



## OPEN Large scale analysis of dataset and simulation biases in SLAM research

Muhammad Latif Anjum<sup>1</sup>, Wajahat Hussain<sup>1</sup>✉, Usama Mudassar<sup>1</sup>, Syed Ali Haider Bukhari<sup>1</sup>, Abrar Anwar Qureshi<sup>1</sup> & Irfan Hussain<sup>2</sup>✉

The performance of visual SLAM and localization methods is generally reported on famous datasets. These datasets are generally captured under human supervision and hence, are prone to human biases. In this work we expose two such biases (capture bias and negative world bias) in well-known SLAM datasets. Photo-realistic simulators provide a platform for gathering data without human supervision and hence human bias. However, not every simulator is suited for benchmarking visual localization methods. This is due to the difficulty of calibrating the first-view camera of these simulators. The calibration parameters (both intrinsic and extrinsic), while routinely provided with the datasets, are not generally available for simulators and virtual worlds. We propose a novel and user-friendly method to calibrate these simulators which is an essential requirement for using them for visual navigation. We demonstrate our method on a well-known simulator (MINOS), as well as a highly popular open world game (GTA-V). Finally, we also analyze the simulation-to-reality gap of these virtual platforms and propose a method to reduce this gap. We show that the performance of visual navigation algorithms (e.g., simultaneous localization and mapping: SLAM) significantly degrades when tested on novel situations available in virtual worlds.

**Keywords** Camera calibration, Simulators, Virtual worlds, Visual navigation

Try naming the objects shown in the first row of Fig. 1. We will wait. Now try the same with the remaining images. This time the task seems relatively easy. There seems to be an obvious difference between the first row and the remaining content. All the images have been captured using a moving camera operated by the human. Content shown in the first row comes from the wearable camera where the operator is not previewing the captured view<sup>1</sup>. The remaining rows show well-known localization datasets<sup>2–8</sup>. The clarity of images in these datasets invites the question of whether the performance of localization (or SLAM) algorithms is benefiting from the human intelligence involved in capturing these datasets? Is there a bias introduced by the operator who intended to benchmark SLAM on these datasets?

Visual SLAM datasets are mainly captured by a collaboration of a few research groups. In the extreme case, a single research group or a single researcher ends up capturing the whole dataset. This is not intentional. While anyone having a camera (mobile phone) can potentially become a data generator for visual tasks, visual SLAM datasets involving specialized sensors (lidar, IMU, stereo camera) can't be crowd sourced. Kinect is one sensor that has gained popularity due to gaming. There have been attempts to gather 3D data in crowd sourced fashion using Kinect<sup>9</sup>. Recently, depth sensors have been introduced in mobile phones. Hopefully, this will help in adding diversity in future datasets.

Interestingly, the visual SLAM community seems to have an understanding while collecting datasets. Datasets gathered in different countries over the years apparently have striking similarities. Datasets are captured early morning or at times having little traffic. Roads and halls appear deserted. Sensors are driven smoothly and a safe distance is maintained from all obstacles. These good habits help avoid dynamic content (moving cars, pedestrians), occlusion, blur and generate feature rich content from canonical views. Surprisingly all these good habits result in ideal conditions for visual odometry. Data gathered from wearable cameras<sup>1</sup> represent more commonly occurring scenarios including dynamic content, closeup interactions with objects resulting in occlusion or low feature content, self occlusion and sudden motions resulting in blur.

How to reduce human bias from these datasets? This reduction will help in assessing in-the-field performance of future SLAM solutions. TartanAir<sup>8</sup> and other SLAM datasets are still captured using human intelligence. Even the crowd sourced dataset gathered from the internet, for large scale 3D reconstruction of ROME iconic spots, is captured by humans<sup>10</sup>. Lately it has been shown that robots can themselves supervise data gathering process<sup>11,12</sup>

<sup>1</sup>Robotics and Machine Intelligence (ROMI) Lab, School of Electrical Engineering and Computer Science (SECS), National University of Sciences and Technology (NUST), Islamabad, Pakistan. <sup>2</sup>Center for Autonomous Robotic Systems, Khalifa University, Abu Dhabi, UAE. ✉email: wajahat.hussain@seecs.edu.pk; irfan.hussain@ku.ac.ae



**Fig. 1.** Human bias in visual SLAM datasets. The video (first row) was captured using a wearable camera while human was performing daily activities in carefree style<sup>1</sup>. The remaining rows show images coming from standard visual SLAM datasets which were carefully captured by the operators. It is hard to recognize the objects in the first row whereas, the content is clear (in the remaining rows), captured from far field with feature rich content. Are these carefully gathered datasets realistic?.

and actually learn new skills, e.g., novel way of grasping the cup, which eluded them due to human bias in labeled datasets. This finding motivates the need of active learning platforms beyond datasets. *Perhaps leveraging virtual platforms in an automatic manner may reduce the human bias.*

Autonomous driving has fueled the development of hi-fidelity simulators for urban road scenes, e.g., CARLA<sup>13</sup> and AirSim<sup>14</sup>. These simulators provide a large variety of environments which have helped the mobile agents to improve their navigational skills. These simulators have enabled mushroom growth in the development of visual SLAM/odometry datasets such as visual-inertial<sup>15</sup>, changing conditions<sup>16,17</sup>, and omnidirectional camera<sup>18,19</sup>.

In addition to autonomous driving simulators, there is large family of indoor simulators<sup>20–23</sup>. These simulators provide considerably larger number of scenes, as compared to autonomous driving counterparts, and allow different activities (interaction with objects) which are necessary for service robots. Indoor scenarios are arguably more challenging for visual tracking and mapping activities due to texture-less surfaces<sup>24</sup>, heavy occlusion<sup>25</sup> and constrained scenarios (narrow corridors)<sup>26,27</sup>.

As compared to above mentioned simulators, open world games, e.g., Grand Theft Auto (GTA-V), offer large number of features not available in research grade simulators. An agent can move between indoor, outdoor, underwater and airborne scenarios seamlessly. However, the agents trained in these recent realistic simulators under perform in the real world<sup>28</sup>. Therefore, there is a need to establish the gap between the simulator versus the real world performance (Sim2Real gap). Finally, we expose the simulation-to-reality gap of the virtual platforms w.r.t. visual tracking. We also propose a simple strategy to reduce this gap.

Finally it is not trivial to leverage these open world platforms for visual localization benchmarking. As these are proprietary platforms there is limited transparency. In this work, we provide a simple yet effective strategy for intrinsic and extrinsic calibration of diverse simulators specially open world platforms. This calibration will enable benchmarking visual localization methods without involving human bias/supervision.

Key contributions of this work are:

1. Our analysis reveals presence of two biases, i.e., capture bias and negative world bias, in visual SLAM datasets, and their impact on SLAM performance.
2. We demonstrate that simulators inherently exhibit a Sim2Real gap. Therefore, we develop a motion-blur addition strategy based on consecutive frame averaging, effectively reducing this gap without the artifacts introduced by deblurring methods.
3. While simulators can overcome these human curator induced biases, intrinsic/extrinsic calibration challenges limits their use in the SLAM evaluation. We propose an efficient simulator's intrinsic/extrinsic calibration method that avoids reverse engineering, suited for proprietary simulators.

## Related work

### Capture bias

Capture bias is one of the fundamental limitations of vision datasets<sup>31</sup>. Capture bias represents a particular habit while capturing an image, e.g., the desired object is always in the middle of the image. Given this capture bias, vision (detection) algorithms can achieve high accuracy by placing the detection box in the center of the image. This limits the performance of such algorithms on challenging datasets where objects can appear at any image

location. Such bias can be reduced by randomly placing the camera w.r.t to the object of interest, an option that is readily available in many simulators and video games.

Discovering unavoidable capture biases is sometimes beneficial, e.g., images are generally captured at the human height, i.e., 4.5 ft<sup>43,44</sup>. This bias was brilliantly utilized in the pioneering work on single view geometry<sup>43</sup>. Single image was converted into a metric 3D model using this capture bias<sup>44</sup>. This led to many novel applications including inserting new objects in single images<sup>45</sup>, converting single images into action space<sup>46</sup> and using acoustic echoes to improve single image 3D reconstruction<sup>47</sup>. Such beneficial biases are rare.

In this work, we report one particular example of capture bias in visual SLAM datasets (Fig. 2). The ability to recognize a location being revisited is known as loop closure<sup>48</sup>. This feature helps to reduce drift captured over longer trajectories. We investigated such loop closures for various datasets. Revisits usually take place with minimal viewpoint change. Visual inspection of the trajectory (first column in Fig. 2) indicates that the revisited track almost overlaps with the initial visits. Furthermore, it is hard to spot any viewpoint change in images associated with revisits (Fig. 2). This makes it easier for the localization algorithm to close loops and remove its drift. Furthermore, this capture bias has strengthened its roots over the years. Oxford Robot Car<sup>30</sup> and, more recently, CARLA scenes<sup>17</sup> datasets include *infinite loop* where the same scene is revisited a large number of times with little view variation (sampled revisits from Oxford Robot Car dataset are shown in Fig. 2).

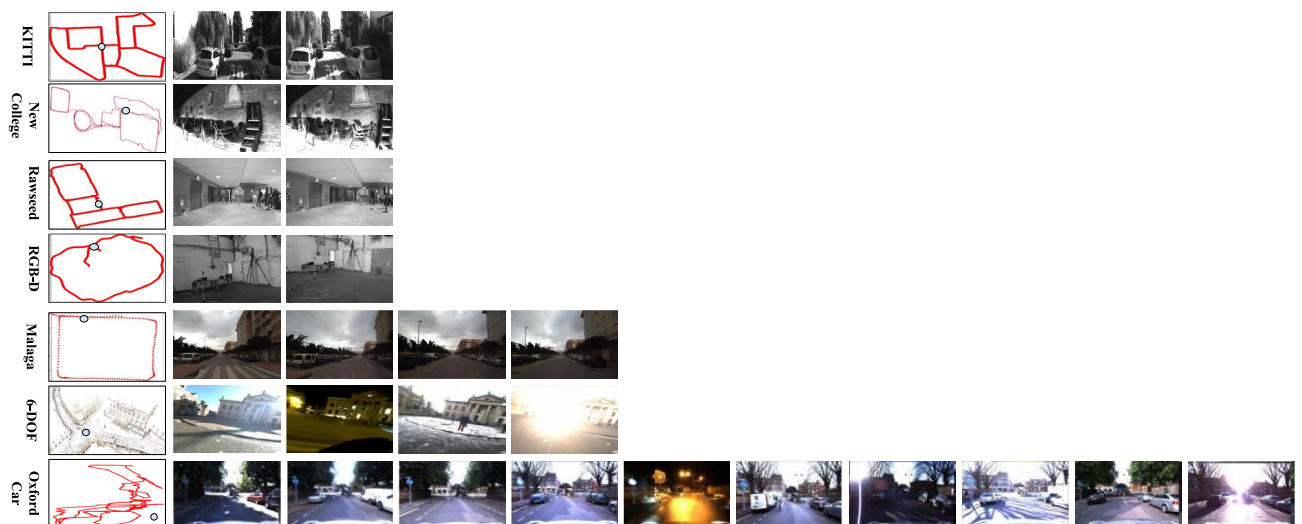
In this work we demonstrate that the performance of state-of-the-art visual SLAM methods considerably degrades when camera moves autonomously within these simulators.

### Negative world bias

Learning from failures is a fundamental ability to improve performance. Performing in completely unfamiliar settings increases the chances of finding failures. These unfamiliar settings are called the negative world<sup>31</sup>. As mentioned before, if a human detector is only judged by its performance on images containing humans, it might under perform in reality when faced with images not containing humans but other vertical content like poles, trees, plants etc.<sup>49</sup>. Given a large amount of negative world data, one can figure out potentially multiple failure cases<sup>50</sup>. This ability to accept and discover weakness has enabled vision algorithms to perform apparently unmanageable tasks e.g., finding needle in a haystack<sup>51</sup>.

Generally, visual SLAM datasets have limited hard/failure cases. In other words, visual SLAM datasets have deficient negative world perspective. In our opinion, lack of failures emanates from being extra careful while capturing these datasets. Sequences which prove hard to initialize or have frequent tracking failures are unfortunately not made part of the dataset. In our opinion, the demo sequence released with the SLAM solutions is generally not the first video captured. Interestingly, this reluctance to include failure cases (initialization failure, tracking failure) is not limited to datasets. We reviewed well-known visual SLAM works (Table 1) from the last two decades, which dates back to era well before the standard datasets, and could spot negligible failure cases.

It appears that with passage of time, reluctance to report failure cases has increased. PTAM<sup>34</sup> stands out w.r.t self criticism. An entire section is dedicated to discussion of failure cases. Authors clearly state that their system is “not yet good enough for any untrained user to simply pick up and use in an arbitrary environment”. DTAM<sup>36</sup> also indicates underlying assumptions for successful operation. Lately, failure cases of other SLAM systems are indicated with little self criticism. These apparent failures are indicated by the community<sup>25</sup>. Last column in Table 1 indicates negligible own mistakes reported in the paper. In our opinion, if the localization/SLAM systems



**Fig. 2.** Capture bias in visual SLAM datasets. Our investigation of loop closing instances in visual SLAM datasets indicates minimal viewpoints changes, which makes loop closure detection relatively straightforward. Compare the view of the first visit with the revisiting views. Trajectory (first column) indicates that the revisited track almost overlaps with the initial visits. There have been some attempts to capture views in changing conditions (times of the day (6 DoF<sup>29</sup>), seasons (Oxford Car<sup>30</sup>)).

SLAM	Failure cases	Failure instances
RatSLAM <sup>32</sup> , 2004		
MonoSLAM <sup>33</sup> , 2007		
PTAM <sup>34</sup> , 2009	✓✓✓	
1 pt RANSAC VO <sup>35</sup> , 2010		
DTAM <sup>36</sup> , 2011	✓✓	
LSDSLAM <sup>37</sup> , 2014		
SVO <sup>38</sup> , 2014		
ORB-SLAM <sup>39</sup> , 2015	✓	1
VINS-Mono <sup>40</sup> , 2018		
ORB-SLAM3 <sup>41</sup> , 2021	✓	2
MASt3R-SLAM <sup>42</sup> , 2025	✓*	

**Table 1.** Failure is a word unknown to me. Limited number of failure cases (initialization failure, tracking failure) are reported with SLAM pipelines. Second column indicates the acknowledgment of failure cases. Last column indicates the number of failure instances reported. ✓\*: Recent MASt3R-SLAM does not include any failure cases; they have however discussed limitations of their approach.

are evaluated in the wild or inside a simulator where the agent can freely move, there would be more failure cases, and the algorithms can build on those cases<sup>27,50</sup> for more robust performance.

## Methods

In this section, we describe two main contributions. Firstly, how to reduce the simulation to reality gap (Sim2Real gap) w.r.t. visual tracking. Secondly, we provide user friendly method to calibrate virtual platforms for visual tracking.

### Reducing simulation to reality gap

If the agent is provided with an environment to keep trying, such agents have shown performance to outclass humans on multiple tasks (playing games<sup>53</sup>, grasping objects<sup>11</sup>). In the seminal work<sup>53</sup>, Mnih et al. demonstrated that agents managed to play Atari games better than humans just by observing the raw images and the game score. Such success cases have accelerated the development of hi-fidelity simulators<sup>54</sup> to leverage the true potential of deep learning. There have also been attempts to use the renowned realistic gaming platforms<sup>55</sup> to extract free data. Even for mobile agents, multiple realistic simulators<sup>20,23,56</sup> have been proposed recently.

However, the agents trained in these recent realistic simulators under perform in the real world<sup>28</sup>. Therefore, there is a need to establish the gap between the simulator versus the real world performance (Sim2Real gap). This gap will guide the progress of improving the simulator as well as upper bound our expectations. Recently,<sup>28</sup> established this gap, related to path planning for navigating agents in the Habitat environment<sup>56</sup>. In this work, we quantize this gap between carefully gathered visual SLAM datasets, realistic appearing simulators, photo-realistic virtual environments, and ego vision datasets consisting of real carefree motion.

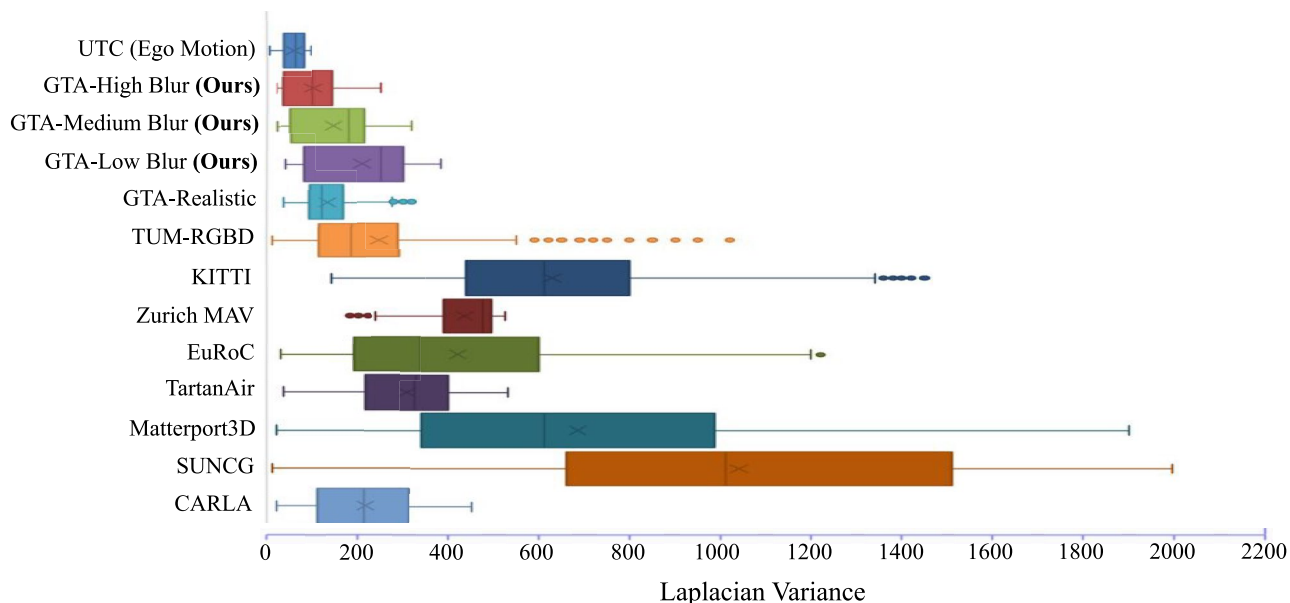
We compared the amount of motion blur that appears in these datasets (Fig. 3) in order to establish Sim2Real gap from visual tracking perspective. Motion blur is one of the key factors that reduces the amount of keypoints detection and causes frequent tracking failures<sup>36</sup>. In order to estimate the amount of motion blur in a given image, we estimated the variance of response of the Laplacian filter<sup>57</sup> when applied to a given image (Eq. 1).

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (1)$$

Laplacian filter is commonly used to detect edges. High variance in its response to a given image indicates edges with different strengths. Low variance indicates lack of distinct edges indicating blurriness in the image. For each dataset, we estimated the variance of the Laplacian kernel for a large number of images.

In order to set the baseline for naturally occurring motion blur, we selected the ego motion dataset<sup>1</sup>, where the human performs normal daily life activities, while wearing a camera (first row in Fig. 1). Video is recorded at standard 30 FPS. The actor is not viewing the camera feed while performing activities. Therefore, viewer bias may be limited in this dataset. Comparing the motion blur present in this carefree motion with careful visual SLAM datasets and photo-realistic environments reveals obvious gap (Fig. 3). Large portion of visual SLAM datasets resides in negligible blur region. We also carried out analysis of variance (ANOVA) test on motion blur in each dataset, followed by Tukey's HSD (Honest Significant Difference) test (Table 2). Based on the adjusted p-value ( $p \geq 0.05$ ), only GTA High Blur (Ours) is statistically similar to UTC (Ego Motion). All other datasets, when compared with UTC (Ego Motion) have  $p = 0$ , indicating they are statistically different. When ordered based on difference of their mean blur (meandiff in Table 2), our GTA-based datasets (with various blur levels) are statistically closest to UTC (Ego Motion) dataset.

Surprisingly, considering the instability of drones, especially in slight wind, the dataset gathered with them (Zurich MAV<sup>7</sup>) has little blur and very sharp images, even in uncontrolled outdoor scenarios (Fig. 3).



**Fig. 3.** Dataset to reality gap: box and whisker plots of laplacian variance in carefree ego motion (UTC), and well-known datasets. High variance of Laplacian operator indicates low blur. Carefully gathered visual SLAM datasets and realistic appearing simulators have little blur as compared to ego vision dataset consisting of real carefree motion. Gap is smallest in our approach and UTC dataset<sup>1</sup> which consists of natural carefree motion. Interestingly, photo-realistic GTA-V<sup>52</sup> does not guarantee blur realism.

Group1	Group2	Meandiff	p-adj	Lower	Upper	Reject
UTC (Ego Motion)	GTA High Blur (Ours)	44.269	0.216	- 96.976	8.437	FALSE
UTC (Ego Motion)	GTA (Realistic)	74.341	0	- 118.275	- 30.407	TRUE
UTC (Ego Motion)	GTA Medium Blur (Ours)	88.675	0	- 135.217	- 42.132	TRUE
UTC (Ego Motion)	GTA Low Blur (Ours)	152.355	0	- 199.159	- 105.552	TRUE
UTC (Ego Motion)	CARLA	156.544	0	- 213.457	- 99.631	TRUE
UTC (Ego Motion)	TUM (RGBD)	188.851	0	- 275.896	- 101.806	TRUE
UTC (Ego Motion)	TartanAir	250.320	0	- 292.793	- 207.846	TRUE
UTC (Ego Motion)	EuRoC	362.744	0	- 438.931	- 286.557	TRUE
UTC (Ego Motion)	Zurich MAV	378.207	0	315.222	441.193	TRUE
UTC (Ego Motion)	KITTI	571.084	0	- 623.765	- 518.402	TRUE
UTC (Ego Motion)	Matterport 3D	628.497	0	- 673.803	- 583.191	TRUE
UTC (Ego Motion)	SUNCG	984.000	0	- 1025.741	- 942.259	TRUE

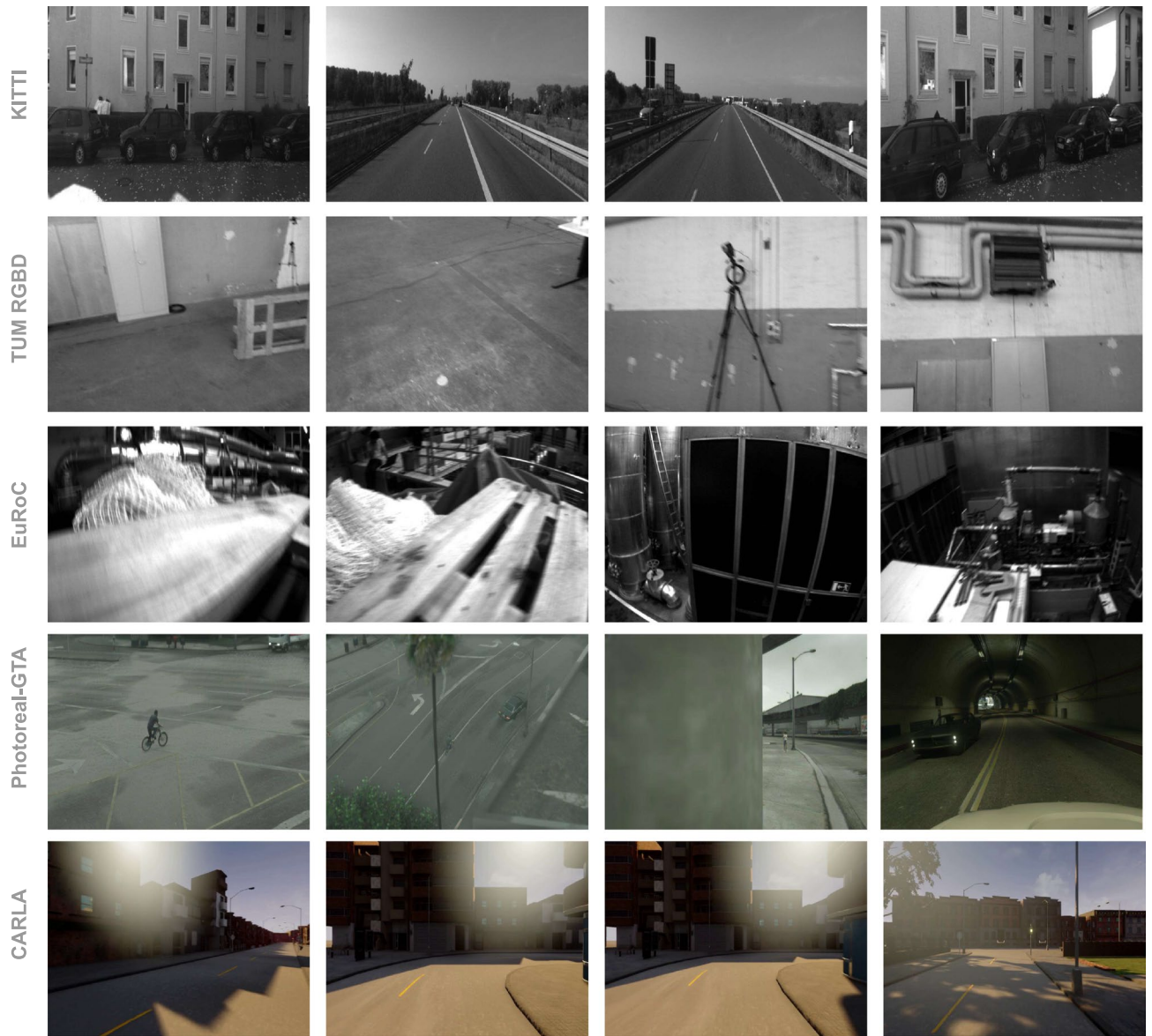
**Table 2.** Tukey's HSD pairwise comparisons of motion blur of each dataset against UTC (Ego Motion).

Another dataset captured in an uncontrolled outdoor road scenario (KITTI<sup>3</sup>) has the next least amount of blur (as indicated by large Laplacian invariance in Fig. 3). TUM-RGBD dataset<sup>4</sup> includes sequences captured with hand held camera and not surprisingly its blur is closer to ego motion dataset. Similarly, large indoor datasets (Matterport 3D<sup>58</sup>, SUNCG<sup>59</sup>) also have images with sharp edges as indicated with low Laplacian variance. These large datasets are adopted in photo-realistic platforms (MINOS<sup>23</sup>, Habitat<sup>56</sup>).

TartanAir dataset<sup>8</sup>, captured in photo realistic virtual environment, also has little amount of motion blur. Autonomous driving simulators (CARLA<sup>13</sup> and AirSim<sup>14</sup>), which imitate urban road scenes with changing light and weather conditions, fare slightly better in terms of motion blur. However, as shown later (Fig. 4), low Laplacian variance here results partly due to motion blur and mainly due to sun glare.

GTA-V is a popular gaming platform known for its photo-realism. Recently,<sup>52</sup> further enhanced the photo-realism of such platforms, by transferring style of real world cities. However, this approach still has low motion blur (second last row in Fig. 4). This indicates that photo-realism does not guarantee low Sim2Real gap. To further investigate the amount of motion blur, we sampled images (Fig. 4) with highest amount of motion blur according to the method suggested above. These images indicate that low variance of the Laplacian operator is due to closeups or texture-less scenes and not due to motion blur.

In order to reduce the gap between carefree and careful motion, we tried two methods. One option is to deblur the video sequence in real world before processing it for visual tracking. Deep image enhancement



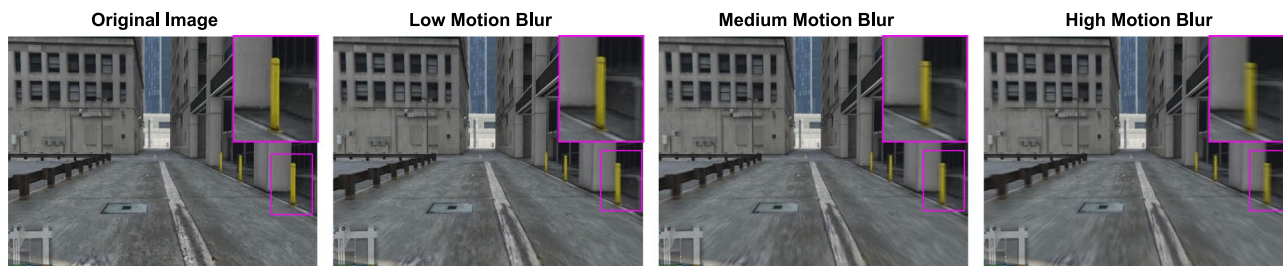
**Fig. 4.** Lack of motion blur: motion blur is indicated by low Laplacian variance. We sampled images with lowest Laplacian variance in different datasets. Note that for these samples low Laplacian variance value is mainly due to close-ups, textureless scenes and sun glare rather than motion blur. Photo-realistic GTA-V<sup>52</sup> generates realistic images. However, they contain little motion blur.

methods have shown impressive performance in denoising images. In this exciting work,<sup>60</sup> revealed the identity of the person whose face was blurred in order to hide her identity. As shown in the experiments, deep deblurring methods do not necessarily improve the tracking performance of visual SLAM methods. Interestingly, in some cases they further degrade the performance.

Instead of avoiding or removing blur, we propose to include motion blur to reduce the Sim2Real gap between realistic simulators and the real world. To this end, we introduce different levels of motion blur (Fig. 5) using the method of averaging consecutive frames<sup>61</sup> (Eq. 2).

$$I_i^B = \frac{1}{b+1} \sum_{i=j-b}^j I_i, \quad (2)$$

where  $I \in \mathbb{R}^{m \times n \times 3}$  is the blur-less image,  $I^B \in \mathbb{R}^{m \times n \times 3}$  is the blurred image, and  $b \in \{1, 2, 3\}$  controls the amount of blur, i.e., low, medium and high. The mean of variance of Laplacian filter response (Fig. 3) is 209.67, 145.99 and 101.58 for low, medium and high blur levels respectively. The corresponding value for the carefree ego motion in<sup>1</sup> is 57.32. As compared to our averaging approach (Fig. 3), especially the high blur level, the remaining datasets exhibit relatively higher values of this metric including lack of blur. Motion blur affects near



**Fig. 5.** Motion blur levels: near field scene appears more blurry as compared to far field.

Simulator	Photo-realistic	Dynamic lighting	Dynamic objects	Dynamic weather	Multiple agents	3D depth	Real counterpart	Customizable	Outdoor	User manual
Gazebo <sup>62</sup>					✓	2D scan				✓
AI2-THOR <sup>20</sup>	✓				✓		✓			✓
Gibson <sup>21</sup>	✓					✓				✓
HoME <sup>22</sup>					✓	✓				✓
MINOS <sup>23</sup>	✓					✓				✓
Habitat <sup>56</sup>	✓					✓				✓
DeepMind <sup>54</sup>						✓				✓
Malmo <sup>63</sup>			✓		✓			✓		✓
ViZDoom <sup>64</sup>			✓		✓	✓		✓		✓
CARLA <sup>13</sup>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AirSim <sup>14</sup>	✓	✓	✓	✓	✓	✓		✓	✓	✓
GTA-V <sup>*</sup>	✓	✓	✓	✓	✓	✓	✓	✓✓	✓	*
Replica <sup>65</sup>	✓					✓				✓
Active Vision <sup>66</sup>	✓					✓				✓

**Table 3.** Simulator comparison.

field objects more as compared to far field (Fig. 5). As shown (Fig. 3), this method reduced the gap between the carefree and careful motion.

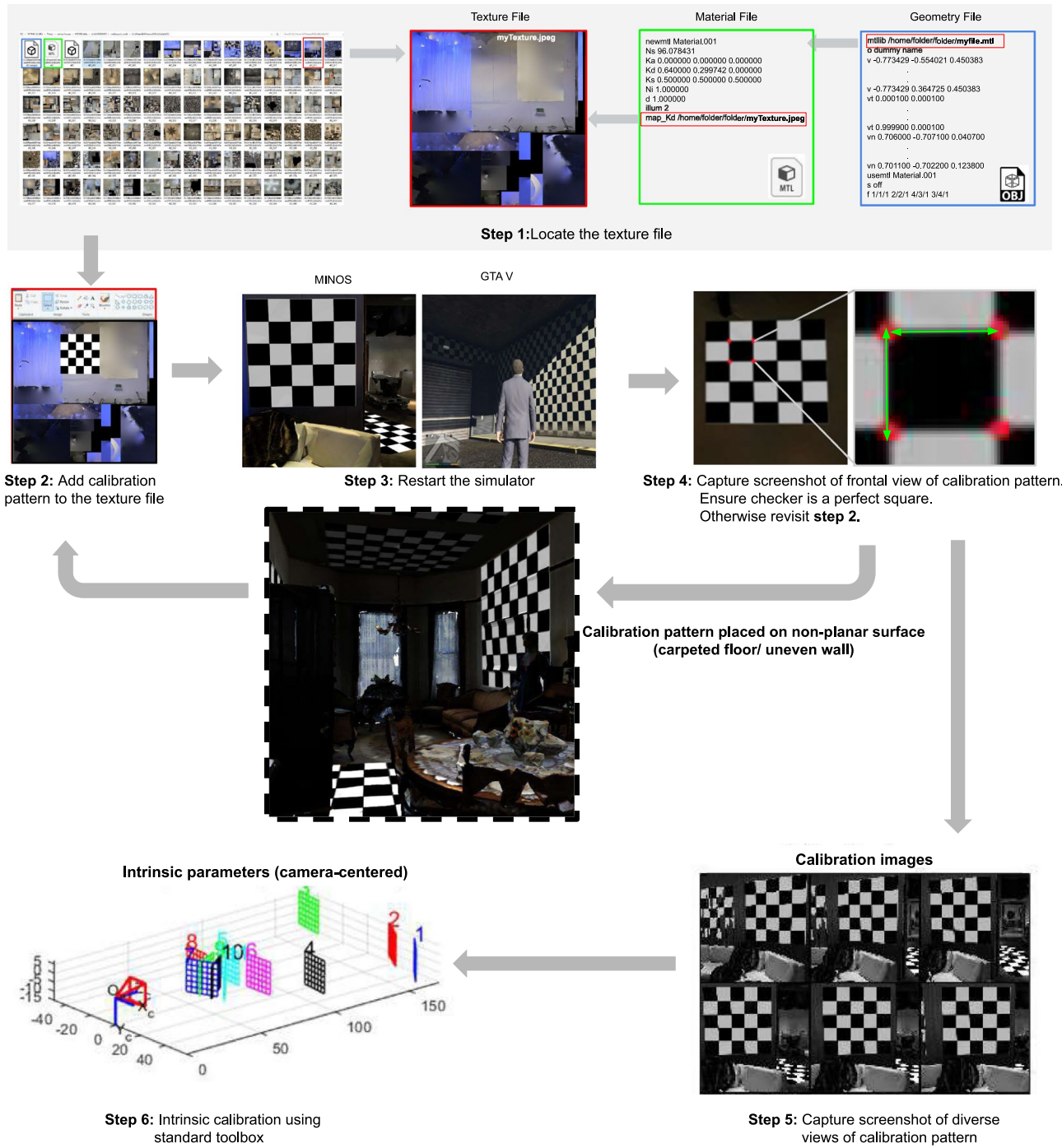
### Calibrating virtual environments

Hi-fidelity simulators provide an opportunity to automate exploration and avoid human bias (capture bias, negative world bias). Instead of extracting datasets<sup>15–19</sup> from these platforms using human intelligence, we propose to directly use these simulators for evaluation of SLAM methods. There is a rich family of simulators (Table 3). We demonstrate our simple calibration strategy on multiple platforms, i.e., MINOS<sup>23</sup> and GTA-V. While MINOS is a realistic indoor simulator containing reconstructed houses providing many different navigation situations, GTA-V is a well-known open world game (indoor, outdoor, underwater and airborne scenarios).

#### *Intrinsic calibration of the virtual environment*

Virtual environment's developer usually sets the camera intrinsic parameters. If these parameters are not provided separately, it is a challenging task to discover them from the source files. Decoding the source files requires patience and computer graphics knowledge, e.g., Sekkat et al.<sup>19</sup> estimated camera focal length using field-of-view (fov) information which is not directly accessible. Proprietary environments may replace source files with the executable files which are not human readable. Even if the camera intrinsics are provided one might wish to recheck their accuracy. Attempts to replicate real camera parameters in the virtual environments generally exclude this very important detail<sup>47,67</sup>. In our opinion, this might be due to ad-hoc (trial and error) approach which does not seem elegant.

Standard method of calibrating the camera requires placing the checker pattern in the environment<sup>68</sup>. This will require editing the 3D mesh to include a new object in the virtual environment which is not a trivial task. In this work, we propose a rather simple approach which is user friendly (Algorithm 1, Fig. 6). Virtual environment consists of the 3D geometry (3D mesh) and the texture file. Checker board is a planar object. Therefore, it can be pasted on any given surface (e.g. wall) just by editing the texture file corresponding to that surface.



**Fig. 6.** User friendly intrinsic calibration of virtual environment.

---

```

1: Input: .obj file (3D geometry), .mtl file (Texture details), .jpeg file (Texture map)
2: Locate the texture file
3: while each checker in the calibration pattern is not a perfect square do
4:   Add calibration pattern by editing the texture image
5:   Restart the simulator environment
6:   Capture screenshot of the calibration pattern without perspective effect
7:   if checkers are not perfect squares then
8:     Select a different surface or resize the calibration image (pattern)
9:   end if
10: end while
11: Capture multiple screenshots of the calibration pattern from diverse views
12: Use a standard camera calibration toolbox (e.g., Bouguet's toolbox)
13: return Intrinsic calibration parameters

```

---

### Algorithm 1. Intrinsic calibration of a virtual environment

Texture file itself is an image, therefore, editing it is straight forward. However, identifying the correct texture file within the 3D model's asset hierarchy can be nontrivial, as textures are often dispersed across multiple package directories (first row in Fig. 6). These textures files are montage images containing textures of multiple objects and scenes in a single file. Therefore, they are, in principal, distinguishable. Another way to locate these texture files is by searching for files containing information about the geometry of the scene (step 1 in Fig. 6). These files have standard extension, e.g., .obj etc. These geometry files contain detailed paths to image files (.png, .jpeg etc.), which are usually texture files, or material files (.mtl) which ultimately contain paths to the texture images. Once the texture file is located and edited with checkered pattern (step 2 in Fig. 6), restarting the virtual environment will display the standard checker board on one of its surfaces (step 3 in Fig. 6).

Next step in standard camera calibration toolbox requires providing metric size of a single checker in the pattern. This again will require making measurements in the underlying mesh of the virtual environment to figure out the size of the checker. Interestingly, this information, although commonly required, is not necessary. We provided random checker size in<sup>68</sup> toolbox and it resulted in same camera focal length (Table 4). However, one needs to ensure that checker remains a perfect square. Edited texture file might be stretched converting a square into a rectangle. Sometimes, the calibration pattern appears on the non-planar surface like carpeted floor, or curtains on the wall. In this case, one has to revisit the texture files and choose a different surface for inserting the calibration pattern or resize the calibration pattern so that checkers appear as perfect squares.

In order to confirm that checker is a perfect square, one should capture a frontal image of the surface containing the checker pattern (step 4 in Fig. 6). Care should be taken that there is no perspective effect. Once the frontal view is captured, both sides of the checker should have equal size in the image space. Capturing multiple views (screenshots) of the embedded checker will help in completing the intrinsic calibration of the first person view of the simulator.

There are alternative camera calibration methods which do not require a calibration object and can work on images captured by the camera only (auto calibration chapter in<sup>69</sup>). However, such methods are not common and require building the camera calibration toolbox from scratch. Our user friendly approach avoids this hurdle.

#### *Extrinsic calibration of the virtual environment*

How accurate is a given simulator as compared to well-known benchmark SLAM datasets? In this work, we evaluate the accuracy of the ground truth pose provided by a given simulator, in a user friendly manner without editing the virtual scene, or making measurements using the underlying 3D mesh. As mentioned earlier this reverse engineering requires sophisticated graphics knowledge.

For a given image  $I \in \mathbb{R}^{m \times n \times 3}$ , the simulator provides the ground truth pose  $H = \begin{pmatrix} R & t \\ 0 & 0 & 0 & 1 \end{pmatrix}$ , where  $R \in \mathbb{R}^{3 \times 3}$  is the 3D rotation and  $t \in \mathbb{R}^{3 \times 1}$  is the 3D translation. We can leverage epipolar geometry to estimate the accuracy of the ground truth pose  $H$ . Assume  $x$  and  $x'$  are the corresponding points in the two different

House	Checker width	Focal length
1	20	960
2	60	714
3	100	713
4	150	712
5	200	713
6	300	712
7	400	713
8	500	712
9	600	714
10	700	712

**Table 4.** Effect of checker size on focal length estimation.

views (images), i.e.,  $I$  and  $I'$ . Here  $x = [a, b, c]^T$  is the homogeneous representation<sup>69</sup> of a pixel with coordinates  $(x, y) = (a/c, b/c)$ . Then corresponding points satisfy the following condition,

$$x'^T F x = 0, \tag{3}$$

where  $F \in \mathbb{R}^{3 \times 3}$  is the fundamental matrix. Assuming we have accurately estimated  $F$ , the corresponding point  $x'$  must exist on the epipolar line  $Fx$ , otherwise  $|x'^T F x|$  represents the inaccuracy of  $H$ , since  $F$  is derived from  $H$  as follows,

$$F = K^{-T} R_{\Delta} K^T [K R_{\Delta} t_{\Delta}]_x, \tag{4}$$

where  $K \in \mathbb{R}^{3 \times 3}$  is the camera intrinsic matrix (estimated in the last section). We are assuming both the views have the same intrinsic parameters.  $[u]_x$  is the skew symmetric form of a vector  $u$ .  $R_{\Delta}$  and  $t_{\Delta}$  represent the relative rotation and translation respectively between the two views, and are derived from corresponding ground truth poses  $H$  and  $H'$  as follows,

$$R_{\Delta} = R' R^{-1} \tag{5}$$

$$t_{\Delta} = -R' R^{-1} t + t' \tag{6}$$

Given Eq. 4, Eq. 5 and Eq. 6, we can solve Eq. 3. However, fundamental matrix requires information about the coordinate system conventions (detailed discussion in Appendix A). If the conventions are not well documented, as is the case in proprietary platforms, there exists a vast search space of possible conventions (Appendix B). We demonstrate that discovering this convention is not trivial (Appendix C).

In this section, we describe our user friendly approach to estimate  $H_{\Delta}$  using the ground truth poses only (Algorithm 2, Fig. 7). Lets assume that the current view (red image frame in Fig. 7) of the simulator has a pose

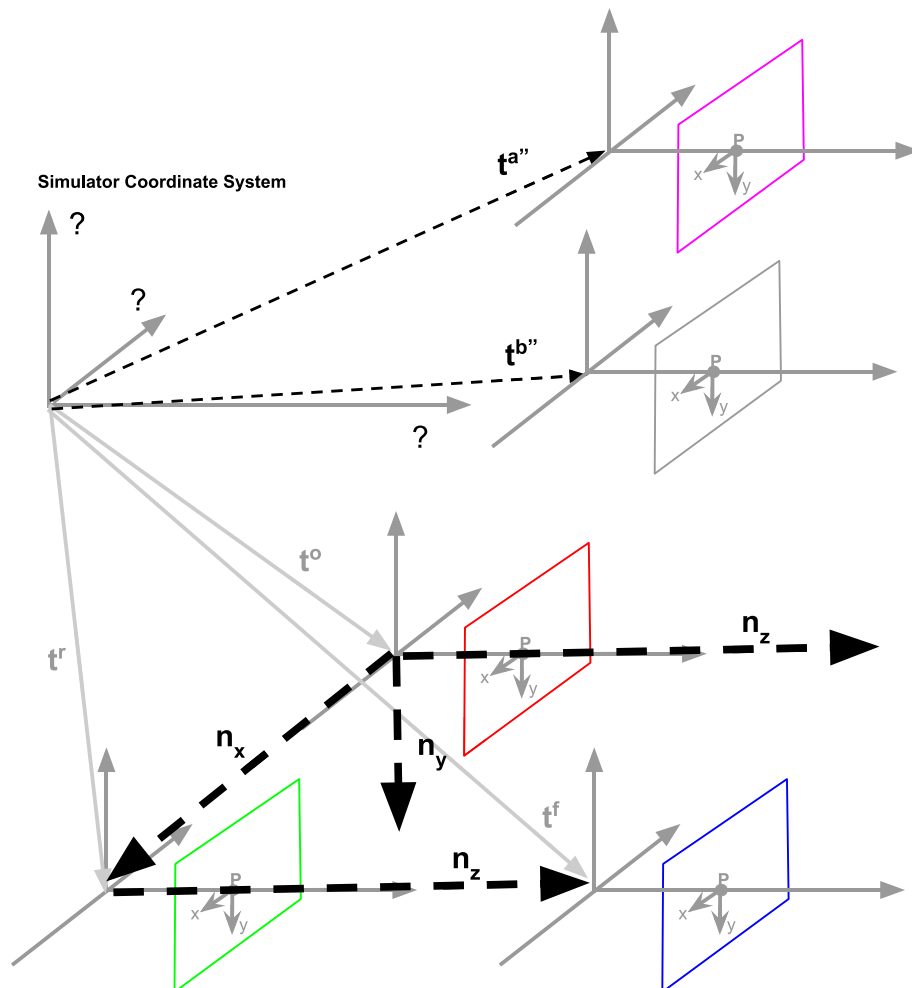


Fig. 7. Transforming unknown coordinate system to known coordinate system: a user friendly approach.

$(R^o, t^o)$ , with unknown convention. Generally the simulators follow the WASD keys to move (W forward, A left, S back and D right). Press D to move right few steps. Lets say the pose of the current view (green image frame in Fig. 7) is  $(R^o, t^r)$ . Note that only translation has changed. Now move forward by pressing W. The pose of the current view (blue image frame in Fig. 7) is  $(R^o, t^f)$ . Again there is no change in rotation.

Given the three translation vectors  $(t^o, t^r$  and  $t^f)$  in unknown coordinate system, we can determine the transformation to a known coordinate system (x-axis rightwards, y-axis downwards and z-axis, according to right-hand rule, forwards) as follows. Let  $n_x, n_y$  and  $n_z$  be the unit vectors in the direction of x-axis, y-axis and z-axis respectively.

$$n_x = \frac{t^r - t^o}{\|t^r - t^o\|} \tag{7}$$

$$n_z = \frac{t^f - t^r}{\|t^f - t^r\|} \tag{8}$$

$$n_y = n_z \times n_x \tag{9}$$

Lets suppose we have any random view (magenta image frame in Fig. 7) of the simulator with pose  $(R^o, t^a)$ . We can transform its translation  $t^a = [a, b, c]^T$  to the new coordinate system, i.e.,  $t^{a''} = [t_1, t_2, t_3]^T$  as follows,

$$t^{a'} = t^a - t^o \tag{10}$$

$$\begin{aligned} t_1 &= n_x^T t^{a'} \\ t_2 &= n_y^T t^{a'} \\ t_3 &= n_z^T t^{a'} \end{aligned} \tag{11}$$

In order to estimate the accuracy of the given simulator using epipolar constraint, we need another view (grey image frame in Fig. 7) which has the same orientation as the current view whose translation vector we have transformed. We can move the view by pressing any of WASD keys. There should be some overlap between these views. Lets say the pose of the new view is  $(R^o, t^b)$ . We will transform its translation vector to the new coordinate system, i.e.,  $t^{b''}$ . Given this, the relative pose  $(R_\Delta, t_\Delta)$  between these two views is given by

$$\begin{aligned} R_\Delta &= I \\ t_\Delta &= t^{b''} - t^{a''} \end{aligned} \tag{12}$$

Using Eq. 12 in Eq. 4 leads to,

$$F = [Kt_\Delta]_x \tag{13}$$

Using Eq. 13 in Eq. 3 will allow to apply epipolar constraints. Our user friendly approach takes only a few minutes to estimate the accuracy of any simulator without its reverse engineering.

1: INPUT:  $R^o, t^o$

▷ Initial pose of the simulator with unknown convention

2: Compute Axes of User Defined Coordinate System:

1. Move right (Press D). Now the pose is  $(R^o, t^r)$
2. Move forward (Press W). Now the pose is  $(R^o, t^f)$
3. Compute direction vectors of the user defined coordinate system:

$$n_x = \frac{t^r - t^o}{\|t^r - t^o\|}, n_z = \frac{t^f - t^r}{\|t^f - t^r\|}, n_y = n_z \times n_x$$

3: Transform Simulator Pose to User Defined Coordinate System:

1. Navigate simulator to random view. Now the pose is  $(R^o, t^a)$
2. Transform its translation  $t^a = [a, b, c]^T$  to the new coordinate system:

$$t^{a'} = t^a - t^o, t_1 = n_x^T t^{a'}, t_2 = n_y^T t^{a'}, t_3 = n_z^T t^{a'}$$

3. Randomly translate by pressing any of WASD keys. Now the pose is  $(R^o, t^b)$ . Transform its translation vector  $t^b$  to the new coordinate system, similar to last step.

4: Compute Fundamental Matrix in User Defined Coordinate System:

1. The relative pose between two random views (from the last part) is given by:  $R_\Delta = I, t_\Delta = t^{b''} - t^{a''}$
2. Fundamental matrix between these views is given by:  $F = [Kt_\Delta]_x$

**Algorithm 2.** Benchmarking (extrinsic calibration) of a virtual environment**Results and discussion****Experimental setup**

Experiments have been conducted inside MINOS simulator. While extracting navigation sequences from MINOS, the start and goal positions are manually provided, while the path from the start to the goal position is planned by a MINOS inbuilt path planner to avoid human bias in selecting favorable path. For the case of loop closures, we manually generated navigation sequences to include multiple loop closures in each sequence. All sequences are extracted at standard frame rate (30 FPS) with  $640 \times 480$  resolution. All default MINOS and GTA-V parameters were used.

Similarly, ORB-SLAM and ORB-SLAM3 have been used as visual SLAM methods for evaluation at their standard thresholds in various parameters. The tracking failure is directly related with number of features required in each frame, which has been kept at default value (1000 ORB features per frame). Similarly, thresholds related to loop closure pipeline were kept at their default setting.

**Evaluating intrinsic and extrinsic calibration**

In order to analyze the accuracy of our user friendly intrinsic and extrinsic calibration of simulators, we evaluate epipolar constraints (Table 5). More concretely we measure the distance between the epipolar line and the corresponding point in pixels. Fundamental matrix required for epipolar line is calculated using our intrinsic and extrinsic calibration. Firstly, we estimate the quality of the ground truth provided with KITTI<sup>3</sup> and TUM-RGBD<sup>4</sup> datasets (Table 5). Interestingly, not all the ground truths are equally accurate. *Results indicate that ground truth provided by simulators (MINOS<sup>23</sup>, GTA-V, Blender<sup>70</sup>) is more accurate than these datasets.* Hence,

	KITTI <sup>3</sup>		TUM-RGBD <sup>4</sup>		MINOS <sup>23</sup>		Blender <sup>70</sup>		GTA-V <sup>*</sup>	
	F	F'	F	F'	F	F'	F	F'	F	F'
Case 1	18.05	8.53	178.03	247.72	6.03	9.70	0.66	0.66	0.90	0.90
Case 2	181.40	62.10	402.68	339.49	10.74	15.64	1.22	1.41	0.79	0.90
Case 3	42.09	71.74	52.88	31.37	4.99	8.31	2.14	2.14	0.94	1.06
Case 4	7.82	48.78	167.80	217.37	1.27	1.59	1.05	1.22	3.85	11.73
Case 5	32.12	40.21	116.72	166.45	3.61	5.42	0.54	0.63	3.15	4.75
Case 6	19.79	3.95	0.57	17.22	6.55	11.61	0.66	0.66	0.82	0.92
Case 7	3.78	6.16	30.59	124.77	2.15	3.12	1.55	2.07	1.89	2.33
Case 8	1.40	10.39	161.24	40.23	5.67	4.95	1.14	1.45	1.13	1.12
Case 9	315.20	135.28	380.00	372.78	7.97	10.48	0.19	0.25	1.20	1.35
Case 10	4.97	29.69	83.21	20.02	0.86	6.08	0.18	0.19	3.89	8.63
Case 11	1.44	3.80	68.57	176.91	10.98	17.85	0.64	0.78	5.33	5.33
Case 12	20.82	14.72	32.59	12.12	3.69	4.61	1.00	1.00	1.99	1.99
Case 13	65.94	47.56	92.15	19.66	1.41	2.21	0.71	1.12	0.69	0.69
Case 14	35.21	1.61	33.88	57.80	2	7.96	1.06	1.06	0.69	0.69
Case 15	7.89	25.02	152.63	36.78	1.76	7.23	0.20	0.22	2.06	2.24
Case 16	143.36	24.14	318.25	297.60	0.19	1.93	3.09	4.98	4.99	5.34
Case 17	34.82	9.54	156.23	182.44	0.97	1.14	0.62	0.84	6.75	11.28
Case 18	34.11	115.01	203.25	223.89	6.85	3.06	4.51	3.81	18.70	23.38
Case 19	4.77	13.39	69.24	21.70	0.46	1.54	0.90	0.90	2.30	2.26
Case 20	0.74	51.05	302.04	233.29	13.14	14.87	0.79	0.90	1.78	1.77
Case 21	18.94	7.33	124.63	121.02	1.89	4.85	0.94	1.06	1.26	1.13
Case 22	13.41	26.00	236.68	259.63	15.36	14.99	3.85	11.73	2.21	3.67
Case 23	17.86	14.87	92.61	95.58	8.37	2.07	3.15	4.75	4.30	3.35
Case 24	5.89	74.15	31.88	70.97	10.38	7.86	0.82	0.92	1.58	1.48
Case 25	53.10	55.80	77.87	83.09	4.52	3.09	1.89	2.33	2.73	2.83
Avg	43.40	36.04	142.65	138.80	5.28	6.89	1.34	1.88	3.04	4.05
Var	4861.86	1173.10	11875.75	11772.96	17.68	23.68	1.27	5.62	12.79	24.91

**Table 5.** Comparison of ground truth of visual SLAM datasets and photo-realistic simulators: in order to judge the accuracy of ground truth pose, we calculated the epipolar line using the intrinsic/extrinsic pose and measured its distance with the corresponding point. Given an image pair and their relative pose, we estimate the distance between epipolar line and its corresponding point in both directions (forward F and backward F'). Ground truth pose provided by simulators, i.e., MINOS, Blender and GTA-V (last three columns) is orders of magnitude accurate than public datasets, i.e., KITTI, TUM-RGBD (first two columns).

simulators provide a much more accurate avenue for visual localization tasks. We achieve average error of 5.28 pixels, 1.34 pixels and 3.04 pixels for MINOS, Blender and GTA-V respectively as compared to 43.40 pixels and 142.65 pixels for KITTI and TUM-RGBD respectively. Furthermore, such low errors for simulators indicate the quality of our calibration method.

### Analyzing capture bias

TartanAir dataset<sup>8</sup> consists of aggressive motion in unstructured environments. Therefore, we selected this dataset to estimate the capture bias. We manually viewed 369 sequences ( $\approx$  510 minutes in length). There are 33 sequences in various categories in TartanAir dataset containing 49 revisited scenes (loop closures). Detailed labels are provided in Appendix D.

We evaluated state-of-the-art visual SLAM family (ORB-SLAM<sup>39</sup> and ORB-SLAM3<sup>41</sup>) on these 33 sequences. Interestingly ORB-SLAM managed to detect loop closures on only 7 occasions out of which 2 were false positives. Surprisingly, ORB-SLAM3 did not close any loop in this dataset (Table 6).

Furthermore, we evaluated ORB-SLAM and ORB-SLAM3 on frequently occurring and less aggressive motion in indoor sequences, which we captured using MINOS (Fig. 8), where ORB-SLAM3 (46 out of 170) considerably outperforms ORB-SLAM (20 out of 170) (Table 6).

In our opinion, large number of missed loop closure opportunities are due to the fact that these systems are tuned for datasets having capture bias including easy loop closures as indicated by large number of loops closed by ORB-SLAM in KITTI and TUM-RGBD dataset (Table 6).

### Analyzing negative world bias

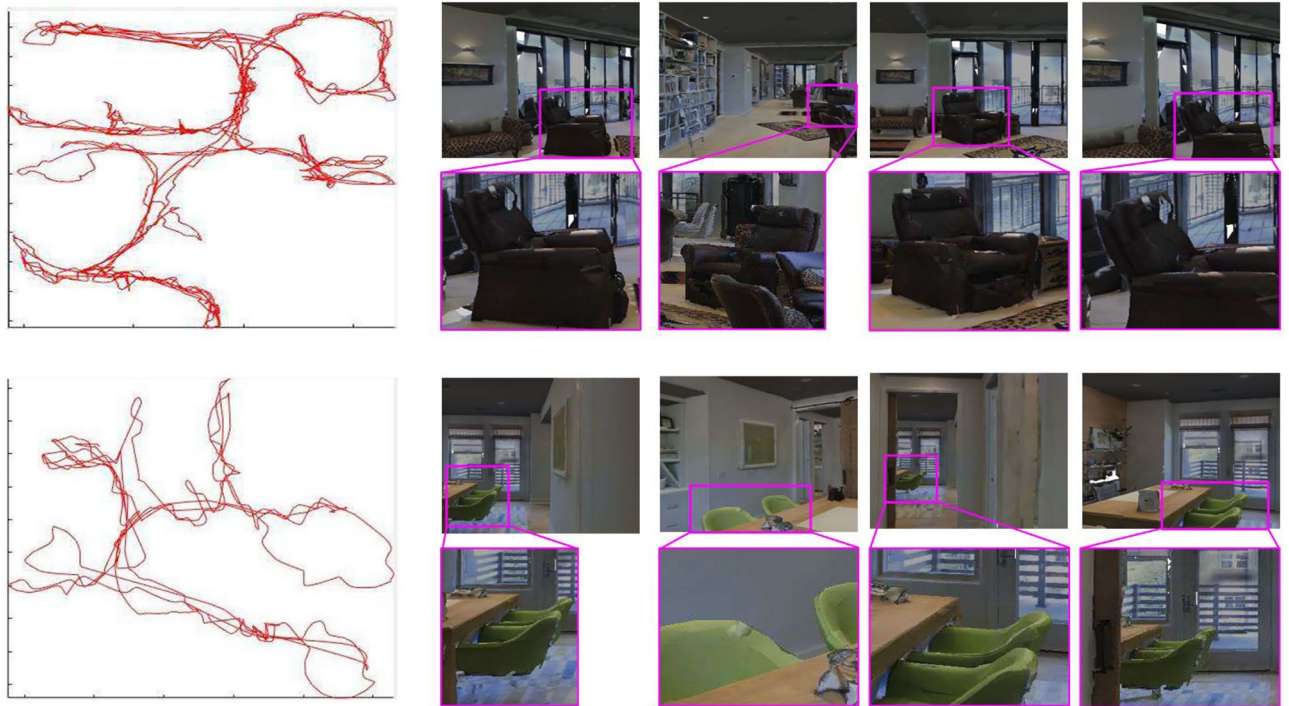
The visual navigation datasets captured by human agents suffer from human bias. In order to benchmark visual tracking performance without human supervision, we auto captured sequences using MINOS inbuilt path planner. The start and goal positions were auto-generated, and the path from start to goal was planned by MINOS (without any human supervision in the process). The tracking performance of ORB-SLAM and its improved version, ORB-SLAM3, is shown in Fig. 9. For the 25 sequences of 13.48m average length, the distance covered by ORB-SLAM (without tracking failure) is 1.38m, while that of ORB-SLAM3 is 9.33m. ORB-SLAM reaches the goal (without tracking failure) in none of 25 sequences, while this number for ORB-SLAM3 is 8/25. Our calibration of the simulator has actually enabled us to generate negative world (failure cases) examples for navigation algorithms.

When the human is allowed to control the camera motion, visual SLAM successfully manages to complete long and challenging routes<sup>24</sup> in this simulator. Kanwal et al.<sup>24</sup> demonstrated that the ORB-SLAM family<sup>39,41</sup> consistently loses tracking across transitions (door crossings), when the camera motion is automatically planned and not controlled by human operator. However, an expert human successfully navigates the camera across these transitions without tracking failures.

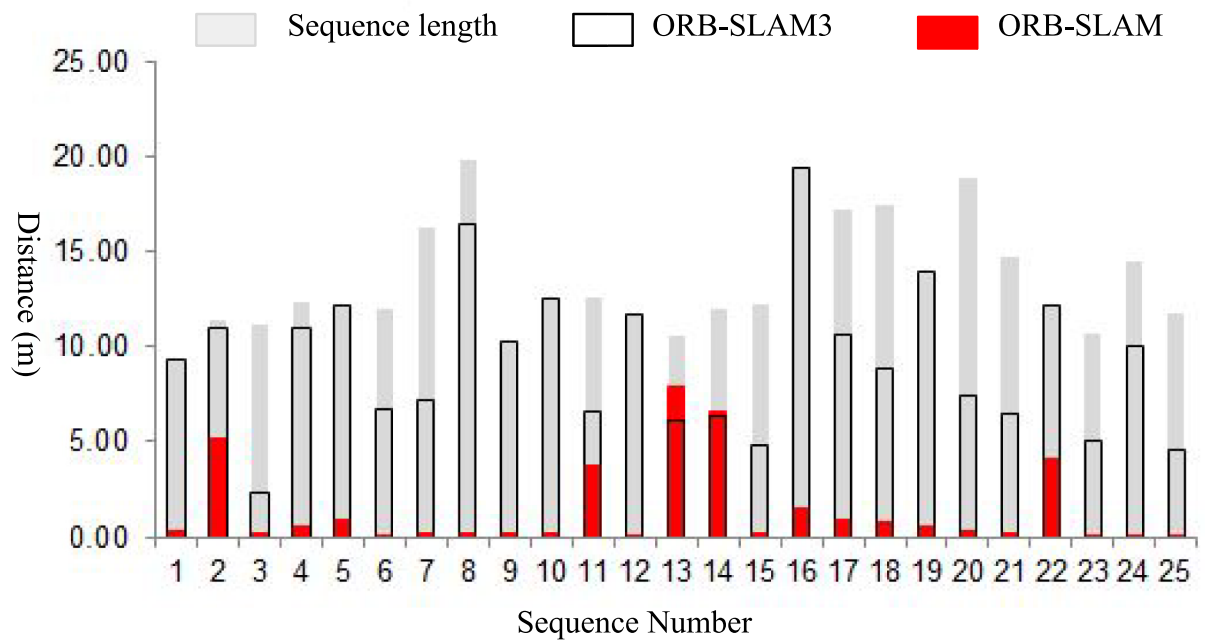
Furthermore, we evaluated ORB-SLAM<sup>39</sup> and SVO<sup>38</sup> on 369 sequences ( $\approx$  510 minutes in length) in various categories of TartanAir dataset which includes aggressive maneuvers. ORB-SLAM runs  $\approx$  318 mins ( $\approx$  62% of the time) while SVO runs  $\approx$  146 mins ( $\approx$  28% of the time) in total across all sequences. Surprisingly, SVO<sup>38</sup> was developed to handle aggressive maneuvers, e.g., drone maneuvers. Detailed results are available in Appendix E.

Sequence	Seq length (M:S)	Available LCs	Loops closed ORB-SLAM3	Loops closed ORB-SLAM
KITTI (06 Seqs)	-	13	12	12
TUM-RGBD (03 Seqs)	-	3	2	3
TartanAir (33 Seqs)	78:28	49	0	5
Seq-01	01:48	9	3	1
Seq-02	02:46	7	4	2
Seq-03	10:55	10	2	2
Seq-04	09:17	37	11	0
Seq-05	09:43	3	1	2
Seq-06	05:12	21	4	3
Seq-07	09:23	3	1	0
Seq-08	04:21	5	3	2
Seq-09	03:48	3	0	2
Seq-10	01:16	6	1	2
Seq-11	04:27	16	4	0
Seq-12	13:55	50	12	4
Total (ours) (12 Seqs)	-	170	46	20

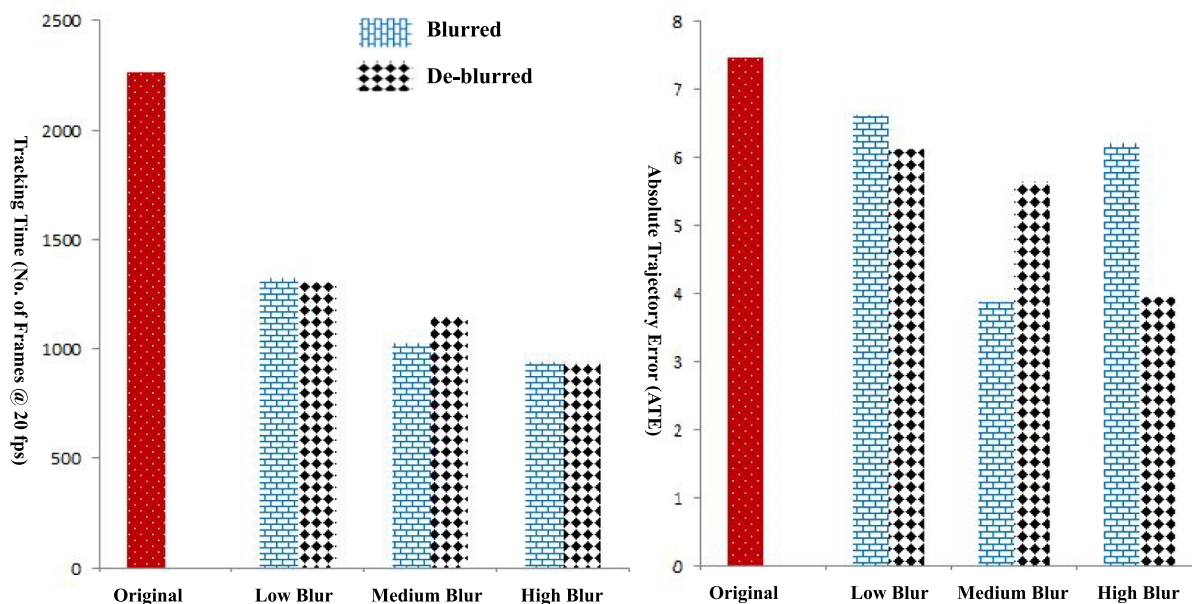
**Table 6.** Effect of capture bias on visual SLAM. ORB-SLAM manages to close almost all the loop closures (LCs) in KITTI and TUM-RGBD dataset. Our sequences contain large number of frequently occurring LCs in indoor scenarios. ORB-SLAM and ORB-SLAM3 miss large number of loop closures in our as well TartanAir<sup>8</sup> dataset. In our opinion, state-of-the-art visual SLAM is tuned on datasets having capture bias, and hence its performance degrades on datasets having slightly challenging LCs.



**Fig. 8.** Natural loop closures: we captured few naturally occurring loop closures where the same object/location is approached with slight change of viewpoint. The trajectory plot on the left indicates revisits as separate tracks whereas in Fig. 2 revisits substantially aligned and were not distinguishable indicating capture bias.



**Fig. 9.** Negative world bias: tracking failure in ORB-SLAM and ORB-SLAM3 on sequences where path is not planned by human. ORB-SLAM reaches the goal in none of the 25 considered sequences. ORB-SLAM3, an improved version, does manage to reach the goal in 8/25 sequences. Average sequence length is 13.48m. Average distance covered by ORB-SLAM is 1.38m, while that of ORB-SLAM3 is 9.33m.



**Fig. 10.** Sim2Real gap: we applied different levels of blur (low, medium, high) on sequences acquired through photo-realistic simulator GTA-V. Tracking performance of state-of-the-art visual SLAM drastically degrades even with low blur. Deep deblur<sup>71</sup> does not necessarily improve tracking performance. In some cases (medium blur) ATE degrades after deblurring. As compared to changing conditions (season, weather, times of the day, dynamic objects), where tracking was successful during entire track length (Table 7), motion blur considerably degrades visual SLAM tracking. Photo-realism in simulators does not guarantee motion blur. Therefore, these photo-realistic platforms overestimate visual SLAM performance.

	Building						Carnival						Little Seoul								
	Low		Medium		High		Low		Medium		High		Low		Medium		High				
	O	B	D	B	D	B	D	O	B	D	B	D	B	D	O	B	D	B	D		
Day	1365	22	23	22	21	20	22	1302	38	37	37	37	39	37	3844	49	50	49	50	48	50
Night	1821	40	40	39	39	38	39	1617	36	36	36	36	36	1	4337	60	4	30	57	31	30
Rain	1546	19	42	18	21	18	21	1693	83	83	42	41	39	39	4112	77	100	61	67	78	99
Snow	1287	19	40	19	19	19	18	1359	45	45	44	45	1	44	3597	54	55	54	54	49	48
Dynamic	1399	27	27	25	25	24	26	1142	43	42	43	42	42	43	3692	94	56	17	17	18	17

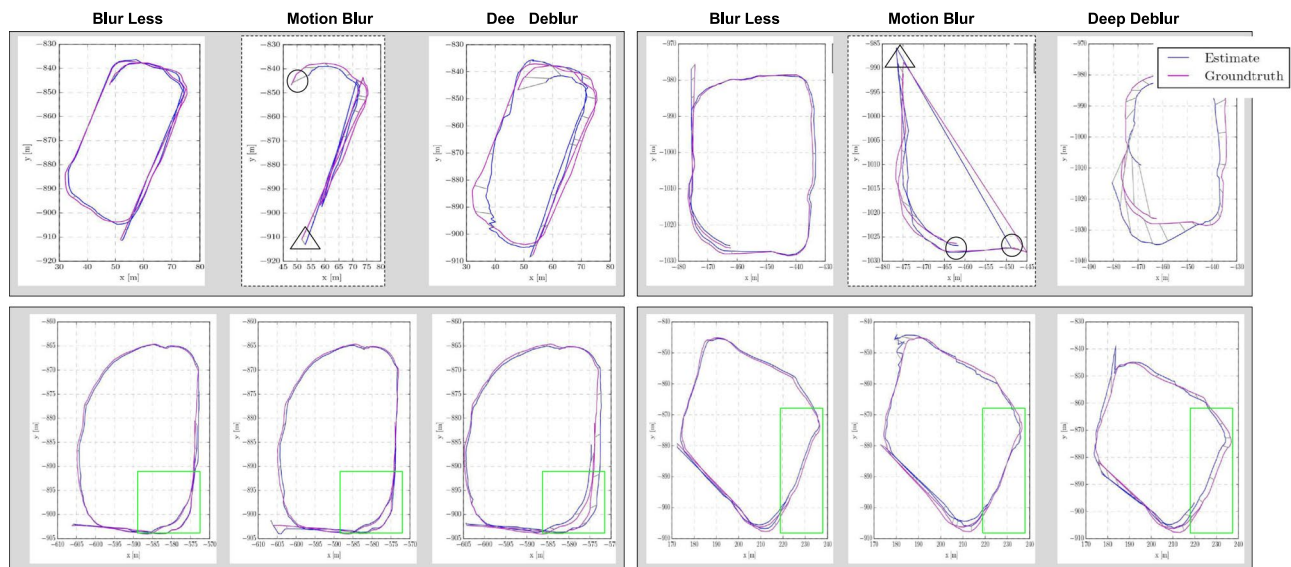
**Table 7.** Sim2Real gap: we applied different levels of blur (low, medium, high) on sequences acquired through photo-realistic simulator GTA-V. O indicates the total length of the sequence in number of frames, B and D indicate the fraction of sequence successfully tracked by visual SLAM after blurring and subsequently deblurring. Tracking performance of state-of-the-art visual SLAM drastically degrades even with low blur. Deep deblur<sup>71</sup> does not necessarily improve tracking performance. As compared to changing conditions (season, weather, times of the day, dynamic objects), where tracking was successful during entire track length, motion blur considerably degrades visual SLAM tracking.

In our opinion, simulators offer opportunities to rigorously evaluate SLAM approaches in challenging and realistic scenarios which are not manageable in real world tests.

### Analyzing motion blur

As mentioned earlier, photo-realistic simulators have little motion blur. In order to evaluate the performance of ORB-SLAM3 in realistic conditions, we added three levels of blur (low, medium, high) to all the sequences consisting of changing conditions (Fig. 10, Table 7). We also evaluated SLAM on improved sequences using a deep deblur approach<sup>71</sup>.

Even low level blur degrades the ORB-SLAM3's tracking time to almost half (Fig. 10). In few cases, deep deblur approach considerably improves the tracking time by avoiding tracking failures (first row in Fig. 11). However, on average deep deblurring<sup>71</sup> offers little improvement in track time (Fig. 10). Increased tracking time, by avoiding tracking failures, using deblurring approach does not necessarily indicate improved performance. On the contrary, in case of medium blur, deep deblur increases the ATE (Fig. 10). This may be due to additional



**Fig. 11.** Why do we need to include motion blur in visual tracking analysis? Motion blur is unavoidable in carefree motion with standard frame rate (30 FPS). Motion blur degrades tracking performance and in some cases causes tracking failures (two examples in the first row). Motion blur degrades tracking performance and in some cases causes tracking failures (two examples in the first row). Triangle indicates the starting points and circle indicates the point of track loss. Removing blur using deep de-blur<sup>71</sup> avoids tracking failures (two examples in the first row). However, de-blurring does not necessarily improve tracking performance. In some cases (two examples in bottom row), the performance of tracking further degrades after de-blurring. The highlighted regions in bottom row demonstrate that estimated (blue) trajectory deviates from ground truth (magenta) trajectory after de-blurring while it was well aligned in the blurred version.

noise added due to the deblurring approach (compare trajectory regions highlighted by green boxes in second row of Fig. 11).

In conclusion, motion blur has a much more devastating effect on visual SLAM as compared to changing conditions. Visual SLAM managed to retain tracking during the entire sequence in different conditions (Table 7) whereas, even little blur degraded the track time considerably, which was negligibly improved by deep deblur. In our opinion, photo-realistic platforms, by ignoring motion blur, overestimate the performance of visual tracking. Our proposed motion blur approach may reduce this dataset/simulation-to-reality gap.

Our work extends the findings reported in the literature. Incorporating real camera limitations improves optical flow estimation using deep networks trained on synthetic photo-realistic data<sup>72</sup>. Lopez et al.<sup>73</sup> showed that single image depth estimation improved for real settings if the deep network was trained on reconstructed outdoor scenes. These limitations are reported for deep learning methods (optical flow, single image depth estimation). In this work, we show that Sim2Real gap affects the non-learning based visual localization approaches as well.

## Conclusion

Although visual SLAM has made significant progress, achieving the level of robustness required for real-world deployment still demands overcoming several existing limitations. We have diagnosed few biases (capture bias, negative world bias) in visual SLAM datasets, which need to be diluted in order to improve out-of-the-lab performance of visual SLAM. Photo-realistic simulators provide an unbiased environment to overcome these human caused biases. We have proposed a user friendly approach to calibrate these simulators. Furthermore, we have proposed an efficient approach to reduce the gap between photo-realistic simulators and real test environments. We hope this effort will assist the community in coming up with robust and successful SLAM agents in real world settings.

## Data availability

Our user-friendly and step-by-step calibration method for both simulators i.e., MINOS and GTA-V can be accessed at: [https://github.com/Abbrar3616/UnbiasedSLAM\\_Simulator](https://github.com/Abbrar3616/UnbiasedSLAM_Simulator). All data generated or analyzed during this study are included in published articles such as KITTI, TUM-RGBD, TartanAir, Blender, Carla and MINOS. GTA-V is a proprietary game, and its commercial use should be discussed with its parent company Rockstar Games.

Received: 17 August 2025; Accepted: 15 December 2025

Published online: 03 January 2026

## References

- Lee, Y.J., Ghosh, J. & Grauman, K. Discovering important people and objects for egocentric video summarization. In *CVPR* (2012).
- Smith, M., Baldwin, I., Churchill, W., Paul, R. & Newman, P. The new college vision and laser data set. *Int. J. Rob. Res.* **28**, 595–599 (2009).
- Geiger, A., Lenz, P. & Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR* (2012).
- Sturm, J., Engelhard, N., Endres, F., Burgard, W. & Cremers, D. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, 573–580 (IEEE, 2012).
- Fontana, G., Matteucci, M. & Sorrenti, D. G. Rawseeds: Building a benchmarking toolkit for autonomous robotics. In *Methods Exp. Tech. Comput. Eng.*, 55–68 (Springer, 2014).
- Blanco-Claraco, J.-L., Moreno-Duenas, F.-A. & Gonzalez-Jimenez, J. The malaga urban dataset: High-rate stereo and lidar in a realistic urban scenario. *Int. J. Robot. Res.* **33**, 207–214 (2014).
- Majdik, A. L., Till, C. & Scaramuzza, D. The zurich urban micro aerial vehicle dataset. *Int. J. Robot. Res.* **36**, 269–273 (2017).
- Wang, W. *et al.* Tartanair: A dataset to push the limits of visual slam. In *IROS* (2020).
- Goransson, R., Aydemir, A. & Jensfelt, P. Kinect@ home: A crowdsourced rgb-d dataset. In *Intell. Autonom. Syst.* **13**, 843–858 (Springer, 2016).
- Agarwal, S. *et al.* Building rome in a day. *Commun. ACM* **54**, 105–112 (2011).
- Pinto, L. & Gupta, A. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *ICRA* (2016).
- Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J. & Quillen, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Robot. Res.* **37**, 421–436 (2018).
- Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. & Koltun, V. Carla: An open urban driving simulator. In *Conference on Robot Learning*, 1–16 (PMLR, 2017).
- Shah, S., Dey, D., Lovett, C. & Kapoor, A. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field Service Robot.*, 621–635 (Springer, 2018).
- Minoda, K., Schilling, F., Wüest, V., Floreano, D. & Yairi, T. Viode: A simulated dataset to address the challenges of visual-inertial odometry in dynamic environments. *IEEE Robot. Autom. Lett.* **6**, 1343–1350 (2021).
- Kaneko, M., Iwami, K., Ogawa, T., Yamasaki, T. & Aizawa, K. Mask-slam: Robust feature-based monocular slam by masking using semantic segmentation. In *CVPR Workshops*, 258–266 (2018).
- Kloukinitiotis, A. *et al.* Carlasceles: A synthetic dataset for odometry in autonomous driving. In *CVPR*, 4520–4528 (2022).
- Sekkat, A. R., Dupuis, Y., Vasseur, P. & Honeine, P. The omniscene dataset. In *ICRA*, 1603–1608 (IEEE, 2020).
- Sekkat, A. R. *et al.* Synwoodscape: Synthetic surround-view fisheye camera dataset for autonomous driving. *IEEE Robot. Autom. Lett.* **7**, 8502–8509 (2022).
- Kolve, E. *et al.* Ai2-thor: An interactive 3d environment for visual ai. arXiv preprint [arXiv:1712.05474](https://arxiv.org/abs/1712.05474) (2017).
- Xia, F. *et al.* Gibson env: Real-world perception for embodied agents. In *CVPR* (2018).
- Brodeur, S. *et al.* Home: A household multimodal environment. arXiv preprint [arXiv:1711.11017](https://arxiv.org/abs/1711.11017) (2017).
- Savva, M., Chang, A. X., Dosovitskiy, A., Funkhouser, T. & Koltun, V. Minos: Multimodal indoor simulator for navigation in complex environments. arXiv preprint [arXiv:1712.03931](https://arxiv.org/abs/1712.03931) (2017).
- Naveed, K., Hussain, W., Hussain, I., Lee, D. & Anjum, M. L. Help me through: Imitation learning based active view planning to avoid slam tracking failures. *IEEE Trans. Robot.* **41**, 4236–4252. <https://doi.org/10.1109/TRO.2025.3582817> (2025).
- Salas, M. *et al.* Layout aware visual tracking and mapping. In *IROS* (2015).
- Ikram, M. H., Khaliq, S., Anjum, M. L. & Hussain, W. Perceptual aliasing++: Adversarial attack for visual slam front-end and back-end. *IEEE Robot. Autom. Lett.* **7**, 4670–4677 (2022).
- Naveed, K., Anjum, M. L., Hussain, W. & Lee, D. Deep introspective slam: deep reinforcement learning based approach to avoid tracking failure in visual slam. *Autonom. Robot.* 1–20 (2022).
- Kadian, A. *et al.* Sim2real predictivity: Does evaluation in simulation predict real-world performance?. *IEEE Robot. Autom. Lett.* **5**, 6670–6677 (2020).
- Sattler, T. *et al.* Benchmarking 6dof outdoor visual localization in changing conditions. In *CVPR* (2018).
- Maddern, W., Pascoe, G., Linegar, C. & Newman, P. 1 year, 1000 km: The oxford robotcar dataset. *Int. J. Rob. Res.* **36**, 3–15 (2017).
- Torralba, A. & Efros, A. A. Unbiased look at dataset bias. In *CVPR* (2011).
- Milford, M. J., Wyeth, G. F. & Prasser, D. Ratslam: a hippocampal model for simultaneous localization and mapping. In *ICRA* (2004).
- Davison, A. J., Reid, I. D., Molton, N. D. & Stasse, O. Monoslam: Real-time single camera slam. *IEEE T-PAMI* **29**, 1052–1067 (2007).
- Klein, G. & Murray, D. Parallel tracking and mapping on a camera phone. In *ISMAR* (2009).
- Civera, J., Grasa, O. G., Davison, A. J. & Montiel, J. M. 1-point ransac for extended kalman filtering: Application to real-time structure from motion and visual odometry. *J. Field Robot.* **27**, 609–631 (2010).
- Newcombe, R. A., Lovegrove, S. J. & Davison, A. J. Dtam: Dense tracking and mapping in real-time. In *ICCV* (2011).
- Engel, J. & Schops, T. & Cremers, D. (Large-scale direct monocular slam. In *ECCV, Lsd-slam*, 2014).
- Forster, C. & Pizzoli, M. & Scaramuzza, D. (Fast semi-direct monocular visual odometry. In *ICRA, Svo*, 2014).
- Mur-Artal, R., Montiel, J. M. M. & Tardos, J. D. Orb-slam: a versatile and accurate monocular slam system. *IEEE Trans. Robot.* **31**, 1147–1163 (2015).
- Qin, T., Li, P. & Shen, S. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* **34**, 1004–1020 (2018).
- Campos, C., Elvira, R., Rodriguez, J. J. G., Montiel, J. M. & Tardos, J. D. Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam. *IEEE Trans. Robot.* **37**, 1874–1890 (2021).
- Murai, R., Dexheimer, E. & Davison, A. J. Mast3r-slam: Real-time dense slam with 3d reconstruction priors. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 16695–16705 (2025).
- Hoiem, D., Efros, A. A. & Hebert, M. Putting objects in perspective. *IJCV* **80**, 3–15 (2008).
- Hedau, V., Hoiem, D. & Forsyth, D. Recovering the spatial layout of cluttered rooms. In *CVPR* (2009).
- Karsch, K., Hedau, V., Forsyth, D. & Hoiem, D. Rendering synthetic objects into legacy photographs. *ACM Trans. Graph. (TOG)* **30**, 1–12 (2011).
- Gupta, A., Satkin, S., Efros, A. A. & Hebert, M. From 3d scene geometry to human workspace. In *CVPR* (2011).
- Hussain, M. W., Civera, J. & Montano, L. Grounding acoustic echoes in single view geometry estimation. In *AAAI* (2014).
- Khaliq, S., Anjum, M. L., Hussain, W., Khattak, M. U. & Rasool, M. Why orb-slam is missing commonly occurring loop closures?. *Autonom. Robot.* **47**, 1519–1535 (2023).
- Manzoor, R., Hussain, W. & Anjum, M. L. Out of dataset, out of algorithm, out of mind: a critical evaluation of ai bias against disabled people. *AI & SOCIETY* 1–11 (2024).
- Naveed, K., Anjum, M. L., Hussain, W. & Lee, D. Deeper introspective slam: How to avoid tracking failures over longer routes? In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 635–641 (IEEE, 2024).
- Doersch, C., Singh, S., Gupta, A., Sivic, J. & Efros, A. What makes paris look like paris? *ACM Trans. Graph.* **31** (2012).
- Richter, S. R., Al Hajja, H. A. & Koltun, V. Enhancing photorealism enhancement. *IEEE T-PAMI* (2022).
- Mnih, V. *et al.* Playing atari with deep reinforcement learning. arXiv preprint [arXiv:1312.5602](https://arxiv.org/abs/1312.5602) (2013).
- Beattie, C. *et al.* Deepmind lab. arXiv preprint [arXiv:1612.03801](https://arxiv.org/abs/1612.03801) (2016).

55. Krahenbuhl, P. Free supervision from video games. In *CVPR* (2018).
56. Savva, M. *et al.* Habitat: A platform for embodied ai research. In *ICCV* (2019).
57. Pech-Pacheco, J. L., Cristobal, G., Chamorro-Martinez, J. & Fernandez-Valdivia, J. Diatom autofocus in brightfield microscopy: a comparative study. In *Proceedings 15th International Conference on Pattern Recognition (ICPR)*, vol. 3, 314–317 (IEEE, 2000).
58. Chang, A. *et al.* Matterport3d: Learning from rgb-d data in indoor environments. arXiv preprint [arXiv:1709.06158](https://arxiv.org/abs/1709.06158) (2017).
59. Song, S. *et al.* Semantic scene completion from a single depth image. In *CVPR*, 1746–1754 (2017).
60. McPherson, R., Shokri, R. & Shmatikov, V. Defeating image obfuscation with deep learning. arXiv preprint [arXiv:1609.00408](https://arxiv.org/abs/1609.00408) (2016).
61. Potmesil, M. & Chakravarty, I. Modeling motion blur in computer-generated images. *ACM SIGGRAPH Comput. Graph.* **17**, 389–399 (1983).
62. Koenig, N. & Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IROS* (2004).
63. Johnson, M., Hofmann, K., Hutton, T. & Bignell, D. The malmo platform for artificial intelligence experimentation. In *IJCAI*, 4246–4247 (2016).
64. Kempka, M., Wydmuch, M., Runc, G., Toczek, J. & Jaskowski, W. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)* (2016).
65. Straub, J. *et al.* The replica dataset: A digital replica of indoor spaces. arXiv preprint [arXiv:1906.05797](https://arxiv.org/abs/1906.05797) (2019).
66. Ammirato, P., Berg, A. C. & Kosecka, J. Active vision dataset benchmark. In *CVPR Workshops*, 2046–2049 (2018).
67. Satkin, S., Lin, J. & Hebert, M. Data-driven scene understanding from 3d models. *CMU* (2012).
68. Bouguet, J.-Y. Camera calibration toolbox for matlab. *Caltech Vision Group* (2004).
69. Hartley, R. & Zisserman, A. *Multiple view geometry in computer vision* (Cambridge University Press, 2003).
70. Community, B. O. *Blender - a 3D modelling and rendering package* (Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018).
71. Pan, J., Bai, H. & Tang, J. Cascaded deep video deblurring using temporal sharpness prior. In *CVPR* (2020).
72. Mayer, N. *et al.* What makes good synthetic training data for learning disparity and optical flow estimation?. *IJCV* **126**, 942–960 (2018).
73. Lopez-Campos, R. & Martinez-Carranza, J. Espada: Extended synthetic and photogrammetric aerial-image dataset. *IEEE Robotics and Automation Letters* **6**, 7981–7988 (2021).

### Author contributions

M.L. Anjum wrote the original manuscript, W. Hussain conceived the experiment(s) and reviewed draft, U. Muddassar, S.A.H. Bukhari and A.A. Qureshi developed the software and conducted the experiment(s), I. Hussain analyzed the results and arranged funding. All authors reviewed the manuscript.

### Funding

This work was supported by the Khalifa University of Science and Technology under Award CIRA-2021-085, Award No. 8434000534, and Award RC1-2018-KUCARS.

### Declarations

### Competing interests

The authors declare no competing interests.

### Disclaimer

The observations presented here are based solely on the datasets examined in this work. Because many SLAM datasets were not included, the conclusions should be interpreted within this scope. We encourage other researchers to look closely at the datasets that fall outside our selection and to assess whether similar biases or limitations may appear elsewhere.

### Additional information

**Supplementary Information** The online version contains supplementary material available at <https://doi.org/10.1038/s41598-025-32990-z>.

**Correspondence** and requests for materials should be addressed to W.H. or I.H.

**Reprints and permissions information** is available at [www.nature.com/reprints](http://www.nature.com/reprints).

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025