

Virtual reality for animal navigation with camera-based optical flow tracking

Ivan Vishniakou^a, Paul G. Plöger^b, Johannes D. Seelig^{a,*}

^a*Center of Advanced European Studies and Research (caesar), Bonn, Germany*

^b*Department of Computer Science, Hochschule Bonn-Rhein-Sieg, Sankt Augustin, Germany*

Abstract

Background Virtual reality combined with spherical treadmills is used across species for studying neural circuits underlying navigation.

New Method We developed an optical flow-based method for tracking treadmill ball motion in real-time using a single high-resolution camera.

Results Tracking accuracy and timing were determined using calibration data. Ball tracking was performed at 500 Hz and integrated with an open source game engine for virtual reality projection. The projection was updated at 120 Hz with a latency with respect to ball motion of 30 ± 8 ms.

Comparison with Existing Method(s) Optical flow based tracking of treadmill motion is typically achieved using optical mice. The camera-based optical flow tracking system developed here is based on off-the-shelf components and offers control over the image acquisition and processing parameters. This results in flexibility with respect to tracking conditions - such as ball surface texture, lighting conditions, or ball size - as well as camera alignment and calibration.

Conclusions A fast system for rotational ball motion tracking suitable for virtual reality animal behavior across different scales was developed and characterized.

Keywords: Virtual Reality, Navigation, Optical Flow, Spherical Treadmill, Ball Tracking, Drosophila, Real-Time Image Processing

*johannes.seelig@caesar.de

1 1. Introduction

2 Virtual reality (VR) is used across species for studying neural circuits
3 underlying behavior [1]. In many implementations, animals navigate through
4 virtual realities on a spherical treadmill - a ball which can be freely rotated
5 around its center of mass [1, 2, 3, 4, 5].

6 Tracking of ball rotation is typically accomplished using optical mice
7 which are based on low-resolution, high-speed cameras integrated with a light
8 source for measuring displacements when moving across a surface. Movement
9 across the surface results in optical flow - the displacement of features across
10 the camera sensor. Such features can for example be speckle-like reflections
11 from surface roughness; comparing these speckle images between different
12 frames then allows computing the displacement using hardware-integrated
13 image processing.

14 Optical mice come however with some limitations for measuring ball ro-
15 tation. First, a single optical mouse measures displacements only in two
16 directions and therefore two mice are required for tracking all three degrees
17 of freedom of ball motion. Secondly, limited or no control over the onboard
18 processing algorithms as well as camera settings requires careful calibration.
19 In particular, if the mouse sensors can't be placed in direct proximity of
20 the ball surface, accurate alignment of the two sensors as well as calibration
21 with respect to surface properties and lighting conditions is necessary [5].
22 As an approach that overcomes some of these limitations, real-time track-
23 ing was developed with a single high-resolution camera for situations where a
24 uniquely patterned ball can be used [6]. In that case, ball orientation was cal-
25 culated by matching each recorded frame to a map of the entire ball surface
26 pattern. Using a high-resolution cameras allows control over all recording
27 parameters and offers the freedom to choose a custom algorithms and test
28 its performance with simulated data [6].

29 Here, we combine the flexibility of optical flow-based tracking with the
30 advantages of using a high-resolution camera. The developed system tracks
31 optical flow of rotational ball movement at 500 Hz using a single camera
32 and is integrated with an open source virtual reality environment and two
33 projectors [7] with a refresh rate of 120 Hz [8] and overall latency of 30 ± 8
34 ms - from detecting ball movement to projecting the updated virtual reality
35 image. This is achieved using off-the-shelf hardware components and open
36 source software. The system is easy to align and calibrate, can be used
37 under different conditions and at different scales, and can be integrated with

38 two-photon imaging or electrophysiology.

39 **2. Materials and Methods**

40 *2.1. Virtual reality setup components*

41 A schematic of the setup is shown in Figure 1. Two digital micromirror
42 devices (DMD, DLP LightCrafter 6500 by Texas Instruments) were used for
43 projecting the VR onto an angled screen from two sides [8, 7]. The projectors
44 are FullHD digital micromirror devices with HDMI and DisplayPort inter-
45 faces allowing them to display images the same way as standard monitors.
46 In our setup, the DMD was set to use DisplayPort input and display the
47 frames at a maximum frame rate of 120 Hz (see Appendix for details). This
48 frame rate can only be achieved when frames are displayed with bit depth
49 1 (binary images). To display different gray levels, a dithering technique is
50 used and the density of white pixels is varied proportionally to the brightness
51 level of the displayed region with an ordered Bayer matrix [9] (see Appendix
52 for details). Two collimated blue LEDs (M470L3, Thorlabs) were used as
53 light sources. Light reflected off the DMD was projected from the outside
54 (seen from the fly’s perspective) onto a screen made from black paper.

55 The spherical treadmill used for the current experiments is described in
56 [5]. Briefly, a 6 mm diameter polyurethane foam ball is held in a cup-shaped
57 holder and suspended in an air stream. The ball is illuminated with two
58 IR LEDs and monitored with a camera. A *Basler acA640-750um* camera
59 was used for tracking. The camera uses a USB 3 interface to communicate
60 with the PC and the bandwidth of 350 MB/s allows transfer of full-frame
61 images ($640 \times 480 \times 8\text{bpp}$ mono color) at a rate of 751 frames per second
62 (fps). For tracking, a frame rate of 500 fps was used, with an exposure
63 time of 100 microseconds under infrared illumination. The resolution was
64 224×140 pixels which results from cropping and 2×2 binning of the full
65 frame. Binning (summing the signal from adjacent pixels into a single one)
66 increased signal-to-noise ratio and reduced the image readout and transfer
67 time. A Computar camera objective M2514-MP2 F1.4, $f = 25\text{mm}$ together
68 with a Computar EX2C extender were used for imaging the ball [5].

69 *2.2. Integration with virtual reality*

70 For virtual reality applications a 3D rendering game engine (Urho3D, an
71 open-source C++ game engine [10]) was used and the x, y- and z-rotations of
72 the ball were mapped to planar motion (forward-, sidestepping- and turning

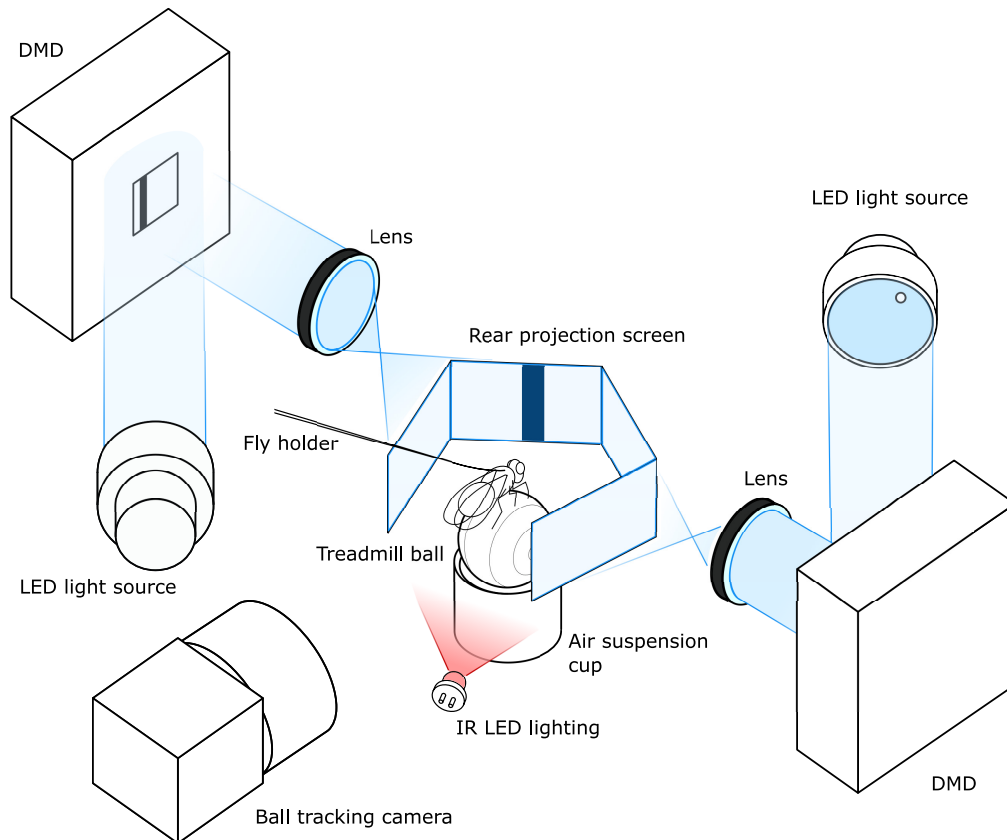


Figure 1: Schematic of behavior setup for *Drosophila melanogaster*. A 6-mm polyurethane ball is air-suspended in a holder and serves as an omnidirectional treadmill. The fly is glued to a thin metal wire. A rear projection screen is used for displaying the virtual reality (VR) in the fly's field of view. Two DMDs, LED light sources, and two projection lenses are combined to project from two sides onto the screen. A single tracking camera is set up to capture ball images at the rate of 500 fps. Infrared LED lighting invisible to the fly is used for illuminating the ball.

73 components) of the animal in the virtual reality (this was done similarly to
74 [6], see Appendix for details).

75 The application is designed to run ball tracking and environment rendering
76 in two separate execution threads. The image processing thread is a loop
77 acquiring camera frames and calculating ball displacements, tuned to run
78 one iteration in under 2 ms, resulting in a tracking frequency of 500 fps. The
79 rendering thread runs the game engine, which models a virtual environment
80 in 3D.

81 Setting up the virtual environment requires specifying the arena geometry
82 as well as lighting and rendering properties. Further, the virtual cameras
83 have to be configured such that the virtual environment is rendered from the
84 angles corresponding to the position of the screen. Additional parameters are
85 required to describe the mapping of ball rotations to movement in the virtual
86 environment; these transformations take into account the size of the ball and
87 the orientation of the tracking camera relative to the forward direction of
88 the animal. This is all done with the in-built Urho3D engine's scripting
89 language which also describes each environment in a separate script file,
90 containing instructions about which objects are to be created in the scene.
91 In each update of the game engine the ball tracking signal is interpreted as the
92 displacement of the animal, and the virtual camera positions and orientations
93 are updated before being rendered and displayed on the screen.

94 Other game engine capabilities used in the application are dithering shader
95 to output binary frames suitable for the DMD and networking to broadcast
96 the state of the VR to any client, for example for monitoring the VR or a
97 script that triggers any other external stimuli. The application can also be
98 used without visual output for tracking the animal's activity. The configu-
99 ration of the virtual environment is described in detail in the Appendix.

100 The tracking data is saved in a text log file: for each camera frame a new
101 line is added with the timestamp of the frame, x-, y-, z- ball displacements
102 and x-, y-, z- position of the animal in the virtual environment. Additionally,
103 the tracking camera is set to output a frame trigger pulse with each recorded
104 frame, which can be used for synchronization with other applications, such
105 as two-photon imaging.

106 All experiments were performed on a PC with an Intel Xeon E5-1620 v3
107 @ 3.50GHz (8 cores) CPU, 32Gb DDR 4 @ 2993MHz of RAM, a NVIDIA
108 Quadro K620, 2Gb DDR3 RAM, 384 CUDA cores GPU and Microsoft Win-
109 dows 8.1 Enterprise edition operating system. OpenCV 3.4.1 and opencv-
110 contrib module were compiled using Microsoft Visual Studio Toolkit 14.0

111 (VS2015) in release configuration with CUDA 9.1 for tests of the GPU-
112 accelerated algorithms.

113 3. Theory/calculation

114 3.1. Modeling of optical flow camera projections

115 In this section a model is developed that describes how optical flow gen-
116 erated by ball rotations around different axes is projected onto the camera
117 sensor. Predictions of this model will be used as fit functions to extract ball
118 rotation parameters from measured optical flow distributions.

119 Ball motion is described by specifying an axis of rotation and an angular
120 velocity (axis-angle representation [11]). Assuming a Cartesian coordinate
121 system O with the origin at the center of the ball and a corresponding ISO-
122 conventional [12] spherical coordinate system S as shown in Figure 2, a point
123 p on the surface of the ball can be described with a vector $\mathbf{p}_S = (r_{ball}, \theta_p, \varphi_p)$.
124 If O is selected so that its z-axis coincides with the axis of ball rotation,
125 it becomes a convenient parametrization since only the azimuthal angle is
126 varying with time if the ball rotates around the polar axis with angular
127 velocity ω :

$$\mathbf{p}_S(t) = \begin{bmatrix} r_{ball} \\ \theta_p \\ \varphi_p + \omega t \end{bmatrix}. \quad (1)$$

128 In the Cartesian coordinate system O the same trajectory is expressed as

$$\mathbf{p}_O(t) = \begin{bmatrix} r_{ball} \sin(\theta_p) \cos(\varphi_p + \omega t) \\ r_{ball} \sin(\theta_p) \sin(\varphi_p + \omega t) \\ r_{ball} \cos(\theta_p) \end{bmatrix}, \quad (2)$$

129 where θ_p and φ_p are the polar and azimuth angle of a point on the surface of
130 the ball and r_{ball} is the constant radius.

131 Using a pinhole camera model [13] we can express the projection of points
132 on the ball surface onto the camera sensor. We here follow the same naming
133 and frame orientation convention as the OpenCV library [14, 15], shown in
134 Figure 3. The trajectory of a point on the ball surface in the camera frame
135 C is

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R \begin{bmatrix} r_{ball} \sin(\theta_p) \cos(\varphi_p + \omega t) \\ r_{ball} \sin(\theta_p) \sin(\varphi_p + \omega t) \\ r_{ball} \cos(\theta_p) \end{bmatrix} + T, \quad (3)$$

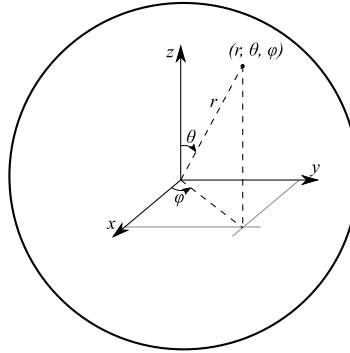


Figure 2: Spherical coordinate system complying with the ISO 31-11 convention: a point is addressed by radius r , polar angle θ and azimuthal angle φ

136 where R is the rotation matrix representing the orientation of the ball's axis
 137 of rotation and T is the position of the ball's reference frame origin in the
 138 camera frame C . The image plane coordinates (u, v) of the projected point
 139 p' are expressed as

$$\begin{aligned} u &= f_x \frac{x_c}{z_c} + center_x \\ v &= f_y \frac{y_c}{z_c} + center_y \end{aligned} \quad (4)$$

140 where f_x and f_y are the focal lengths expressed in pixel units and $(center_x,$
 141 $center_y)$ is the image center. The ball is centered in the camera's field of
 142 view and therefore $T = [0, 0, d]$, where d is the distance between the center
 143 of the ball and the camera aperture.

144 By calculating the ball point projection between two consecutive frames
 145 at time t and $t + \Delta t$, with Δt equal to the frame period, one can calculate the
 146 displacement of the point projection and predict optical flow. If all camera
 147 parameters as well as ball location and ball size are known, the model can
 148 be used to calculate the optical flow distribution on the camera. The model
 149 was implemented using the symbolic math library Sympy.

150 The angular velocity vector can be expressed as a sum of three orthogonal
 151 angular velocity components. For a camera-centered ball, it is convenient to
 152 choose the axes of a frame aligned with the camera frame with its origin at
 153 the ball's center. This gives three distinct rotations which can be registered
 154 by the camera as x-, y- and z- rotations (see Figure 3).

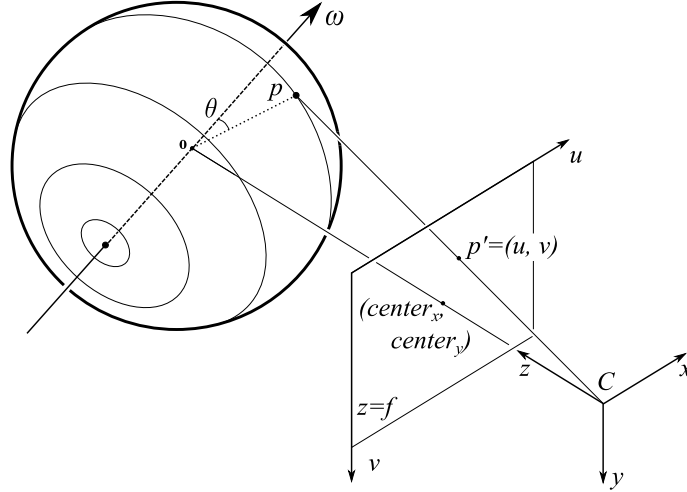


Figure 3: Projection of points on the ball with a pinhole camera model. Point p on the ball surface can be described with coordinates (r, θ, φ) in a polar coordinate system aligned with the ball's instant axis of rotation. According to expression (3) it can be expressed in Cartesian coordinates (x, y, z) in the camera's reference frame C and projected onto the point $p' = (u, v)$ in the image plane according to equation (4).

155 Filling in the matrix R in expression (3) to obtain ball rotations around
 156 the x , y , and z camera frame axes, respectively, and combining it with (4)
 157 yields the trajectories of the ball's point projections on the camera (for ro-
 158 tation around the specified axis):

$$x : \begin{cases} u = center_x - \frac{f_x r_{ball} \cos(\theta_p)}{d + r_{ball} \sin(\theta_p) \cos(\omega t + \varphi_p)} \\ v = center_y + \frac{f_y r_{ball} \sin(\theta_p) \cos(\omega t + \varphi_p)}{d + r_{ball} \sin(\theta_p) \cos(\omega t + \varphi_p)} \end{cases} \quad (5)$$

$$y : \begin{cases} u = center_x + \frac{f_x r_{ball} \sin(\theta_p) \cos(\omega t + \varphi_p)}{d - r_{ball} \sin(\theta_p) \sin(\omega t + \varphi_p)} \\ v = center_y + \frac{f_y r_{ball} \sin(\theta_p) \cos(\omega t + \varphi_p)}{d - r_{ball} \sin(\theta_p) \sin(\omega t + \varphi_p)} \end{cases} \quad (6)$$

$$z : \begin{cases} u = center_x + \frac{f_x r_{ball} \sin(\theta_p) \cos(\omega t + \varphi_p)}{d + r_{ball} \cos(\theta_p)} \\ v = center_y + \frac{f_y r_{ball} \sin(\theta_p) \cos(\omega t + \varphi_p)}{d + r_{ball} \cos(\theta_p)} \end{cases} \quad (7)$$

159 As seen in Figure 4 a-c, rotations around the three different camera frame
 160 axes produce recognizably different velocity field distributions. To distinguish
 161 them, a ring-shaped region of interest is selected around the center of the ball
 162 (Figure 4, d-f). For each point in the ROI, two principal directions are se-
 163 lected (radial - orthogonal to the ROI circle at this point - and tangential

164 to the ROI circle at this point, directed counterclockwise). The optical flow
165 vector is then projected onto these directions to find its radial and tangential
166 components with respect to the ROI (Figure 4, g-i). These radial and tan-
167 gential components of the optical flow build distinct distributions over the
168 ROI, which allows to separate the motion components (Figure 4, j-l).

169 These distributions can be fitted well by the following two functions

$$\begin{cases} f_{rad}(\varphi) = c_{xy\ rad}(\omega_{xy} \cdot \Delta t) \sin(\varphi + \phi) \\ f_{tan}(\varphi) = c_{xy\ tan}(\omega_{xy} \cdot \Delta t) \cos(\varphi + \phi) + c_z(\omega_z \cdot \Delta t), \end{cases} \quad (8)$$

170 where $c_{xy\ rad}$ and $c_{xy\ tan}$ (px/rad) are calibration factors relating the angular
171 displacement for angular velocity vectors lying in the xy-plane to the respec-
172 tive optical flow (see Appendix for details). Similarly, c_z (px/rad) is a factor
173 relating angular displacement for angular velocity vectors lying on the z-axis
174 to optical flow; $(\omega_{xy} \cdot \Delta t)$ and $(\omega_z \cdot \Delta t)$ are the angular displacements about
175 a rotational axis lying in the xy-plane or on the z-axis, respectively, and ϕ
176 is the orientation of the axis of rotation in the xy-plane. Note, that radial
177 and tangential components of optical flow induced by ω_{xy} have different co-
178 efficients, since the magnitudes of the respective optical flow distributions do
179 not match, while the ω_z rotation induces only tangential optical flow, and a
180 single coefficient c_z is sufficient. The calibration factors subsume all setup
181 parameters, such as camera focal length, and can be determined as described
182 below and in the Appendix.

183 3.2. Finding calibration factors by function fitting

184 Calibration factors were determined experimentally by recording a se-
185 quence of frames with two cameras pointed at the ball center and positioned
186 orthogonally to each other (see Figure 14 in the Appendix). The z-component
187 is a planar (that is, in the camera focal plane) motion which can be measured
188 accurately with optical flow (as was verified using simulated as well as motor
189 actuated ground truth data), the xy- coefficient can be found by regression
190 between the estimates of the ball motion based on the two cameras. This is
191 described in more detail in the Appendix.

192 3.3. Tracking algorithm

193 The ball tracking algorithm takes two consecutive frames of ball motion
194 and calculates the three independent rotational displacements of the ball, i.e.
195 the x-, y- and z-axis angular displacements, by fitting their projections onto

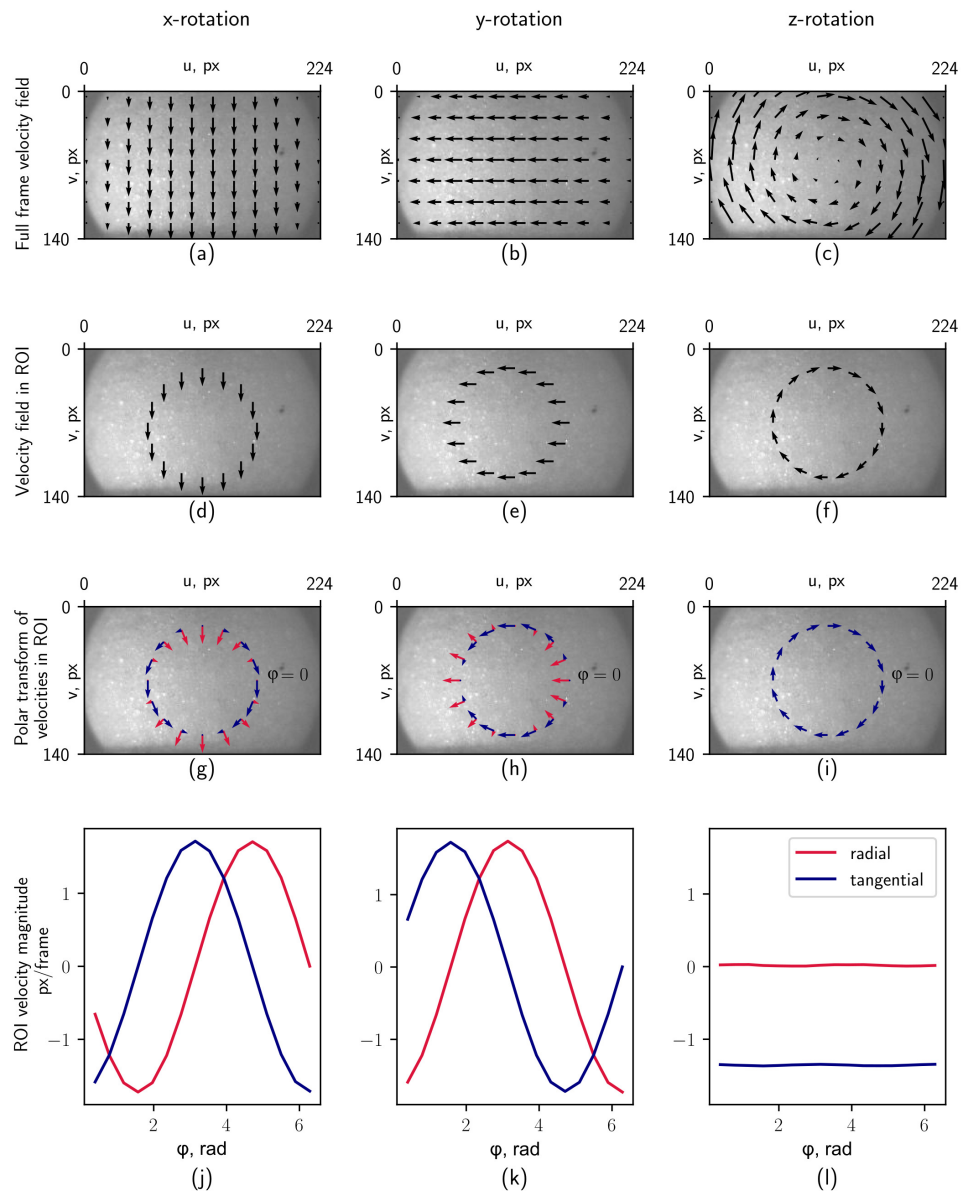


Figure 4: Predicted optical flow produced by rotations of the ball around the respective three independent axes of rotation.

196 the radial and tangential directions with function (8) in a circular ROI. Since
197 the distribution of optical flow in the circular ROI is very noisy the optical
198 flow is averaged over a band (limited by a minimal inner and maximal outer
199 ring, see Appendix for how ROI size was determined) in the radial direction
200 to increase robustness. This results in the following algorithm:

Data: Two grayscale images of the ball video
Result: Angular velocity vector $(\omega_x, \omega_y, \omega_z)$
Input: $frame_1, frame_2$

- 1 polar transform $frame_1, frame_2$ with respect to the center of the ROI
- 2 crop the polar-transformed $frame_1, frame_2$ to keep only the ROI
- 3 $flow_u, flow_v \leftarrow$
201 `compute_optical_flow(frame_1(cropped), frame_2(cropped))`
- 4 $flow_{ROI_{rad}}, flow_{ROI_{tan}} \leftarrow$ average $flow_u, flow_v$ over radius
- 5 fit $flow_{ROI_{rad}}, flow_{ROI_{tan}}$ with function (8) by finding
 $(\omega_{xy} \cdot \Delta t), \phi, (\omega_z \cdot \Delta t)$
- 6 $(\omega_x, \omega_y, \omega_z) \leftarrow \omega_{xy} \sin(\phi), \omega_{xy} \cos(\phi), \omega_z$
 return $(\omega_x, \omega_y, \omega_z)$

202 Data from the algorithm's intermediate steps are shown in Figure 5. The
203 polar transform (Figure 5, b) is calculated before computing optical flow,
204 allowing the following optimizations: first, the ring-shaped ROI transforms
205 into a rectangular image section, where one dimension corresponds to the
206 azimuthal angle of the ROI φ and the other to the radius ρ . This means
207 that the ROI can be isolated by cropping the image and optical flow can
208 be calculated for a small fraction of the full frame. Secondly, in the polar-
209 transformed ROI, the optical flow u and v components correspond to the
210 radial and tangential motions directly, which eliminates the need of flow
211 re-projection. The dependence of the optical flow calculation on the ROI
212 position which results from the polar transform (see Figure 5, b) is included
213 in the experimentally determined calibration coefficients.

214 Other optimizations included passing over the polar-transformed and
215 cropped frame to the next iteration to save on those operations, and us-
216 ing the previous iteration's function fitting result as a prior for the solver in
217 the next frame. Function fitting is done with Downhill Solver included in
218 OpenCV, which implements the Nelder-Mead downhill simplex method [16].

219 For a compiled executable of the VR application please send an email to
220 ivan.vishniakou@caesar.de or johannes.seelig@caesar.de.

221 4. Results

222 4.1. Tracking algorithm implementation

223 For real-time applications the optical flow calculation was implemented
224 in a C++ program using OpenCV 3.4.1. Best performance in terms of pro-
225 cessing speed and tracking accuracy was achieved with the Farneback optical
226 flow algorithm on a 60×140 ROI scaled down to 30×140 (see the Appendix
227 for a comparison of a range of optical flow algorithms and for selecting ROI
228 parameters). Preprocessing (polar transform, cropping and resizing) together
229 with optical flow estimation took 1.45 ms (standard deviation $s = 0.54$ ms),
230 the function fitting by downhill solver was capped at 30 iterations to keep its
231 run-time under 0.4 ms. These parameters allow to achieve real-time track-
232 ing at 500 frames per second with the current PC configuration. All time
233 measurements were performed with the `cv::TickMeter` class.

234 We noticed that occasionally frames were dropped, for example when
235 starting the program or due to other operating system related processes.
236 Such frame drops can be detected in the saved data file by looking for time
237 stamps with a separation of more than 2 ms. If a frame is dropped, the next
238 displacement is calculated with the latest available frame before the frame
239 drop.

240 4.2. Evaluation of tracking accuracy

The evaluation of tracking accuracy is based on comparing ground truth $\omega_{\mathbf{GT}}$ and estimated angular velocity $\omega_{\mathbf{est}}$. Three metrics are selected for this comparison: the absolute error (in degrees per frame)

$$\varepsilon_{magn_abs} = abs(\|\omega_{\mathbf{est}}\| - \|\omega_{\mathbf{GT}}\|),$$

the relative (percentage) magnitude error

$$\varepsilon_{magn_relative} = \frac{\varepsilon_{magn_abs}}{\|\omega_{\mathbf{GT}}\|} \cdot 100\%,$$

and the orientation error (in degrees), which is the angle between the ground truth and estimated rotation vectors in the plane spanned by these two vectors,

$$\varepsilon_{orient} = \cos^{-1} \left(\frac{\omega_{\mathbf{est}} \cdot \omega_{\mathbf{GT}}}{\|\omega_{\mathbf{est}}\| \|\omega_{\mathbf{GT}}\|} \right).$$

241 Two ground truth datasets were generated for testing the tracking per-
242 formance (see Appendix for details). One consisted of simulated data which

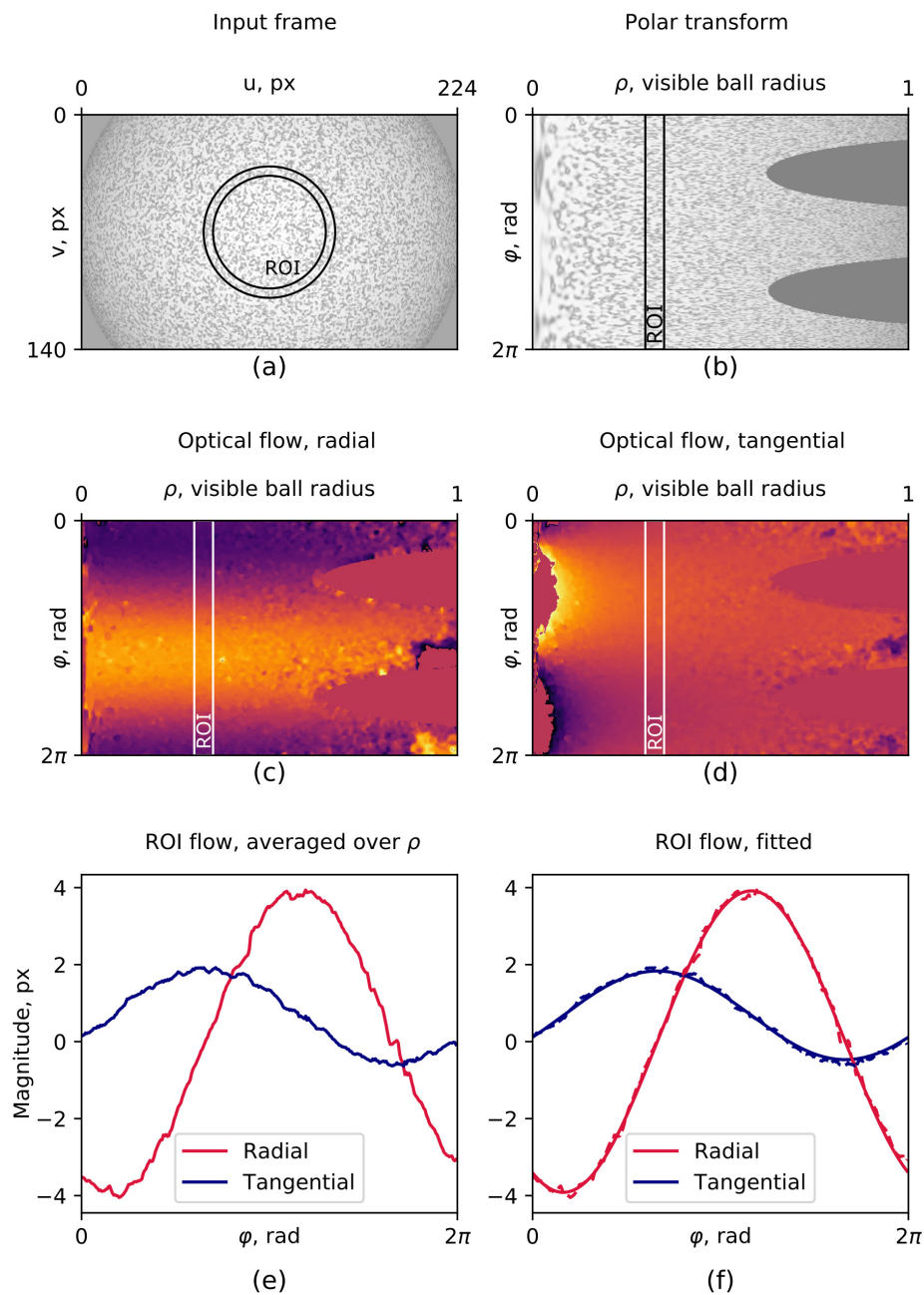


Figure 5: Ball rotation evaluation from optical flow. Input frame (a) is polar-transformed (b) and optical flow (c, d) is extracted by comparing it to the previous frame. The distribution of optical flow in the ROI (e) is then fitted with function (8) (f).

243 offered control of all rotation parameters, while however not fully replicating
244 surface texture or lighting and camera conditions encountered in the actual
245 experiment. A second dataset was generated by rotating a 60 mm ball (10×
246 scale model for the 6 mm ball used for fly behavior) with a stepper motor.

247 Using the simulated data with accelerated rotations the operation range
248 of the tracking algorithm was determined (see Appendix for details). Figure
249 7 shows the comparison of the integrated tracked trajectory with ground
250 truth data. A value of 50° error for the x-axis corresponds to a 2.6 mm
251 error in a planar trajectory. The tracking error depends on the rotation
252 speed of the ball (for the ROI and parameters used here, see Figure 15 in the
253 Appendix). The average tracking error stays under 10% (magnitude) and
254 7.5° (orientation) for rotations up to 1.70° per frame, which corresponds to
255 45 mm/s linear speed for a 6 mm diameter ball.

256 Using a stepper-motor a ball was rotated at constant speed around a
257 single axis (see Appendix for details). The integrated rotation magnitude
258 error and orientation errors for one rotation sequence are shown in Figure 6.

259 4.3. VR system timing

260 The run-time of the tracking algorithm was measured using internal tim-
261 ing. The total latency of the combined tracking and VR setup was estimated
262 using a high-speed camera filming the VR display: the display was triggered
263 to change state, and the state change was detected with the tracking camera.
264 This sequence contains all the latency components for one cycle of a closed
265 loop VR experiment.

266 The main component of the latency is the display device input lag. It
267 depends strongly on the selected video mode (Table 1) and the lowest values
268 were found at 120 Hz update rate with V-sync disabled.

Table 1: Latency measurement of a camera – virtual reality – display system

Display mode	Median latency, ms
120 Hz	25 ± 8 ms
120 Hz, V-sync	33 ± 8 ms
60 Hz	50 ± 16 ms
60 Hz, V-sync	50 ± 16 ms

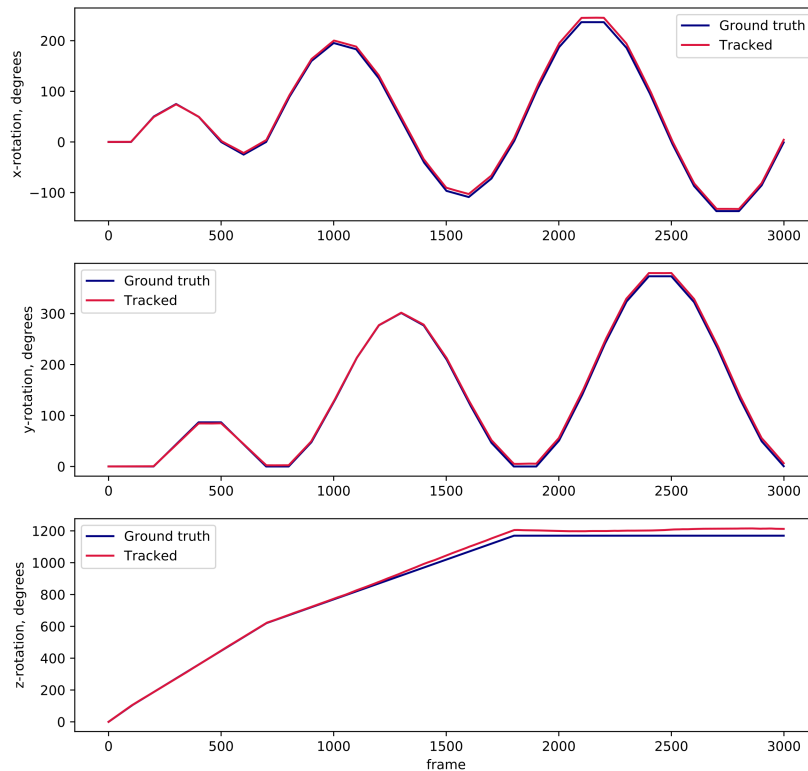


Figure 6: Integrated tracked trajectories and ground truth for stepping motor actuated ball movement at constant speed.

269 *4.4. Behavior experiments in VR*

270 For testing the setup for fly behavior, we used a simple VR with a single
271 bright stripe of 10 degree width and 45 degree height projected onto a two-
272 part screen similar to [7] of 9 mm height and 2 times 18 mm width made out
273 of black paper. The fly's head was fixed to its thorax with UV glue, and the
274 wings were glued to the tethering pin using UV glue to encourage walking
275 behavior [17]. The stripe was visible for 4/5th of the flies virtual surroundings
276 and disappeared for 1/5th behind the fly. This resulted in robust orientation
277 behavior with the fly walking away from the stripe (see Figure 8).

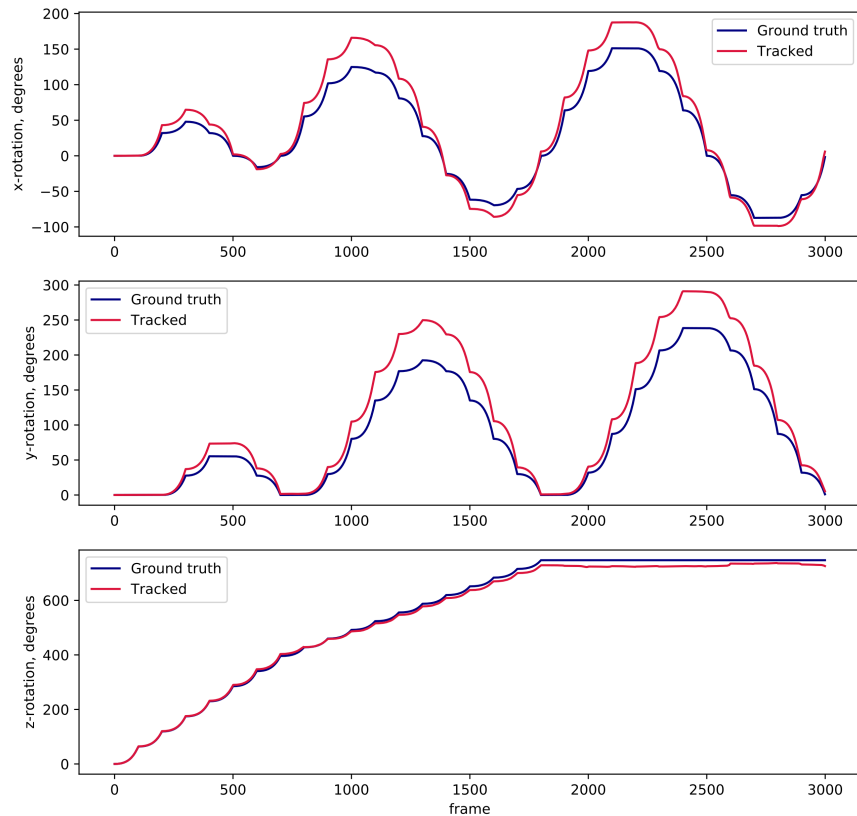


Figure 7: Integrated tracked trajectories and ground truth of ball motion for an example of simulated, accelerated rotations.

278 5. Discussion

279 An optical flow-based method for tracking of spherical treadmill motion
280 using a single high-resolution camera was developed and integrated with
281 virtual reality projection. Treadmill motion was monitored at 500 Hz, the
282 display update rate was 120 Hz, and the total latency for an entire VR cycle
283 was 30 ± 8 ms (from reading out ball movement to projecting the updated
284 VR frame on the display). The system combines the flexibility of optical flow
285 based tracking with the convenience of using a single high resolution camera,
286 off-the-shelf components, and open source software.

287 Compared with an optical mouse based system the approach developed

Table 2: Latency time breakdown for the closed-loop virtual reality setup

Component	Time, ms
Camera exposure	0.1
Sensor readout	1.2
Transfer to PC	<2
Tracking algorithm	2
VR rendering + Transfer to DMD over DisplayPort + display latency	25 ± 8
Total	30 ± 8

288 here offers independent control over the different components of the system,
289 such as camera settings and optical flow processing algorithm. Image pro-
290 cessing is therefore not limited to a preset and unknown algorithm, but a
291 suitable approach can be selected and its parameters can be tuned (see Ap-
292 pendix for a comparison of all optical flow algorithms tested). Additionally,
293 the resulting tracking accuracy and the operation range (which for all optical
294 flow based methods depends on the combination of field of view, movement
295 speeds and processing algorithm) can be tested using simulated data. Dif-
296 ferent from approaches using optical mice, tracking of all three degrees of
297 freedom of ball rotation is achieved with a single camera which can be po-
298 sitioned flexibly and can easily aligned and calibrated. This also facilitates
299 adapting the system to a variety of experimental conditions and scales.

300 As an alternative to optical flow approaches, a solution relying on a unique
301 patterning of the ball surface was implemented in [6]. While this pattern
302 matching approach avoids integration errors, optical flow can on the other
303 hand track any surface with sufficient speckle contrast under a variety of
304 lighting conditions and can be adapted to different ball sizes.

305 Compared to cameras, optical mouse sensors typically have a higher frame
306 rate (at the cost of lower pixel resolution) and time jitter and latencies are
307 shorter on a dedicated processing chip. For example, [3] measured the tem-
308 poral accuracy of a customized optical mouse sensor and found a tracking
309 latency of less than $500 \mu\text{s}$, compared to 2 ms here. However, as the latency
310 time breakdown shows (see Table 2), the delay is mostly caused by the stan-
311 dard display interface (DisplayPort). Since the camera tracking latencies and

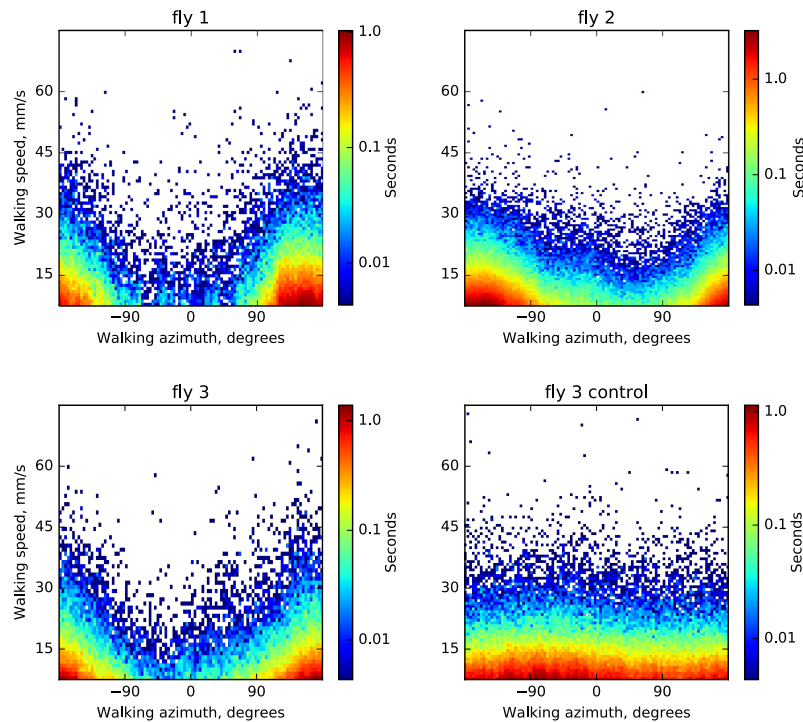


Figure 8: Walking speed and walking direction of three flies walking in closed loop with a bright stripe on a black screen (see text for details). Color indicates histogram bins (binned with a grid of 100×100 pixels) of time spent in each state for a total trial length between 15 and 25 minutes (depending on the fly)). In darkness the flies don't show a pronounced orientation preference (as shown for fly 3, control).

312 jitter are small compared to the display latencies and jitter, this should have
313 a very limited impact on VR behavior. The overall latency of the closed-
314 loop virtual reality system was 30 ± 8 ms on average, comparable to optical
315 mice-based solutions and about three times faster than another camera-based
316 pattern matching approach [6]. The display update rate of typical virtual
317 reality setups with data transfer through the display port is generally limited
318 to 60 to 120 Hz. Typical latencies found with such PC and camera-based
319 solutions are between 50-90 ms [18, 6] and have been shown to be accept-
320 able for closed-loop behavior experiments with a variety of species [1, 18]. We
321 tested the system for behavioral performance in fruit flies and found a robust
322 avoidance response of a bright stripe under the display conditions tested.

323 Overall, a VR system based on optical flow measurements with a high-

324 resolution camera was developed with short latencies suitable for VR exper-
325 iments at different scales.

326 6. Acknowledgements

327 We would like to thank Santosh Thoduka for suggestions on camera cal-
328 ibration and helpful discussions, Fiona Ross and Stephanie Miceli for initial
329 behavioral tests, Andres Flores for help with behavior analysis and designing
330 hardware for stimulus projection, Bernd Scheiding for electronics support,
331 and Dan Turner-Evans for help with IR illumination.

332 7. Appendix

333 7.1. Tracking settings

334 Tracking-specific settings are stored in the `Config\tracking.cfg` file and
335 include the path to the camera configuration file, the name of the camera to
336 be used for tracking, as well as calibration coefficients as explained in section
337 3.1. Those parameters can be found in the corresponding sections of the
338 config file:

```
339 [ camera ]  
340 LoadCameraSettings=true  
341 CameraSettingsFile=Config\camera_settings.pfs  
342 TrackingCameraName=ball_cam  
343  
344 [ calibration ]  
345 BallCenterX=112.0  
346 BallCenterY=70.0  
347 BallRadius=116.0  
348 CxyRad=100.31  
349 CxyTan=76.85  
350 Cz=20.63
```

351 The camera was set to capture images at 500 fps, and the exposure time
352 was set to 100 μ s to prevent motion blurring (and required using sufficient
353 infrared intensity). The resolution was set to 224×140 , the highest resolution
354 that was handled by our system in the available 2 ms per frame processing
355 time. The settings of the camera can be saved in the Pylon configuration file
356 and set to be loaded in the tracking configuration.

357 *7.2. DMD settings*

358 In order to use the LightCrafter 6500 DMDs at a 120 Hz display rate,
359 they need to be connected through the DisplayPort interface and have to be
360 configured to use Dual pixel clock mode, run in Video Pattern mode: this
361 is an operation mode where each frame arriving through the video interface
362 (RGB 8bpp) is regarded as 24 independent bit planes and each can be inde-
363 pendently selected to be displayed with an arbitrary timing diagram. In this
364 mode the most significant bit of the green channel is taken and displayed for
365 7 ms, resulting in 120 fps display rate. Additionally, it might be necessary to
366 enable 120 Hz refresh rate in the Windows display adapter settings for both
367 DMDs, which are visible to the operating system as normal displays.

368 The most convenient way to automate the DMD configuration is by saving
369 these settings in a batch file (sequence of commands sent to the DMD) and
370 upload it to the DMD as a startup script (this is a feature supported by
371 LightCrafter 6500).

372 *7.3. Virtual environment settings*

373 Virtual reality has several configuration parameters which are specified
374 in the `Config\vr.cfg`. It has game-engine specific parameters in the cor-
375 responding section, like the position of the VR window in screen coordi-
376 nates and its size. Since we used two DMDs, it is spanned over twice
377 the resolution of the DMD in width and located to the right of the main
378 screen. `VSync` is disabled, since it is otherwise introduces additional la-
379 tency to the display. `minFps` is set equal to the refresh rate of the displays;
380 `maxFps` and `maxInactiveFps` are capping the update rate of the game en-
381 gine. `maxInactiveFps` is the frame rate limit for a window while it is not in
382 focus, i.e. when the user is working with other applications. This is almost
383 always the case, since the game window is not visible for the computer user
384 and other applications, for example for monitoring the VR, are used. The
385 default value is 60 fps, and therefore needs to be increased in order to use
386 the full capacity of the 120 Hz DMD displays.

The “transforms” section of the configuration defines how the ball ro-
tation maps to motion in the virtual environment. The tracking signal is
calibrated so that it presents angular displacements of the ball (in radians)
about the x-, y-, and z- axes of the camera. Depending on how the camera
is oriented relative to forward walking direction of the animal, the mapping
varies. `ballXYZtoArenaXYZ` is a 3×3 rotation matrix, and

$$\text{arena_displacement} = \text{ballXYZtoArenaXYZ} \cdot \text{ball_displacement}.$$

Similarly, the turning in the virtual environment is calculated from the ball displacement as a dot product with `ballXYZtoArenaYaw`:

```
arena_yaw_displacement = ballXYZtoArenaYaw · ball_displacement.
```

387 In the provided example the tracking camera is placed behind the ball and
388 is aligned with the forward walking direction of the fly:

```
389 [ transforms ]  
390 ballXYZtoArenaXYZ=0 0 -3.0 0 0 0 3.0 0 0  
391 ballXYZtoArenaYaw=0 -57.3248 0
```

392 The 3 mm radius of the ball is incorporated in the motion transform, and
393 the yaw rotation contains a conversion from radians to degrees.

394 *7.4. Arena scripts*

395 Although the Urho3D game engine allows saving and loading 3D envi-
396 ronments in XML format, a more readable and convenient way is to declare
397 environment properties in a script. Urho3D supports a compiled scripting
398 language called AngelScript [19], which is object-oriented, and features syn-
399 tax similar to C++. The scene is represented in the game engine as a hier-
400 archical tree structure: each entity of the 3D environment is a node that can
401 be attached to either other nodes or to the root node of the scene. The coor-
402 dinates of the nodes are always interpreted in the coordinate frame of their
403 parent nodes. In-detail information is provided in the game engine’s manual
404 [20], and sample arenas are documented. Dithering, which is required for
405 displaying grayscale images with a DMD operating in binary display mode is
406 done with a postprocessing shader, which is added to the rendering pipeline
407 of the game engine. The configuration of the VR display is also shown in a
408 documented example script.

409 *7.5. Ground truth datasets*

410 One ground truth dataset was recorded with a 60 mm polystyrene foam
411 ball (10 × scale model) mounted on a stepper motor [6]. The motor was
412 controlled using Grbl firmware for Arduino boards [21]. The distance from
413 the camera to the ball is was scaled accordingly 10× as well. Sequences of
414 1000 frames were recorded with an angular step size of 0.281, 0.562, 1.124
415 and 1.686 degrees per frame, in two orientations corresponding to y- and
416 z-rotations.

417 A second ground truth dataset consisted of simulated data which allowed
418 full control over all parameters: a 3D scene was reconstructed with Blender
419 3D including a camera with focal parameters identical to the actual tracking
420 setup, a light source and a ball with grainy surface texture. The model of
421 the ball was rotated and rendered using Blender’s API and in-built Python
422 interpreter. Two simulated rotation sequences were created: The first one
423 was used for calibration and accuracy validation and consisted of rotations
424 of 1° per frame (corresponding to about half of the maximal walking speed
425 of the fly), with 30 different rotation axes and 100 frames for each axis. The
426 second dataset consisted of rotations of exponentially increasing magnitude
427 from 0° to 2° per frame, intended to test the operating range of the tracking
428 algorithm. The orientations of the rotational axes for both datasets were
429 selected evenly distributed.

430 An actual tracking camera image is shown side-to-side with a simulated
431 image in Figures 9 and 10: the real image has out-of-focus areas, some bright
432 speckles from direct reflection of the lighting, and an overall uneven bright-
433 ness distribution.

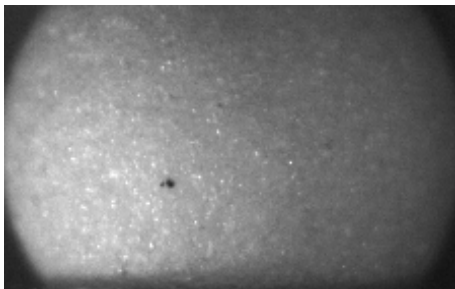


Figure 9: Frame captured by tracking camera in VR setup.

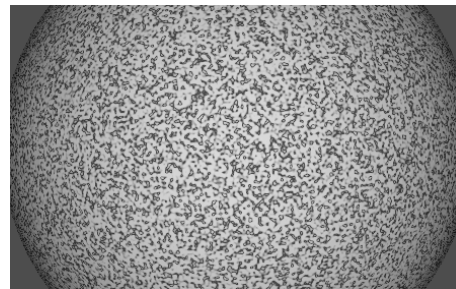


Figure 10: Frame from simulated ball rotation sequence.

434 7.6. Optical flow algorithms

435 Algorithms for computing optical flow vary significantly in computational
436 complexity, run-time and resulting accuracy. OpenCV library features sev-
437 eral implementations of the optical flow estimation algorithms (as of version
438 3.4.1) and the algorithms tested here were: Lucas-Kanade sparse feature
439 tracking [22], Gunnar Farneback’s algorithm of dense optical flow [23], the
440 optical flow algorithm by Brox et. al. [24], Dual TV- L_1 [25], Dense Inverse

441 Search (DIS) [26], SimpleFlow [27], DeepFlow [28] and PCAFlow [29] (see
442 Table 3).

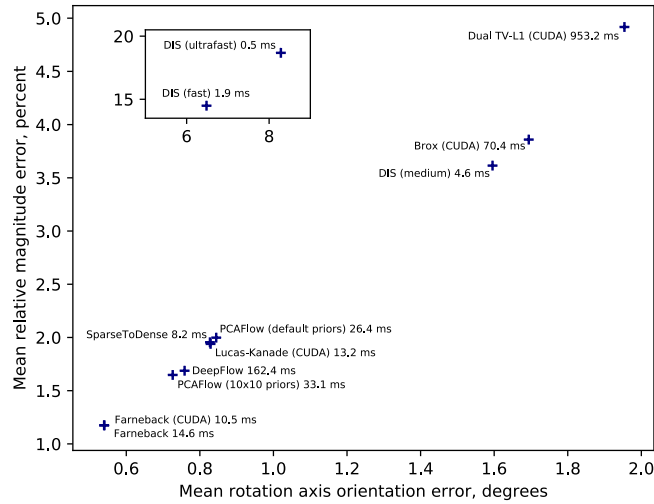
443 The speedup through reducing the size of the ROI varies depending on the
444 algorithm, which motivated a comparison of the run-times depending on the
445 input size (Fig 12). To achieve a tracking frequency of 500 frames per second,
446 the run-time should not exceed 2 ms, preferably be less to accommodate the
447 frame preprocessing and ball motion estimation calculations. The algorithms
448 relying on sparse features pre-selection fail in smaller ROIs (Sparse-to-dense,
449 PCAFlow). Farneback, although being slow with default parameters, could
450 be sped up significantly by reducing the number of scale levels and by limiting
451 the internal iterations. It turned out to be the algorithm most suitable for
452 the application, since it can handle a ROI of up to 60×140 px in under 2
453 ms when tuned for speed.

Table 3: Tracking performance on the the simulated rotation dataset, using full frame optical flow.

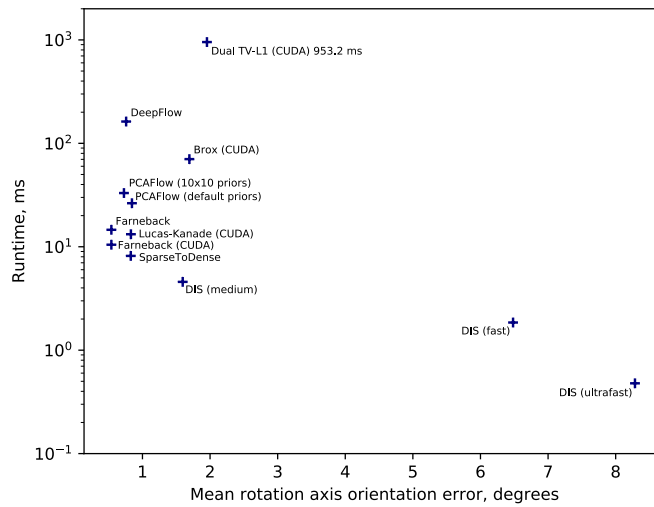
Algorithm	Mean magnitude error, %	Mean orientation error, degrees	Time, ms
Dual TV-L1 (CUDA)	4.9	1.95	953.2
DeepFlow	1.7	0.76	162.4
Brox (CUDA)	3.9	1.69	70.4
PCAFlow (10x10 priors)	1.6	0.73	33.1
PCAFlow (default priors)	2.0	0.84	26.4
Farneback	1.2	0.54	14.6
Lucas-Kanade (CUDA)	1.9	0.83	13.2
Farneback (CUDA)	1.2	0.54	10.5
SparseToDense	2.0	0.83	8.2
DIS (medium)	3.6	1.6	4.6
DIS (fast)	14.5	6.48	1.9
DIS (ultrafast)	18.7	8.29	0.5

454 7.7. Selecting ROI parameters

455 The ROI is a small part of the full frame used to identify the rotation
456 direction of the ball from optical flow. The ROI is limited by two concentric



(a)



(b)

Figure 11: Ball rotation estimation errors (a) and run-time of the optical flow methods on full frame of the simulated dataset

457 rings with their centers coinciding with the center of the frame, which is also
 458 selected to agree with the center of the tracked ball. By specifying the radius
 459 of the ROI ρ_{ROI} (px) and its width $\Delta\rho_{ROI}$ (px), one can vary the run-times

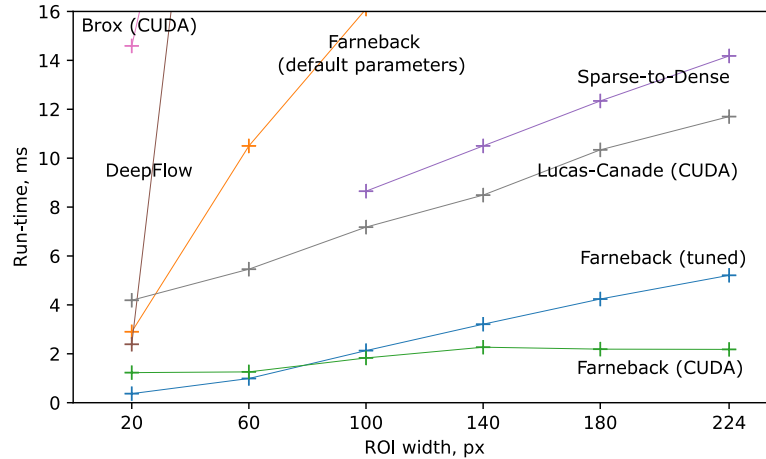


Figure 12: Run-time of optical flow algorithms depending on input size (ROI width \times 140 px). Measured as average over 3000 frames with simulated dataset with accelerating rotations.

460 of the fitting algorithm and its accuracy. The ROI parameters are chosen by
 461 running the tracking algorithm on simulated datasets (constant 1° per frame
 462 angular velocity) and finding the variance in the estimated angular velocity
 463 magnitude and orientation compared to ground truth. Figure 13 shows the
 464 summed magnitude and orientation error (normalized) depending on ρ_{ROI} ,
 465 using $\Delta\rho_{ROI} = 10$ px. The lowest value was achieved for $\rho_{ROI} = 0.27$, $\rho_{max} =$
 466 60 px.

467 7.8. Calibration coefficients

468 The coefficients $c_{xy \text{ rad}}$, $c_{xy \text{ tan}}$ and c_z in expression (8) are required for
 469 relating the optical flow in pixels (as determined with the optical flow algo-
 470 rithm) to the actual ball displacement in radians. Apart from a unit conver-
 471 sion factor these coefficients depend on the camera magnification, the size
 472 of the ball, and the position of the ROI on the ball. These factors can be
 473 determined in any of the following ways: by correlating optical flow induced
 474 by known ball rotations (with a ground truth dataset); similar, but using
 475 modelled optical flow with the pinhole camera model as in expressions (5-7,
 476 provided the optical parameters of the camera are known); or using a second
 477 synchronized camera. These methods are described in more detail in the

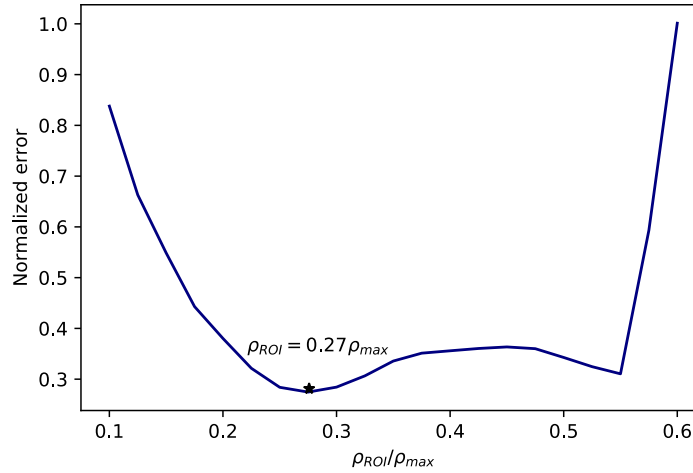


Figure 13: Summed tracking orientation and rotation errors (normalized), obtained by running the tracking algorithm with $\Delta\rho_{ROI} = 10$ px. ρ_{max} is the visual radius of the ball (116 px).

478 following subsections.

479 7.8.1. Calibration with ground truth dataset

480 The optical flow in the tracking ROI was calculated on simulated data and
 481 its distributions are fitted with function (8). Then the calibration constants
 482 can be found as the ratios:

$$\begin{aligned}
 c_z &= \frac{\omega_{zGroundTruth}}{\omega_{zFitted}} \\
 c_{xy \text{ rad}} &= \frac{\omega_{xyGroundTruth}}{\omega_{xy \text{ radFitted}}} \\
 c_{xy \text{ tan}} &= \frac{\omega_{xyGroundTruth}}{\omega_{xy \text{ tanFitted}}},
 \end{aligned} \tag{9}$$

483 where $\omega_{xyGroundTruth}, \omega_{zGroundTruth}$ are the known rotations, and $\omega_{xy \text{ radFitted}},$
 484 $\omega_{xy \text{ tanFitted}}$ are the results of fitting radial and tangential optical flow distri-
 485 butions in the ROI assuming $c_{xy \text{ rad}} = c_{xy \text{ tan}} = c_z = 1.0$.

Using the rendered ball rotation dataset with constant rotations, the fol-

lowing calibration factors were found:

$$c_{xy \text{ rad}} = 100.31 \text{ px/rad} = 1.75\text{px}/^\circ,$$

$$c_{xy \text{ tan}} = 76.85 \text{ px/rad} = 1.34\text{px}/^\circ,$$

$$c_z = 20.63 \text{ px/rad} = 0.36\text{px}/^\circ.$$

486 *7.8.2. Calculation of the calibration factors from the scene model*

487 Using the ball projection model (equations 3 and 4) with substituted cam-
488 era and scene parameters, optical flow distribution in the tracking ROI can
489 be predicted and used for calibration similar to the previous method. With
490 the help of a scaled-up calibration setup, the camera and scene parameters
491 were measured as listed in Table 4.

Table 4: Measured parameters of the tracking setup scale model

Parameter	Value
Distance to the ball	1400 mm
Ball radius	30 mm
Image resolution	224 × 140 px
Ball image radius	116 px

492 Using these measurements, and the pinhole camera model (4) the focal
493 length of the camera can be calculated (assuming the image projection center
494 is exactly in the middle of the frame, $center_x = 112 \text{ px}$, $center_y = 70 \text{ px}$):

$$116 \text{ px} = f \frac{30 \text{ mm}}{1400 \text{ mm}}$$

$$f = 116 \text{ px} \frac{1400 \text{ mm}}{30 \text{ mm}} = 5413 \text{ px}$$

$$f_x = f_y = 5410 \text{ px}$$

$$center_x = 112 \text{ px}$$

$$center_y = 70 \text{ px}$$

495 The model with these parameters was used to predict the optical flow
496 induced by different rotations according to expressions ((5) - (7)).

497 The calibration factors found with this method were close to ones obtained
 498 with ground truth rotations:

$$c_{xy \text{ rad}} = 95.37 \text{ px/rad} = 1.66\text{px}/^\circ,$$

$$c_{xy \text{ tan}} = 70.31 \text{ px/rad} = 1.21\text{px}/^\circ,$$

$$c_z = 22.28 \text{ px/rad} = 0.39\text{px}/^\circ.$$

499 *7.8.3. Two-camera calibration*

500 According to the mathematical model described above the z-component
 501 of the angular velocity measured by the tracking camera is detected as the
 502 xy-plane component by a second camera filming the same rotation from an
 503 orthogonal direction (see Figure 14).

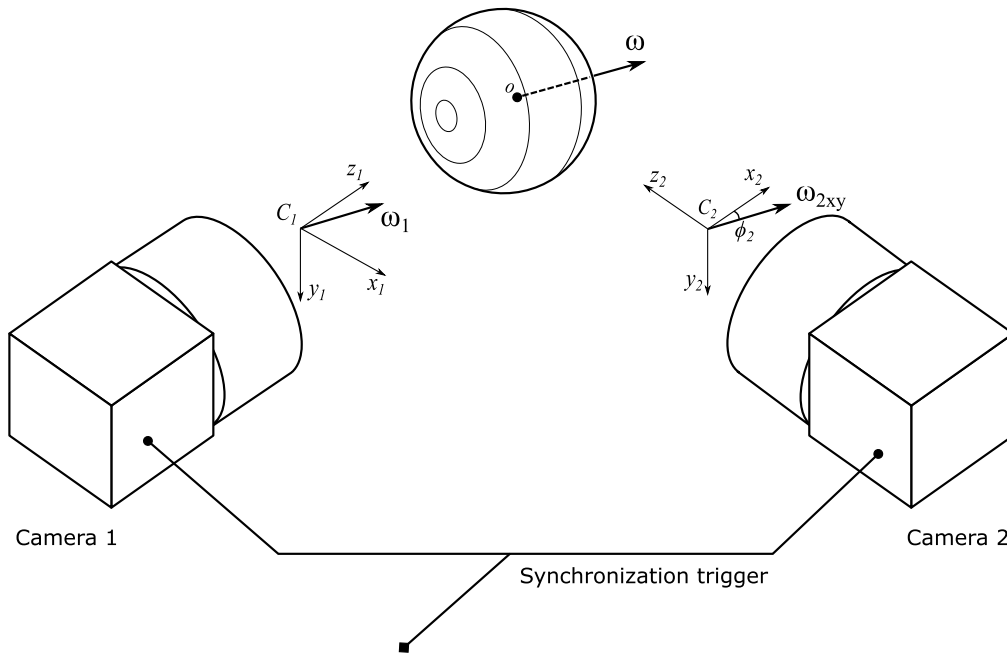


Figure 14: Calibration procedure using two cameras. The z-component of angular velocity of the ball in the reference frame of camera 1 is projected onto the x-axis in the reference frame of camera 2. Calibration is performed by correlating those components to find c_{xy} .

504 As shown using the optical flow model (Figure 4), a rotation of the ball
 505 around the camera's z-axis results in all of its points moving along circular

506 trajectories in the frame plane. This means that the angular displacement of
507 the ball equals the angular displacement of the points in the frame along their
508 trajectories, and the tangential optical flow in the ROI induced by this motion
509 is proportional to the radius of the ROI, meaning that c_z can be estimated
510 with a single camera without measuring any additional parameters.

511 Different from z-factors, $c_{xy \text{ rad}}$ and $c_{xy \text{ tan}}$ depend on the camera prop-
512 erties and the scene geometry. Therefore, a second camera is introduced,
513 filming the ball from an orthogonal direction, as shown in Figure (14). The
514 z-component measured by camera 1 is then equal to the x-component mea-
515 sured by camera 2, and camera 1 can therefore be used to calibrate xy-
516 tracking by camera 2:

$$\begin{cases} c_{xy \text{ rad}} = \frac{c_z \text{ estimated} \omega_{z1}}{\omega_{xy \text{ rad}2} \cos \phi_2} \\ c_{xy \text{ tan}} = \frac{c_z \text{ estimated} \omega_{z1}}{\omega_{xy \text{ tan}2} \cos \phi_2}, \end{cases} \quad (10)$$

517 where $c_z \text{ estimated}$ is estimated as described, ω_{z1} is measured by camera 1,
518 $\omega_{xy \text{ rad}2}$, $\omega_{xy \text{ tan}2}$ and ϕ_2 are found by fitting function (8) on radial and tan-
519 gential optical flow in the ROI from camera 2, respectively, initially assuming
520 $c_{xy \text{ rad}} = c_{xy \text{ tan}} = 1.0$. ϕ_2 is the orientation of the ball's axis of rotation in
521 the xy-plane of camera 2, and the cosine yields the x-axis projection of that
522 rotation, corresponding to measured z-rotation by camera 1.

523 To perform this calibration, both cameras need to be set up with identical
524 objectives and aligned at the same distance from the tracked ball at right
525 angles (see Figure 14); the ball is let to spin freely (at a speed within the
526 range where the tracking algorithm is valid) with a stream of air while the
527 cameras capture frames synchronized by an external triggering signal.

- 528 [1] D. A. Dombeck, M. B. Reiser, Real neuroscience in virtual worlds,
529 Current opinion in neurobiology 22 (2012) 3–10.
- 530 [2] A. C. Mason, M. L. Oshinsky, R. R. Hoy, Hyperacute directional hearing
531 in a microscale auditory system, Nature 410 (2001) 686.
- 532 [3] G. K. Lott, M. J. Rosen, R. R. Hoy, An inexpensive sub-millisecond sys-
533 tem for walking measurements of small animals based on optical com-
534 puter mouse technology, Journal of neuroscience methods 161 (2007)
535 55–61.

- 536 [4] D. A. Dombeck, A. N. Khabbaz, F. Collman, T. L. Adelman, D. W.
537 Tank, Imaging large-scale neural activity with cellular resolution in
538 awake, mobile mice, *Neuron* 56 (2007) 43–57.
- 539 [5] J. D. Seelig, M. E. Chiappe, G. K. Lott, A. Dutta, J. E. Osborne,
540 M. B. Reiser, V. Jayaraman, Two-photon calcium imaging from head-
541 fixed *Drosophila* during optomotor walking behavior, *Nature methods*
542 7 (2010) 535–540.
- 543 [6] R. J. Moore, G. J. Taylor, A. C. Paulk, T. Pearson, B. van Swinderen,
544 M. V. Srinivasan, FicTrac: A visual method for tracking spherical mo-
545 tion and generating fictive animal paths, *Journal of Neuroscience Meth-*
546 *ods* 225 (2014) 106–119.
- 547 [7] H. Haberkern, M. A. Basnak, B. Ahanonu, D. Schauder, J. D. Cohen,
548 M. Bolstad, C. Bruns, V. Jayaraman, On the adaptive behavior of head-
549 fixed flies navigating in two-dimensional, visual virtual reality (2018).
- 550 [8] D. Turner-Evans, S. Wegener, H. Rouault, R. Franconville, T. Wolff,
551 J. D. Seelig, S. Druckmann, V. Jayaraman, Angular velocity integration
552 in a fly heading circuit, *eLife* 6 (2017) e23496.
- 553 [9] B. E. Bayer, An optimum method for two-level rendition of continuous
554 tone pictures, in: *IEEE International Conference on Communications*,
555 June, 1973, volume 26, 1973.
- 556 [10] Urho3D contributors, Urho3d: A cross-platform 2d and 3d game engine,
557 <https://urho3d.github.io/>, 2019.
- 558 [11] B. Palais, R. Palais, Eulers fixed point theorem: The axis of a rotation,
559 *Journal of Fixed Point Theory and Applications* 2 (2007) 215–220.
- 560 [12] E. W. Weisstein, Spherical coordinates. From
561 MathWorld—A Wolfram Web Resource, 2005. URL:
562 <http://mathworld.wolfram.com/SphericalCoordinates.html>,
563 visited on 20.03.2018.
- 564 [13] P. Sturm, Pinhole camera model, in: *Computer Vision*, Springer, 2014,
565 pp. 610–613.

- 566 [14] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools
567 (2000).
- 568 [15] G. Bradski, Camera calibration and 3d reconstruction. From OpenCV Reference Manual, 2016. URL:
569 https://docs.opencv.org/3.2.0/d9/d0c/group__calib3d.html,
570 visited on 15.03.2019.
571
- 572 [16] J. A. Nelder, R. Mead, A simplex method for function minimization,
573 The computer journal 7 (1965) 308–313.
- 574 [17] M. J. Grabowska, J. Steeves, J. Alpay, M. Van De Poll, D. Ertekin,
575 B. van Swinderen, Innate visual preferences and behavioral flexibility
576 in drosophila, Journal of Experimental Biology 221 (2018) jeb185918.
- 577 [18] J. R. Stowers, M. Hofbauer, R. Bastien, J. Griessner, P. Higgins, S. Fa-
578 rooqui, R. M. Fischer, K. Nowikovskiy, W. Haubensak, I. D. Couzin,
579 et al., Virtual reality for freely moving animals, Nature methods 14
580 (2017) 995.
- 581 [19] A. Jönsson, contributors, Angelscript, 2018. URL:
582 <https://www.angelcode.com/angelscript/>, visited on 14.03.2019.
- 583 [20] Urho3D contributors, Urho3d scene model, 2019. URL:
584 https://urho3d.github.io/documentation/1.7/_scene_model.html,
585 visited on 14.03.2019.
- 586 [21] S. K. J. Simen Svale Skogsrud, Grbl, <https://github.com/grbl/grbl>,
587 2009.
- 588 [22] J.-Y. Bouguet, Pyramidal implementation of the affine lucas kanade
589 feature tracker description of the algorithm, Intel Corporation 5 (2001)
590 4.
- 591 [23] G. Farneback, Two-frame motion estimation based on polynomial ex-
592 pansion, in: Scandinavian conference on Image analysis, Springer, 2003,
593 pp. 363–370.
- 594 [24] T. Brox, A. Bruhn, N. Papenberg, J. Weickert, High accuracy optical
595 flow estimation based on a theory for warping, in: European conference
596 on computer vision, Springer, 2004, pp. 25–36.

- 597 [25] C. Zach, T. Pock, H. Bischof, A duality based approach for realtime
598 tv-l 1 optical flow, in: Joint Pattern Recognition Symposium, Springer,
599 2007, pp. 214–223.
- 600 [26] T. Kroeger, R. Timofte, D. Dai, L. Van Gool, Fast optical flow using
601 dense inverse search, in: European Conference on Computer Vision,
602 Springer, 2016, pp. 471–488.
- 603 [27] M. Tao, J. Bai, P. Kohli, S. Paris, Simpleflow: A non-iterative, sublinear
604 optical flow algorithm, in: Computer Graphics Forum, volume 31, Wiley
605 Online Library, 2012, pp. 345–353.
- 606 [28] P. Weinzaepfel, J. Revaud, Z. Harchaoui, C. Schmid, Deepflow: Large
607 displacement optical flow with deep matching, in: Computer Vision
608 (ICCV), 2013 IEEE International Conference on, IEEE, 2013, pp. 1385–
609 1392.
- 610 [29] J. Wulff, M. J. Black, Efficient sparse-to-dense optical flow estimation
611 using a learned basis and layers, in: Proceedings of the IEEE Conference
612 on Computer Vision and Pattern Recognition, 2015, pp. 120–130.

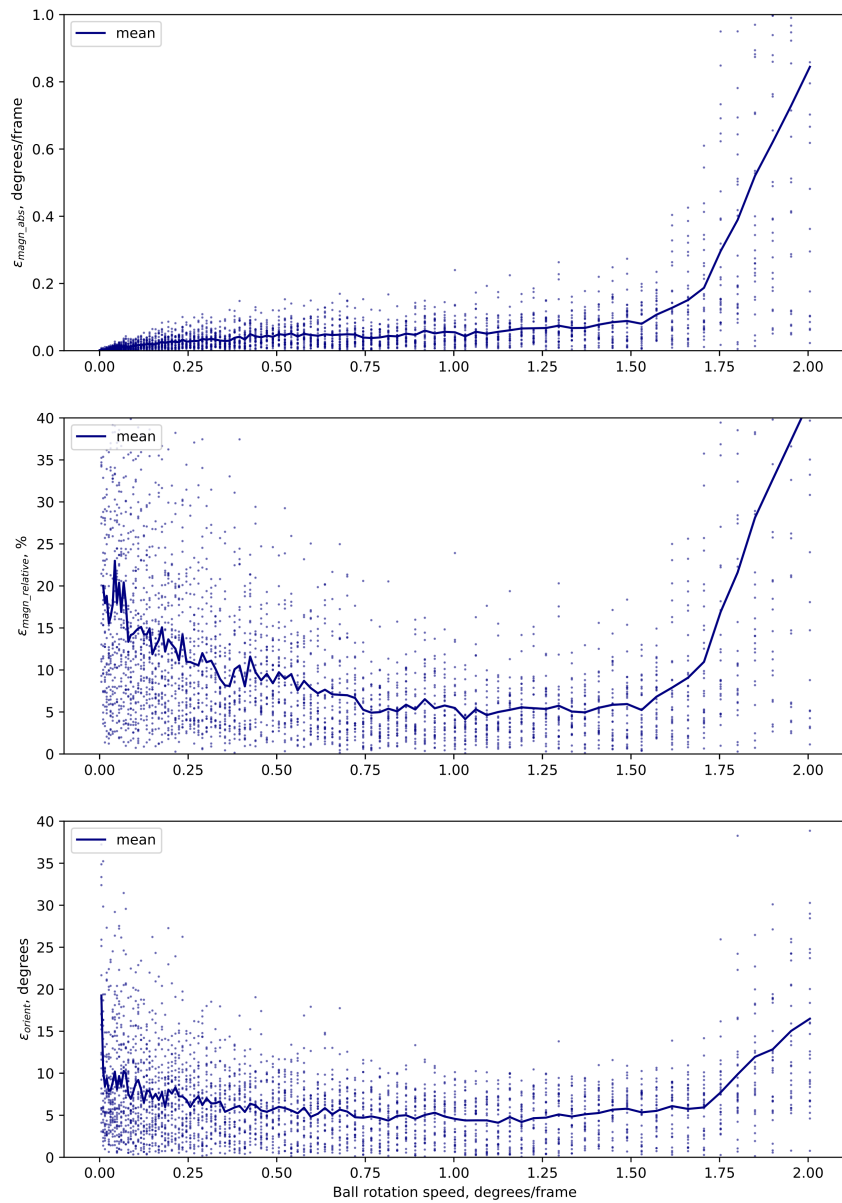


Figure 15: Tracking errors for an example of simulated, accelerated rotations, corresponding to the data shown in Figure 7.