CompoundRay: An open-source tool for high-speed and high-fidelity rendering of compound eyes

Blayze Millward ^{1*}, Steve Maddock ¹, Michael Mangan ^{1*}

*For correspondence:

b.f.millward@sheffield.ac.uk;
mmangan@sheffield.ac.uk

¹Dept of Computer Science, The University of Sheffield, Sheffield, UK

[†]First author

Abstract Revealing the functioning of non-standard visual systems such as compound eyes is of interest to biologists and engineers alike. A key investigative method is to replicate the sensory apparatus using artificial systems, allowing for investigation of the visual information that drives animal behaviour when exposed to environmental cues. To date, ‘Compound Eye Models’ (CEMs) have largely explored the impact of features such as spectral sensitivity, field of view, and angular resolution on behaviour. Yet, the role of shape and overall structure have been largely overlooked due to modelling complexity. However, modern real-time raytracing technologies are enabling the construction of a new generation of computationally fast, high-fidelity CEMs. This work introduces new open-source CEM software (*CompoundRay*) alongside standardised usage techniques, while also discussing the difficulties inherent with visual data display and analysis of compound eye perceptual data. *CompoundRay* is capable of accurately rendering the visual perspective of a desert ant at over 5,000 frames per second in a 3D mapped natural environment. It supports ommatidial arrangements at arbitrary positions with per-ommatidial heterogeneity.

Introduction

Insects visually solve an array of complex problems including the detection and tracking of fast moving prey *Wiederman et al. (2017)*, long distance navigation *Wehner (2020)* and even 3D depth estimation *Nityananda et al. (2018)*. These capabilities are realised using sensory apparatus that is fundamentally different from those of mammals. Therefore revealing the functional properties of the insect visual system offers insights for biologists as well as inspiration for engineers looking to develop novel artificial imaging systems *Land and Fernald (1992)*; *Land (1997)*; *Arendt (2003)*; *Song et al. (2013)*.

Arthropods possess two primary visual sensors known as compound eyes. Each eye is constructed from a patchwork of self-contained light-sensing structures known as ommatidia, each featuring a lens, a light guide and a cluster of photosensitive cells (Figure 1a). Ommatidia are physically interlocked with their neighbours, together forming a bulbous outer structure (the compound eye itself) that can vary in size, shape, and curvature, offering a range of adaptations for particular tasks and environments *Land and Nilsson (2002)* (Figure 1b). The properties of the lens and photo-receptors are fixed for individual ommatidia but can vary across regions of an individual's eye *Meyer and Labhart (1993)* as well as between individuals of different castes *Collett and Land (1975)* and species *Land (1989)*. This arrangement of independent, interlocked light sensing elements and lenses differs greatly from the mammalian system that utilises a single lens to project a high-resolution image onto a retina.

Creation of compound eye models (CEMs), in both hardware or software, are a well-established

Figure 1. **a.** A diagram of a single ommatidium, seen with lensing apparatus at the top that guides light into the photo-sensitive cells below **b.** Images of real compound eyes consisting of hundreds to thousands of ommatidia. Upper right and lower image credits *Matthew Barber*. Upper Left image credit *Moussa Direct Ltd.* under the CC BY-SA 3.0 license. **c.** Previous compound eye-view recreation methods (*Neumann (2002)*; *Polster et al. (2018)*; *Wystrach et al. (2016)*, top to bottom).

mechanism to explore the information provided by compound eyes and assess their impact on behaviour. Insights derived from this methodology include demonstration of the benefits of a wide field of view and low resolution for visual navigation *Zeil et al. (2003)*; *Vardy and Moller (2005)*; *Mangan and Webb (2009)*; *Wystrach et al. (2016)*, and the role played by non-visible light sensing for place recognition *MÖLLER (2002)*; *Stone et al. (2014)*; *Differt and Möller (2016)* and direction sensing *Gkanias et al. (2019)*; *Lambrinos et al. (1997)*. Yet, simulated CEMs tend to suffer from a common design shortcoming that limits their ability to accurately replicate insect vision. Specifically, despite differences in the sampling techniques used (e.g. see *Baddeley et al. (2012)*; *Mangan and Webb (2009)* for custom sampling approaches, *Neumann (2002)*; *Basten and Mallot (2010)* for rendering based cubemapping CEMs, and *Polster et al. (2018)* for ray-casting methods) all contemporary CEMs sample from a single nodal point. In contrast, the distributed arrangement of ommatidia on distinct 3D eye surfaces provides insects with a multi-nodal system that generates different information for different eye shapes.

To facilitate exploration of such features, an ideal rendering system would allow light to be sampled from different 3D locations through individually configured ommatidia replicating the complex structure of real compound eyes. High-fidelity compound-vision rendering engines with some of these features (though notably slower than real-time) were developed previously *Giger (1996)*; *Collins (1998)*; *Polster et al. (2018)*, but were not widely adopted. Such issues are rapidly diminishing as dedicated ray-casting hardware emerges (e.g. Nvidia (Santa Clara, California, United States) RTX series GPUs *Purcell et al. (2005)*; *Burgess (2020)*) that allows for the capture of visual data from multiple locations in parallel at high speed. Such systems present an ideal tool to replicate insect vision in unprecedented accuracy fast enough to allow effective exploration of the compound eye design space itself *Millward et al. (2020)*. A next-generation insect eye renderer should:

1. Allow for the arrangement of an arbitrary number of ommatidia at arbitrary 3D points.
2. Allow for the configuration of individual ommatidial properties (such as lensing arrangement or spectral sensitivity).
3. Perform beyond real-time to allow exploration of the design space.

This paper presents a ray-casting based renderer, *CompoundRay*, that leverages modern hardware-accelerated ray-tracing graphics pipelines to capture all three of these criteria, allowing researchers in the fields of compound vision and biorobotics to quickly explore the impact varying eye designs have on an autonomous agent's ability to perceive the world.

Figure 2. How differing rendering approaches form differing focal points. **a.** In 3D raycasting-based rendering rays are emitted into a scene and *sample* the 3D geometry they contact, potentially forming many focal points. **b.** In 3D projection-based rendering, the polygons that comprise the 3D geometry are projected down to a viewing plane, through one focal point. **c.** A compound eye requires many focal points in order to image its surroundings.

Materials and Methods

Raycasting-based Insect Eye Renderer

The majority of real-time rendering approaches are projection-based approaches inspired by the simple *pin-hole* camera. As Figure 2b shows, their images are derived by directly projecting the faces and vertices of the scene's 3D geometry through a singular focal point and on to a projection plane. However, this approach (*polygon projection*) lacks the ability to simulate real-world light transport (and thus, lensing systems) accurately – in particular the spreading of light through reflection, refraction and diffraction, all of which have to be approximated using resource-heavy workarounds such as shadow maps *Stamminger and Drettakis (2002)*, reflection maps *Miller and Hoffman (1984)* or other lookup maps for large precomputed data *Lopes and Fernandes (2014)*; *Schäfer et al. (2012)*.

In the real world, practically all light has been subject to at least one of these properties – for instance, objects in shadow are still visible due to interreflections *Langer (1999)*, and glass or heat gradients can act as refracting lenses *Primmerman et al. (1991)*. A polygon projection approach will struggle with these optical phenomena as the process directly transforms the faces of the 3D object from the scene onto the camera's view plane, in effect applying a projection transform onto a plane or other regular surface forming a singular focal point (Figure 2b). In contrast to this, lenses or reflective surfaces in a real scene effectively act to create additional, sometimes complex, projection surfaces with multiple focal points that combine to form the final composite-perspective view seen by an observer (Figure 2a).

Ray-based methods offer an alternative to projective transform rendering: rays are sent out from the virtual view-point, simulating – in reverse – the paths of photons from the scene into the cone of vision, accumulating colour from the surface they interact with (Figure 2a). This allows for surfaces such as mirrors and lenses to accurately reflect light being rendered. The term *ray-based methods* here is used as an umbrella term for all rendering approaches that primarily use the intersection of rays and a scene in order to generate an image. In particular, we refer to *ray casting* as the act of sampling a scene using a single ray (as per *Roth (1982)*) – a process that CompoundRay performs multiple times from each ommatidium – and *ray tracing* as the act of using multiple **recursive** ray casts to simulate light bouncing around a scene and off of objects within, similar to early work in the field of shading *Appel (1968)*; *Whitted et al. (1979)*.

These approaches are incredibly computationally complex, as each ray (of which there can be thousands or even millions per pixel) needs to be tested against every object in the scene (which

can itself be composed of millions of objects) to see if they intersect. Furthermore, any reflective objects require further intersections as they reflect light rays back into the scene. Historically, these ray-based methods have been limited to use in offline cases where individual frames of an animation can take many hours to render, such as in cinema *Christensen et al. (2018)*, and prior to that in architectural and design drafting *Appel (1968)*; *Roth (1982)*. In recent years, however, graphics processing units (GPUs) have been increasing in capability. In order to better capture the photo-realistic offerings of ray-based rendering methods, GPU manufacturers have introduced dedicated programmable ray-tracing hardware into their graphics pipelines *Purcell et al. (2005)*. These so-called "RT cores" allow for efficient parallel triangle-line intersection to be performed, allowing for billions of rays to be cast – in real-time – into scenes even as complex as those found in modern game environments.

As compound eyes consist of a complex lensing structure formed over a non-uniform surface, they naturally form a multitude of focal points across a surface (Figure 2c). These projection surfaces are often unique and varied, meaning it is practically infeasible to find a single projective transform to produce the appropriate composite view in a single projection-based operation. As a result, ray-based methods become the natural choice for simulating compound vision, as opposed to the more commonly seen projection-based methods. By utilising modern hardware it is possible to effectively capture the optical features of the compound eye from the ommatidia up, in a high-fidelity yet performant way.

Modelling Individual Ommatidia

As the compound eyes of insects contain many hundreds or even thousands of lenses (Figure 1b), and each acts to focus light to its own point (forming a unique perspective), ray-based methods become the natural technology to use to implement a simulator that can capture their unique sensory abilities. Here, a single ommatidia is first examined, simulated and then integrated into a complete compound eye simulation.

Each individual ommatidium in a compound eye captures light from a cone of vision (defined by the *ommatidial acceptance angle*), much in the way that our own eyes observe only a forward cone of the world around us. However, in the case of the ommatidium all light captured within the cone of vision is focused to a singular point on a small photoreceptor cluster, rather than the many millions of photoreceptors in the human eye that allow for the high-resolution image that we experience – in this way, the vision of a singular ommatidium is more akin to the singular averaged colour of all that is visible within its given cone of vision – it's *sampling domain*, much as a pixel from a photo is the average colour of all that lies 'behind' it.

All objects within the ommatidium's sampling domain will invariably impart influence on the final amount of sensed light, the total of which must be integrated, in much the same way that the *rendering equation* *Kajiya (1986)*; *Immel et al. (1986)* is in traditional computer graphics – using a Monte Carlo approximation *Kajiya (1986)*. Within a computer it is practically impossible to account for all infinite light rays converging on to a point, as such the amount of light received across the angular space of the viewing cone (which must be accrued in alignment with the the ommatidium's *sampling function* – as the lens gathers more light from forward angles than those diverging away from the ommatidial axis) can be *sampled* a discrete number of times to approximate the entire sampling domain. Increasing the sample count will result in a more accurate estimate of the light converging on the point.

Previous works *Collins (1998)*; *Polster et al. (2018)* have implemented this approximation of the light sampling function by assuming the influence of individual light rays, distributed statically across the sampling cone, is modulated dependent on the angle to the ommatidial axis via a Gaussian function with a full width at half maximum (FWHM) equal to the acceptance angle of the ommatidium, as seen in Figure 3a. *CompoundRay* also uses the Gaussian function as a basis for sampling modulation, however, unlike other works it is not approximated via a static sampling pattern with weighting biasing the forward direction (as is seen in 3a:i). Instead, sampling *direction* is modulated

Figure 3. **a.** The gaussian sampling curve (red) with a full width at half maximum (FWHM) equal to the acceptance angle of each ommatidium. **i.** Example statically distributed sampling rays, length indicates weight. **ii.** Example stochastically distributed sampling rays, equal weight, distributed according to gaussian distribution. **b.** A concentric circular static sampling pattern (red dots) aliases away the 3 concentric dark-grey rings present on a surface. **c.** The per-sample colours as seen from a single ommatidium mapping to the samples in **b**, note the lack of any influence from the dark-grey concentric rings, which have been aliased out.

and chosen at random in relation to the Gaussian probability curve of the sampling function on a temporal, frame-by-frame basis. Each sample is assumed to have an equal weight, with the visual input to a single ommatidium then being formed as the average of all samples (Figure 3a:ii).

Temporal modulation eliminates the potential of structured aliasing forming, as demonstrated in Figures 3b&c whereby, for instance in the case where per-ommatidium sampling is performed using static concentric circles, circular patterns matching the frequency would be aliased out of the ommatidium's sampling domain if occurring out-of-phase with the sampling points. Increasing per-ommatidium sampling rate *can* ease these aliasing problems (even in the case of a static sampling pattern) at close range, but only serves to delay triggering structured aliasing until further distances where structural frequency matches sampling pattern again.

Choosing to stochastically vary the sampled direction on a frame-by-frame basis removes the structured nature of any single frame's aliasing, as that same view point will generate different aliasing artifacts on the next frame, having the net effect of reducing structured aliasing when considering sequences of frames (Figure 3c) rather than single frames at a time, as opposed to any static sampling method (Figure 3b). One benefit of this approach is that averaging renderings from the same perspective becomes synonymous with an increased sampling rate, indeed, this is how the renderer increases sampling rate internally: by increasing the rendering volume for the compound perspective, effectively batch-rendering and averaging frames from the same point in time. However, as each ommatidium will sample the simulated environment differently for every frame rendered, small variations in the colour received at any given ommatidium occur over time. This point is explored further in the results section.

In the natural compound eye, light focused from each ommatidium's sampling domain is transported via a crystalline cone and one or more rhabdomeres toward the receiving photoreceptor cluster – a process that has been modelled in past works *Song et al. (2009)*. However, modelling the biophysical intricacies of the internal ommatidium structure is considered beyond the scope of this software, and is considered a post-processing step that could be run on the generated image, considering each pixel intensity as a likelihood measure of photon arrival at an ommatidium.

181 From single ommatidia to full compound eye
182 By arranging the simulated ommatidia within 3D space so as to mimic the way real compound eyes
183 are arranged, an image can be generated that captures the full compound eye view. For this task,
184 per-ommatidium facet diameter and acceptance angle can be used to generate the cone of influ-
185 ence of each ommatidium, and simulated ommatidia can be placed at any position and orientation
186 within the environment. As the system is based on ray-tracing techniques, orientation and position
187 can be set with a single per-ommatidium data point, effectively allowing each ommatidium to act
188 independently of each other. This has the benefit of allowing the system to simulate any type of
189 eye surface shape, and even multiple eyes simultaneously, fulfilling the first of the three criteria:
190 *Allowing for the arrangement of an arbitrary number of ommatidia at arbitrary 3D points.*

191 **The CompoundRay Software Pipeline**

192 The renderer is written in C++ and the Nvidia CUDA GPU programming language and allows for a 3D
193 environment representing a given experimental setup to be rendered from the perspective of an
194 insect. The core of the renderer runs in parallel on the GPU (the *device*) as a series of CUDA shader
195 programs, which are driven by a C++ program running on the host computer (the *host*). Environ-
196 ments are stored in *GL Transmission Format* (glTF) *Robinet et al. (2014)* files consisting of 3D objects
197 and cameras, the latter of which can be of two types: compound (structured sets of ommatidia)
198 or traditional (perspective, orthographic and panoramic). Traditional cameras are implemented to
199 aid the user in the design and analysis of their assembled 3D environment. Compound cameras
200 contain all relevant information for rendering a view of the environment from the perspective of
201 a compound eye. Each camera stores the information required for rendering its view (such as
202 its orientation, position, field of view or ommatidial structure) in an on-device **data record** data
203 structure.

204 Figure 4 shows the operational design of the renderer from the device side. The renderer is built
205 on top of Nvidia's *Optix Parker et al. (2010)* raytracing framework, which is a **pipeline**-based system:
206 a pipeline is assembled, which then allows for parallel per-pixel rendering of the environment.
207 A **pipeline** consists of a **ray generator shader**, a **geometry acceleration structure (GAS)** and
208 numerous **material shaders**. A shader is a small program that is designed to be run in a massively-
209 parallel fashion. In a typical application, an individual shader program will be loaded onto a GPU
210 and many thousands of instances of the program will be executed, with varying input parameters.
211 The returned values of each instance are then returned to the 'host-side' of the application, allowing
212 for the optimisation of tasks that are well-suited to parallelisation.

213 A **ray generator shader** spawns rays to sample the environment with – these are dependent
214 on the type, position and orientation of the chosen camera. In the case of a panoramic camera (in
215 the display pipeline), for instance, rays are spawned in a spherical manner around a central point
216 (the *position* component of the currently selected camera's ray generation configuration record).
217 In the case of a compound eye (in the ommatidial pipeline), each ray is generated as a function
218 of the relative position and direction of a single ommatidium to the eye's position and orientation
219 (note that the "eye"-type data records in Figure 4 contain links to per-eye ommatidial arrays that
220 define the ommatidial configuration of the eye).

221 A **GAS** stores the 3D geometry of the scene on the device in a format optimised for ray-geometry
222 intersection, and also provides functions to perform these intersection tests. Upon the intersec-
223 tion of a generated ray and the environment the **material shader** associated with the intersected
224 geometry is run. These material shaders are responsible for the calculation and retrieval of the
225 correct colour for the given geometry. For example, if a ray were to fall onto a blade of grass, the
226 material shader would be expected to compute the green colour that could be seen, taking into
227 account any light that might be affecting the scene. Currently CompoundRay implements only a
228 simple texture lookup, interpolating the nearest pixels of the texture associated with the geometry
229 intersected.

230 Two pipelines are used in the renderer: an **ommatidial pipeline** and a **display pipeline**. The

Figure 4. GPU-side renderer operational diagram. Featured are the joined *ommatidial* and *display* rendering pipelines (including display buffers) as well as the on-device per-camera configuration records and related memory allocations. Data-structure memory allocations are shown in grey (indicating in-place data) or purple (indicating pointers to other sections of data). Vertical stacks of memory indicate contiguously allocated structures. Data-structures marked with an orange square are automatically updated as their host-side copies change; those with red circles are manually updated via API calls. Circled numbers 1 to 4 indicate the path of processing through both pipelines required to display a compound eye view.

231 **ommatidial pipeline** handles sampling of the environment through the currently active com-
232 pound eye, saving all samples from all ommatidia to a buffer stored within the eye's "Raw Om-
233 matidial View" portion of its data record. First it generates per-ommatidium rays (Figure 4-1) via
234 instances of the ommatidial ray generation shader. It then uses these to sample the environment
235 through the GAS and appropriate material shaders, finally storing each ommatidium's view as a
236 vector of single-pixel tri-colour samples in the raw ommatidial view buffer (Figure 4-2).

237 Conversely, the **display pipeline** handles the generation of the user-facing **display buffer** (Fig-
238 ure 4-4). In the case of generating a compound eye view display, *ommatidial projector* shaders are
239 spawned (one per output pixel) and used to lookup (Figure 4-3) and average the appropriate values
240 from the per-camera raw ommatidial view buffer. These values are then re-projected into a human-
241 interpretable view – in the case of the image seen at Figure 4-4, using a spherical orientation-wise
242 Voronoi diagram projected into 2D by mapping angular yaw and pitch to 2D (x,y) coordinates. How-
243 ever for practical applications, this re-projection can be to a simple vector of all ommatidial values,
244 allowing an algorithm to examine the data on a per-ommatidium basis. In the case where the cur-
245 rent camera is non-compound the display pipeline becomes the only active pipeline, handling the
246 initiation of ray generation, GAS intersection and materials shading for any given camera view in a
247 single pass. Both pipelines share a common geometry acceleration structure and material shaders
248 so as to save device memory.

249 A full compound eye rendering (as indicated via numbered red circles in Figure 4) consists of first
250 rendering the ommatidial view via the ommatidial pipeline to the current camera's raw ommatidial
251 view buffer using the associated eye data record. After this, the raw ommatidial view buffer is
252 re-projected onto the display buffer as a human-interpretable view via the ommatidial projector
253 shaders within the display pipeline. Alternately, rendering using a traditional camera skips steps 1
254 and 2 and instead simply renders to the display view by passing fully through the display pipeline
255 using the data record associated with the selected camera.

256 Scene Composure and Use

257 Using the glTF file format for scene storage allows the experimental configuration of 3D models,
258 surface textures and camera poses to be packed into a single file (although per-eye ommatidial
259 configurations are stored separately as CSVs and linked to within the glTF file). The glTF format
260 is a JSON-based human-readable file format for 3D scene storage that is readily supported by a
261 number of popular 3D editors and has allowances for extensions within its specification. As such,
262 all configurable parts of the scene that relate to CompoundRay are placed within each camera's
263 reserved "extras" property. The renderer is programmed to ingest these extra properties, expand-
264 ing the scene definition beyond standard glTF without compromising the readability of scene files
265 by other third-party 3D modelling suites.

266 While the software comes packaged with a stand-alone executable that can load and render
267 compound-eye scenes, this is not the most powerful way of using the tool. When compiled, a
268 shared object library is also generated that can then be called from more accessible languages,
269 such as Python *Van Rossum and Drake (2009)*. Through the use of this, Python's *ctypes* system
270 and a helper library that is bundled with the code, it can be used directly with Python and the
271 Numpy *Harris et al. (2020)* mathematical framework. This allows for a significantly wider exper-
272 imental scope through automated configuration – for instance, one can programmatically move
273 an insect eye around a simulated world extracting views directly from the renderer to be used
274 alongside neural modelling frameworks accessible under Python. This is the recommended way
275 of using the rendering system due to the increased utility that it affords.

276 Results

277 In this section two 3D environments are used to benchmark and examine the performance of *Com-*
278 *poundRay*. The first is a lab environment inspired by those used in insect cognitive experiments *Of-*
279 *stad et al. (2011)* and the second is a 3D scan of a real-world natural environment covering an area

Figure 5. Rendering from eyes with ommatidia placed on arbitrary surfaces in one lab environment and one 3D scan of a natural outdoor environment. **a.** Spherically uniformly distributed ommatidia. **b.** Ommatidia clumped to form a horizontal acute zone **c.** A design inspired by the *Erbenochile erbeni*.

of a number of square kilometers. The natural environment was constructed using structure-from-motion photogrammetry using photos captured from a drone over rural Harpenden, England (3D model subject to upcoming publication, available via contact of Dr. Joe Woodgate, Queen Mary University of London).

Criterion 1: Arbitrary arrangements of Ommatidia

As stated in criterion 1 of the required feature list (defined in the introduction), the renderer must support arbitrary arrangements of ommatidia within 3D space. Figure 5 shows two different environments rendered using three differing eye designs, from a simple spherical model with spherical uniformly distributed ommatidia that more closely align with current compound eye rendering methods, to a hypothetical arbitrary arrangement similar to that of the now long-extinct *Erbenochile erbeni*. While Figures 5a&b are still possible to simulate using traditional projection-based methods due to their ommatidial axes converging onto one central spot, 5c demonstrates an eye with multiple unique focal points across an irregular surface.

Criterion 2: Inhomogeneous Ommatidial Properties

Criterion 2 states that it should be possible to specify heterogeneous ommatidial optical properties across an eye. As demonstrated in Figure 6, the renderer achieves this by allowing for shaping of the acceptance cone of each ommatidium. In doing so, Figure 6 also demonstrates the importance of heterogeneous ommatidial configurations.

Figures 6b&c show a non-uniform eye design with homogeneous, globally-identical, ommatidial acceptance angles. It can be seen that in b, where the acceptance angle is lower, aliasing occurs as objects are only partially observed through any given ommatidium, causing blind spots to form, introducing noise into the image. Conversely, in c, where the acceptance angle is large, the image becomes blurry, with each ommatidium over-sampling portions of its neighbour's cone of vision. Sub-figure a, however, demonstrates a heterogeneous ommatidial distribution, in which the acceptance angles toward the dorsal and ventral regions of the eye are larger compared with the

Figure 6. A single environment as viewed through: **a.** An eye with heterogeneous ommatidial acceptance angles. **b&c.** Eye designs with homogeneous ommatidial acceptance angles, note the sharp aliased edges in **b** and the blurred edges in **c**. Above each eye view is a diagram demonstrating the type of coverage allowed by each design into the eye's spherical field of view, showing the under- and over-sampled nature of **b** & **c** compared to **a**.

angles of those around the eye's horizontal acute zone. This mirrors what is seen in nature, where ommatidial field of view is seen to vary in order to encompass the entirety of an insect's spherical visual field *Land and Nilsson (2002)*. As can be seen in the generated image, this is important as it appropriately avoids creating blind-spots in the dorsal and ventral regions while also avoiding over-sampling (blurring) in the horizontally acute region, resulting in a much clearer picture of the observed environment while also minimising the required number of ommatidia. This maximisation of utility while minimising structural complexity is a hallmark of compound eye design.

Criterion 3: Speed

Minimum Sample Count

In order for rapid exploration and experimentation of the design space of compound eyes, criterion 3 (as defined in the introduction) requires that any such rendering system should be able to perform at real-time or faster. However, performance of the system is dependent on the number of rays required to render a given scene from a given eye; higher total ray counts will require more compute power. As discussed in the section "*Modelling Individual Ommatidia*", per-ommatidial sampling choice can have a dramatic impact on the image generated, meaning that lower ray counts – while increasing performance – will decrease the accuracy of the image produced. As a result, it is important to be able to define some minimum number of samples required for any given eye in any given environment in a reliable and repeatable manner. Previous works *Polster et al. (2018)* have attempted to establish a baseline number of samples required per ommatidium before additional samples become redundant only via qualitative visual analysis of the results produced. Here we perform quantitative analysis of images produced by the renderer in order to more definitively

326 define this baseline.

327 Unlike previous works, the temporally stochastic nature of *CompoundRays*' sampling approach
 328 can be used to aid the measurement of impact that sampling rate has on the final image. By
 329 measuring the per-ommatidium spread (here, standard deviation) of the Euclidean distance of the
 330 received colour as plotted in RGB colour space over a range of frames captured from a static com-
 331 pound camera, the precision of the light sampling function approximation can be measured. To
 332 control for the varying field of views (FOV) of each ommatidium (which impact the number of sam-
 333 ples required, as larger FOVs require more sampling rays to accurately capture the ommatidium's
 334 sampling domain), we normalize the measure of spread recorded at each ommatidium against its
 335 degree of coverage, in steradians.

Figure 7. A single viewpoint viewed with 1 through to 700 samples per ommatidium, left to right, logarithmic scale. **Top:** how changing sample count changes the generated image. Note that the left side (with lower per-ommatidial samples) is more “jagged” due to aliasing. **Middle:** relative standard deviations of each ommatidium across 1000 frames, where difference is measured as the Euclidean distance between the two colours. **Bottom:** plot of the average standard deviation of an eye per sample rays per ommatidium, normalised to one steradian. Standard deviations decrease both on average (**bottom**) and per-ommatidia (**middle**) as per-ommatidium sampling rate increases.

336 Figure 7 shows this measure of spread within the compound eye across a range of sample-
 337 rays-per-ommatidium, from 1 to 700. As the number of samples increases the spread decreases
 338 logarithmically. We propose iteratively increasing samples until the maximum spread is below
 339 some arbitrarily defined threshold value. Here, we use a threshold of 1%, meaning that the max-
 340 imum expected standard deviation of the most deviant ommatidium on an eye (normalised to
 341 one steradian) should be no greater than 1% of the maximum difference between two received
 342 colours, here defined as the length of vector $[255, 255, 255]$. This method allows for standardised,
 343 eye-independent configuration of per-ommatidial sampling rate.

344 However, the required number of samples per eye is highly positively correlated with the visual
 345 frequency and intensity of the simulated environment at any given point, with locations exhibiting

Figure 8. Standard deviation, averaged over all ommatidia in an eye, mapped over the natural and lab environments, showing a range of “hotspots” of high variance in renderings. These hotspots must be accounted for when calculating minimum sample ray count. **Inserts bottom left & right:** the maximum (over all ommatidia) standard deviation recorded in the eye at each location.

high contrast seen from a distance requiring higher sampling rates in order to properly resolve. This can be seen in the the skyline in Figure 7, which remains dominant throughout the majority of the sampling rates, consistently forming the source of the highest spread. Figure 8 shows how the perceived variance changes over two example environments. The varying environment presents an obstacle in our attempts to standardise sampling rates, as deriving a suitable minimum sampling rate at one point in an environment – for example, where the visual frequency was, on average, low – could lead to aliasing occurring in more visually complex areas of the same environment, potentially introducing bias for or against certain areas if analysis on the generated data were sensitive to the variance introduced.

One method to attempt to control for this would be to search the environment for the location of highest variance and perform minimum sampling rate calculations there. This is the method this work follows, using a genetic algorithm to find the location of highest variance before iteratively increasing sampling rate until a frame-wise standard deviation of less than 1% is observed in the single ommatidium with the highest standard deviation. Alternatively, sampling rates could be updated in an online fashion by storing a rolling average of the maximum sample spread and increasing and decreasing sampling rate to keep the average below the considered acceptable spread percentage (here 1%).

Performance

Figure 9 shows the rendering speed, in frames-per-second, against the number of sample rays being emitted in each frame when running using a selection of high-end consumer-grade graphics cards and our two sample environments. Marked on the graph are the total per-frame sample counts required by common insect eyes, chosen to encourage a maximum per-ommatidium standard deviation of 1%.

As can be seen, performance improves significantly when the renderer is able to utilise the next-generation ray-tracing hardware available in the newer RTX series of NVidia graphics cards,

Figure 9. The average FPS (500 samples) per total number of rays dispatched into the scene for 3 different graphics cards in the two environments depicted in Figure 5. Marked are expected FPS values for a desert ant *Schwarz et al. (2011)*, honey bee *Greiner et al. (2004)* and dragonfly *Labhart and Nilsson (1995)*; *Land and Nilsson (2002)*.

consistently out-performing older cards in the larger natural environment, and outperforming in the smaller lab environment when using more than 30,000 rays. Counter-intuitively, the previous generation of NVidia graphics cards were found to out-perform the RTX series in the (relatively small) lab environment at lower ray counts (below 30,000). This appears to be due to static overhead involved with the use of the RTX series' ray-tracing cores, which may also be the cause of the small positive trend seen between 1,000 and 8,000 samples when using the 2080Ti. Another potentially counter-intuitive result that can be seen here is the higher performance of the dragonfly eye in the natural (high polygon) environment verses in the low-polygon laboratory environment. This is because the laboratory environment produces significantly higher visual contrast (emphasised in Figure 8) due to its stark black/white colour scheme, resulting in a requirement of almost double the number of rays per steradian to target 1% standard deviation, resulting in higher ray counts overall, particularly in models with a high number of ommatidia.

Challenges of Visualisation and Eye Comparison

Compound eyes are inherently different from our own, and the CompoundRay rendering system allows for data collection on arbitrary compound surfaces. With this comes the risk of anthropomorphising the data received. At its base level, the eye data gathered is most 'true' when considered as a single vector of ommatidial samples, much in the same way that the neural systems of any insect will interpret the data, perhaps with some spatial relational information encoded. In contrast, all the images that we have presented here have been projected onto a 2D plane in order to be displayed. Great care must be taken when considering these projections, as they can introduce biases that may be easy to overlook. For instance, in Figures 5a&b, orientation-wise spherical projection mapping was used to artificially place the ommatidia across the 2D spaces, forming panoramic Voronoi plots of observed light at each ommatidium. In these projections, much as is seen in 2D projections of Earth, the points at the poles of the image are artificially inflated, giving a bias in terms of raw surface area affected toward ommatidia present in the ventral and dorsal regions of the eye.

In the case of the first two eye designs shown in Figure 5 this projection bias can be controlled for by using a projection method that controls for the area of a projected sphere into 2D space, such as the HEALPix *Górski et al. (2005)*, Mollweide or other such equal-area projection schemes *Sousa et al. (2019)*. However, this does not work for all eye designs. The design present in Figure 5c is inspired by that of the *Erbenochile erbeni* (1b, upper left), which exhibits a largely cylindrical shape. For this eye design, mapping onto a sphere (even using equal-area projection) would introduce bias, as either near-parallel ommatidia would occlude each another in the case of orientation-wise projection (as is seen happening in Figure 1c) or ommatidia further from the centre would be biased against as they are moved artificially closer together through projection onto the sphere than would happen to ommatidia on the same eye closer to the centre of the projection sphere.

These projection biases demonstrate how fundamentally difficult it is to compare eye designs that do not share an underlying surface structure, unless those surfaces happen to be affine transformations of one another (they are *affinely equivalent*). For any surfaces that are affinely equivalent, the surface itself can be used as a projection surface. In this way, for instance, the outputs of two cubic eyes could be compared to each other by performing all comparison calculations on the cube's surface, interpolating between points where an ommatidia exists on one eye but not another. It is trivial to compare compound eye images from eyes with the same structure in terms of ommatidial placement, and eyes with surfaces that are affinely equivalent can also be compared, however further precautions must be taken when comparing the images from two or more eyes that do not have compatible projection surfaces so as not to unduly bias a portion of one surface over any other.

Discussion

This paper has introduced a new compound eye perspective rendering software, *CompoundRay*, that addresses the need *Millward et al. (2020)*; *Taylor and Baird (2017)*; *Stürzl et al. (2015)* for a principled, fast, compound-perspective rendering pipeline. In particular, the tool supports arbitrary instantiation of heterogeneous ommatidia at any place in space, and can perform rendering tasks at framerates on the order of thousands per second using consumer-grade graphics hardware. It has demonstrated the utility afforded by the software and highlighted the importance of using higher-fidelity compound-eye perspective rendering systems by demonstrating the visual differences introduced by altering these often unaltered traits. It has also set a grounding for reasonable use of the software to ensure reproducibility and reduce biasing introduced as a result of the variety present in simulated environments and compound eye designs while warning of potential dangers that might arise during data analysis. With the introduction of the *CompoundRay* rendering library it is possible to simulate in a timely manner the visual experience of structurally diverse compound eyes in complex environments, digitally captured or otherwise manually constructed.

This work lays the technical foundations for future research into elements of compound eye design that have previously been given little consideration. In particular, *CompoundRay* can act as the previously missing step to integration of recently catalogued real eye designs *Baird and Taylor (2017)* and increasing capture of insect environments *Stürzl et al. (2015)*; *Risse et al. (2018)* and routes *Risse et al. (2017)*. Further to this, it can be used in the further exploration of recently emerging work surrounding microsaccadic sampling in drosophila *Juusola et al. (2017)*, as well as aid with similar vision-centric neurological studies *Kemppainen et al. (2021)*; *Wystrach et al. (2016)*; *Viollet (2014)*. On top of the potential biological insights *CompoundRay* may be able to offer, the ability to rapidly search the design space of compound eyes will enable faster iteration on insect-inspired physical sensors *Floreano et al. (2013)*; *Song et al. (2013)*, allowing for design experimentation before manufacture. The insights gained from more thoroughly exploring the insect visual perspective—and its design relative to visual feature extraction—will help guide the development of artificial visual systems by considering not only visual post-processing steps, but also the intrinsic structure and design of the image retrieval system itself.

The CompoundRay software is open-sourced on GitHub (at <https://github.com/manganlab/eye-renderer>) to allow for easy and systematic comparisons to be made between varying models of insect behaviour, without the introduction of implementation-specific differences in rendering of the insect perspective. In the future, the library can easily be extended to include further extensions to the modelling environment such as shadows, polarisation *Zeil et al. (2014)* or solar compass patterning in the virtual sky *Gkanias et al. (2019)*.

Acknowledgements

Thanks to Dr. James Knight (University of Sussex) for his input and feedback on the manuscript, and Dr. Joe Woodgate (Queen Mary University of London) for provision of the natural environment 3D model. Graphics cards were supplied by the Brains on Board research project (EP/P006094/1) and project partner NVidia. Grant funding was provided by EPSRC awards EP/P006094/1 and EP/S030964/1.

References

- Appel A.** Some techniques for shading machine renderings of solids. In: *AFIPS '68 (Spring): Proceedings of the April 30-May 2, 1968, spring joint computer conference* New York, New York, USA: ACM Press; 1968. p. 37–45. <http://portal.acm.org/citation.cfm?doid=1468075.1468082>, doi: 10.1145/1468075.1468082.
- Arendt D.** Evolution of eyes and photoreceptor cell types. *The International Journal of Developmental Biology*. 2003; 47(7/8):563–571.
- Baddeley B**, Graham P, Husbands P, Philippides A. A Model of Ant Route Navigation Driven by Scene Familiarity. *PLoS Computational Biology*. 2012 1; 8(1):e1002336. <https://dx.plos.org/10.1371/journal.pcbi.1002336>, doi: 10.1371/journal.pcbi.1002336.
- Baird E**, Taylor G. X-ray micro computed-tomography. *Current Biology*. 2017; 27:R289–R291. www.idoimaging.com, doi: 10.1016/j.cub.2017.01.066.
- Basten K**, Mallot HA. Simulated visual homing in desert ant natural environments: efficiency of skyline cues. *Biological Cybernetics*. 2010; 102(5):413–425.
- Burgess J.** RTX on - The NVIDIA TURING GPU. *IEEE Micro*. 2020 8; 40(2):36–44. doi: 10.1109/MM.2020.2971677.
- Christensen P**, Fong J, Shade J, Wooten W, Schubert B, Kensler A, Friedman S, Kilpatrick C, Ramshaw C, Banister M, Rayner B, Brouillat J, Liani M. RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering. *ACM Transactions on Graphics*. 2018 8; 37(3):1–21. <https://dl.acm.org/doi/abs/10.1145/3182162>, doi: 10.1145/3182162.
- Collett TS**, Land MF. Visual control of flight behaviour in the hoverfly *Syrirta pipiens* L. *Journal of Comparative Physiology A*. 1975 3; 99(1):1–66. <https://link.springer.com/article/10.1007/BF01464710>, doi: 10.1007/BF01464710.
- Collins S.** Reconstructing the Visual Field of Compound Eyes. In: *Eurographics Rendering Workshop* Eurographics, Springer Wien. ISBN; 1998. p. 81–92. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.2032>.
- Differt D**, Möller R. Spectral Skyline Separation: Extended Landmark Databases and Panoramic Imaging. *Sensors*. 2016 9; 16(10):1614. doi: 10.3390/s16101614.
- Floreano D**, Pericet-Camara R, Viollet S, Ruffier F, Brückner A, Leitel R, Buss W, Menouni M, Juston R, Dobrzynski MK, L'éplattenier G, Recktenwald F, Mallot HA, Franceschini N. Miniature curved artificial compound eyes. *Proceedings of the National Academy of Sciences of the United States of America*. 2013; 110(23):9267–9272. www.pnas.org/cgi/doi/10.1073/pnas.1219068110, doi: 10.1073/pnas.1219068110.
- Giger AD.** Honeybee vision: analysis of pattern orientation. PhD thesis, The Australian National University; 1996.
- Gkanias E**, Risse B, Mangan M, Webb B. From skylight input to behavioural output: A computational model of the insect polarised light compass. *PLOS Computational Biology*. 2019 7; 15(7):e1007123. <https://dx.plos.org/10.1371/journal.pcbi.1007123>, doi: 10.1371/journal.pcbi.1007123.

- 493 **Górski KM**, Hivon E, Banday AJ, Wandelt BD, Hansen FK, Reinecke M, Bartelmann M. HEALPix: A Framework
494 for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. The Astrophysical
495 Journal. 2005 4; 622(2):759–771. <https://iopscience.iop.org/article/10.1086/427976><https://iopscience.iop.org/article/10.1086/427976/meta>, doi: 10.1086/427976.
- 497 **Greiner B**, Ribi WA, Warrant EJ. Retinal and optical adaptations for nocturnal vision in the halictid bee *Mega-*
498 *lopta genalis*. Cell and Tissue Research. 2004 4; 316(3):377–390. <https://link.springer.com/article/10.1007/s00441-004-0883-9>, doi: 10.1007/S00441-004-0883-9.
- 500 **Harris CR**, Millman KJ, der Walt S, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith
501 NJ, Kern R, Hoyer MP, Stephan, van Kerkwijk MH, Brett M, Haldane A, delRío JF, Wiebe M, Peterson P, Gérard-
502 Marchant P, et al. Array programming with NumPy. Nature. 2020 9; 585(7825):357–362. <https://doi.org/10.1038/s41586-020-2649-2>, doi: 10.1038/s41586-020-2649-2.
- 504 **Immel DS**, Cohen MF, Greenberg DP. A radiosity method for non-diffuse environments. ACM SIG-
505 GRAPH Computer Graphics. 1986 8; 20(4):133–142. <https://dl.acm.org/doi/abs/10.1145/15886.15901>, doi:
506 10.1145/15886.15901.
- 507 **Juusola M**, Dau A, Song Z, Solanki N, Rien D, Jaciuch D, Dongre SA, Blanchard F, De Polavieja GG, Hardie RC,
508 Takalo J. Microsaccadic sampling of moving image information provides *Drosophila* hyperacute vision. eLife.
509 2017 9; 6. doi: 10.7554/eLife.26117.
- 510 **Kajiya JT**. The rendering equation. ACM SIGGRAPH Computer Graphics. 1986 8; 20(4):143–150. <https://dl.acm.org/doi/abs/10.1145/15886.15902>, doi: 10.1145/15886.15902.
- 512 **Kemppainen J**, Scales B, Haghighi KR, Takalo J, Mansour N, McManus J, Leko G, Saari P, Hurcomb J, Antohi A,
513 Suuronen JP, Blanchard F, Hardie RC, Song Z, Hampton M, Eckermann M, Westermeier F, Frohn J, Hoekstra
514 H, Lee CH, et al. Binocular Mirror-Symmetric Microsaccadic Sampling of Hyperacute 3D-Vision. bioRxiv. 2021
515 5; p. 2021.05.03.442473. <https://www.biorxiv.org/content/10.1101/2021.05.03.442473v1><https://www.biorxiv.org/content/10.1101/2021.05.03.442473v1.abstract>, doi: 10.1101/2021.05.03.442473.
- 517 **Labhart T**, Nilsson DE. The dorsal eye of the dragonfly *Sympetrum*: specializations for prey detection against
518 the blue sky. Journal of Comparative Physiology A. 1995 4; 176(4):437–453. <https://link.springer.com/article/10.1007/BF00196410>, doi: 10.1007/BF00196410.
- 520 **Lambrinos D**, Kobayashi H, Pfeifer R, Maris M, Labhart T, Wehner R. An Autonomous Agent Navigating with a
521 Polarized Light Compass. Adaptive Behavior. 1997 7; 6(1):131–161. https://journals.sagepub.com/doi/abs/10.1177/105971239700600104?casa_token=54Yzl6XPZWsaAAAA%3ArByXARHNZgTdbNjHWA2p-aqYHJhV_rs5v__Gq55LrLbmjm6IXs_dZOZmEYq_ucbY2nprHR3daIDJ, doi: 10.1177/105971239700600104.
- 524 **Land MF**, Fernald RD. The Evolution of Eyes. Annual Review of Neuroscience. 1992 3; 15(1):1–29. <http://www.annualreviews.org/doi/10.1146/annurev.ne.15.030192.000245>, doi: 10.1146/annurev.ne.15.030192.000245.
- 526 **Land MF**. Variations in the Structure and Design of Compound Eyes. In: *Facets of Vision* Berlin, Heidelberg:
527 Springer Berlin Heidelberg; 1989.p. 90–111. http://www.springerlink.com/index/10.1007/978-3-642-74082-4_5, doi: 10.1007/978-3-642-74082-4_5.
- 529 **Land MF**. VISUAL ACUITY IN INSECTS. Annual Review of Entomology. 1997 1; 42(1):147–177. <http://www.annualreviews.org/doi/10.1146/annurev.ento.42.1.147>, doi: 10.1146/annurev.ento.42.1.147.
- 531 **Land MF**, Nilsson DE. Animal eyes. Oxford University Press; 2002.
- 532 **Langer MS**. When Shadows Become Interreflections. International Journal of Computer Vision 1999 34:2. 1999;
533 34(2):193–204. <https://link.springer.com/article/10.1023/A:1008131719047>, doi: 10.1023/A:1008131719047.
- 534 **Lopes DAR**, Fernandes AR. Atmospheric scattering - state of the art. In: *EPCG 2014 21º Encontro Português de*
535 *Computação Gráfica* Instituto Politécnico de Leiria; 2014. <http://hdl.handle.net/1822/30959>.
- 536 **Mangan M**, Webb B. Modelling place memory in crickets. Biological cybernetics. 2009 10; 101(4):307. doi:
537 10.1007/s00422-009-0338-1.
- 538 **Meyer EP**, Labhart T. Morphological specializations of dorsal rim ommatidia in the compound eye of dragon-
539 flies and damselflies (Odonata). Cell & Tissue Research. 1993 4; 272(1):17–22. <https://link.springer.com/article/10.1007/BF00323566>, doi: 10.1007/BF00323566.
- 540

- 541 **Miller GS**, Hoffman CR. Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environ-
542 ments. In: *ACM SIGGRAPH 1984 Course Notes for Advances Computer Graphics Animation - SIGGRAPH '84 New*
543 *York, New York, USA; 1984.* <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.4747>.
- 544 **Millward B**, Maddock S, Mangan M. Towards Insect Inspired Visual Sensors for Robots. In: *UKRAS20:*
545 *Robots into the real world* UK-Robotics and Autonomous Systems (RAS) Network; 2020. p. 140–141. doi:
546 10.31256/do2ik3h.
- 547 **MÖLLER R**. Insects Could Exploit UV–Green Contrast for Landmark Navigation. *Journal of Theoretical Bi-*
548 *ology.* 2002 2; 214(4):619–631. <https://www.sciencedirect.com/science/article/pii/S0022519301924841>, doi:
549 10.1006/JTBI.2001.2484.
- 550 **Neumann TR**. Modeling Insect Compound Eyes: Space-Variant Spherical Vision. In: *Biologically Motivated Com-*
551 *puter Vision* Springer, Berlin, Heidelberg; 2002.p. 360–367. [http://link.springer.com/10.1007/3-540-36181-2_](http://link.springer.com/10.1007/3-540-36181-2_36)
552 [36](http://link.springer.com/10.1007/3-540-36181-2_36), doi: 10.1007/3-540-36181-2{_}36.
- 553 **Nityananda V**, Tarawneh G, Henriksen S, Umeton D, Simmons A, Read JCA. A Novel Form of Stereo Vision in
554 the Praying Mantis. *Current Biology.* 2018 2; 28(4):588–593. doi: 10.1016/J.CUB.2018.01.012.
- 555 **Ofstad TA**, Zuker CS, Reiser MB. Visual place learning in *Drosophila melanogaster*. *Nature.* 2011 6;
556 474(7350):204–207. <http://www.nature.com/articles/nature10131>, doi: 10.1038/nature10131.
- 557 **Parker SG**, Bigler J, Dietrich A, Friedrich H, Hoberock J, Luebke D, McAllister D, McGuire M, Morley K, Robison A,
558 Stich M. OptiX: a general purpose ray tracing engine. *ACM Transactions on Graphics (TOG).* 2010 7; 29(4):1–13.
559 <https://dl.acm.org/doi/abs/10.1145/1778765.1778803>, doi: 10.1145/1778765.1778803.
- 560 **Polster J**, Petrasch J, Menzel R, Landgraf T. Reconstructing the visual perception of honey bees in complex 3-D
561 worlds. *arXivorg.* 2018 11; [https://www.researchgate.net/publication/329057328_](https://www.researchgate.net/publication/329057328_Reconstructing_the_visual_perception_of_honey_bees_in_complex_3-D_worlds)
562 [Reconstructing_the_visual_](https://www.researchgate.net/publication/329057328_Reconstructing_the_visual_perception_of_honey_bees_in_complex_3-D_worlds)
[perception_of_honey_bees_in_complex_3-D_worlds](https://www.researchgate.net/publication/329057328_Reconstructing_the_visual_perception_of_honey_bees_in_complex_3-D_worlds).
- 563 **Primmerman CA**, Murphy DV, Page DA, Zollars BG, Barclay HT. Compensation of atmospheric optical distortion
564 using a synthetic beacon. *Nature* 1991 353:6340. 1991; 353(6340):141–143. [https://www.nature.com/articles/](https://www.nature.com/articles/353141a0)
565 [353141a0](https://www.nature.com/articles/353141a0), doi: 10.1038/353141a0.
- 566 **Purcell TJ**, Buck I, Mark WR, Hanrahan P. Ray tracing on programmable graphics hardware. In: *ACM SIGGRAPH*
567 *2005 Courses - SIGGRAPH '05* New York, New York, USA: ACM Press; 2005. p. 268. [http://portal.acm.org/citation.](http://portal.acm.org/citation.cfm?doid=1198555.1198798)
568 [cfm?doid=1198555.1198798](http://portal.acm.org/citation.cfm?doid=1198555.1198798), doi: 10.1145/1198555.1198798.
- 569 **Risse B**, Mangan M, Stürzl W, Webb B. Software to convert terrestrial LiDAR scans of natural environ-
570 ments into photorealistic meshes. *Environmental Modelling and Software.* 2018 1; 99:88–100. doi:
571 10.1016/j.envsoft.2017.09.018.
- 572 **Risse B**, Mangan M, Webb B, Del Pero L. Visual Tracking of Small Animals in Cluttered Natural Environments
573 Using a Freely Moving Camera. In: *Proceedings - 2017 IEEE International Conference on Computer Vision Work-*
574 *shops, ICCVW 2017*, vol. 2018-January Institute of Electrical and Electronics Engineers Inc.; 2017. p. 2840–2849.
575 doi: 10.1109/ICCVW.2017.335.
- 576 **Robinet F**, Arnaud R, Parisi T, Cozzi P. glTF: Designing an Open-Standard Runtime Asset Format. In: Engel
577 W, editor. *GPU Pro 5: Advanced Rendering Techniques*, vol. 1, 1 ed. New York, New York: CRC Press; 2014.p.
578 375–392. doi: <https://doi.org/10.1201/b16721>.
- 579 **Roth SD**. Ray casting for modeling solids. *Computer Graphics and Image Processing.* 1982 2; 18(2):109–144.
580 doi: 10.1016/0146-664X(82)90169-1.
- 581 **Schäfer H**, Süßmuth J, Denk C, Stamminger M. Memory efficient light baking. *Computers & Graphics.* 2012 5;
582 36(3):193–200. doi: 10.1016/J.CAG.2011.12.001.
- 583 **Schwarz S**, Narendra A, Zeil J. The properties of the visual system in the Australian desert ant *Melophorus*
584 *bagoti*. *Arthropod Structure & Development.* 2011 3; 40(2):128–134. doi: 10.1016/J.ASD.2010.10.003.
- 585 **Song YM**, Xie Y, Malyarchuk V, Xiao J, Jung I, Choi KJ, Liu Z, Park H, Lu C, Kim RH, Li R, Crozier KB, Huang Y,
586 Rogers JA. Digital cameras with designs inspired by the arthropod eye. *Nature.* 2013 5; 497(7447):95–99.
587 <http://www.nature.com/articles/nature12083>, doi: 10.1038/nature12083.

- 588 **Song Z**, Coca D, Billings S, Postma M, Hardie RC, Juusola M. Biophysical Modeling of a Drosophila Photoreceptor. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2009; 5863 LNCS(PART 1):57–71. https://link.springer.com/chapter/10.1007/978-3-642-10677-4_7, doi: 10.1007/978-3-642-10677-4{_}7.
- 592 **Sousa L**Md, Poggio L, Kempen B. Comparison of FOSS4G Supported Equal-Area Projections Using Discrete Distortion Indicatrices. ISPRS International Journal of Geo-Information 2019. 2019 8; 8(8):351. <https://www.mdpi.com/2220-9964/8/8/351/htmhttps://www.mdpi.com/2220-9964/8/8/351>, doi: 10.3390/IJGI8080351.
- 595 **Stamminger M**, Drettakis G. Perspective Shadow Maps. In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques - SIGGRAPH '02* New York, New York, USA: ACM Press; 2002. doi: 10.1145/566570.
- 598 **Stone T**, Mangan M, Ardin P, Webb B. Sky segmentation with ultraviolet images can be used for navigation. In: *Robotics: Science and Systems X*; 2014. p. 9. <http://eprints.lincoln.ac.uk/24748/1/Stone%20et%20al%20%282014%29.pdf>.
- 601 **Stürzl W**, Grixia I, Mair E, Narendra A, Zeil J. Three-dimensional models of natural environments and the mapping of navigational information. Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology. 2015 6; 201(6):563–584. doi: 10.1007/S00359-015-1002-Y.
- 604 **Taylor G**J, Baird E. INSECT VISION: SEGMENTATION TO SIMULATIONS. In: *3rd International Conference on Tomography of Materials and Structures* Lund University; 2017. p. 160. <https://meetingorganizer.copernicus.org/ICTMS2017/ICTMS2017-160.pdf>.
- 607 **Van Rossum G**, Drake FL. Python 3 Reference Manual. Scotts Valley, CA: CreateSpace; 2009.
- 608 **Vardy A**, Moller R. Biologically plausible visual homing methods based on optical flow techniques. Connection Science. 2005; 17(1-2):47–89.
- 610 **Viollet S**. Vibrating Makes for Better Seeing: From the Fly's Micro-Eye Movements to Hyperacute Visual Sensors. Frontiers in Bioengineering and Biotechnology. 2014 4; 2(APR):9. <http://journal.frontiersin.org/article/10.3389/fbioe.2014.00009/abstract>, doi: 10.3389/fbioe.2014.00009.
- 613 **Wehner R**. Desert Navigator. Harvard University Press; 2020. doi: 10.4159/9780674247918/HTML.
- 614 **Whitted T**, Turner, Whitted, Turner. An improved illumination model for shaded display. In: *Proceedings of the 6th annual conference on Computer graphics and interactive techniques - SIGGRAPH '79*, vol. 13 New York, New York, USA: ACM Press; 1979. p. 14. <http://portal.acm.org/citation.cfm?doid=800249.807419>, doi: 10.1145/800249.807419.
- 618 **Wiederman SD**, Fabian JM, Dunbier JR, O'Carroll DC. A predictive focus of gain modulation encodes target trajectories in insect vision. eLife. 2017 7; 6. doi: 10.7554/eLife.26478.
- 620 **Wystrach A**, Dewar A, Philippides A, Graham P. How do field of view and resolution affect the information content of panoramic scenes for visual navigation? A computational investigation. Journal of Comparative Physiology A. 2016; 202:87–95. <https://link.springer.com/content/pdf/10.1007%2Fs00359-015-1052-1.pdf>, doi: 10.1007/s00359-015-1052-1.
- 624 **Zeil J**, Hofmann MI, Chahl JS. Catchment areas of panoramic snapshots in outdoor scenes. Journal of the Optical Society of America A. 2003; 20(3):450–469. doi: 10.1364/josaa.20.000450.
- 626 **Zeil J**, Ribi WA, Narendra A. Polarisation Vision in Ants, Bees and Wasps. In: *Polarized Light and Polarization Vision in Animal Sciences* Berlin, Heidelberg: Springer Berlin Heidelberg; 2014.p. 41–60. http://link.springer.com/10.1007/978-3-642-54718-8_3, doi: 10.1007/978-3-642-54718-8{_}3.