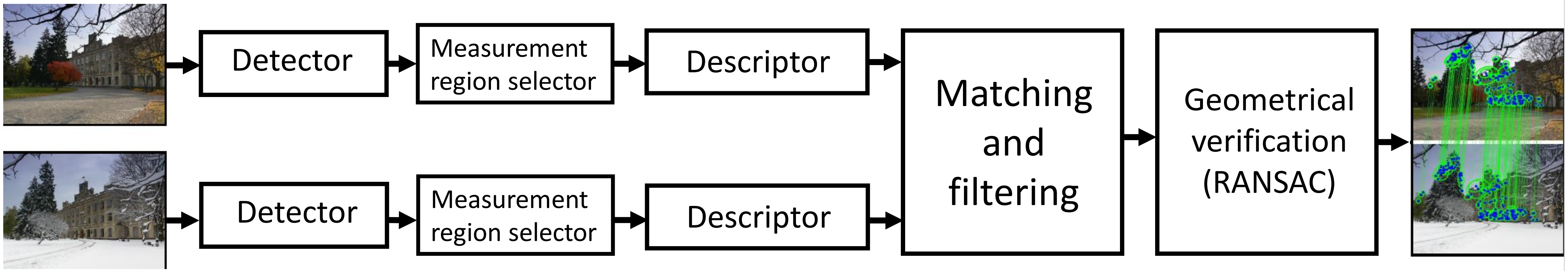
Local Features: from Paper to Practice

presented by Dmytro Mishkin

joint work with

Anastasia Mishchuk, Milan Pultar, Filip Radenovic, Daniel Barath, Michal Perdoch, Eduard Trulls, Kwang Moo Yi, Yuhe Jin, Jiri Matas

Wide baseline stereo pipeline



Single feature visualization

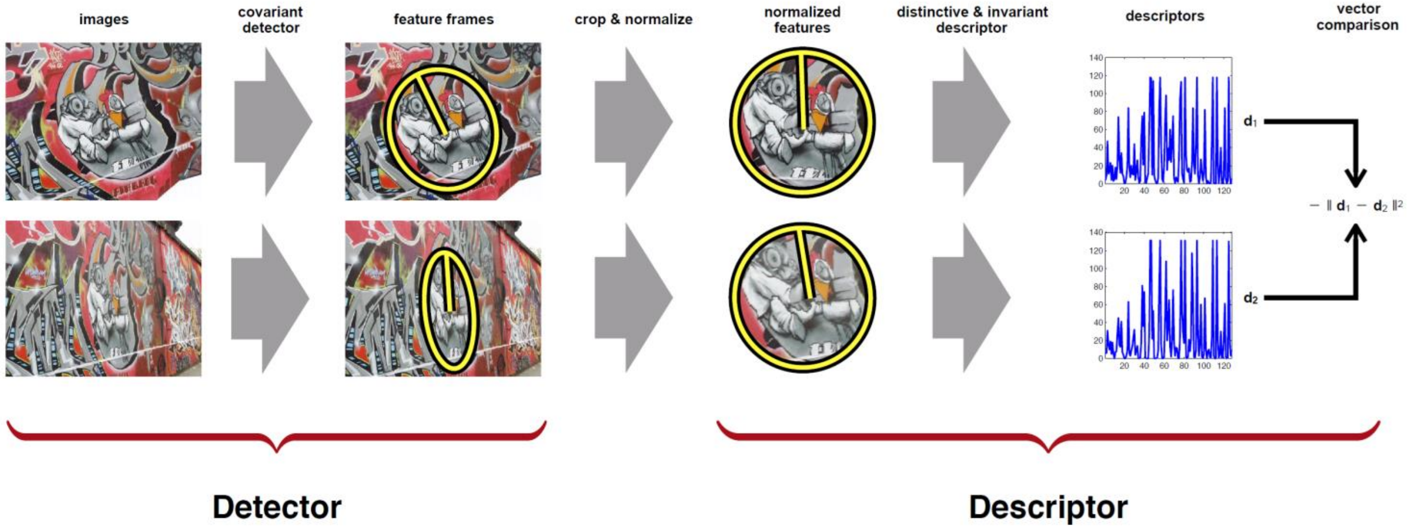


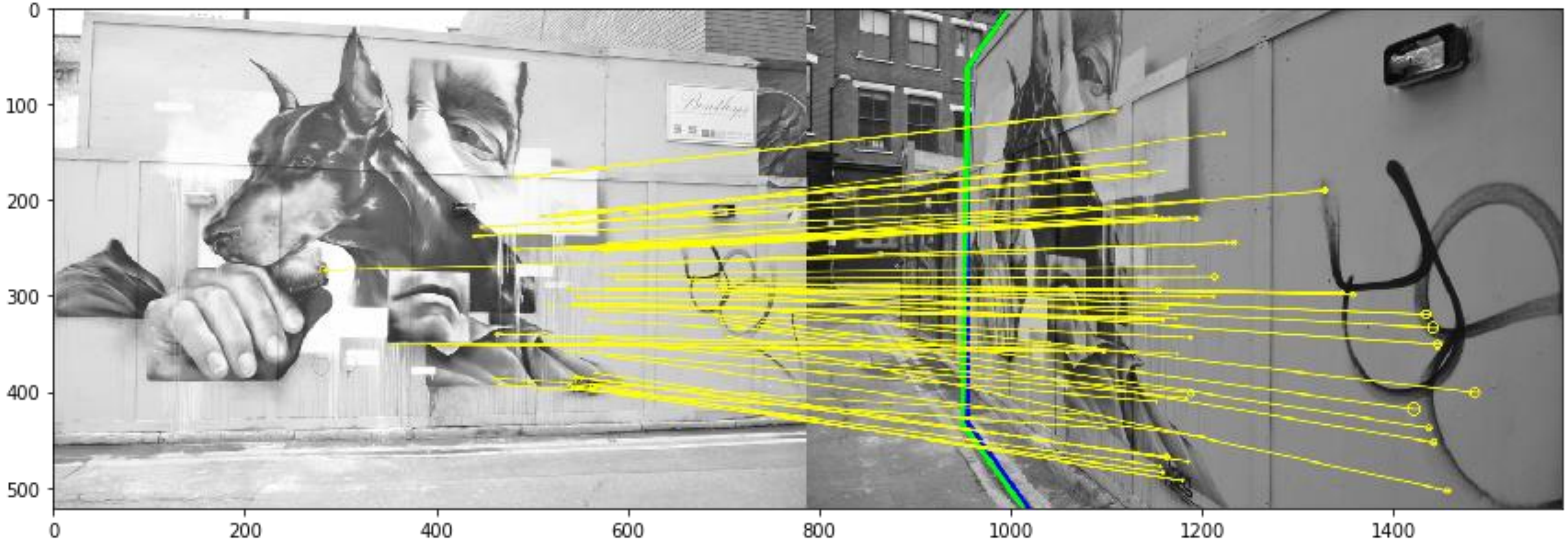
Image credit: Andrea Vedaldi, ICCVW 2017

Toy example for illustration: matching with OpenCV SIFT

```python
#Load images
img1 = cv2.imread('img/v_dogman/1.ppm')
img2 = cv2.imread('img/v_dogman/6.ppm')

#Detect & describe with SIFT
det = cv2.xfeatures2d.SIFT_create(3000)
kps1, descs1 = det.detectAndCompute(img1,None)
kps2, descs2 = det.detectAndCompute(img2,None)

# Match with chosen strategy
tentative_matches = match(descs1, descs2)

# Find geometric model and inliers
src_pts = np.float32([ kps1[m[0]].pt for m in tentative_matches ]).reshape(-1,2)
dst_pts = np.float32([ kps2[m[1]].pt for m in tentative_matches ]).reshape(-1,2)
H, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 1.0)

# Draw
H_gt = np.loadtxt('img/v_dogman/H_1_6')
draw_matches(kps1, kps2, tentative_matches, mask, H, H_gt, img1, img2)
```

[Try yourself: https://github.com/ducha-aiki/matching-strategies-comparison](https://github.com/ducha-aiki/matching-strategies-comparison)

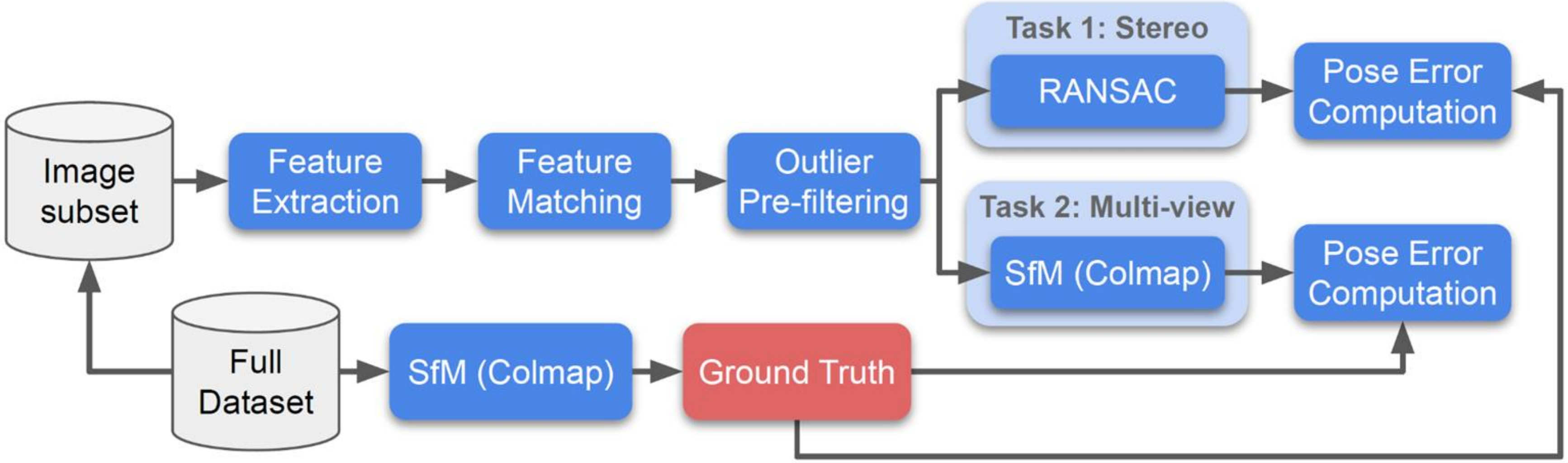
Toy example for illustration: matching with OpenCV SIFT



Recovered 1st to 2nd image projection,
ground truth 1st to 2nd image project,
inlier correspondences

Not a toy benchmark: pose accuracy on PhotoTourism data

Image Matching across Wide Baselines: From Paper to Practice. Jin et.al, arXiv 2020



Noname manuscript No.
(will be inserted by the editor)

Image Matching Across Wide Baselines: From Paper to Practice

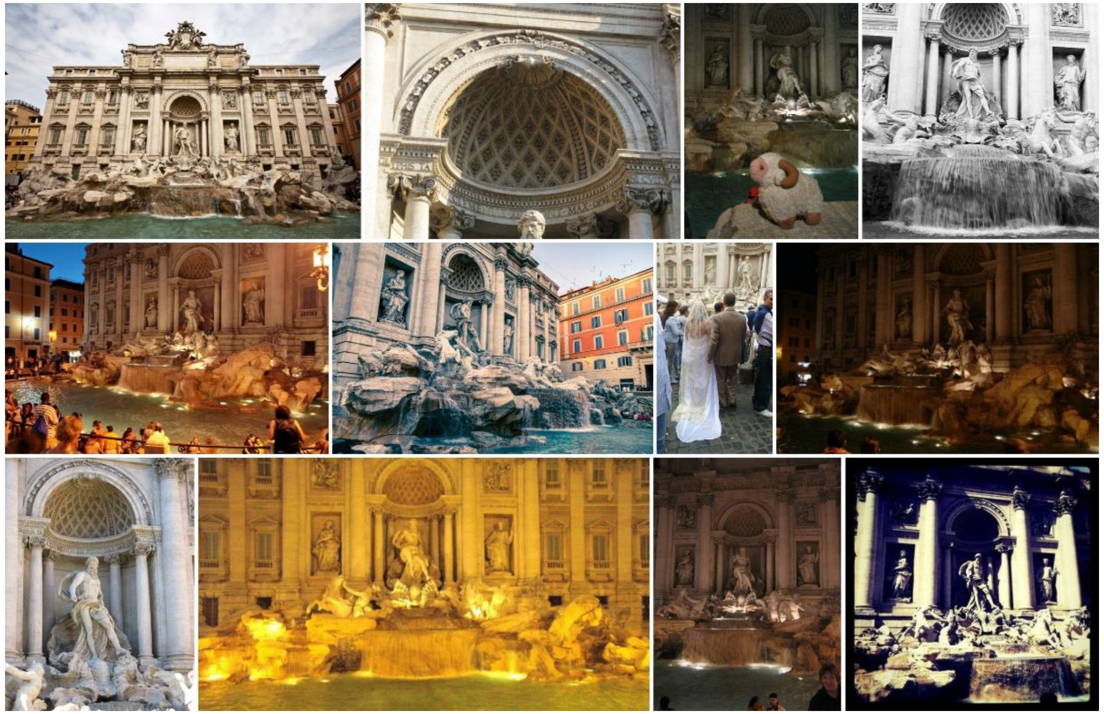
Yuhe Jin · Dmytro Mishkin · Anastasiia Mishchuk · Jiri Matas · Pascal Fua · Kwang Moo Yi · Eduard Trulls

Received: date / Accepted: date

- 15 training image sets
- 10 test image sets, 100 images each
- Metric: pose accuracy

Check out:
Image Matching: Local Features & Beyond
CVPR Workshop: Friday, June 19, 2020

The Phototourism Dataset

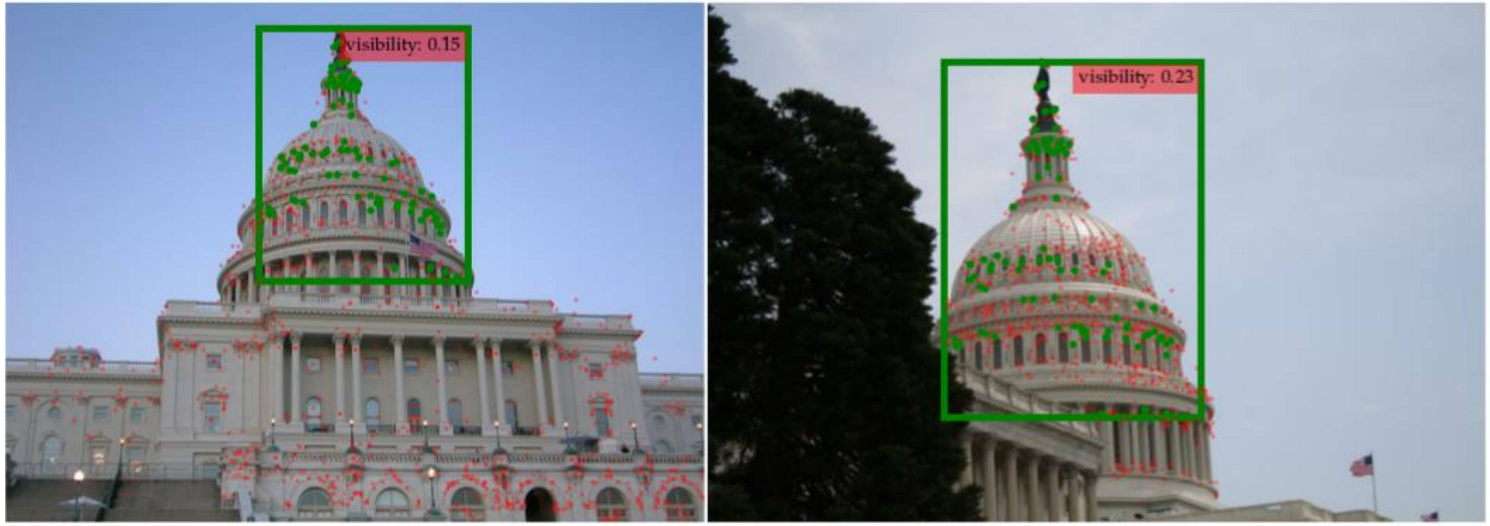


- 30k images from YCC100M dataset, in 26 scenes
- “Ground truth” established by COLMAP reconstruction
- The basis of Image Matching Competitions 2019 & 2020

Not a toy benchmark: pose accuracy on PhotoTourism data

Figure 2. **Phototourism dataset.** Some images in our dataset and their corresponding depth maps, with occlusions shown in red.

Image sets vary by common visible area



Co-visibility computation example.
3D points from COLMAP -> project into images.
Green if seen on both,
Red, if seen on one
Visibility – min(area of bounding box of the green points)

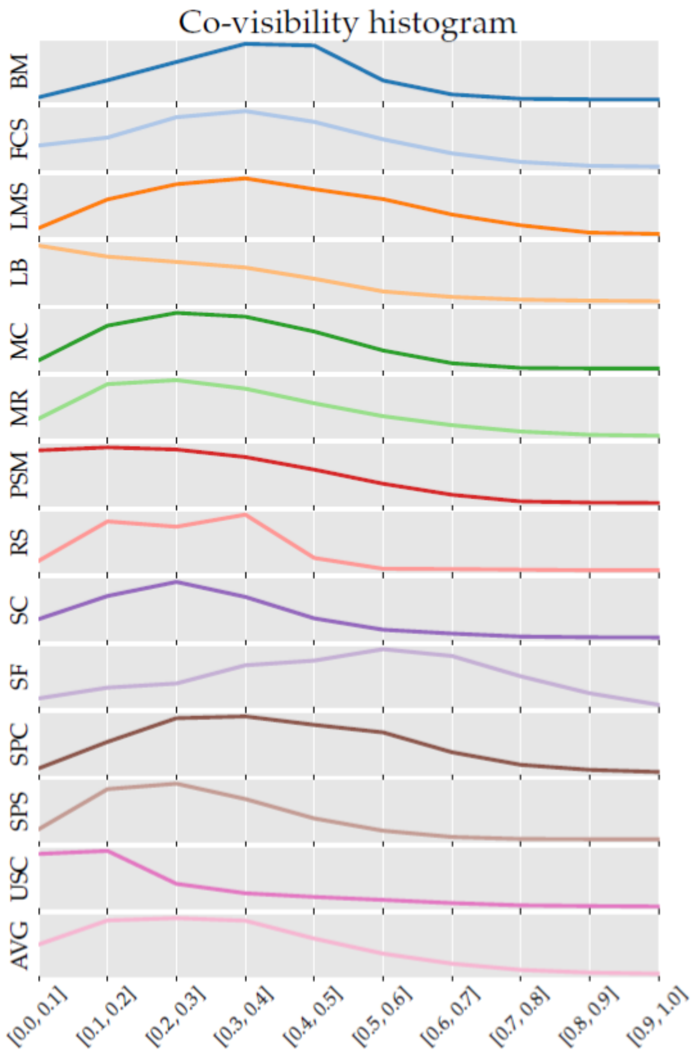
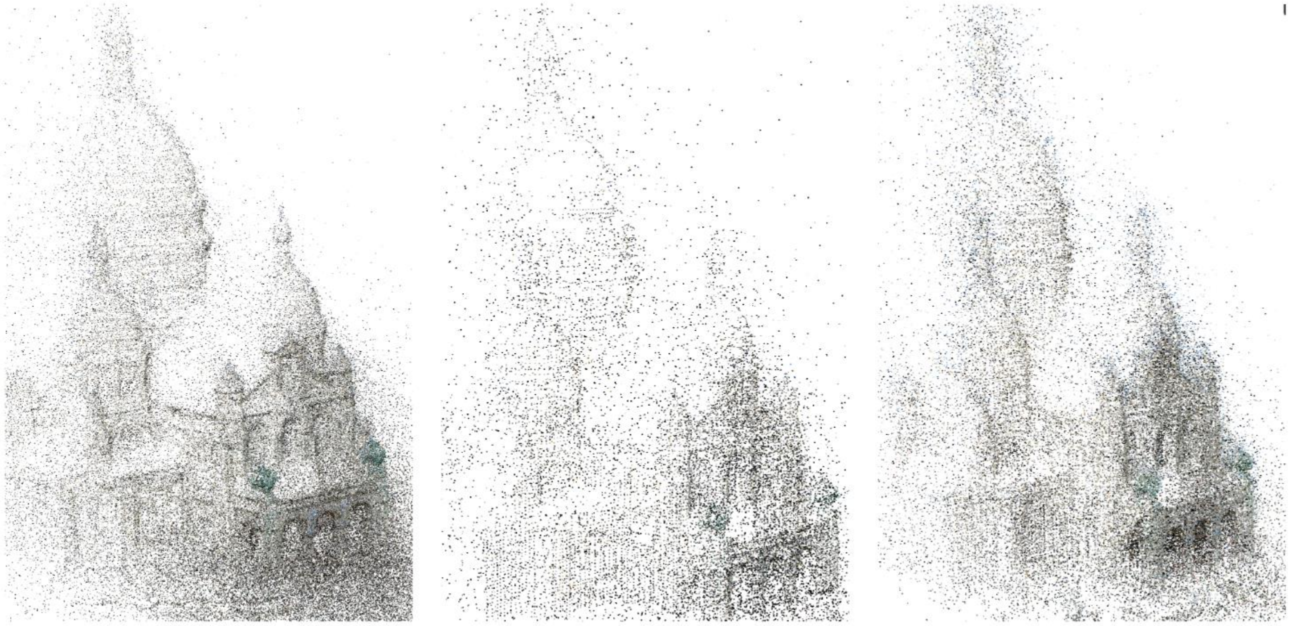


Figure 14. **Co-visibility histogram –** Breakdown for each scene in the validation and test sets. Notice how the statistics may significantly change from scene to scene.

Metric computation

1. RANSAC $\rightarrow$ fundamental matrix F
2. Essential matrix E from F: $E = K_1^T F \, K_2$
3. Camera pose R, t = cv2.recoverPose(E)
4. Decompose (R,t) into rotation and translation components, keep only rotation, get the angular error
5. Threshold angular error for set of thresholds and get accuracy per threshold
6. Calculate mAA @ 10º

Can we trust Colmap “Ground truth”?

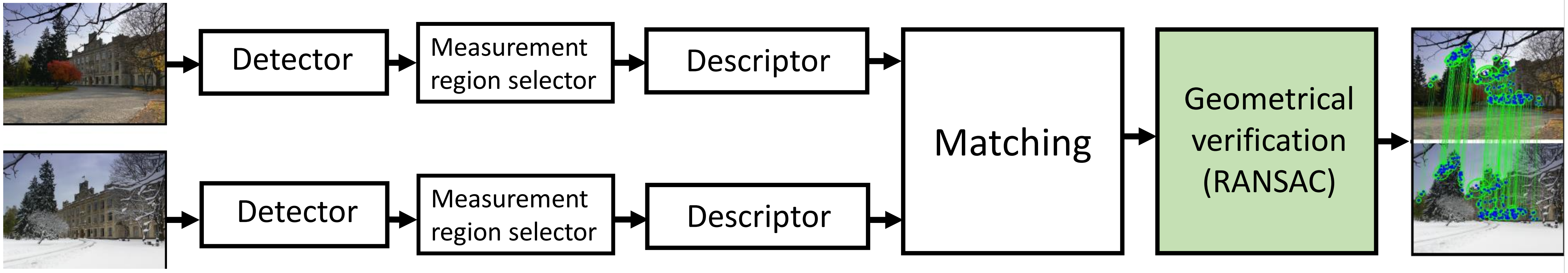


(a) SIFT (b) SuperPoint (c) R2D2

Yes, we can!

Feature used	Number of images			
	100 vs. all	200 vs. all	400 vs. all	800 vs. all
SIFT [55]	0.46° / 0.13°	0.42° / 0.11°	0.32° / 0.08°	0.39° / 0.08°
SuperPoint [34]	2.09° / 1.57°	2.09° / 1.54°	1.87° / 1.21°	2.53° / 0.53°
R2D2 [80]	0.41° / 0.14°	0.29° / 0.09°	0.28° / 0.09°	0.21° / 0.06°

Table 3 Pose convergence in SfM. We report the mean/median of the difference (in degrees) between the poses extracted with the full set of 1179 images for “Sacre Coeur”, and different subsets of it, for three local feature methods – to keep the results comparable we only look at the 100 images in common across all subsets. We report the maximum among the angular difference between rotation matrices and translation vectors. The estimated poses are stable, with as low as 100 images.

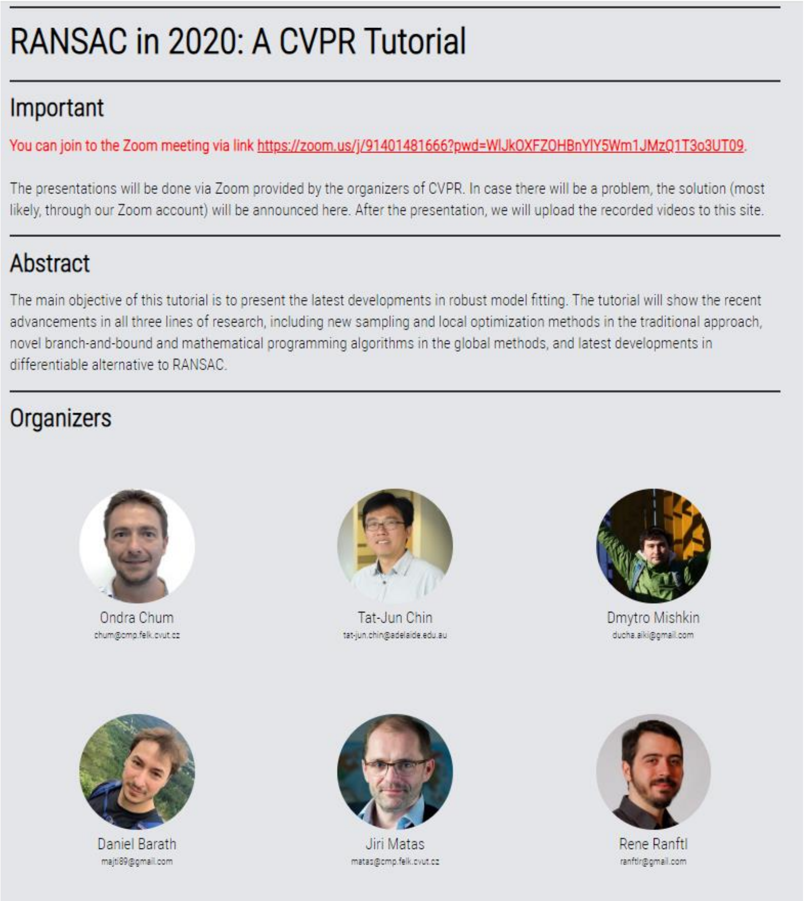


Geometric verification (RANSAC)

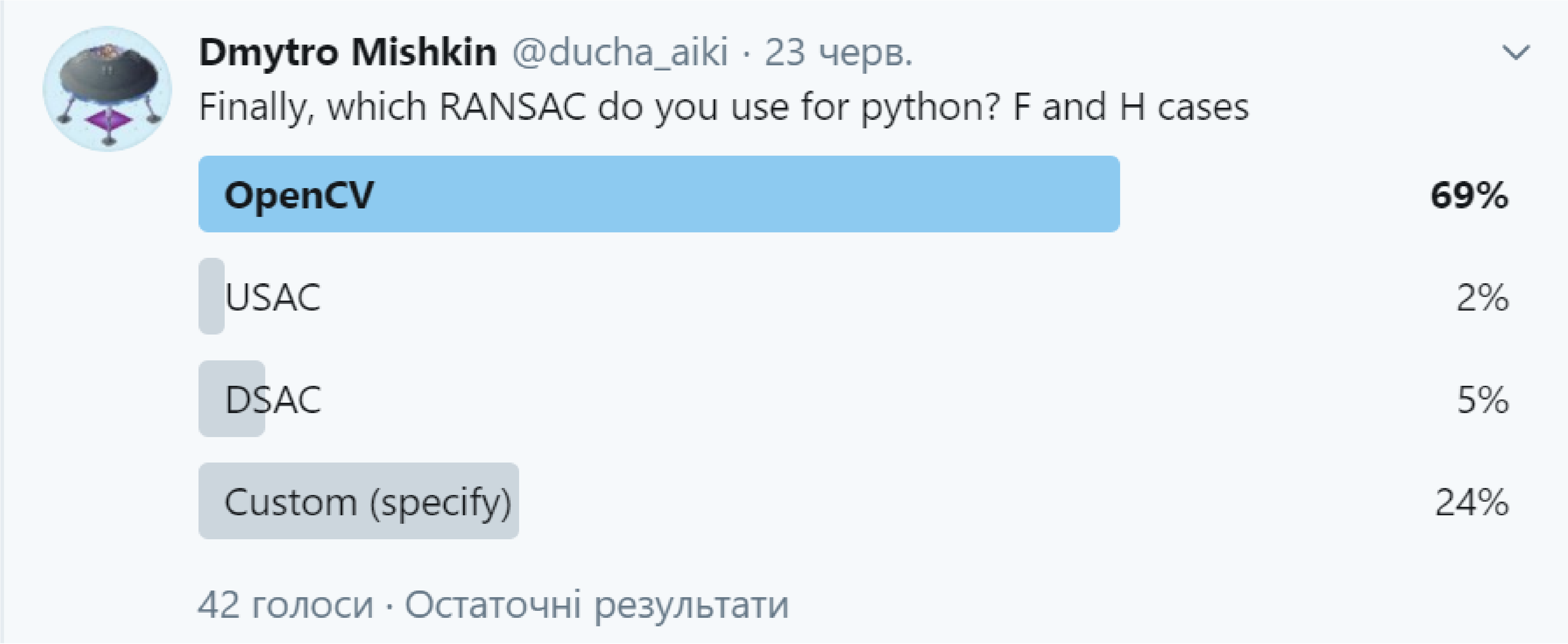
RANSAC: fitting the data with gross outliers

If you are not familiar with modern RANSACs, please, check the CVPR2020 “RANSAC in 2020” tutorial
http://cmp.felk.cvut.cz/cvpr2020-ransac-tutorial/

Slides are already there and videos will be uploaded soon



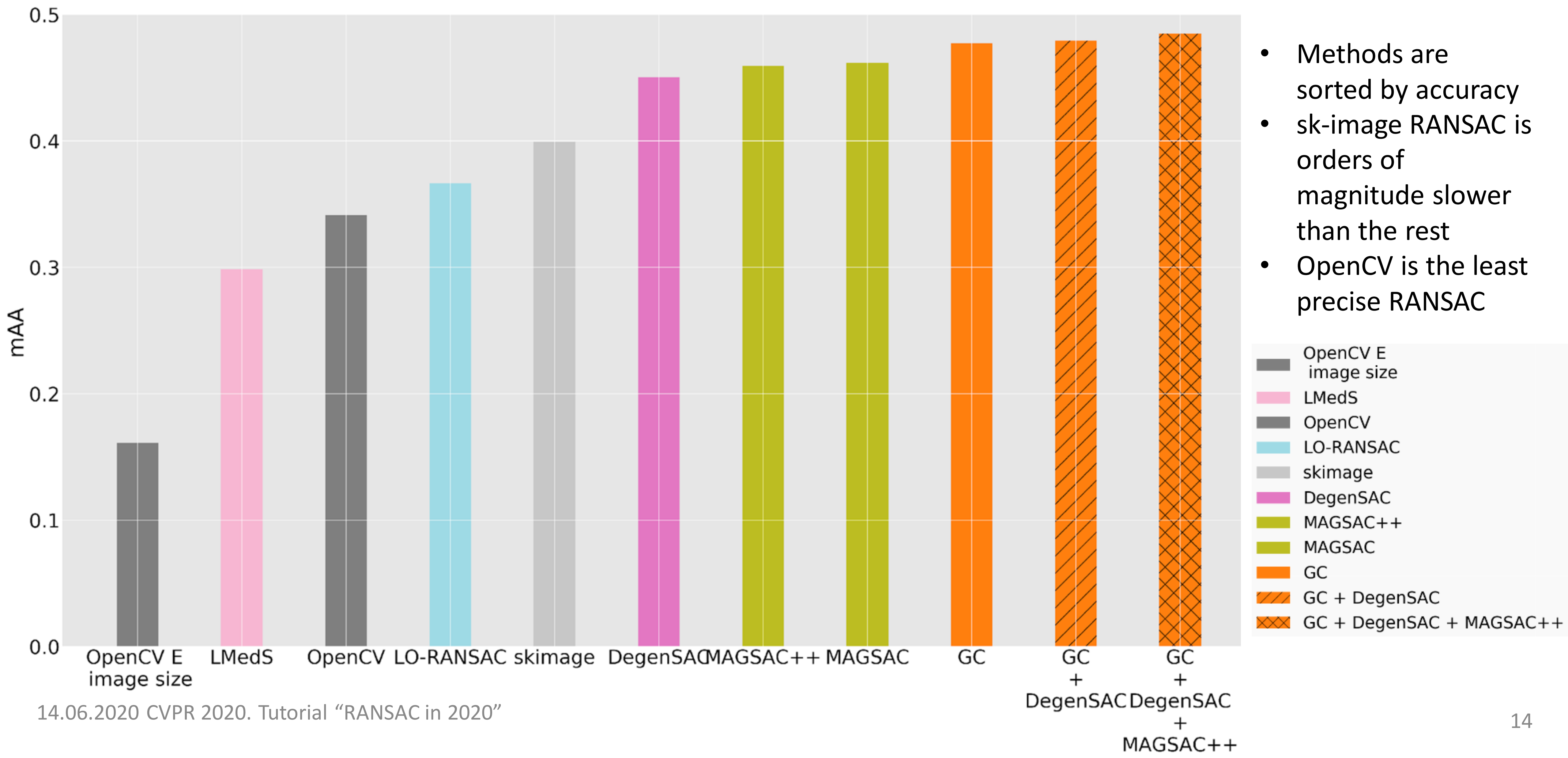
Is OpenCV RANSAC is a way to go?



OpenCV functions:
cv2.findHomography()
cv2.findFundamentalMatrix()

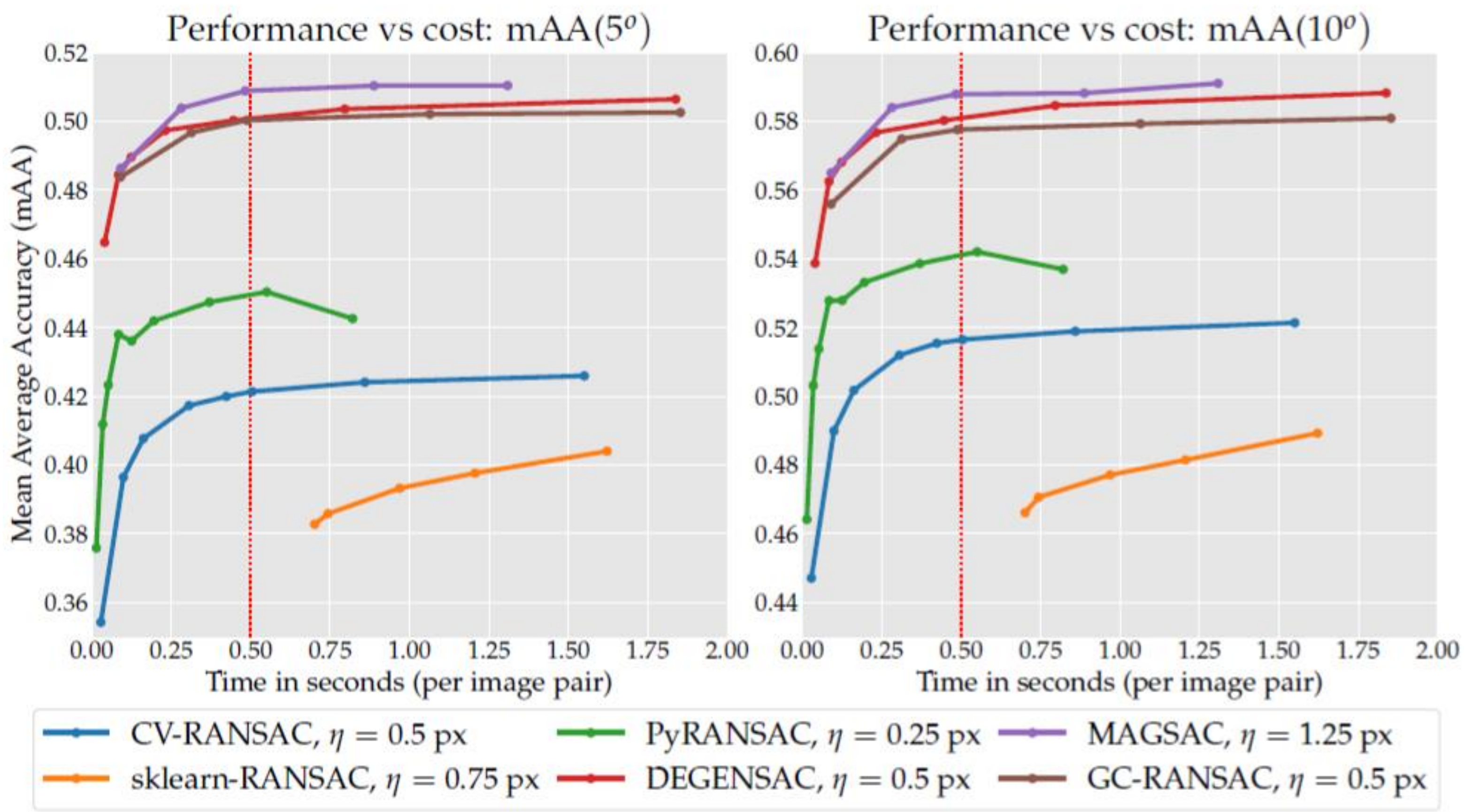
https://twitter.com/ducha_aiki/status/1142847831516037120

Classical F methods, 1k iterations



- Methods are sorted by accuracy
- sk-image RANSAC is orders of magnitude slower than the rest
- OpenCV is the least precise RANSAC

Classical F methods in Jin et.al 2020



Jin et.al 2020:

- Feature: SIFT, 8k points
- Vary maxIters, measure time.
- Advanced methods (MAGSAC, GC, DegenSAC) are better for both per second and per iteration

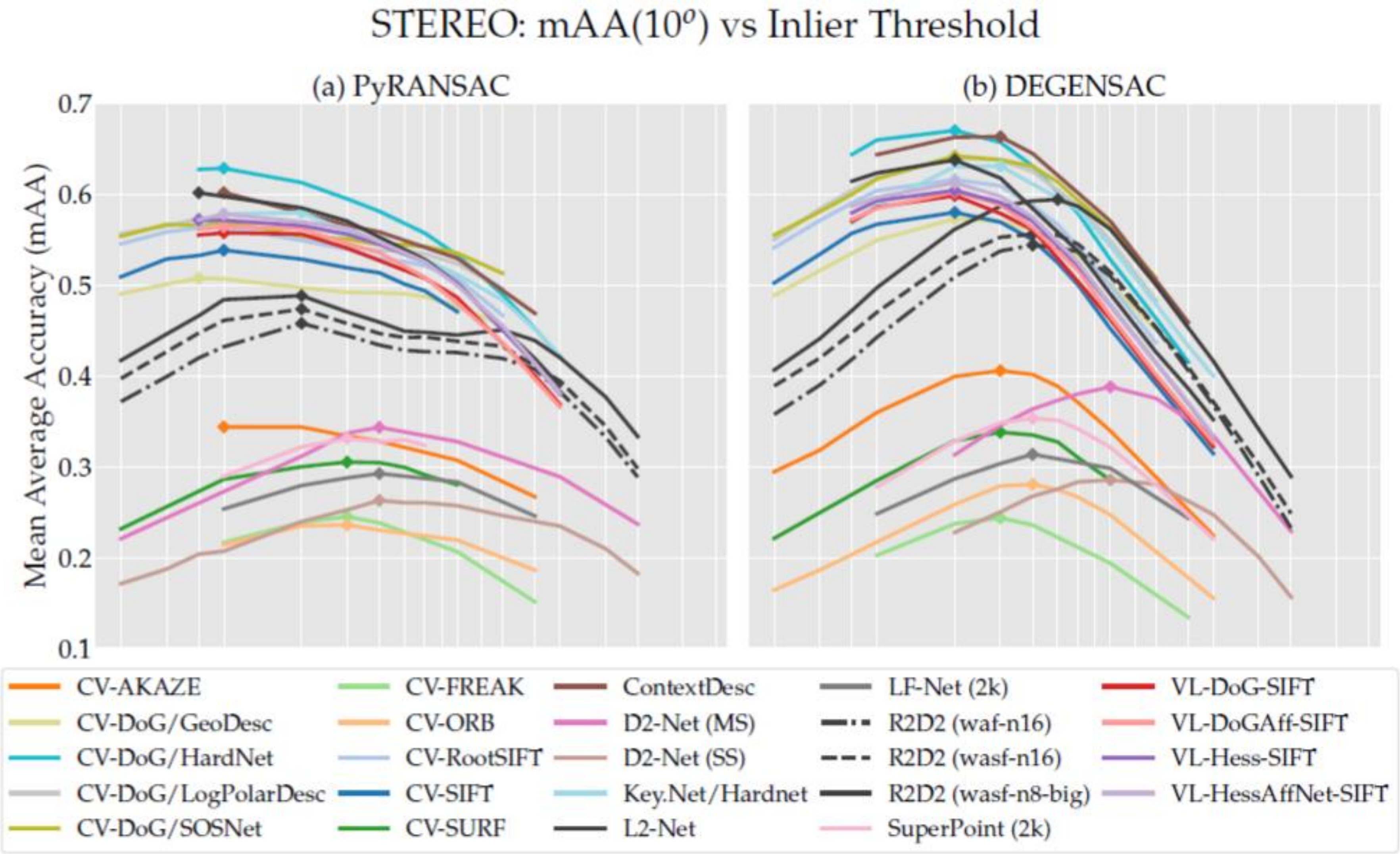
This tutorial:

- Benchmark was run on 4 different machines.
- Instead we fix the number of iterations for 1k, 10k, 100k, 1M
- We tune all parameters of the methods e.g. “spatial coherence term” for GC-RANSAC, which improves its results significantly

MAGSAC and MAGSAC++ github.com/danini/magsac (CVPR 2019 & CVPR 2020)
DEGENSAC github.com/ducha-aiki/pydegensac Chum et. al CVPR 2005. **pip install pydegensac**
GC-RANSAC github.com/danini/graph-cut-ransac Barath and Matas. Graph-cut RANSAC. CVPR 2018

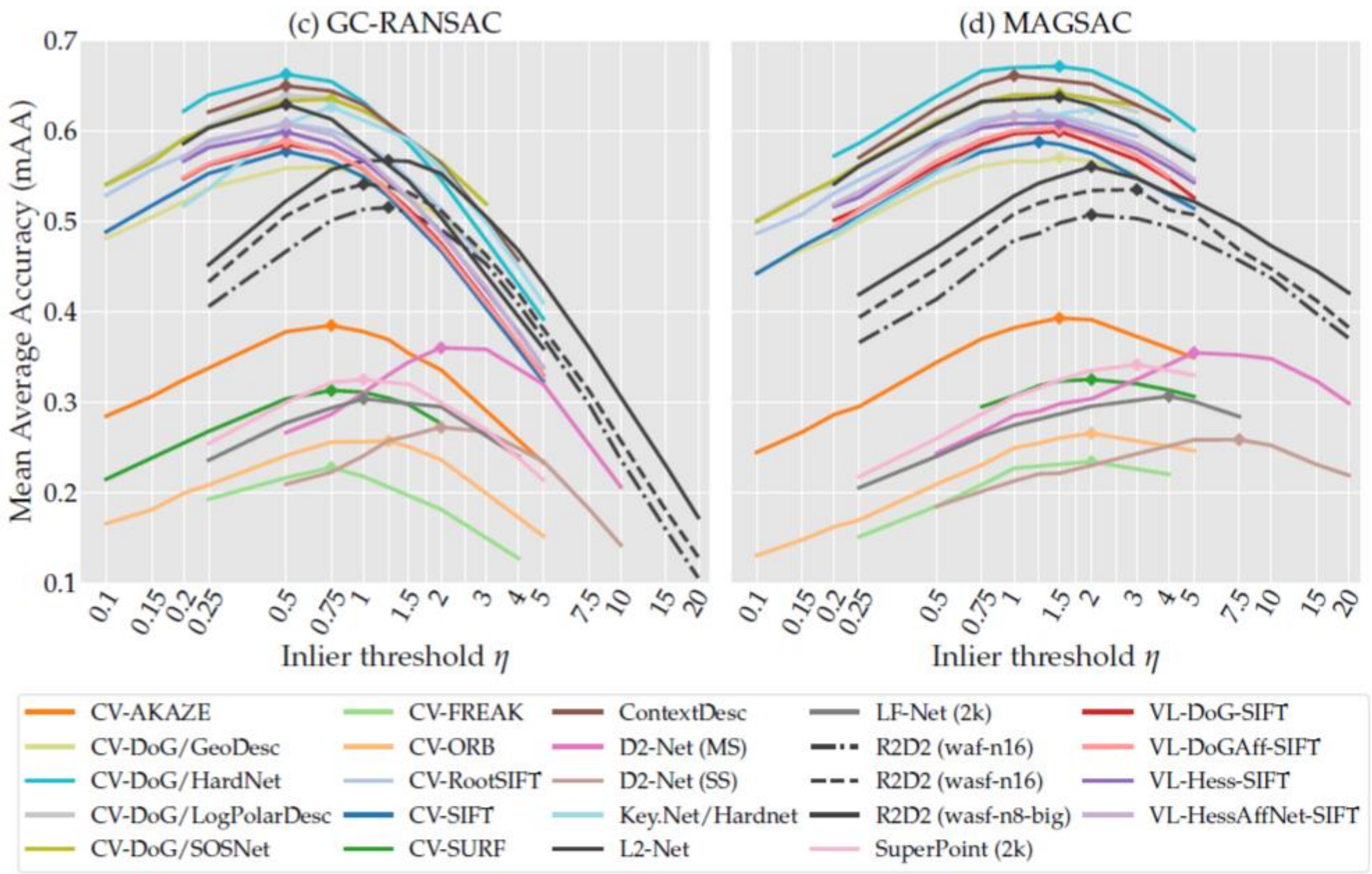
You need to tune each RANSAC for each local feature

Don`t look at descriptors & detectors yet comparison, there is a better plot later



You need to tune each RANSAC for each local feature

Don`t look at descriptors & detectors yet comparison, there is a better plot later



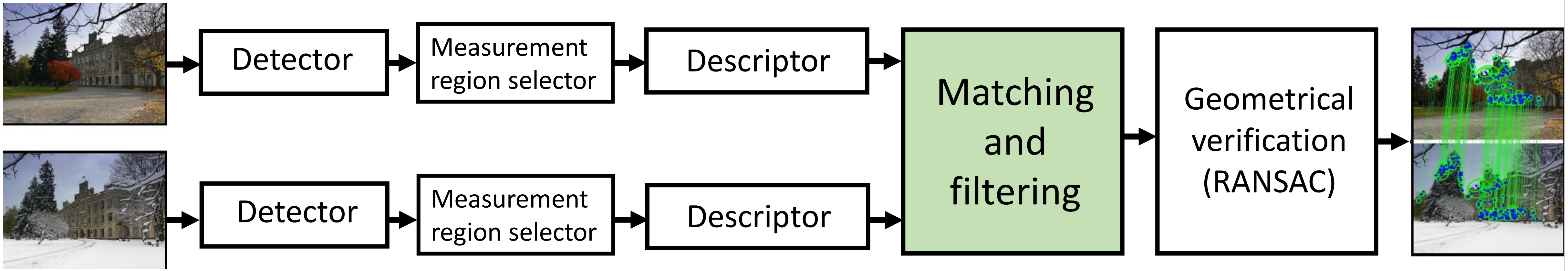
You need to tune each RANSAC for each local feature

	η_{PyR}	η_{DEGEN}	η_{GCR}	η_{MAG}	r_{stereo}	$r_{\text{multiview}}$
CV-SIFT	0.25	0.5	0.5	1.25	0.85	0.75
CV-$\sqrt{\text{SIFT}}$	0.25	0.5	0.5	1.25	0.85	0.85
CV-SURF	0.75	0.75	0.75	2	0.85	0.90
CV-AKAZE	0.25	0.75	0.75	1.5	0.85	0.90
CV-ORB	0.75	1	1.25	2	0.85	0.95
CV-FREAK	0.5	0.5	0.75	2	0.85	0.85
VL-DoG-SIFT	0.25	0.5	0.5	1.5	0.85	0.80
VL-DoGAff-SIFT	0.25	0.5	0.5	1.5	0.85	0.80
VL-Hess-SIFT	0.2	0.5	0.5	1.5	0.85	0.80
VL-HessAffNet-SIFT	0.25	0.5	0.5	1	0.85	0.80
CV-DoG/HardNet	0.25	0.5	0.5	1.5	0.90	0.80
KeyNet/Hardnet	0.5	0.75	0.75	2	0.95	0.85
CV-DoG/L2Net	0.2	0.5	0.5	1.5	0.90	0.80
CV-DoG/GeoDesc	0.2	0.5	0.75	1.5	0.90	0.85
ContextDesc	0.25	0.75	0.5	1	0.95	0.85
CV-DoG/SOSNet	0.25	0.5	0.75	1.5	0.90	0.80
CV-DoG/LogPolarDesc	0.2	0.5	0.5	1.5	0.90	0.80
D2-Net (SS)	1	2	2	7.5	—	—
D2-Net (MS)	1	2	2	5	—	—
R2D2 (wasf-n8-big)	0.75	1.25	1.25	2	—	0.95

And per matching method

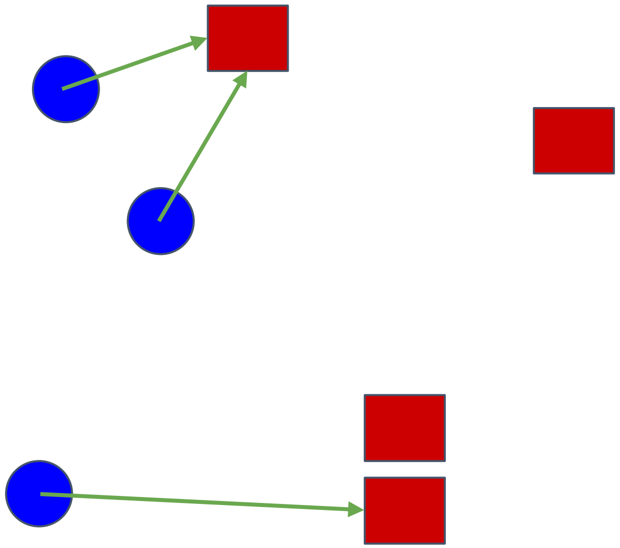
F-summary: recommendations

- If you haven’t tuned the RANSAC, even the best local feature would not work
- Performance of different RANSACs varies significantly, and all the methods have to be tuned to perform well
- Don’t use OpenCV or sk-image F-RANSACs, use GC-RANSAC, MAGSAC or DEGENSAC (all available with python bindings)
- Implementation matters (see USAC fail)



Matching and filtering strategies

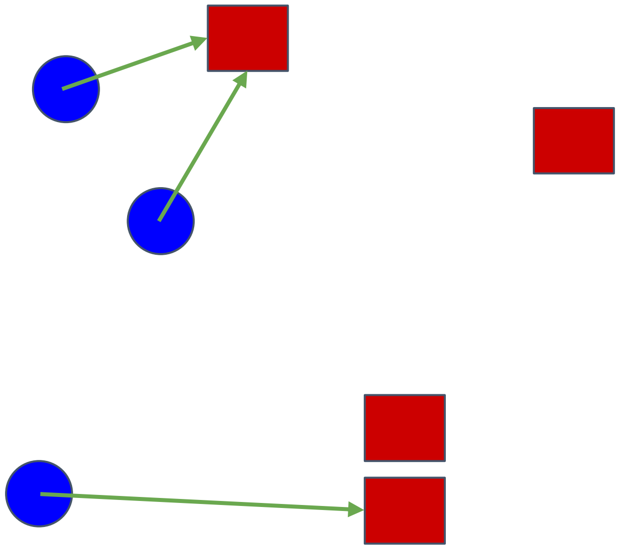
Nearest neighbor (NN) strategy



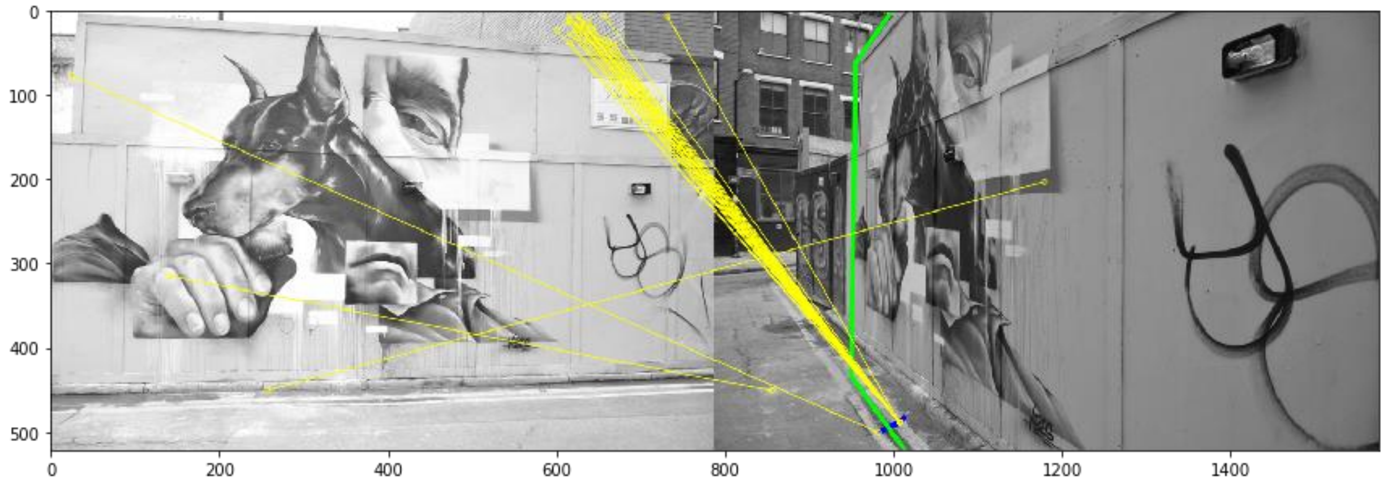
Features from img1 are matched to features from img2

You can see, that it is asymmetric and allowing “many-to-one” matches

Nearest neighbor (NN) strategy

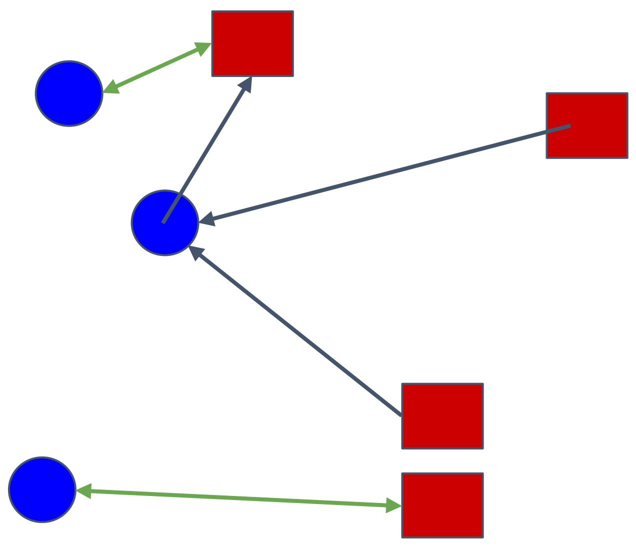


Features from img1 are matched to features from img2



OpenCV RANSAC failed to find a good model with NN matching
Found 1st image projection: blue, ground truth: green,
Inlier correspondences: yellow

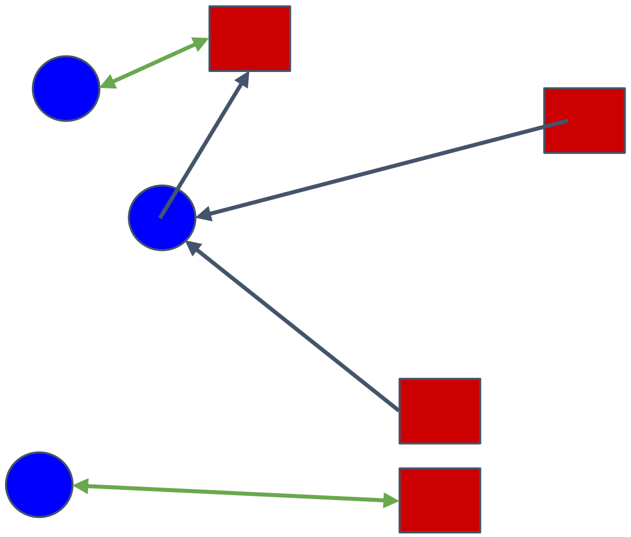
Mutual nearest neighbor (MNN) strategy



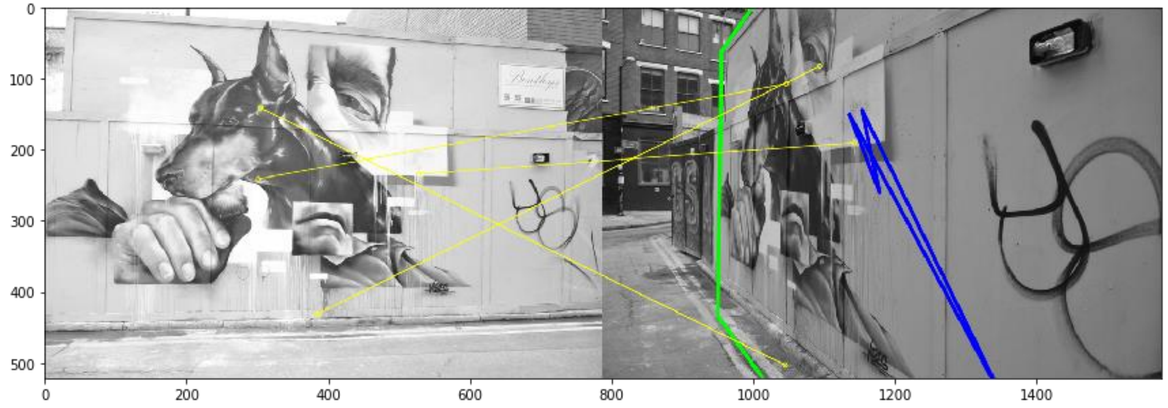
Features from img1 are matched to features from img2

Only cross-consistent (mutual NNs) matches are retained.

Mutual nearest neighbor (MNN) strategy



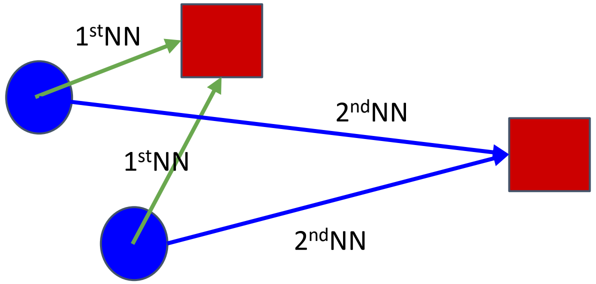
Features from img1 are matched to features from img2
Only cross-consistent (mutual NNs) matches are retained.

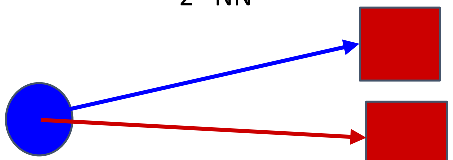


OpenCV RANSAC failed to find a good model with MNN matching
No one-to-many connections, but still bad
Found 1st image projection: blue, ground truth: green ,
inlier correspondences: yellow

Second nearest neighbor ratio (SNN) strategy

1^{st}NN/2^{nd}NN < 0.8, keep

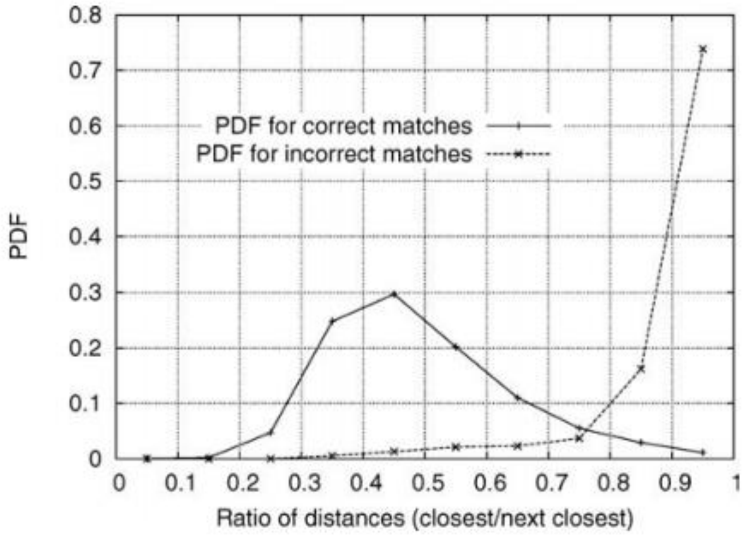




1^{st}NN/2^{nd}NN > 0.8, drop

Features from img1 are matched to features from img2

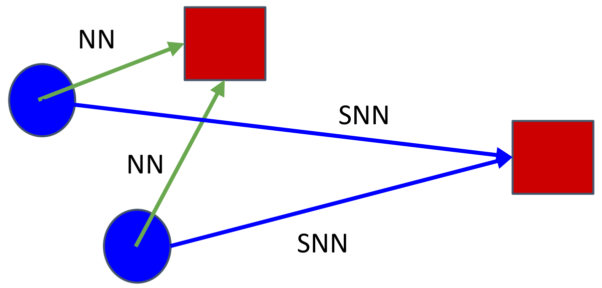
- we look for 2 nearest neighbors
  - If both are too similar (1^{st}NN/2^{nd}NN ratio > 0.8) → discard
  - If 1^{st} NN is much closer (1^{st}NN/2^{nd}NN ratio ≤ 0.8) →

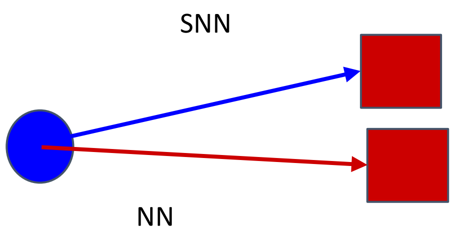


D. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004

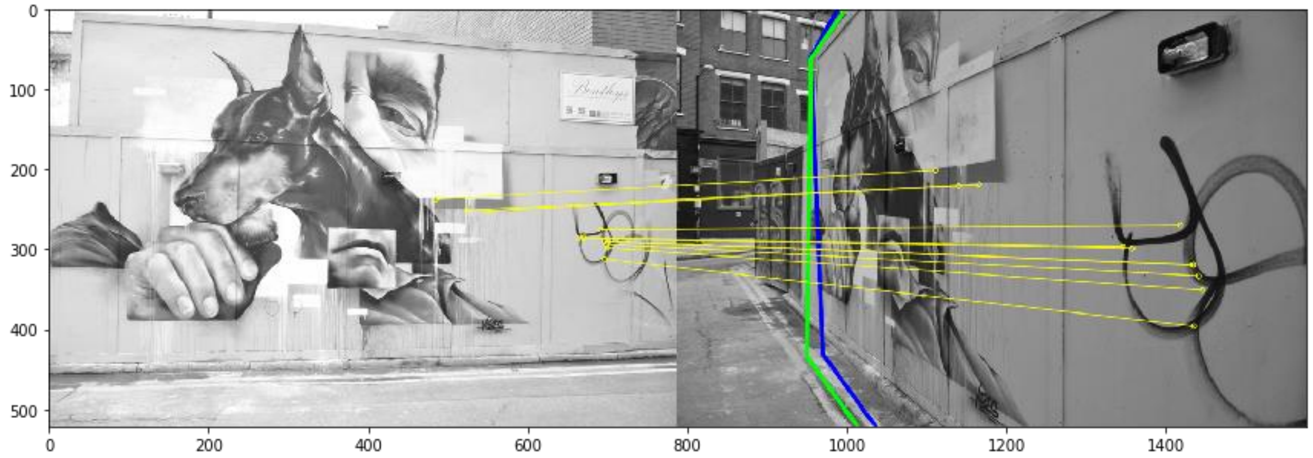
Second nearest neighbor ratio (SNN) strategy

NN/SNN < 0.8, keep





NN/SNN > 0.8, drop



OpenCV RANSAC found a model roughly correct
No one-to-many connections, but still bad
Found 1st image projection: blue, ground truth: green ,
inlier correspondences: yellow

1st geometrically inconsistent nearest neighbor ratio (FGINN) strategy

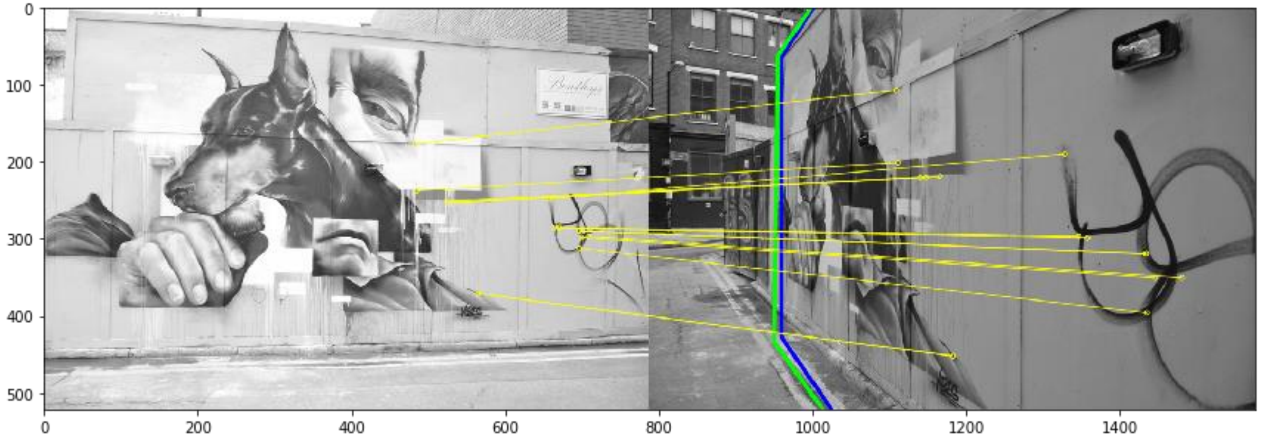
SNN ratio is cool, but what about symmetrical, or too closely detected features? Ratio test will kill them.
Solution: look for 2nd nearest neighbor, which is far enough from 1st nearest.

Mishkin et al., “MODS: Fast and Robust Method for Two-View Matching”, CVIU 2015

1st geometrically inconsistent nearest neighbor ratio (FGINN) strategy

SNN ratio is cool, but what about symmetrical, or too closely detected features? Ratio test will kill them.

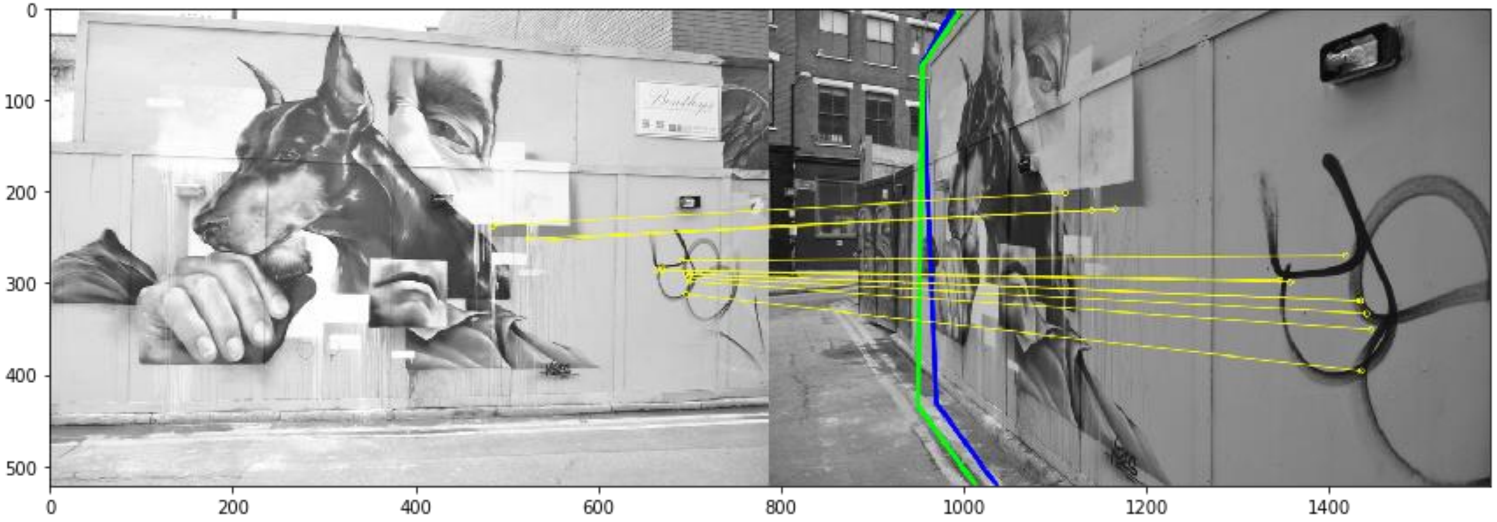
Solution: look for 2nd nearest neighbor, which is far enough from 1st nearest.



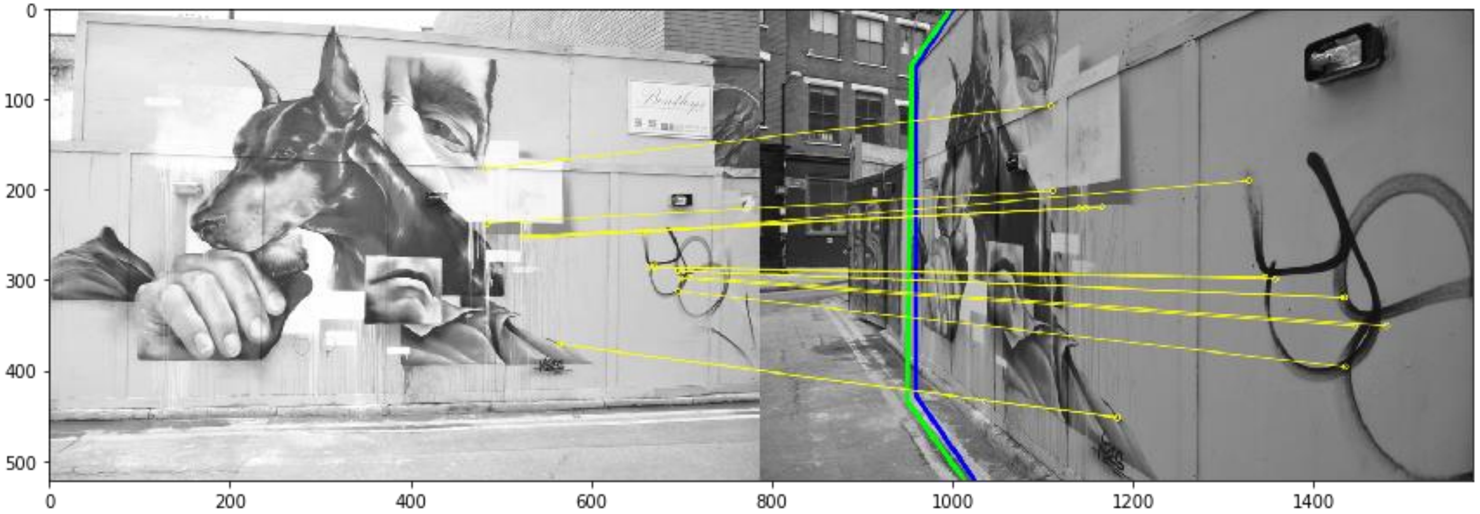
Mishkin et al., “MODS: Fast and Robust Method for Two-View Matching”, CVIU 2015

SNN vs FGINN

SNN: roughly correct



FGINN: more correspondences, better geometry found



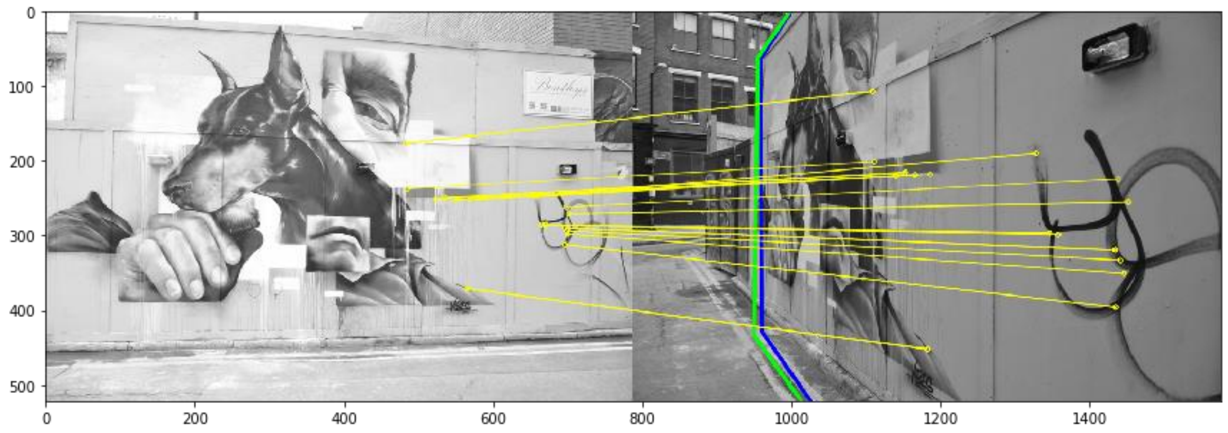
Mishkin et al., “MODS: Fast and Robust Method for Two-View Matching”, CVIU 2015

Symmetrical FGINN

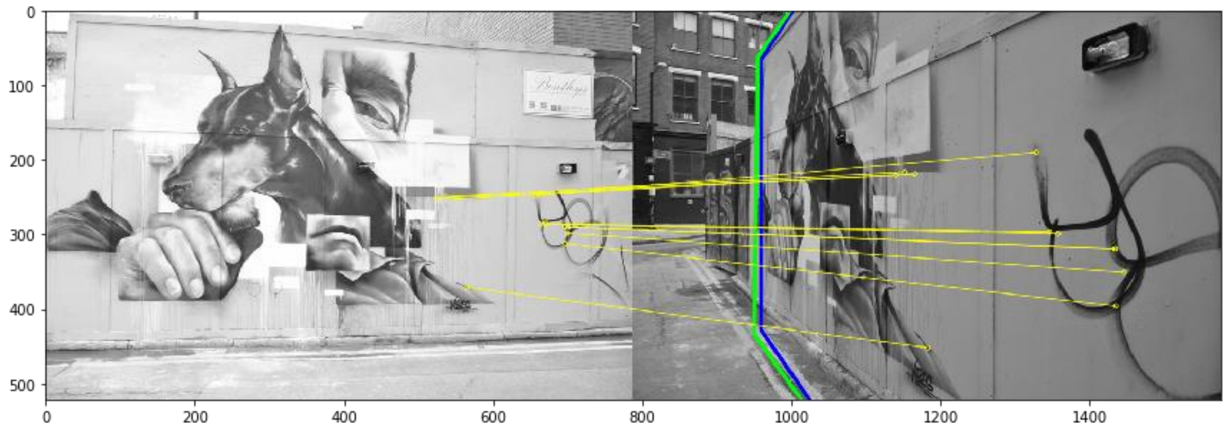
Recall, that FGINN is still asymmetric:
Matching (Img1 → Img2) ≠ (Img2 → Img1)

We can do both
(Img1 → Img2) and (Img2 → Img1)

and keep all FGINNs ("either" strategy)



or only cross-consistent FGINNs
("both" strategy)



You should tune matching threshold as well ☺

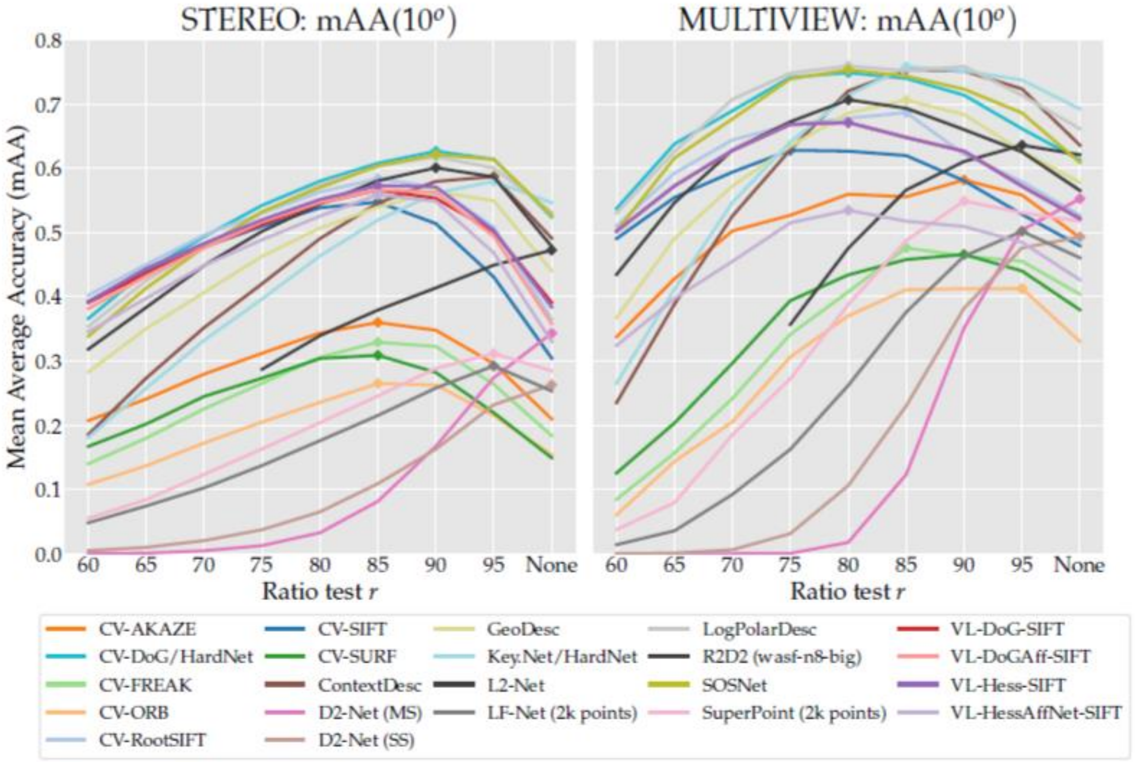


Fig. 10 Validation – Optimal ratio test r for matching with "both". We evaluate bidirectional matching with the "both" strategy (the best one), and different ratio test thresholds r, for each feature type. We use 8k features (2k for SuperPoint and LF-Net). For stereo, we use PyRANSAC.

- The best strategy is mutual NN + ratio test.
- By playing with threshold, you can make almost any local feature a "winner
- The ONLY local feature, not benefit from ratio test is D2Net

Best strategy is mutual NN + ratio test even for binary features

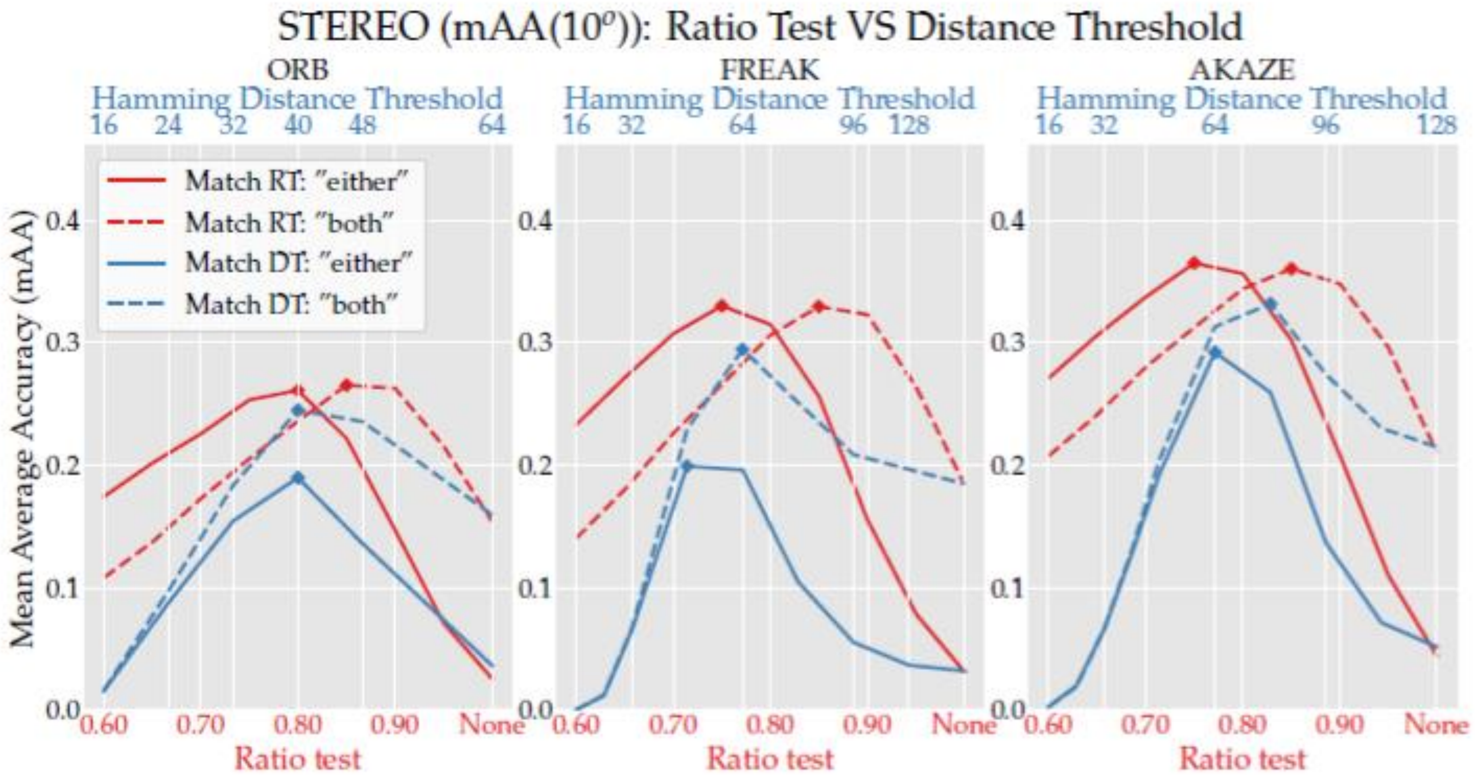
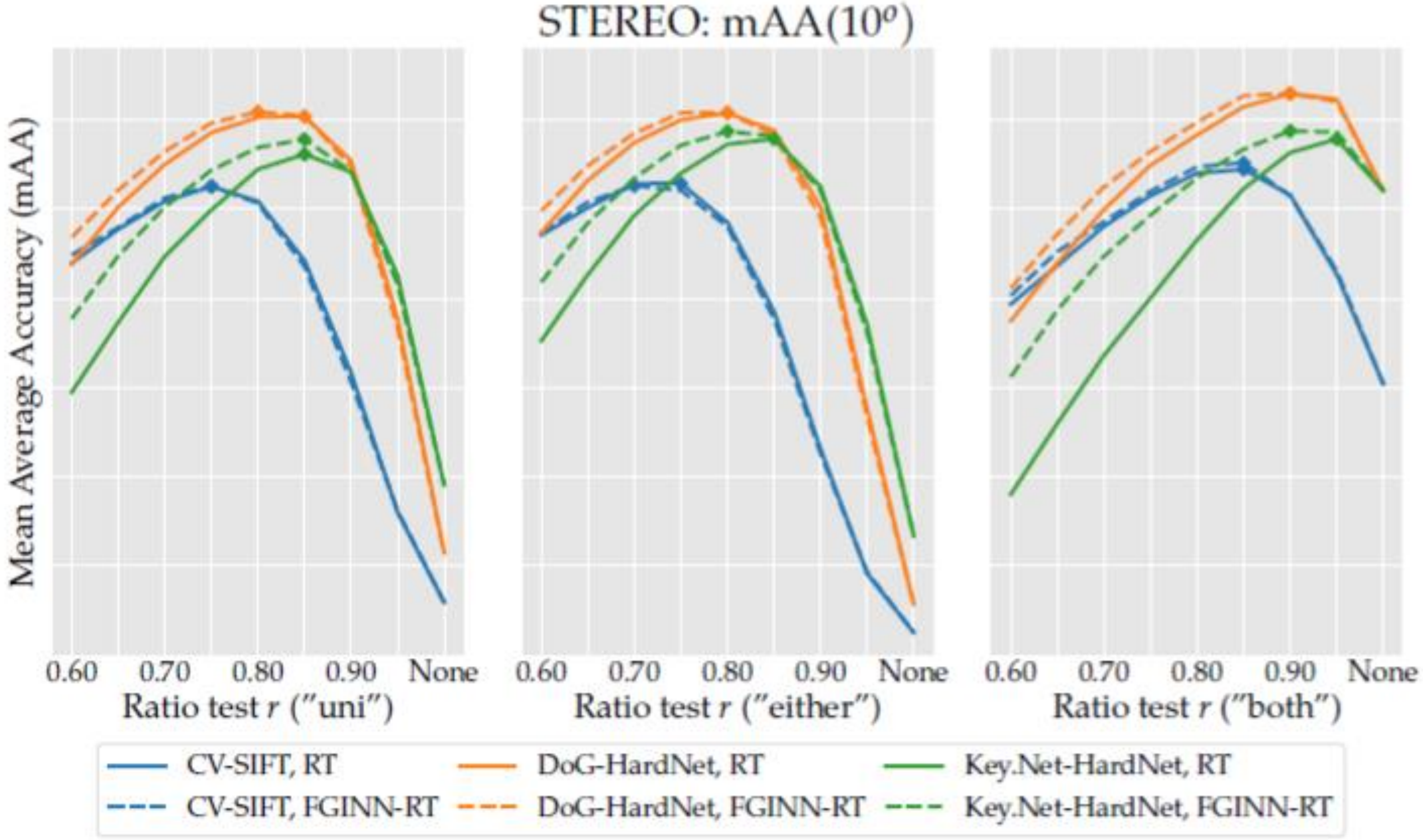
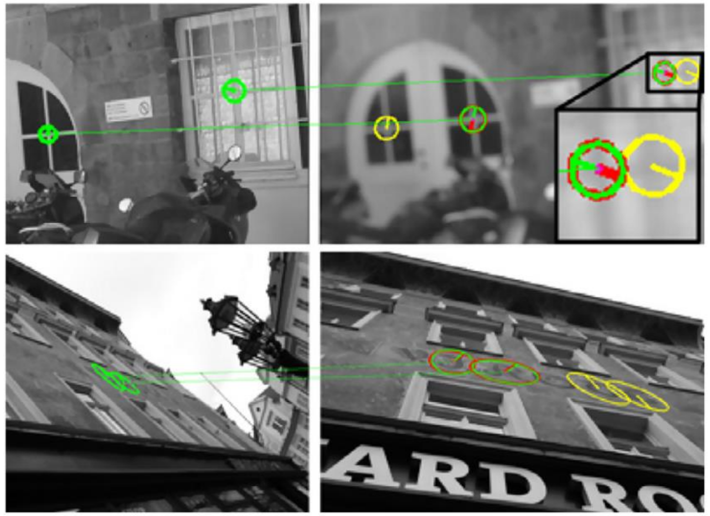


Fig. 13 Validation – Matching binary descriptors. We filter out non-discriminative matches with the ratio test or a distance threshold. The latter (the standard) performs worse in our experiments.

FGINN is less sensitive to the threshold value than SNN

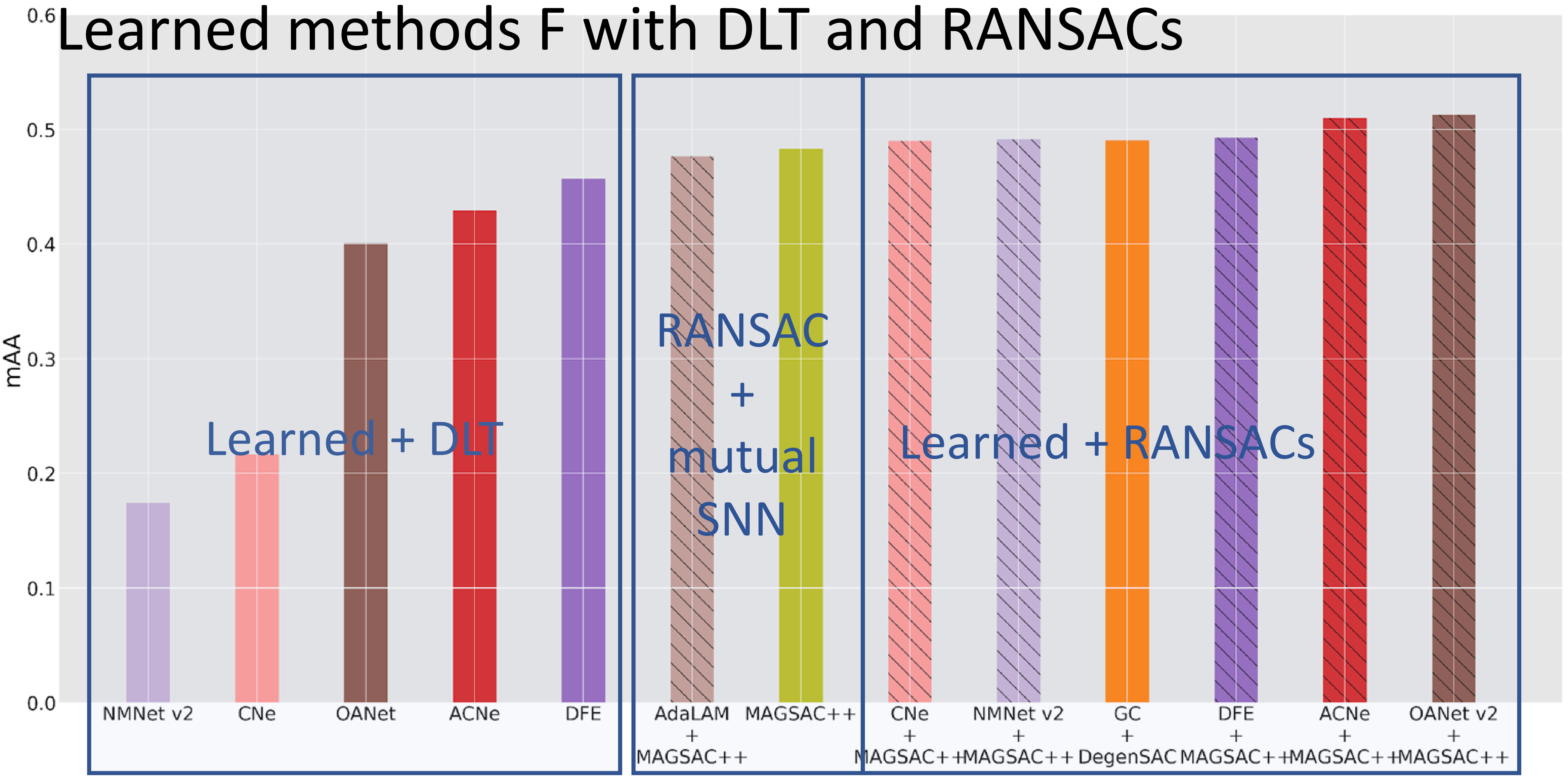




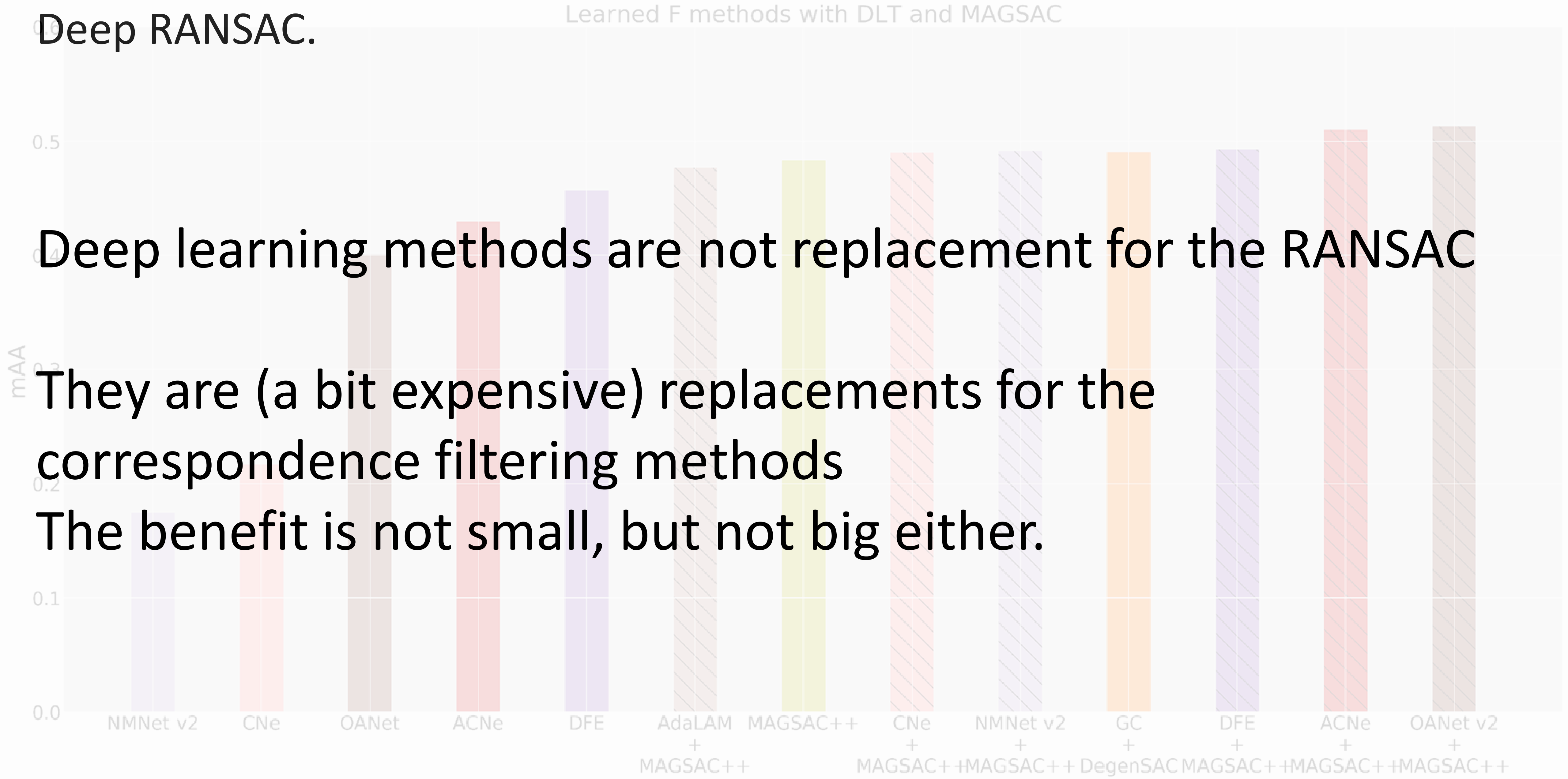
But there is (almost) no difference in performance, when both are tuned.

What about deep learning methods?

Learned methods F with DLT and RANSACs



Deep RANSAC.



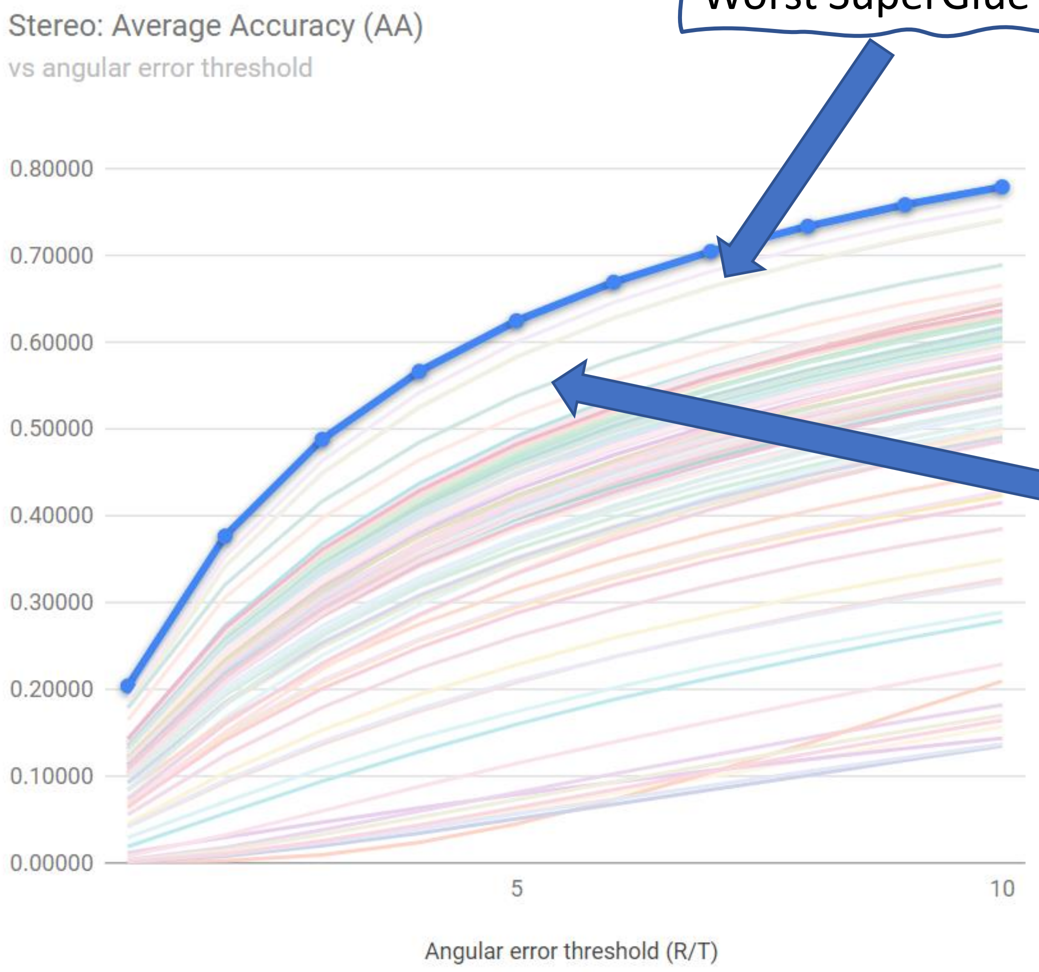
Deep learning methods are not replacement for the RANSAC

They are (a bit expensive) replacements for the correspondence filtering methods

The benefit is not small, but not big either.

Except…for SuperGlue

Worst SuperGlue submission (among 3 variants)



- SuperGlue takes much richer input, than the most of the methods in our study. SuperGlue uses all raw keypoints and descriptors from both images.

The best non-SuperGlue submission

SuperGlue: graph NN + optimal transport matcher

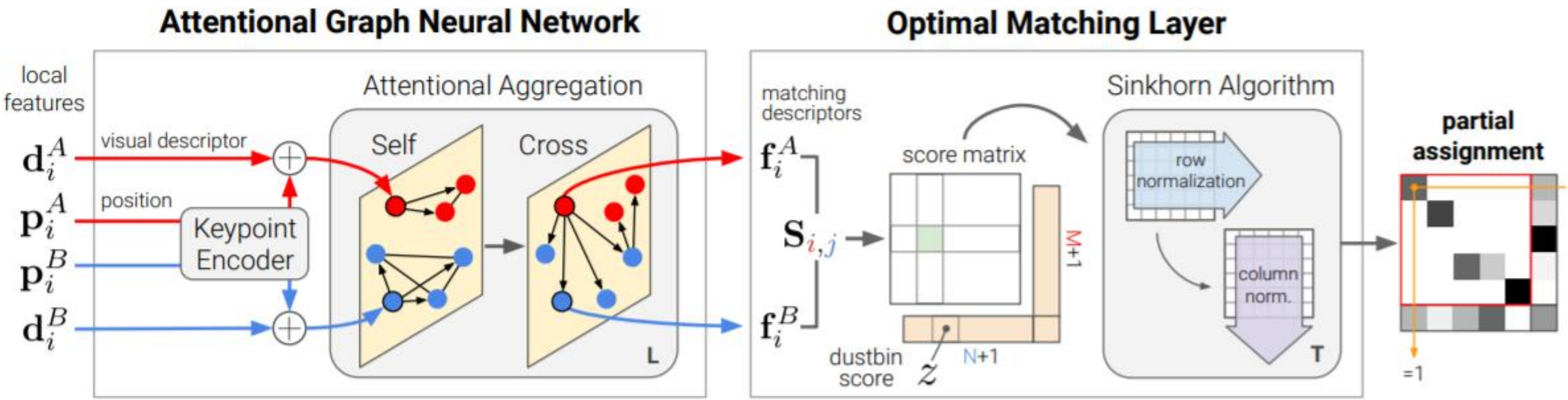
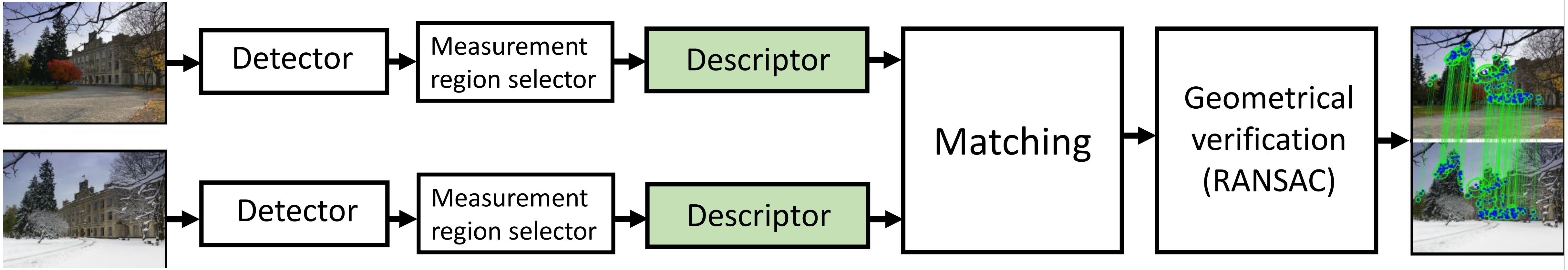


Figure 3: **The SuperGlue architecture.** SuperGlue is made up of two major components: the *attentional graph neural network* (Section 3.1), and the *optimal matching layer* (Section 3.2). The first component uses a *keypoint encoder* to map keypoint positions $\mathbf{p}$ and their visual descriptors $\mathbf{d}$ into a single vector, and then uses alternating self- and cross-attention layers (repeated L times) to create more powerful representations $\mathbf{f}$. The optimal matching layer creates an M by N score matrix, augments it with dustbins, then finds the optimal partial assignment using the Sinkhorn algorithm (for T iterations).

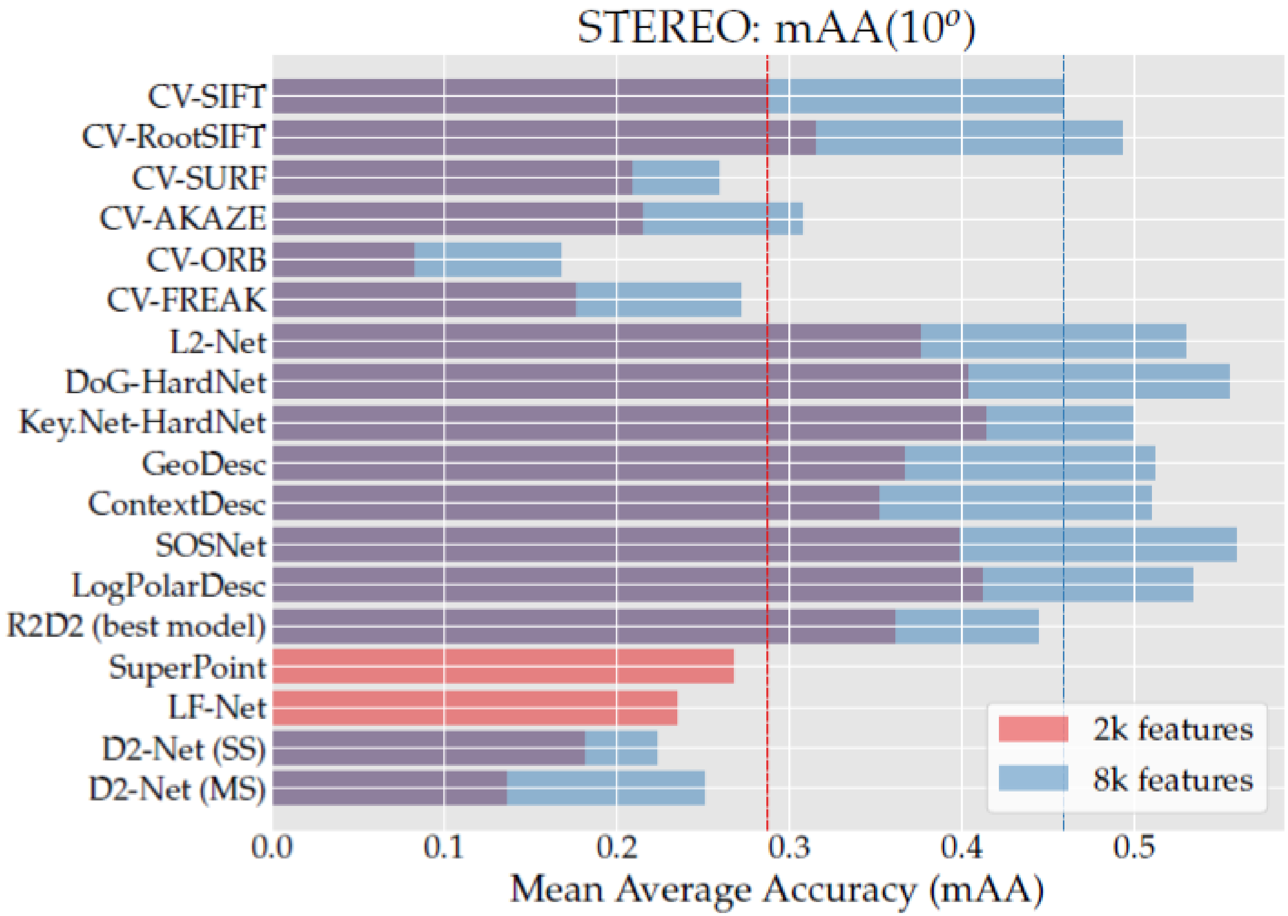
Sarlin et. al SuperGlue: Learning Feature Matching with Graph Neural Networks , arXiv 2019



Descriptor: HardNet (NIPS, 2017)

Mishchuk et.al. Working hard to know your neighbor’s margins: Local descriptor learning loss. NIPS 2017

DoG + HardNet is the state-of-the-art for stereo



- DoG-HardNet gain over rootSIFT is just 4% mAP
- All end-to-end learned methods are worse than SIFT
- Even with 2k keypoints

DoG + SoSNet is the state-of-the-art for stereo 8k

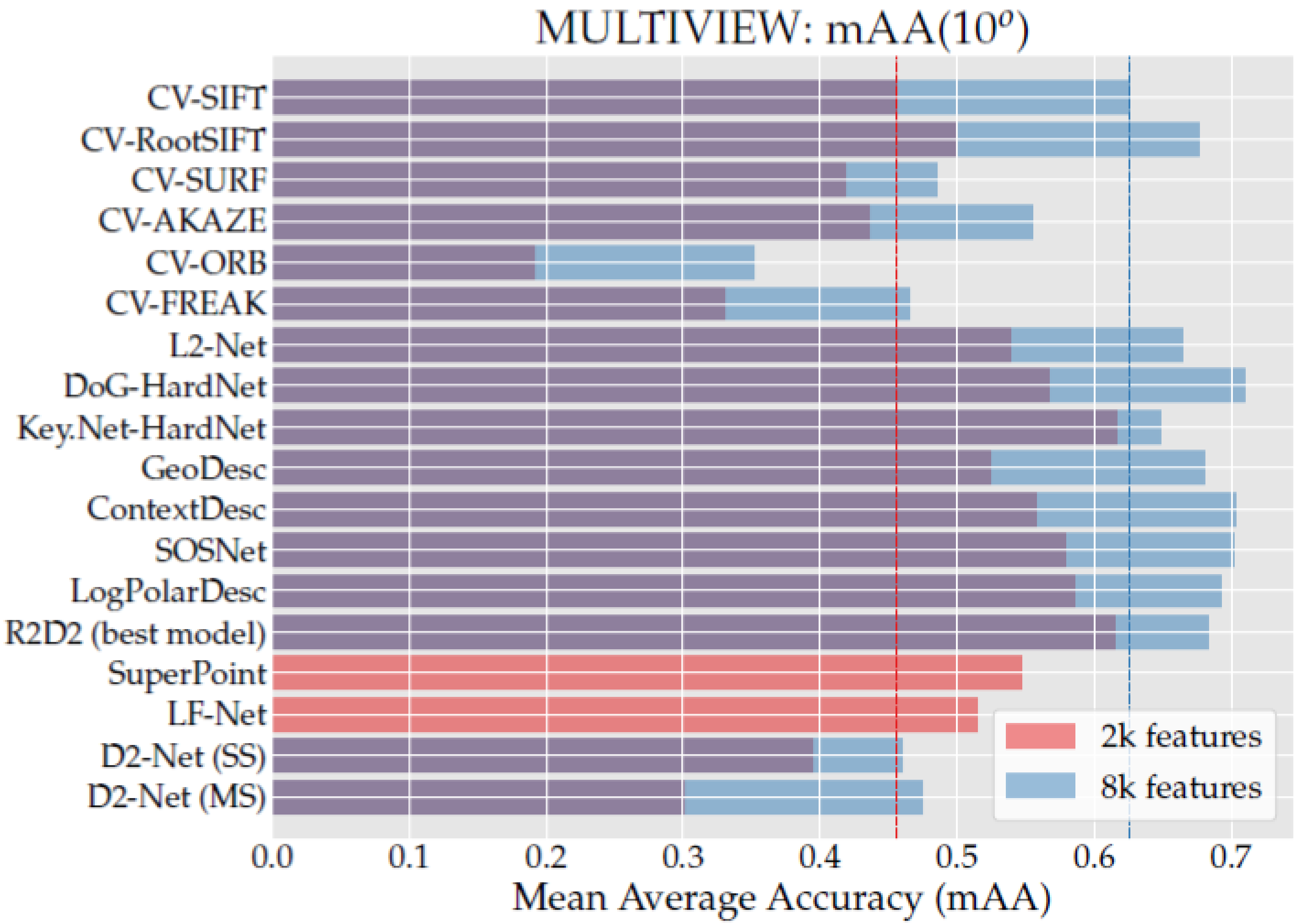
Method	NF	PyRANSAC NI↑	PyRANSAC mAA(10°)↑	DEGENSAC NI↑	DEGENSAC mAA(10°)↑	MAGSAC NI↑	MAGSAC mAA(10°)↑	Rank
CV-SIFT	7861.1	167.6	.3996	243.6	.4584	297.4	.4583	13
VL-SIFT	7880.6	179.7	.3999	261.6	.4655	326.2	.4633	12
VL-Hessian-SIFT	8000.0	204.4	.3695	290.2	.4450	348.9	.4335	14
VL-DoGAff-SIFT	7892.1	171.6	.3984	250.1	.4680	317.1	.4666	10
VL-HesAffNet-SIFT	8000.0	209.3	.3933	299.0	.4679	350.0	.4626	11
CV-$\sqrt{\text{SIFT}}$	7860.8	192.3	.4228	281.7	.4930	347.5	.4941	9
CV-SURF	7730.0	107.9	.2280	113.6	.2593	145.3	.2552	18
CV-AKAZE	7857.1	131.4	.2570	246.8	.3074	301.8	.3036	16
CV-ORB	7150.2	123.7	.1220	150.0	.1674	178.9	.1570	21
CV-FREAK	8000.0	123.3	.2273	131.0	.2711	196.7	.2656	17
L2-Net	7861.1	213.8	.4621	366.0	.5295	481.0	.5252	5
DoG-HardNet	7861.1	286.5	.4801	432.3	.5543	575.1	.5502	2
DoG-HardNetAmos+	7861.0	265.7	.4607	398.6	.5385	528.7	.5329	3
Key.Net-HardNet	7997.6	448.1	.3997	598.3	.4986	815.4	.4739	8
GeoDesc	7861.1	205.4	.4328	348.5	.5111	453.4	.5056	7
ContextDesc	7859.0	278.2	.4684	493.6	.5098	544.1	.5143	6
SOSNet	7861.1	281.6	.4784	424.6	.5587	563.3	.5517	1
LogPolarDesc	7861.1	254.4	.4574	441.8	.5340	591.2	.5238	4
D2-Net (SS)	5665.3	280.8	.1933	482.3	.2228	781.3	.2032	20
D2-Net (MS)	6924.1	278.2	.2160	470.6	.2506	741.2	.2321	19
R2D2 (wasf-n8-big)	7940.5	457.6	.3683	842.2	.4437	998.9	.4236	15

- But SoSNet/HardNet gain over rootSIFT is not that big.
- Results are not consistent with HPatches

KeyNet + HardNet is the state-of-the-art for stereo 2k

		PyRANSAC		DEGENSAC		MAGSAC		
Method	NF	NI↑	mAA(10°)↑	NI↑	mAA(10°)↑	NI↑	mAA(10°)↑	Rank
CV-SIFT	2048.0	84.9	.2489	79.0	.2875	99.2	.2805	11
CV-$\sqrt{\text{SIFT}}$	2048.0	84.2	.2724	88.3	.3149	106.8	.3125	9
CV-SURF	2048.0	37.9	.1725	72.7	.2086	87.0	.2081	14
CV-AKAZE	2048.0	96.1	.1780	91.0	.2144	115.5	.2127	13
CV-ORB	2031.8	56.3	.0610	63.5	.0819	71.5	.0765	18
CV-FREAK	2048.0	62.5	.1461	65.6	.1761	78.4	.1698	16
L2-Net	1936.3	66.1	.3131	92.4	.3752	114.7	.3691	5
DoG-HardNet	1936.3	111.9	.3508	117.7	.4029	150.5	.4033	4
Key.Net-HardNet	2048.0	134.4	.3272	174.8	.4139	228.4	.3897	1
GeoDesc	1936.3	98.9	.3127	103.9	.3662	129.7	.3640	6
ContextDesc	2048.0	118.8	.2965	124.1	.3510	146.4	.3485	8
SOSNet	1936.3	111.1	.3536	132.1	.3976	149.6	.4092	3
LogPolarDesc	1936.3	118.8	.3569	124.9	.4115	161.0	.4064	2
D2-Net (SS)	2045.6	107.6	.1157	134.8	.1355	259.3	.1317	17
D2-Net (MS)	2038.2	149.3	.1524	188.4	.1813	302.9	.1703	15
LF-Net	2020.3	100.2	.1927	106.5	.2344	141.0	.2226	12
SuperPoint	2048.0	120.1	.2577	126.8	.2964	127.3	.2676	10
R2D2 (wasf-n16)	2048.0	191.0	.2829	215.6	.3614	215.6	.3614	7

DoG + HardNet is the state-of-the-art for Multiview 8k



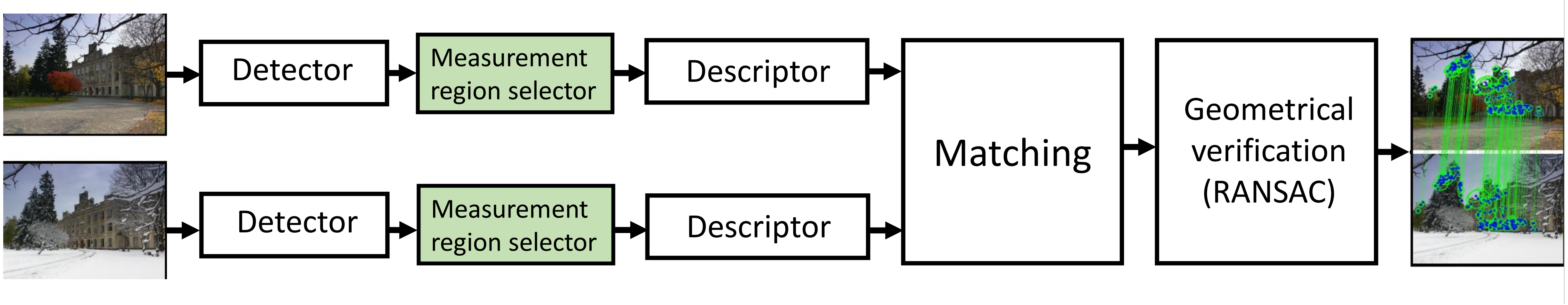
DoG + HardNet is the state-of-the-art for Multiview 8k

Method	NL↑	SR↑	RC↑	TL↑	mAA(5^o)↑	mAA(10^o)↑	ATE↓	Rank
CV-SIFT	2577.6	96.7	94.1	3.95	.5309	.6261	.4721	13
VL-SIFT	3030.7	97.9	95.4	4.17	.5273	.6283	.4669	12
VL-Hessian-SIFT	3209.1	97.4	94.1	4.13	.4857	.5866	.5175	15
VL-DoGAff-SIFT	3061.5	98.0	96.2	4.11	.5263	.6296	.4751	11
VL-HesAffNet-SIFT	3327.7	97.7	95.2	4.08	.5049	.6069	.4897	14
CV-$\sqrt{\text{SIFT}}$	3312.1	98.5	96.6	4.13	.5778	.6765	.4485	8
CV-SURF	2766.2	94.8	92.6	3.47	.3897	.4846	.6251	17
CV-AKAZE	4475.9	99.0	95.4	3.88	.4516	.5553	.5715	16
CV-ORB	3260.3	97.2	91.1	3.45	.2697	.3509	.7377	21
CV-FREAK	2859.1	92.9	91.7	3.53	.3735	.4653	.6229	19
L2-Net	3424.9	98.6	96.2	4.21	.5661	.6644	.4482	9
DoG-HardNet	4001.4	99.5	97.7	4.34	.6090	.7096	.4187	1
DoG-HardNetAmos+	3550.6	98.8	96.9	4.28	.5879	.6888	.4428	5
Key.Net-HardNet	3366.0	98.9	96.7	4.32	.5391	.6483	.4622	10
GeoDesc	3839.0	99.1	97.2	4.26	.5782	.6803	.4445	7
ContextDesc	3732.5	99.3	97.6	4.22	.6036	.7035	.4228	2
SOSNet	3796.0	99.3	97.4	4.32	.6032	.7021	.4226	3
LogPolarDesc	4054.6	99.0	96.4	4.32	.5928	.6928	.4340	4
D2-Net (SS)	5893.8	99.8	97.5	3.62	.3435	.4598	.6361	20
D2-Net (MS)	6759.3	99.7	98.2	3.39	.3524	.4751	.6283	18
R2D2 (wasf-n8-big)	4432.9	99.7	97.2	4.59	.5775	.6832	.4333	6

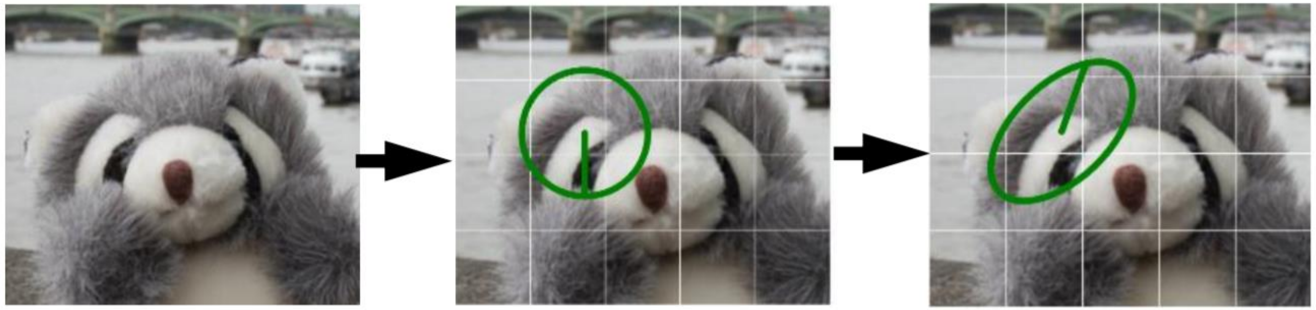
KeyNet+ HardNet is the state-of-the-art for Multiview 2k

Method	NL↑	SR↑	RC↑	TL↑	mAA(5°)↑	mAA(10°)↑	ATE↓	Rank
CV-SIFT	1081.2	87.6	87.4	3.70	.3718	.4562	.6136	12
CV-$\sqrt{\text{SIFT}}$	1174.7	90.3	89.4	3.82	.4074	.4995	.5589	11
CV-SURF	1186.6	90.2	88.6	3.55	.3335	.4184	.6701	14
CV-AKAZE	1383.9	94.7	90.9	3.74	.3393	.4361	.6422	13
CV-ORB	683.3	74.9	73.0	3.21	.1422	.1914	.8153	18
CV-FREAK	1075.2	87.2	86.3	3.52	.2578	.3297	.7169	16
L2-Net	1253.3	94.7	92.6	3.96	.4369	.5392	.5419	8
DoG-HardNet	1338.2	96.3	93.7	4.03	.4624	.5661	.5093	5
Key.Net-HardNet	1276.3	97.8	95.7	4.49	.5050	.6161	.4902	1
GeoDesc	1133.6	93.6	91.3	4.02	.4246	.5244	.5455	9
ContextDesc	1504.9	95.6	93.3	3.92	.4529	.5568	.5327	6
SOSNet	1317.4	96.0	93.8	4.05	.4739	.5784	.5194	4
LogPolarDesc	1410.2	96.0	93.8	4.05	.4794	.5849	.5090	3
D2-Net (SS)	2357.9	98.9	94.7	3.39	.2875	.3943	.7010	15
D2-Net (MS)	2177.3	98.2	93.4	3.01	.1921	.3007	.7861	17
LF-Net	1385.0	95.6	90.4	4.14	.4156	.5141	.5738	10
SuperPoint	1184.3	95.6	92.4	4.34	.4423	.5464	.5457	7
R2D2 (wasf-n16)	1228.4	99.4	96.2	4.29	.5045	.6149	.4956	2

R2D2 is runner-up



Measurement region selector: orientation



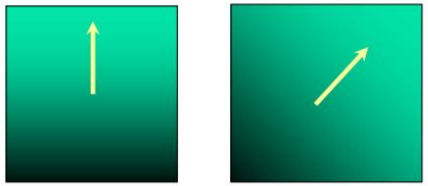
Which patch should we describe?

Detector: x, y, scale
Should we rotate patch?
Should we deform patch?

Handcrafted: dominant orientatior

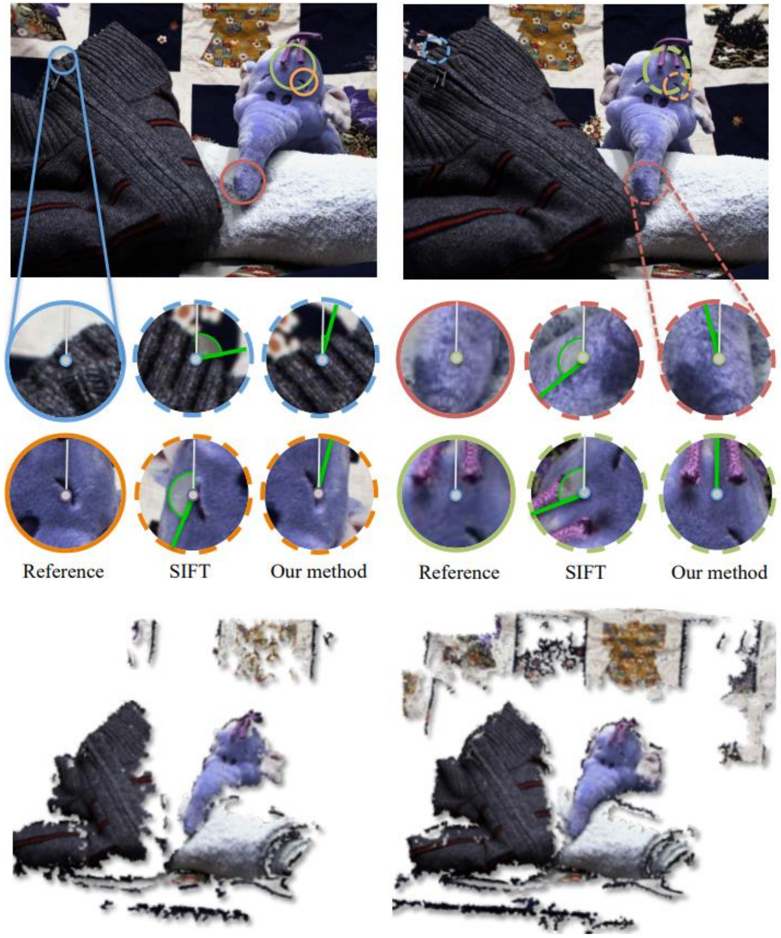
- Find local orientation

Dominant direction of gradient



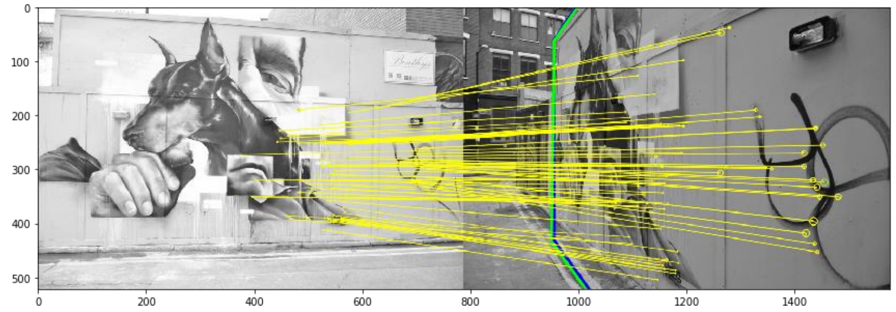
D. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004

Learned orientation: CNN



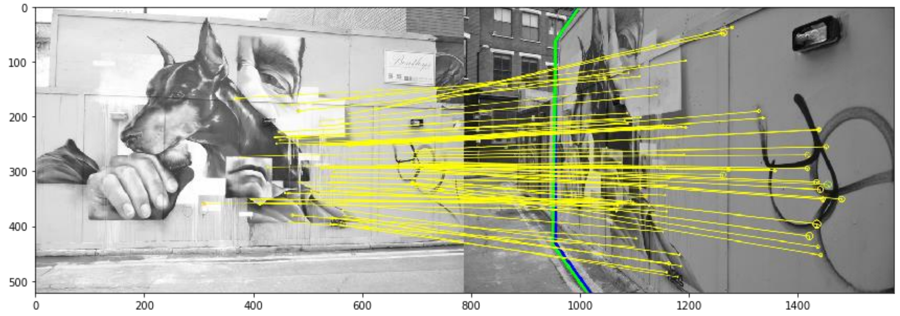
Yi et al. Learning to Assign Orientations to Feature Points CVPR 2016

If images are upright for sure: don`t detect orientation

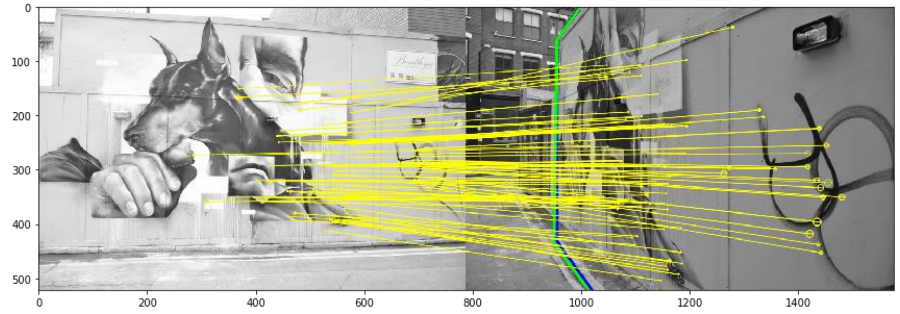


Dominant gradient orientation:
123 inliers

DoG + HardNet matches +FGINN union + RANSAC.
Found 1st image projection: blue,
ground truth: green ,
inlier correspondences: yellow



Learned orientation:
140 inliers



Constant orientation:
181 inliers

If images are upright for sure: don`t detect orientation

	CV-$\sqrt{\text{SIFT}}$		HardNet		SOSNet		LogPolarDesc	
	NI↑	mAA(10^o)↑	NI↑	mAA(10^o)↑	NI↑	mAA(10^o)↑	NI↑	mAA(10^o)↑
Standard	281.7	0.4930	432.3	0.5543	424.6	0.5587	441.8	0.5340
Upright	270.0	0.4878	449.2	0.5542	432.9	0.5554	461.8	0.5409
Δ (%)	-4.15	-1.05	+3.91	-0.02	+1.95	-0.59	+4.53	+1.29
Upright++	358.9	0.5075	527.6	0.5728	508.4	0.5738	543.2	0.5510
Δ (%)	+27.41	+2.94	+22.04	+3.34	+19.74	+2.70	+22.95	+3.18

Upright: angle -> zero, remove duplicates.

Upright++: angle -> zero, remove duplicates, get more features until again 8k

If images are upright for sure: don`t detect orientation

	CV-$\sqrt{\text{SIFT}}$		HardNet		SOSNet		LogPolarDesc	
	NL$\uparrow$	mAA(10^o)$\uparrow$	NL$\uparrow$	mAA(10^o)$\uparrow$	NL$\uparrow$	mAA(10^o)$\uparrow$	NL$\uparrow$	mAA(10^o)$\uparrow$
Standard	3312.1	0.6765	4001.4	0.7096	3796.0	0.7021	4054.6	0.6928
Upright	3485.1	0.6572	3594.6	0.6962	4025.1	0.7054	3737.4	0.6934
Δ (%)	+5.22	-2.85	-10.17	-1.89	+6.04	+0.47	-7.82	+0.09
Upright++	4404.6	0.6792	4250.4	0.7231	3988.6	0.7129	4414.1	0.7109
Δ (%)	+32.99	+0.40	+6.22	+1.90	+5.07	+1.54	+8.87	+2.61

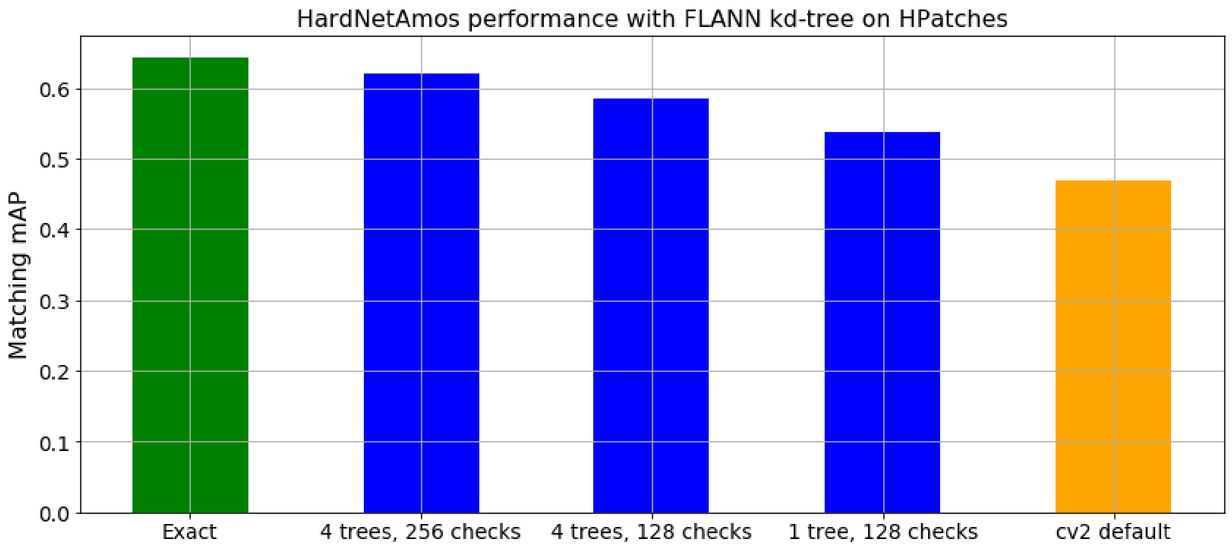
Upright: angle -> zero, remove duplicates.

Upright++: angle -> zero, remove duplicates, get more features until again 8k

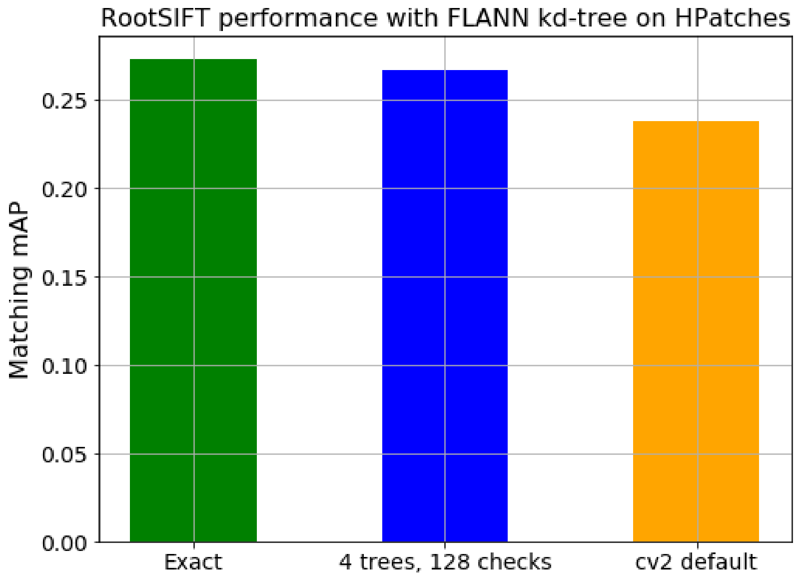
Approximate nearest neighbor search: not for free

HPatches matching score: exact search vs tuned FLANN vs OpenCV default

HardNet



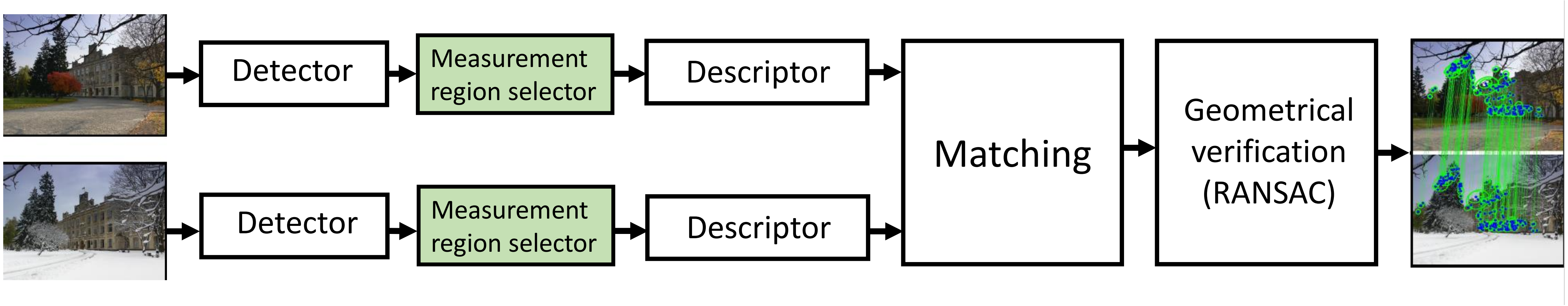
RootSIFT



Approximate nearest neighbor search: not for free

1-18% mAA loss

	CV-$\sqrt{\text{SIFT}}$		HardNet		SOSNet		D2Net	
	NI↑	mAA(10^o)↑	NI↑	mAA(10^o)↑	NI↑	mAA(10^o)↑	NI↑	mAA(10^o)↑
Exact	281.7	0.4930	432.0	0.5532	424.3	0.5575	470.6	0.2506
FLANN	274.6	0.4879	363.3	0.5222	339.8	0.5179	338.9	0.2046
Δ (%)	-2.52	-1.03	-15.90	-5.60	-19.92	-7.10	-27.99	-18.36



AffNet (ECCV 2018)
Measurement region selector

AffNet: learning measurement region

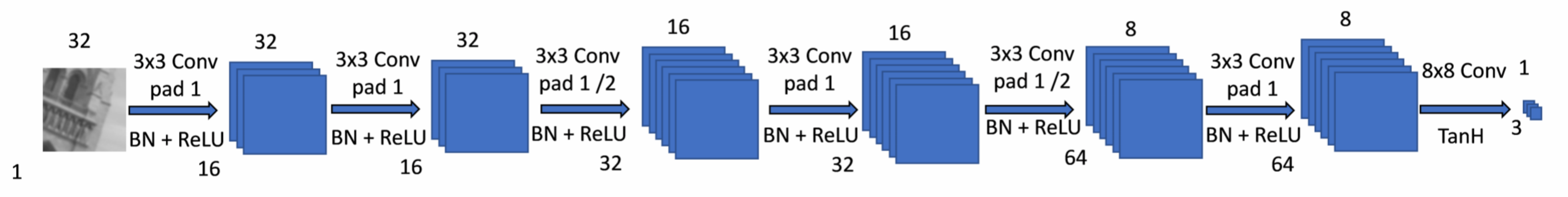
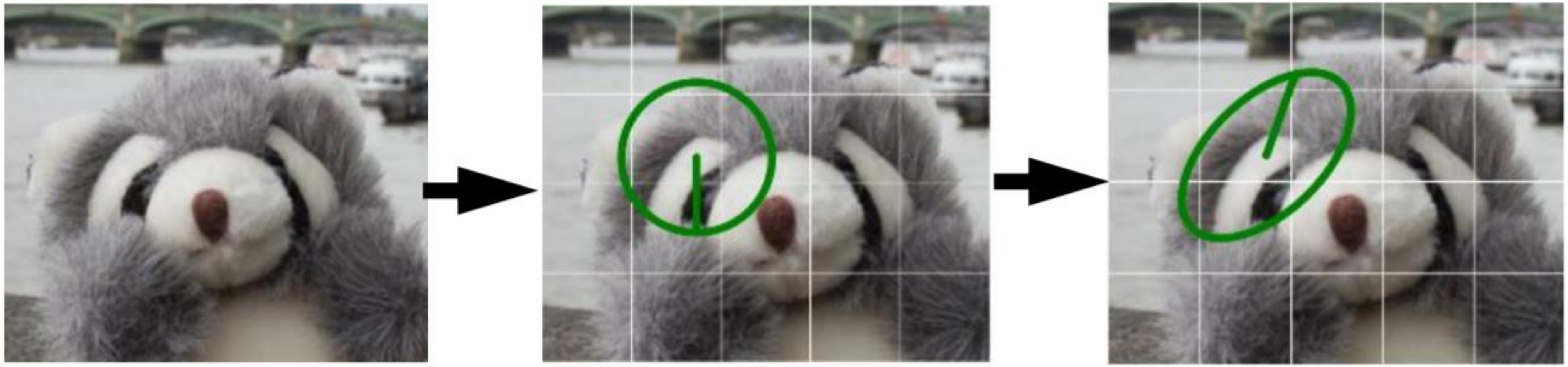
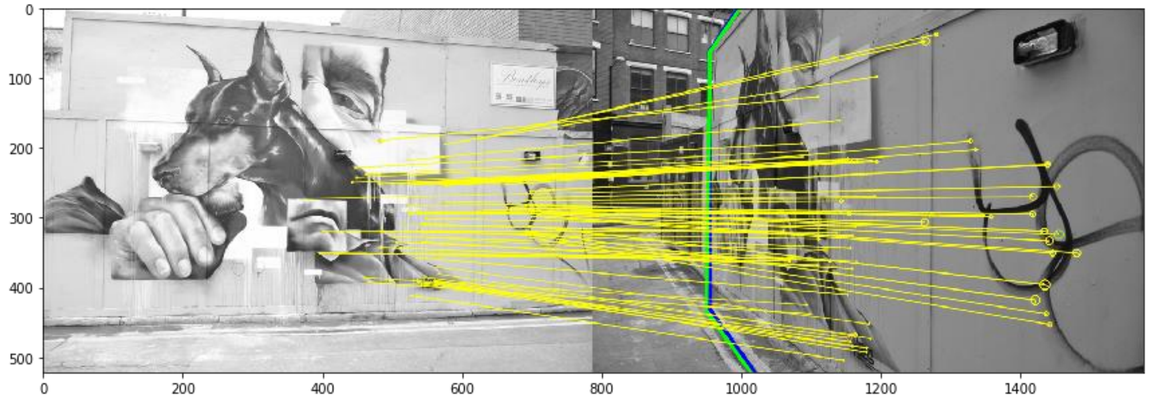


Fig. 4. AffNet. Feature map spatial size – top, # channels – bottom. /2 stands for stride 2.



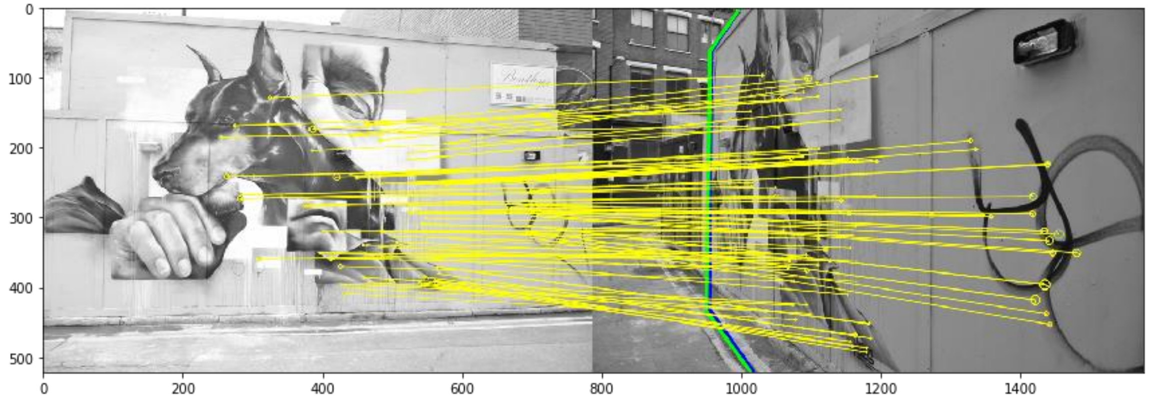
Mishkin et.al. Repeatability Is Not Enough: Learning Affine Regions via Discriminability. ECCV 2018

Do AffNet help? Yes, if the problem is hard



DoG + HardNetAmos: 123 inliers

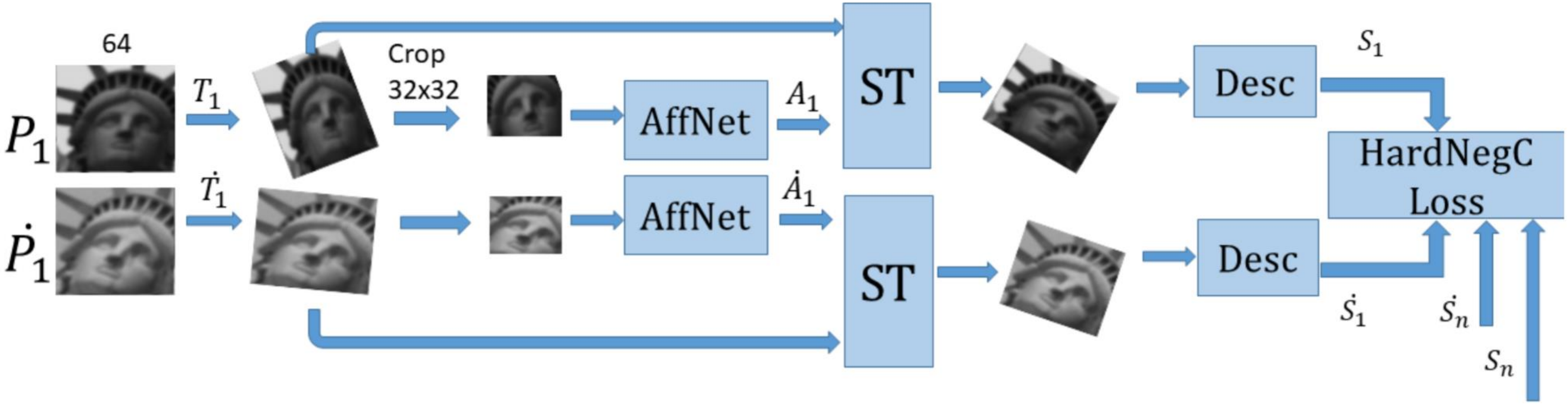
FGINN union + RANSAC.
Found 1st image projection: blue,
ground truth: green ,
inlier correspondences: yellow



DoG + AffNet + HardNetAmos : **165** inliers

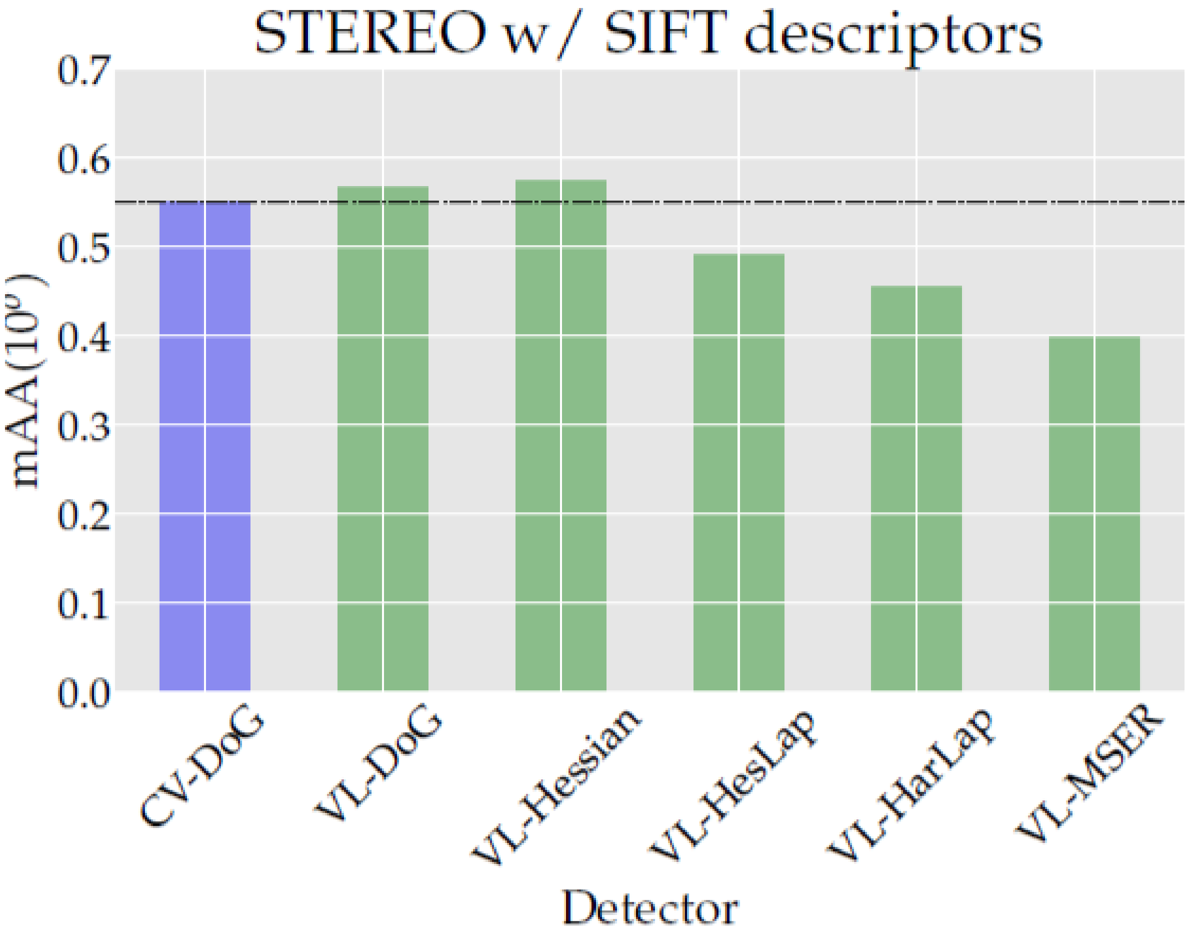
AffNet: learning measurement region

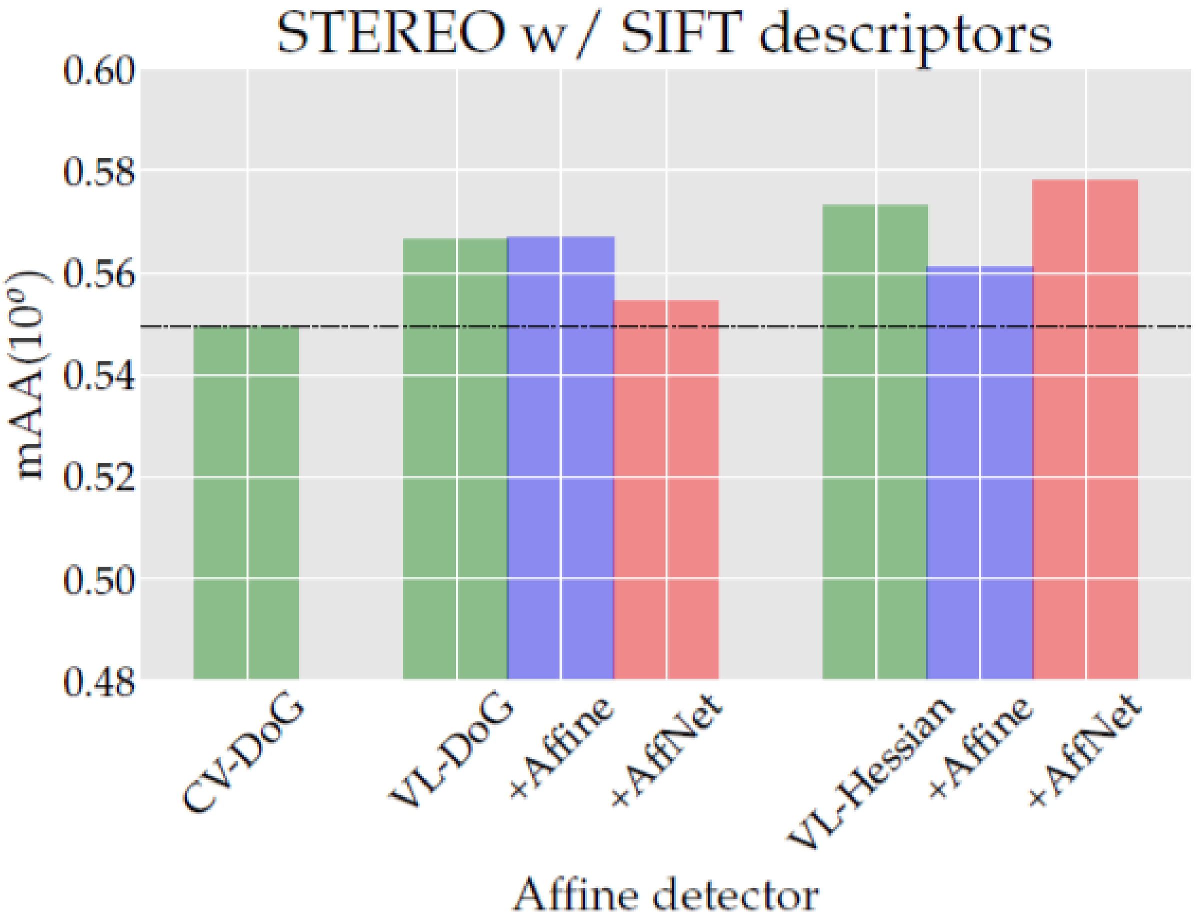
- Find affine shape such that maximizes difference between positive and hardest-in-batch negative examples



- Positive-only learning (Yi et. Al, CVPR2015) leads to degenerated ellipses
- Triplet margin (HardNet) – unstable in training affine shape

Does AffNet makes sense? Not for PhotoTourism data





Does AffNet makes sense? Not much for PhotoTourism data

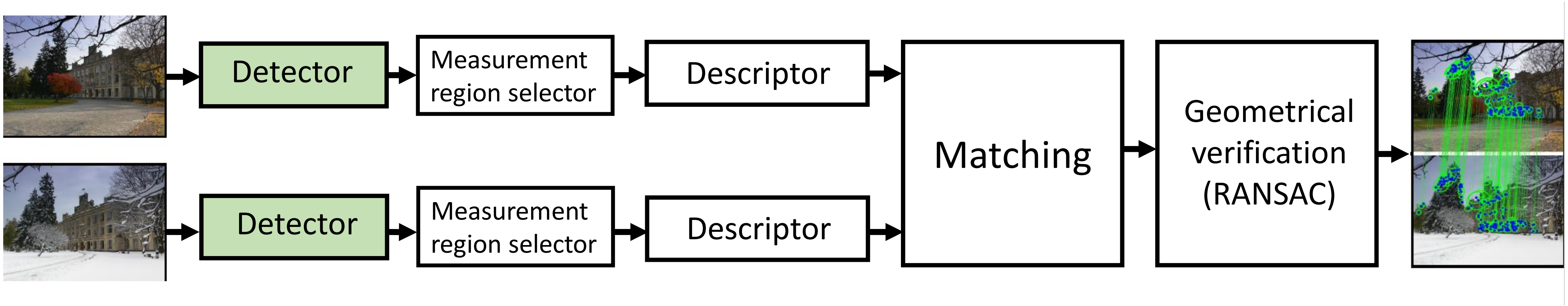
		PyRANSAC		DEGENSAC		MAGSAC		
Method	NF	NI↑	mAA(10^o)↑	NI↑	mAA(10^o)↑	NI↑	mAA(10^o)↑	Rank
CV-SIFT	7861.1	167.6	.3996	243.6	.4584	297.4	.4583	13
VL-SIFT	7880.6	179.7	.3999	261.6	.4655	326.2	.4633	12
VL-Hessian-SIFT	8000.0	204.4	.3695	290.2	.4450	348.9	.4335	14
VL-DoGAff-SIFT	7892.1	171.6	.3984	250.1	.4680	317.1	.4666	10
VL-HesAffNet-SIFT	8000.0	209.3	.3933	299.0	.4679	350.0	.4626	11

Stereo

Method	NL↑	SR↑	RC↑	TL↑	mAA(5^o)↑	mAA(10^o)↑	ATE↓	Rank
CV-SIFT	2577.6	96.7	94.1	3.95	.5309	.6261	.4721	13
VL-SIFT	3030.7	97.9	95.4	4.17	.5273	.6283	.4669	12
VL-Hessian-SIFT	3209.1	97.4	94.1	4.13	.4857	.5866	.5175	15
VL-DoGAff-SIFT	3061.5	98.0	96.2	4.11	.5263	.6296	.4751	11
VL-HesAffNet-SIFT	3327.7	97.7	95.2	4.08	.5049	.6069	.4897	14

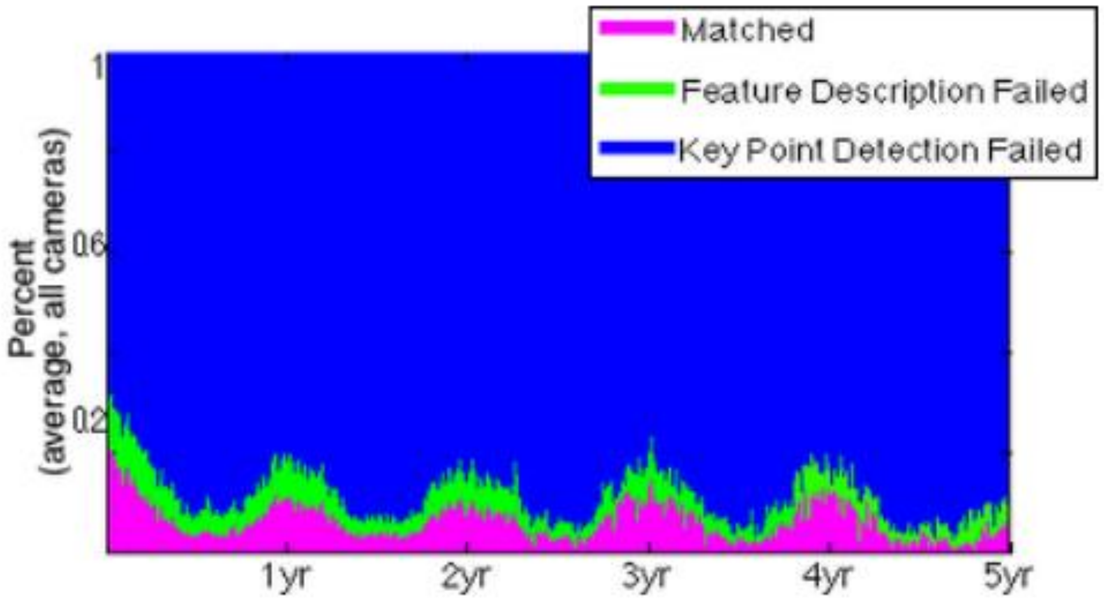
Multiview

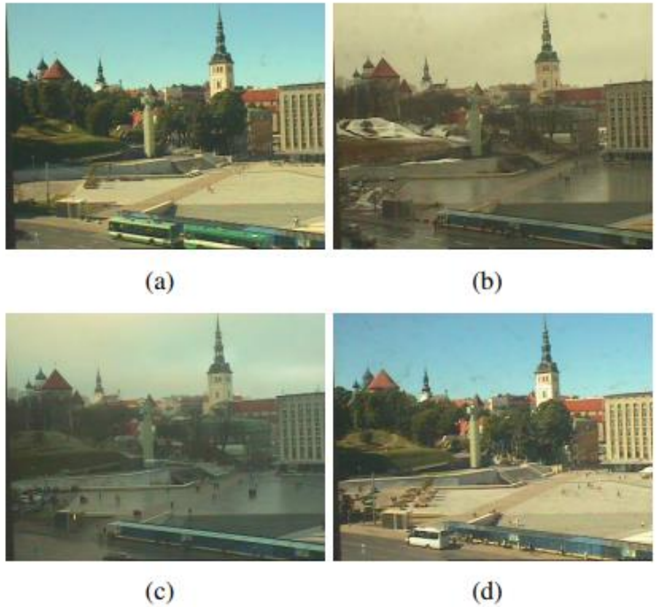
Local feature detector



Detector is the often failure point of the whole process

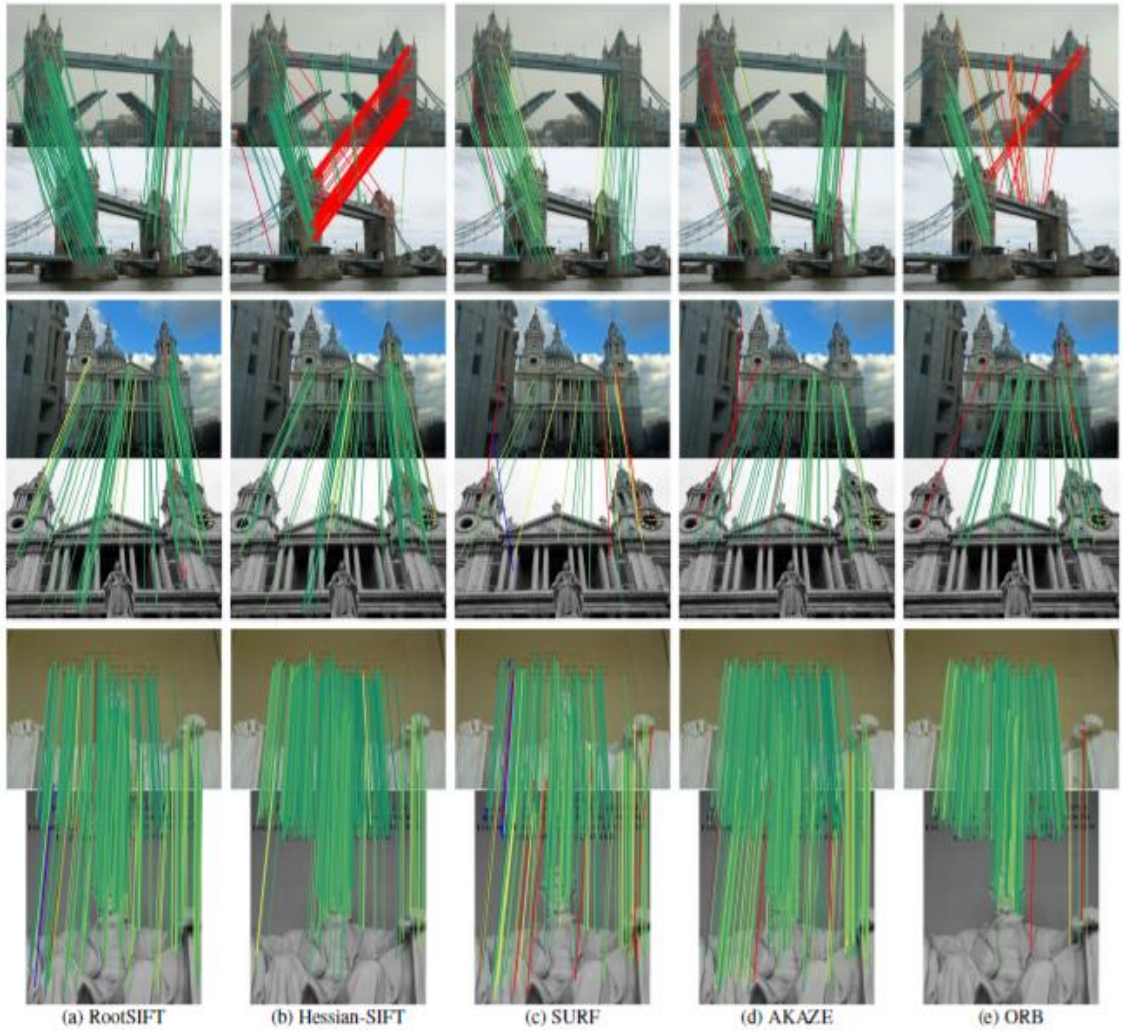
- Yet we still use 10-20 y.o stuff like SIFT or FAST, because nothing significantly better for practical purposed have been proposed
- So let`s stick to the basics

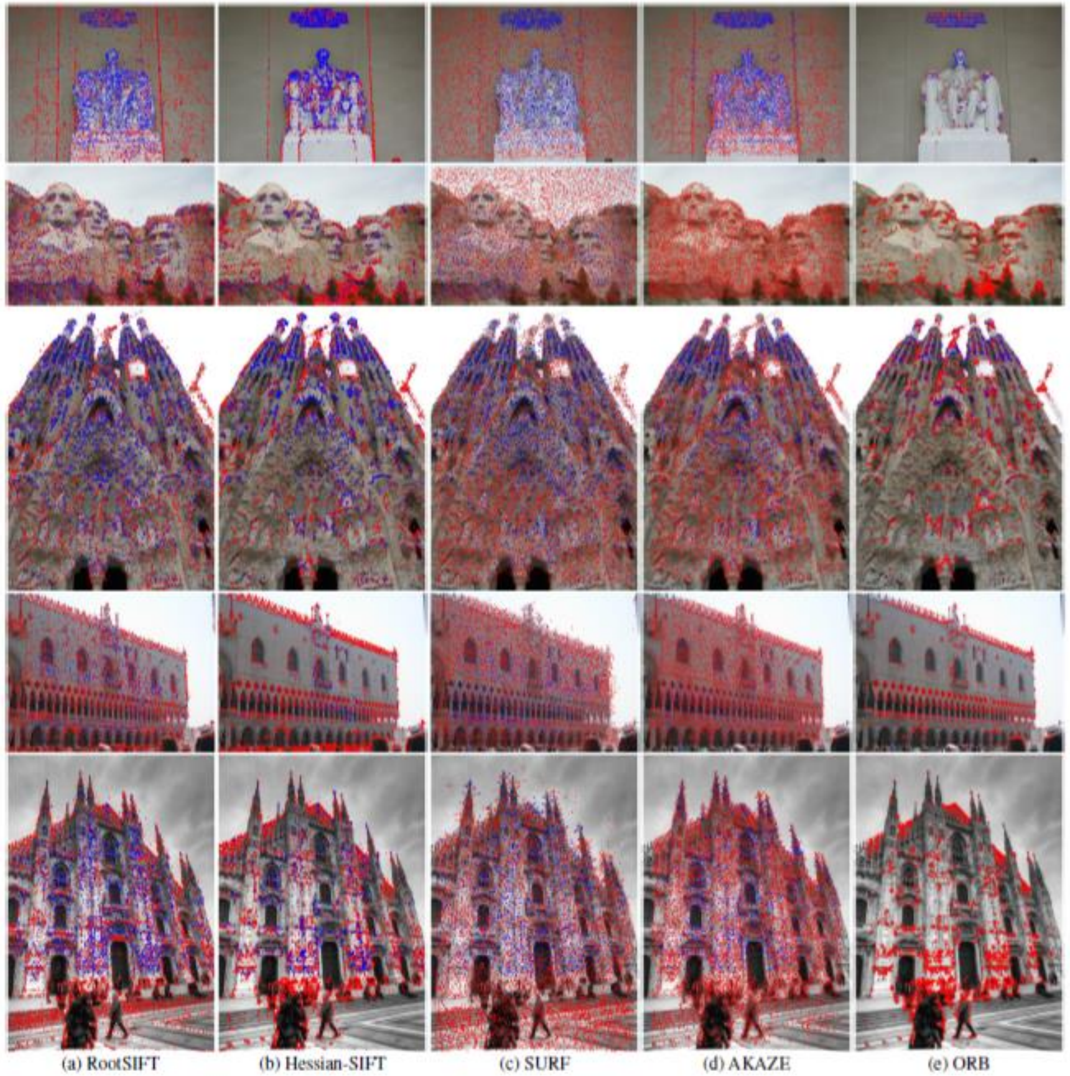




Stylianou et.al, WACV 2015. Characterizing Feature Matching Performance Over Long Time Periods

Qualitive comparison: classical features

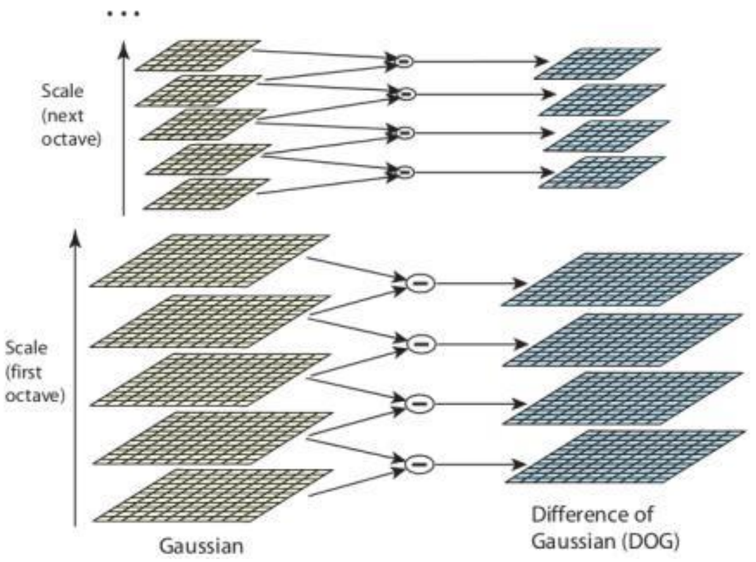




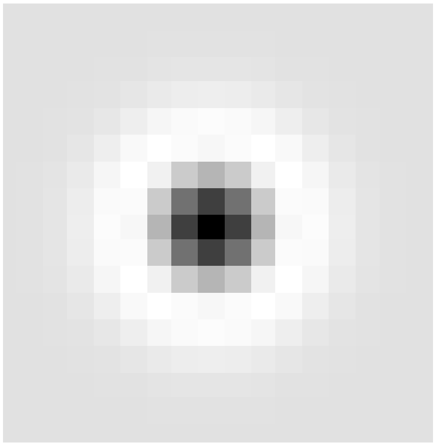
SIFT is the DoG detector + SIFT descriptor

- Really, there is not such thing, as SIFT detector.
- But everyone so got used to name DoG as SIFT ☹

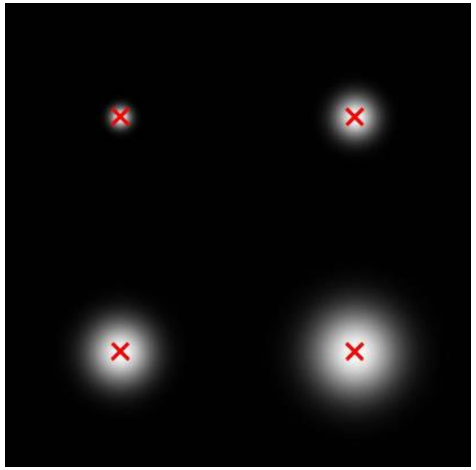
Gaussian scalespace, “stack of gradually smoothed versions” of original image



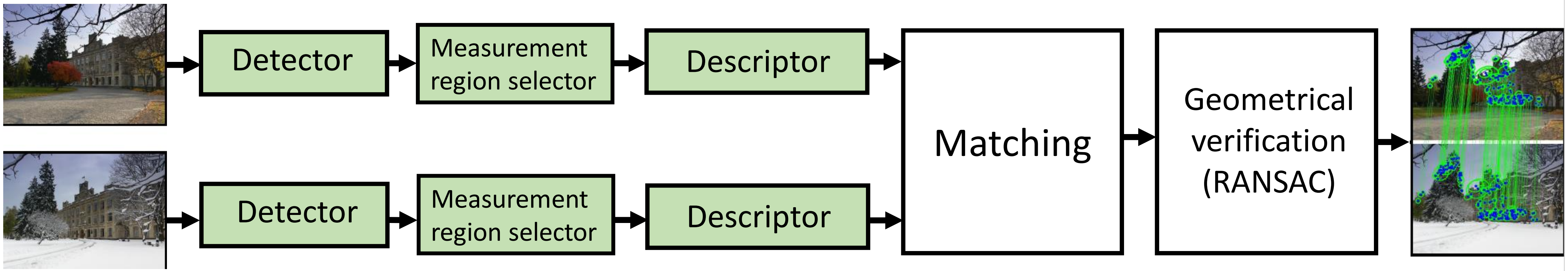
https://docs.opencv.org/3.4.3/da/df5/tutorial_py_sift_intro.html



DoG filter is a simple blob template



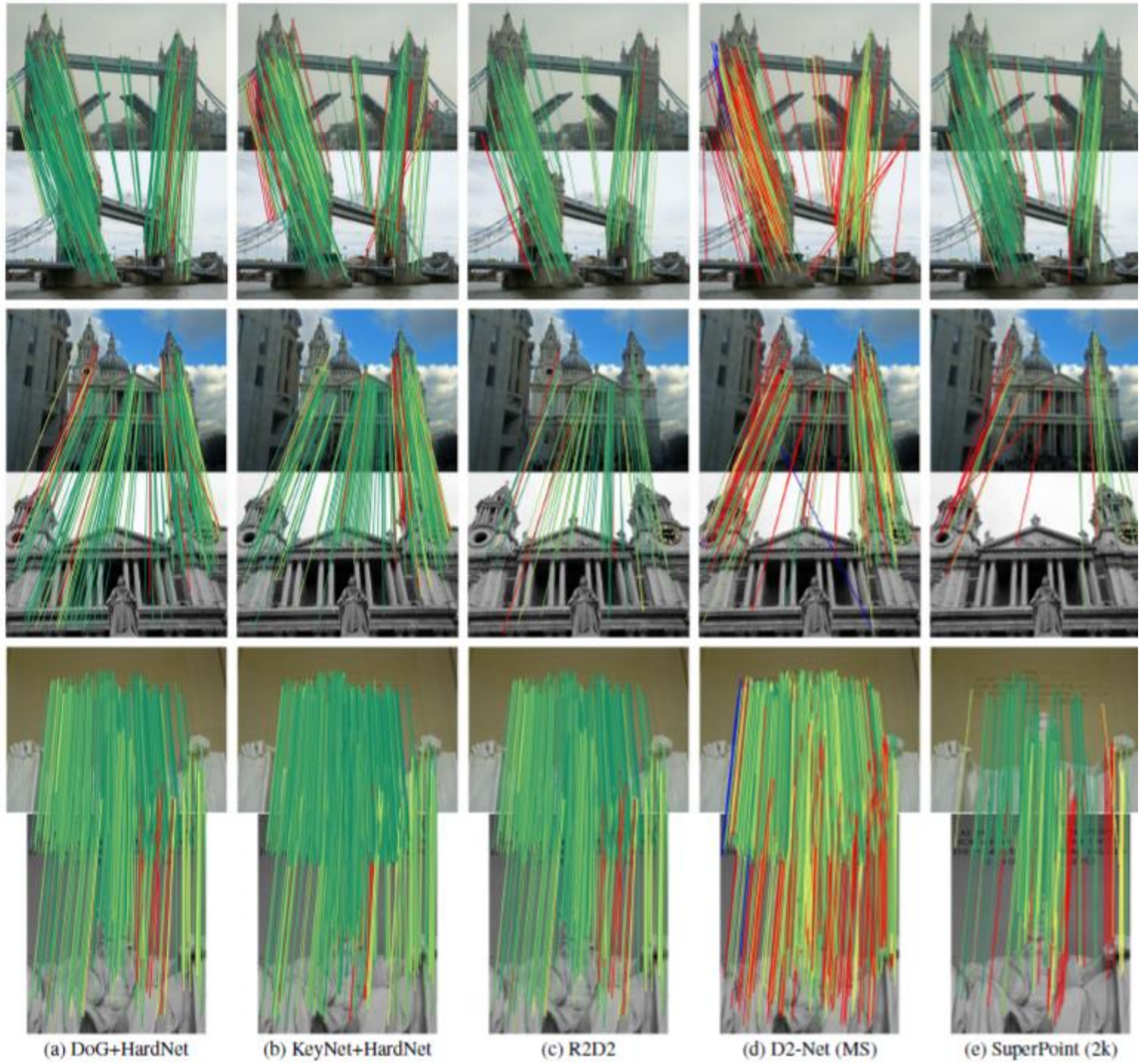
Detections on synthetic image

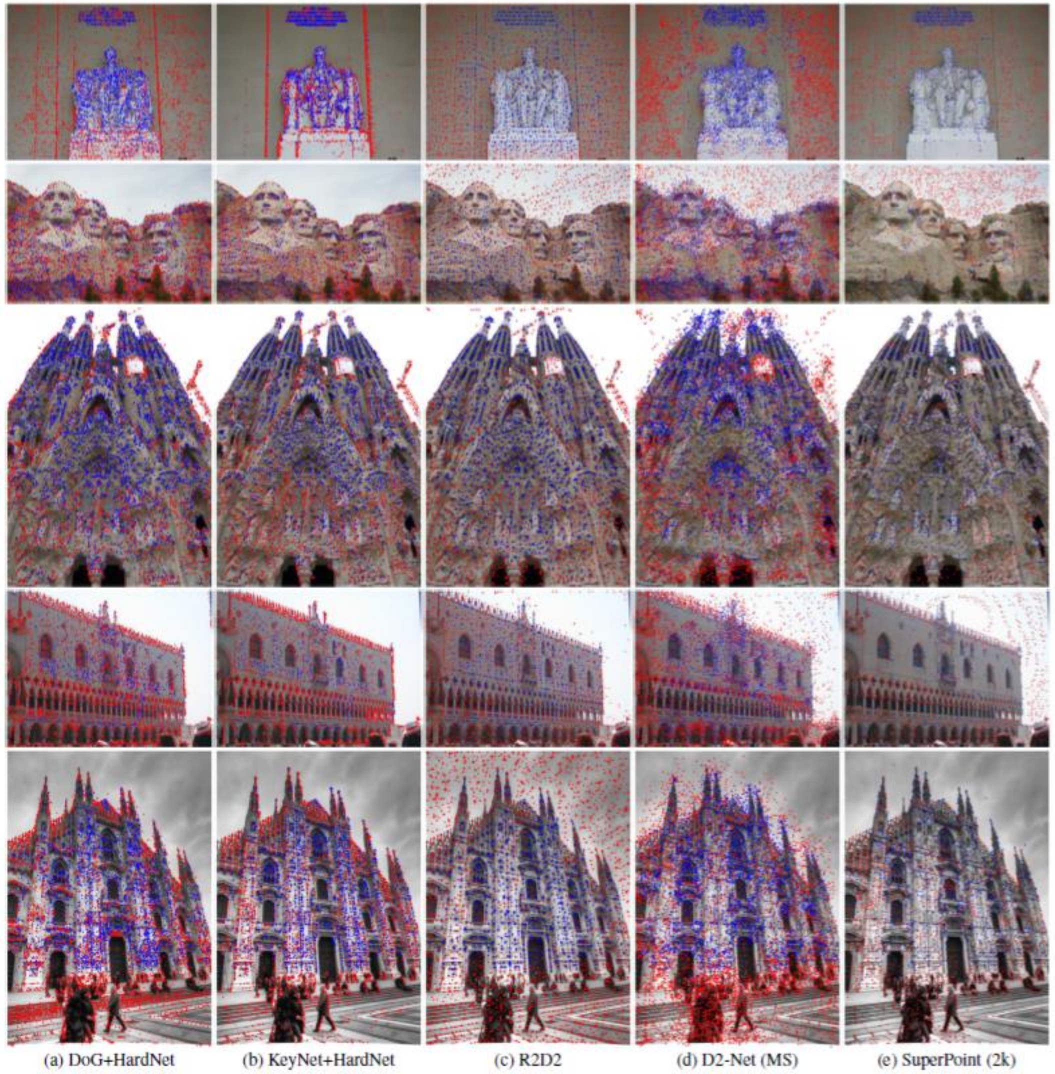


Joint detectors and descriptors

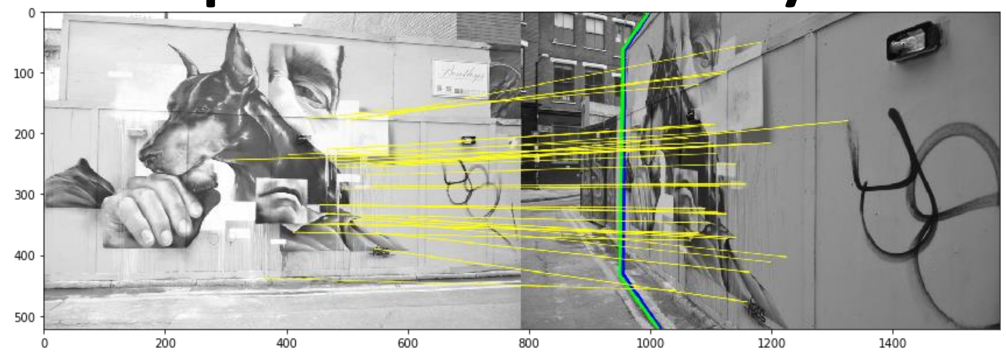
SuperPoint (CVPRW 2017)
DELF (ICCV 2017)
D2Net (CVPR 2019)

Qualitive comparison: learned features

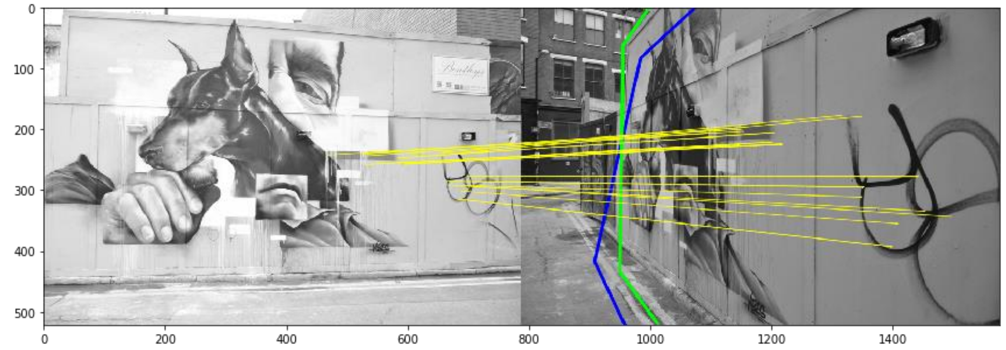




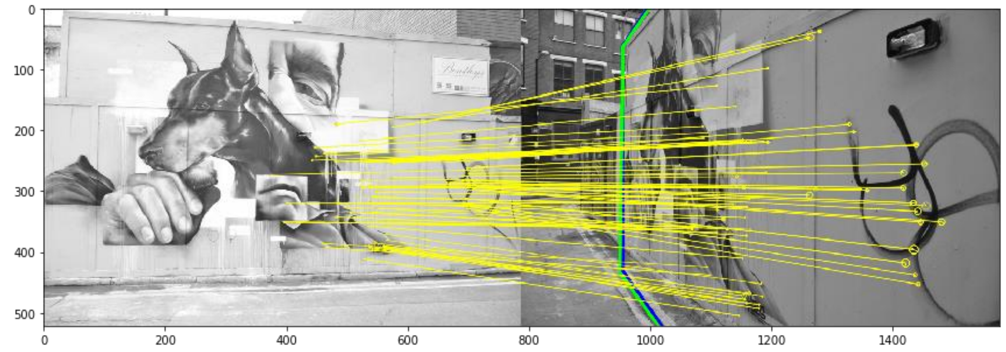
Comparison on toy example



SuperPoint: 51 inliers

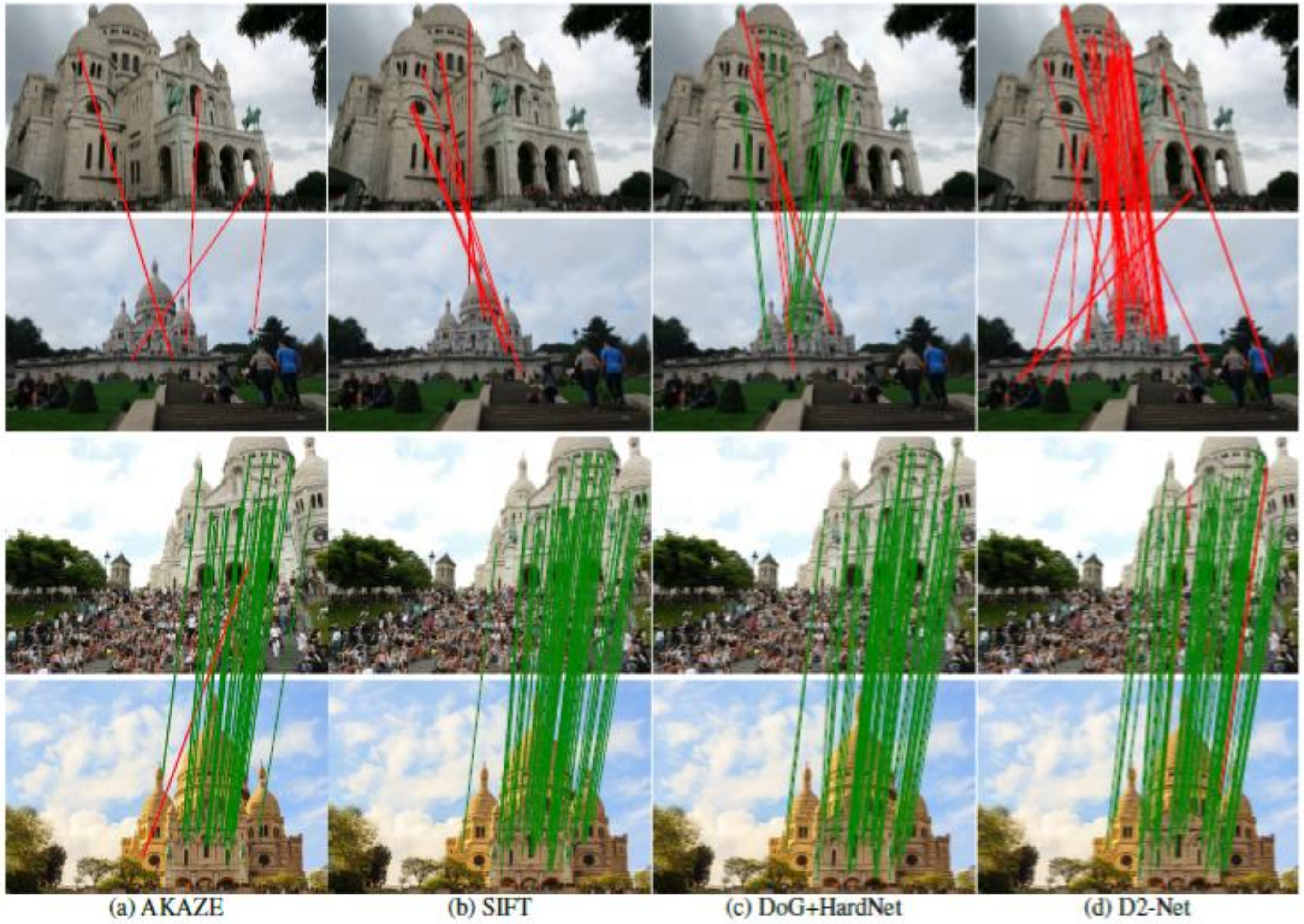


D2Net: 26 inliers, incorrect geometry

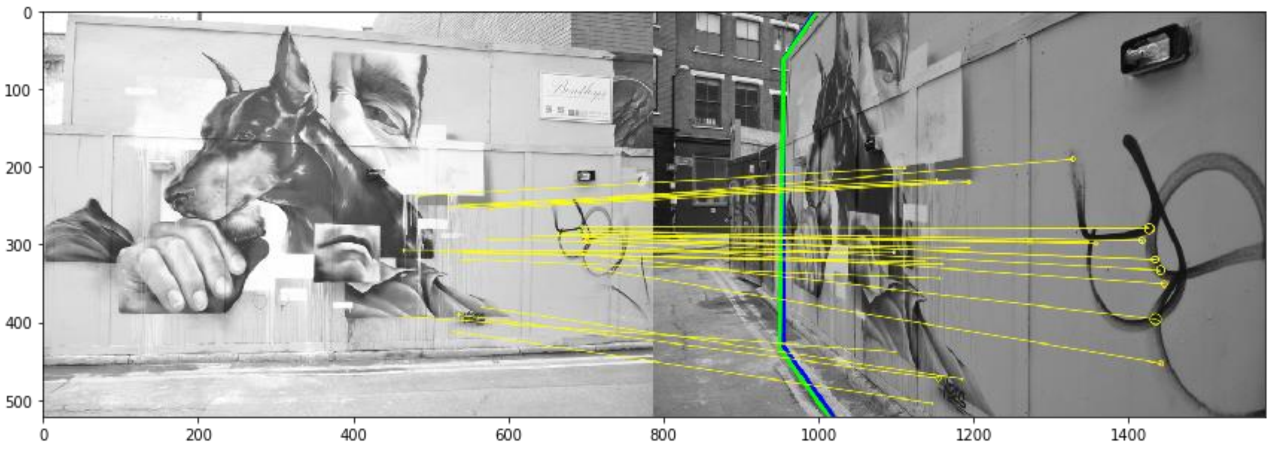


DoG + HardNet: 123 inliers

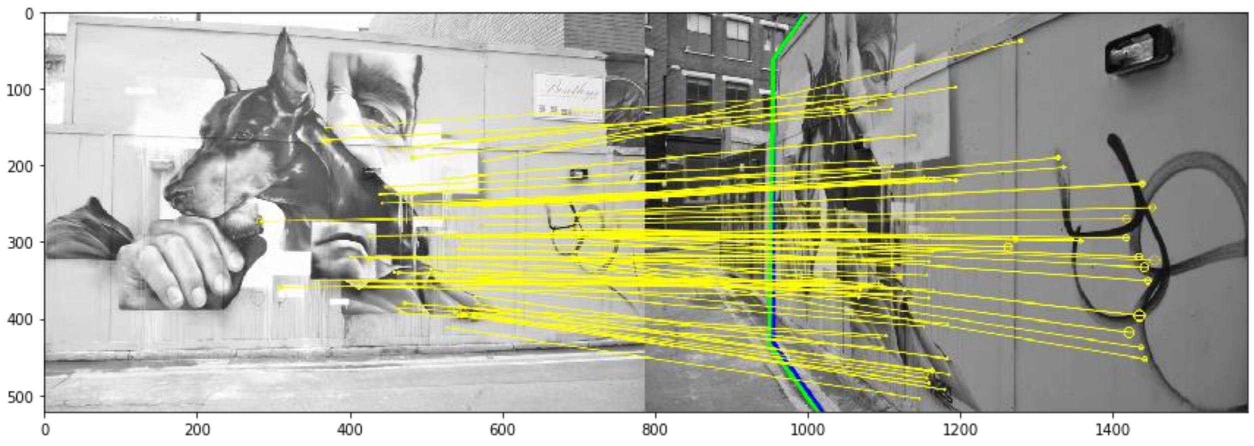
Some qualitive examples



All things together



SIFT + SNN match + OpenCV RANSAC:
27 inliers



SIFT + NoOri + HardNet + FGINN union match
+ CMP RANSAC:
179 inliers

I really need to match this

- View synthesis: MODS

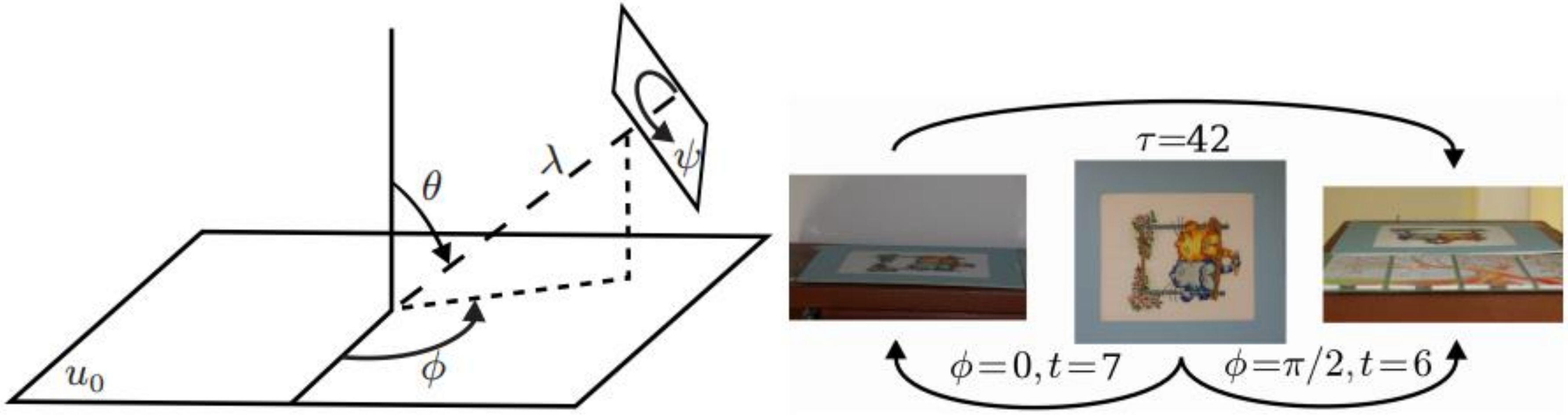
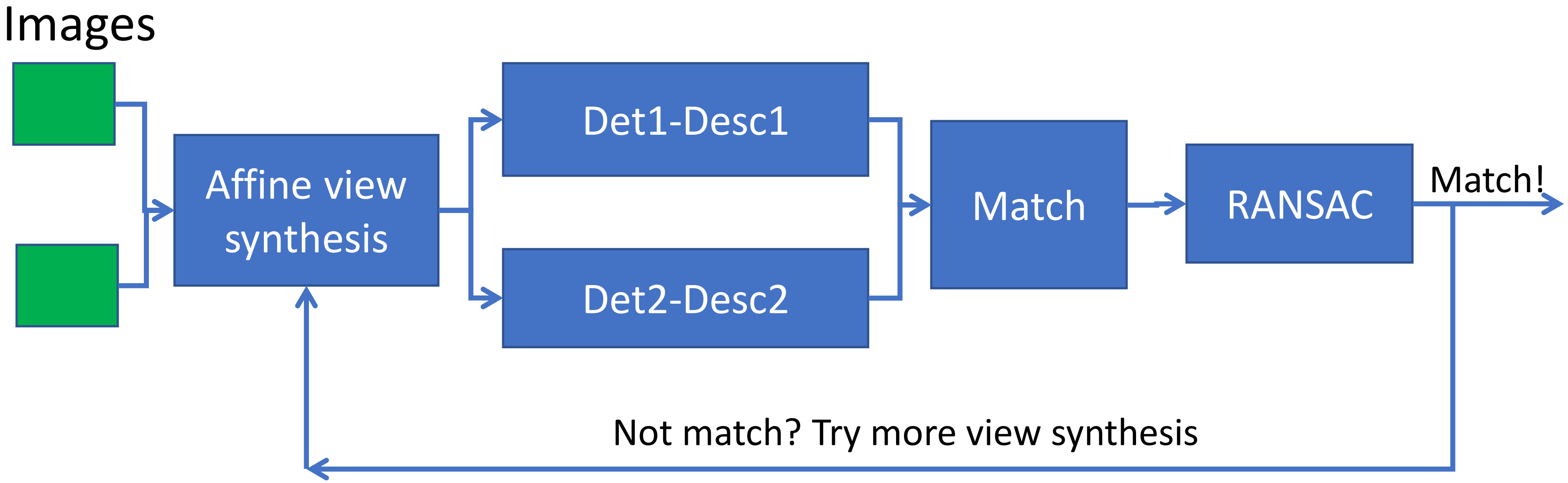


Figure 1: (left) the affine camera model (1). Latitude $\theta = \arccos 1/t$ – latitude, longitude ϕ, scale λ scale. (right) Transitional tilt τ for absolute tilt t and rotation ϕ.

D. Mishkin, J. Matas and M. Perdoch. **MODS: Fast and Robust Method for Two-View Matching**, CVIU 2015,

MODS (controller and preprocessor)



MODS handles angular viewpoint difference up to:

- 85° for planar scenes
- 30° for structured

D. Mishkin, J. Matas and M. Perdoch. **MODS: Fast and Robust Method for Two-View Matching**, CVIU 2015,

Thank you for your attention

- If you DO NOT need correspondences & camera pose → DO NOT use local features.
  Use global descriptor (ResNet101 GeM) + fast search (faiss)
- Step 0: try OpenCV (root)SIFT
- Use proper RANSAC
- Matching → use FGINN in two-way mode
- Need to be faster → ORB/SuperPoint.
- Custom data → train on your own dataset
- If images are upright, DO NOT DETECT the ORIENTATION
- Landmark data → DELF
- Try SuperGlue, it seems to be super cool

Dmytro Mishkin
ducha.aiki@gmail.com