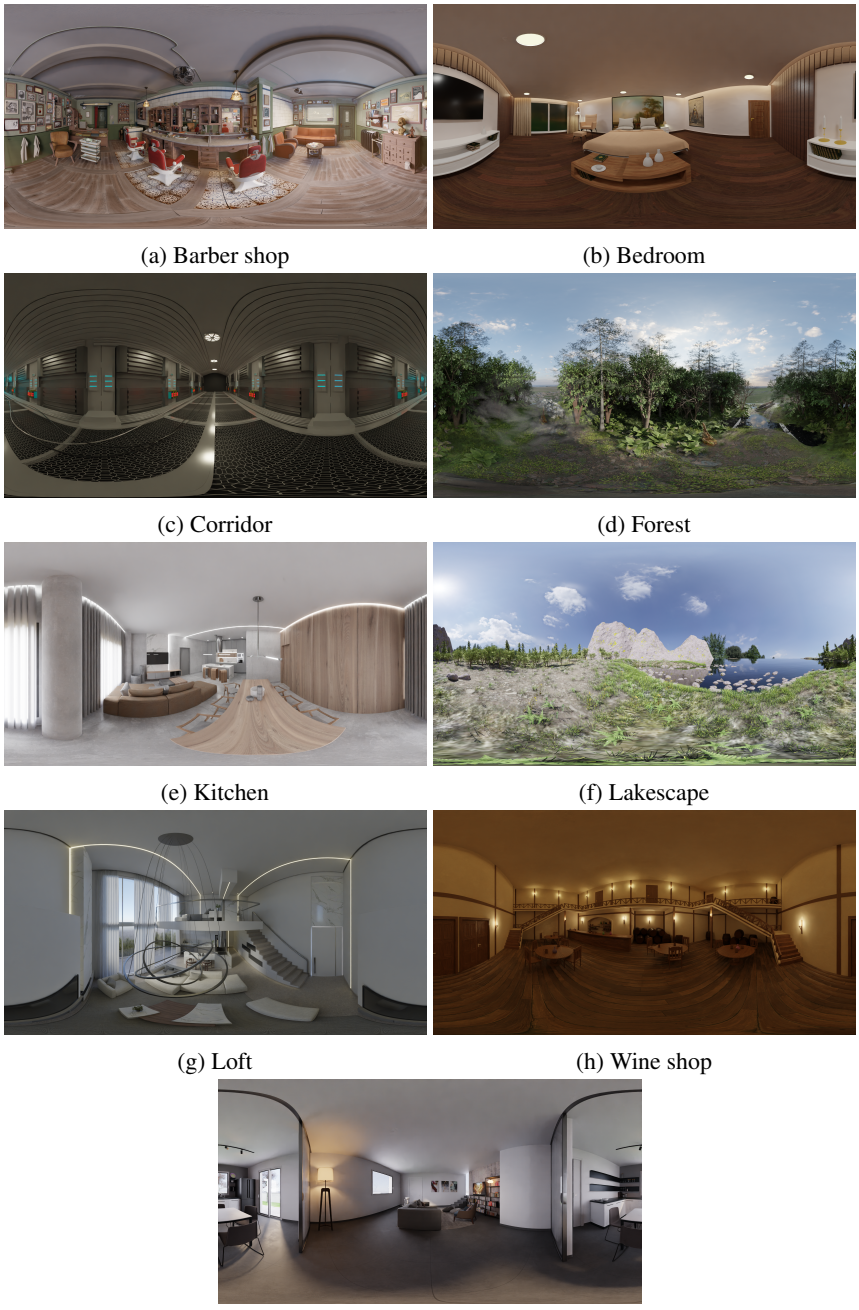
A Visual scenes

A.1 Visualization of the scenes



(a) Barber shop (b) Bedroom

(c) Corridor (d) Forest

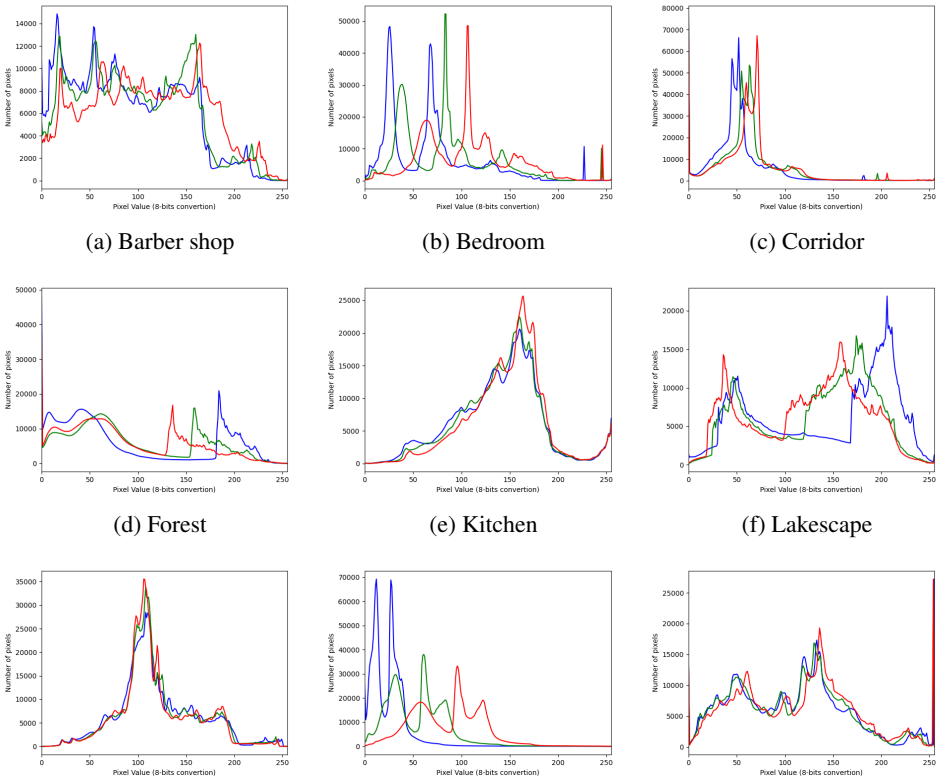
(e) Kitchen (f) Lakescape

(g) Loft (h) Wine shop

(i) Flat

Figure 5: Visualizations of all nine distinct 3D photorealistic scenes used for optical flow truth data generation.

A.2 Histograms of the visual scenes



(a) Barber shop (b) Bedroom (c) Corridor

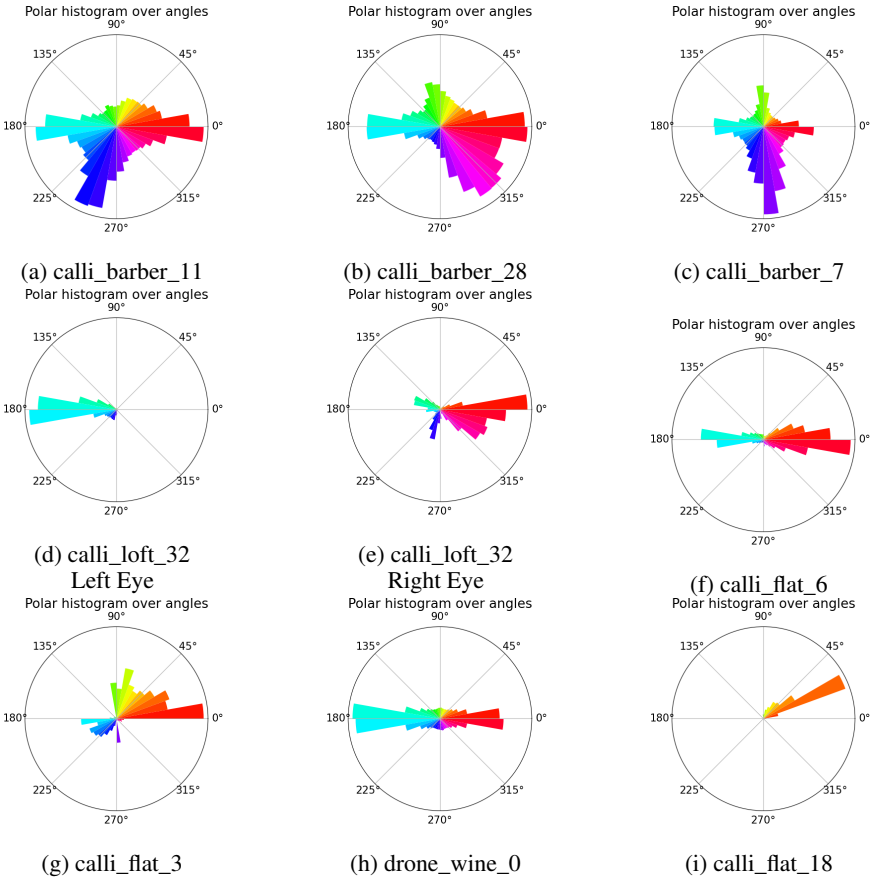
(d) Forest (e) Kitchen (f) Lakescape

(g) Loft (h) Wine shop (i) Flat

Figure 6: Histograms of RGB pixel intensity (converted to 8-bits) of the different scenes used in *FlyView*

B Motion flow

B.1 Kinematic distribution of flight trajectory data



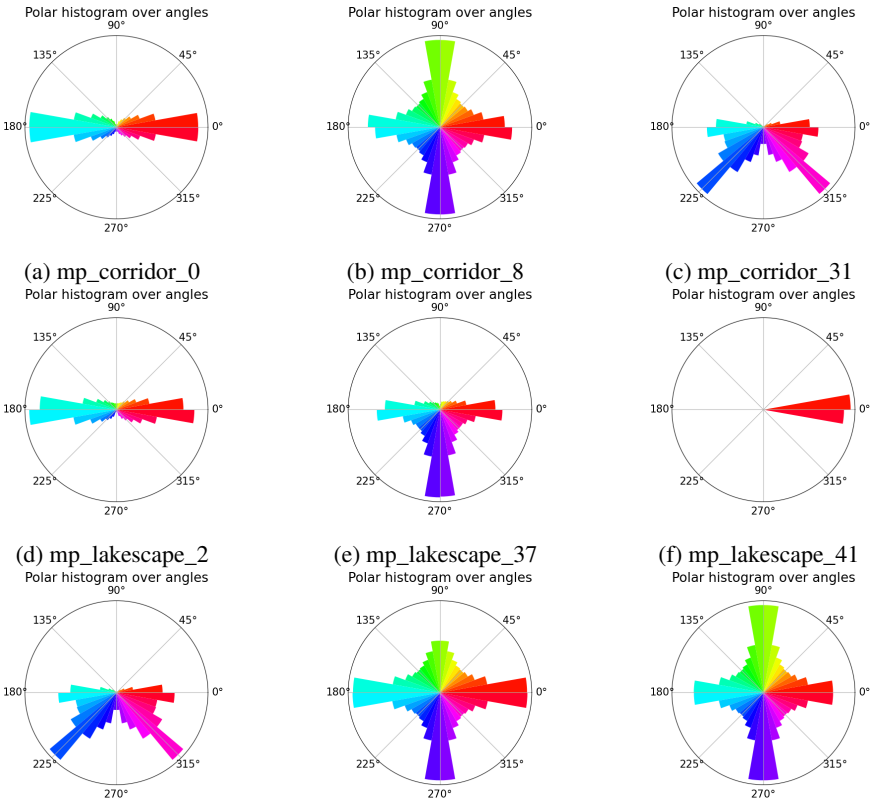
(a) calli_barber_11 (b) calli_barber_28 (c) calli_barber_7

(d) calli_loft_32 Left Eye (e) calli_loft_32 Right Eye (f) calli_flat_6

(g) calli_flat_3 (h) drone_wine_0 (i) calli_flat_18

Figure 7: Summed polar histograms of motion flow direction for *Calliphora* and drone flight trajectories. Figures are identified by the trajectory id in the dataset.

B.2 Kinematic distribution of motion primitives



(a) mp_corridor_0 (b) mp_corridor_8 (c) mp_corridor_31

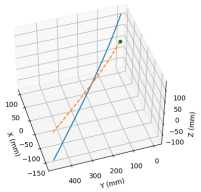
(d) mp_lakescape_2 (e) mp_lakescape_37 (f) mp_lakescape_41

(g) mp_interior_34 (h) mp_interior_54 (i) mp_interior_42

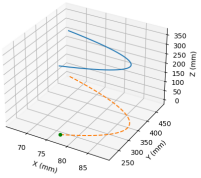
Figure 8: Summed polar histograms of motion flow direction for specific motion primitives. Figures are identified by the trajectory id in the dataset.

B.3 Representative tracks

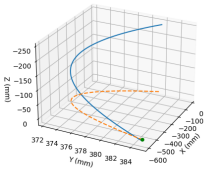
The following section show some representative tracks of the drone and fly trajectories.



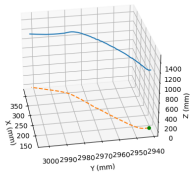
(a) calli_loft_32



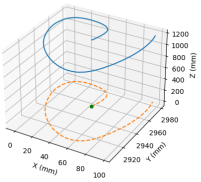
(b) calli_forest_34



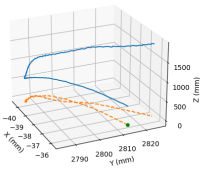
(c) calli_forest_35



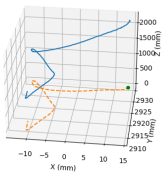
(d) drone_wine_10



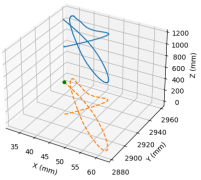
(e) drone_wine_3



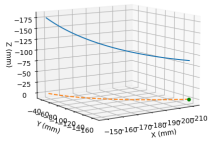
(f) drone_bedroom_19



(g) drone_bedroom_18



(h) drone_wine_4

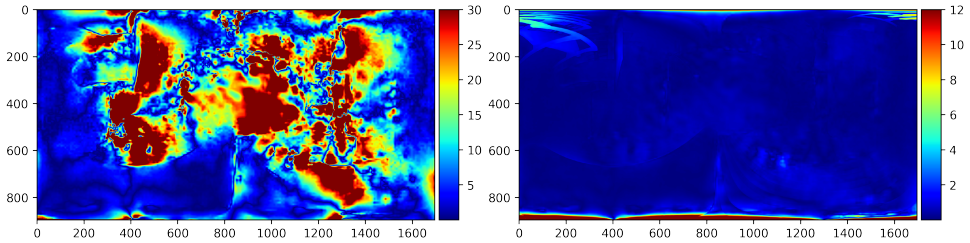


(i) calli_barber_20

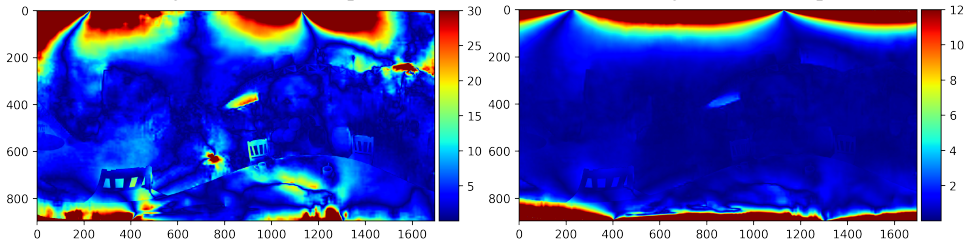
Figure 9: 3D plots of tracks used for the collection of data in *FlyView*. Blue line is the trajectory. Orange dashed line: projection of the trajectory on the ground reference plane; green dot: initial location of the agent.

B.4 Angular and magnitude errors

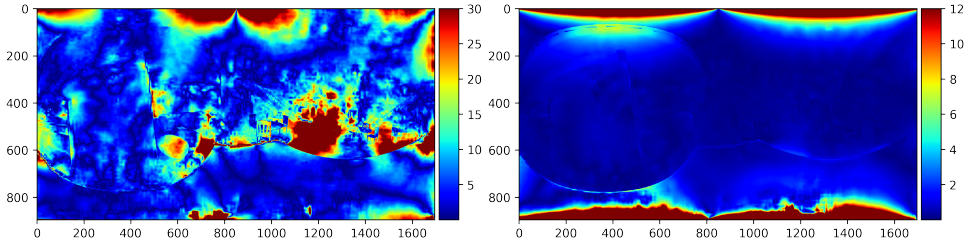
The following section shows angular and magnitude error heatmaps from the pre-trained RAFT-large network evaluated on *FlyView*, highlighting its difficulty in accurately estimating the motion flow in the distorted areas at the top and bottom of the images.



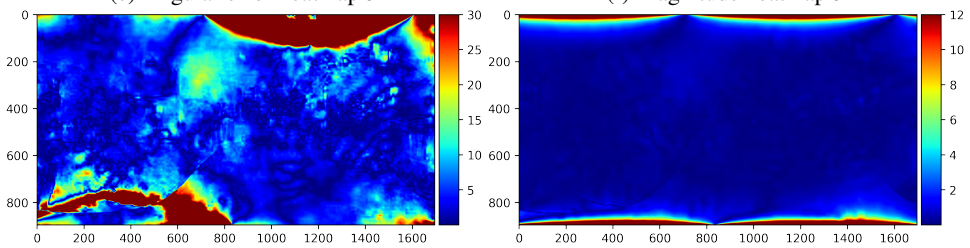
(a) Angular error heatmap 1 (b) Magnitude heatmap 1



(c) Angular error heatmap 2 (d) Magnitude heatmap 2



(e) Angular error heatmap 3 (f) Magnitude heatmap 3



(g) Angular error heatmap 4 (h) Magnitude heatmap 4

Figure 10: Heatmaps showing error in motion flow estimation by the pre-trained RAFT-large network tested on *FlyView*. Heatmaps use red to denote angular errors $> 25°$ and magnitude errors > 10 px.

C Dataset and subsets

C.1 Content of the dataset

The *FlyView* dataset includes the visual images, depth map, forward and backward motion flow maps, and pose of the three cameras for each of the 42,472 frames generated. *FlyView* provides the intrinsic and extrinsic parameters of the camera system. The intrinsic parameters are generated using the Scaramuzza camera model [42]. In addition, *FlyView* comes with images of a calibration board allowing the user to compute the intrinsic parameters of each individual camera using a different large field-of-view camera model, such as the Mei model [43] or Kannala-Brandt model [44]. Masks representing the edges of the visual field of *Drosophila* are provided for the panoramic view, allowing the user to select the view of either the left or right compound eye. The dataset is self-contained and will be made fully accessible upon publication of this paper, through links provided in our GitHub repository. It is supplemented by real video data from a pair of fisheye cameras undergoing linear motion in an indoor lab environment, for which only self-motion truth data are provided.

C.2 Description of the motions

Table 3: Summary of trajectories within our virtual dataset. Here, R stands for pure rotational motion, T stands for pure translational motion, and $R + T$ stands for a combination of both.

Trajectories	**Type of motion**	**Number of trajectories**	**Number of frames**	**Frames per second**
motion primitives	R, T	55	13,750	25
DJI Tello drone	$R, T, R + T$	20	16,025	200
blowfly *Calliphora*	$R + T$	36	12,700	16,620

C.3 Description of scenarios

Table 4: Description of the different scenarios encountered within the dataset. MP: motion primitive; R: rotation; T: translation.

Subsets	**Motion type**	**Scene**	**Number of trajectories**	**Total Frames**	**Indoor/ Outdoor**	**Bright/ Dark**	**Motion description**	**Scale of apparent displacement**
S1	MP	Lakescape	14	3500	Outdoor	Bright	R, T	Large
S1‘	*Calliphora*	Lakescape	11	4000	Outdoor	Bright	R+T	Small
S2	MP	Forest	13	3250	Outdoor	Bright	R, T	Large
S2‘	*Calliphora*	Forest	3	1500	Outdoor	Bright	R+T	Small
S3	MP	Interior	14	3500	Indoor	Bright	R, T	Large
S4	MP	Corridor	14	3500	Indoor	Dark	R, T	Large
S5	Drone	Wine shop	12	12511	Indoor	Dark	R+T, R, T	Small
S6	Drone	Bedroom	8	3514	Indoor	Bright	R+T, R, T	Small
S7	*Calliphora*	Flat	12	3300	Indoor	Bright	R+T	Small
S8	*Calliphora*	Barber shop	9	2900	Indoor	Bright	R+T	Small
S9	*Calliphora*	Loft	1	1000	Indoor	Bright	R+T	Small
S10	MP	3D cubes	1	20	Outdoor	Dark	T	Large

C.4 Recommended split

Datasets for machine learning applications are usually split into three separate sets each dedicated to a specific stage of the training strategy, comprising training, validation, and test data.

We suggest below a split of the data with respect to the different subsets available, which leads to 29,925 frames for the training subset, 10,050 frames for the validation subset, and 2,500 frames for the test subset.

- Train : 29,925 frames (73.1%)
  - S1 + S1’
  - S3

  - S5
  - S6
  - S8
- Validation : 10,050 frames (23.6%)
  - S2
  - S4
  - S7
- Test : 2,500 frames (5.9%)
  - S9
  - S2'

Note that dataset S10 comprising 3D cubes is not included within any of these subsets, as it does not represent a realistic flight scene, and is shared instead as a dataset that can easily be reproduced by other users.

C.5 List of trajectories

The following tables list the different trajectories used in *FlyView*.

Motion primitives

Table 5: Description of the different motion primitives generated. R: rotation; T: translation.

ID	Subset	Name	Motion type	Number of frames
0	S4	mp_corridor_0	T	250
1	S2	mp_forest_1	R	250
2	S1	mp_lakescape_2	T	250
3	S3	mp_kitchen_3	T	250
4	S4	mp_corridor_4	T	250
5	S2	mp_forest_5	T	250
6	S1	mp_lakescape_6	T	250
7	S3	mp_kitchen_7	R	250
8	S4	mp_corridor_8	R	250
9	S2	mp_forest_9	T	250
10	S1	mp_lakescape_10	T	250
11	S3	mp_kitchen_11	T	250
12	S4	mp_corridor_12	T	250
13	S1	mp_lakescape_13	R	250
14	S3	mp_kitchen_14	T	250
15	S4	mp_corridor_15	T	250
16	S2	mp_forest_16	T	250
17	S1	mp_lakescape_17	T	250
18	S3	mp_kitchen_18	T	250
19	S4	mp_corridor_19	T	250
20	S2	mp_forest_20	R	250
21	S1	mp_lakescape_21	T	250
22	S3	mp_kitchen_22	T	250
23	S4	mp_corridor_23	T	250
24	S2	mp_forest_24	T	250
25	S1	mp_lakescape_25	T	250
26	S3	mp_kitchen_26	R	250
27	S4	mp_corridor_27	R	250
28	S2	mp_forest_28	R	250
29	S1	mp_lakescape_29	T	250
30	S3	mp_kitchen_30	T	250
31	S4	mp_corridor_31	T	250
32	S2	mp_forest_32	T	250
33	S1	mp_lakescape_33	T	250
34	S3	mp_kitchen_34	T	250
35	S4	mp_corridor_35	R	250
36	S2	mp_forest_36	R	250
37	S1	mp_lakescape_37	R	250
38	S3	mp_kitchen_38	R	250
39	S4	mp_corridor_39	R	250
40	S2	mp_forest_40	R	250
41	S1	mp_lakescape_41	R	250
42	S3	mp_kitchen_42	R	250
43	S4	mp_corridor_43	R	250
44	S2	mp_forest_44	R	250
45	S1	mp_lakescape_45	R	250
46	S3	mp_kitchen_46	R	250
47	S4	mp_corridor_47	R	250
48	S2	mp_forest_48	R	250
49	S1	mp_lakescape_49	R	250
50	S3	mp_kitchen_50	R	250
51	S4	mp_corridor_51	R	250
52	S2	mp_forest_52	R	250
53	S1	mp_lakescape_53	R	250
54	S3	mp_kitchen_54	R	250

Drone trajectories

Table 6: Description of the different *Calliphora* trajectories used in FlyView. R: rotation; T: translation.

ID	Subset	Name	Motion type	Number of frames
0	S5	drone_wine_0	R+T	5,387
1	S5	drone_wine_1	R+T	580
2	S5	drone_wine_2	R+T	499
3	S5	drone_wine_3	R+T	511
4	S5	drone_wine_4	R+T	525
5	S5	drone_wine_5	R+T	521
6	S5	drone_wine_6	R+T	839
7	S5	drone_wine_7	R+T	968
8	S5	drone_wine_8	R+T	701
9	S5	drone_wine_9	R+T	1,449
10	S5	drone_wine_10	R+T	117
11	S6	drone_bedroom_11	R+T	119
12	S6	drone_bedroom_12	R+T	714
13	S6	drone_bedroom_13	R+T	365
14	S6	drone_bedroom_14	R+T	435
15	S6	drone_bedroom_15	R+T	914
16	S5	drone_wine_16	R+T	414
17	S6	drone_bedroom_17	R+T	361
18	S6	drone_bedroom_18	R+T	373
19	S6	drone_bedroom_19	R+T	233

Calliphora trajectories

Table 7: Description of the different *Calliphora* trajectories used in FlyView. R: rotation; T: translation.

ID	Subset	Name	Motion type	Number of frames
0	S7	calli_flat_0	R+T	400
1	S7	calli_flat_1	R+T	200
2	S7	calli_flat_2	R+T	400
3	S7	calli_flat_3	R+T	100
4	S7	calli_flat_4	R+T	100
5	S7	calli_flat_5	R+T	100
6	S7	calli_flat_6	R+T	400
7	S8	calli_barber_7	R+T	400
8	S8	calli_barber_8	R+T	200
9	S7	calli_flat_9	R+T	200
10	S7	calli_flat_10	R+T	200
11	S8	calli_barber_11	R+T	300
12	S1'	calli_lakescape_12	R+T	400
13	S1'	calli_lakescape_13	R+T	400
14	S1'	calli_lakescape_14	R+T	400
15	S7	calli_flat_15	R+T	400
16	S7	calli_flat_16	R+T	400
17	S1'	calli_lakescape_17	R+T	400
18	S7	calli_flat_18	R+T	400
19	S1'	calli_lakescape_19	R+T	400
20	S8	calli_barber_20	R+T	400
21	S1'	calli_lakescape_21	R+T	400
22	S8	calli_barber_22	R+T	400
23	S1'	calli_lakescape_23	R+T	400
24	S8	calli_barber_24	R+T	300
25	S1'	calli_lakescape_25	R+T	300
26	S8	calli_barber_26	R+T	300
27	S1'	calli_lakescape_27	R+T	300
28	S8	calli_barber_28	R+T	300
29	S1'	calli_lakescape_29	R+T	300
30	S8	calli_barber_30	R+T	300
31	S1'	calli_lakescape_31	R+T	300
32	S9	calli_loft_32	R+T	1000
33	S2'	calli_forest_33	R+T	500
34	S2'	calli_forest_34	R+T	500
35	S2'	calli_forest_35	R+T	500

D 3D assets

This section describes where all 3D assets used in the *FlyView* dataset can be found, and the license under which they were released. Most 3D assets being made by Blender artists, we do not provide the original Blender files used in our data generation. However, the original files can be downloaded from the links to the artists' pages that are given below. The Sample scene provides Blender files containing the camera definition and a minimal scene, to aid understanding of how the optical flow truth data was generated.

Barber shop

- Author : blender.org
- Original asset : https://svn.blender.org/svnroot/bf-blender/trunk/lib/benchmarks/cycles/barbershop_interior/
- License : CC-BY

Bedroom

- Author : https://www.cgtrader.com/harshitverma1308
- Original asset : https://www.cgtrader.com/free-3d-models/interior/bedroom/bedroom-ca552be4-08d8-427b-8f5a-8391962983fe
- License : Royalty Free License

Corridor

- Author : https://www.cgtrader.com/iamcyberalex
- Original asset : https://www.cgtrader.com/free-3d-models/space/spaceship/dead-space-sci-fi-corridor-by-cyberalex
- License : Royalty Free License

Forest

- Author : https://www.cgtrader.com/ankitsarkar
- Original asset : https://www.cgtrader.com/free-3d-models/scanned/various/realistic-forest-model-for-blender
- License : Royalty Free License

Kitchen

- Author : https://www.cgtrader.com/jpartsky
- Original asset : https://www.cgtrader.com/free-3d-models/interior/living-room/interior-b
- License : Royalty Free License

Lakescape

- Author : https://www.cgtrader.com/mohdarbaaz3
- Original asset : https://www.cgtrader.com/3d-models/exterior/landscape/scene-lake
- License : Royalty Free License
- Price paid : $4.80

Loft

- Author : https://www.cgtrader.com/jpartsky
- Original asset : https://www.cgtrader.com/free-3d-models/interior/living-room/project-b-54e79cb8-6763-471e-9d42-1e7e6cf01e14
- License : Royalty Free License

Wine shop

- Author : https://www.cgtrader.com/harshitverma1308
- Original asset : https://www.cgtrader.com/free-3d-models/interior/other/old-wine-shop
- License : Royalty Free License

Flat

- Author : Flavio Della Tommasa
- Original asset : https://download.blender.org/demo/cycles/flat-archiviz.blend
- License : CC-BY

Sample

The example scene was homemade with basic Blender items. This scene is fully available online and is intended to aid future users in understanding the data generation and the setup of the camera system. The sample scene is provided by the University of Oxford and released under the Creative Commons Attribution-NonCommercial-ShareAlike license (CC BY-NC-SA).

- Author : Alix Leroy
- Original asset : https://github.com/Ahleroy/FlyView
- License : CC BY-NC-SA

E Camera and positioning

E.1 μCT scan measurement

We used a μCT scan of an female adult *Calliphora* to measure the baseline between the two eyes. Whilst the notion of a stereo baseline is well defined for two camera sensors, it is ambiguous for a pair of compound eyes, as each of the several hundred to tens of thousands of facets is a lens. We therefore decided to measure the two most distant points of the eyes from the anteroposterior axis anatomically. The measured distance was 3.51 mm which was rounded to 4 mm in Blender.

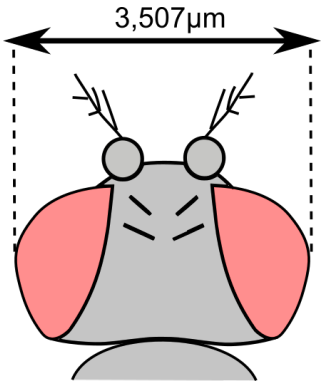


Figure 11: Measured baseline distance between the compound eyes of a female adult *Calliphora*.

E.2 Camera setup comparison

The equirectangular camera model allowed us reconstruct a virtual stereo camera setup that reflects the distribution of optical axes in *Drosophila* [34].

Table 8: Comparison of the camera setup and distribution of the optical axes in *Drosophila*.

	Virtual camera setup	***Drosophila***
Monocular horizontal field of view	180°	Up to 180°
Monocular vertical field of view	180°	Up to 180 °
Binocular overlap	20°	approximately 20°
Total Horizontal field of view	340°	Up to 340°
Sensor resolution	900×900 px	750 ommatidia per eye
Horizontal angular sampling density	0.2°/ px	5°/ ommatidia
Vertical angular sampling density	0.2°/ px	5°/ ommatidia
Baseline	4 mm	<1 mm

The sensor resolution is larger than the number of photoreceptors in each compound eye of *Drosophila* allowing to study a large spectrum of insect species. The baseline is based on *Calliphora* measurements but stereo disparity is only relevant below $1m$ at the maximum resolution. It

F Accuracy of the motion capture system

A Vicon Vantage motion capture system was used to track spherical retroreflective markers in a volume of $20 \times 6 \times 4$ m. A total of 22 VICON Vantage V16 cameras are set up in such a way that their field of view covers the entire volume in which the markers need to be detected. These cameras use LED strobes to emit pulsed near-infrared light (850 nm), and have an image sensor probing the reflection of the emitted wavelength from surfaces.

Objects that are intended to be tracked, such as the DJI Tello drone, are equipped with markers highlighting their position to the Vicon system. The Vicon system is calibrated by a consensus over thousands of calibration points captured in space that allows the relative position of each camera to be identified in the global system. The accuracy with which a trajectory can be collected with the motion capture system depends on both the intrinsic accuracy of pose estimation system and its ability to detect the markers in the first place.

F.1 Detection of markers

Initial data collection of drone trajectories revealed certain difficulties in detecting and localizing the different markers on the drone body. There are two main reasons for this. Firstly, the markers used are the smallest spherical markers available for the motion capture system (only 4 mm diameter), reducing the capability to detect the markers at large distances. Secondly, because the main body of the drone is small, the markers were fixed relatively close to each other. This led the system to fuse one marker's position with another's and to erroneously identify a phantom marker in between the two.

To improve the accuracy of the system, larger markers (spheres of 9 mm diameter) were then attached to the drone. Because the new markers were larger, only two of them could be fixed on the main body. Whilst the drone's body surface could accommodate more markers, additional markers would risk collision with the rotary blades of the drone. These new markers were successfully fixed on both the main body and the protective structure surrounding the rotors. This new setup resulted in a more robust and stable detection of the markers, allowing us to track the drone at higher speeds and even during extreme maneuvers.

F.2 Evaluating the intrinsic accuracy of the motion capture system

In order to evaluate the accuracy of our motion capture system, we compared the outputs of the motion capture system with a calibrated Direct-Drive Linear Actuator ACT140DL-1500 controlled by an Aerotech Automation 3200 whose expected accuracy after calibration is ±5µm. To do so, we first expressed their outputs in the same reference frame.

Since the motion capture system outputs a 3D signal, whereas the linear motors output a 1D signal, we assumed that the cameras were moving on a straight line. We therefore computed a 3D linear

regression on the motion capture system points using a RANSAC algorithm to remove outliers. The result is a 3D vector giving us the 3D direction of the linear motors in the motion capture frame.

For a 0.5 mm residual threshold, 100% of the points are considered inliers by the RANSAC system. For a 0.1 mm residual threshold, 66% of the points are considered as inliers. This is already a good indication that the motion capture system has accurate detection, at least on axes orthogonal to the linear motor's direction.

Once the 3D direction was found, we projected all the points collected by the motion capture system onto the X-axis ($[1, 0, 0]$ in the motion capture frame). To do so, we calculated the rotation matrix from the 3D direction to the X-axis using Rodriguez's formula. The final result is a motion capture signal aligned to a single axis, similar to the signal reported from the linear motors.

The two systems were not synchronized together, and the acquisition of the linear motors signal was started with a delay of a few seconds. Before estimating the time shift between the two signals, we had to resample the motion capture signal (Figure 12 a), since the linear motors' sampling frequency was about 1000Hz (Figure 12 b), whereas the motion capture system's frequency was about 200Hz.

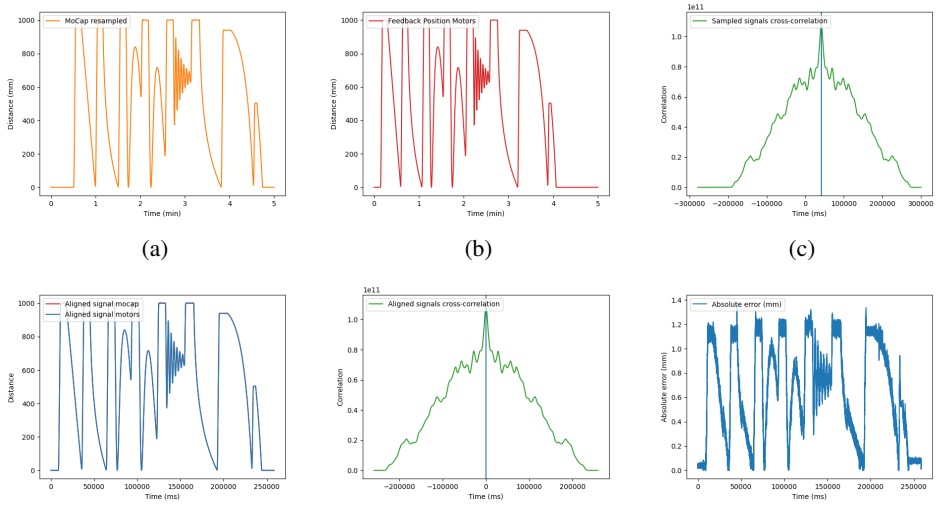


Figure 12: (a) The resampled signal of the motion capture system. (b) The feedback position of the linear motors. (c) The cross-correlation of the two signals. (d) The two signals aligned, showing an almost perfect match. (e) The cross-correlation of the aligned signals. (f) The absolute error between the two signals.

Once the signal was resampled, we computed the cross-correlation between the two signals. We found the lag between the signals (Figure 12 c), and were able to shift the signals and remove unshared parts (motion capture system starts acquisition before the linear motors and the acquisition of the motors finishes after the motion capture). We found that the aligned signals match almost perfectly (Figure 12 d).

F.2.1 Errors between the two signals

The Pearson correlation coefficient between the two signals is > 0.9999, indicating an almost perfect correlation between the two signals. Results given in Table 9 confirm the very high accuracy of the motion capture system.

Table 9: Differences between the two systems' signals

Metric	Value
Maximum (mm)	1.797
Minimum (mm)	8.518e-07
Average (mm)	0.897
Standard Deviation (mm)	0.553
Median (mm)	0.937
First Quartile (mm)	0.417
Last Quartile (mm)	1.474
Pearson correlation coefficient	> 0.9999

The absolute distance between the signals was found to be proportional to the distance travelled. The further from the origin point, the larger the difference. This highlighted the need for a new factory calibration of the linear motors.

G Applications of FlyView

As a bio-informed dataset, *FlyView* is intended to be useful to the study of biological and computer vision alike. It therefore represents the panoramic sampling of a fly's hexagonal imaging array using rectangular images and a square pixel array. A lifelike analysis of the fly's motion vision system can in principle be implemented through a second stage of data processing. For instance, the luminance signals resulting from a hexagonal array of photoreceptors can be approximated by filtering the images appropriately, and processing the regions of interest using masks outlining the field of view of the fly's compound eyes (See Figure 1a). This would enable the *FlyView* dataset to be used in biological analysis of the fly's visual system and its response to luminance signals. This could be used to advance our understanding of how these signals propagate within the neural circuits of the visual system, and particularly how this enables detection of specific patterns of optical flow when navigating.

Nevertheless, because deep learning techniques boast state-of-the-art results in motion flow estimation, the *FlyView* dataset was primarily designed with machine learning applications in mind. A key application of the dataset is for autonomous vehicles equipped with omnidirectional cameras. Motion flow estimation in this case requires the processing of panoramic images obtained in a wide range of motions including those involved in flight, which *FlyView* provides. For example, continuing efforts to miniaturise aerial drone platforms aim to limit sensor payload to allocate more weight and volume to batteries, thereby offering enhanced autonomy and operating range. Such micro-drones could use large field of view cameras to navigate in complex GPS-denied environments without the use of LiDAR, in applications ranging from agriculture and maintenance, to monitoring and reconnaissance. *FlyView* could serve as an initial training ground for such systems in both indoor and outdoor environments.

H Real data for testing

Whilst *FlyView* contains only synthetic data, real data could also be interesting for testing scenarios. As mentioned in Section 1, collecting accurate optical flow ground truth can be challenging with natural data. Existing natural datasets such as Kitti [14, 15] were able to extract only sparse motion flow ground truth, which can nevertheless be useful provided that there is no requirement for sub-pixel optical flow accuracy in the end-application. A similar method was in fact used in our first attempt to collect data for *FlyView*, but because of multiple bottlenecks, we pivoted from collecting sparse natural data to a dense and accurate synthetic dataset.

This initial data collection involved tracking a pair of fisheye cameras with the MoCap system described above. Whilst the resulting data does not contain any motion flow ground truth, it can still be used to validate models trained on large field-of-view images that aim to estimate self-motion, where the detection of optical flow represents an intermediate stage in the processing rather than an end in itself. For this reason, we have released videos of the camera pair and the corresponding motion state of the camera system as a supplement to *FlyView*. These experimental data will be made available on the GitHub repository at the same time as the rest of the *FlyView* dataset. The following sections provide more information on the context and the content of the real dataset.

H.1 Initial data collection

We collected real data in the lab, using a divergent stereo vision system composed of a pair of fisheye cameras with submillimetre tracking of the cameras using the same MoCap system presented elsewhere in this paper. These real data were collected using two 8 mm focal length lenses that output images different to the equirectangular projection used in our synthetic dataset *FlyView*. Figure 13 shows the camera setup used during our initial data collection.

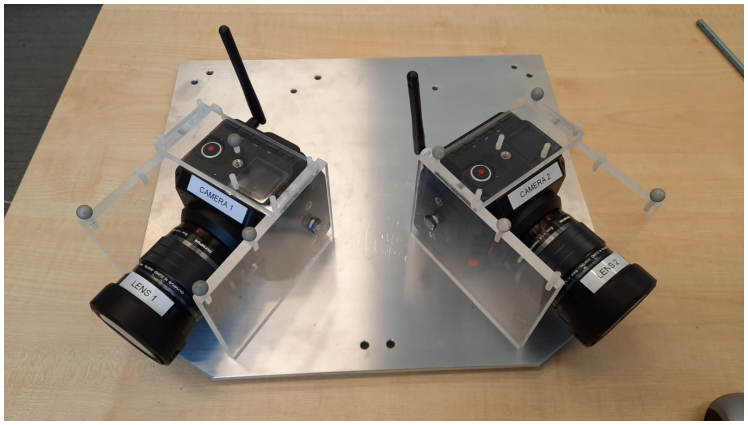


Figure 13: Camera setup fixed on an aluminium baseplate to ensure fixed and repeatable positioning of the two cameras.

The real cameras were calibrated using the Scaramuzza camera model [42] that we then integrated into a custom-built version of Blender to reproduce the camera model with the ray-tracing system of the Blender's Cycles engine. Whilst the camera model was integrated with CUDA and OptiX-enabled kernels, the resulting ground truth generation was slow and potentially inaccurate, especially around the optical axis where a strong flow singularity was observed. This singularity in the optical flow field could be avoided by augmenting the order of the Scaramuzza polynomial function used to re-project the world onto the sensor, but not without having a severe impact on the computational time. For this reason, we decided to switch to a pure synthetic dataset with an equirectangular camera.

H.2 Content of the dataset

The stereo vision system was fixed to a linear actuator (See Figure 14). A broader range of motions was initially planned, but was abandoned for the reasons explained above.

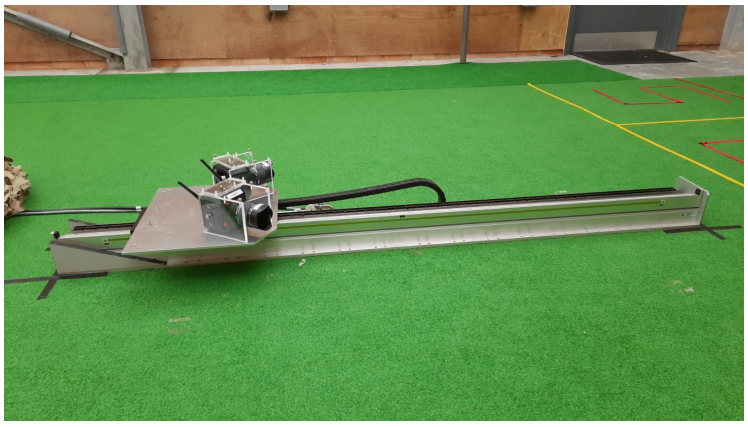


Figure 14: Camera setup fixed on the linear actuator.

For these translational motions, a total of 9 sessions of trajectories were generated. Each session consisted of 5 minutes of motion back and forth along a single axis with different kinematics. Each session was performed 3 times, once with an empty environment and twice with visible rigid and static obstacles positioned randomly. Having different camera views for the exact same motion allows the generation of a wide range of optical flow patterns compatible with a unique relative pose. The obstacles consisted of pipes that were painted with dark or bright colors resulting in different pixel brightness and features to be displayed in a single image.

Figure 15: View from a camera with multiple obstacles facing it.

All sessions were captured using the motion capture system to within submillimetre accuracy.

The data resulting from these sessions includes the RAW visual images for both cameras captured at a 30 Hz frame-rate, the position of the linear motors commanded for each trajectory, and the tracked position of the camera-setup captured by the motion capture system.