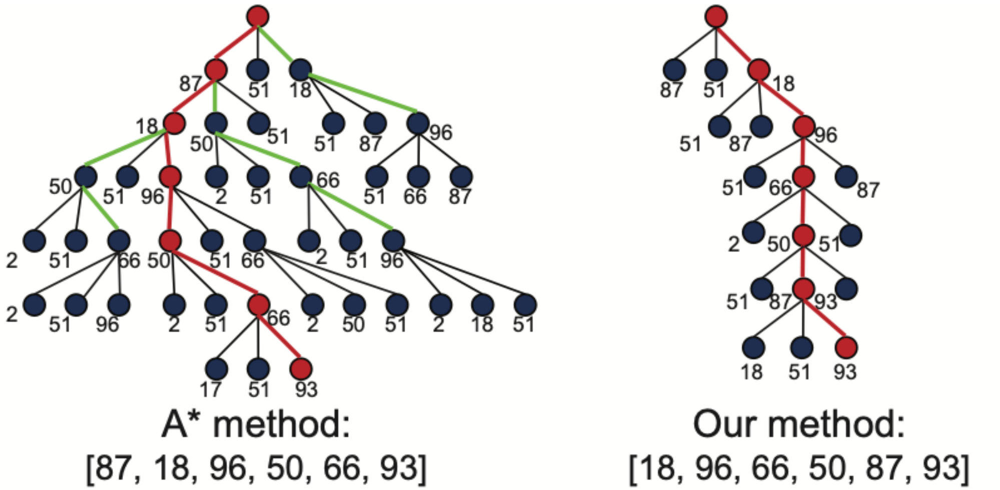
MaxCon via Reinforcement Learning

2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Extended version in IEEE PAMI
Giang Truong, Huu Le, Erchuan Zhang, David Suter, and Syed Zulqarnain Gilani. “Unsupervised Learning for Maximum Consensus Robust Fitting: A Reinforcement Learning Approach”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 45.3 (2023), pp. 3890–3903. doi: 10.1109/TPAMI.2022.3178442.

Unsupervised Learning for Robust Fitting: A Reinforcement Learning Approach

Giang Truong* Huu Le** David Suter* Erchuan Zhang* Syed Zulqarnain Gilani*
*School of Science, Edith Cowan University, Australia
**Department of Electrical Engineering, Chalmers University of Technology
{h.truong, d.suter, erchuan.zhang, s.gilani}@ecu.edu.au, huul@chalmers.se



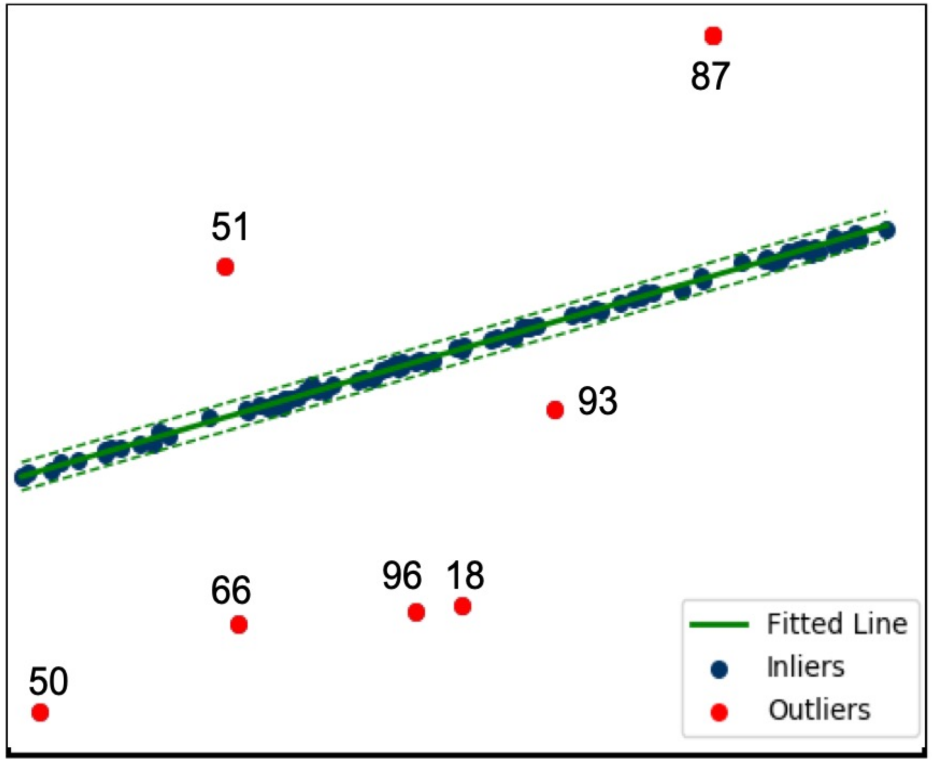


Figure 1: Illustration of the solutions found by our unsupervised learning method (right) and a globally optimal algorithm [4] (left). The number shows the specific index of points in the point set. The admissible heuristic in A* method brings the search into some fruitless subparts (green line) before discovering optimal solution (red line). Our agent learns to remove outliers by traversing from the initial state to the goal state in the minimal number of steps (the states are numbered based on the index of the removed point). Observe that both methods terminate at the same solution (i.e., both remove the same set of outliers).

Essence

- Find a **policy** that, given a start state (data – all of the data), finds a goal state (feasible subset) by removing one data item at a time, and minimizes the number of such steps.
- **(deep) Learning based approaches** essentially use (deep) learning networks to encode states, learning history

Reward function – the “hints” that an action was a good action. State encoding.

$$e(s_{\mathcal{S}}) = \begin{cases} 0 & \text{if } \ f(\mathcal{S}) \leq \epsilon \\ -1 - \lambda \dfrac{f(\mathcal{S})}{\max(f(\mathcal{S}))} & \text{otherwise.} \end{cases}$$

Generalizes CVPR paper version – that version has lambda=0
max(f(S)) is the maximum sized residual for the data set (i.e., the value of the max residual at the first fit
Note: trying to maximise reward – so minimize negative reward

RL methods use an exponentially decaying weighted sum of past rewards

$$\mathbf{S}_{\mathcal{S}} = [\mathbf{H} \ \ \mathbf{b}_{\mathcal{S}} \ \ \mathbf{v}_{\mathcal{S}}]$$

$$\mathbf{b}_{\mathcal{S}}[i] = \begin{cases} 1 \text{ if } \mathbf{x}_i \in \mathcal{B}_{\mathcal{S}} \\ -1 \text{ otherwise.} \end{cases} \qquad \mathbf{v}_{\mathcal{S}}[i] = \begin{cases} 1 \ \text{ if } \mathbf{x}_i \in \mathcal{V}_{\mathcal{S}} \\ -1 \ \text{ otherwise.} \end{cases} \tag{7}$$

H is simply the coordinates of data points, b and v use 1,-1 encoding of basis and elimination history – vertex cover!

Algorithm 1 Main algorithm.

Initialize an empty experience replay memory $\mathcal{M}$.
for episode e = 1 to L **do**
    Take a set of putative measurements $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$
    Obtain maximum residual $f(\mathcal{S})$ and basis $\mathcal{B}_{\mathcal{S}}$ by solving (4)

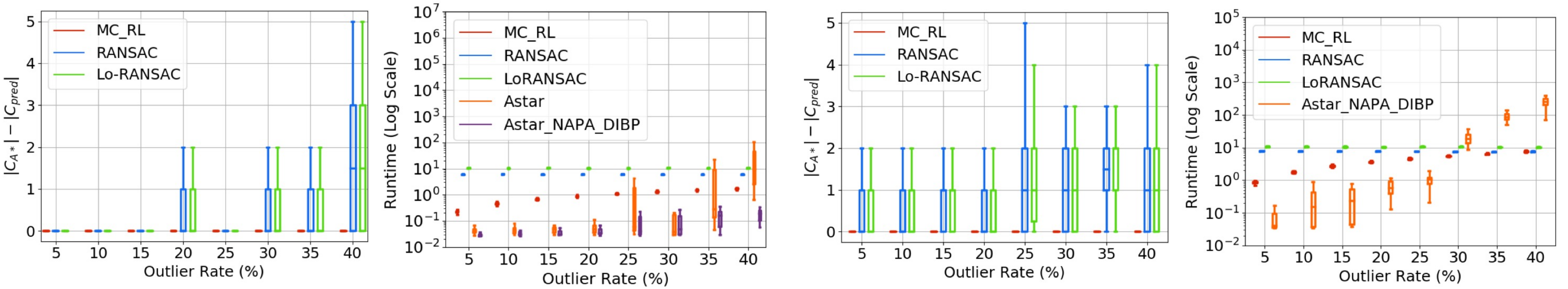
L_infinity fit

    Initialize first state $s^{(t=0)}$
    while $(f(\mathcal{S}) > \epsilon)$ **do**

$$a_t = \begin{cases} \text{random action } a \in \mathcal{A}(\mathcal{B}_{\mathcal{S}}), & \text{w.p.}\varepsilon \\ \underset{a \in \mathcal{A}(\mathcal{B}_{\mathcal{S}})}{\arg\max}\, \hat{Q}(s^{(t)}, a|\Theta), \text{otherwise} \end{cases}$$

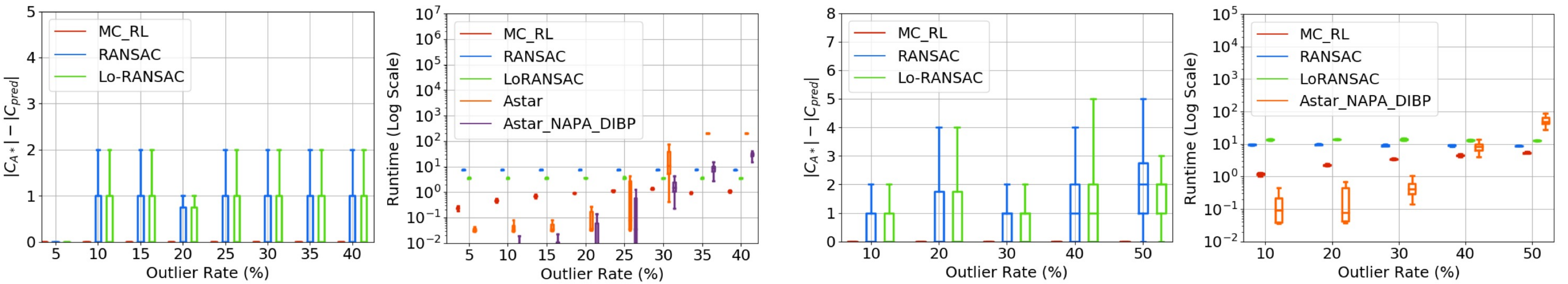
Mix “exploration” with “experience guidance” (Q-learning – expected reward/policy)

        Get reward $e(s^t))$ and move to next state s^{t+1}
        Add tuple $(s^t, a^t, s^{t+1}, e(s^t))$ to $\mathcal{M}$
        Sample random batch from $\mathcal{M}$
        Update network parameter Θ
    end while
end for



(a) $N = 100$ (b) $N = 200$

Fig. 5: 2D line fitting on (a) 100 and (b) 200 synthetic data points with various outlier rates. *Left*: Distribution of differences between predicted consensus and global optimal solution (obtained using A*) with baseline models. Note: lower is better. *Right*: Run-time of our method compared to baseline models. Our method is very competitive for high number of points with high rate of outliers in terms of achieving optimal solution and less runtime. Note that A* is dropped out in the comparison of runtime for $N = 200$ since it is very slow in general for high number of outliers.

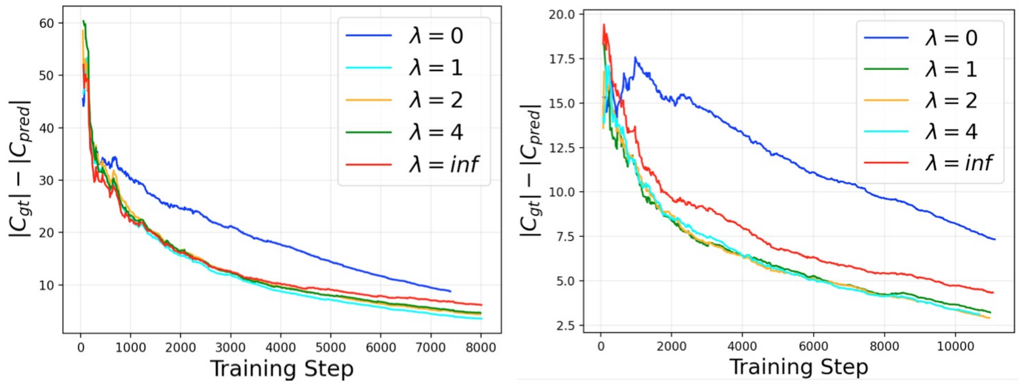


(a) $N = 100$ (b) $N = 200$

Fig. 6: 3D plane fitting on (a) 100 and (b) 200 points with varying outlier rates. *Left*: Difference between predicted consensus and global optimal solution (using A*) with baseline models (lower is better). *Right*: Run-time of our method compared to baseline models.

TABLE 1: Linearized Fundamental Matrix Estimation on (semi-synthetic) ModelNet40 dataset. The average consensus size ($|C|$: in percentage) and the processing time (t: in seconds) are reported for each method on various outlier rates. The best values of each metric are highlighted.

	MC_RL2		MC_RL		LO-RANSAC		RANSAC		SLCM		ULCM		GC-RANSAC		MAGSAC++		OANet		A*_NAPA_DIBP	
Outlier	$\lvert C\rvert$	t	$\lvert C\rvert$	t	$\lvert C\rvert$	t	$\lvert C\rvert$	t	$\lvert C\rvert$	t	$\lvert C\rvert$	t	$\lvert C\rvert$	t	$\lvert C\rvert$	t	$\lvert C\rvert$	t	$\lvert C\rvert$	t
10%	92.4	0.37	92.2	0.36	91.5	2.61	91.4	2.51	90.4	0.04	90.2	0.04	91.4	**0.01**	91.5	0.07	90.6	0.02	**92.9**	74.9
20%	**83.8**	0.71	83.5	0.71	82.6	2.43	82.5	2.35	81.9	0.04	80.2	0.04	82.5	**0.01**	82.6	0.03	82.3	0.02	N/A	≥ 1000
30%	**74.2**	1.12	74.0	1.11	72.9	2.47	72.8	2.31	72.5	0.04	70.4	0.04	72.7	**0.01**	72.9	0.01	72.8	0.02	N/A	≥ 1000
40%	**64.1**	1.54	62.6	1.49	62.0	2.39	61.8	2.32	61.1	0.04	60.1	0.04	61.7	**0.01**	61.9	0.02	61.2	0.02	N/A	≥ 1000
50%	**54.9**	1.91	53.6	1.89	52.9	3.41	52.6	3.35	51.5	0.04	50.3	0.04	52.4	**0.01**	52.8	0.01	51.5	0.02	N/A	≥ 1000



ModelNet40 dataset   KITTI dataset

Fig. 7: Comparison between different choices of reward functions

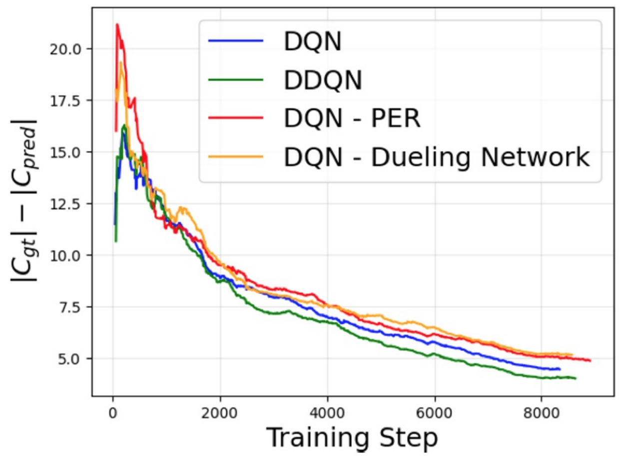


Fig. 8: Comparison of the performance of some variants of Deep Q

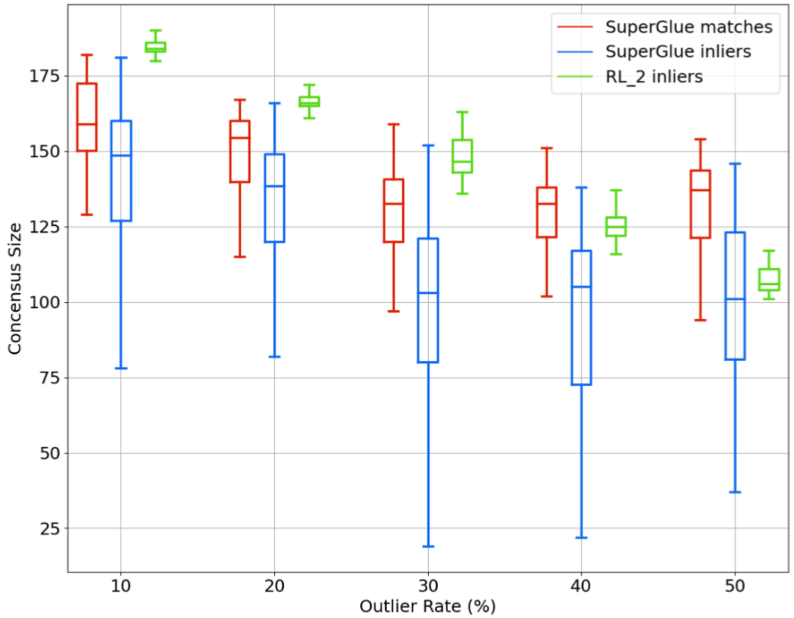
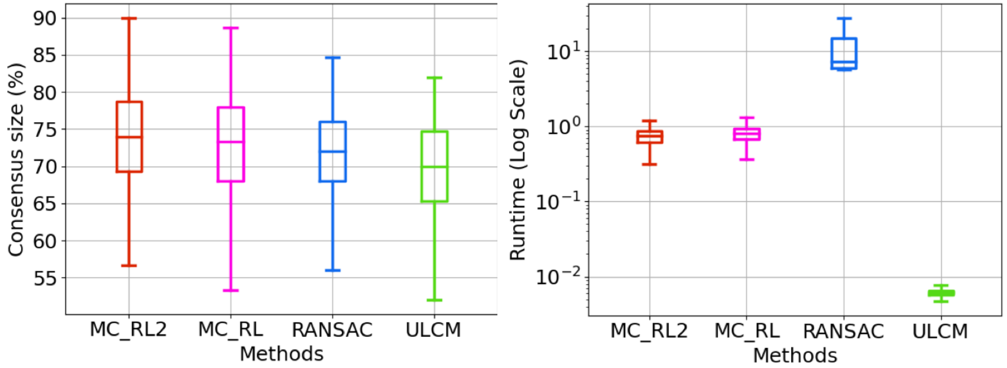
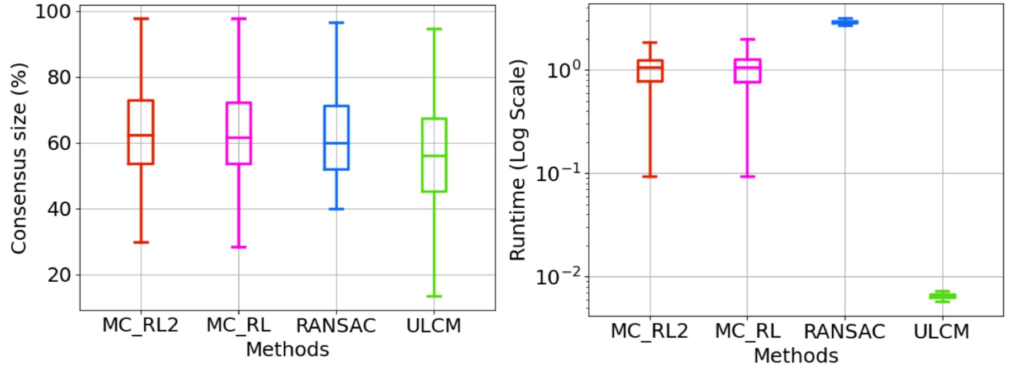


Fig. 9: Comparison of the performance of SuperGlue and our methods on ModelNet40 dataset. The distribution of consensus size per outlier rate is reported. “SuperGlue matches” are the counts before pruning technical outliers from those returned by SuperGlue. “SuperGlue inliers” are the counts after pruning. On average, our method clearly outperforms Superglue on outlier rates up to 40%. Our method never declares outliers to be inliers



(a) KITTI Dataset



(b) Roman Forum Dataset

Fig. 10: Linearized Fundamental Matrix Estimation on real datasets. *Left*: Consensus size (%). *Right*: Run times in log scale. Similar to the experiment on the ModelNet40 dataset, our method, on average, finds $2\% - 5\%$ higher consensus with affordable time (~ 1 second).

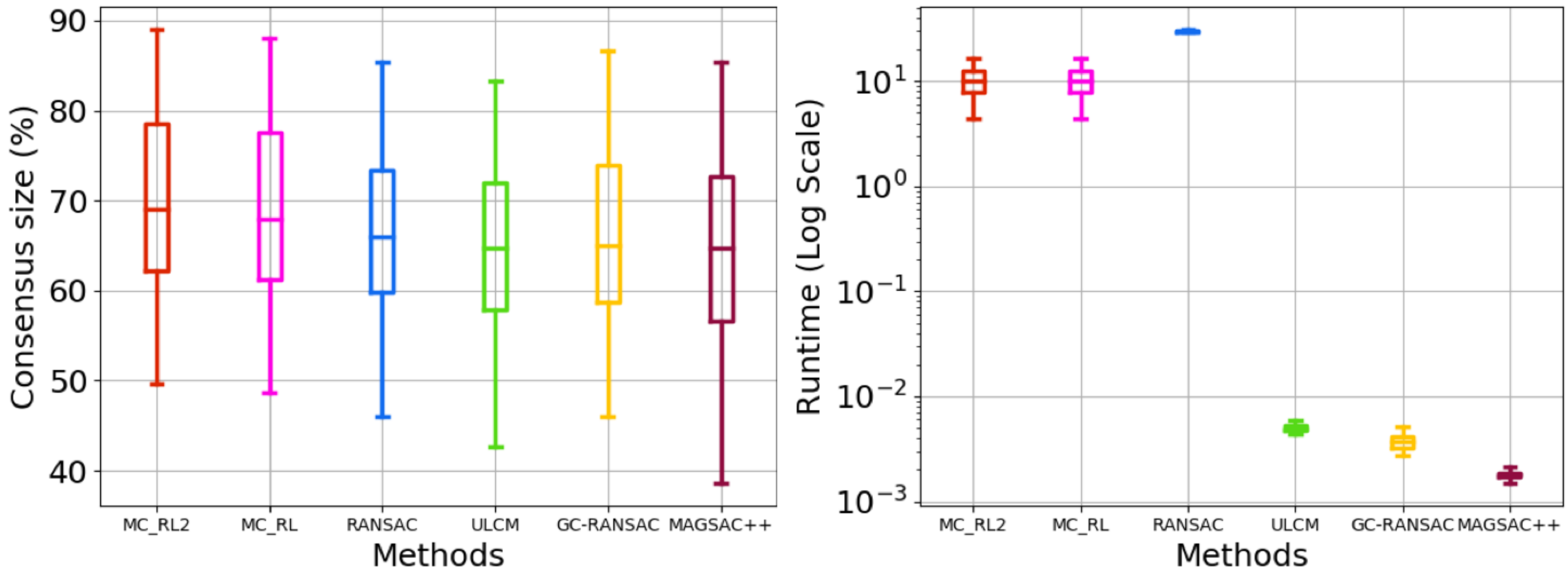


Fig. 11: Homography ftting on the KITTI dataset. *Left*: Consensus size (%) comparison. *Right*: Run time comparison in log scale.

TABLE 3: Comparison between different choices of reward functions during evaluation before/after local tree refinement. The consensus size (in percentage) is reported. The best values are highlighted. Note: the outlier ratios are notional target ratios, because we don't check whether a generated outlier actually is an outlier (it might fall inside the tolerance), the actual outlier ratio is typically smaller than the target which is why the inliers can be higher than 100%-outlier%.

Outlier Rate	$\lambda = \mathbf{0}$	$\lambda = \mathbf{1}$	$\lambda = \mathbf{2}$	$\lambda = \mathbf{4}$	$\lambda = \infty$
	ModelNet40				
10%	91.64/92.39	**91.76/92.47**	91.25/91.87	91.24/91.93	91.67/92.40
20%	82.56/83.59	**83.27/83.89**	82.92/83.81	82.90/83.76	82.77/83.76
30%	72.50/73.37	**73.25/74.25**	73.20/74.13	72.91/73.82	72.90/74.22
40%	61.09/62.24	**61.69/62.59**	61.44/62.49	61.00/62.24	60.37/61.54
	KITTI				
Unknown	69.03/71.50	**71.75/73.59**	71.47/73.29	71.55/73.35	71.24/73.06