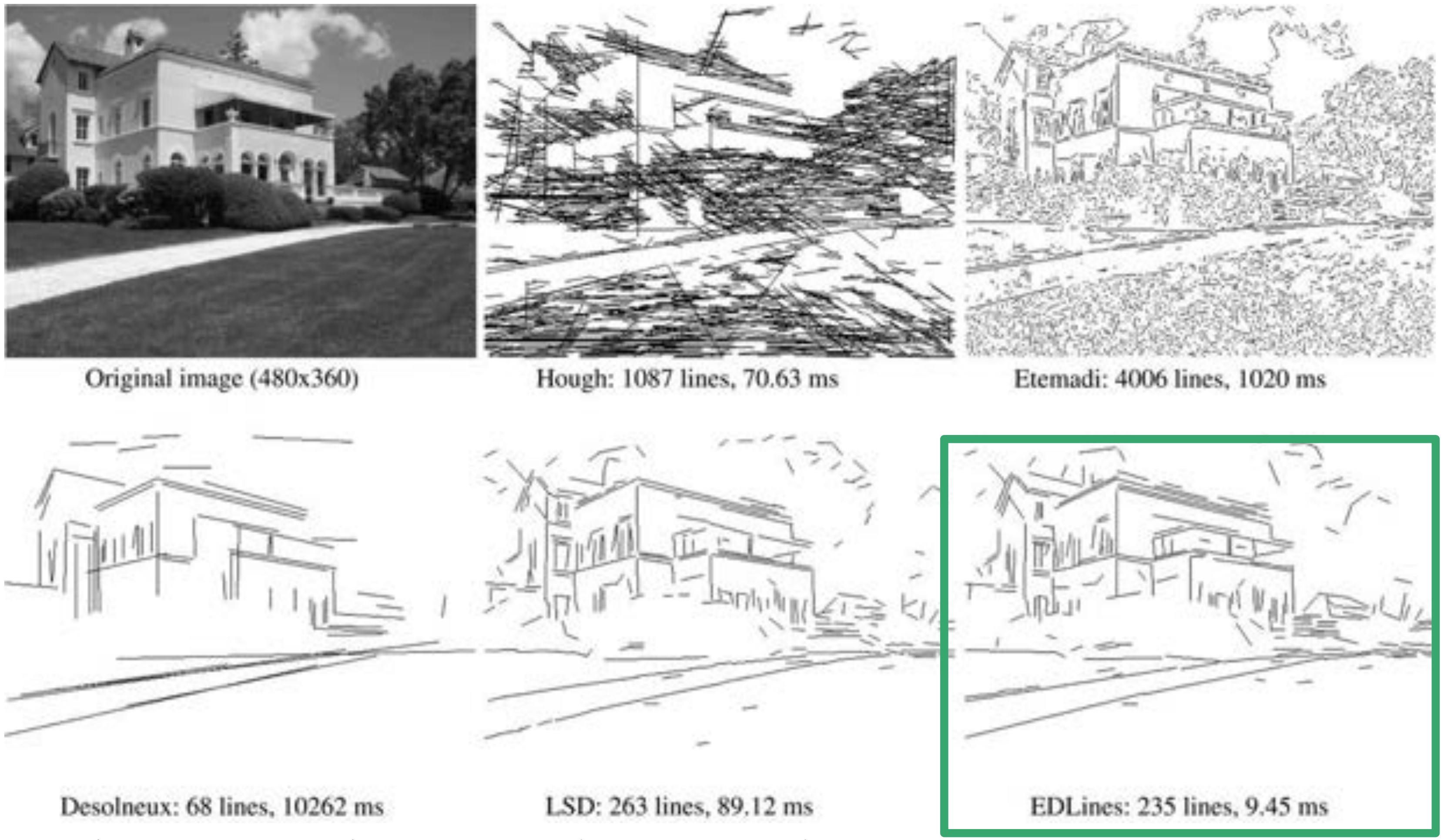
They predict the line segment endpoints in the borders of relevant objects:

- Human-labeled data
- Limited to home scenes

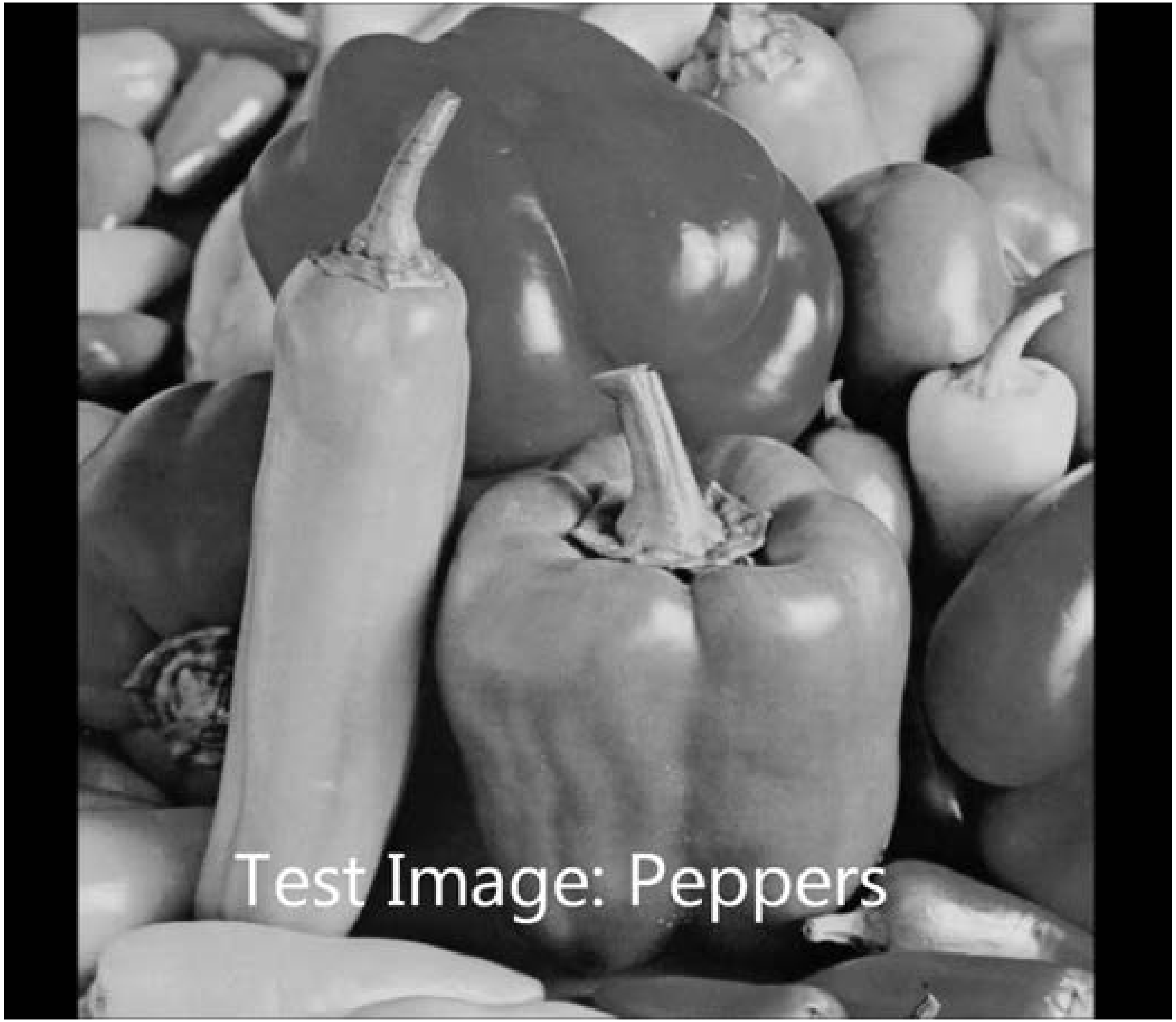
- **HAWP** (Xue, 2020)
- **LETR** (Xu, 2021)
- **ELSD** (Zhang, 2021)
- **F-Clip** (Dai, 2021)

General purpose line segment detectors

(Hough, 1962) (Etemadi, 1992)

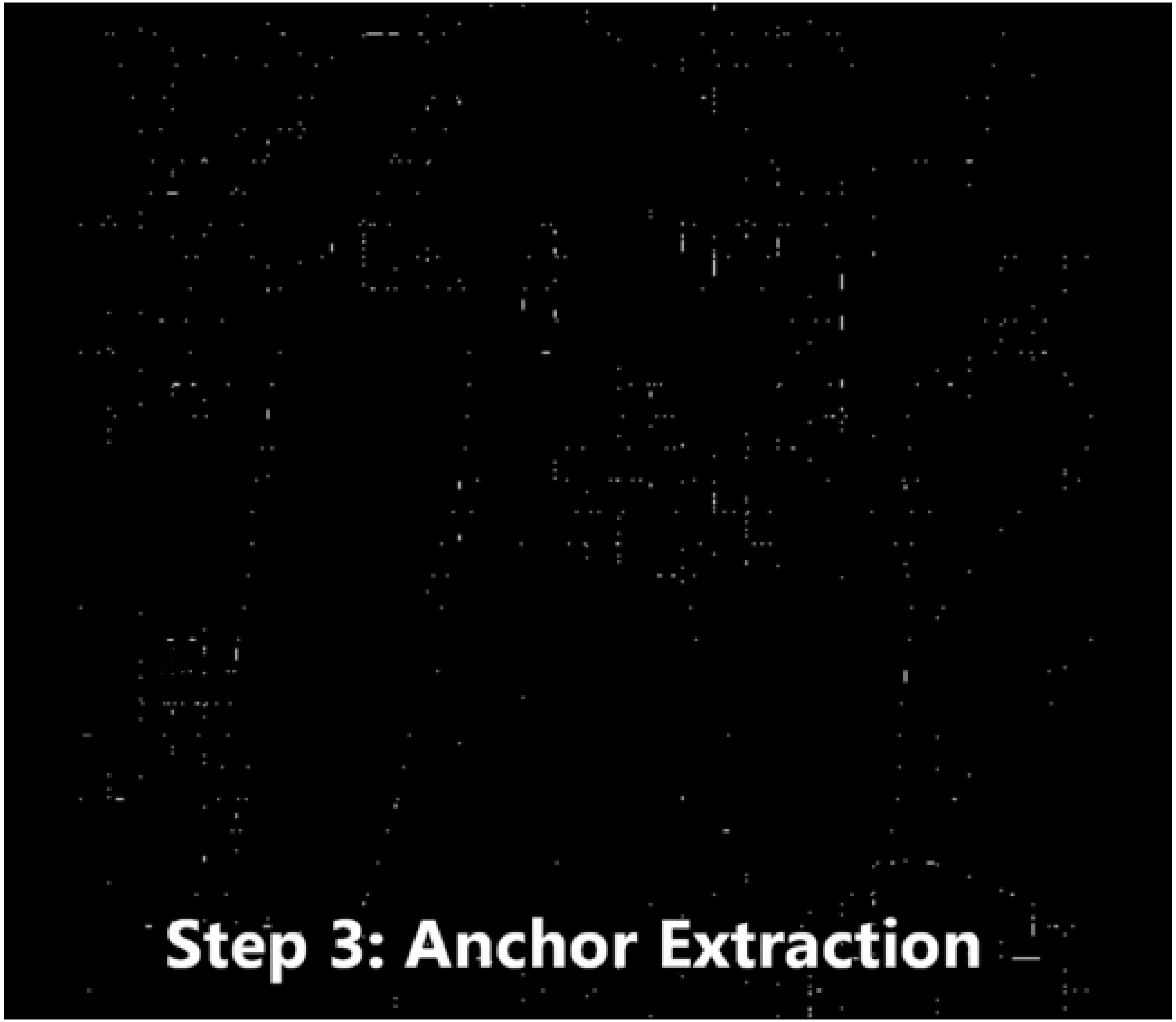


(Desolneux, 2008) (Von Gioi, 2008) (Akinlar, 2011)



Edge Drawing

Akinlar, C., & Topal, C. (2011). EDLines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, *32*(13), 1633-1642.



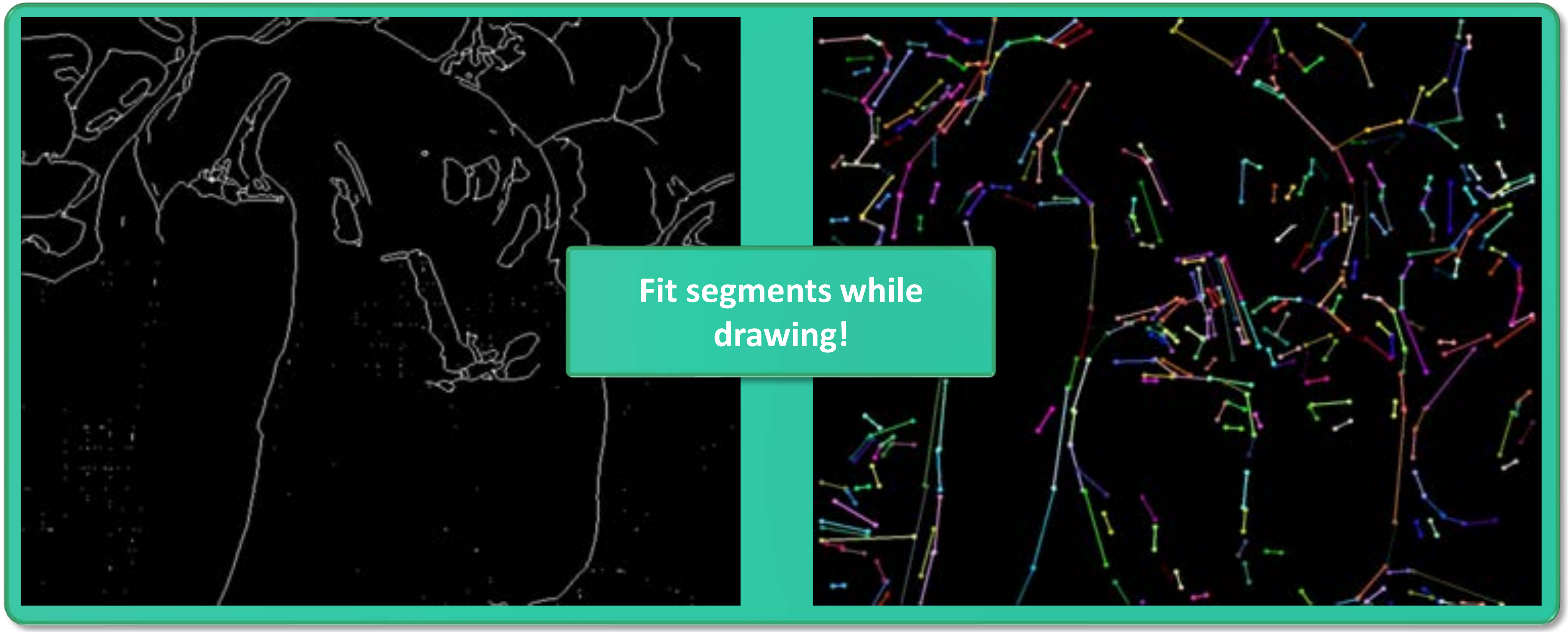
Edge Drawing

Akinlar, C., & Topal, C. (2011). EDLines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, *32*(13), 1633-1642.

Enhanced Line SEgment Drawing (ELSED)

In ELSED we propose to **merge**:

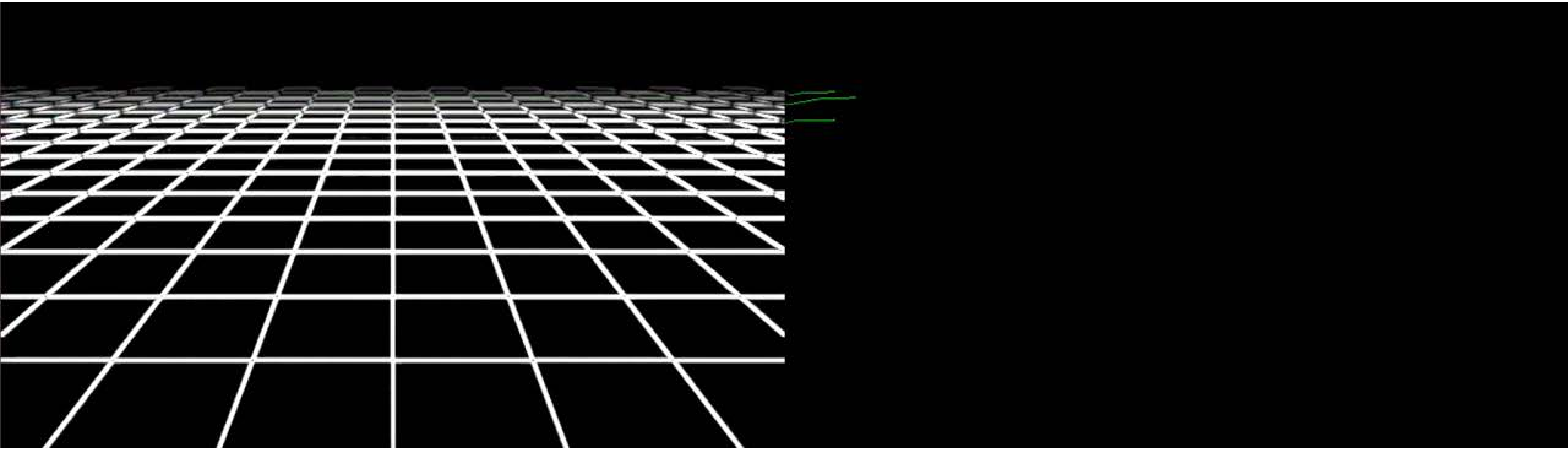
- 4) Edge drawing
- 5) Line segment fitting



Suárez, I., Buenaposada, J. M., & Baumela, L. (2021). ELSED: Enhanced Line SEgment Drawing. *Submitted to Pattern Recognition*

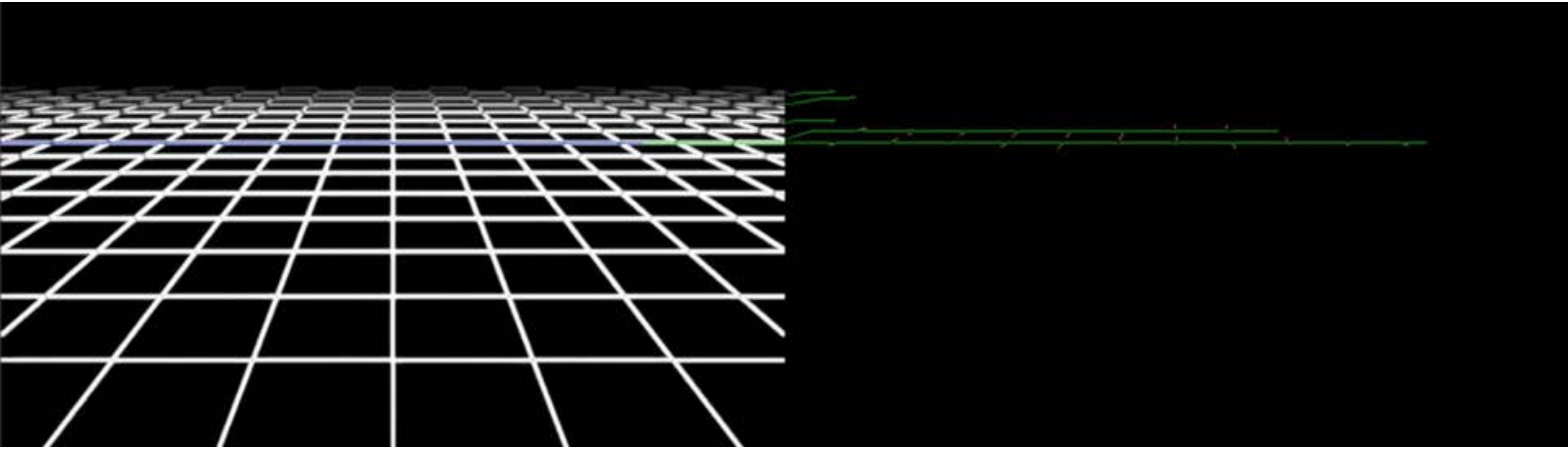
Enhanced Line SEgment Drawing (ELSED)

Because we are merging both steps, we can take advantage of it to improve the drawing process



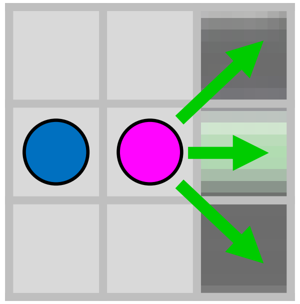
Enhanced Line SEgment Drawing (ELSED)

Because we are merging both steps, we can take advantage of it to improve the drawing process

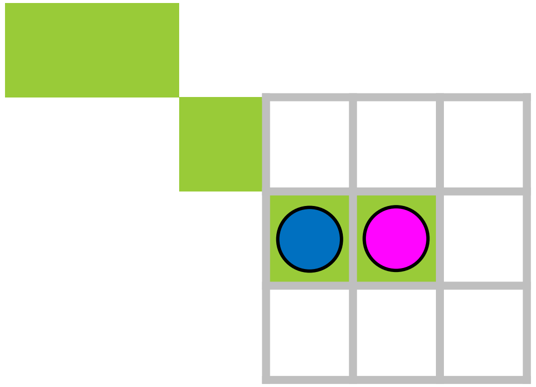


Enhanced Line SEgment Drawing (ELSED)

Because we are merging both steps, we can take advantage of it to improve the drawing process



Horizontal current
edge pixel
Go Right

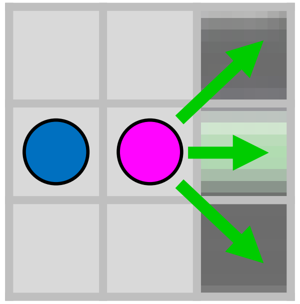


Diagonal case with
vertical edge
Go Right

Enhanced Line SEgment Drawing (ELSED)

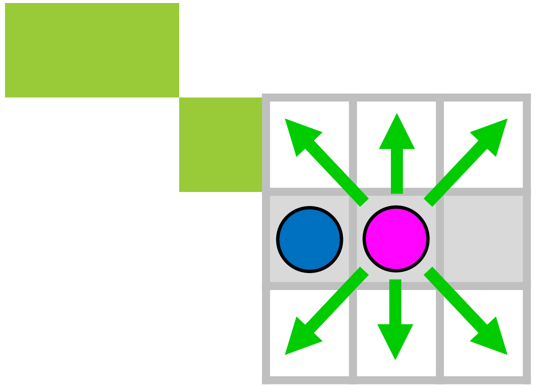
Because we are merging both steps, we can take advantage of it to improve the drawing process

Common case



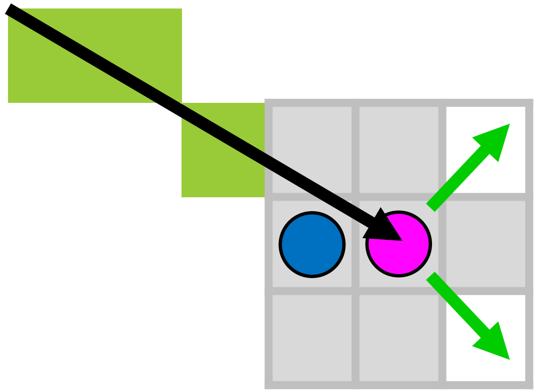
Horizontal current edge pixel
Go Right

Edge Drawing



Diagonal case with vertical edge
Go Right

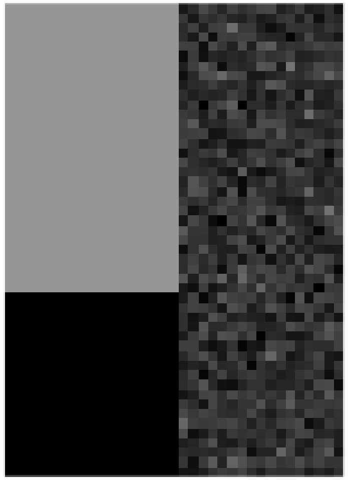
Enhanced Edge Drawing (EED)

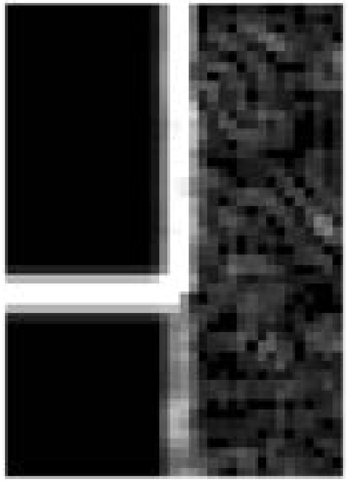


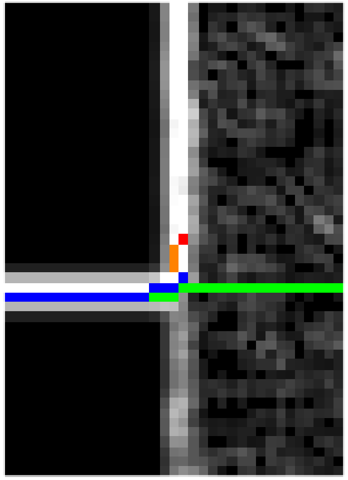
Diagonal case with vertical edge
Go Right

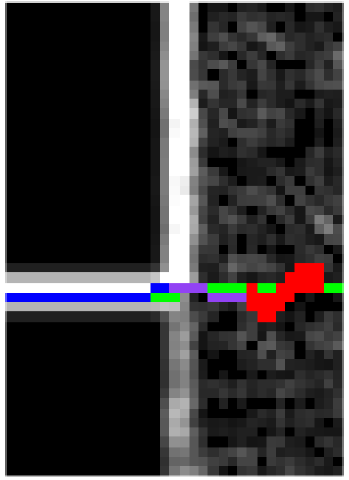
Line direction

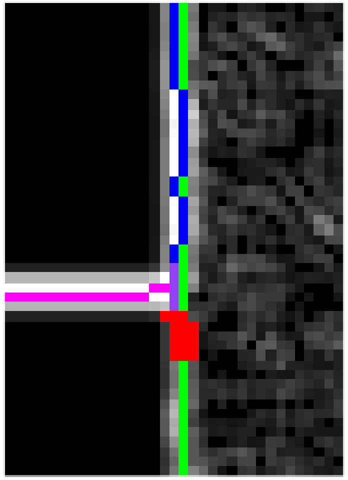
ELSED: Jump over discontinuities

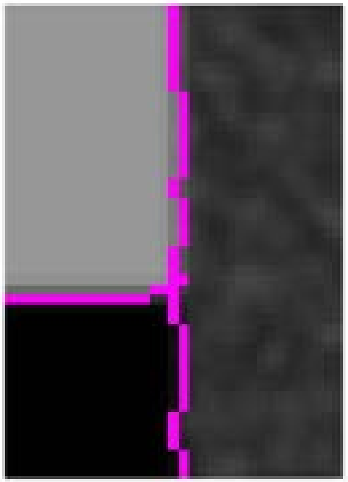

(a) Original Image

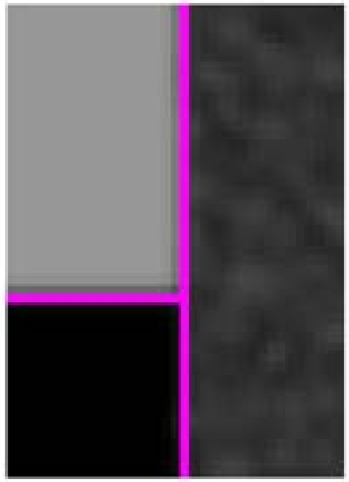

(b) Blurred Image gradient


(d) Discontinuity detection

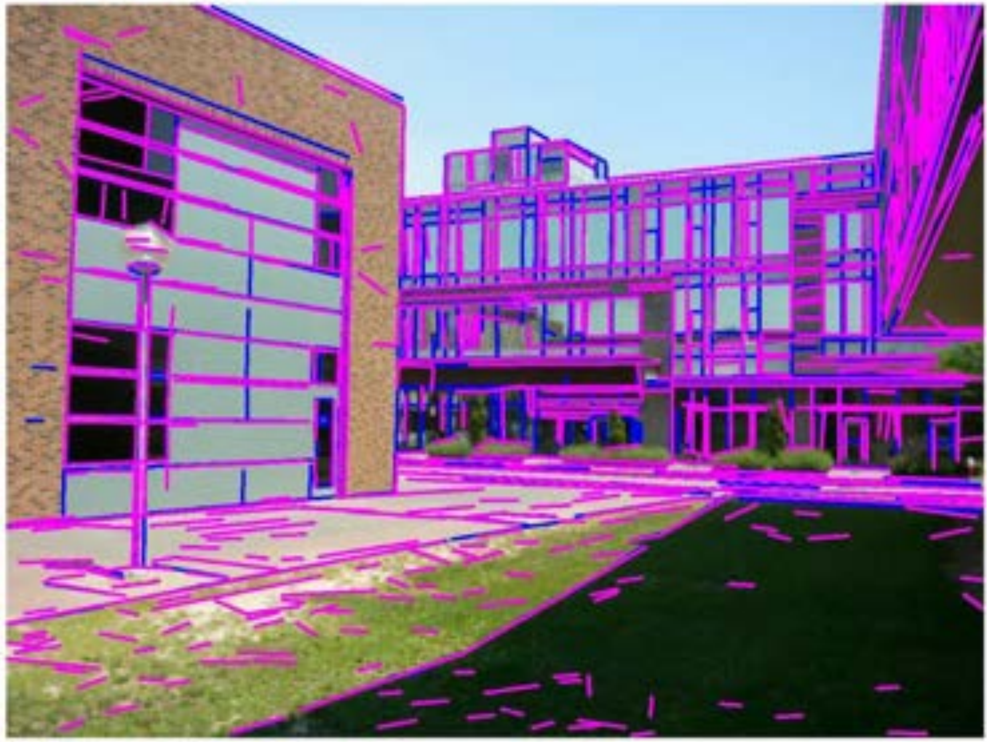

(e) First discontinuity check

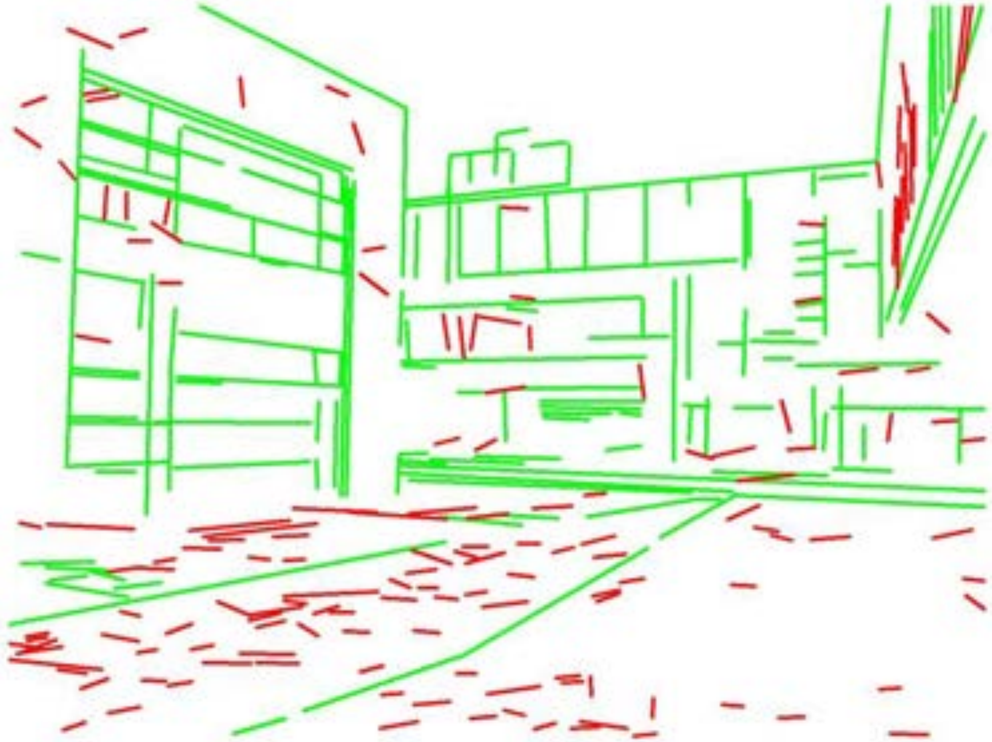

(g) 2$^{\text{nd}}$ discontinuity check


(h) Detected edges


(i) Detected Segments

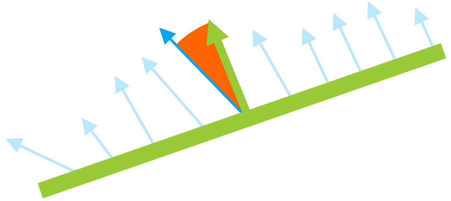
Line segment validation





Ground Truth · Detections · True positive detections · False positive detections

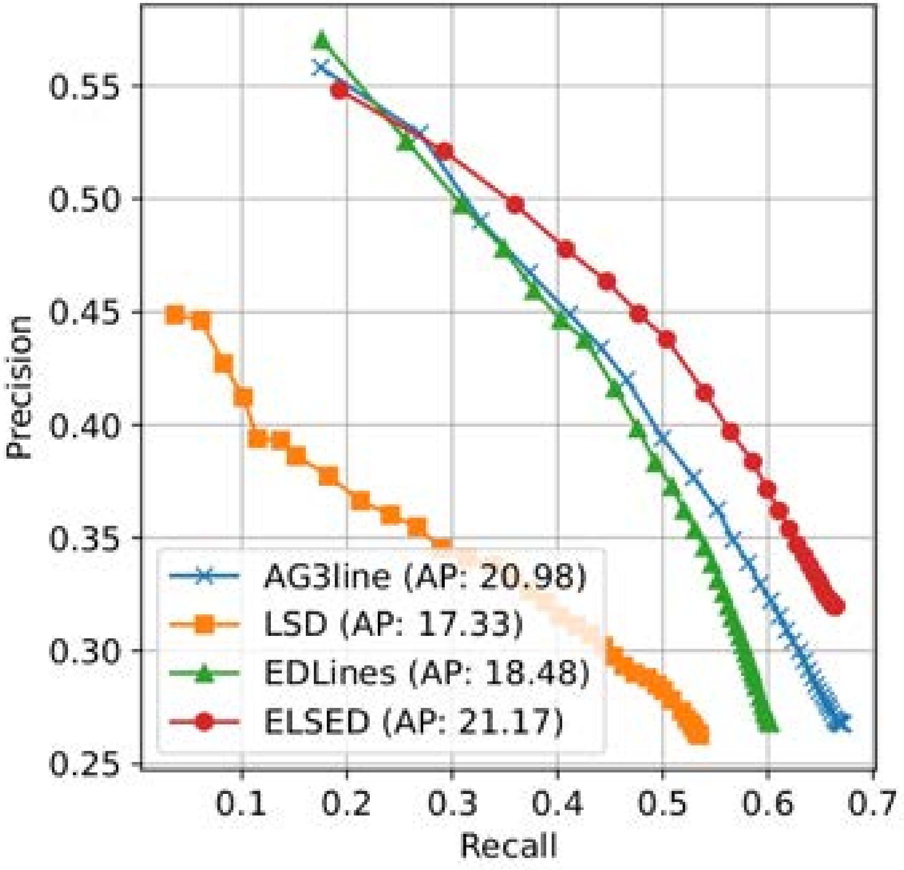
- We sort the detected segments based on the number of pixels with an **angular error** < T_{valid}
- Discarding the worst detections



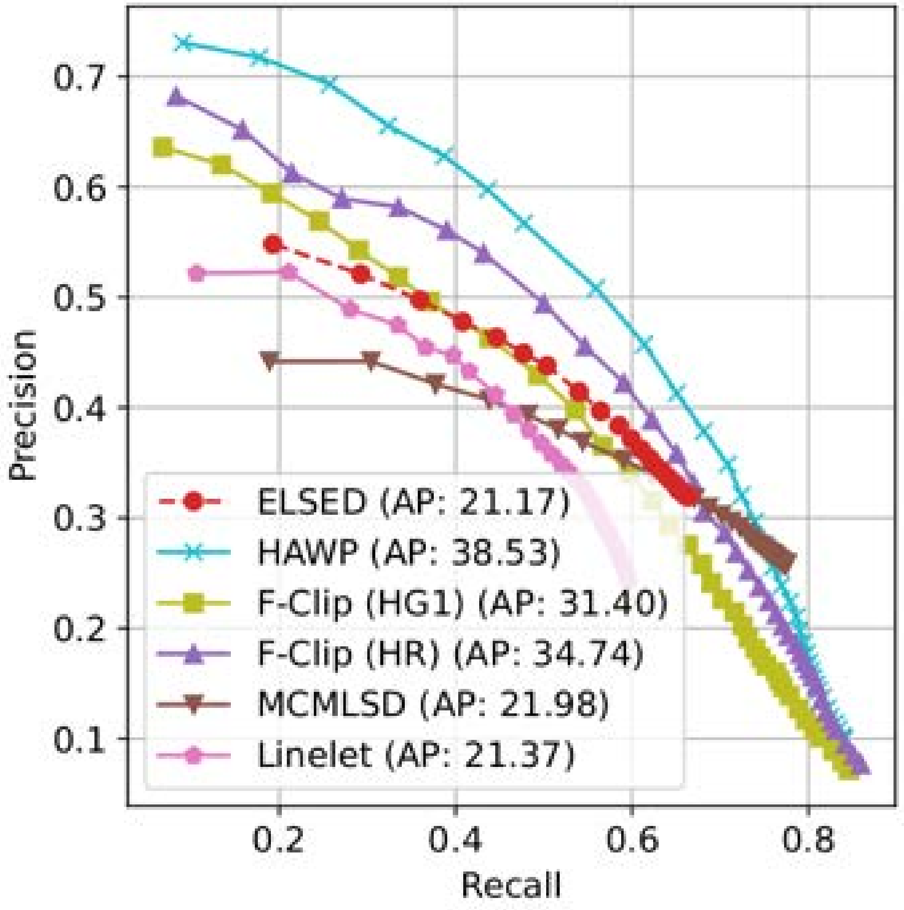
Results: Line segment detection

ELSED obtains the best results among fast detectors, being competitive with slow methods

Efficient approaches



Non-efficient approaches



Results: Repeatability

The previous metric depends on the hand-labeled segments of the scene.

$$\text{repeatability} = \frac{\sum_{i,j \in \mathtt{A}^*} \mathbf{x}_i^{\mathcal{A}} \cap_{\mathbf{x}_i^{\mathcal{A}}} \mathbf{x}_j^{\mathcal{A}|\mathcal{B}}}{\sum_i |\mathbf{x}_i^{\mathcal{A}}| + \sum_j \left|\mathbf{x}_j^{\mathcal{A}|\mathcal{B}}\right|} + \frac{\sum_{i,j \in \mathtt{B}^*} \mathbf{x}_i^{\mathcal{B}} \cap_{\mathbf{x}_i^{\mathcal{B}}} \mathbf{x}_j^{\mathcal{B}|\mathcal{A}}}{\sum_i |\mathbf{x}_i^{\mathcal{B}}| + \sum_j \left|\mathbf{x}_j^{\mathcal{B}|\mathcal{A}}\right|}$$

Viewpoint changes

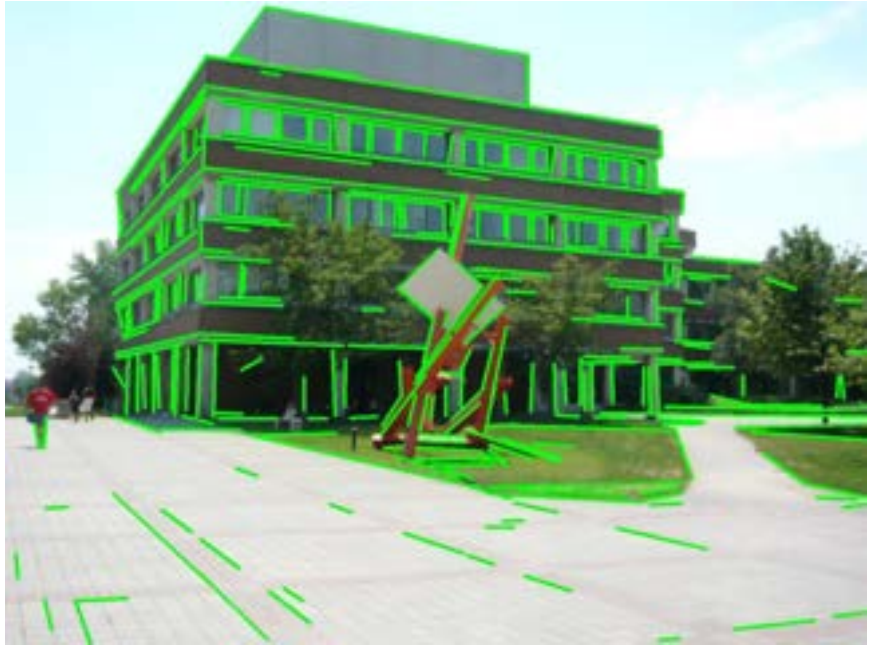
Lighting changes

Method	Length Repeatability	Num Segs. Repeatability
LSD	0.53934	0.48707
EDLines	0.54352	0.47242
AG3line	0.43582	0.40002
ELSED	**0.55928**	**0.50285**
MCMLSD	0.50322	0.39440
Linelet	0.52102	0.43427
HAWP	0.40056	0.41248
SOLD^2	0.46161	0.47957
F-Clip (HG1)	0.39784	0.43560
F-Clip (HR)	0.40969	0.43413

ELSED: The fastest detector

Moreover, ELSED is extremely fast

This also provides state-of-the-art results in the accuracy-vs-speed curve. In fact, ELSED is the fastest detector

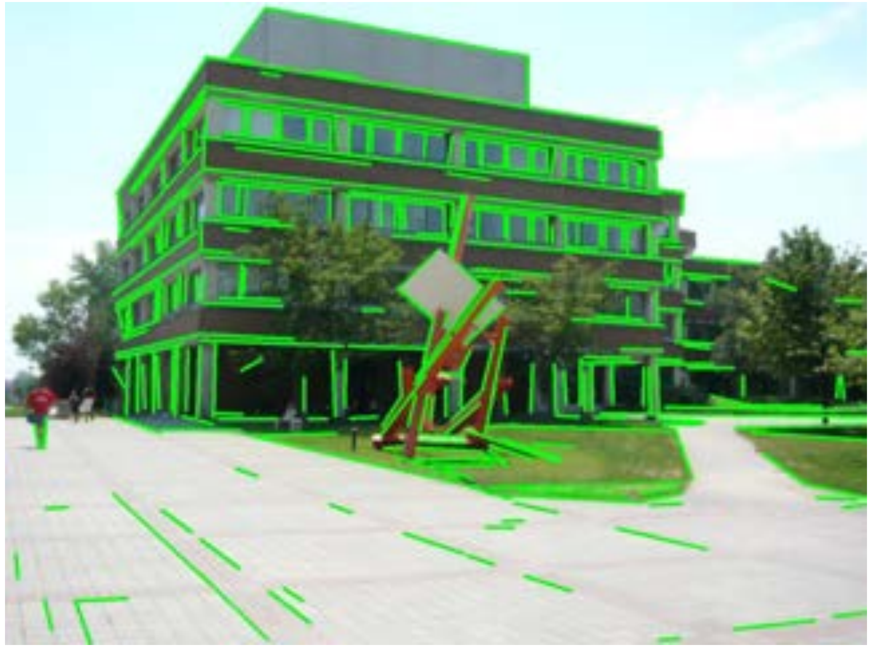


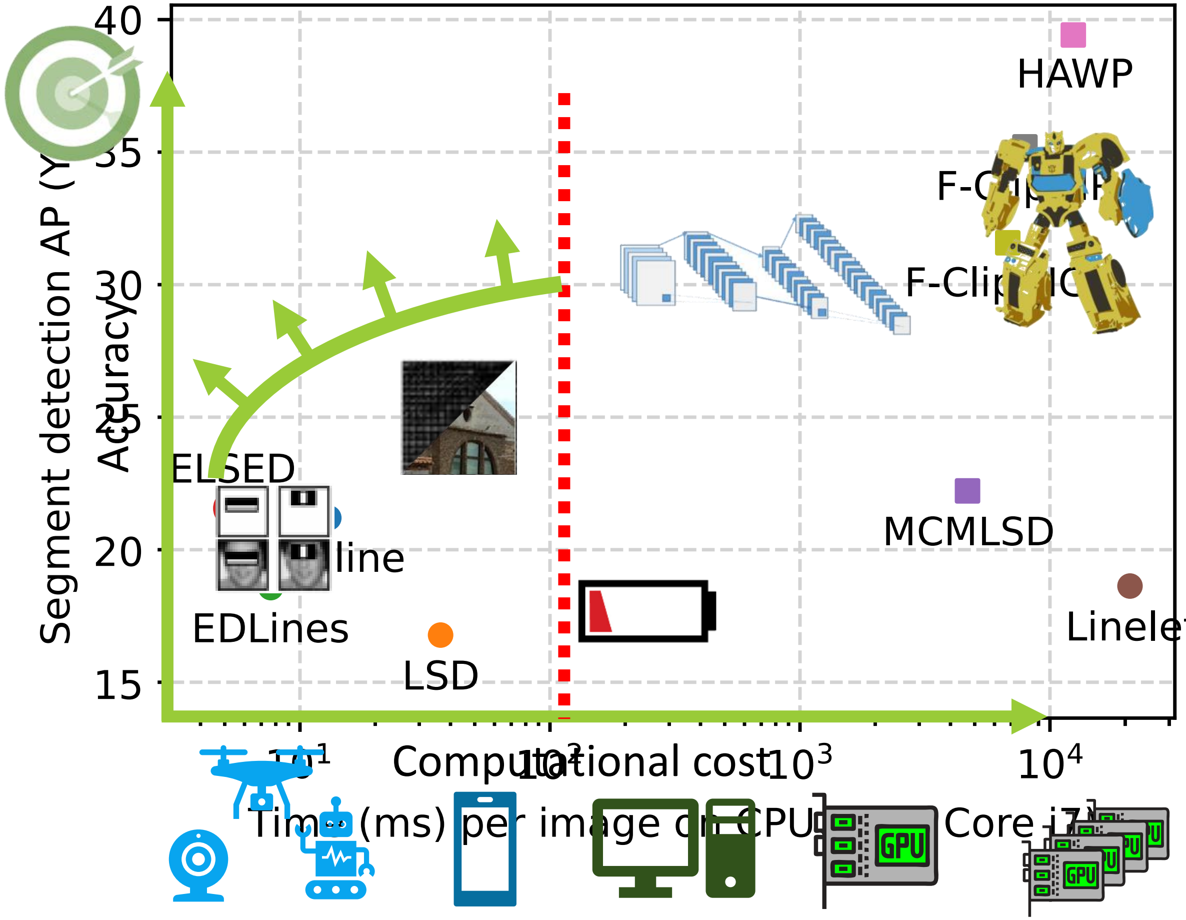
Method	Intel Core i7	Snapdragon	Exynox
LSD	36.51 (±1.60)	58.68 (±0.81)	390.91 (±0.92)
EDLines	7.64 (±0.33)	13.79 (±0.15)	65.79 (±0.16)
AG3line	13.04 (±0.76)	18.57 (±0.20)	100.54 (±0.19)
ELSED-NJ	**4.18 (±0.23)**	**8.28 (±0.03)**	**45.84 (±0.02)**
ELSED	5.38 (±0.30)	10.20 (±0.07)	59.99 (±0.16)
MCMLSD	4.68K (±1.78K)	**GeForce GTX 1050**	
Linelet	20.9K (±10.1K)		
HAWP	12.4K (±0.8K)	212.25 (±8.35)	
SOLD^2	3.17K (±0.52K)	417.72 (±6.78)	
F-Clip HR	7.94K (±0.15K)	47.39 (±1.46)	
F-Clip HG1	6.78K (±0.22K)	11.00 (±0.47)	

ELSED: The fastest detector

Moreover, ELSED is extremely fast

This also provides state-of-the-art results in the accuracy-vs-speed curve. In fact, ELSED is the fastest detector



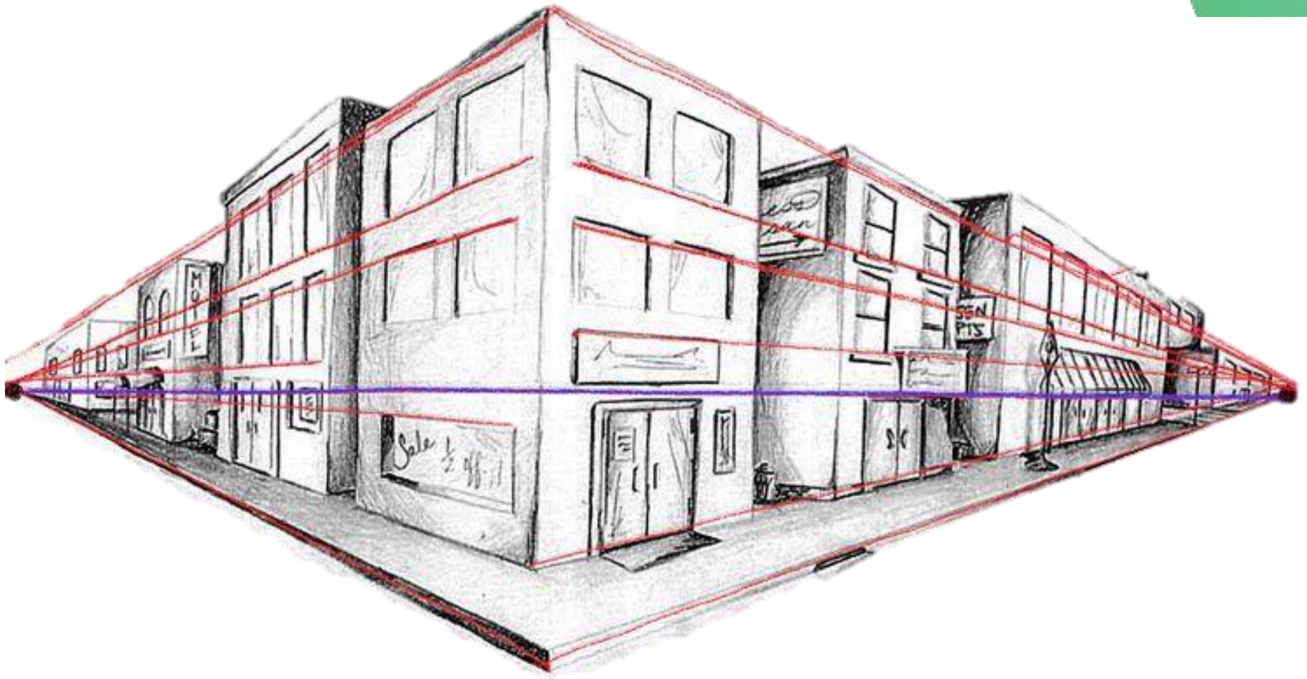


Index

0. Abstract
1. Introduction
2. Line segment detection
3. **Full line detection and vanishing point estimation**
4. Local feature description
5. Industrial results
6. Conclusions

Vanishing Point Estimation: Definition

"A vanishing point is a point on the image plane where the two-dimensional perspective projections of mutually parallel lines in three-dimensional space appear to converge"

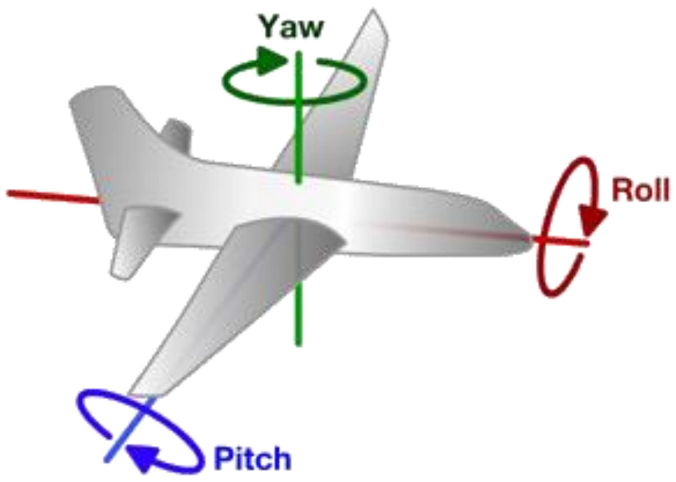


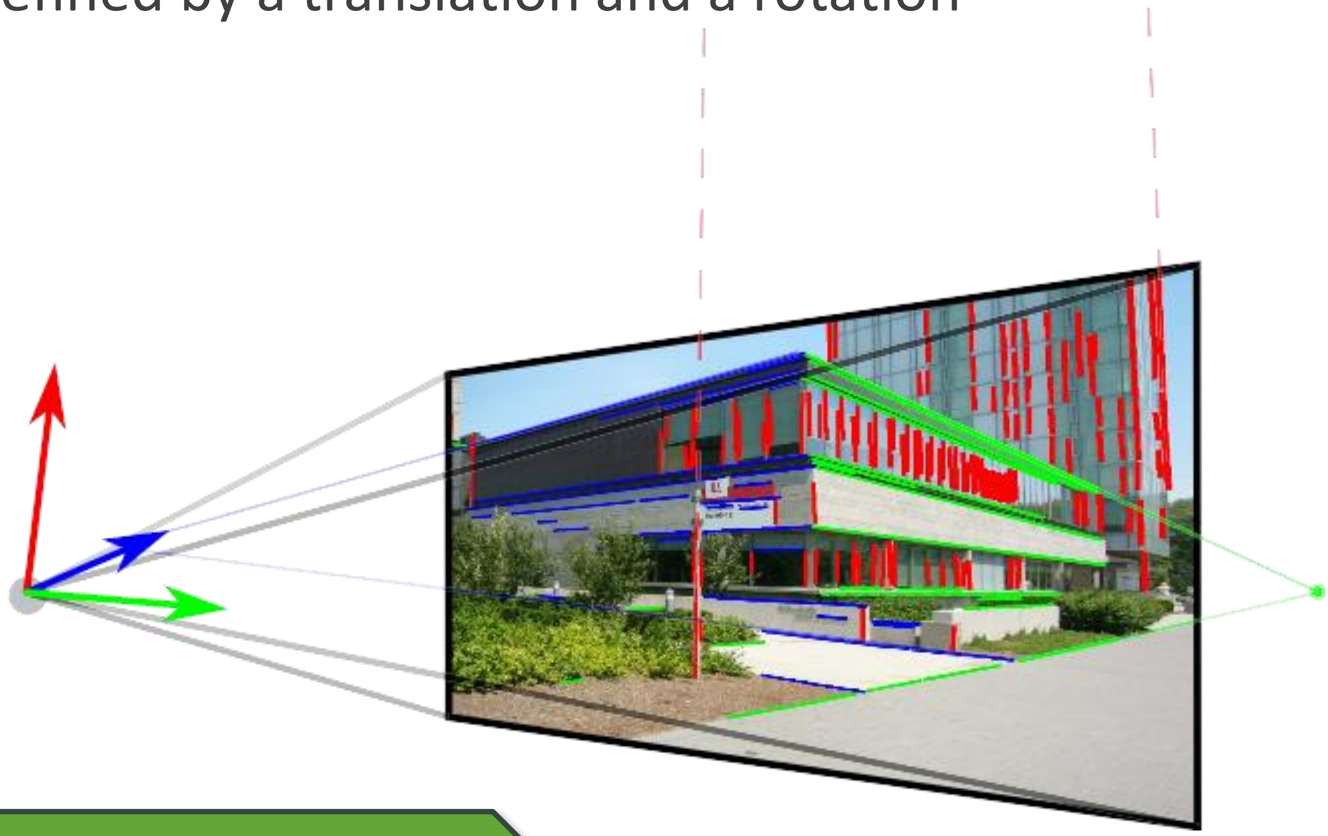
pinterest.es/pin/310044755583385782

They arise in the so-called:
Manhattan World

Vanishing Point Estimation

The camera position in the world is defined by a translation and a rotation

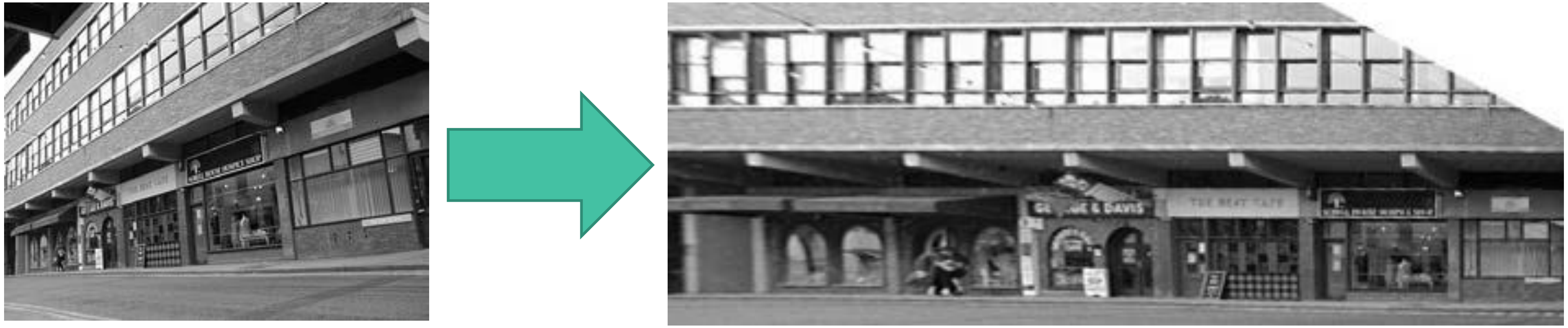




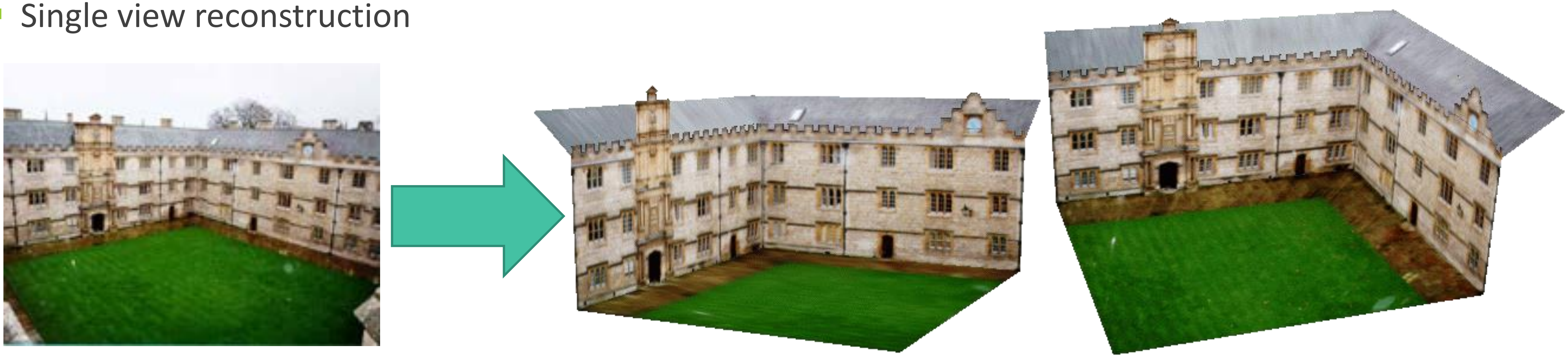
The rotation can be directly estimated from the vanishing points

Vanishing Point Estimation: Other applications

- Plane rectification via partial inertial parameters

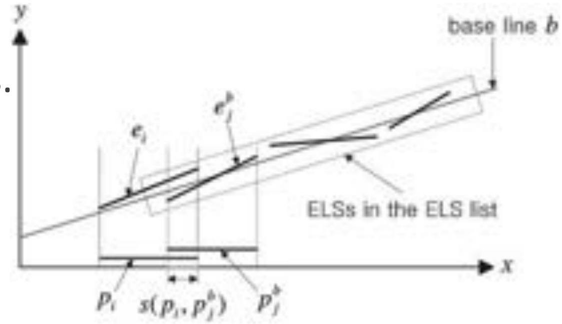


- Single view reconstruction



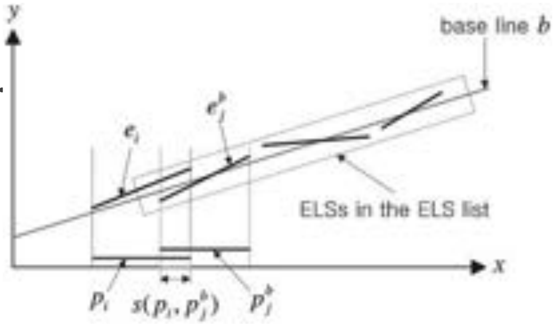
Previous work: Grouping lines approaches

- Heuristic based:
  - (Jang 2002) Fast line segment grouping method for finding globally more favorable line segments.
  - (Zuo, 2017) Robust visual SLAM with point and line features
    - Group two segments if some distances from their middle and end points are small.
  - (Yang, 2017) Direct monocular odometry using points and lines
    - Organize the candidates into buckets with similar middle point locations and orientations.



Previous work: Grouping lines approaches

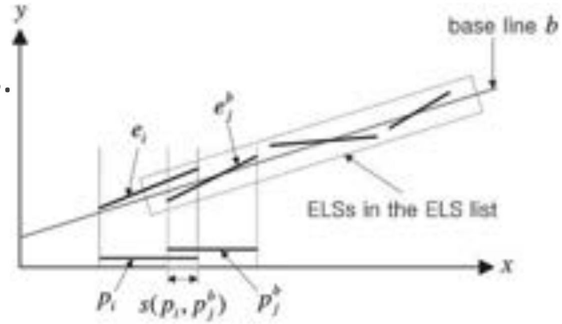
- Heuristic based:
  - (Jang 2002) Fast line segment grouping method for finding globally more favorable line segments.
  - (Zuo, 2017) Robust visual SLAM with point and line features
    - Group two segments if some distances from their middle and end points are small.
  - (Yang, 2017) Direct monocular odometry using points and lines
    - Organize the candidates into buckets with similar middle point locations and orientations.



- (Bandera, 2006) Mean shift based clustering of Hough domain for fast line segment detection.

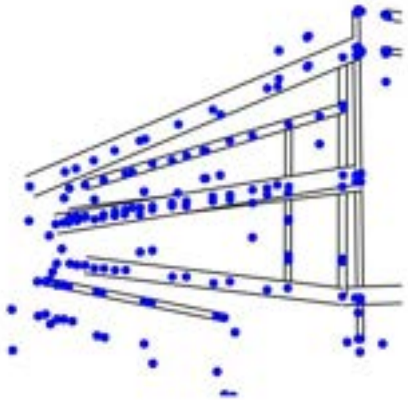
Previous work: Grouping lines approaches

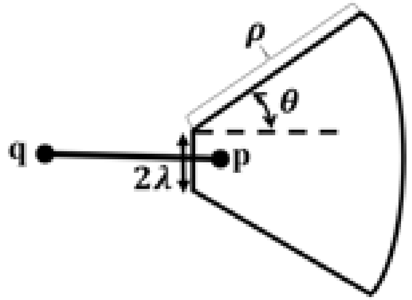
- Heuristic based:
  - (Jang 2002) Fast line segment grouping method for finding globally more favorable line segments.
  - (Zuo, 2017) Robust visual SLAM with point and line features
    - Group two segments if some distances from their middle and end points are small.
  - (Yang, 2017) Direct monocular odometry using points and lines
    - Organize the candidates into buckets with similar middle point locations and orientations.



- (Bandera, 2006) Mean shift based clustering of Hough domain for fast line segment detection.

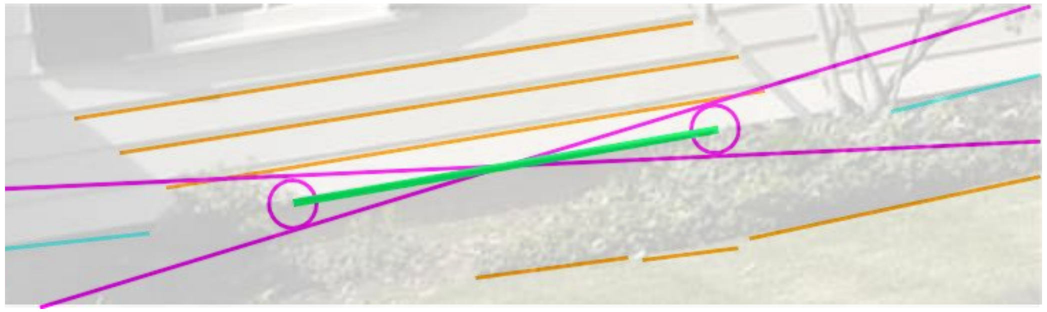
- Probabilistic models use the a contrario methodology:
  - (Lezama, 2014) Finding Vanishing Points via Point Alignments in Image Primal and Dual Domains
  - (Rajaei, 2018) Gestaltic grouping of line segments

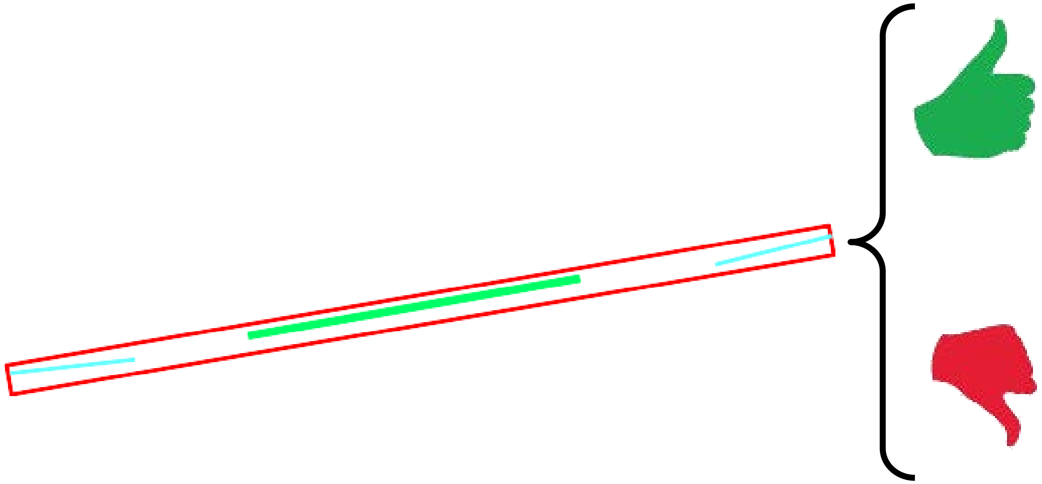


FSG: A statistical approach to line detection via fast segments grouping

LINE HYPOTHESIS GENERATION



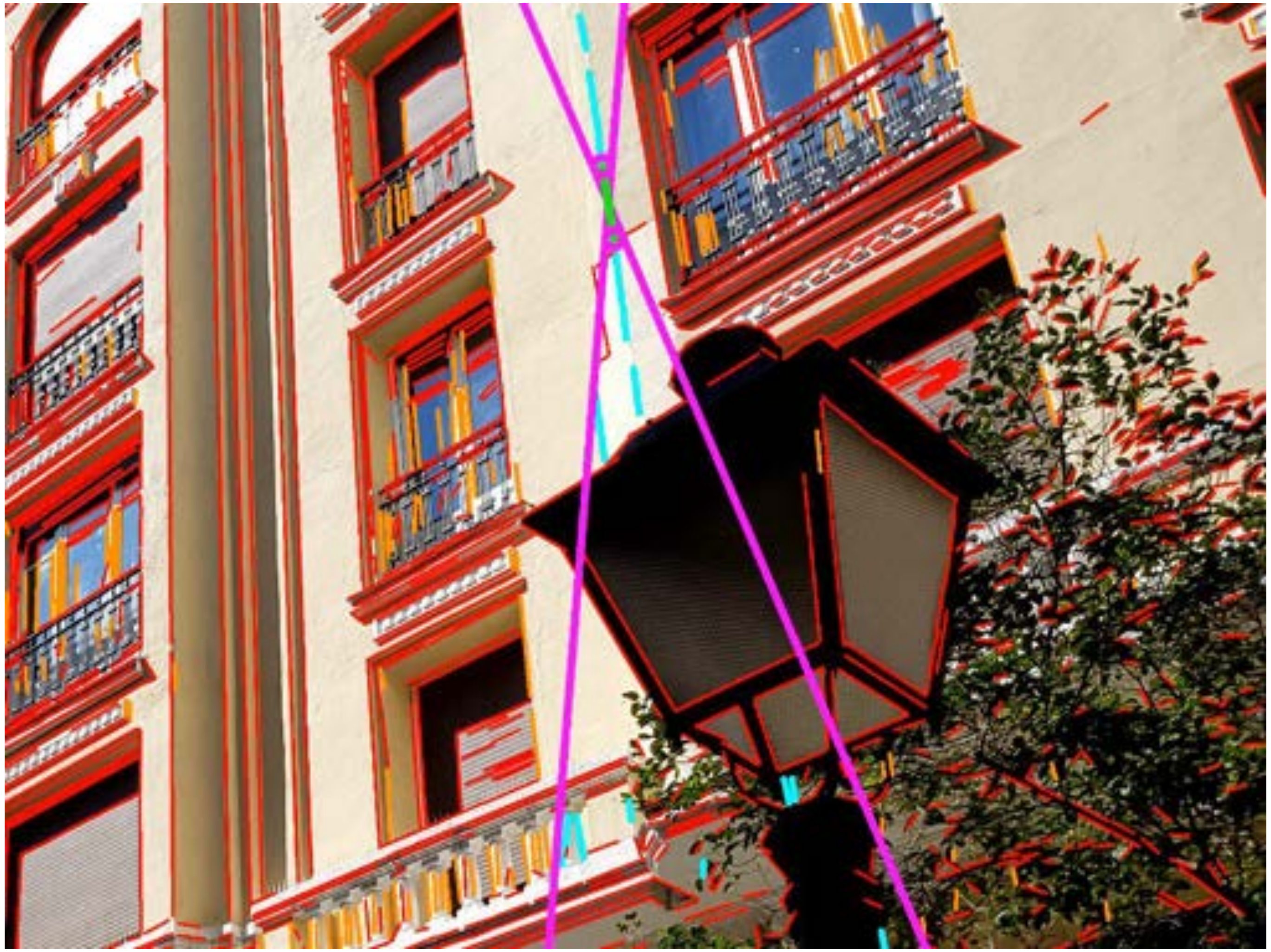
LINE VALIDATION CRITERIA



Suárez, I., Muñoz, E., Buenaposada, J. M., & Baumela, L. (2018, October). FSG: A statistical approach to line detection via fast segments grouping. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 97-102). IEEE.

FSG: Line hypothesis generation

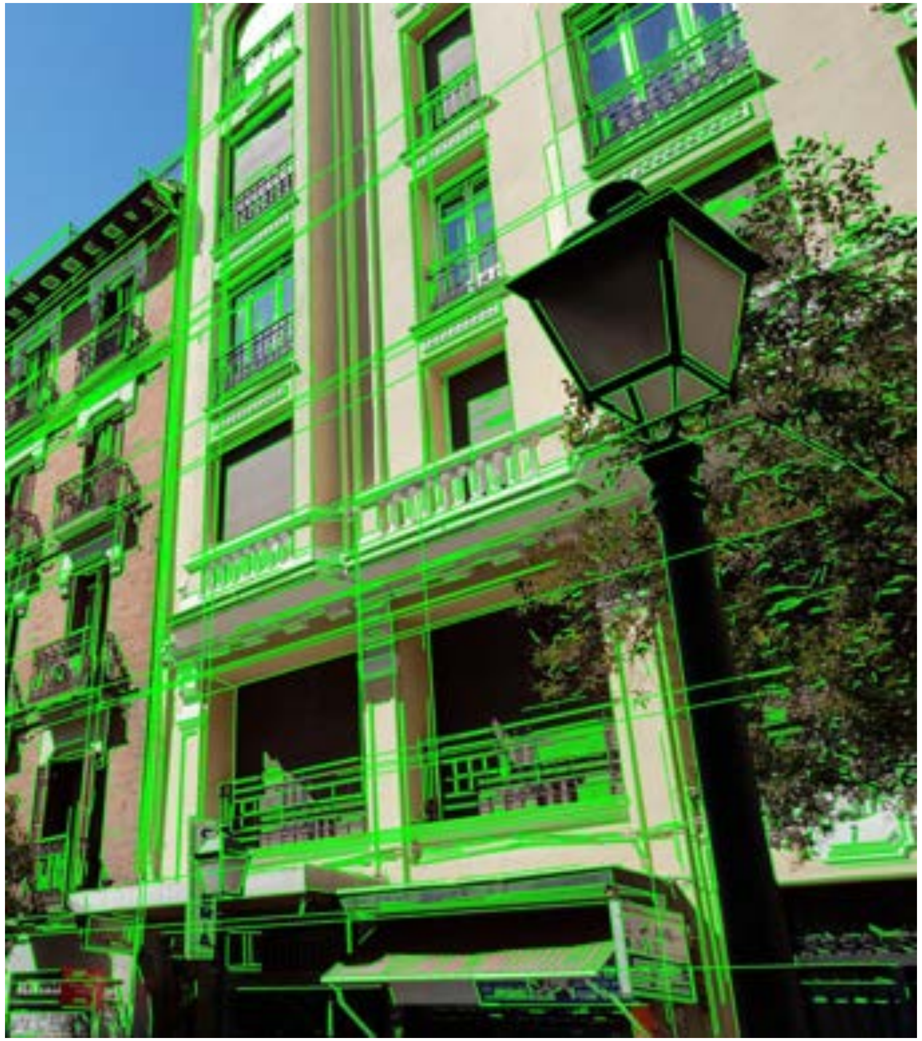
FSG: Line hypothesis generation



FSG: Line hypothesis generation

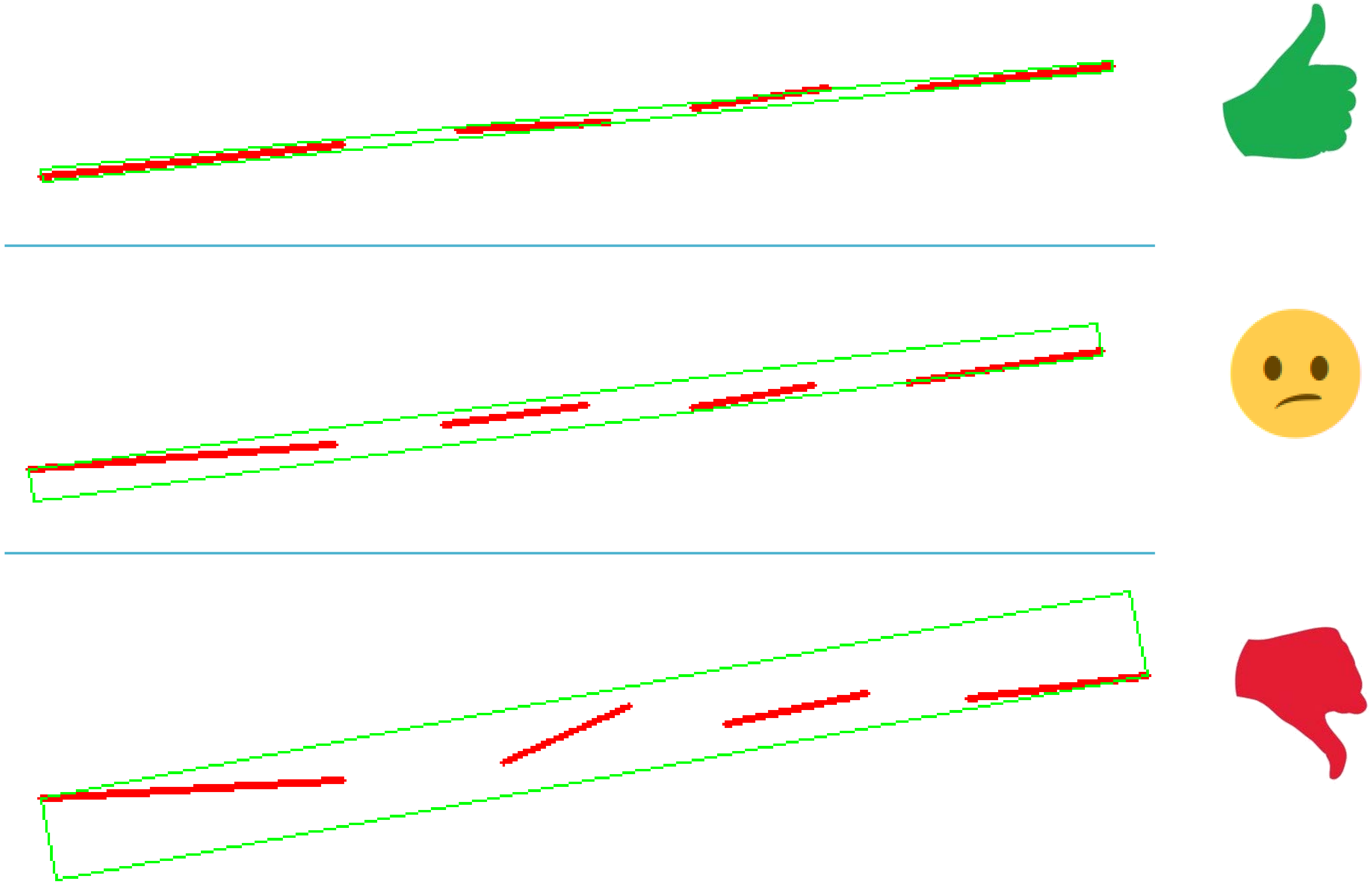
Initial line segments (VonGioi, 2010)

Resulting lines



FSG: Statistical Validator

- We propose a statistical validator to check whether a group of segments is well aligned



The validation criterion should consider

Segments length

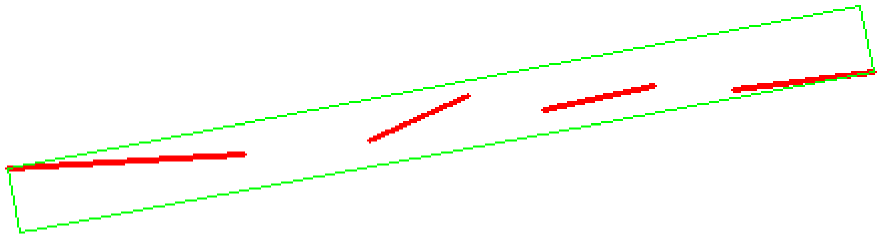
Closeness

Well-aligned

FSG: Statistical Validator

- In an image with s segments. We evaluate the probability that a set of c segments from H fall into B.

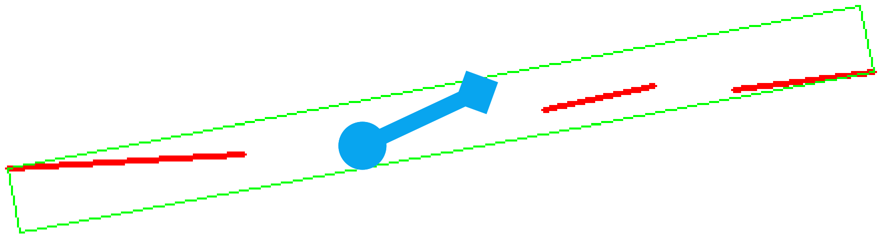
$$\mathbb{E}_H(\mathcal{S}, \mathcal{C}_s) = \binom{s}{c} P[\mathcal{C}_h \in B],$$



FSG: Statistical Validator

- In an image with s segments. We evaluate the probability that a set of c segments from H fall into B.

$$\mathbb{E}_H(\mathcal{S}, \mathcal{C}_s) = \binom{s}{c} P[\mathcal{C}_h \in B],$$



- Product of the probability of each segment to fall into B:

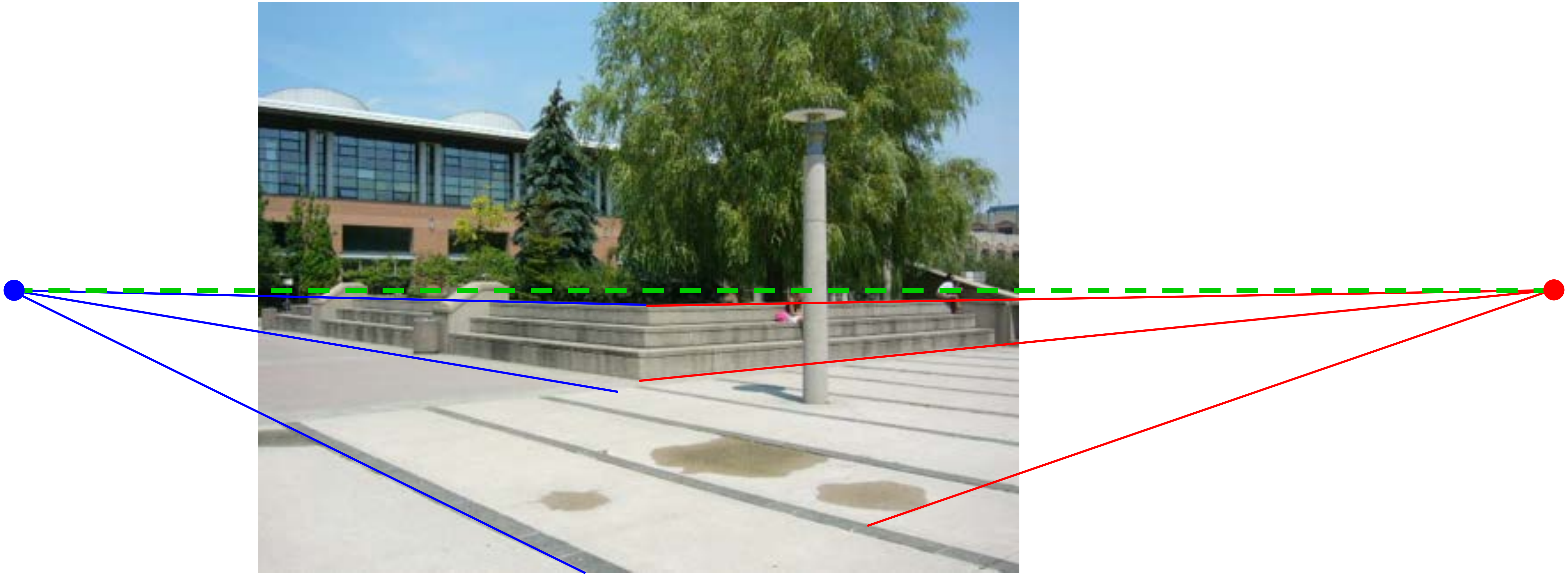
$$P[\mathcal{C}_h \in B] = \prod_{i=0}^{s} \underbrace{P[b(s_i)]}_{\text{First endpoint}} \cdot \underbrace{P[e(s_i)|b(s_i), l(s_i)]}_{\text{Second endpoint}},$$

Robust Vanishing point from lines

- Accurate vanishing point estimator (Lezama, 2014)
- Fast vanishing point estimator (based on RANSAC) (Zhang, 2016)

Evaluation of horizon line estimation

We evaluate the estimated vanishing points with the standard metric Horizon line error:

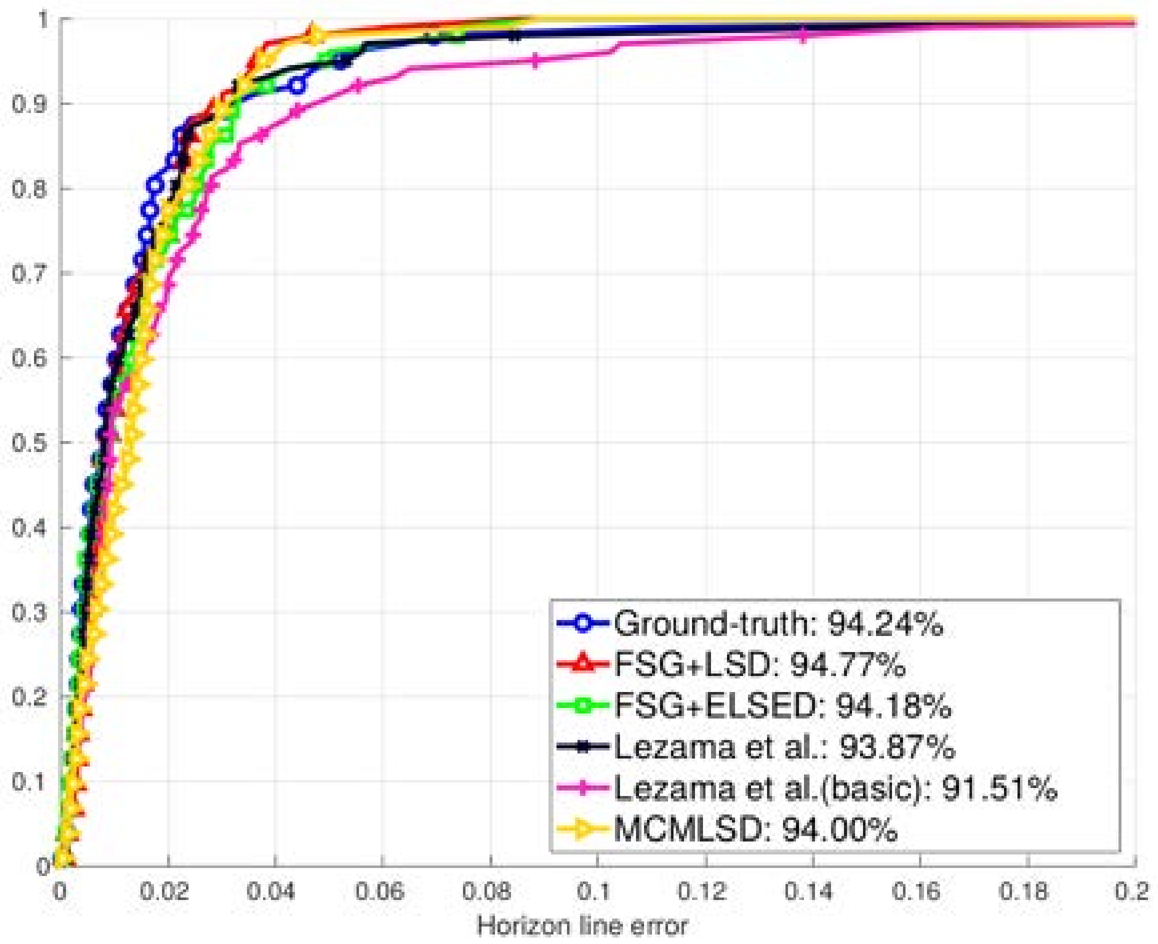


Evaluation of horizon line estimation

We evaluate the estimated vanishing points with the standard metric Horizon line error:

Quantitative Results

- Segments Grouped at **6 ms/frame** (Intel core i7) with state-of-the-art accuracy
- Cumulative error distribution:

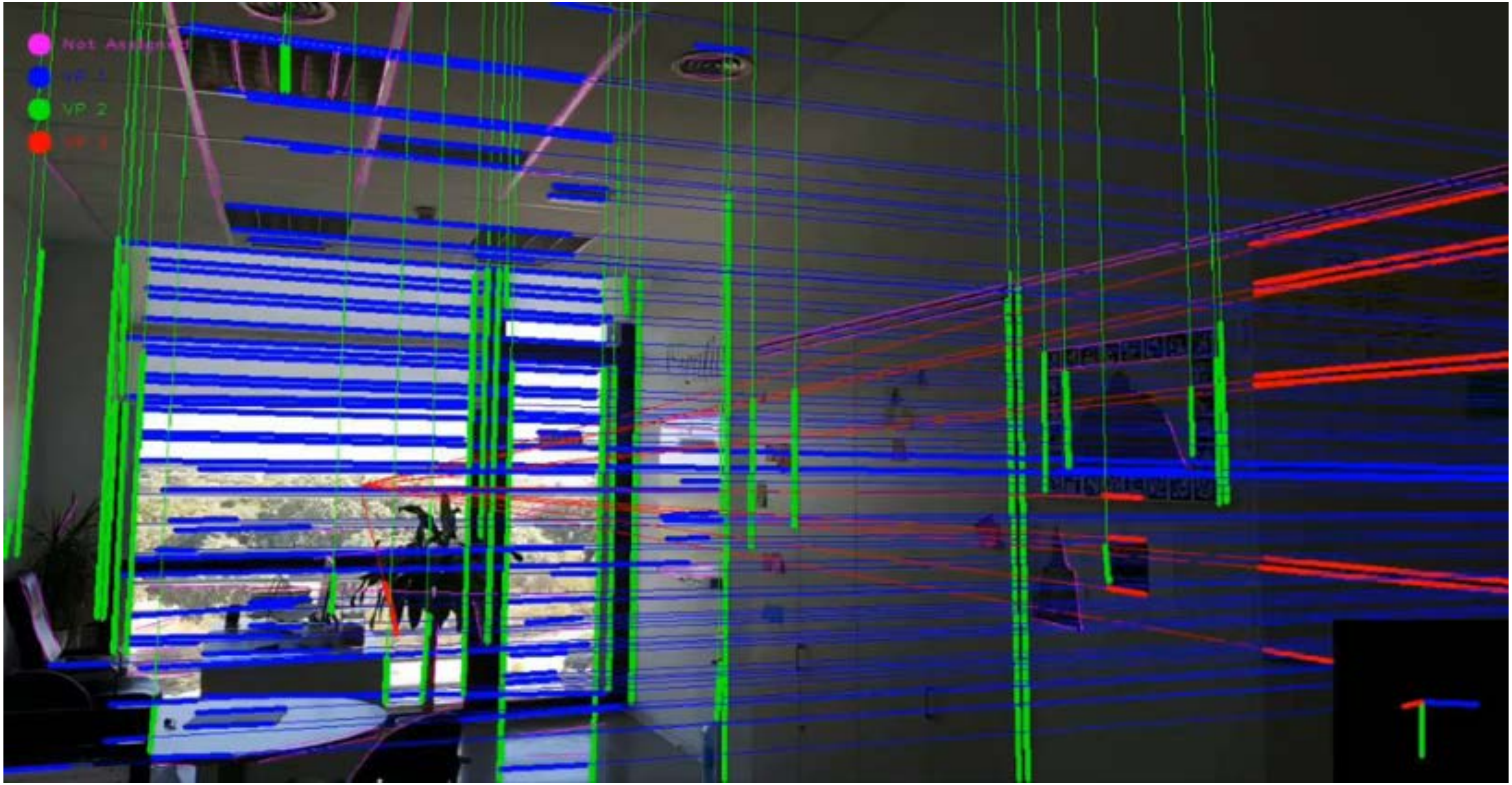


Local Segments Detectors	
LSD	**ELSED**
36.51 ms	**5.38 ms**

Segments Grouping Algorithms	
FSG	(Lezama 2014)
5.89 ms	14961 ms

Global Segment Detectors	
PPTH	MCMLSD
21.63 ms	4686 ms

Qualitative results (Smartphone)



Index

0. Abstract
1. Introduction
2. Line segment detection
3. Full line detection and vanishing point estimation
4. **Local feature description**
5. Industrial results
6. Conclusions

Previous work: SIFT Descriptor

- SIFT (Lowe, 1999) is the most widely used descriptor:

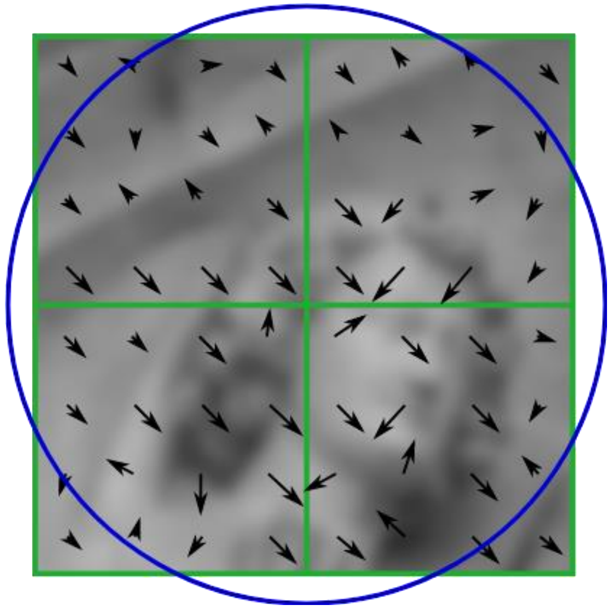


Image gradients

Previous work: SIFT Descriptor

- Uses the histograms of gradients in a fixed grid:

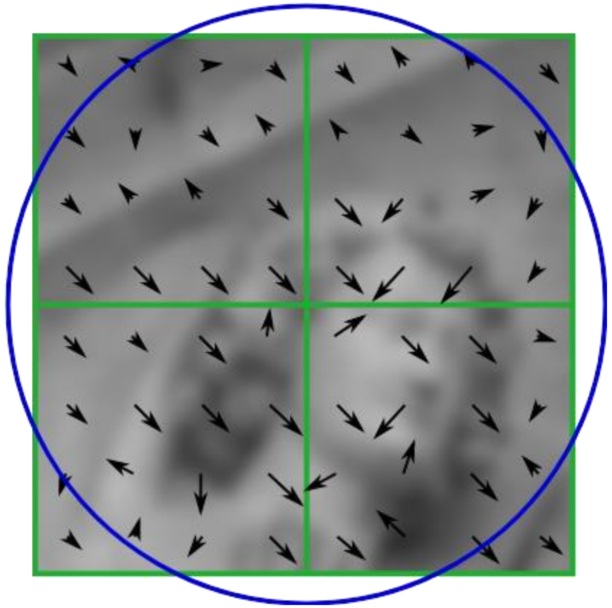
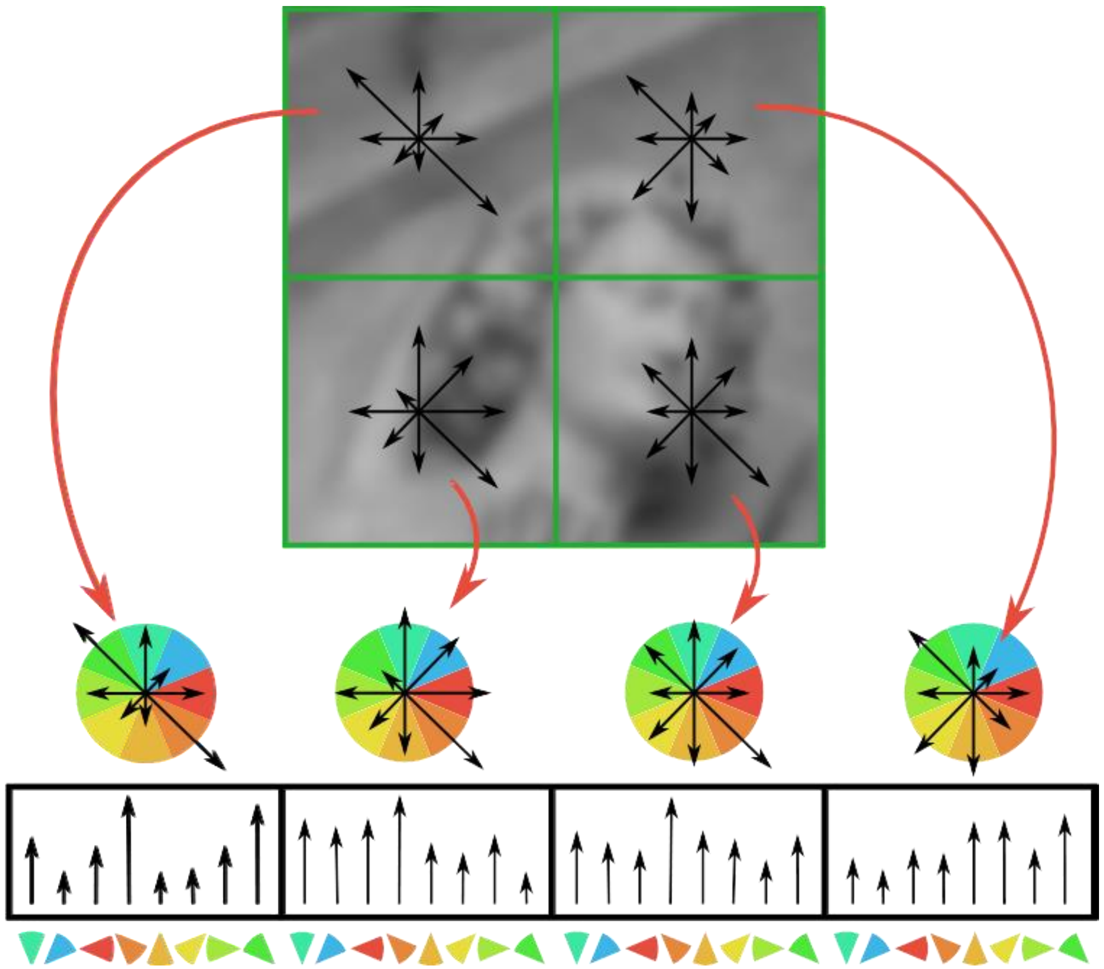


Image gradients



Previous work: SIFT Descriptor

- Uses the histograms of gradients in a fixed grid:

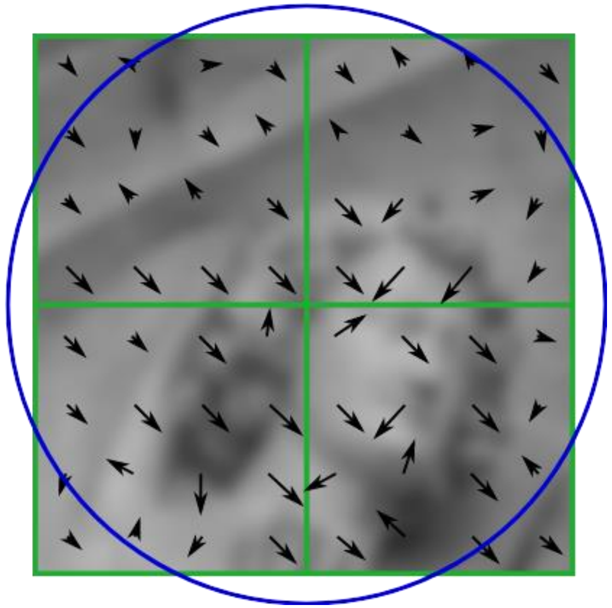
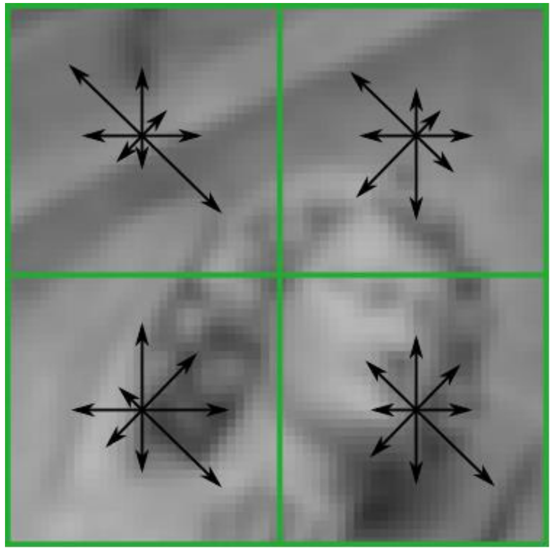


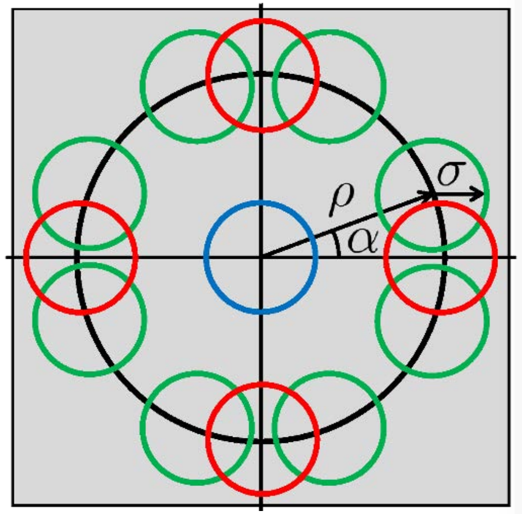
Image gradients



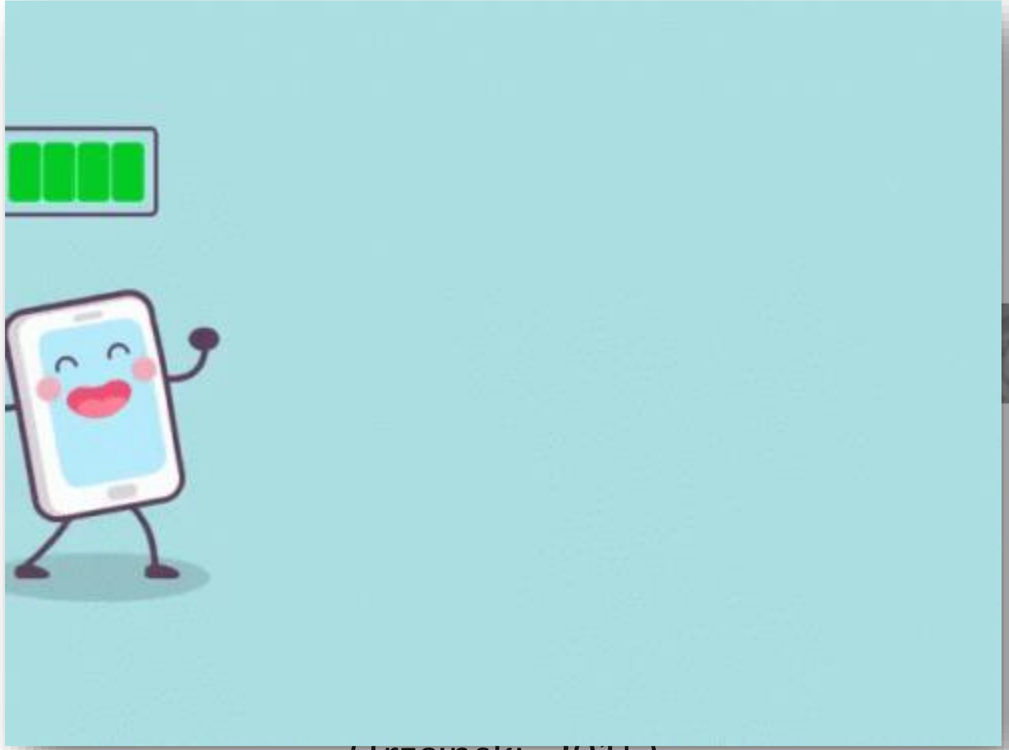
Gradient orientation histograms:
- **Fixed grid**: 2 x 2
- **Fixed scale**: The cell size
- **Dense**: Uses all patch pixels

Previous work: Inefficient learnt descriptors

Instead of fixing the grid, the scale and the gradient as information, this can be learnt:

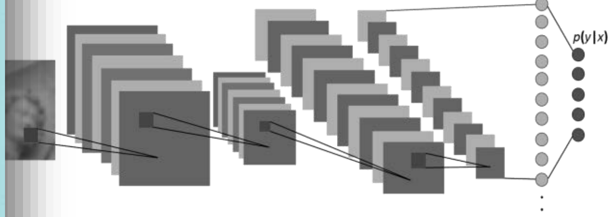


DLCO – VGG
(Simonyan, 2014)



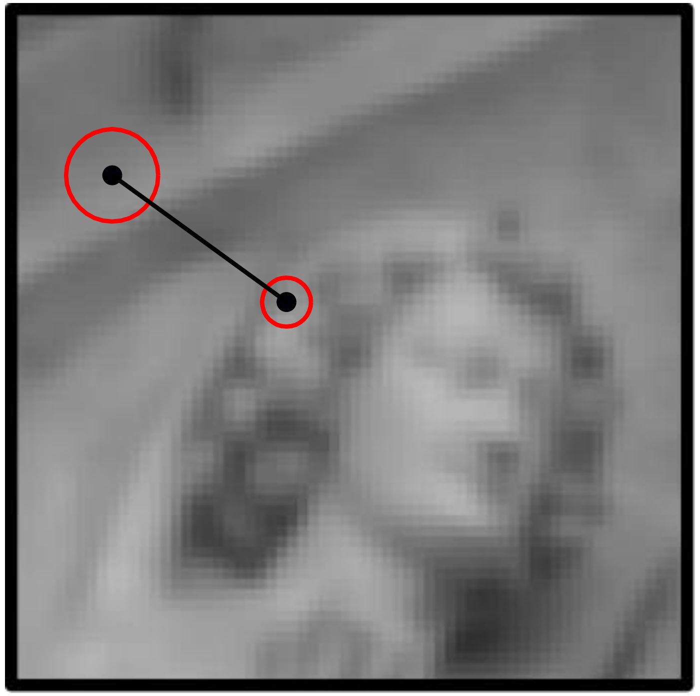
(Trzcinski, 2015)

https://tenor.com/view/battery-phone-drain-gif-11086262



Deep Learning descriptors
Tfeat, L2Net, DOAP,
HardNet, SOSNet, CDbin

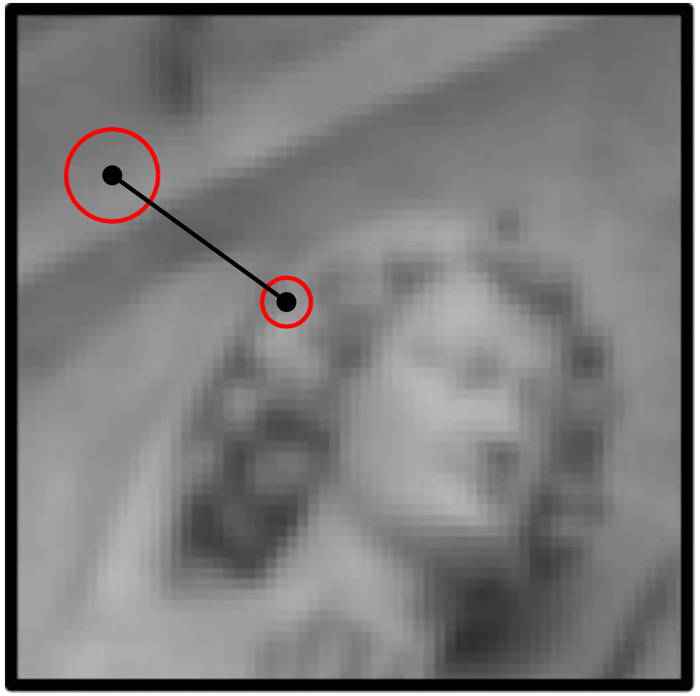
Previous work: Efficient descriptors



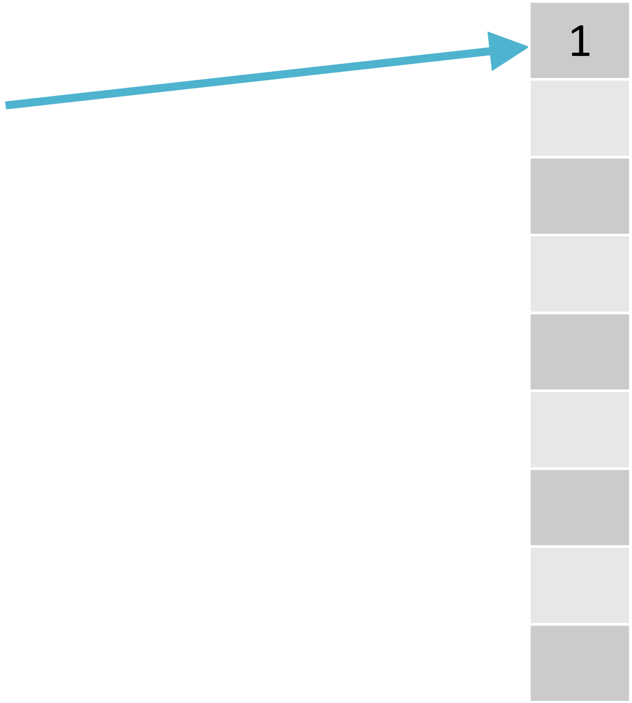
Efficient alternatives to SIFT are based in **approximate gradient by comparing pixel intensities**

- Sparse approach → Fast

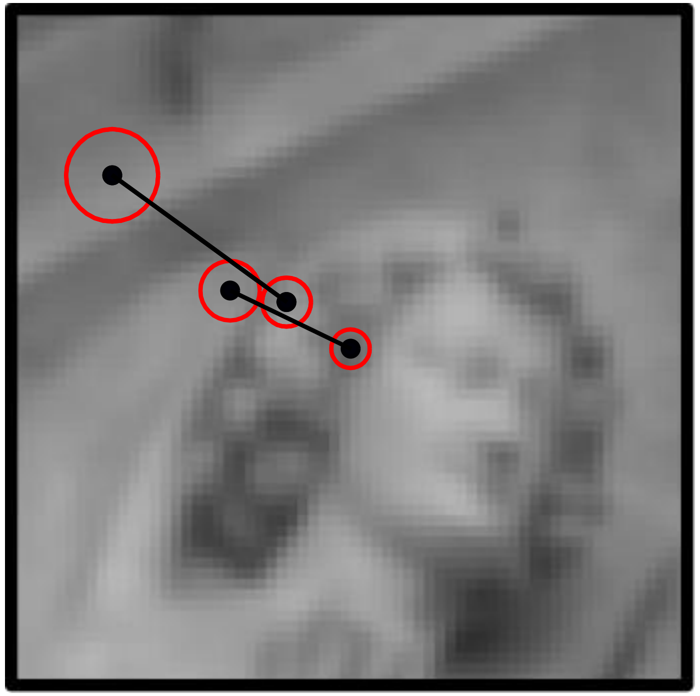
Previous work: Efficient descriptors



Computed descriptor



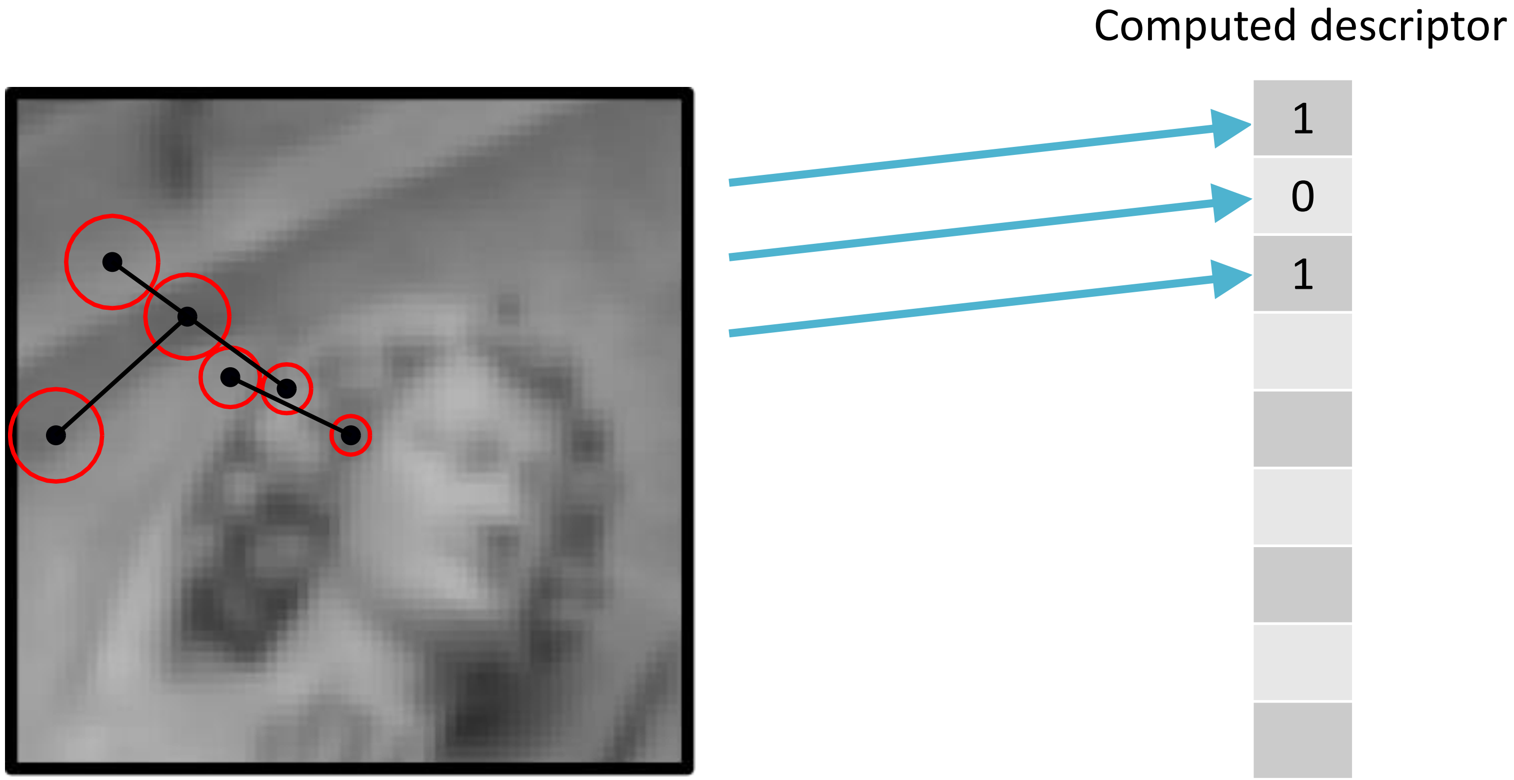
Previous work: Efficient descriptors



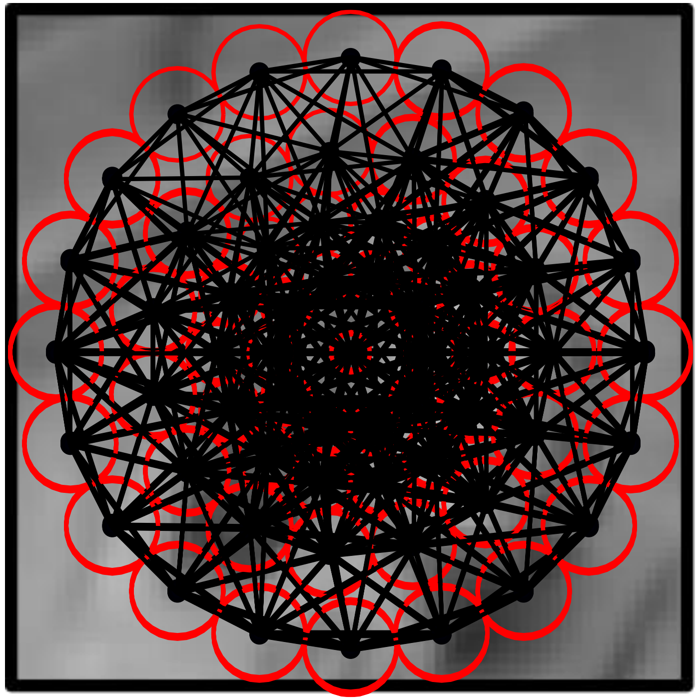
Computed descriptor

1
0

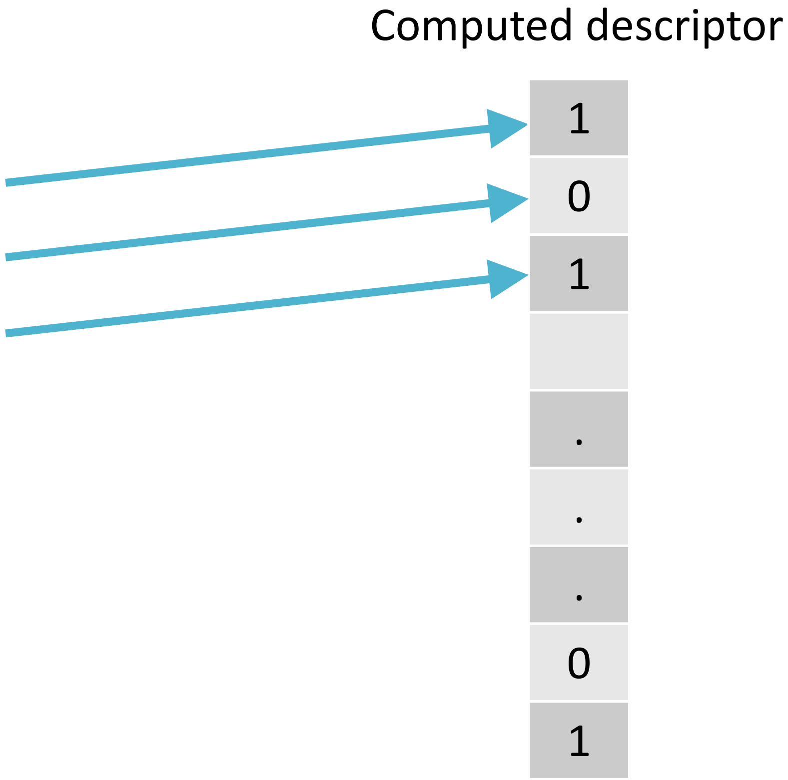
Previous work: Efficient descriptors



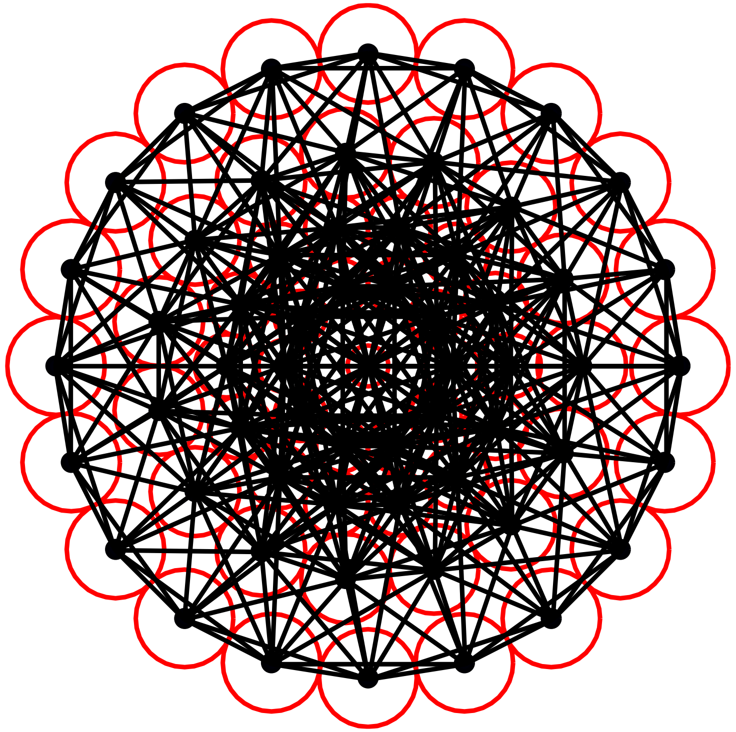
Previous work: Efficient descriptors



BRISK



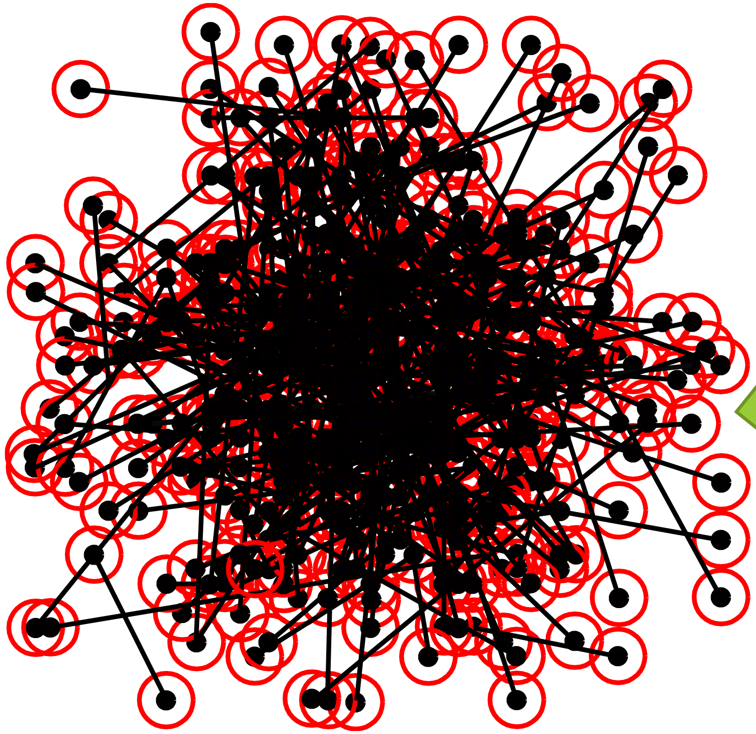
Efficient descriptors comparison



Pattern	**Grid**	BRISK	
	Random		
	Learnt		

Leutenegger, S., Chli, M., & Siegwart, R. (2011). BRISK: Binary robust invariant scalable keypoints. In ICCV (pp. 2548-2555).

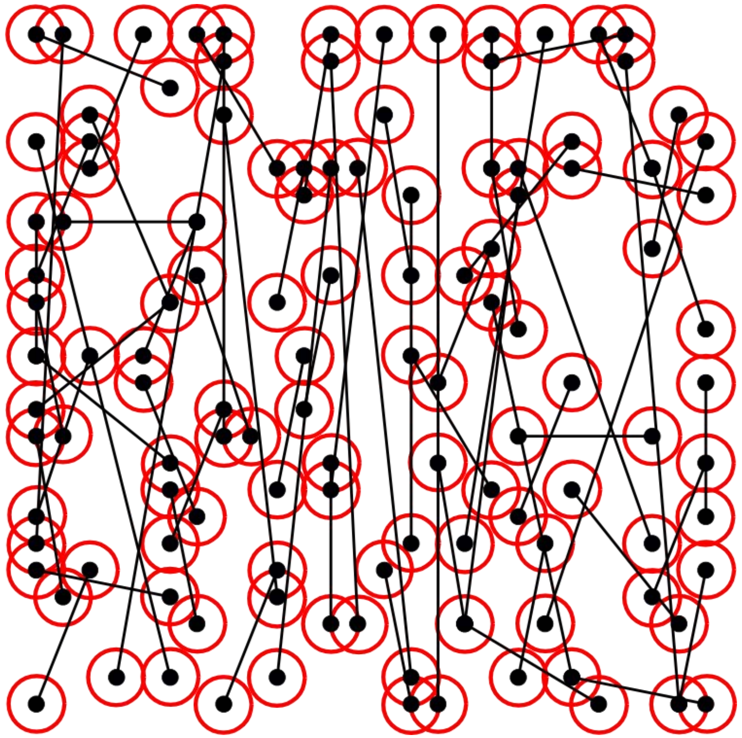
Efficient descriptors comparison



Pattern			
	Grid	BRISK	
	Random	BRIEF	
	Learnt		

Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010, September). Brief: Binary robust independent elementary features. In ECCV (pp. 778-792).

Efficient descriptors comparison

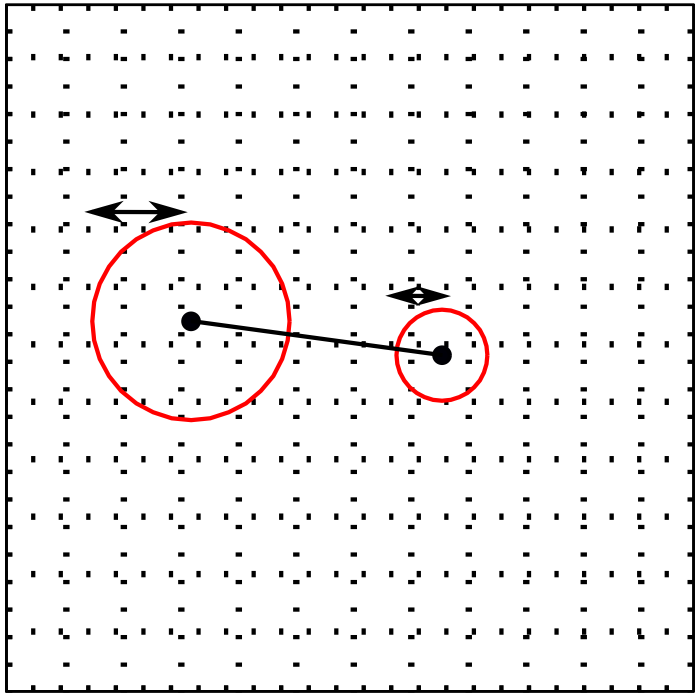


Pattern	Grid	BRISK	
	Random	BRIEF	
	Learnt	ORB	

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. R. (2011, November). ORB: An efficient alternative to SIFT or SURF. In ICCV (Vol. 11, No. 1, p. 2).

Efficient descriptors: Measurement function

The proposed measurement functions learn not only the description pattern but also the description scale



		Measurement Scale	
		Fixed	**Learnt**
Pattern	**Grid**	BRISK	
	Random	BRIEF	
	Learnt	ORB	

Efficient descriptors: Measurement function

The **Thresholded Average Box measure** that learns both:

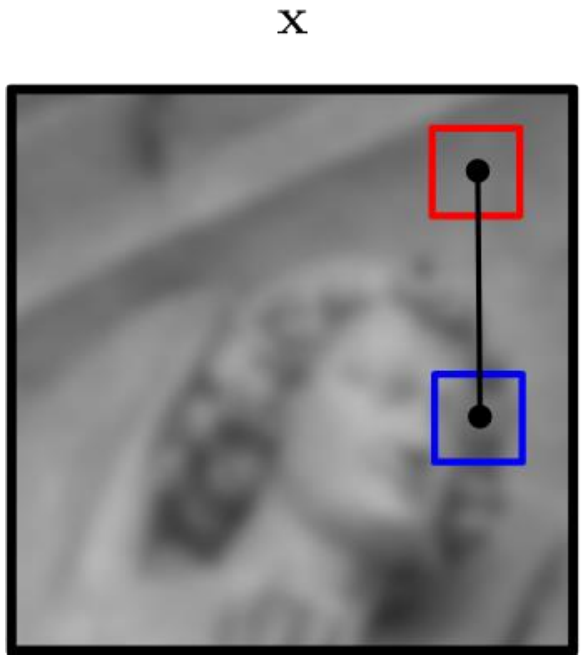
- Descriptor Pattern
- Descriptor Measurements Scale

		Measurement Scale	
		Fixed	**Learnt**
Pattern	**Grid**	BRISK	
	Random	BRIEF	
	Learnt	ORB	**BELID**

Suárez, I., Sfeir, G., Buenaposada, J. M., & Baumela, L. (2019, July). BELID: Boosted efficient local image descriptor. In *Iberian Conference on Pattern Recognition and Image Analysis* (pp. 449-460). Springer, Cham.

Thresholded Average Box measure

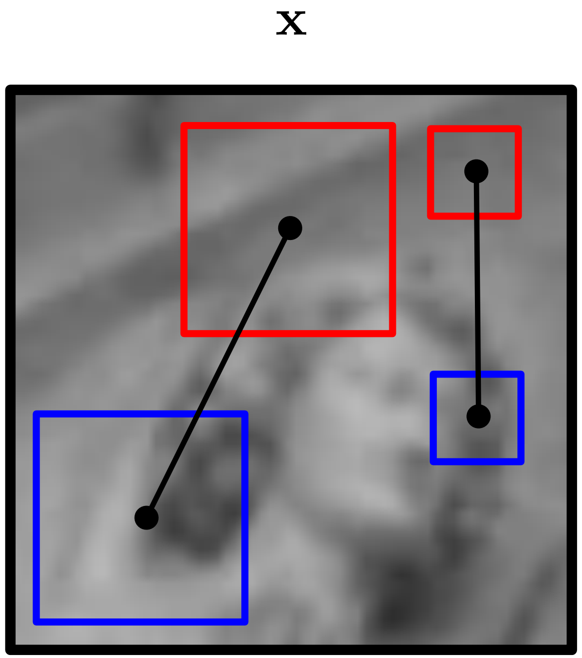
The measurement function is the **difference of the average gray level in two boxes**



Patch to describe

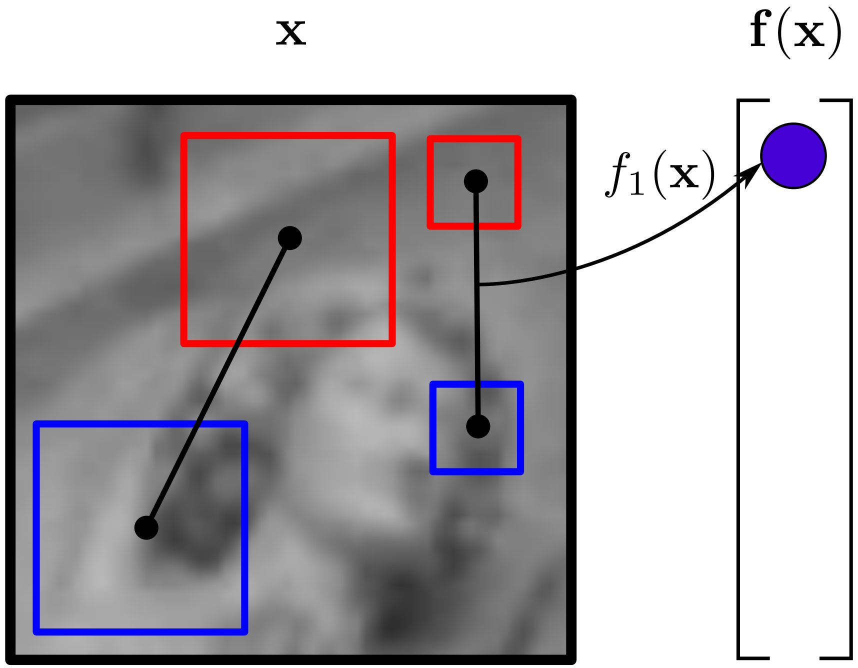
Thresholded Average Box measure

The measurement function is the **difference of the average gray level in two boxes**



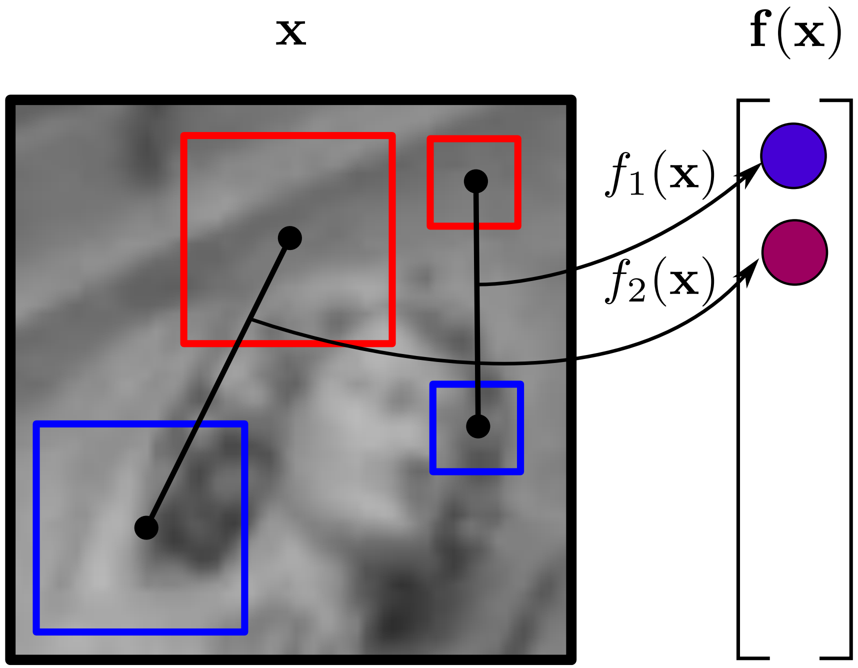
Thresholded Average Box measure

The measurement function is the **difference of the average gray level in two boxes**



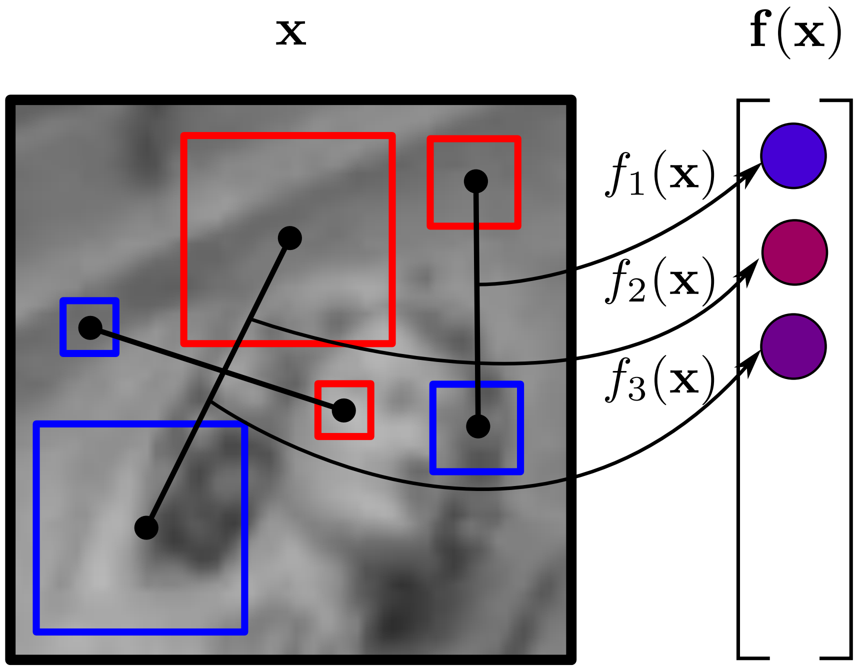
Thresholded Average Box measure

The measurement function is the **difference of the average gray level in two boxes**



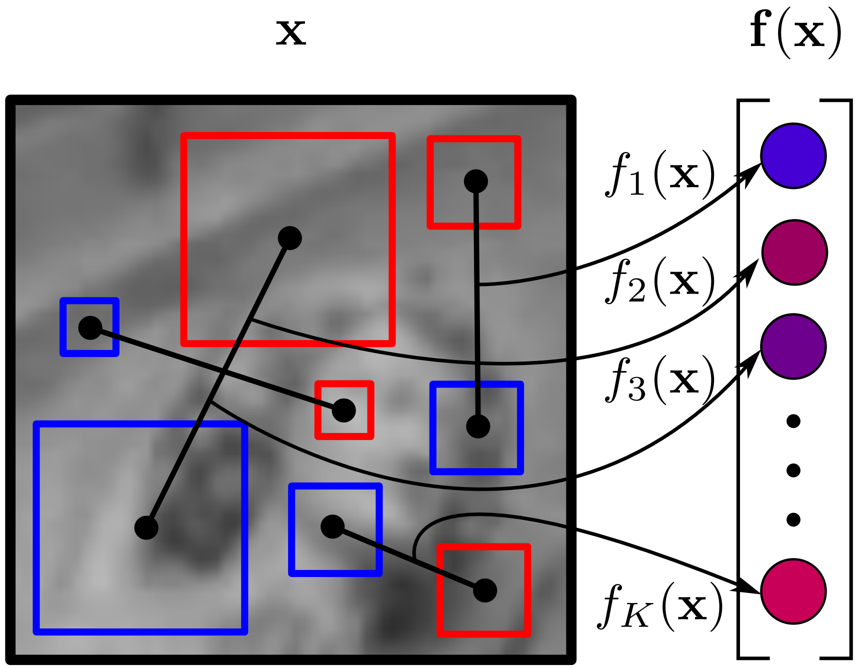
Thresholded Average Box measure

The measurement function is the **difference of the average gray level in two boxes**



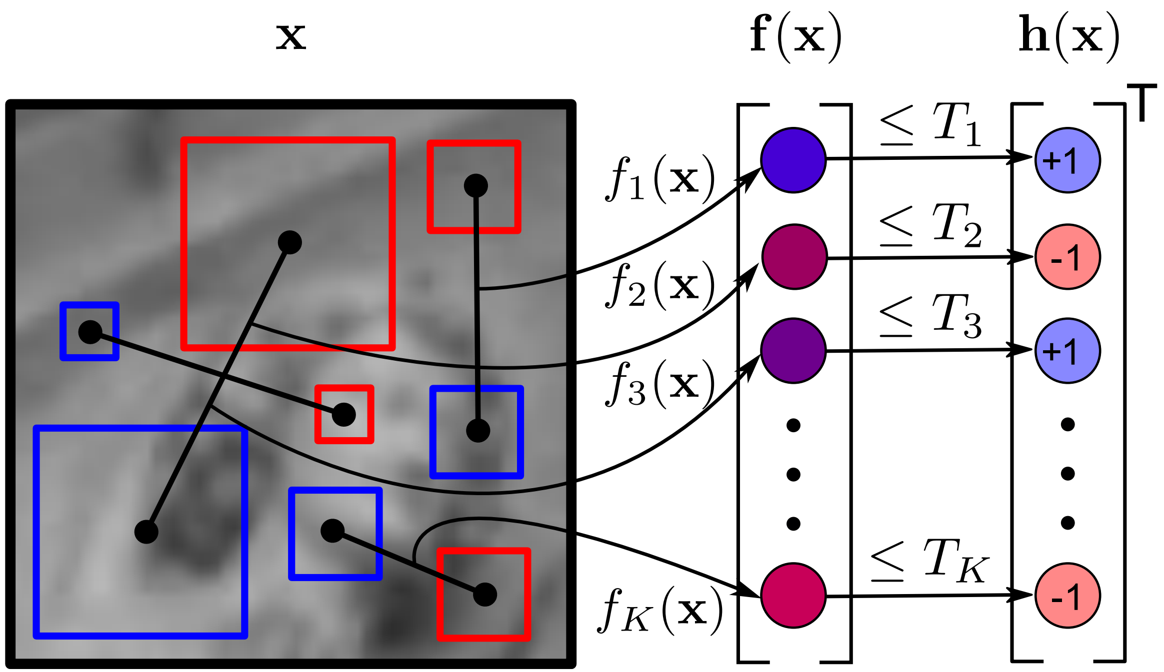
Thresholded Average Box measure

The measurement function is the **difference of the average gray level in two boxes**



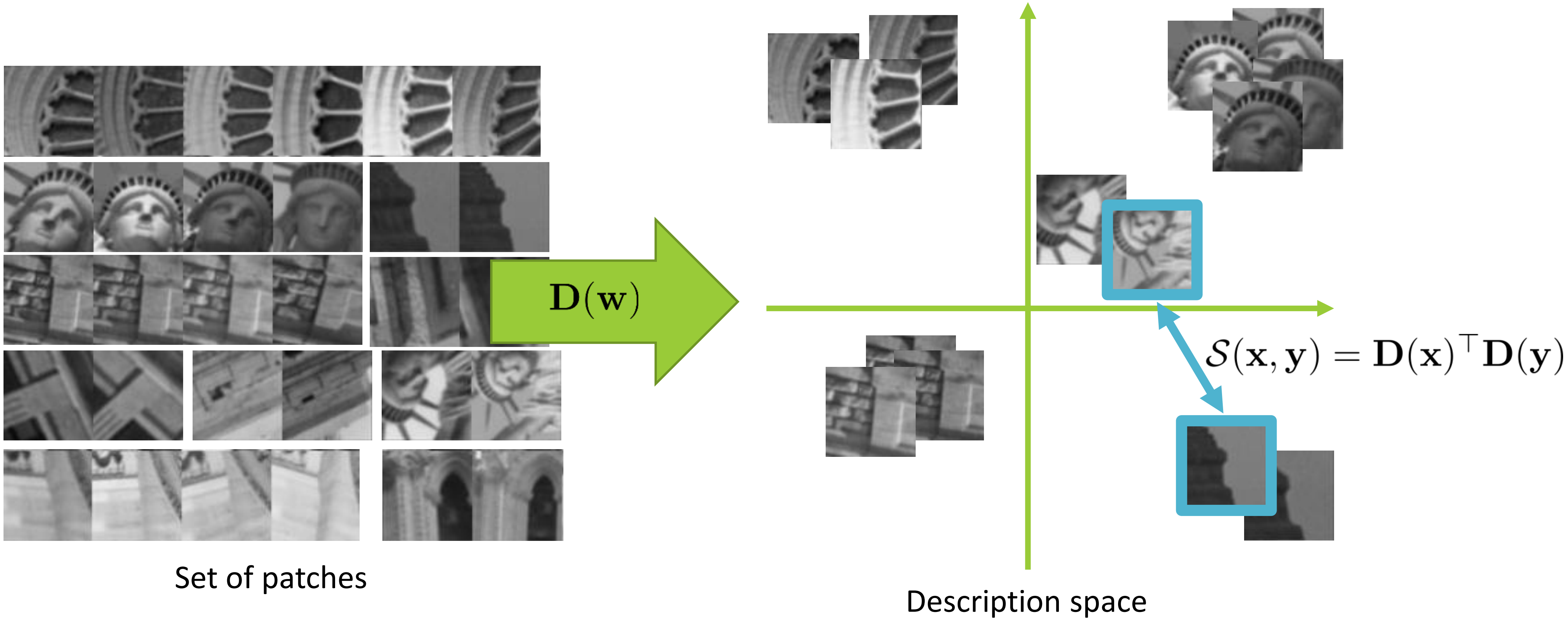
Thresholded Average Box measure

Each measurement function is **thresholded** to obtain the weak-descriptors.



How to select a good set of measurement functions?

Distance is description space: Similarity learning



Winder, S. A., & Brown, M. (2007, June). Learning local image descriptors. In Proc. *CVPR* (pp. 1-8)

How to select a good set of measurement functions?

BELID

BEBLID

BAD & HashSIFT

How to select a good set of measurement functions?

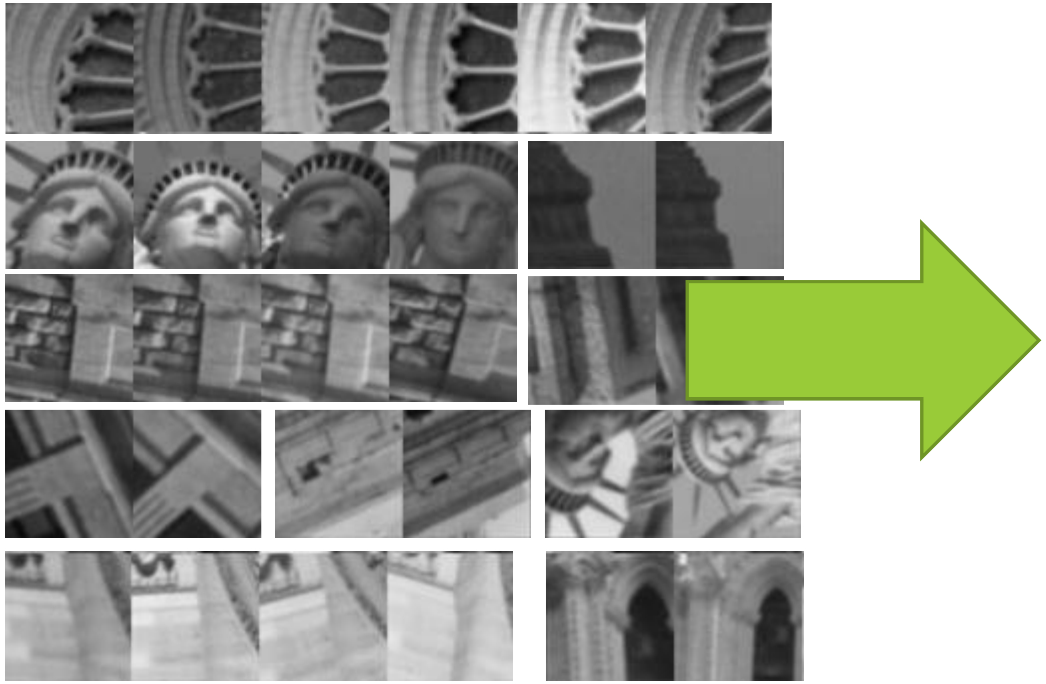
BELID

BEBLID

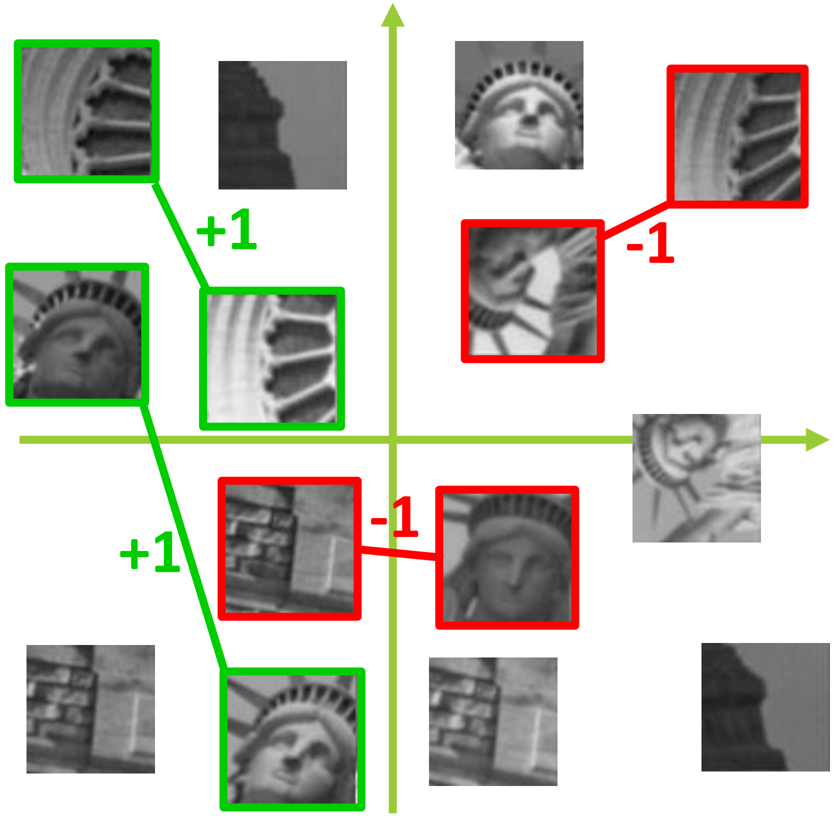
BAD & HashSIFT

Suárez, I., Sfeir, G., Buenaposada, J. M., & Baumela, L. (2019, July). BELID: Boosted efficient local image descriptor. In *Iberian Conference on Pattern Recognition and Image Analysis* (pp. 449-460). Springer, Cham.

BELID: Boosting training process



Set of patches

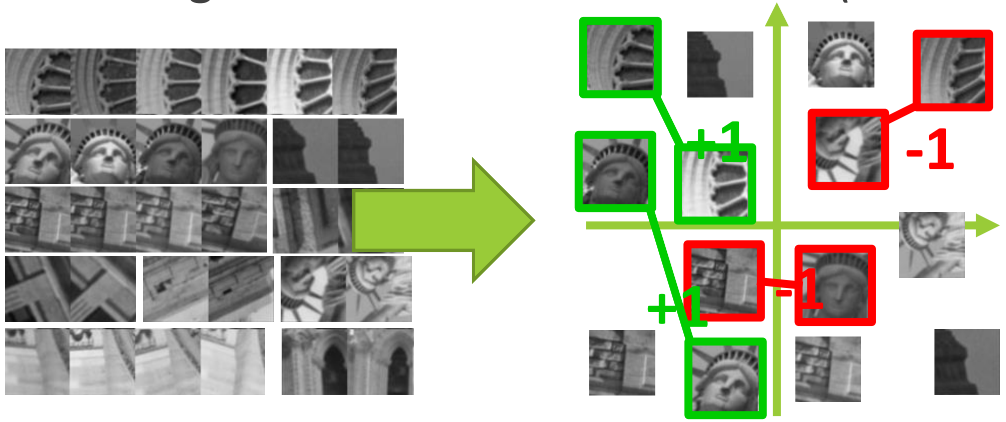


Description space

Binary classification problem { $+1$, -1 }

BELID: Boosting training process

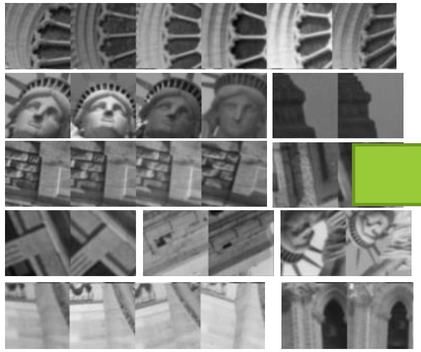
- Training on Brown Balanced Dataset (Winder, 2007). We define our descriptors as:

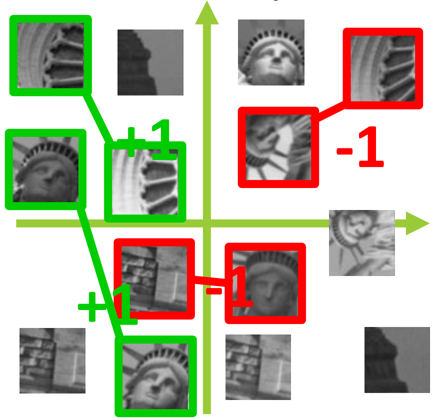


$$\mathbf{h}(\mathbf{x}) = \left[\sqrt{\alpha_1} h_1(\mathbf{x}), \ldots, \sqrt{\alpha_K} h_K(\mathbf{x})\right]$$

BELID: Boosting training process

- Training on Brown Balanced Dataset (Winder, 2007). We define our descriptors as:





0,34	-0.12	...	-0.0065

$$\mathbf{h}(\mathbf{x}) = \left[\sqrt{\alpha_1} h_1(\mathbf{x}), \ldots, \sqrt{\alpha_K} h_K(\mathbf{x})\right]$$

- And we train in a binary classification problem:

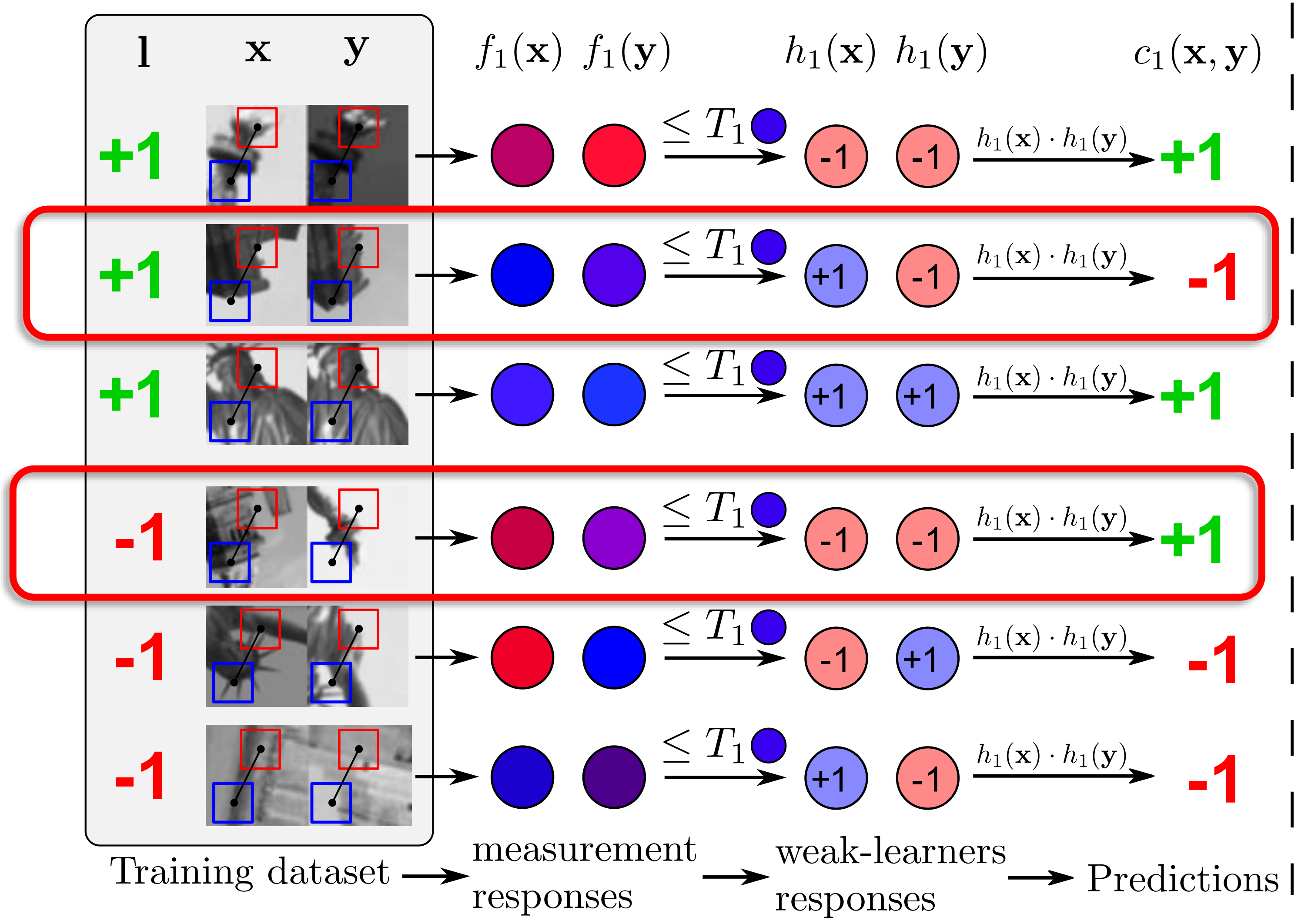
$$\mathrm{C} : (\mathbf{x}, \mathbf{y}) \rightarrow sign\left(\alpha_1 c_1(\mathbf{x}, \mathbf{y}) + \ldots + \alpha_K c_K(\mathbf{x}, \mathbf{y})\right) \rightarrow \pm 1$$

+1 / -1

$$c_i(\mathbf{x}, \mathbf{y}) = h_i(\mathbf{x}) \cdot h_i(\mathbf{y})$$

-1 -1 +1

The Boosting framework: 1st WL



1. Find the weak-learner **h** that maximizes the **Youden's index**.

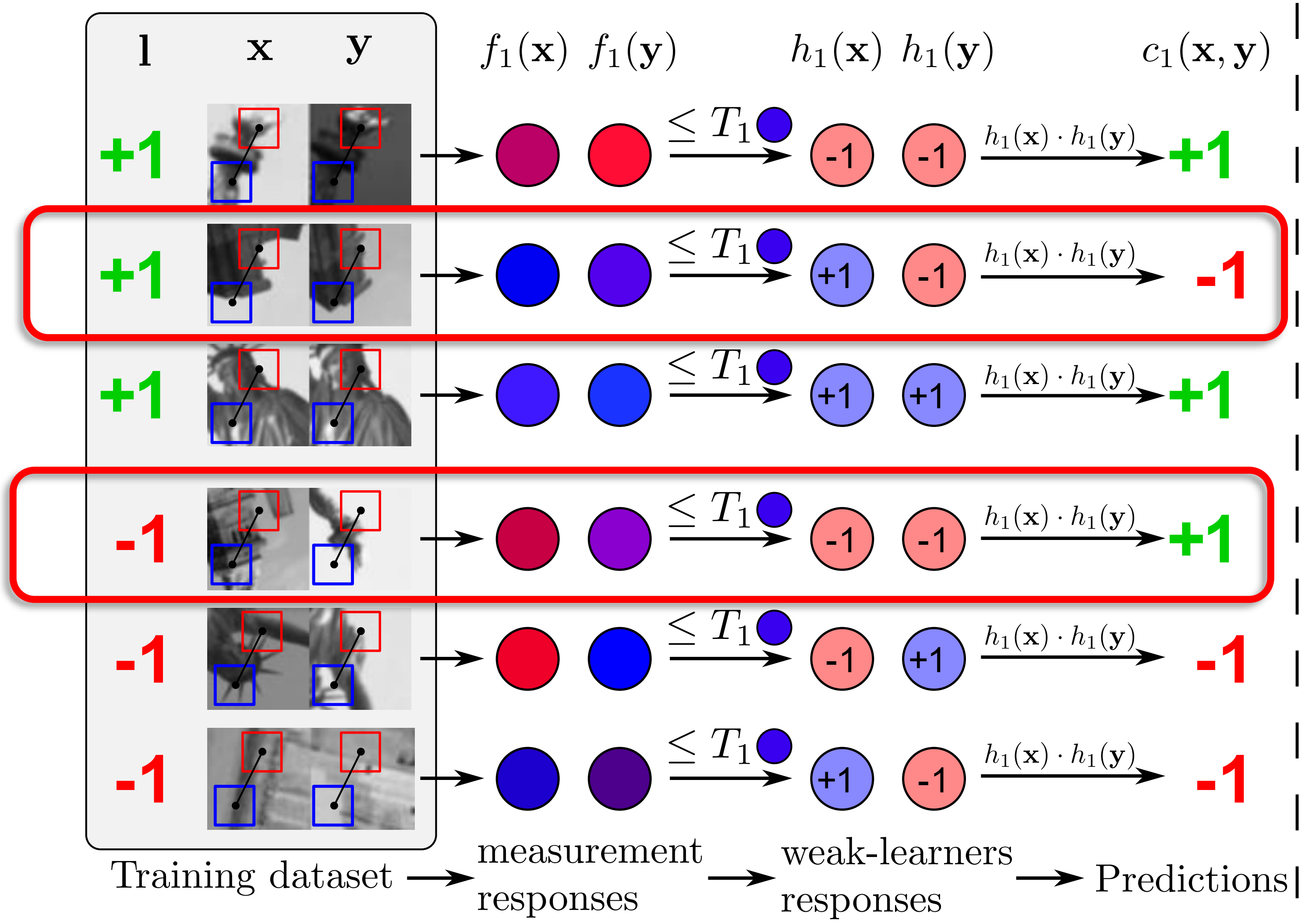
2. Calculate the error

$$\epsilon_1 = \sum_{\substack{i=1 \\ c_1(\mathbf{x}_i, \mathbf{y}_i) \neq l_i}}^{n} w_{i,1}$$

3. Choose the **h** weight:

$$\alpha_1 = \frac{1}{2} \ln \left(\frac{1-\epsilon_1}{\epsilon_1} \right)$$

The Boosting framework: 1st WL



1. Find the weak-learner **h** that maximizes the **Youden's index**.

2. Calculate the error

$$\epsilon_1 = \sum_{\substack{i=1 \\ c_1(\mathbf{x}_i, \mathbf{y}_i) \neq l_i}}^{n} w_{i,1}$$

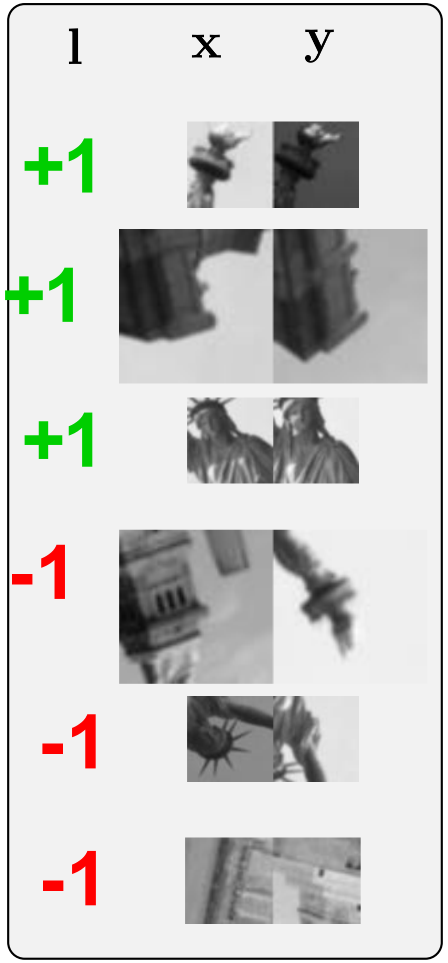
3. Choose the **h** weight:

$$\alpha_1 = \frac{1}{2} \ln \left(\frac{1-\epsilon_1}{\epsilon_1} \right)$$

4. Re-weight the samples:

$$w_{i,2} = w_{i,1} e^{-l_i \alpha_1 c_1(\mathbf{x}_i, \mathbf{y}_i)}$$

The Boosting framework: 2nd WL



Training dataset

1. Find the weak-learner **h** that maximizes the **Youden's index**.

2. Calculate the error

$$\epsilon_1 = \sum_{\substack{i=1 \\ c_1(\mathbf{x}_i,\mathbf{y}_i)\neq l_i}}^{n} w_{i,1}$$

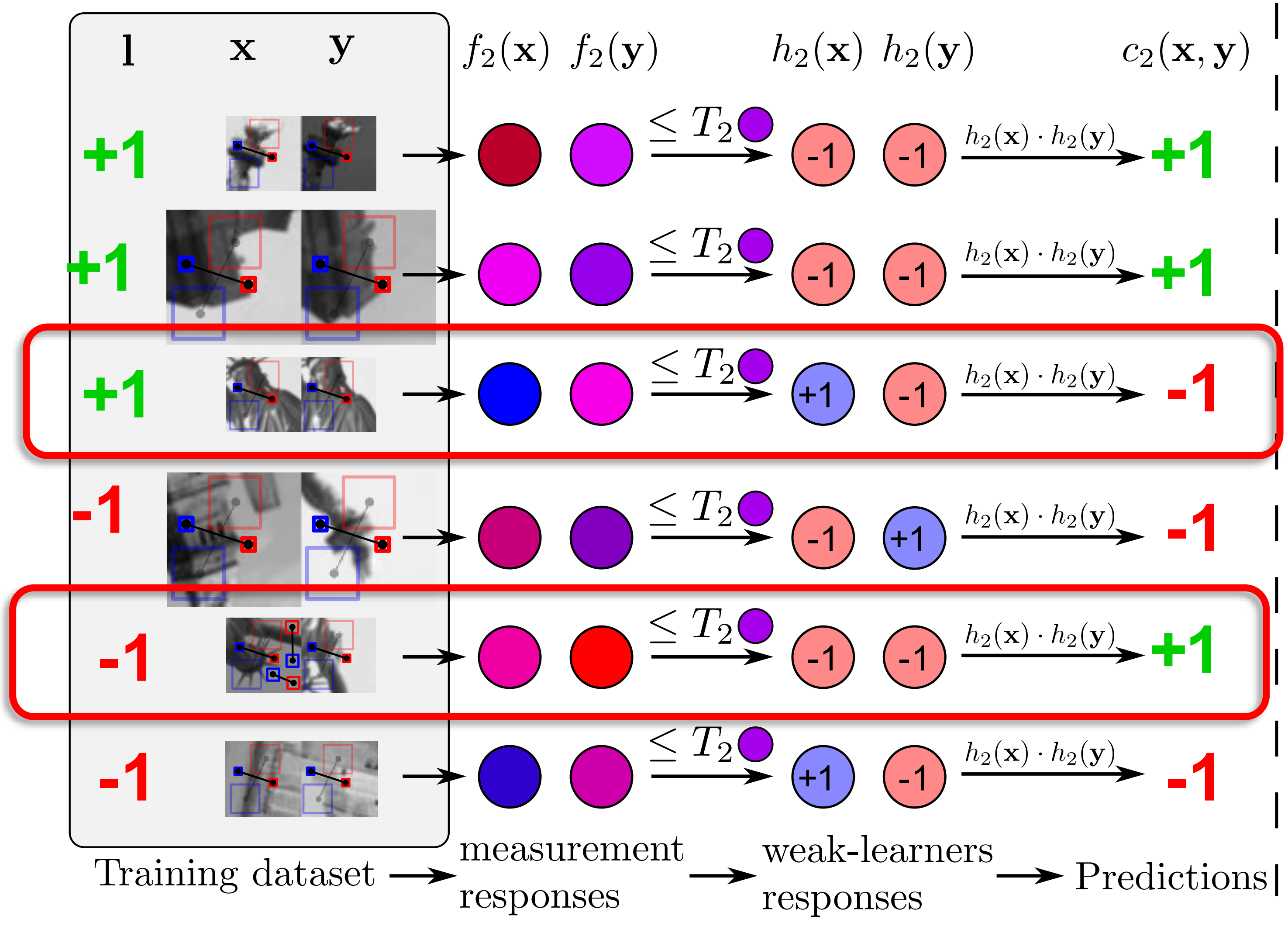
3. Choose the **h** weight:

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right)$$

4. Re-weight the samples:

$$w_{i,2} = w_{i,1} e^{-l_i \alpha_1 c_1(\mathbf{x}_i,\mathbf{y}_i)}$$

The Boosting framework: 2nd WL



1. Find the weak-learner $\mathbf{h}$ that maximizes the **Youden's index**.

2. Calculate the error

$$\epsilon_2 = \sum_{\substack{i=1 \\ c_2(\mathbf{x}_i, \mathbf{y}_i) \neq l_i}}^{n} w_{i,2}$$

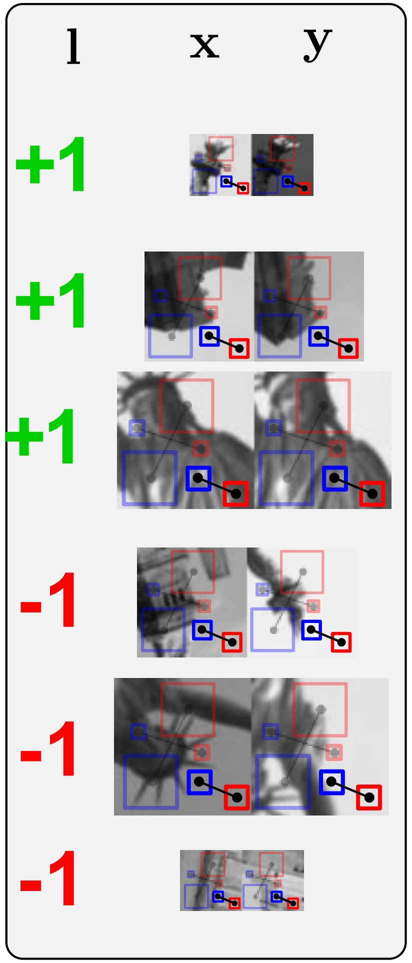
3. Choose the $\mathbf{h}$ weight:

$$\alpha_2 = \frac{1}{2} \ln \left(\frac{1-\epsilon_2}{\epsilon_2} \right)$$

4. Re-weight the samples:

$$w_{i,3} = w_{i,2} e^{-l_i \alpha_2 c_2(\mathbf{x}_i, \mathbf{y}_i)}$$

The Boosting framework: 3rd WL



$f_t(\mathbf{x})$ $f_t(\mathbf{y})$ $h_t(\mathbf{x})$ $h_t(\mathbf{y})$ $c_t(\mathbf{x},\mathbf{y})$

Training dataset $\longrightarrow$ measurement responses $\longrightarrow$ weak-learners responses $\longrightarrow$ Predictions

1. Find the weak-learner $\mathbf{h}$ that maximizes the **Youden's index**.

2. Calculate the error

$$\epsilon_t = \sum_{\substack{i=1 \\ c_t(\mathbf{x}_i,\mathbf{y}_i)\neq l_i}}^{n} w_{i,t}$$

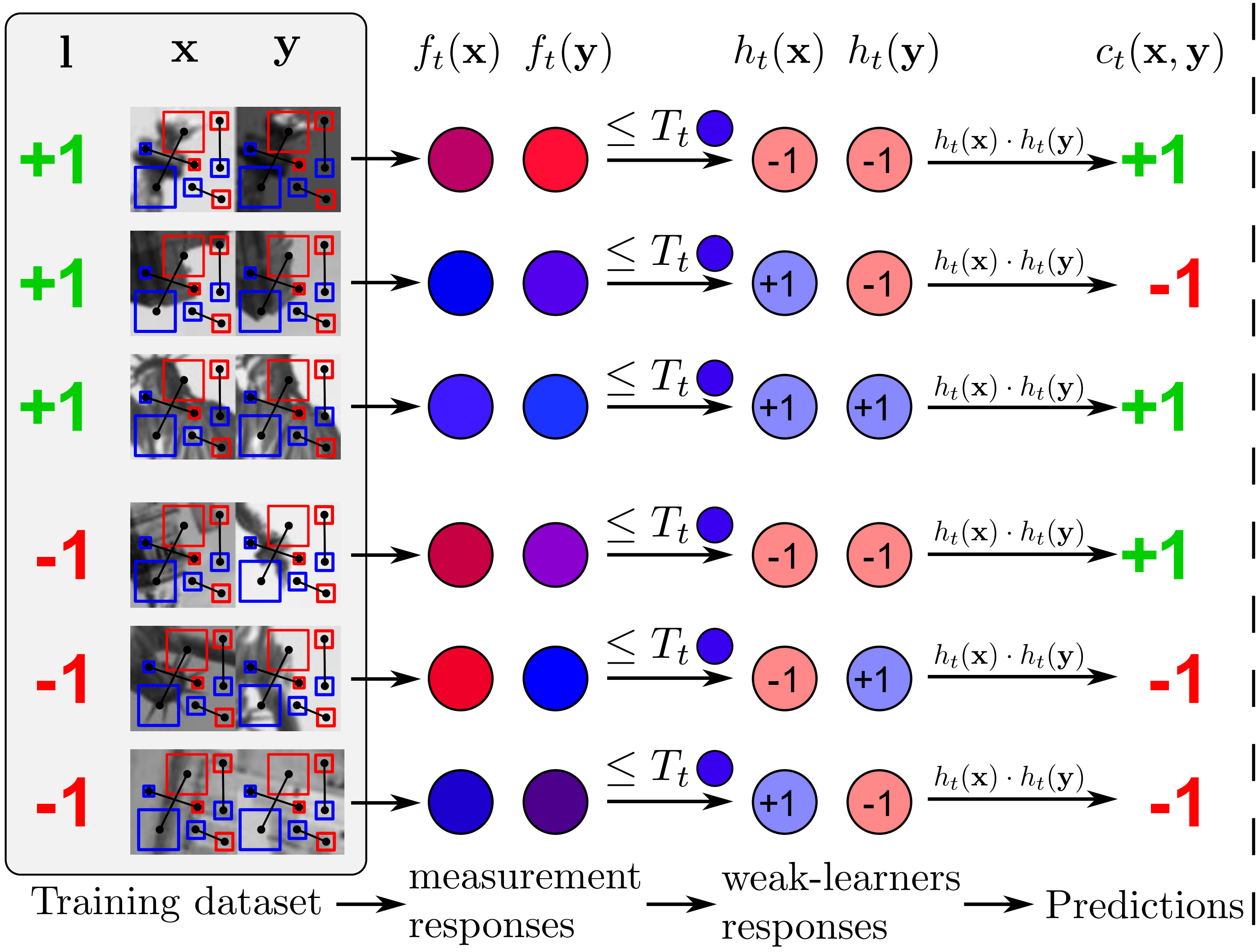
3. Choose the $\mathbf{h}$ weight:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

4. Re-weight the samples:

$$w_{i,t+1} = w_{i,t}e^{-l_i\alpha_t c_t(\mathbf{x}_i,\mathbf{y}_i)}$$

The Boosting framework



1. Find the weak-learner **h** that maximizes the **Youden's index**.

2. Calculate the error

$$\epsilon_t = \sum_{\substack{i=1 \\ c_t(\mathbf{x}_i,\mathbf{y}_i)\neq l_i}}^{n} w_{i,t}$$

3. Choose the **h** weight:

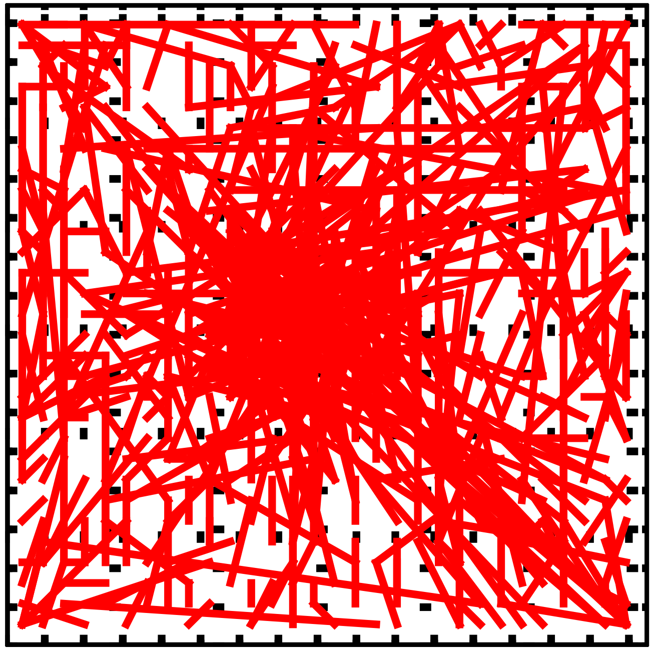
$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

4. Re-weight the samples:

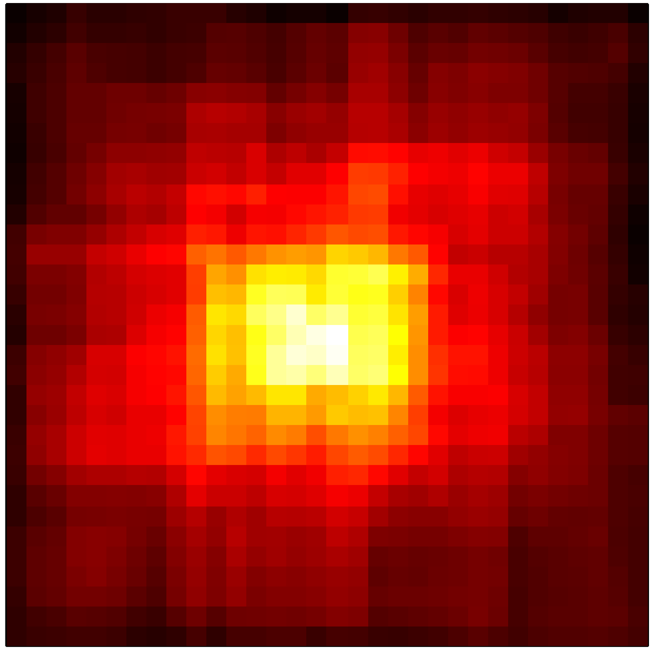
$$w_{i,t+1} = w_{i,t}e^{-l_i\alpha_t c_t(\mathbf{x}_i,\mathbf{y}_i)}$$

BELID: Result of the training process

Selected measurement regions



Heatmap of the weighted measurement regions

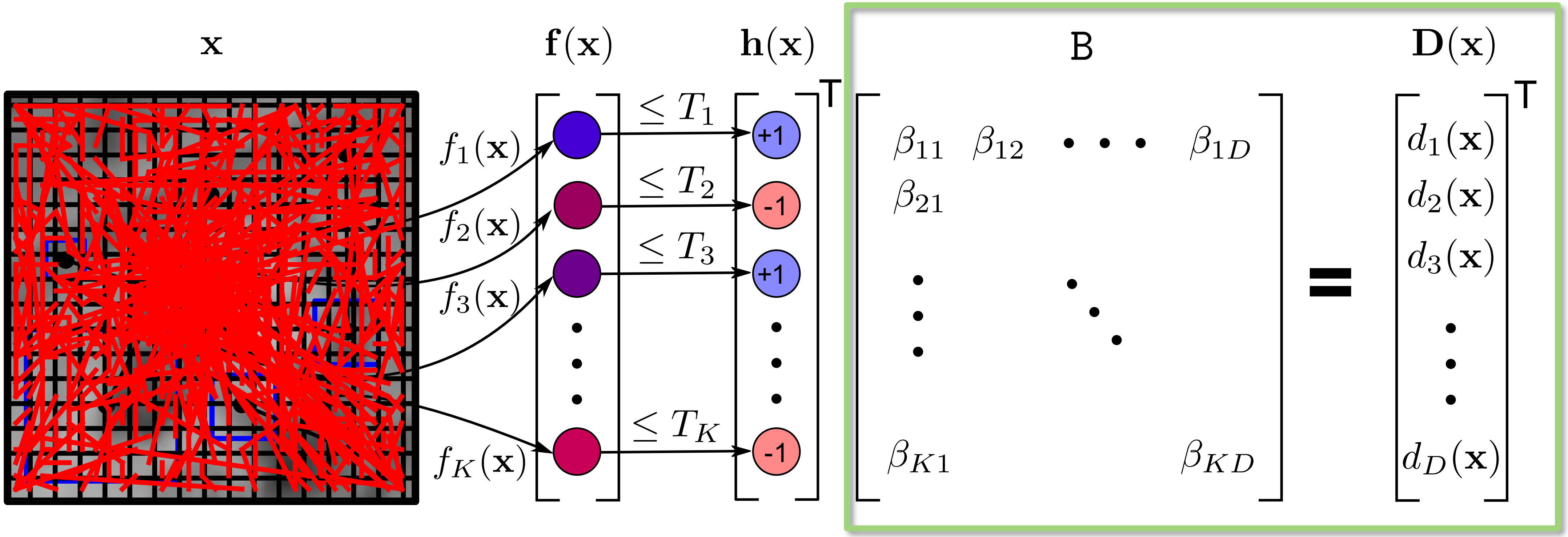


Training results for 512 weak-learners

BELID: Final projection

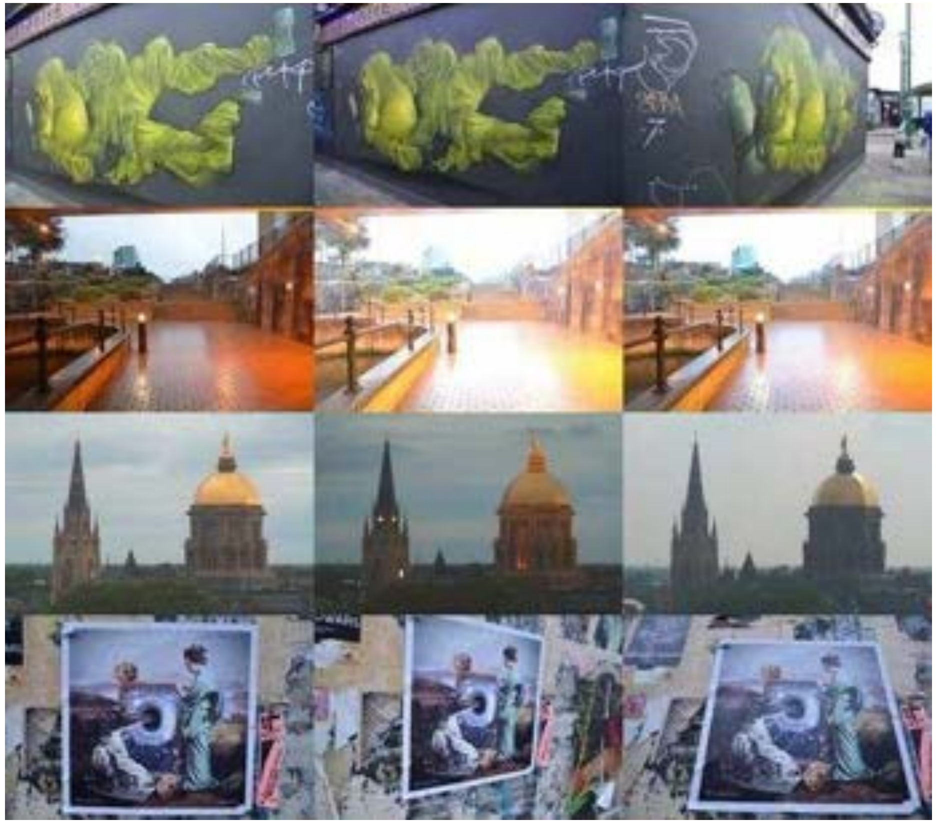
Linear Transformation:

- Give more weight to the best h(x).
- Models the correlation between the h's.



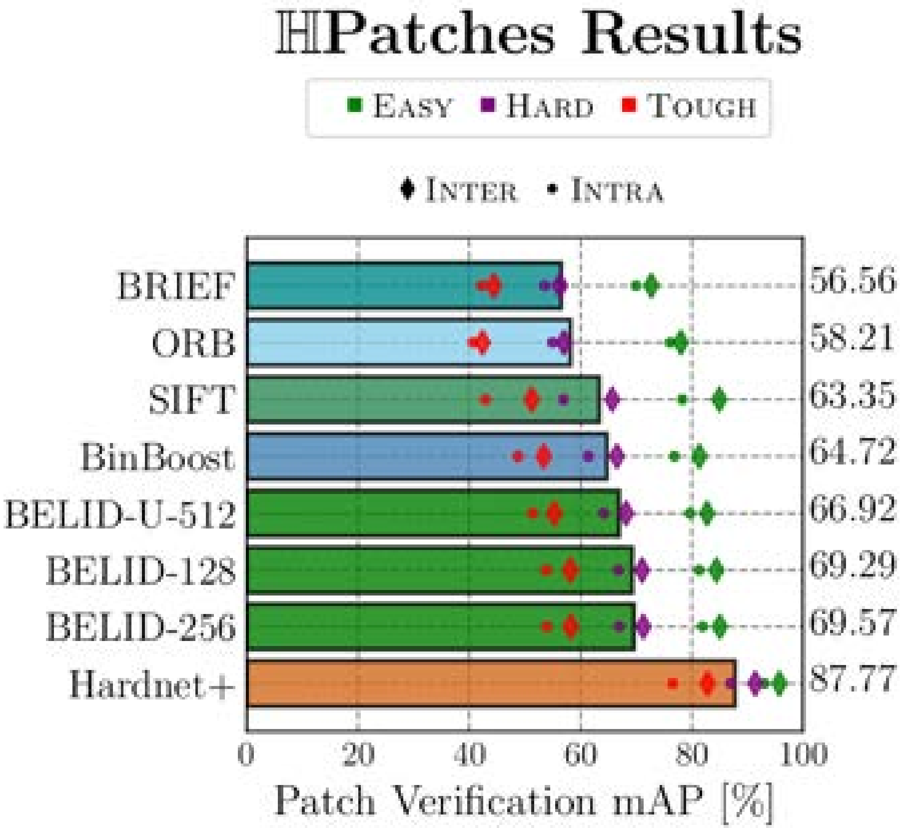
BELID Evaluation results: Hpatches

Hpatches is a multitask dataset of patches:



(Balntas, 2017)

Results for the verification task:

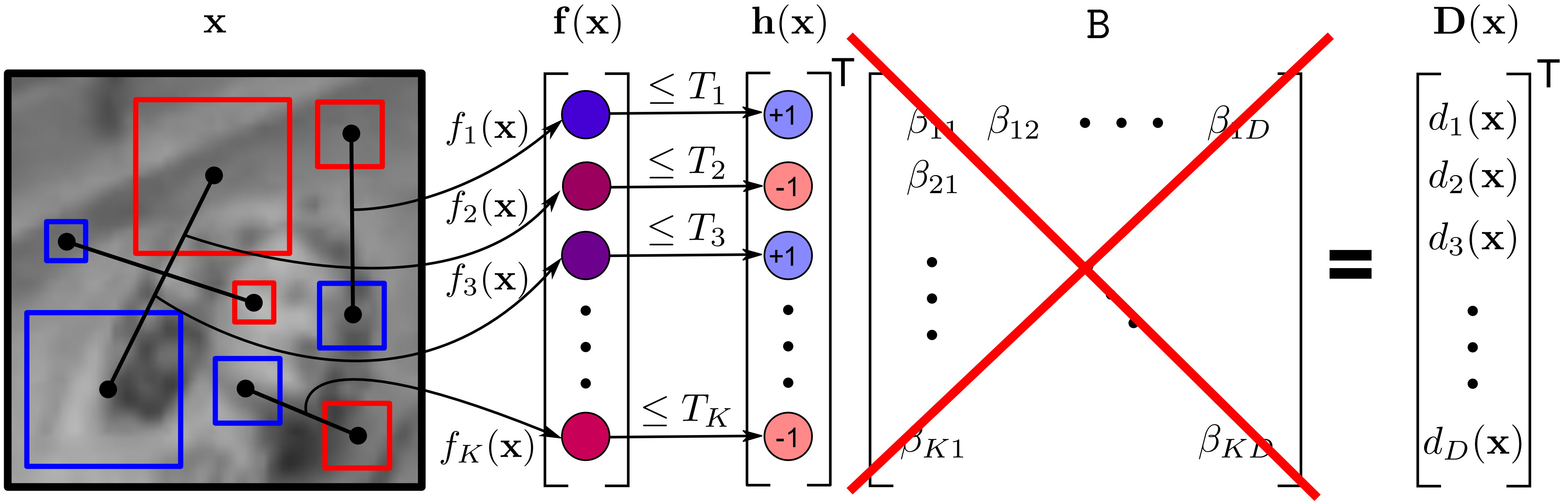


How to select a good set of measurement functions?

Suárez, I., Sfeir, G., Buenaposada, J. M., & Baumela, L. (2020). BEBLID: Boosted efficient binary local image descriptor. *Pattern Recognition Letters*, *133*, 366-372.

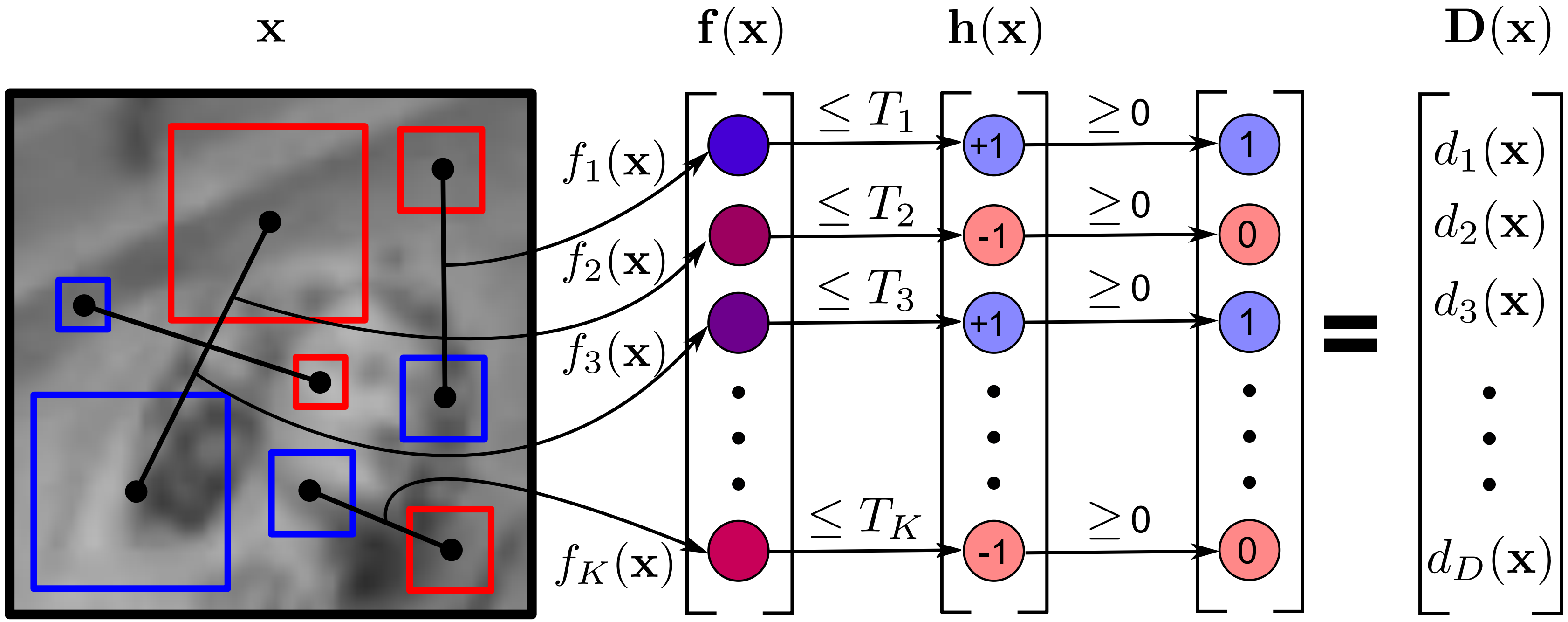
BEBLID: Descriptor binarization

To speed up the description and binarize the result, we remove B



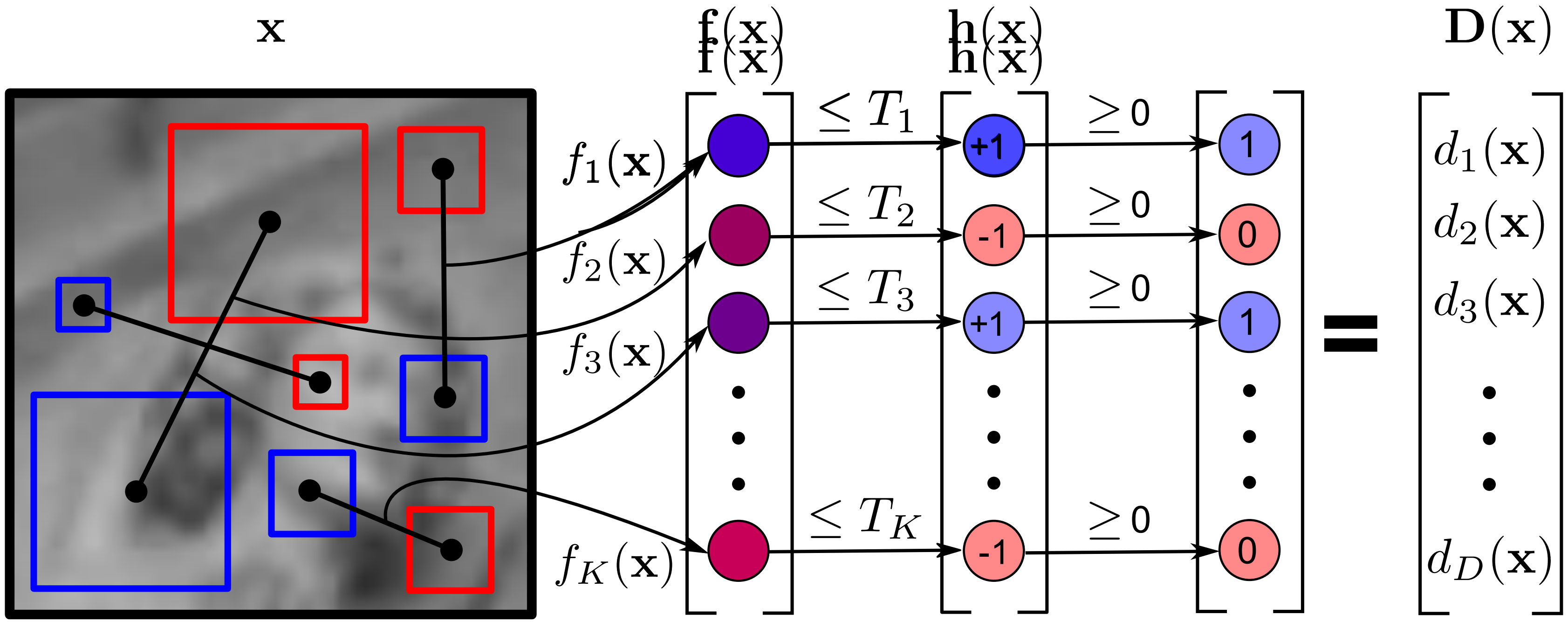
BEBLID: Descriptor binarization

To speed up the description and binarize the result, we remove B



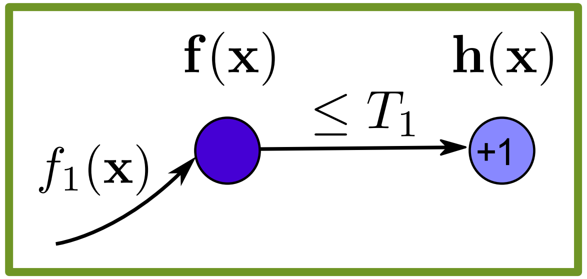
BEBLID: Descriptor binarization

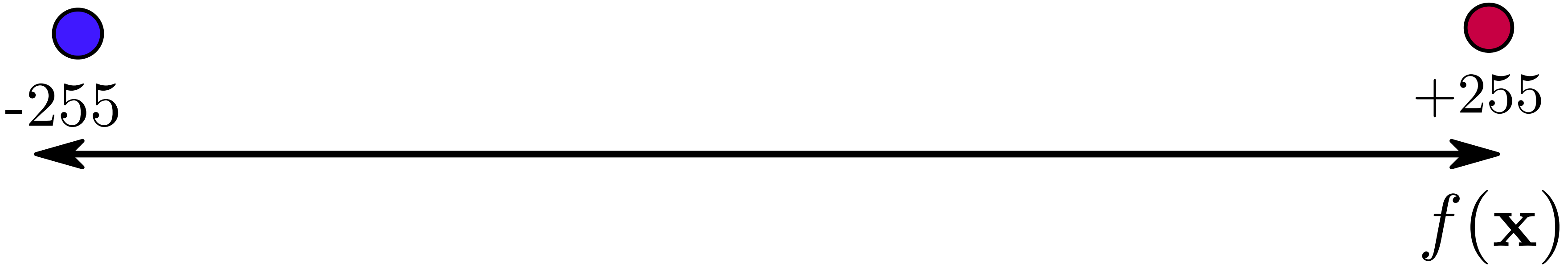
To speed up the description and binarize the result, we remove B



BEBLID: New WL threshold search

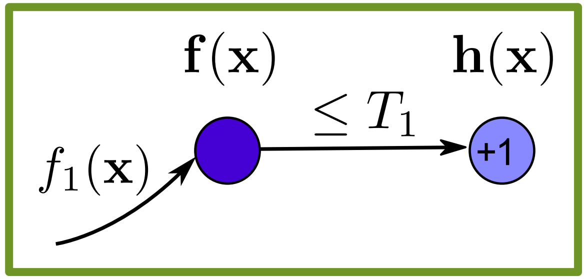
- How to find the optimal threshold T_1 ?
  - Let's plot the values of the measurement function

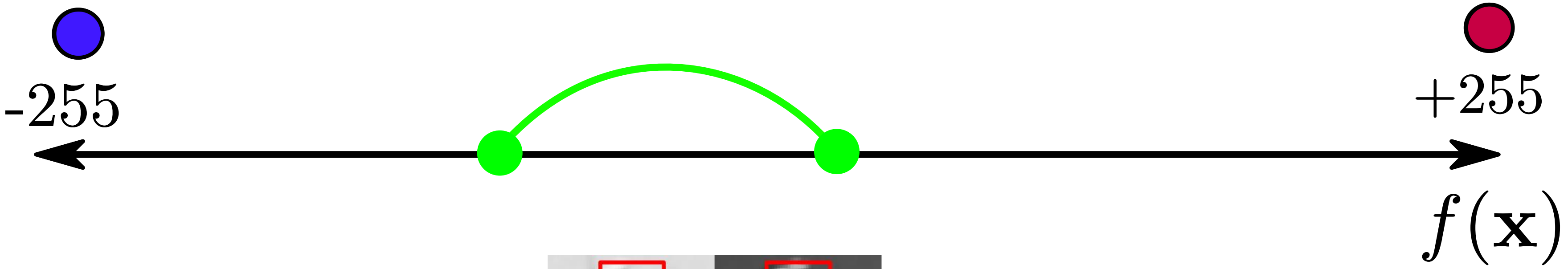


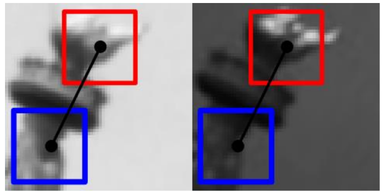


BEBLID: New WL threshold search

- How to find the optimal threshold T_1?
  - Let's plot the values of the measurement function

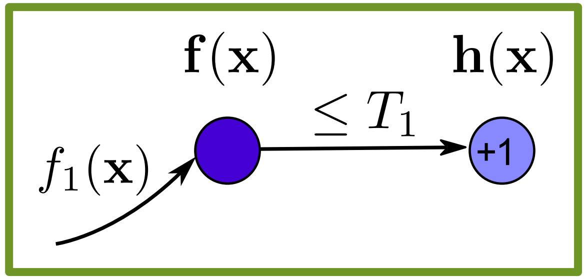


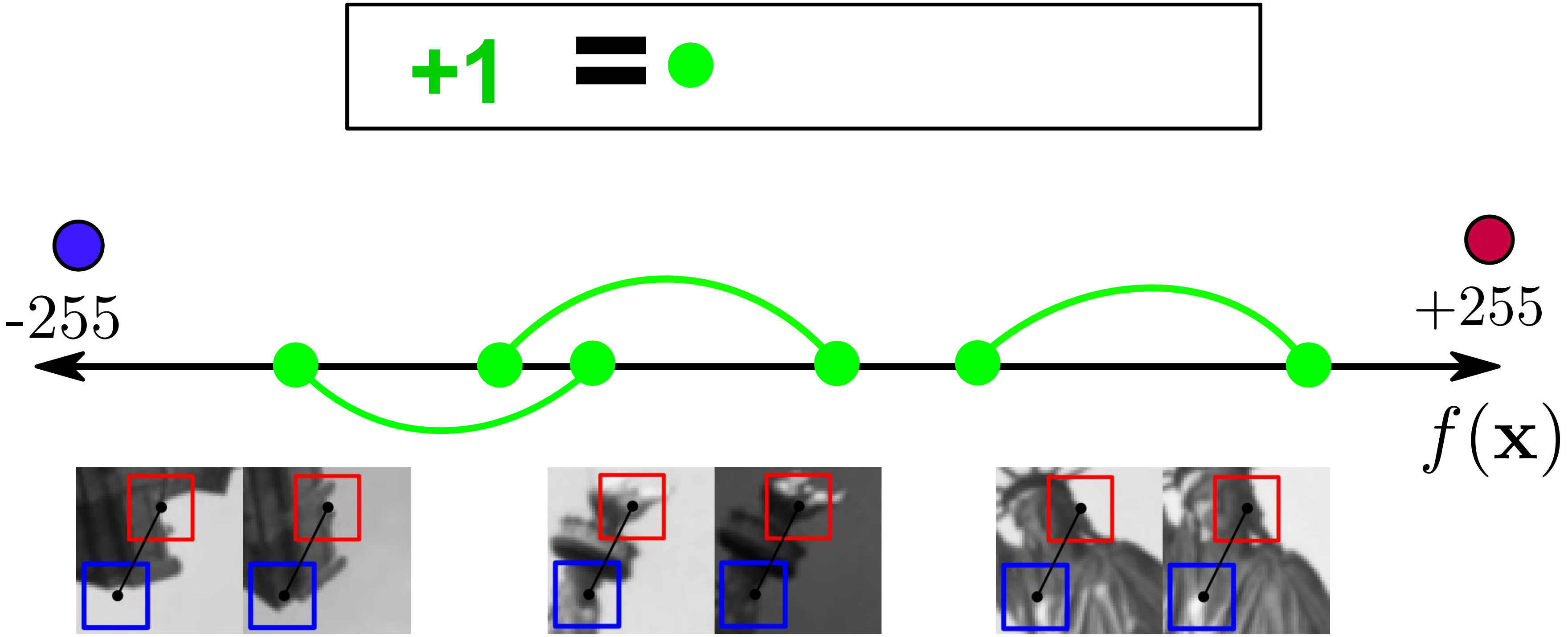




BEBLID: New WL threshold search

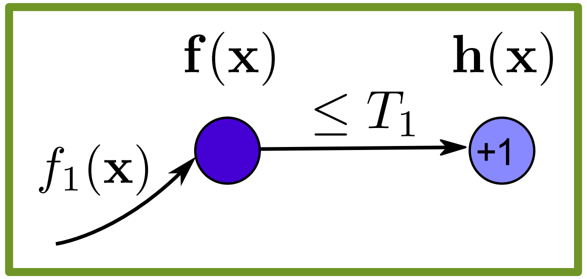
- How to find the optimal threshold T_1?
  - Let's plot the values of the measurement function

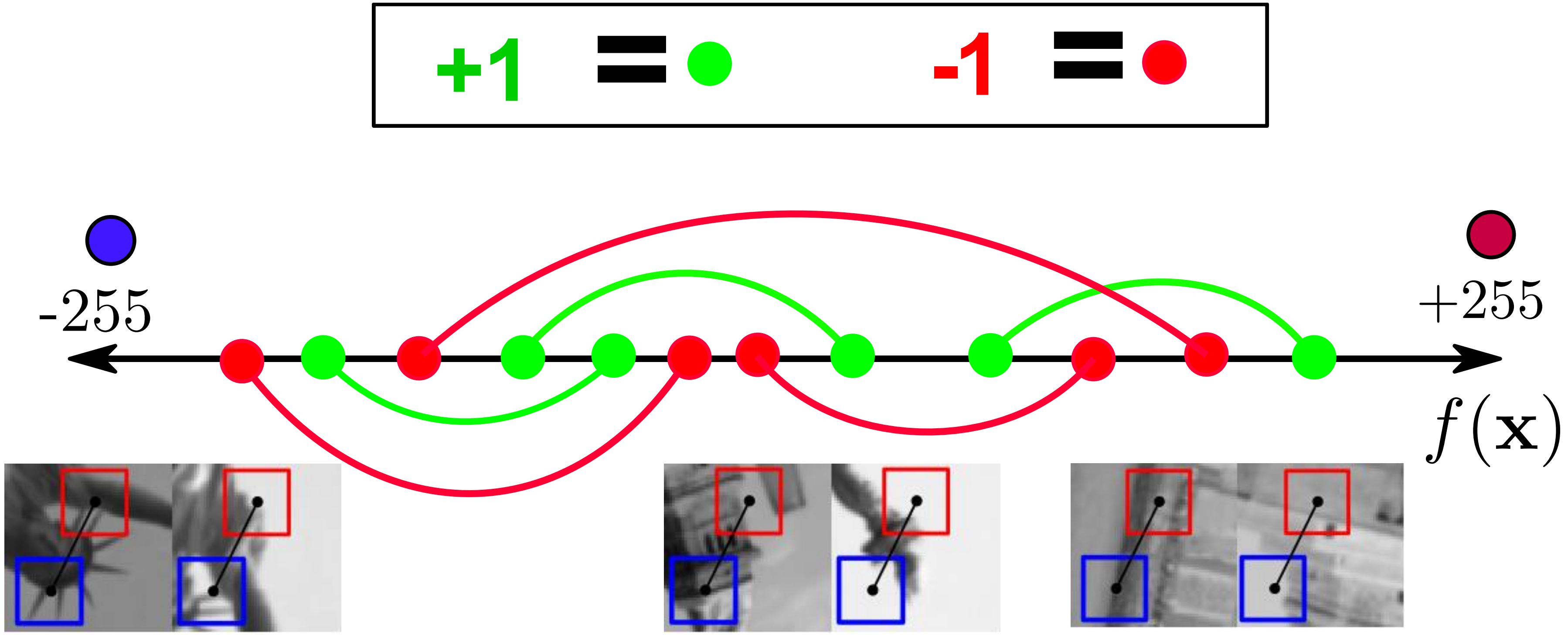




BEBLID: New WL threshold search

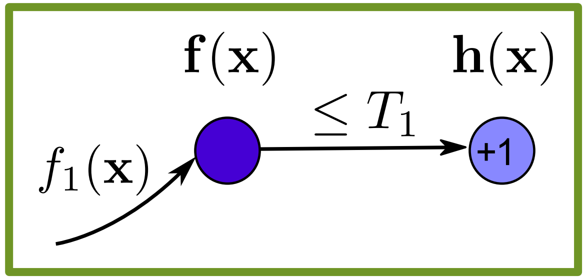
- How to find the optimal threshold T_1 ?
  - Let's plot the values of the measurement function

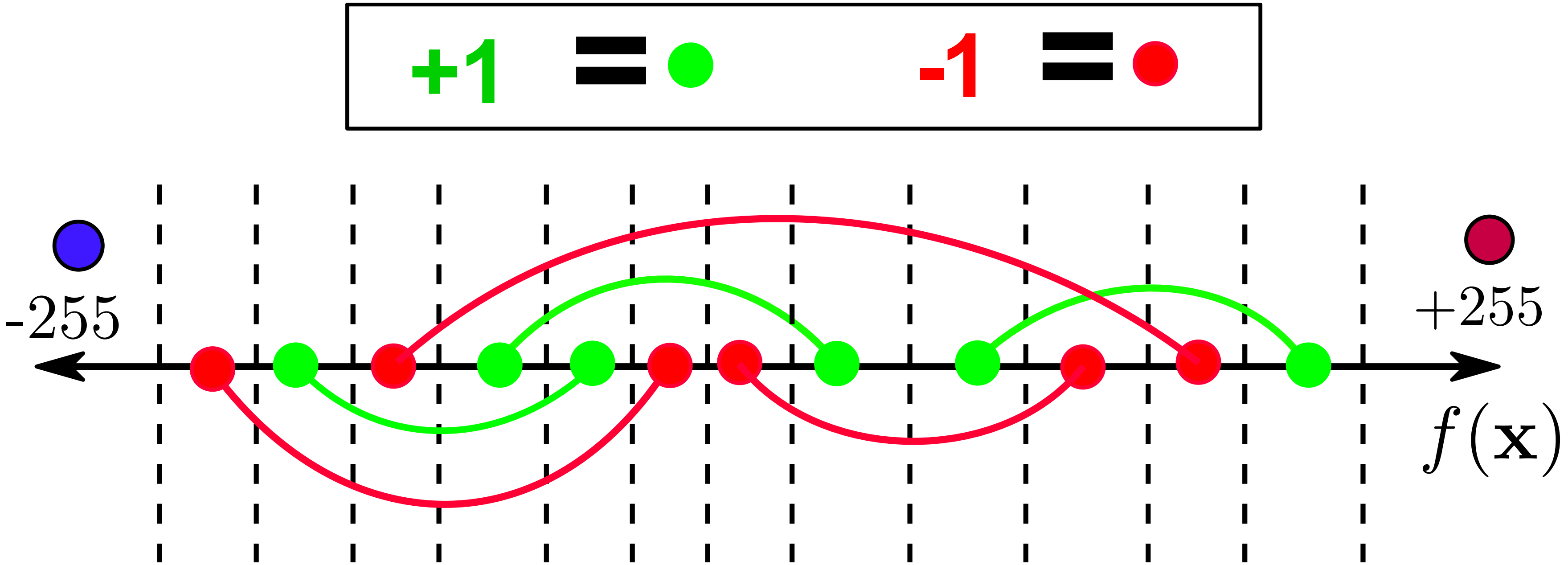




BEBLID: New WL threshold search

- How to find the optimal threshold T_1?
  - Let's plot the values of the measurement function

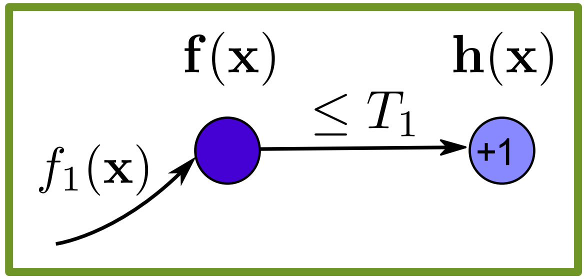


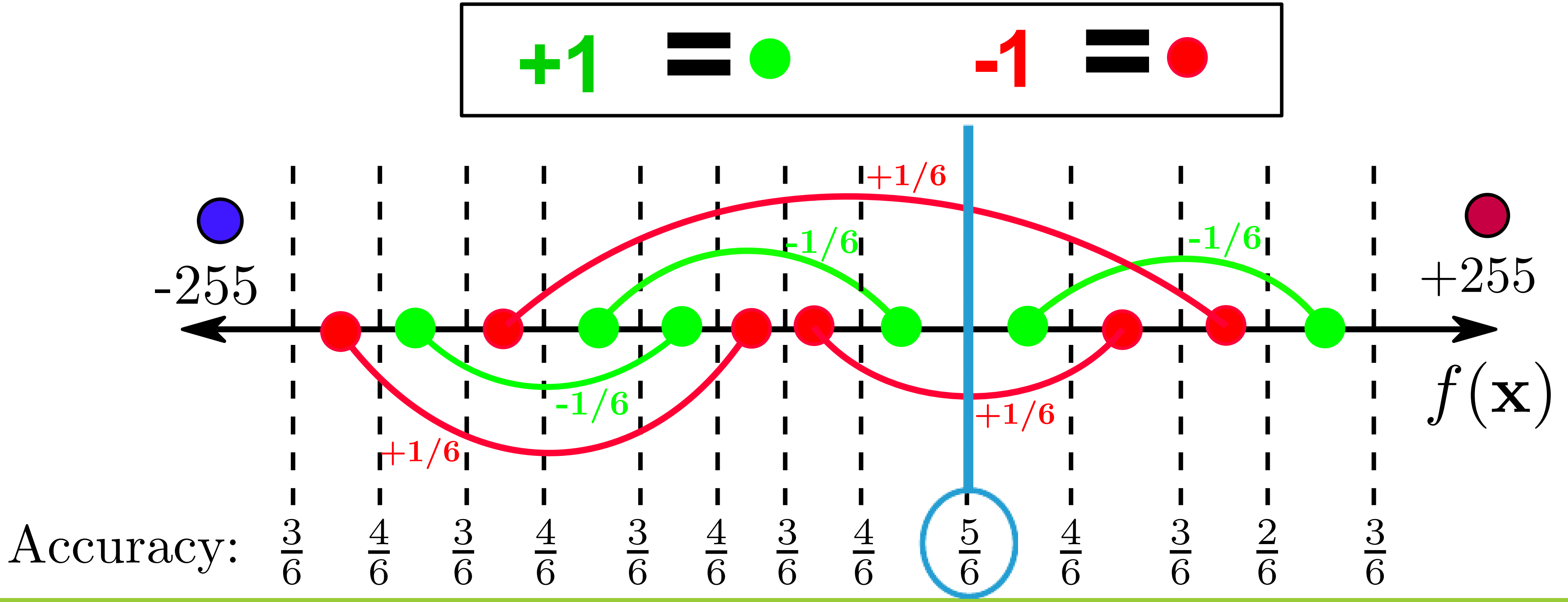


Accuracy:

BEBLID: New WL threshold search

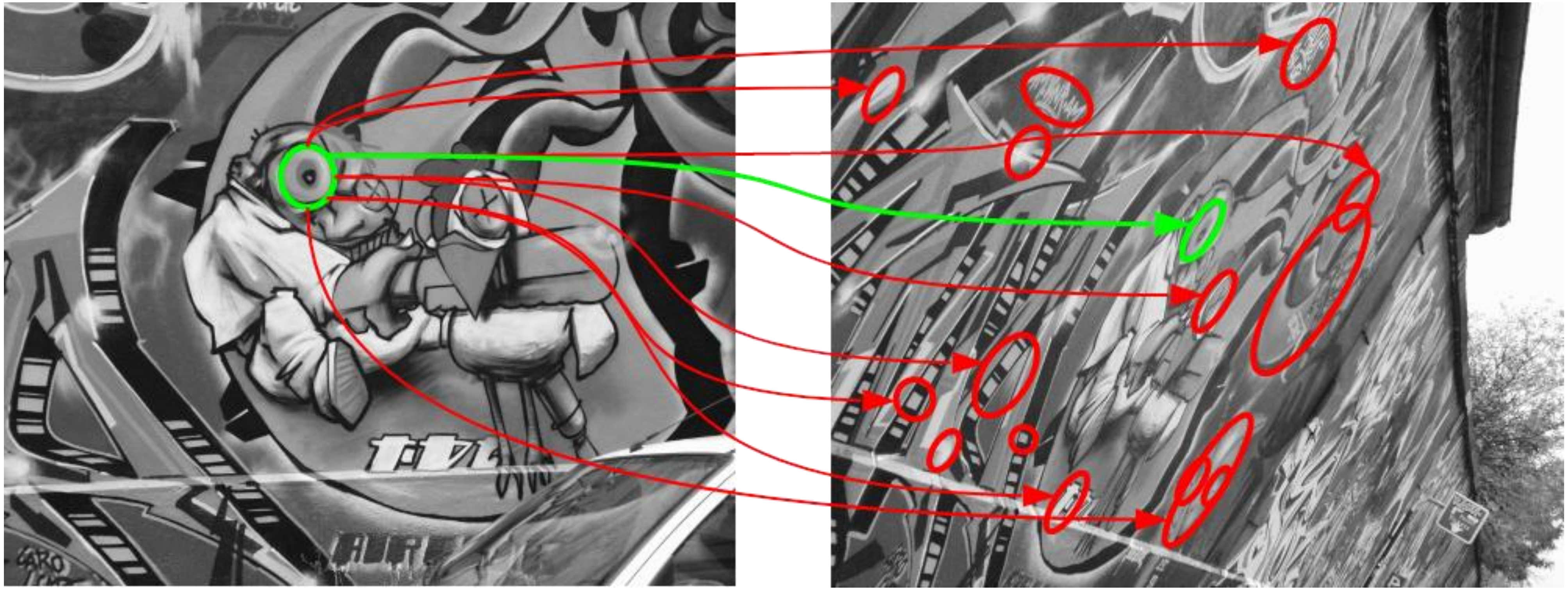
- How to find the optimal threshold T_1?
  - Let's plot the values of the measurement function





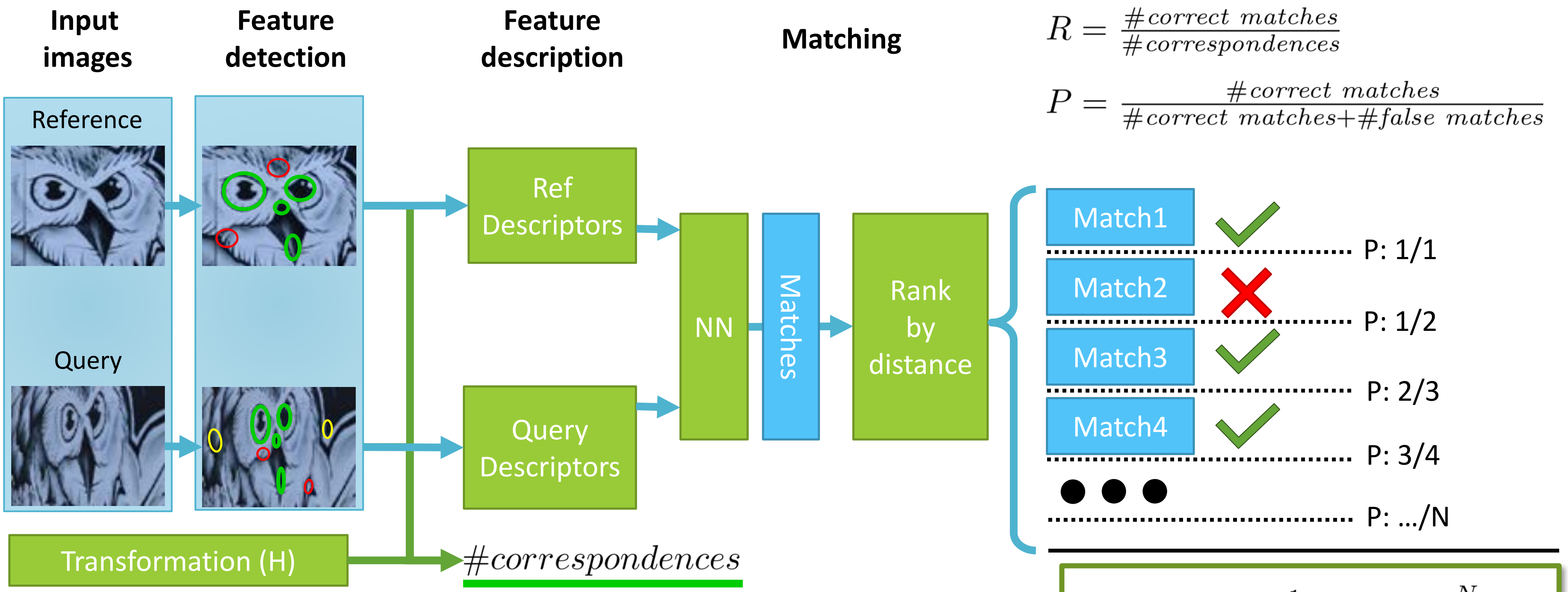
BEBLID: Learning an unbalanced problem

Image matching and Patch retrieval are highly unbalanced problems:



We change the training data set balance → **20% positive patch pairs, 80% negatives**

Evaluation metric: Image matching mAP



$$R = \frac{\#correct\ matches}{\#correspondences}$$

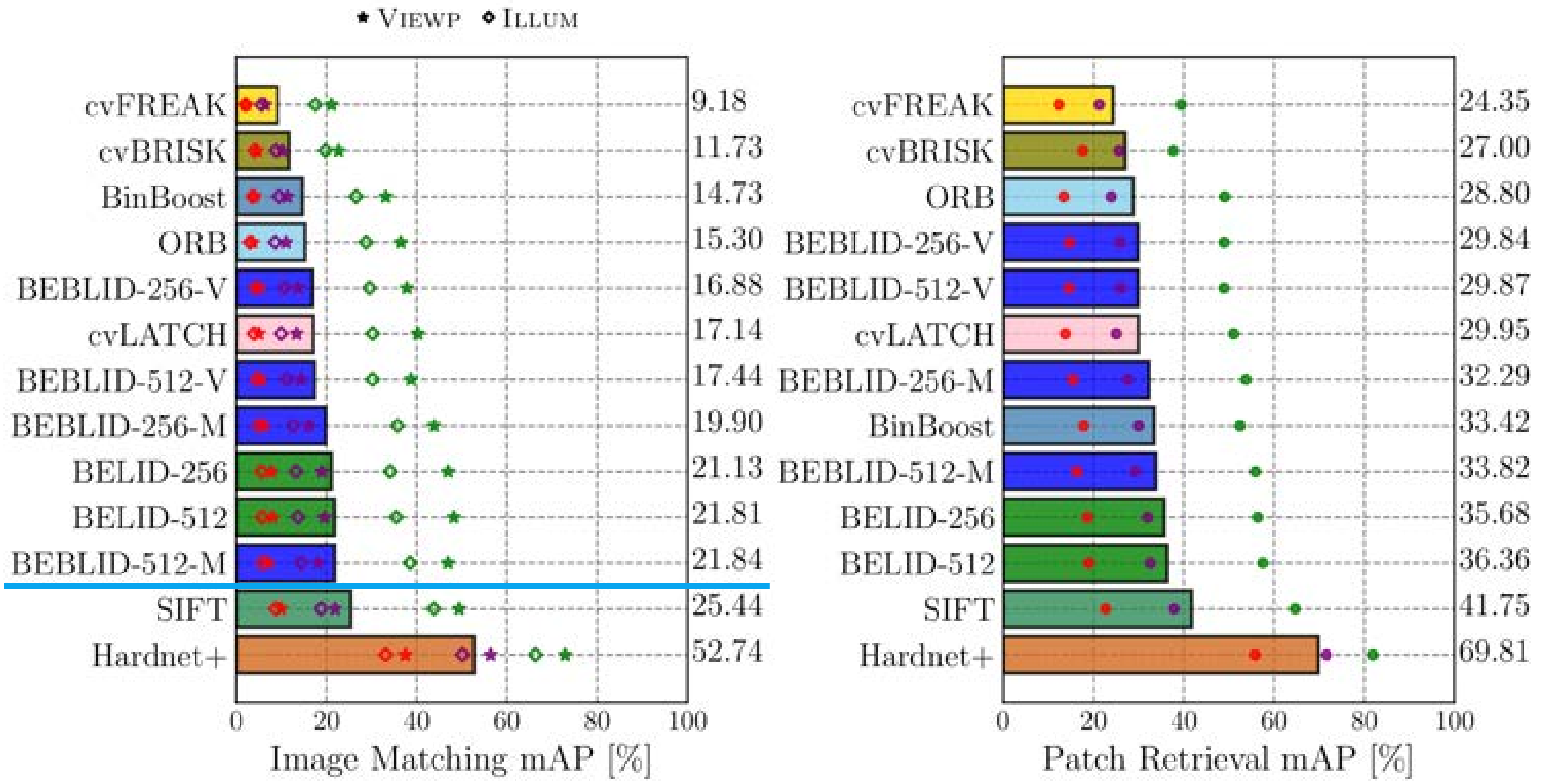
$$P = \frac{\#correct\ matches}{\#correct\ matches + \#false\ matches}$$

$$\mathrm{AP} = \frac{1}{\#correspondences} \sum_{i=1}^{N} P_i$$

Balntas, V., Lenc, K., Vedaldi, A., & Mikolajczyk, K. (2017). HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of CVPR* (pp. 5173-5182).
Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE TPAMI*, *27*(10), 1615-1630.

BEBLID: HPatches results

We improve our BELID results in Matching and retrieval



Description times

Method	Size	Intel Core i7 8750H	Exynox Octa S	Snapdragon 855
BRISK	512b	14.94 (±0.31)	164.75 (±4.10)	19.38 (±0.19)
ORB	256b	12.07 (±0.33)	100.04 (±1.16)	16.40 (±0.25)
LDB	256b	17.48 (±0.8)	161.30 (±4.44)	27.32 (±0.11)
LATCH	512b	101.89 (±1.67)	1509.66 (±24.35)	159.02 (±0.24)
<u>BEBLID</u>	256b	1.56 (±0.05)	20.04 (±0.31)	4.75 (±0.06)
<u>BEBLID</u>	512b	2.84 (±0.07)	31.62 (±0.26)	7.18 (±0.21)

cv 〉 xfeatures2d 〉 BEBLID 〉

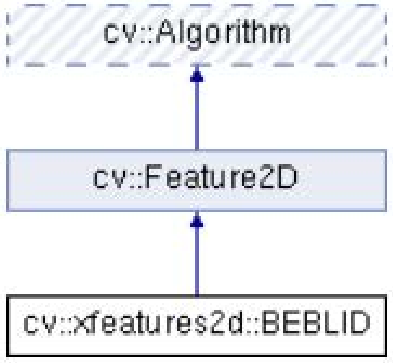
cv::xfeatures2d::BEBLID Class Reference

Extra 2D Features Framework » Experimental 2D Features Algorithms

Class implementing **BEBLID** (Boosted Efficient Binary Local Image Descriptor), described in [232] . Mo

```
#include <opencv2/xfeatures2d.hpp>
```

Inheritance diagram for cv::xfeatures2d::BEBLID:



OpenCV

BEBLID: OpenCV

THE DESCRIPTOR HAS BEEN ADDED TO OPENCV!

How to select a good set of measurement functions?

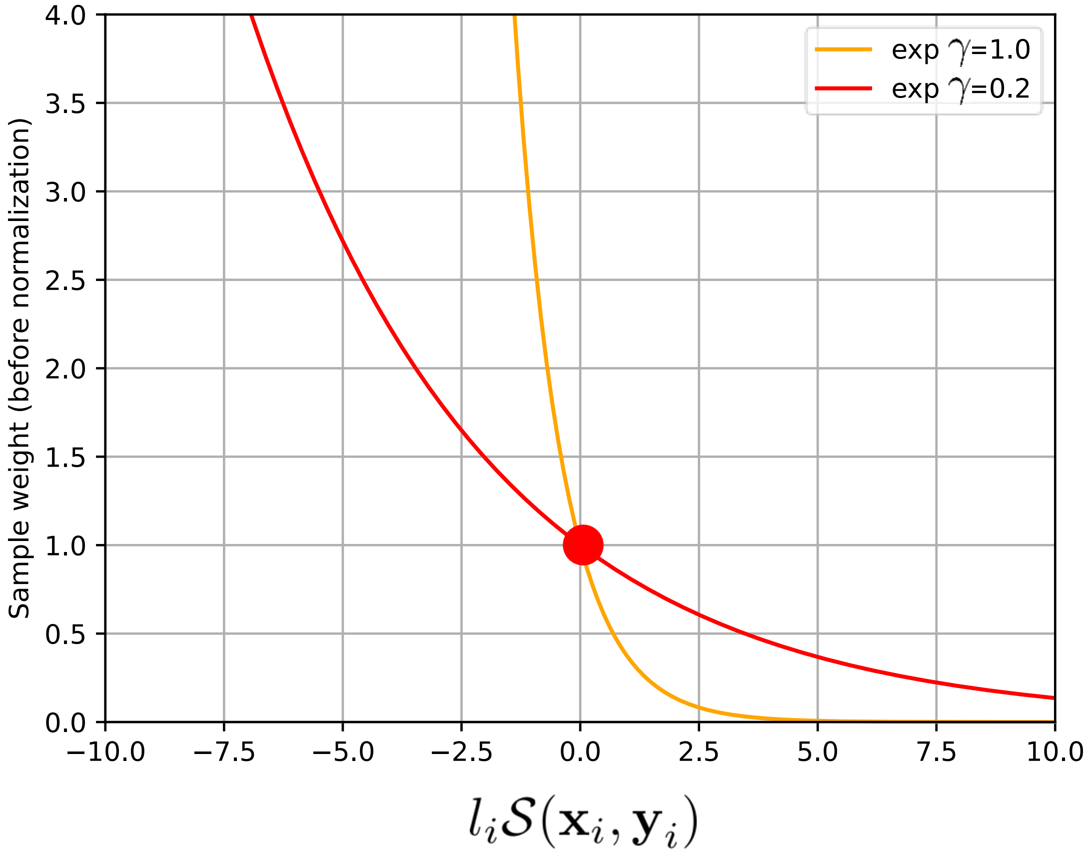
BELID

BEBLID

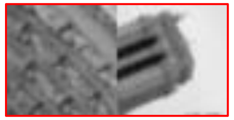
BAD & HashSIFT

Suárez, I., Buenaposada, J. M., & Baumela, L. (2021). Revisiting Binary Local Image Description for Resource Limited Devices. *IEEE Robotics and Automation Letters*, *6*(4), 8317-8324.

Choosing loss function and margin

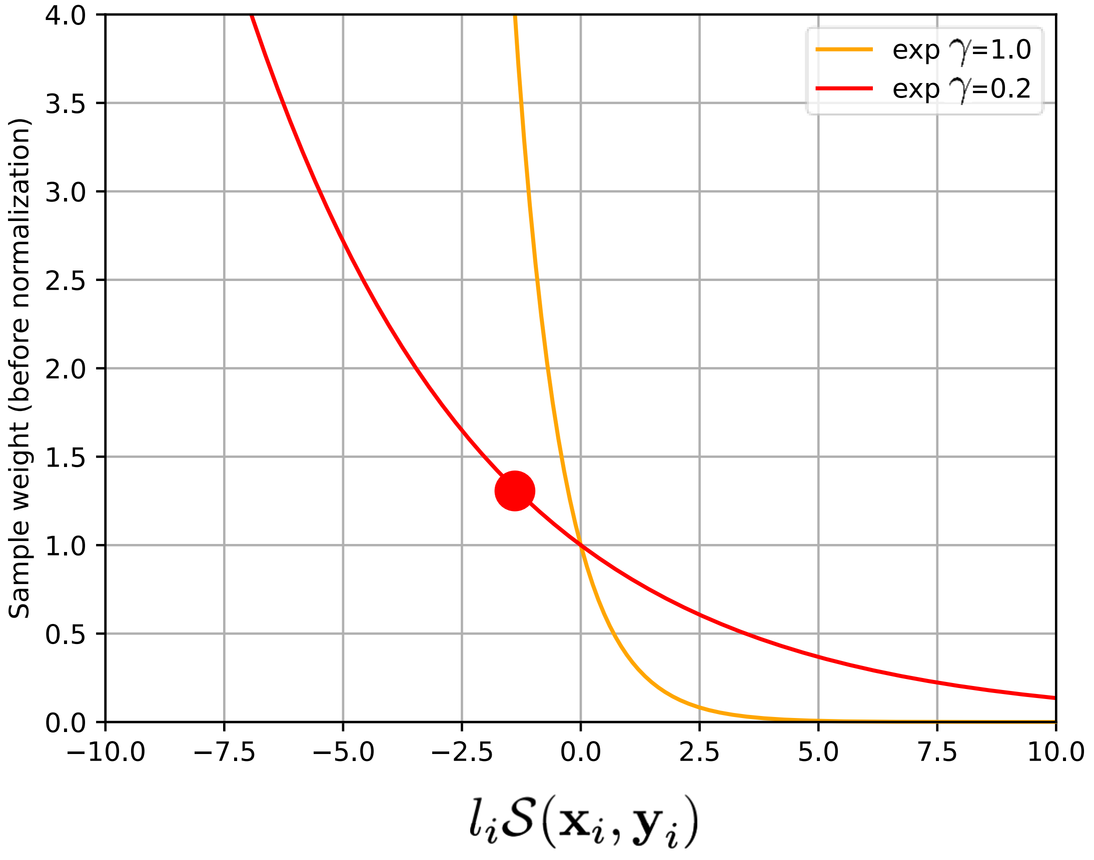


$$\mathcal{L}_{\text{Adaboost}} = \sum_{i=1}^{N} \exp\left[-\gamma l_i \mathcal{S}(\mathbf{x}_i, \mathbf{y}_i)\right]$$

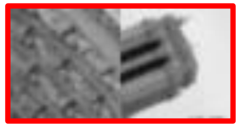


Wrongly classified example

Choosing loss function and margin

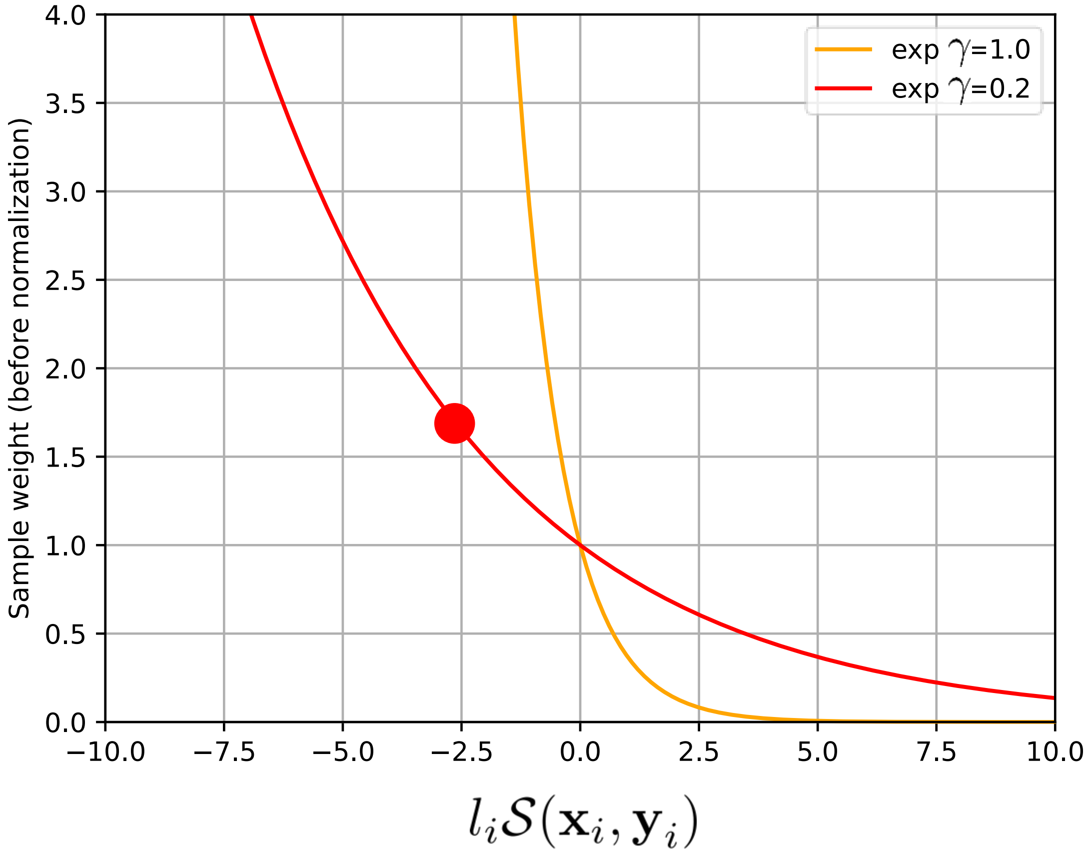


$$\mathcal{L}_{\text{Adaboost}} = \sum_{i=1}^{N} \exp\left[-\gamma l_i \mathcal{S}(\mathbf{x}_i, \mathbf{y}_i)\right]$$

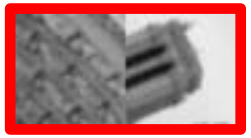


Wrongly classified example

Choosing loss function and margin

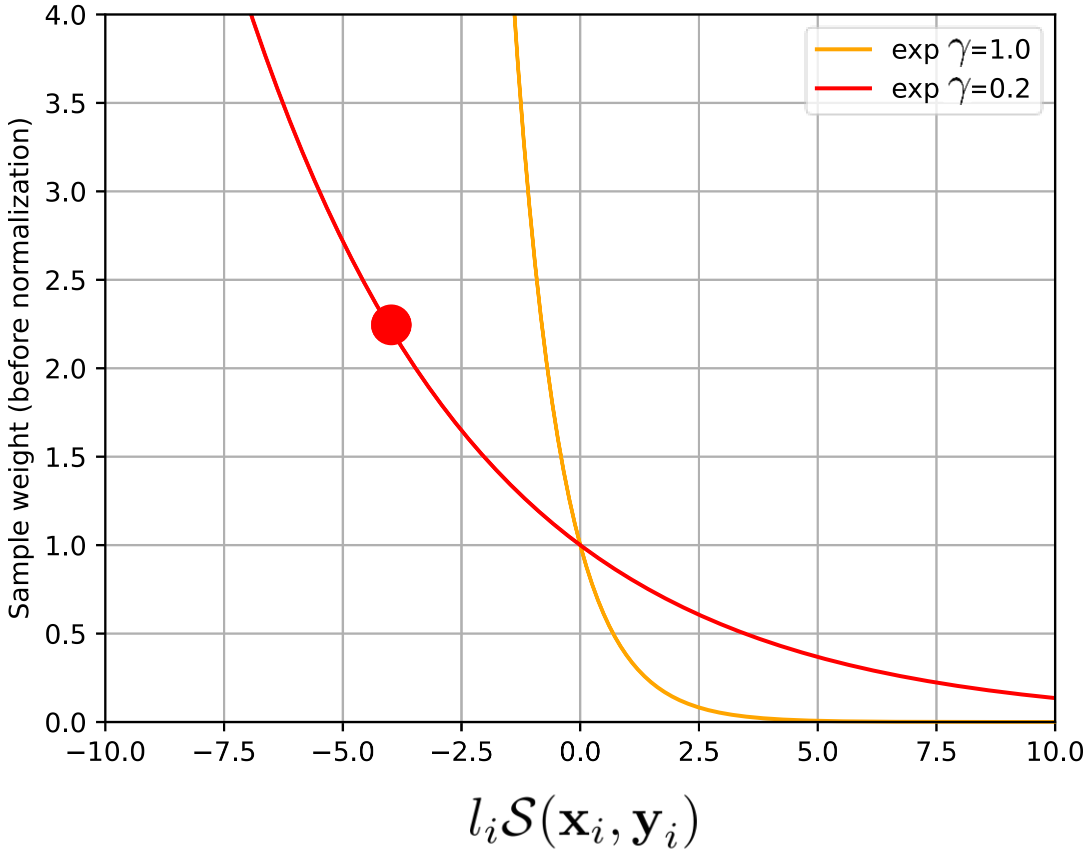


$$\mathcal{L}_{\text{Adaboost}} = \sum_{i=1}^{N} \exp\left[-\gamma l_i \mathcal{S}(\mathbf{x}_i, \mathbf{y}_i)\right]$$

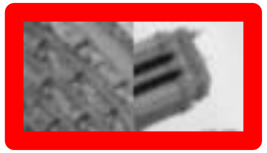


Wrongly classified example

Choosing loss function and margin

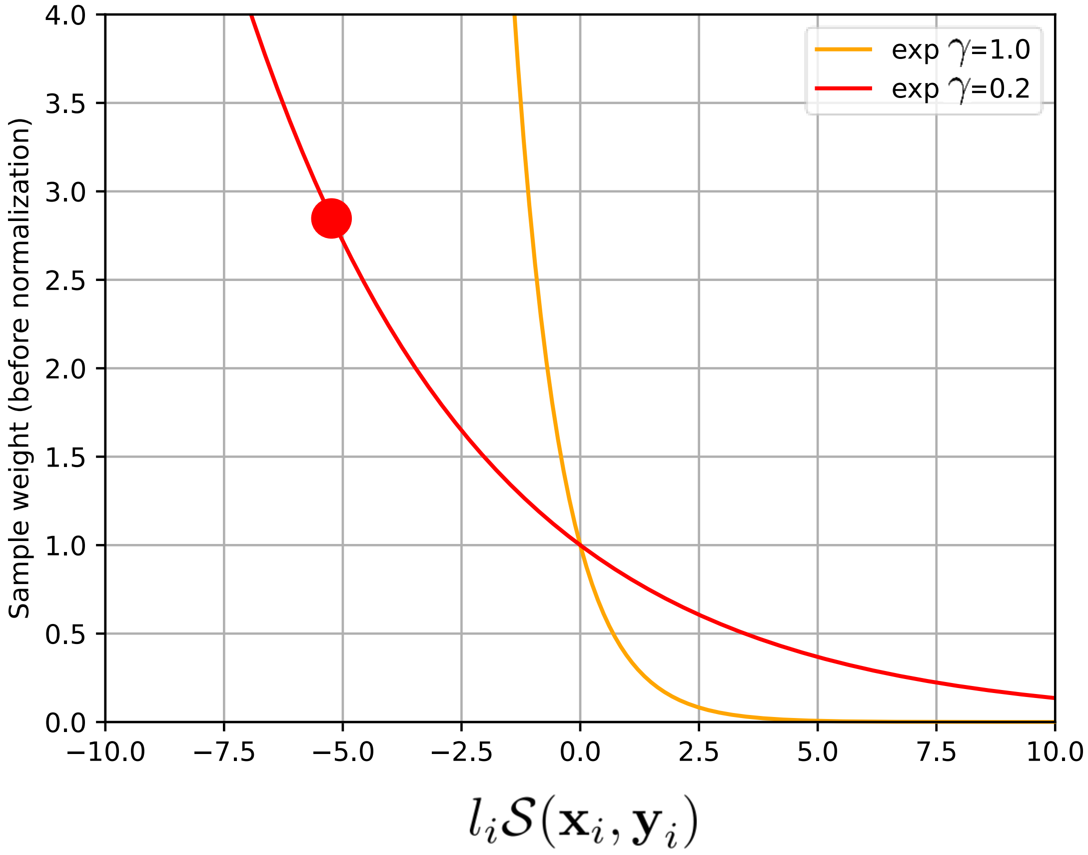


$$\mathcal{L}_{\text{Adaboost}} = \sum_{i=1}^{N} \exp\left[-\gamma l_i \mathcal{S}(\mathbf{x}_i, \mathbf{y}_i)\right]$$

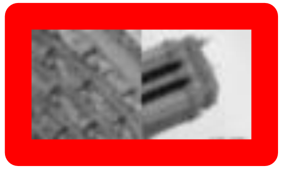


Wrongly classified example

Choosing loss function and margin

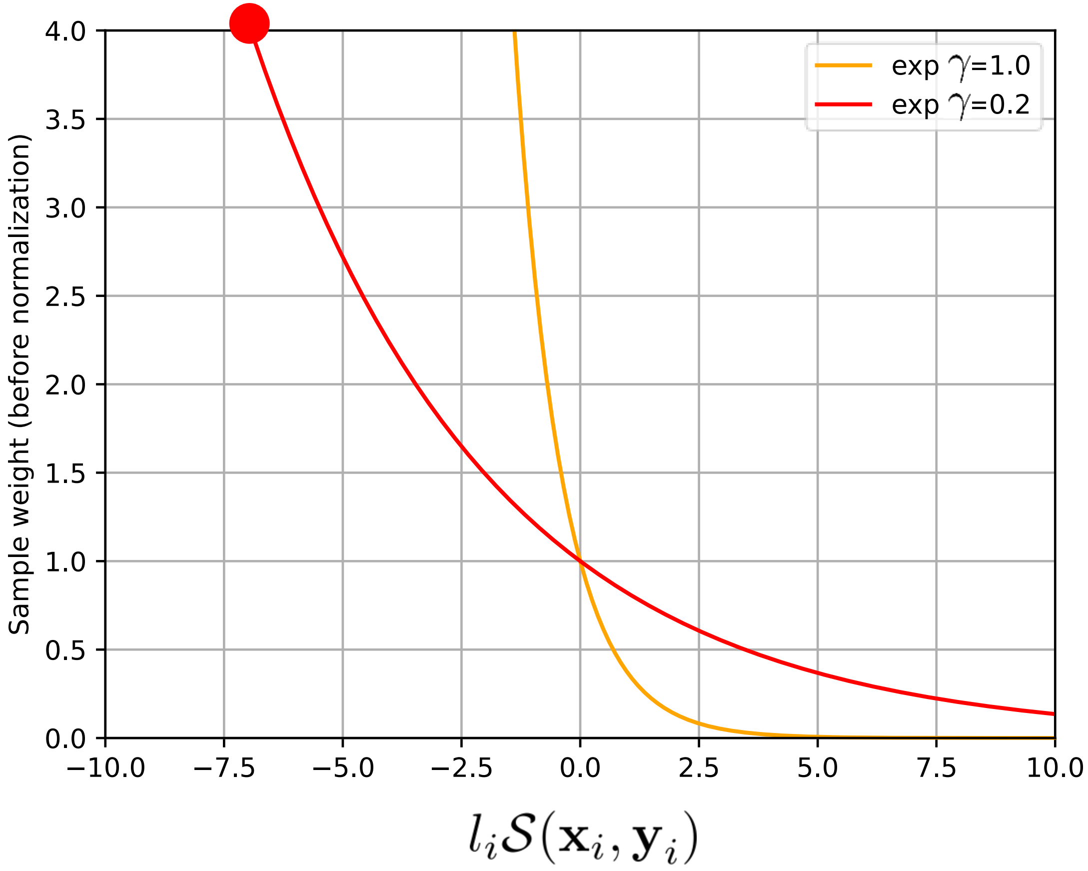


$$\mathcal{L}_{\text{Adaboost}} = \sum_{i=1}^{N} \exp\left[-\gamma l_i \mathcal{S}(\mathbf{x}_i, \mathbf{y}_i)\right]$$



Wrongly classified example

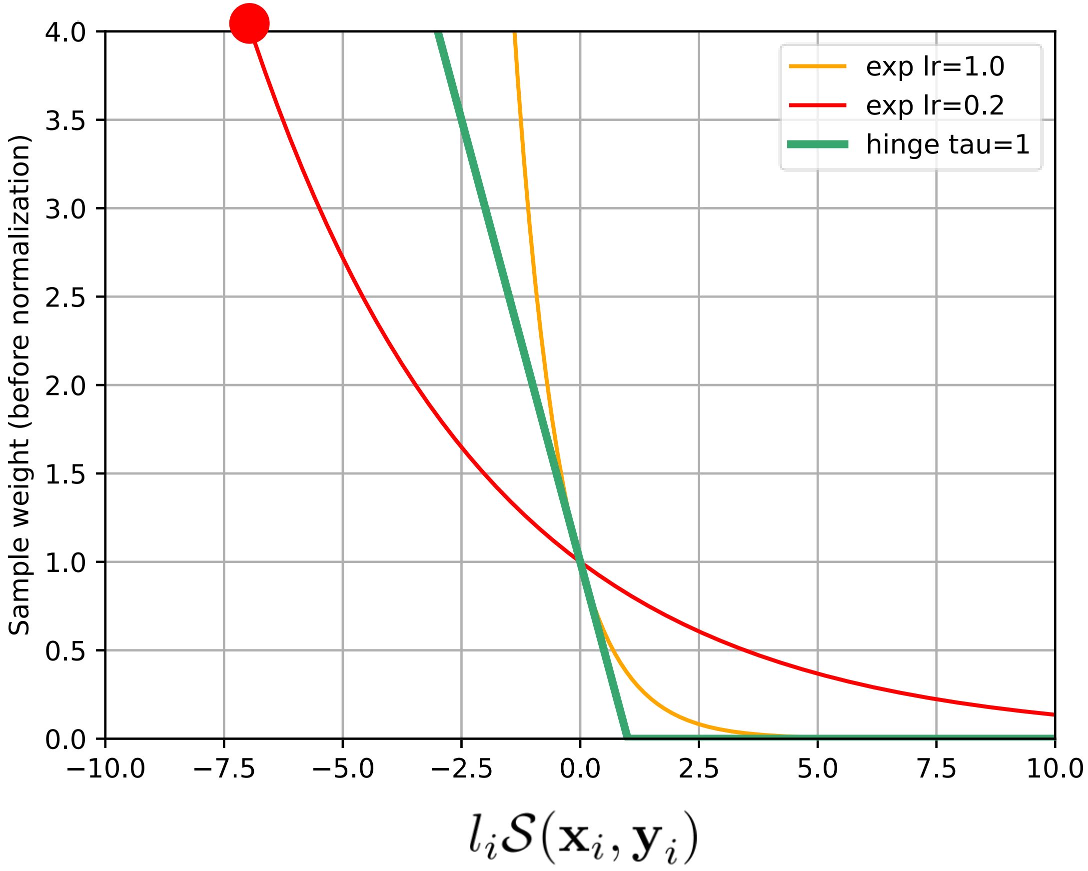
Choosing loss function and margin



$$\mathcal{L}_{\text{Adaboost}} = \sum_{i=1}^{N} \exp\left[-\gamma l_i \mathcal{S}(\mathbf{x}_i, \mathbf{y}_i)\right]$$

Wrongly classified example

Choosing loss function and margin



$$\mathcal{L}_{\text{Adaboost}} = \sum_{i=1}^{N} \underline{\exp}\left[-\gamma l_i \mathcal{S}(\mathbf{x}_i, \mathbf{y}_i)\right]$$

Outliers gain weight exponentially

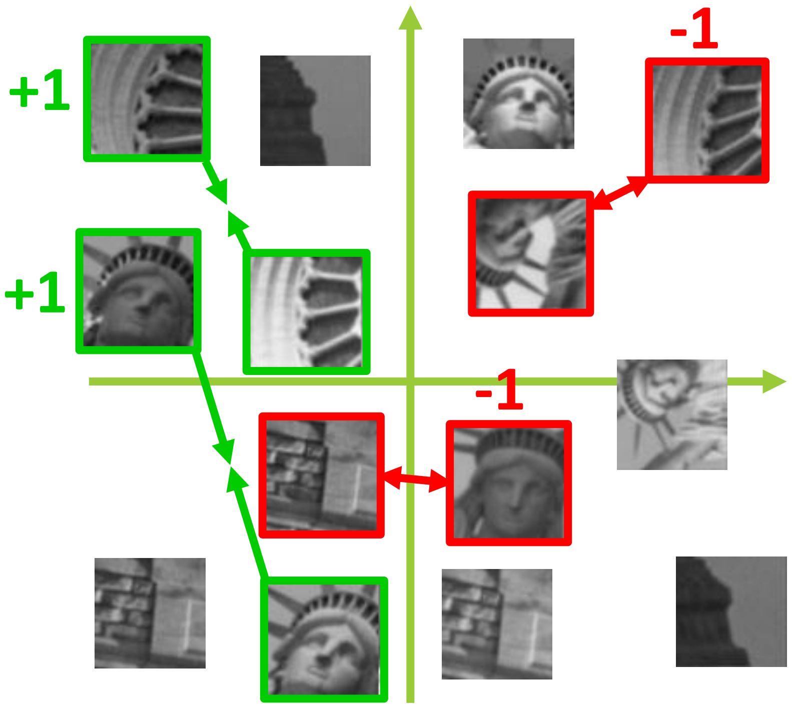
$$\mathcal{L}_{\text{Hinge}} = \sum_{i=1}^{N} \underline{\max\left[0, \tau\right.} - l_i \mathcal{S}(\mathbf{x}_i, \mathbf{y}_i)\left.\right]$$

Hinge (contrastive) loss is more robust

Wrongly classified example

Triplet ranking loss VS contrastive loss

Contrastive pairwise loss



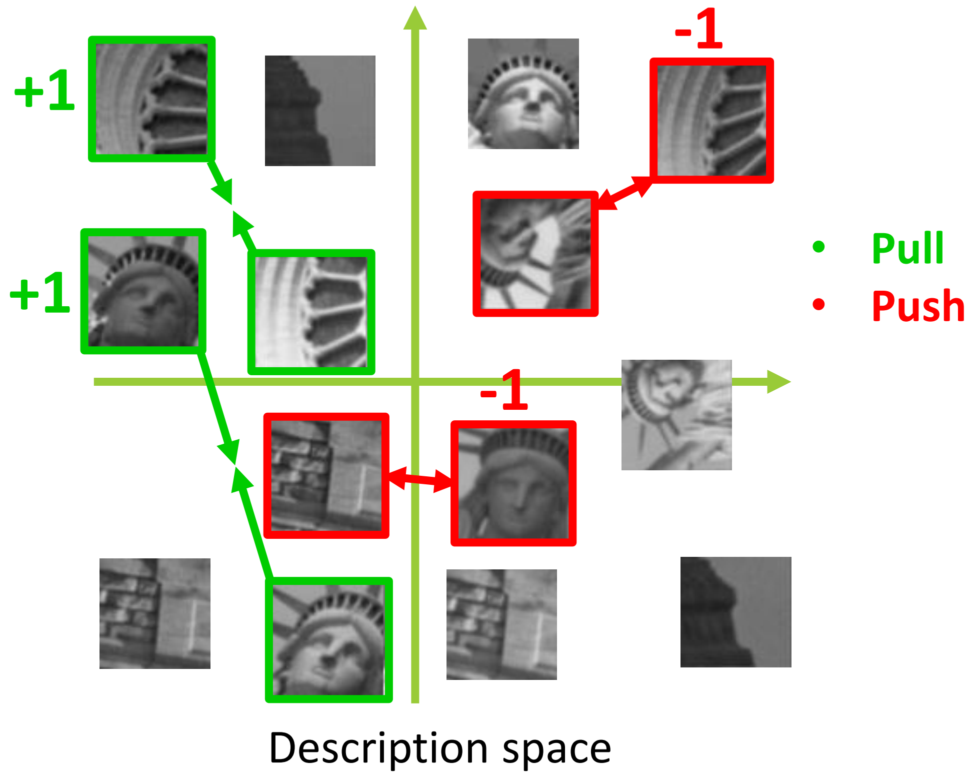
Description space

- Pull
- Push

$$\mathcal{L}_{\text{Contrastive}} = \sum_{i=1}^{N} \max\left[0, \tau - l_i \mathcal{S}(\mathbf{x}_i, \mathbf{y}_i)\right]$$

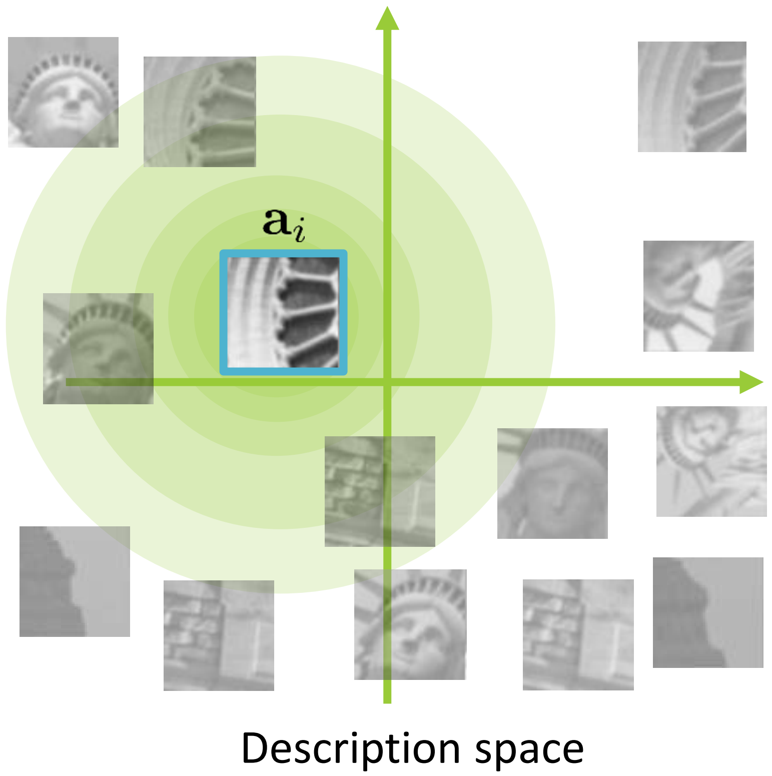
Triplet ranking loss VS contrastive loss

Contrastive pairwise loss



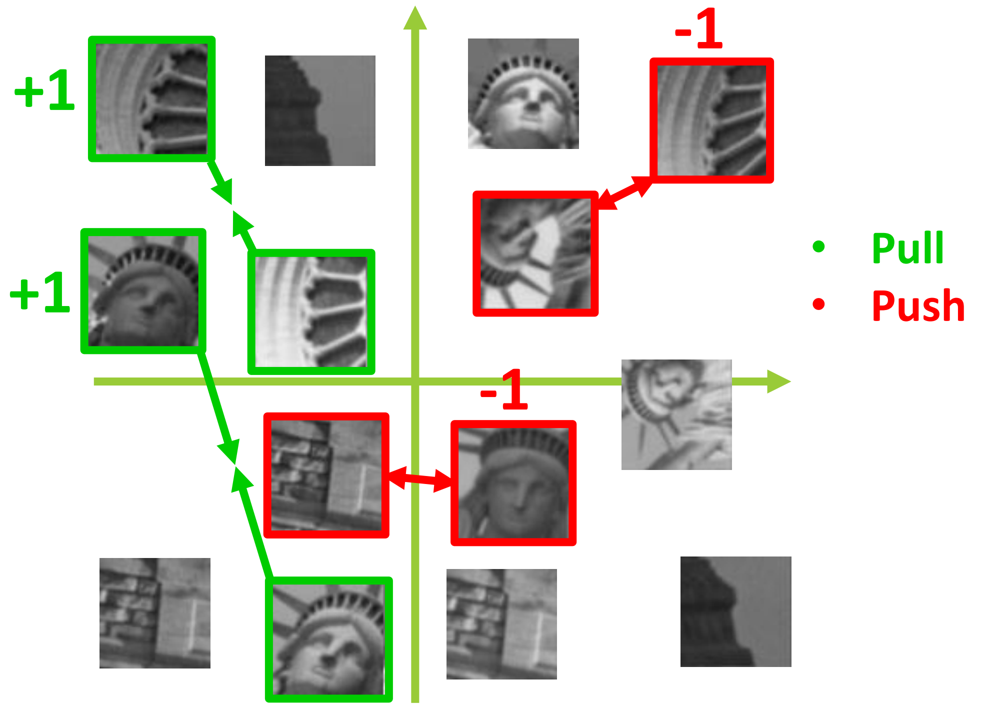
$$\mathcal{L}_{\text{Contrastive}} = \sum_{i=1}^{N} \max\left[0, \tau - l_i \mathcal{S}(\mathbf{x}_i, \mathbf{y}_i)\right]$$

Triplet ranking loss (TRL)



Triplet ranking loss VS contrastive loss

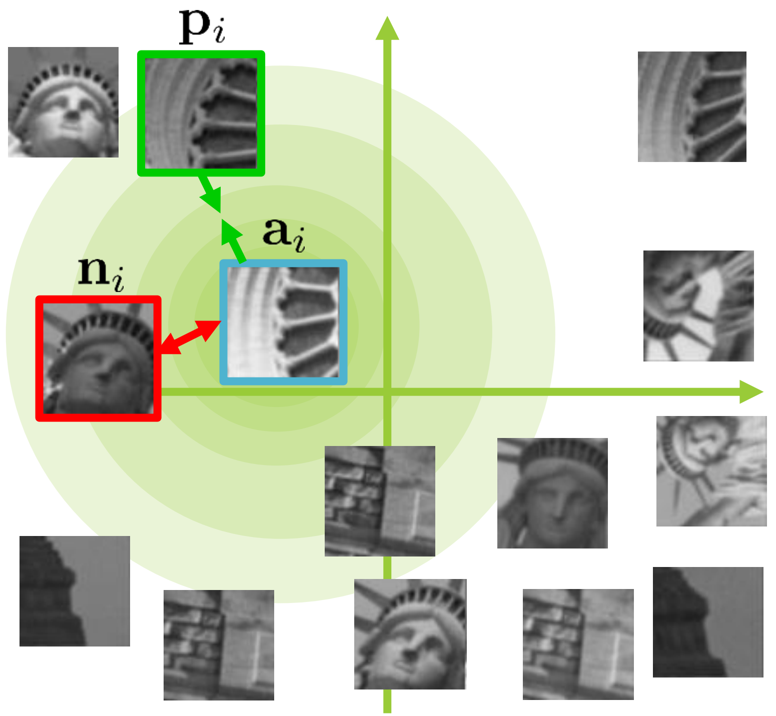
Contrastive pairwise loss



Description space

$$\mathcal{L}_{\text{Contrastive}} = \sum_{i=1}^{N} \max\left[0, \tau - l_i \mathcal{S}(\mathbf{x}_i, \mathbf{y}_i)\right]$$

Triplet ranking loss (TRL)



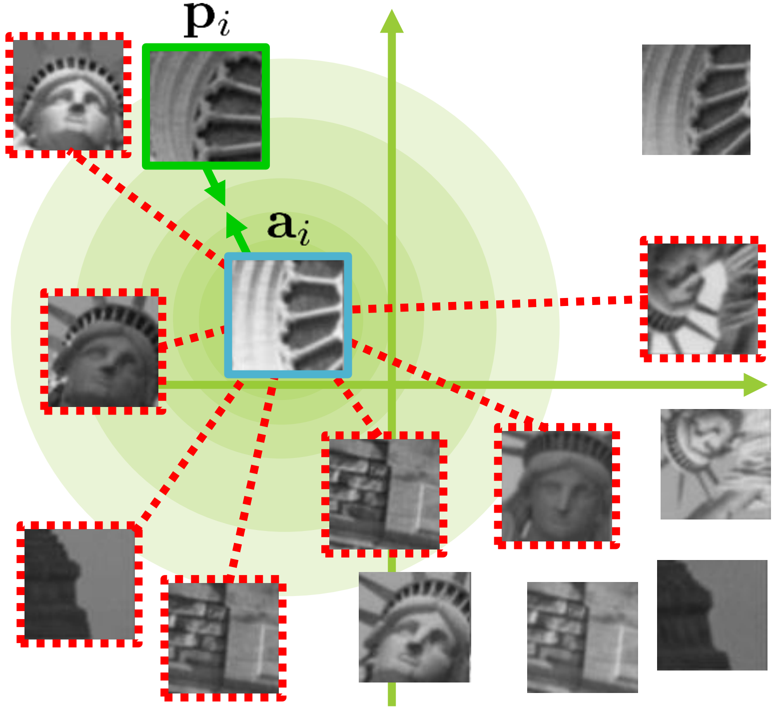
Description space

$$\mathcal{L}_{\text{TLR}} = \sum_{i=1}^{N} \max\left[0, \tau - \underline{\mathcal{S}\left(\mathbf{a}_i, \mathbf{p}_i\right)} + \underline{\mathcal{S}\left(\mathbf{a}_i, \mathbf{n}_i\right)}\right]$$

Hard Negative Mining (HNM) and anchor swap

- Select meaningful triplets is very important!

Triplet ranking loss (TRL)

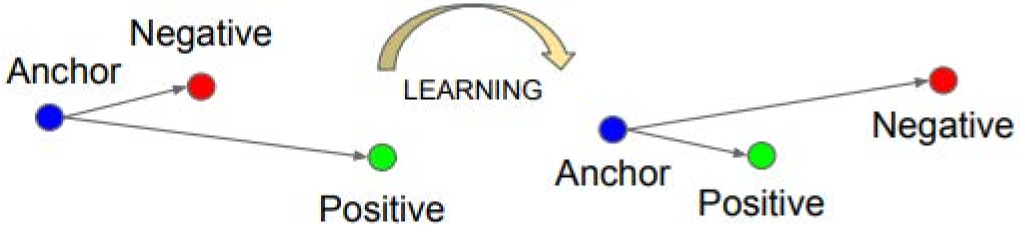


Description space

$$\mathcal{L}_{\mathrm{TLR}} = \sum_{i=1}^{N} \max\left[0, \tau - \mathcal{S}\left(\mathbf{a}_i, \mathbf{p}_i\right) + \mathcal{S}\left(\mathbf{a}_i, \mathbf{n}_i\right)\right]$$

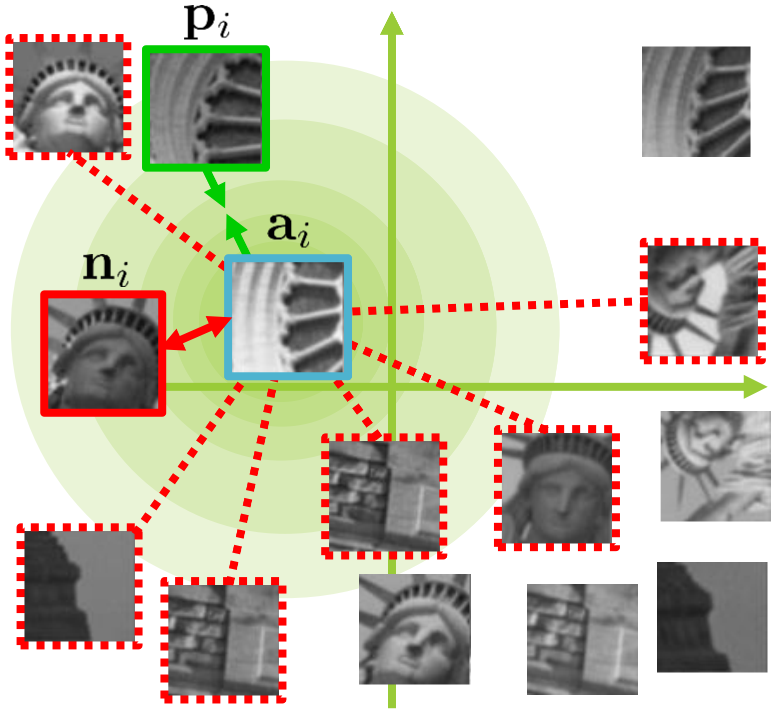
Hard Negative Mining (HNM) and anchor swap

- Select meaningful triplets is very important!
- We choose the hardest negative in a batch of 256 samples.
  - **Distance**: Hamming Norm of the current descriptors -> Progressively harder triplets
  - **Anchor swap**: if the negative is closer to the positive than to the anchor, **we swap them**



(Schroff, 2015)

Triplet ranking loss (TRL)



Description space

$$\mathcal{L}_{\mathrm{TLR}} = \sum_{i=1}^{N} \max\left[0, \tau - \underbrace{\mathcal{S}\left(\mathbf{a}_i, \mathbf{p}_i\right)} + \underbrace{\mathcal{S}\left(\mathbf{a}_i, \mathbf{n}_i\right)}\right]$$

BAD Overview

In BAD we use a greedy scheme like in boosting, but we now select the features that directly optimize the loss function:

Algorithm 4 Feature selection algorithm

Input: $\mathcal{P}$, patches with 3D point class label

Output: $\mathbf{h} = [h_1, ..., h_K]$

BAD Overview

In BAD we use a greedy scheme like in boosting, but we now select the features that directly optimize the loss function:

Algorithm 4 Feature selection algorithm

Input: $\mathcal{P}$, patches with 3D point class label

Output: $\mathbf{h} = [h_1, ..., h_K]$

$\mathbf{h} := \emptyset$

for all $k = 1, ..., K$ **do**

Select one weak-descriptor
(thresholded average box measure)
h_k

$\mathbf{h} := \mathbf{h} \cup h_k$

end for

return $\mathbf{h}$

BAD Overview

In BAD we use a greedy scheme like in boosting, but we now select the features that directly optimize the loss function:

Algorithm 4 Feature selection algorithm

Input: $\mathcal{P}$, patches with 3D point class label

Output: $\mathbf{h} = [h_1, ..., h_K]$

$\mathbf{h} := \emptyset$

for all $k = 1, ..., K$ **do**

$\mathcal{T} := \text{sampleTriplets}(\mathcal{P}, N)$

$\mathcal{F} := \text{sampleFeatures}()$

$\mathcal{L}^* := \infty, \theta^* := -\infty$

Select the best measure f^* from $\mathcal{F}$ with best threshold θ^*

$h_k := \{+1 \text{ if } f^*(\mathbf{x}) \leq \theta^*; -1 \text{ otherwise}\}$

$\mathbf{h} := \mathbf{h} \cup h_k$

end for

return $\mathbf{h}$

BAD Overview

In BAD we use a greedy scheme like in boosting, but we now select the features that directly optimize the loss function:

Algorithm 4 Feature selection algorithm

Input: $\mathcal{P}$, patches with 3D point class label

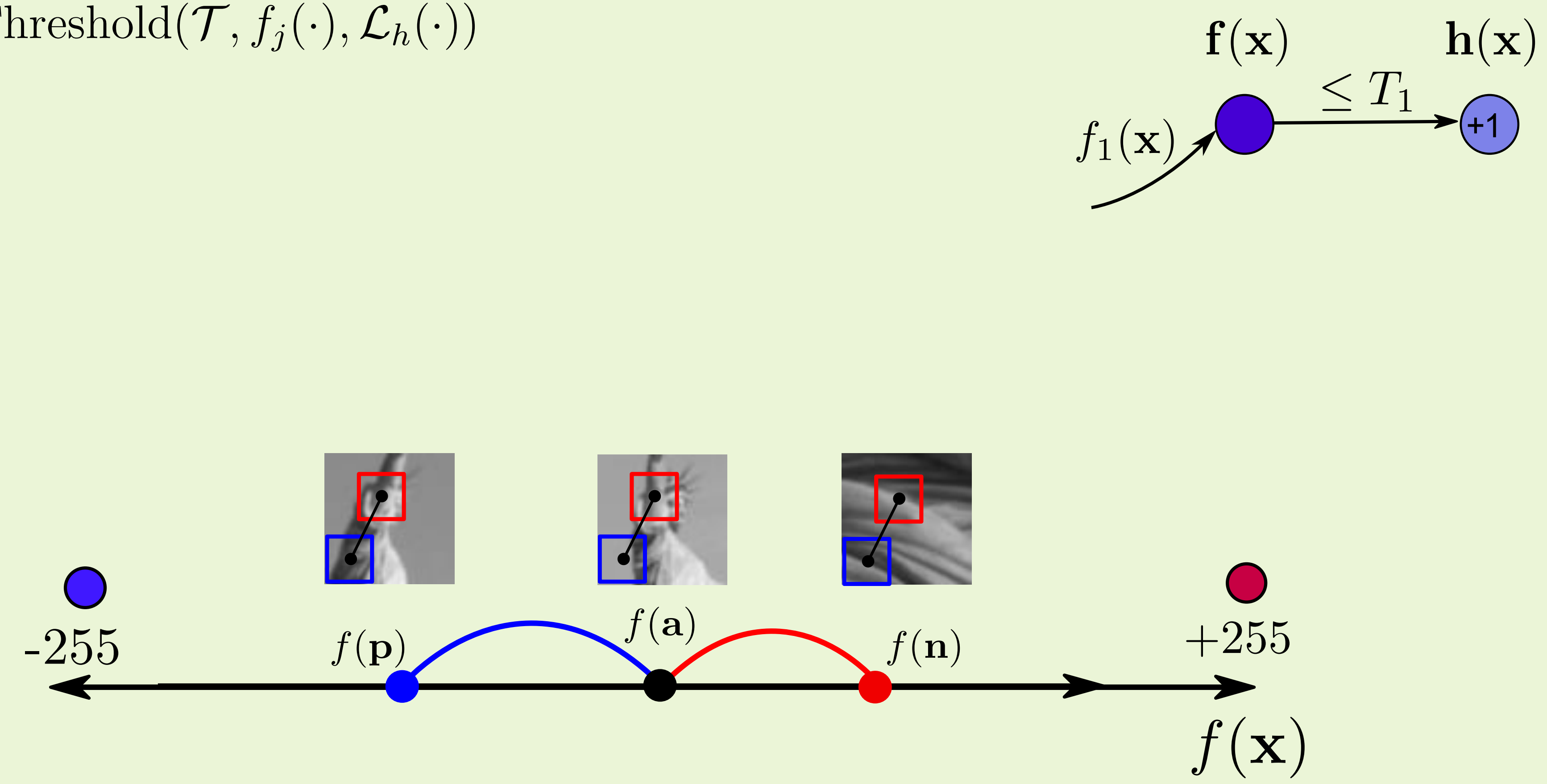
Output: $\mathbf{h} = [h_1, ..., h_K]$

$\mathbf{h} := \emptyset$
for all $k = 1, ..., K$ **do**
  $\mathcal{T} := \text{sampleTriplets}(\mathcal{P}, N)$
  $\mathcal{F} := \text{sampleFeatures}()$
  $\mathcal{L}^* := \infty, \theta^* := -\infty$
  for all $j = 1, ..., J$ **do**
    $\theta_j, \mathcal{L}_j := \text{findThreshold}(\mathcal{T}, f_j(\cdot), \mathcal{L}_h(\cdot))$
    if $\mathcal{L}_j < \mathcal{L}^*$ **then**
      $\mathcal{L}^* := \mathcal{L}_j;\ f^* := f_j;\ \theta^* := \theta_j$
    end if
  end for
  $h_k := \{+1 \text{ if } f^*(\mathbf{x}) \leq \theta^*; -1 \text{ otherwise}\}$
  $\mathbf{h} := \mathbf{h} \cup h_k$
end for
return $\mathbf{h}$

Find the threshold that minimizes the loss

We propose a general algorithm to select a threshold for the Box Average Difference (BAD) measure. It supports any feature selection loss function $\mathcal{L}_{\text{fs}}$

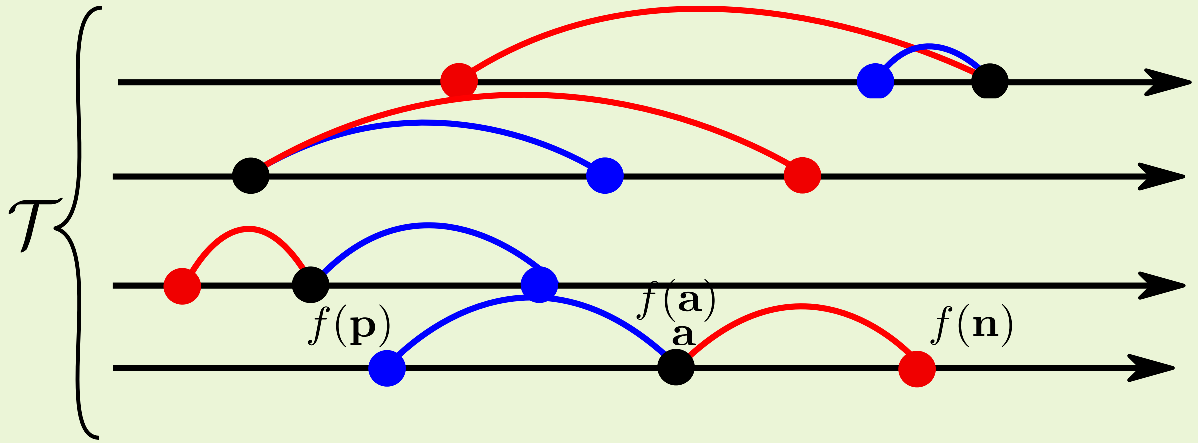
$$\theta_j, \mathcal{L}_j := \text{findThreshold}(\mathcal{T}, f_j(\cdot), \mathcal{L}_h(\cdot))$$



Find the threshold that minimizes the loss

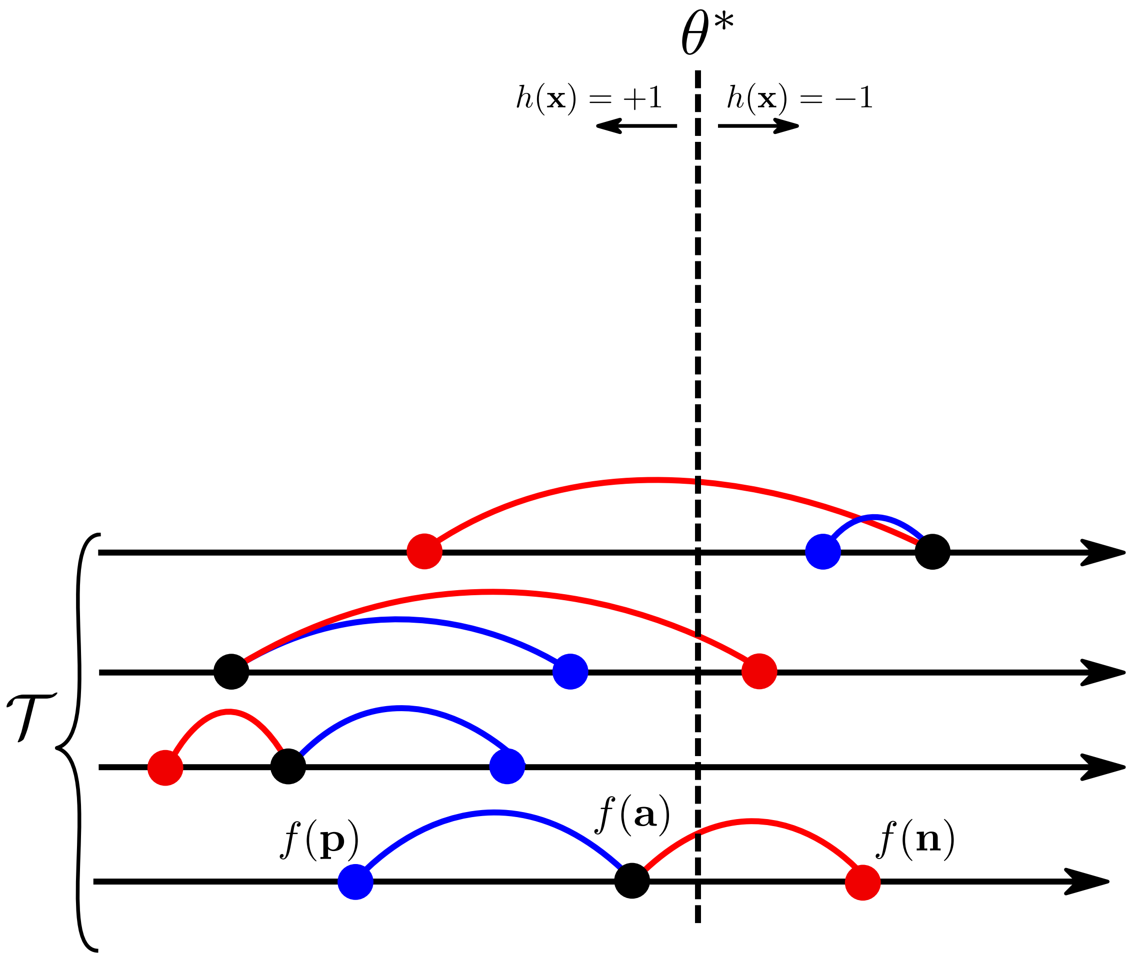
We propose a general algorithm to select a threshold for the Box Average Difference (BAD) measure. It supports any feature selection loss function $\mathcal{L}_{\text{fs}}$

$$\theta_j, \mathcal{L}_j := \text{findThreshold}(\mathcal{T}, f_j(\cdot), \mathcal{L}_h(\cdot))$$



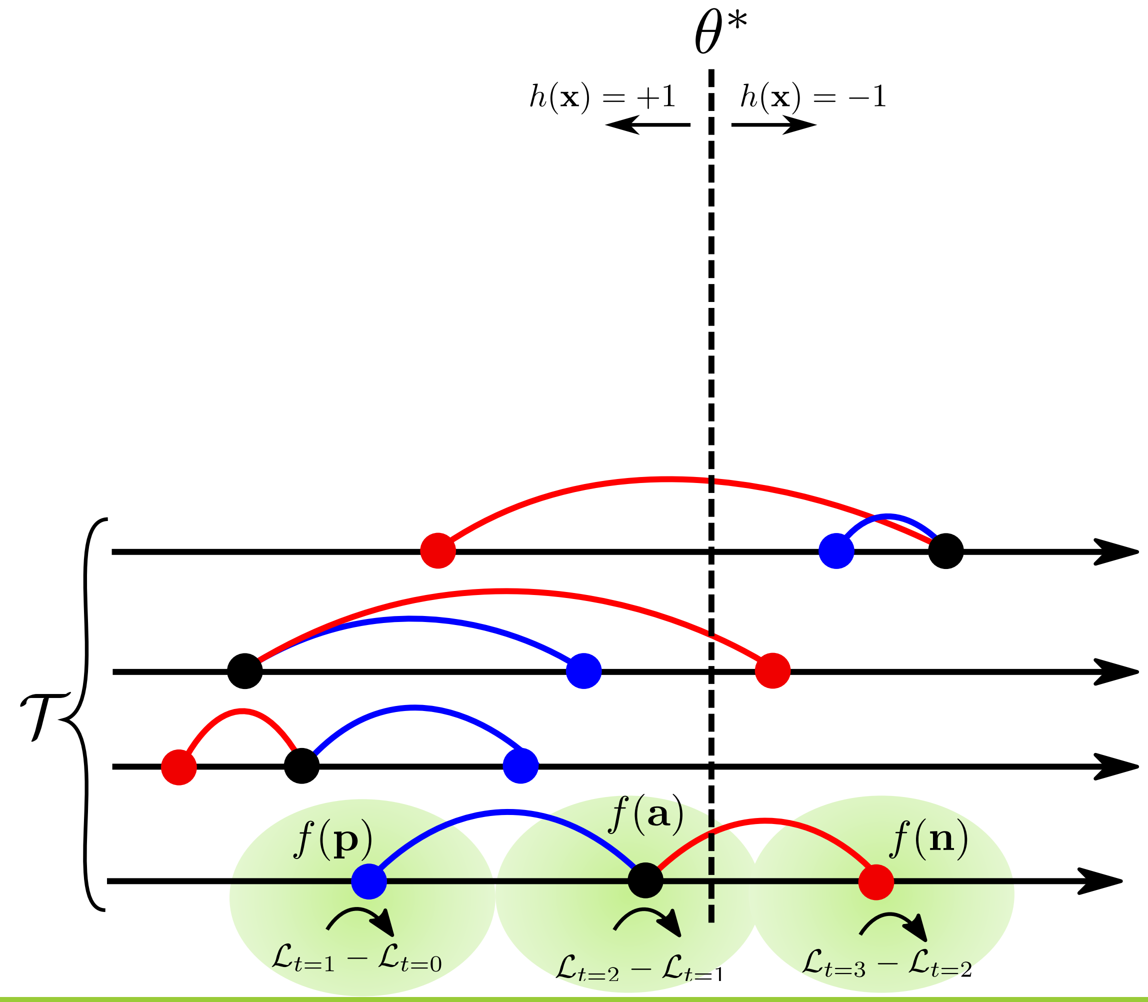
Find the threshold that minimizes the loss

We propose a general algorithm to select a threshold for the Box Average Difference (BAD) measure. It supports any feature selection loss function $\mathcal{L}_{\text{fs}}$



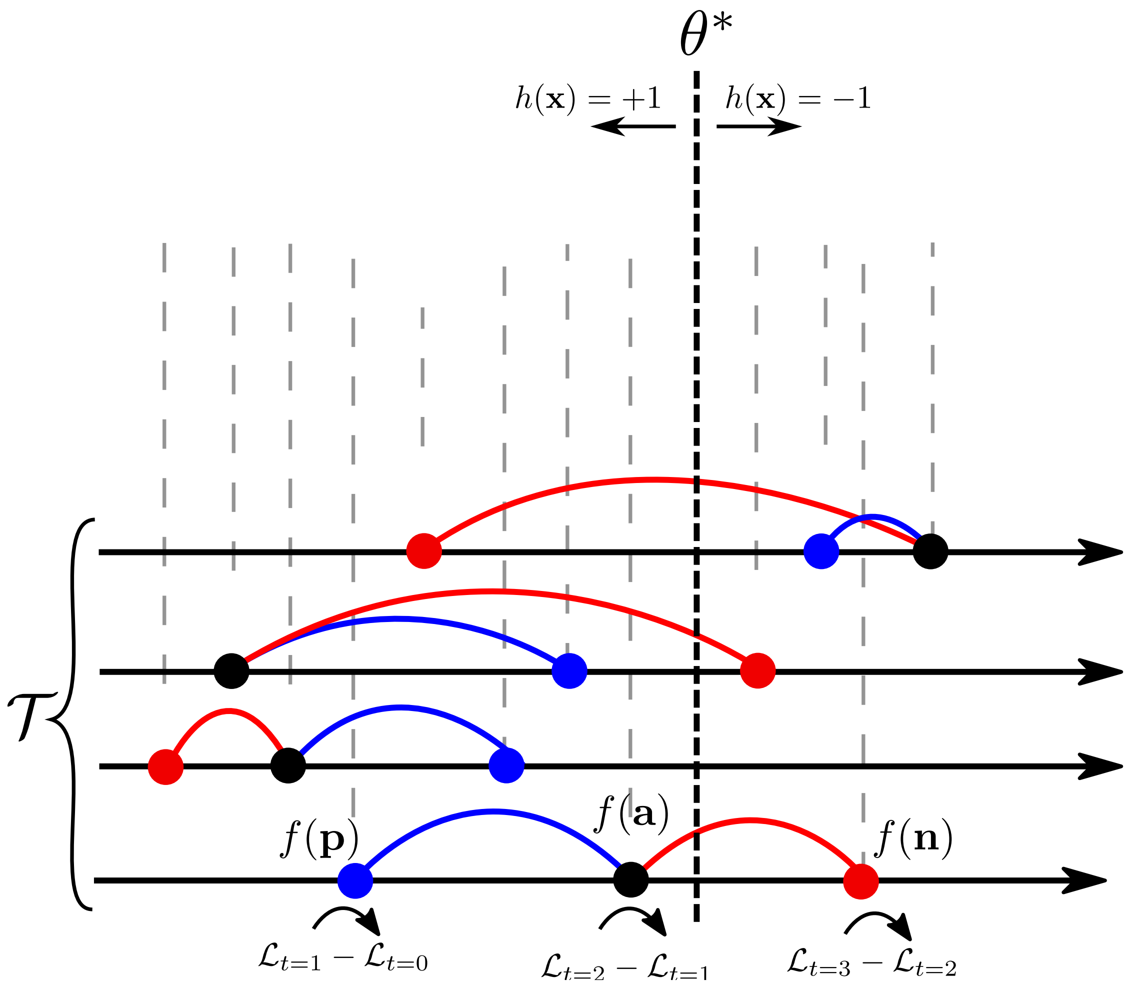
Find the threshold that minimizes the loss

We propose a general algorithm to select a threshold for the Box Average Difference (BAD) measure. It supports any feature selection loss function $\mathcal{L}_{\text{fs}}$



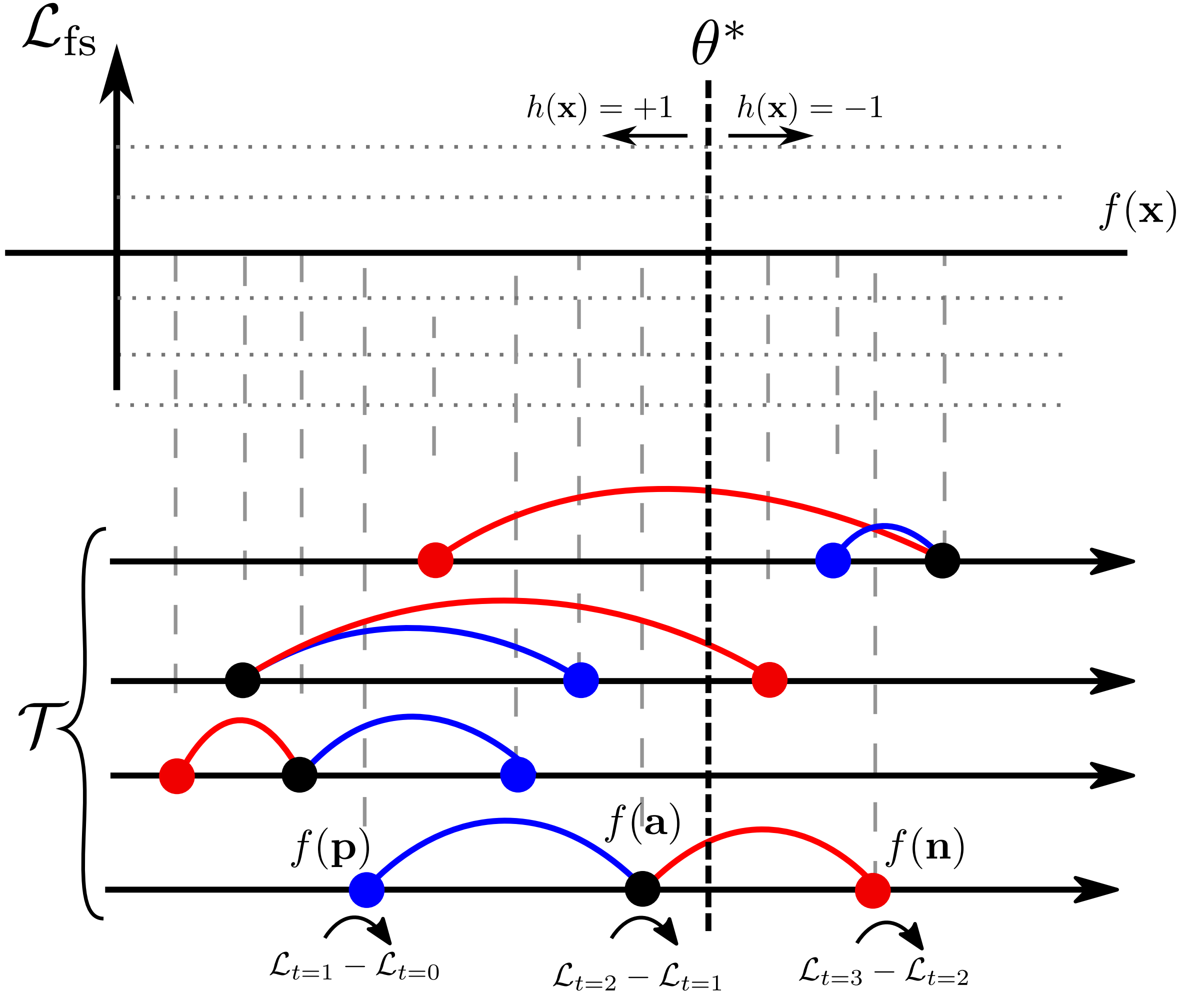
Find the threshold that minimizes the loss

We propose a general algorithm to select a threshold for the Box Average Difference (BAD) measure. It supports any feature selection loss function $\mathcal{L}_{\text{fs}}$



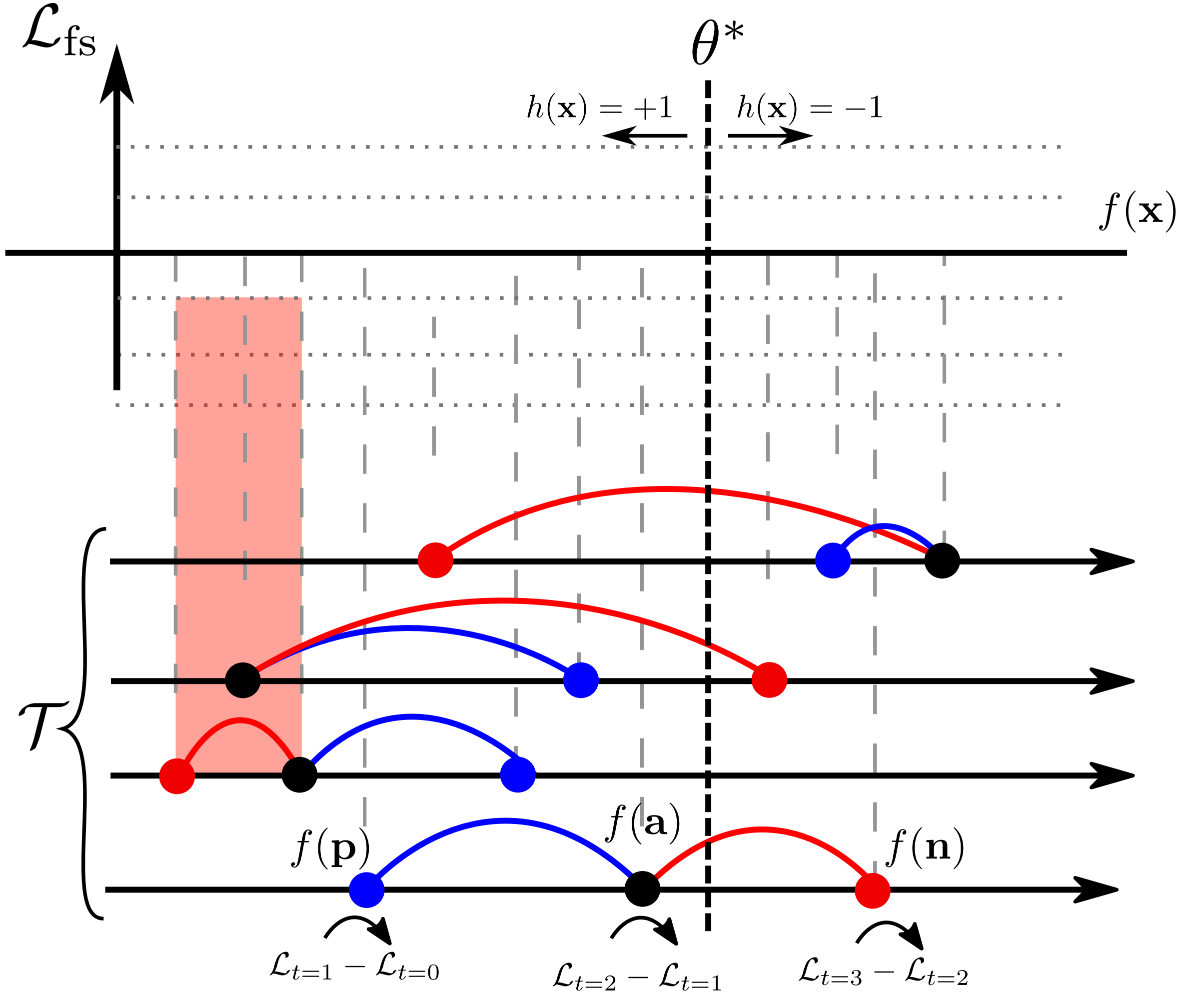
Find the threshold that minimizes the loss

We propose a general algorithm to select a threshold for the Box Average Difference (BAD) measure. It supports any feature selection loss function $\mathcal{L}_{\text{fs}}$



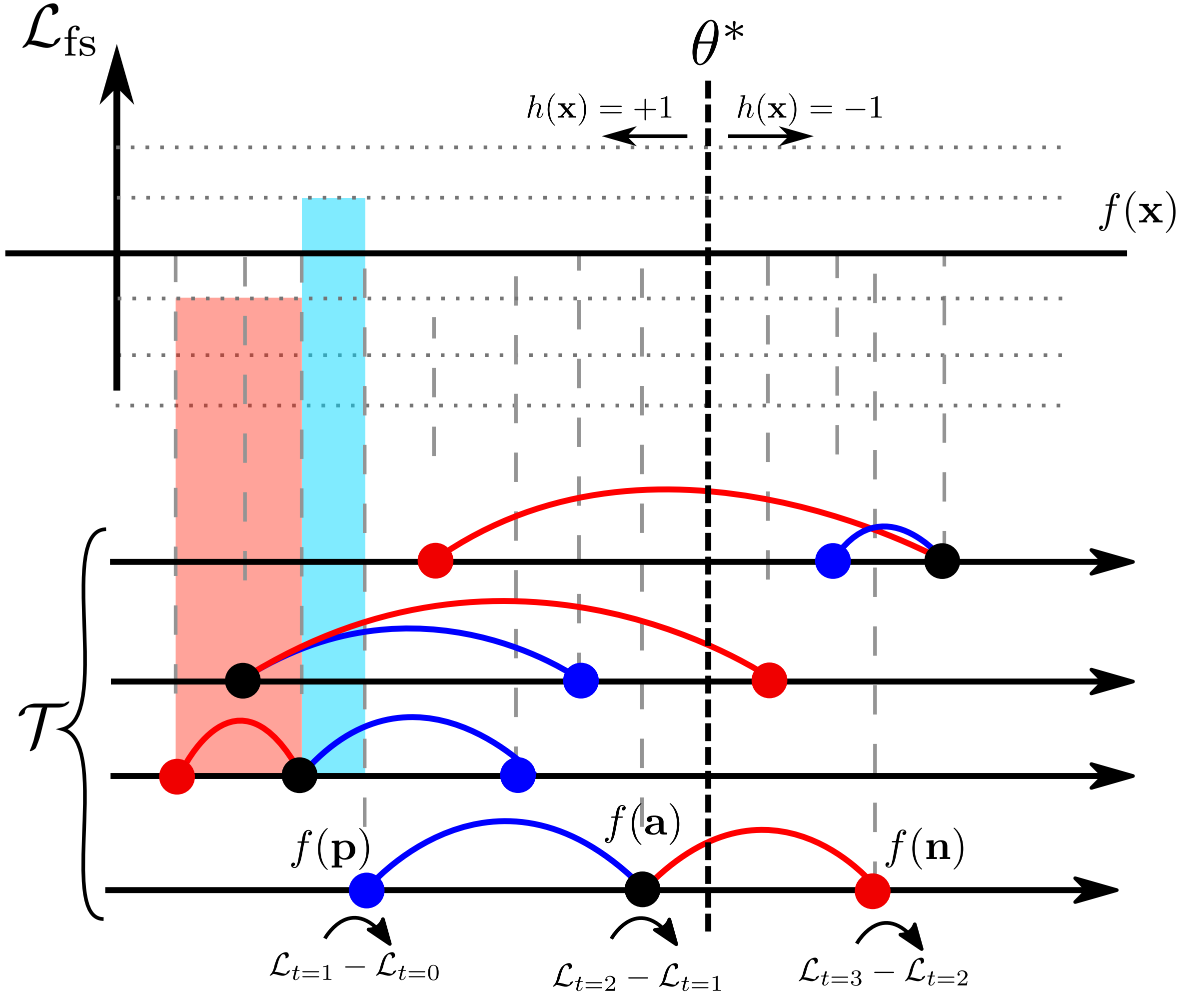
Find the threshold that minimizes the loss

We propose a general algorithm to select a threshold for the Box Average Difference (BAD) measure. It supports any feature selection loss function $\mathcal{L}_{\text{fs}}$



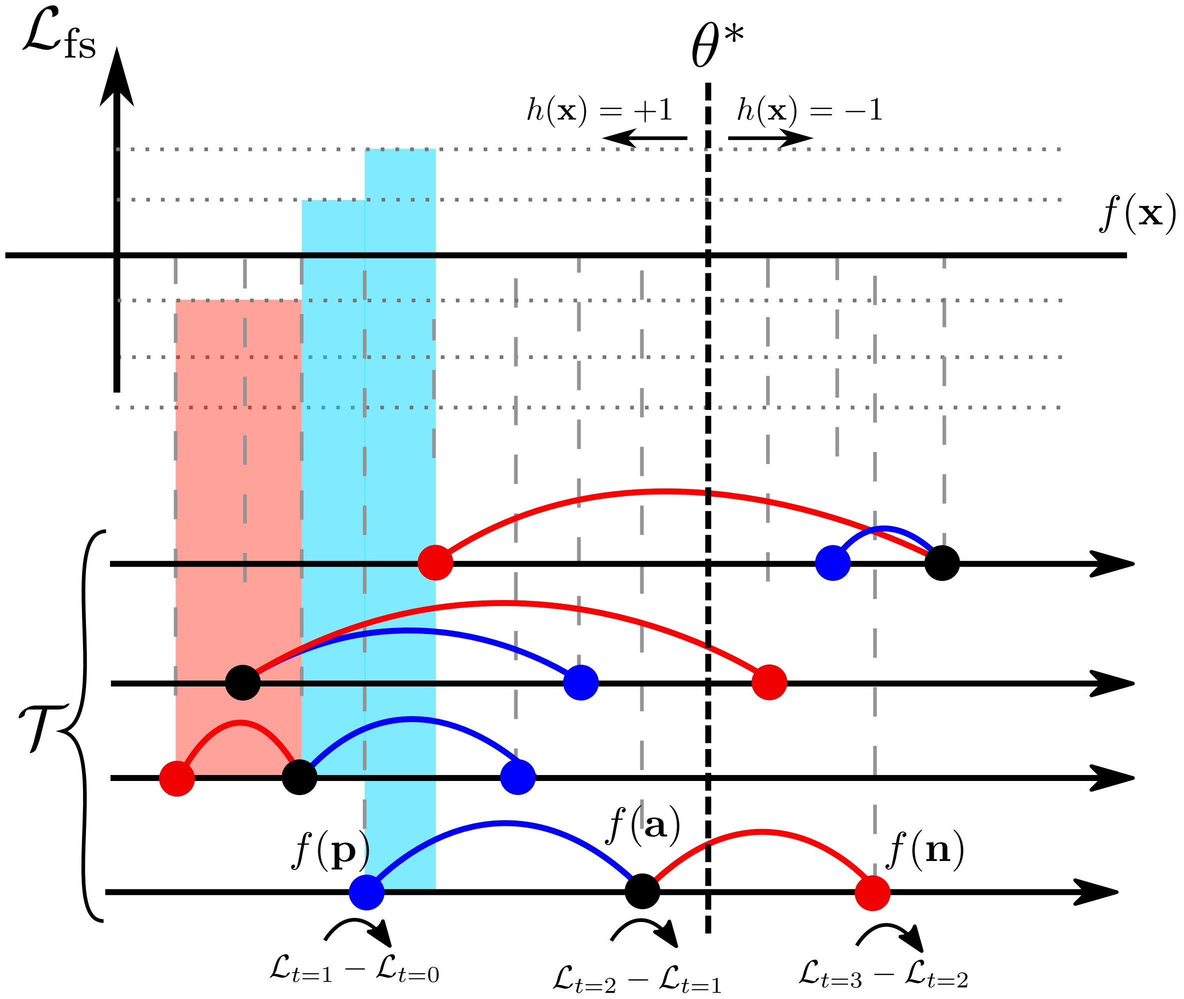
Find the threshold that minimizes the loss

We propose a general algorithm to select a threshold for the Box Average Difference (BAD) measure. It supports any feature selection loss function $\mathcal{L}_{\text{fs}}$



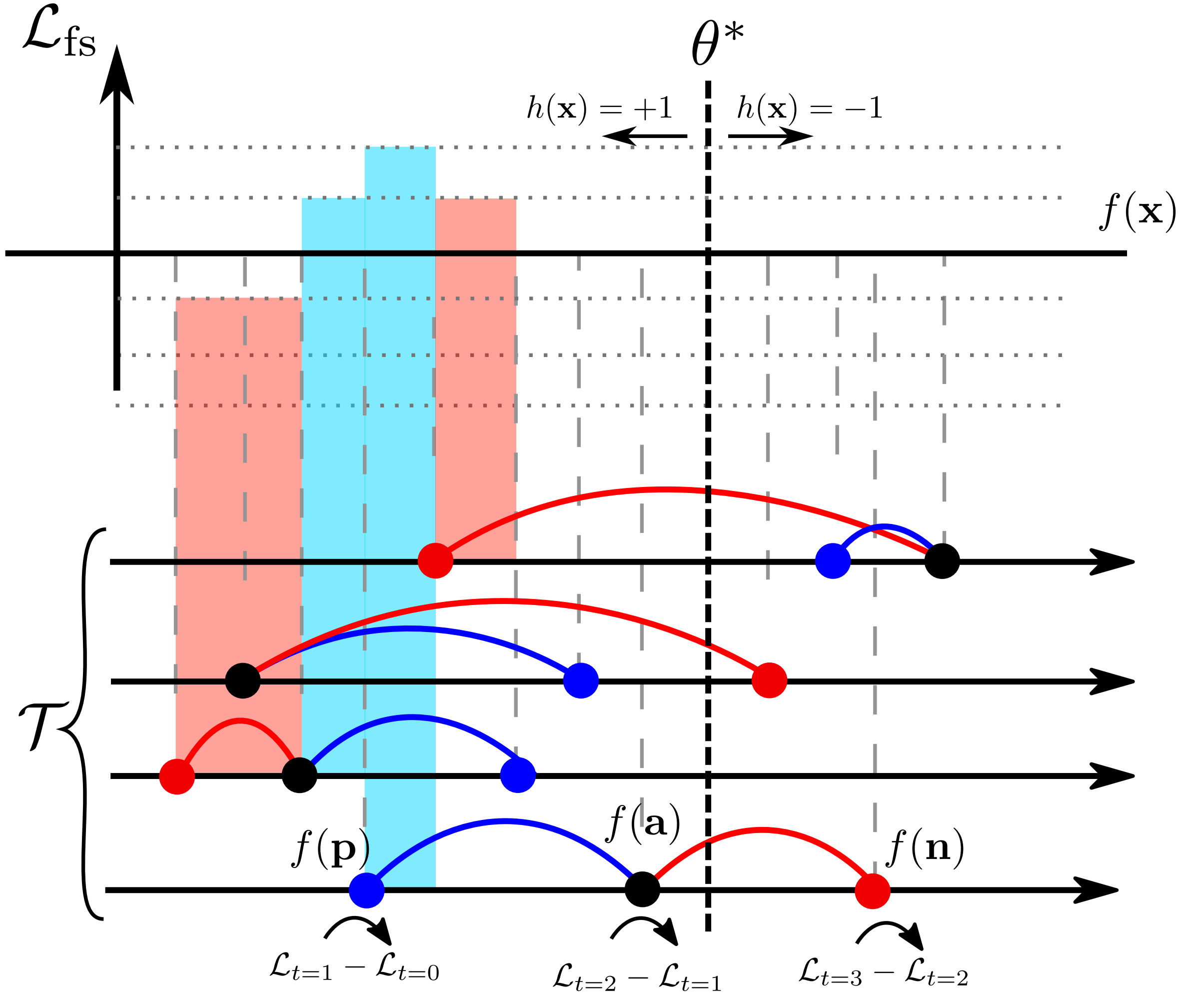
Find the threshold that minimizes the loss

We propose a general algorithm to select a threshold for the Box Average Difference (BAD) measure. It supports any feature selection loss function $\mathcal{L}_{\text{fs}}$



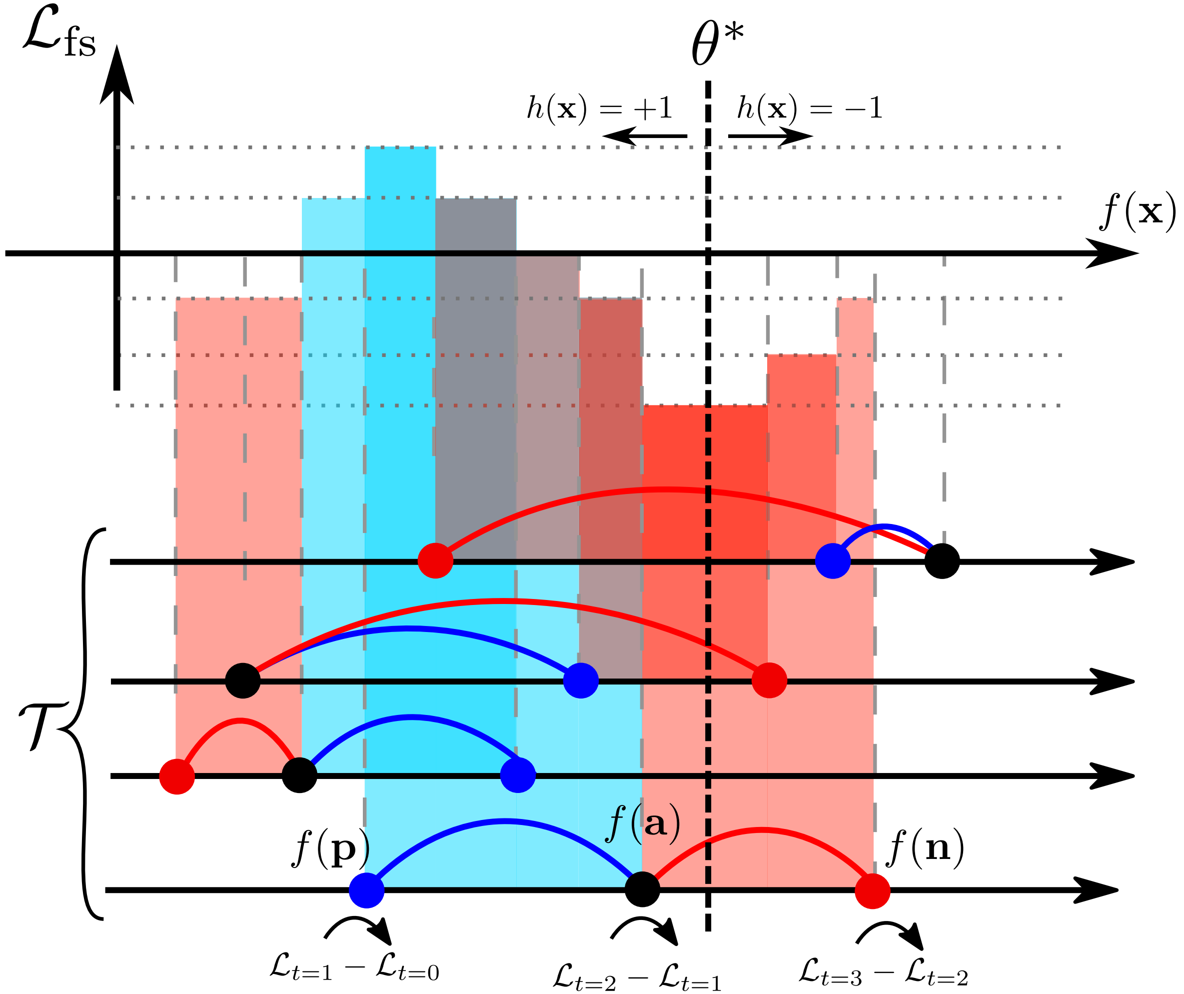
Find the threshold that minimizes the loss

We propose a general algorithm to select a threshold for the Box Average Difference (BAD) measure. It supports any feature selection loss function $\mathcal{L}_{\text{fs}}$



Find the threshold that minimizes the loss

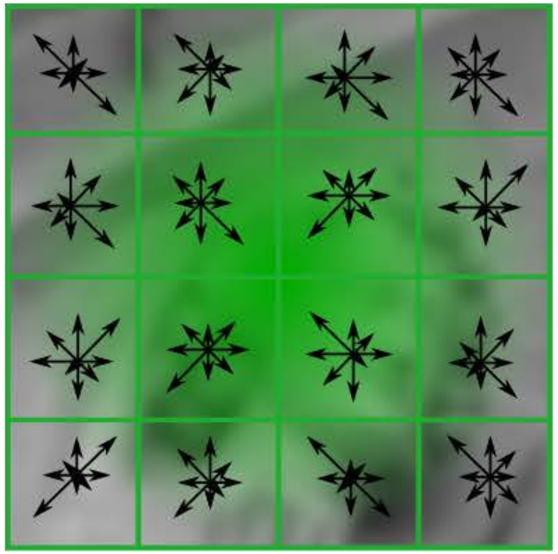
We propose a general algorithm to select a threshold for the Box Average Difference (BAD) measure. It supports any feature selection loss function $\mathcal{L}_{\text{fs}}$



HashSIFT

We studied how powerful is our training scheme to binarize a set of good features.

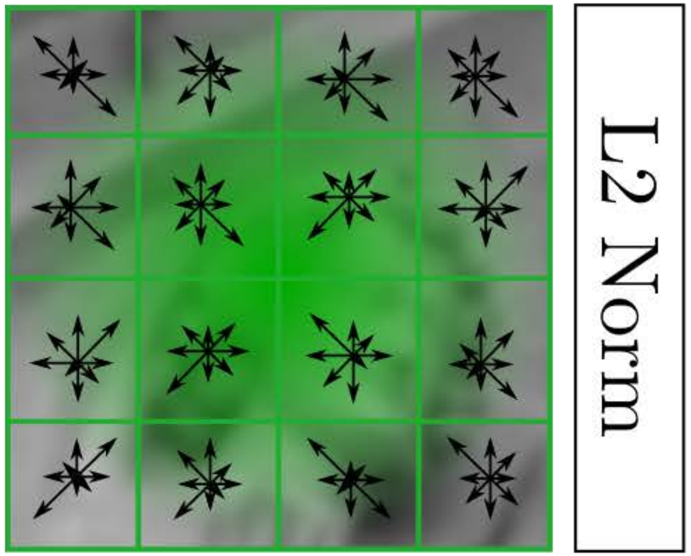
To this end we decided to binarize the Histogram of oriented gradients from SIFT:



HashSIFT

We studied how powerful is our training scheme to binarize a set of good features.

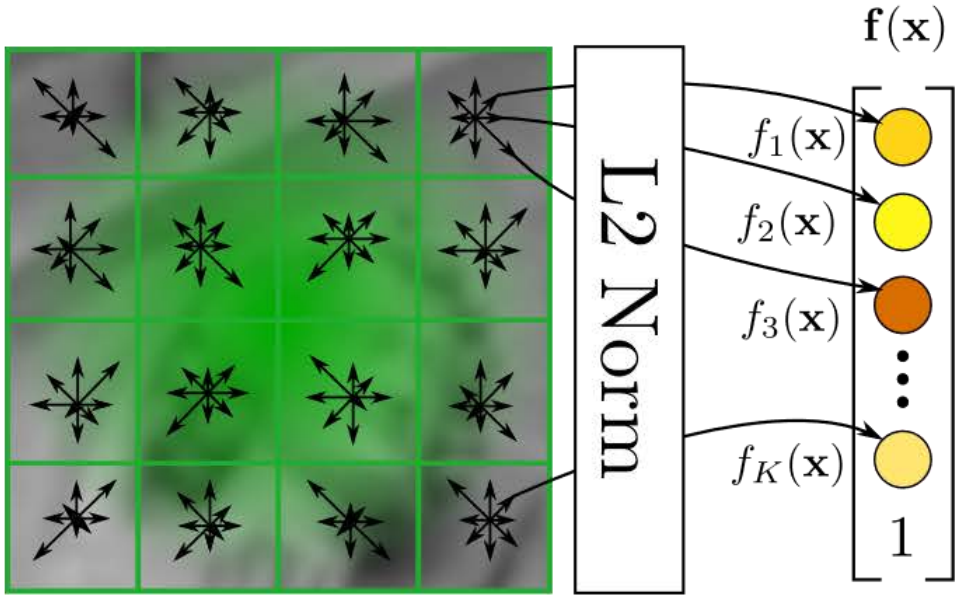
To this end we decided to binarize the Histogram of oriented gradients from SIFT:



HashSIFT

We studied how powerful is our training scheme to binarize a set of good features.

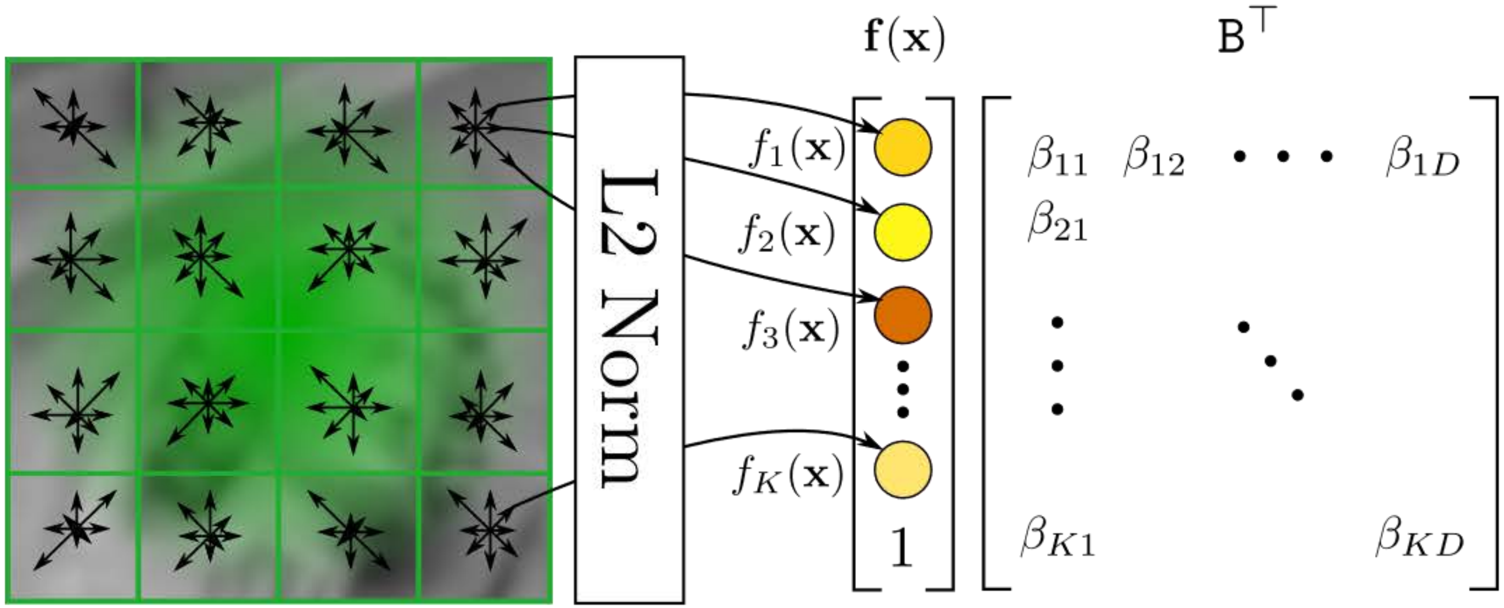
To this end we decided to binarize the Histogram of oriented gradients from SIFT:



HashSIFT

We studied how powerful is our training scheme to binarize a set of good features.

To this end we decided to binarize the Histogram of oriented gradients from SIFT:

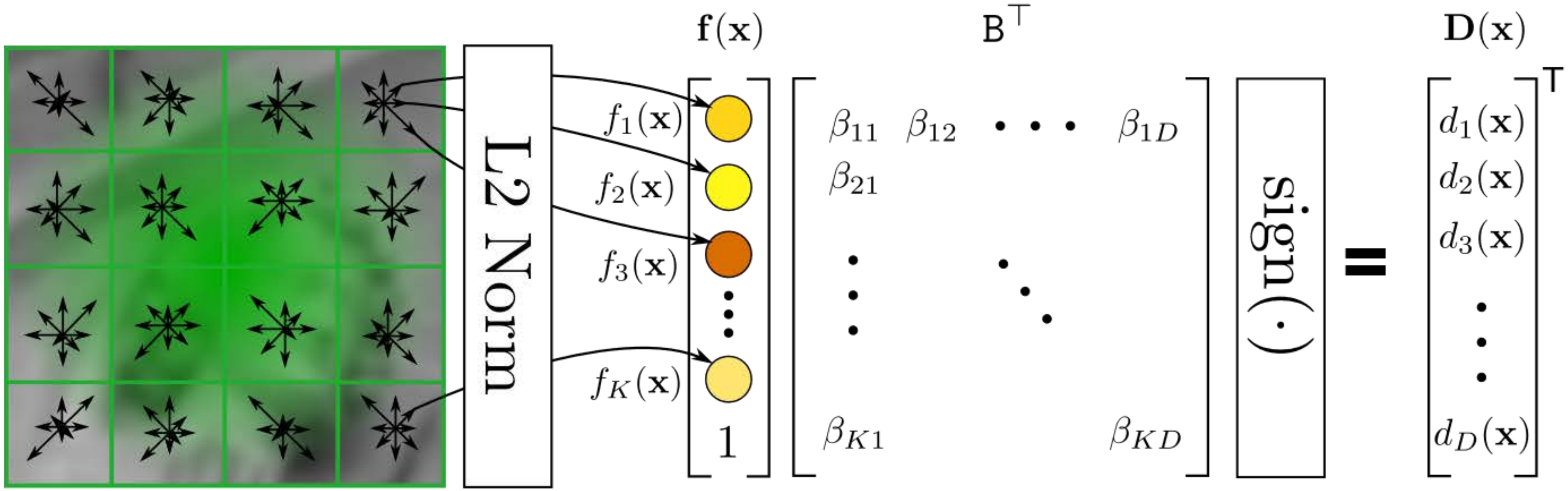


- We optimize **B** with gradient descent.

HashSIFT

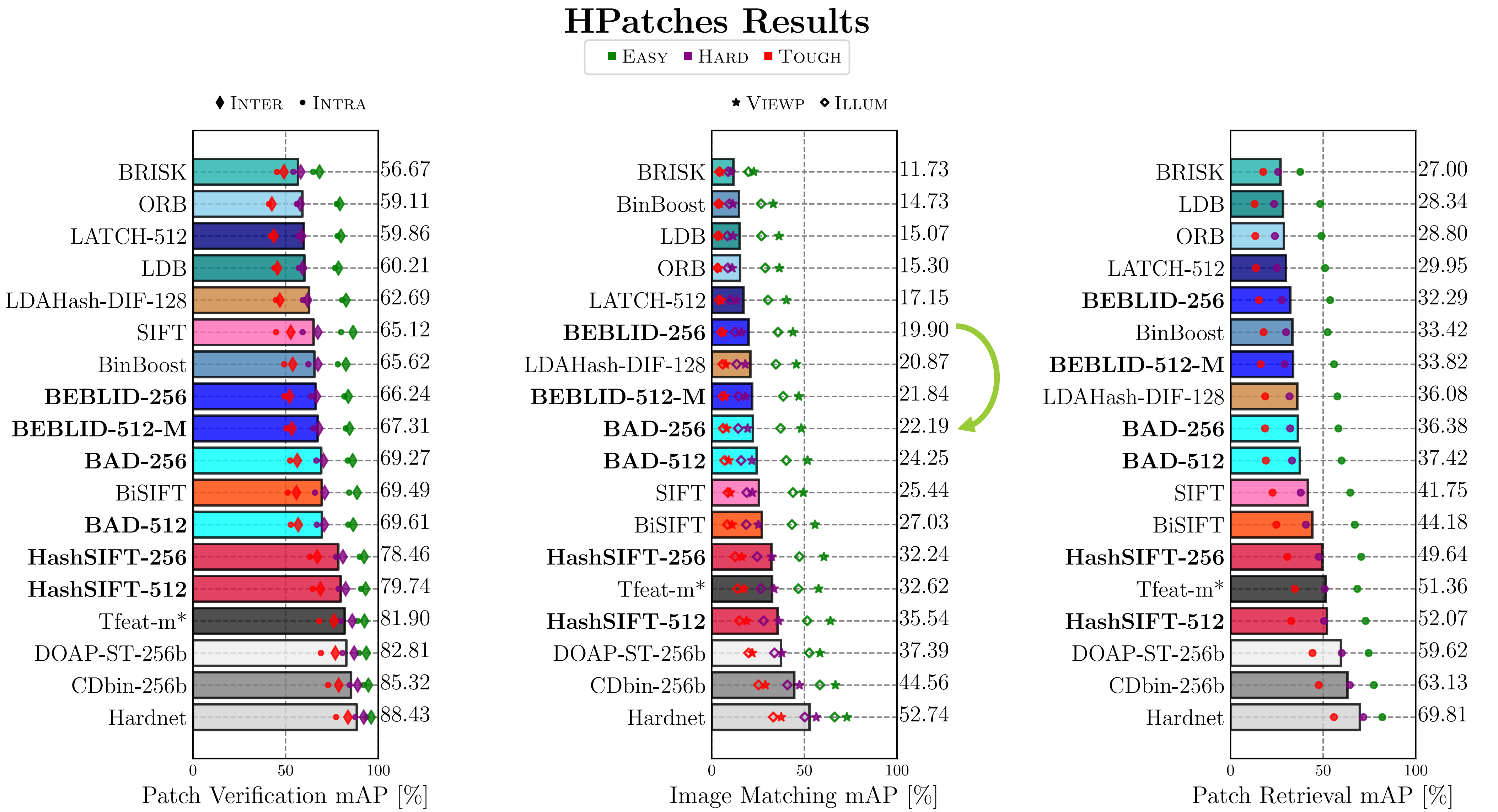
We studied how powerful is our training scheme to binarize a set of good features.

To this end we decided to binarize the Histogram of oriented gradients from SIFT:



- We optimize **B** with gradient descent.
- We approximate sign( ) by tanh( )

Results: Accuracy in planar image matching



Results: Accuracy in planar image matching

We revisit descriptors based on
Box Average Differences to propose BAD,
fast and more accurate than ORB

ORB vs BAD

Structure from Motion in the ETH benchmark

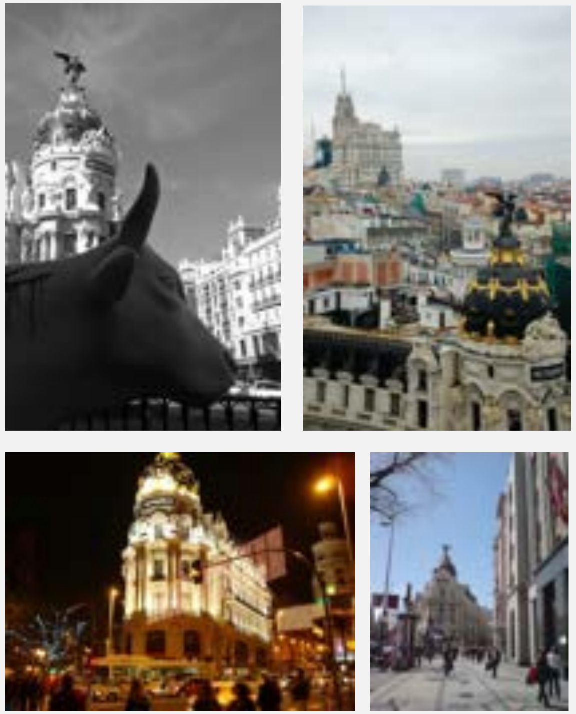
We evaluate in a realistic SfM problem reconstructing cities with images from multiple sources (for example Flickr).

Fountain
(11 images)

Herzjesu
(8 images)

South Building
(128 images)

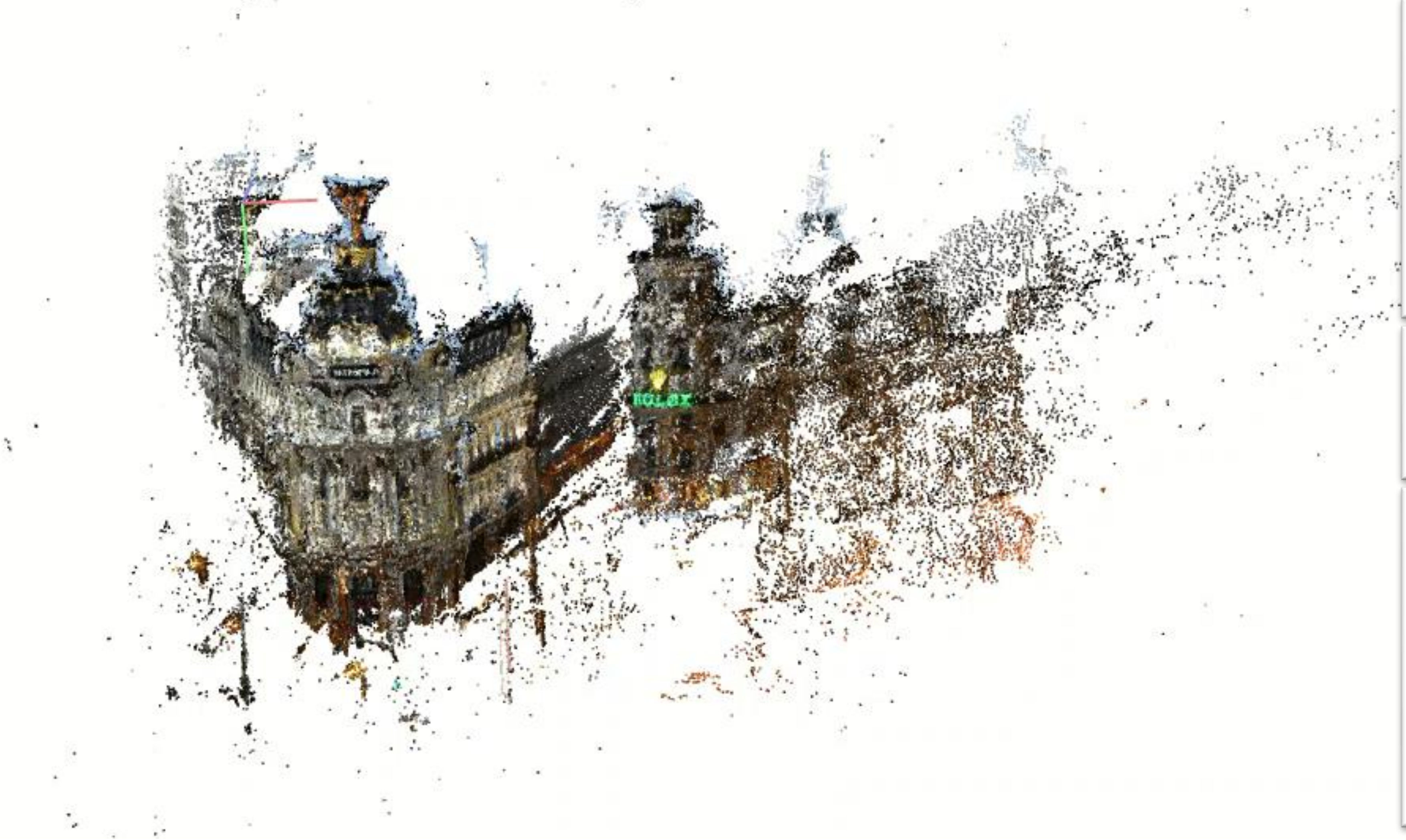
Madrid Metropolis
(1344 images)



Structure from Motion in the ETH benchmark




Madrid Metropolis reconstruction (BAD-256)





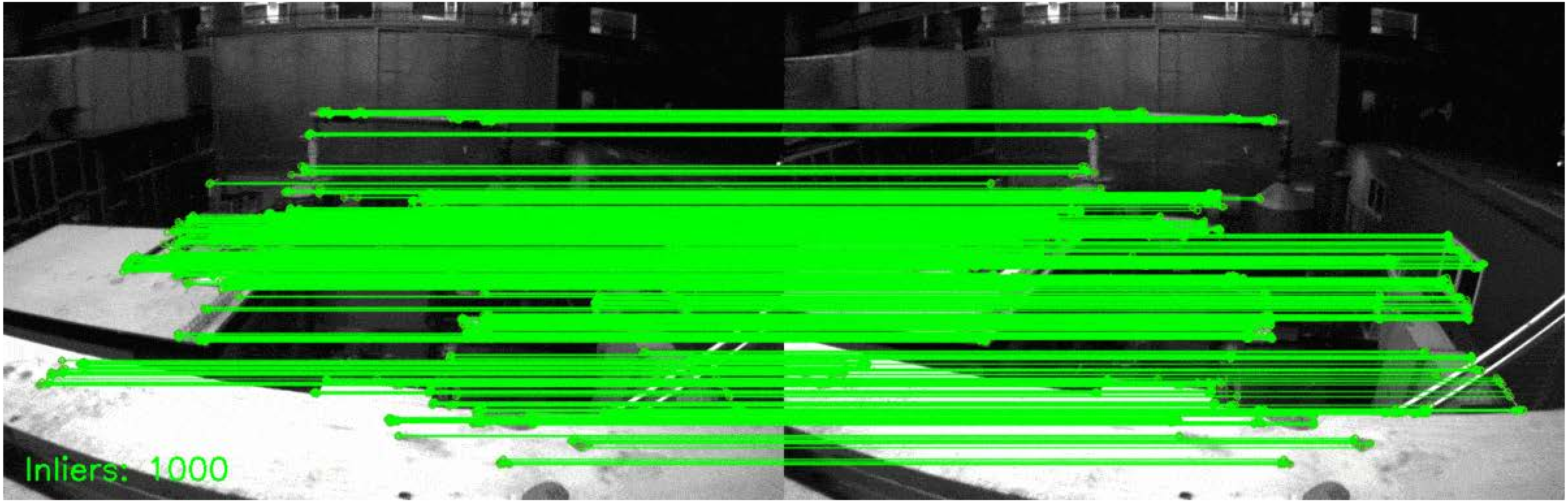
Structure from Motion in the ETH benchmark

We evaluate in a realistic SfM problem reconstructing cities with images from multiple sources (for example Flickr).

	# Registered	# Sparse Points	# Obervations	Track Length	# Dense Points
Madrid Metropolis (1344 images)					
ORB	457	135826	576138	4.241736	1085693
LATCH	573	186886	759581	4.064408	1245053
BEBLID-256	549	174257	705651	4.049484	1153261
BEBLID-512	609	199483	803847	4.029652	1191395
BAD-256	600	192638	789466	4.098184	1236144
BAD-512	622	189523	812243	4.285723	1268840
RSIFT	729	286519	1136306	3.965901	1349061
Binboost	514	143622	629993	4.386466	1129936
LDAHash-DIF-128	592	233862	804944	3.441961	1046695
HashSIFT-256	720	298920	1075450	3.597785	1202895
HashSIFT-512	720	305237	1160738	3.802743	1387138
TFeat	690	262790	986470	3.753834	1233791
HardNet	849	359610	1438909	4.001304	1436234
CDbin-256b	769	260690	1108018	4.250328	1347656

Stereo vision in Micro Aerial Vehicle (MAV)

Here we show BAD matching features in a non-planar scene of the EuRoC MAV Dataset. These matches can be used to compute the essential matrix of the MAV and thus its position.



https://www.flaticon.es/autores/rukanicon

Method	Size	Intel Core i7 8750H	Exynox Octa S	Snapdragon 855
BRISK	512b	14.94 (±0.31)	164.75 (±4.10)	19.38 (±0.19)
ORB	256b	12.07 (±0.33)	100.04 (±1.16)	16.40 (±0.25)
LDB	256b	17.48 (±0.8)	161.30 (±4.44)	27.32 (±0.11)
LATCH	512b	101.89 (±1.67)	1509.66 (±24.35)	159.02 (±0.24)
BEBLID	256b	1.56 (±0.05)	20.04 (±0.31)	4.75 (±0.06)
BEBLID	512b	2.84 (±0.07)	31.62 (±0.26)	7.18 (±0.21)
BAD	256b	1.53 (±0.04)	20.04 (±0.28)	4.40 (±0.06)
BAD	512b	2.77 (±0.08)	31.63 (±0.33)	6.28 (±0.15)
SIFT	128f	187.24 (±5.28)	2519.86 (±21.41)	572.30 (±5.70)
BinBoost	256b	84.65 (± 2.28)	786.80 (±60.82)	123.33 (± 1.73)
BiSIFT	482b	30.16 (± 0.77)	293.38 (±38.60)	46.29 (±3.02)
HashSIFT	256b	31.41 (± 0.73)	211.76 (±2.66)	48.36 (± 1.32)
HashSIFT	512b	33.80 (± 1.67)	259.17 (±0.62)	57.35 (± 0.42)
Tfeat-m*	128f	332.05 (± 7.41)	4110.61 (±46.59)	1028.94 (±59.25)
CDbin	256b	1184.82 (± 26.24)	23219.06 (± 274)	5363.37 (±383.79)
Hardnet	128f	763.08 (± 15.27)	15640.32 (±112.8)	3645.39 (±274.19)

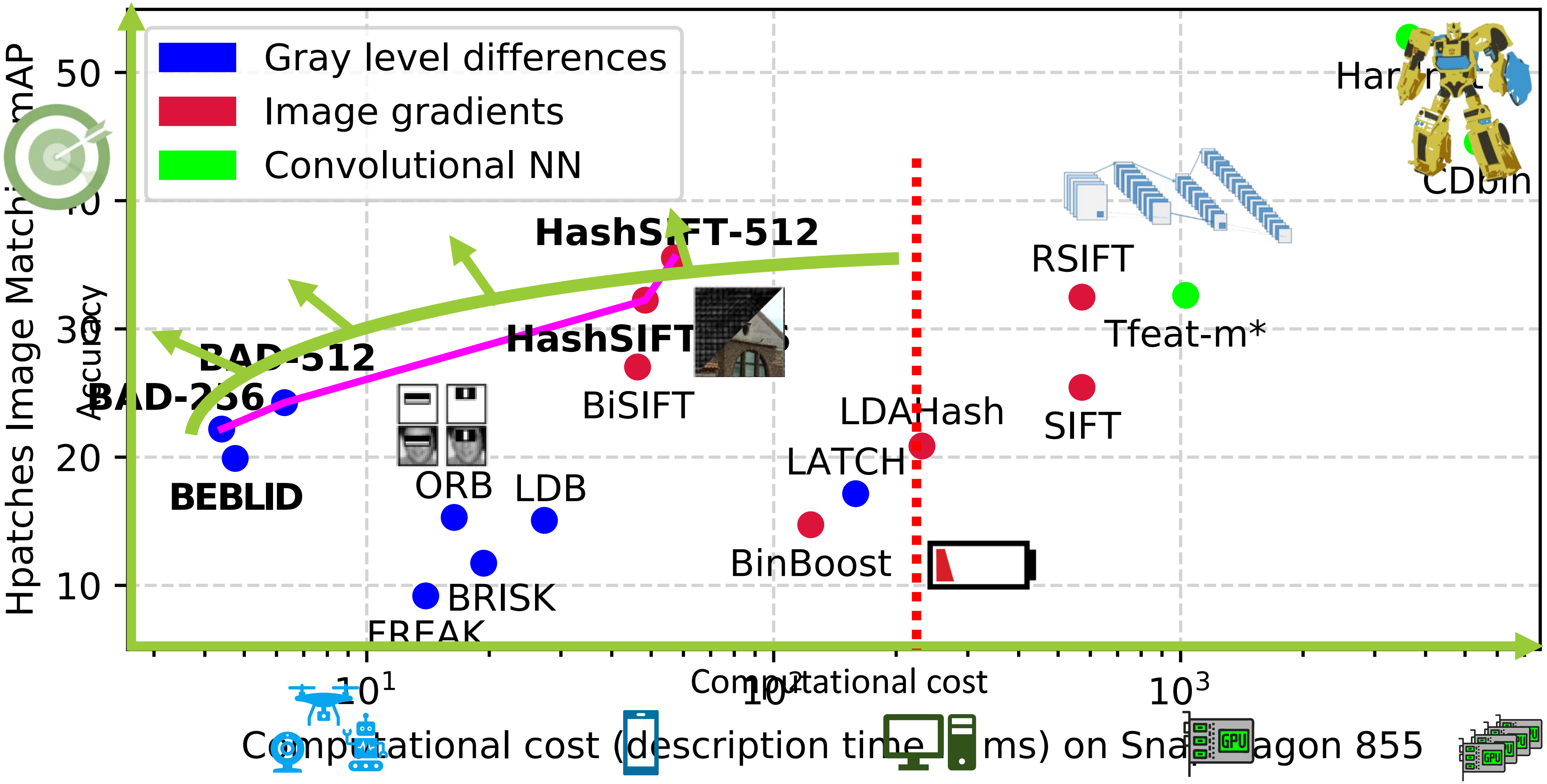
Description times

Results: Energy consumption

- BAD and BEBLID are the fastest descriptors
- We also evaluate our descriptors in terms of **energy consumption** per frame:

Method	FPS	CPU time (ms)	Estimated Power use (%)	Temp. increase (C°)	Discharge (mAh)
ORB	29.177	73.8	0.00028	0.00045	0.0118
BAD-256	29.970	**73.4**	**0.00026**	**0.00030**	**0.0093**
BAD-512	**29.977**	89.4	0.00031	0.00036	0.0104
BEBLID-512	29.91	90.4	0.00031	0.00037	0.0108
SIFT	5.548	1062.1	0.00337	0.00370	0.0798
BinBoost	9.841	565.3	0.00180	0.00146	0.0408
HashSIFT-256	**18.124**	**291.1**	**0.00097**	**0.00084**	**0.0217**
HashSIFT-512	15.630	353.3	0.00119	0.00105	0.0262
Tfeat-m*	**1.569**	**3441.9**	**0.01147**	**0.00968**	**0.2663**
CDbin-256b	0.319	20195	0.06708	0.05396	1.3614
Hardnet	0.459	13099	0.04335	0.03561	0.9245

State-of-art in the Accuracy vs. resource consumption curve



Index

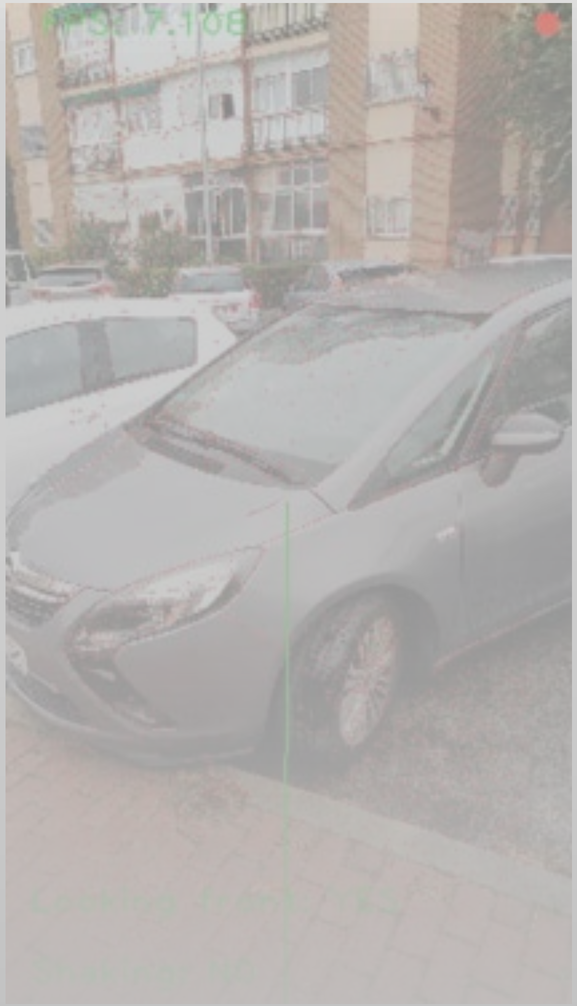
0. Abstract
1. Introduction
2. Line segment detection
3. Full line detection and vanishing point estimation
4. Local feature description
5. **Industrial results**
6. Conclusions

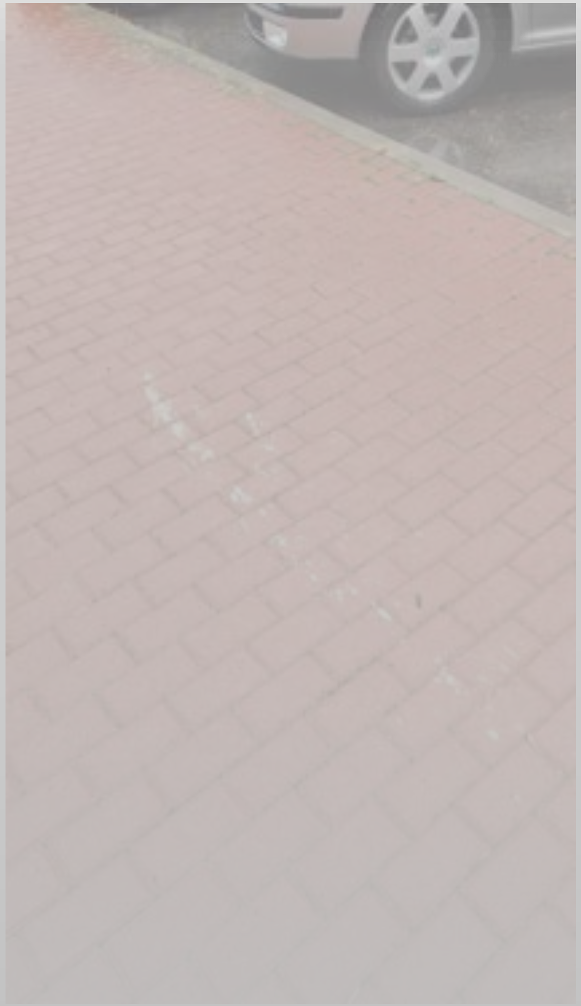
Vanishing point estimation for training

Input image

Image warped with VP's

The Graffter - wallview





Index

0. Abstract
1. Introduction
2. Line segment detection
3. Full line detection and vanishing point estimation
4. Local feature description
5. Industrial results
6. **Conclusions**

Publications

Suárez, I., Muñoz, E., Buenaposada, J. M., & Baumela, L. (2018, October). FSG: A statistical approach to line detection via fast segments grouping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 97-102). IEEE.

Suárez, I., Sfeir, G., Buenaposada, J. M., & Baumela, L. (2019, July). BELID: Boosted efficient local image descriptor. In *Iberian Conference on Pattern Recognition and Image Analysis* (pp. 449-460). Springer, Cham.

Suárez, I., Sfeir, G., Buenaposada, J. M., & Baumela, L. (2020). BEBLID: Boosted efficient binary local image descriptor. *Pattern Recognition Letters*, *133*, 366-372.

Suárez, I., Buenaposada, J. M., & Baumela, L. (2021). Revisiting Binary Local Image Description for Resource Limited Devices. *IEEE Robotics and Automation Letters*, *6*(4), 8317-8324.

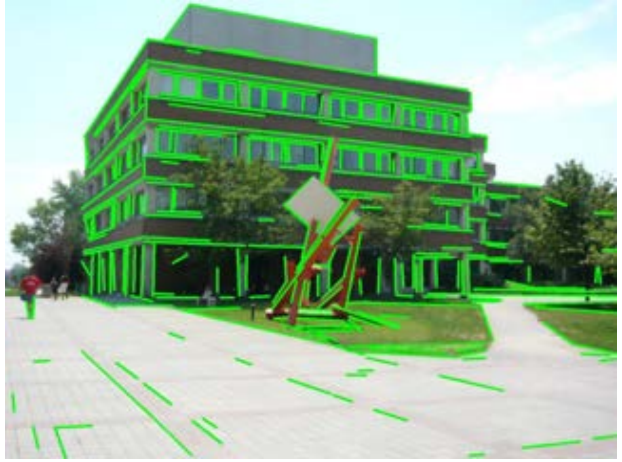
Suárez, I., Buenaposada, J. M., & Baumela, L. (2021). ELSED: Enhanced Line SEgment Drawing. Under review in Pattern Recognition.

Future work

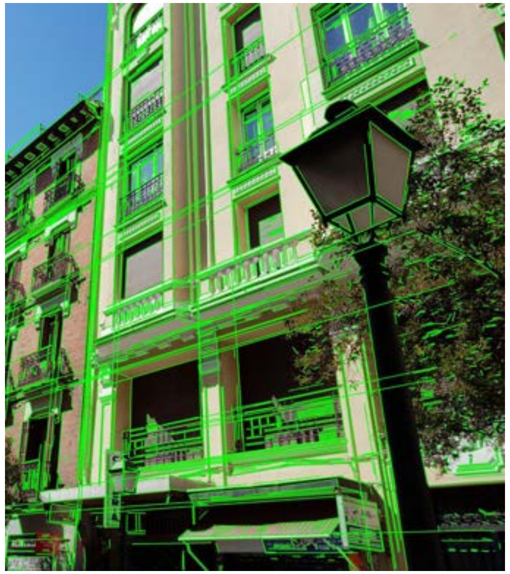
- Create a Pull Request in OpenCV for: BAD, HashSIFT and ELSED
- Line segment description and matching (ETH internship)
- SLAM mobile system based on BAD
- BAD-based place recognition and image retrieval system
- Web mixed reality applications

Conclusions

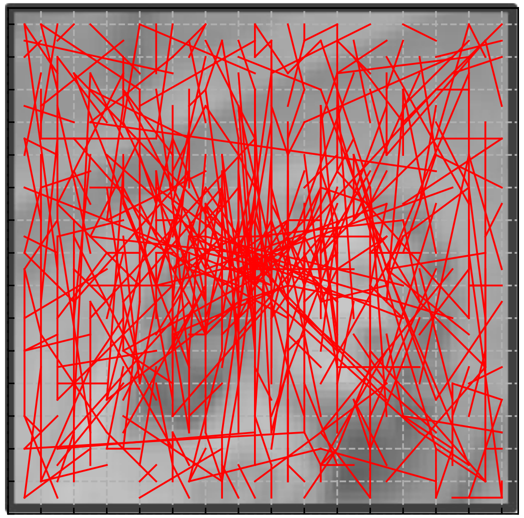
- The proposed work set a **new state of the art** in the accuracy VS computational cost curve



ELSED



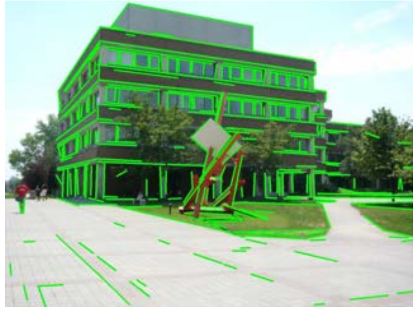
FSG



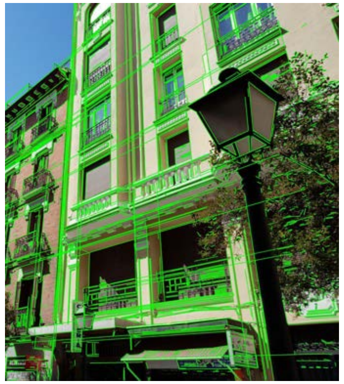
Efficient descriptors

Conclusions

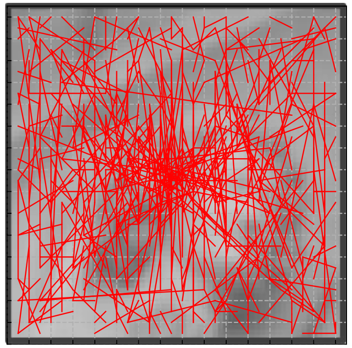
- The proposed work set a **new state of the art** in the accuracy VS computational cost curve



ELSED

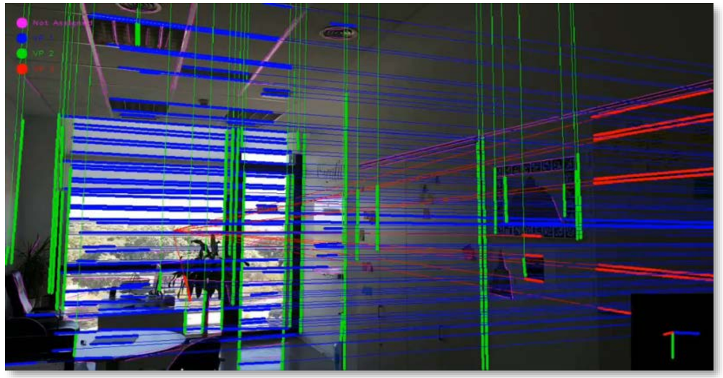


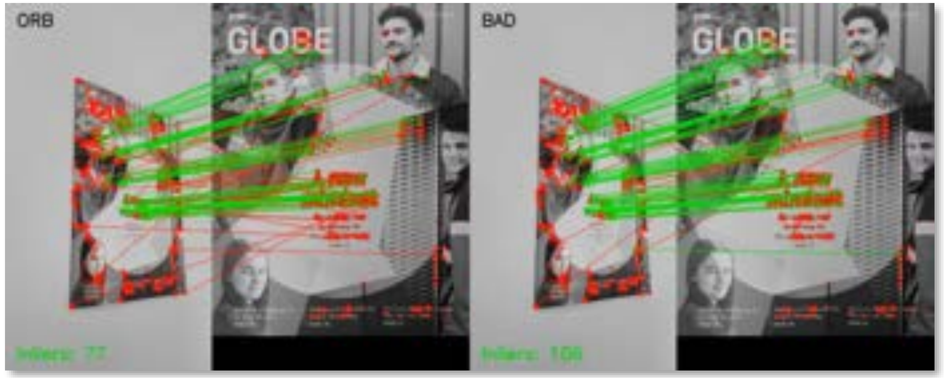
FSG



Efficient descriptors

- These new methods improve the quality of high-level tasks
- They open the door to a new generation of CV application in resource-limited devices





Thanks!

Bibliography

- **(Hough, 1962)** Hough, P. V. (1962). *U.S. Patent No. 3,069,654*. Washington, DC: U.S. Patent and Trademark Office.
- **(Etemadi, 1992)** Etemadi, A., 1992. Robust segmentation of edge data, in: Internat. Conf. on ImageProcessing and its Applications, pp. 311–314.
- **(Matas, J., 2000)** Matas, J., Galambos, C., & Kittler, J. (2000). Robust detection of lines using the progressive probabilistic hough transform. *Computer vision and image understanding*, *78*(1), 119-137.
- **(Jang, 2002)** Jang, J. H., & Hong, K. S. (2002). Fast line segment grouping method for finding globally more favorable line segments. *Pattern Recognition*, *35*(10), 2235-2247.
- **(Bandera, 2006)** Bandera, A., Pérez-Lorenzo, J. M., Bandera, J. P., & Sandoval, F. (2006). Mean shift based clustering of Hough domain for fast line segment detection. *Pattern Recognition Letters*, *27*(6), 578-586.
- **(Winder, 2007)** Winder, S. A., & Brown, M. (2007, June). Learning local image descriptors. In *Proc. of CVPR* (pp. 1-8). IEEE.
- **(Fernandes, 2008)** Fernandes, L. A., & Oliveira, M. M. (2008). Real-time line detection through an improved Hough transform voting scheme. *Pattern recognition*, *41*(1), 299-314.
- **(VonGioi, 2010)** Von Gioi, R. G., Jakubowicz, J., Morel, J. M., & Randall, G. (2008). LSD: A fast line segment detector with a false detection control. *TPAMI*, *32*(4), 722-732.
- **(Akinlar, 2011)** Akinlar, C., & Topal, C. (2011). EDLines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, *32*(13), 1633-1642.

Bibliography

- **(Lezama, 2014)** Lezama, J., Grompone von Gioi, R., Randall, G., & Morel, J. M. (2014). Finding vanishing points via point alignments in image primal and dual domains. In *Proc. CVPR* (pp. 509-515).
- **(Zhang, 2014)** Zhang, L., Lu, H., Hu, X., & Koch, R. (2016). Vanishing point estimation and line classification in a Manhattan world with a unifying camera model. *IJCV*, *117*(2), 111-130.
- **(Lowry, 2015)** Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., & Milford, M. J. (2015). Visual place recognition: A survey. *IEEE Transactions on Robotics*, *32*(1), 1-19.
- **(Schroff, 2015)** Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of CVPR* (pp. 815-823).
- **(Almazan, 2017)** Almazan, E. J., Tal, R., Qian, Y., & Elder, J. H. (2017). Mcmlsd: A dynamic programming approach to line segment detection. In *Proc. CVPR* (pp. 2031-2039).
- **(Balntas, 2017):** Balntas, V., Lenc, K., Vedaldi, A., & Mikolajczyk, K. (2017). HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proc. of CVPR* (pp. 5173-5182).
- **(Yang, 2017)** Yang, S., & Scherer, S. (2017, May). Direct monocular odometry using points and lines. In *Proc. ICRA* (pp. 3871-3877). IEEE.
- **(Zuo, 2017)** Zuo, X., Xie, X., Liu, Y., & Huang, G. (2017, September). Robust visual SLAM with point and line features. In *Proc. IROS* (pp. 1775-1782). IEEE.
- **(Rajaei, 2018)** Rajaei, B., & von Gioi, R. G. (2018). Gestaltic grouping of line segments. *Image Processing On Line*, *8*, 37-50.

Bibliography

- **(Hu, 2019)** Hu, Y., Hugonot, J., Fua, P., & Salzmann, M. (2019). Segmentation-driven 6d object pose estimation. In *Proc. of CVPR* (pp. 3385-3394).
- **(Zhou, 2019)** Zhou, Y., Qi, H., & Ma, Y. (2019). End-to-end wireframe parsing. In *Proc. ICCV* (pp. 962-971).
- **(Wang , 2020)** Wang, S., Yu, L., Li, C., Fu, C. W., & Heng, P. A. (2020, August). Learning from extrinsic and intrinsic supervisions for domain generalization. In Proc. Of *ECCV* (pp. 159-176). Springer, Cham.
- **(Xue, 2020)** Xue, N., Wu, T., Bai, S., Wang, F., Xia, G. S., Zhang, L., & Torr, P. H. (2020). Holistically-attracted wireframe parsing. In *Proc. of CVPR* (pp. 2788-2797).
- **(Dai, 2021)** Dai, X., Yuan, X., Gong, H., & Ma, Y. (2021). Fully Convolutional Line Parsing. *arXiv preprint arXiv:2104.11207*.
- **(Xu, 2021)** Xu, Y., Xu, W., Cheung, D., & Tu, Z. (2021). Line segment detection using transformers without edges. In *Proc. of CVPR* (pp. 4257-4266).
- **(Zhang, 2021)** Zhang, H., Luo, Y., Qin, F., He, Y., & Liu, X. (2021). ELSD: Efficient Line Segment Detector and Descriptor. In *Proc. of ICCV (pp. 2969-2978).*
- Icons from:
  - https://www.flaticon.es/autores/rukanicon
  - https://pin.it/1bIeeEw