Reconstructing the visual perception of honey bees in complex 3-D worlds

Johannes Polster[1], Julian Petrasch[1], Randolf Menzel[2], Tim Landgraf[1*]

1 Dahlem Center of Machine Learning and Robotics, Institute for Computer Science, Freie Universität Berlin, Berlin, Germany
2 Institute for Neurobiology, Freie Universität Berlin, Berlin, Germany

*** tim.landgraf@fu-berlin.de**

Abstract

Over the last decades, honeybees have been a fascinating model to study insect navigation. While there is some controversy about the complexity of underlying neural correlates, the research of honeybee navigation makes progress through both the analysis of flight behavior and the synthesis of agent models. Since visual cues are believed to play a crucial role for the behavioral output of a navigating bee we have developed a realistic 3-dimensional virtual world, in which simulated agents can be tested, or in which the visual input of experimentally traced animals can be reconstructed. In this paper we present implementation details on how we reconstructed a large 3-dimensional world from aerial imagery of one of our field sites, how the distribution of ommatidia and their view geometry was modeled, and how the system samples from the scene to obtain realistic bee views. This system is made available as an open-source project to the community on `http://github.com/bioroboticslab/bee_view`.

Introduction

Honey bees are extraordinary navigators. They orient themselves in an area of several square kilometers around their hives and they communicate spatial properties of remote resources via the waggle dance [24]. In the last decade, harmonic radar was used to trace the flights of navigating bees [17]. Recent results suggest that bees can robustly find their nest, even with an invalidated path integrator achieved by displacing the animal in a black box - or disturbed sun compass - induced by pausing the internal clock via anesthesia [4]. Honey bees have been shown to perform shortcut flights between known and dance-advertised sites over novel terrain [13], a behavior that indicates that geometrical relationships between location are represented in or computed by yet unknown neural structures. Experimental evidence for different strategies, such as path integration and visual guidance using picture memories, have been provided [6, 21]. However, it is still unknown how those components are combined and at which level of abstraction the different components are available to a navigating bee [5, 8].

While this question may ultimately be answered through electro-physiological studies, we think the flight behavior of navigating bees may provide clues about the nature of visual information that is used for a navigational tasks (such as finding back to the colony). Experimental studies that analyzed flight trajectories so far only looked at rather basic features, such as velocities or angles, which were then compared between treatments.

Figure 1. Close up photographs of a bee's head *(Megachile Fortis)* from the side (left) and from the front (right). The individual hexagonally shaped facets (the box on the right shows the magnified area of the blue rectangle) and the central ocellus on top of the head can be distinguished. The compound eyes have an ellipsoid shape and are roughly parallel to each other. The strong curvature of the eye results in a large field of view (FOV): the honeybee has close to a 360° FOV, only limited by the region blocked by the thorax of the bee. Photographs courtesy of the U.S. Geological Survey.

We have mapped a large area (2.73 km^2 in size) with a quadrocopter and have created a virtual representation of our test site's visual environment. We implemented the imaging geometry of the honey bee's complex eyes and are able to reconstruct the visual input available to a flying bee given her position in the world. Previously recorded flight trajectories of bees can now be replayed in the virtual world and hypotheses regarding to which information bees use for a given navigational task can be tested.

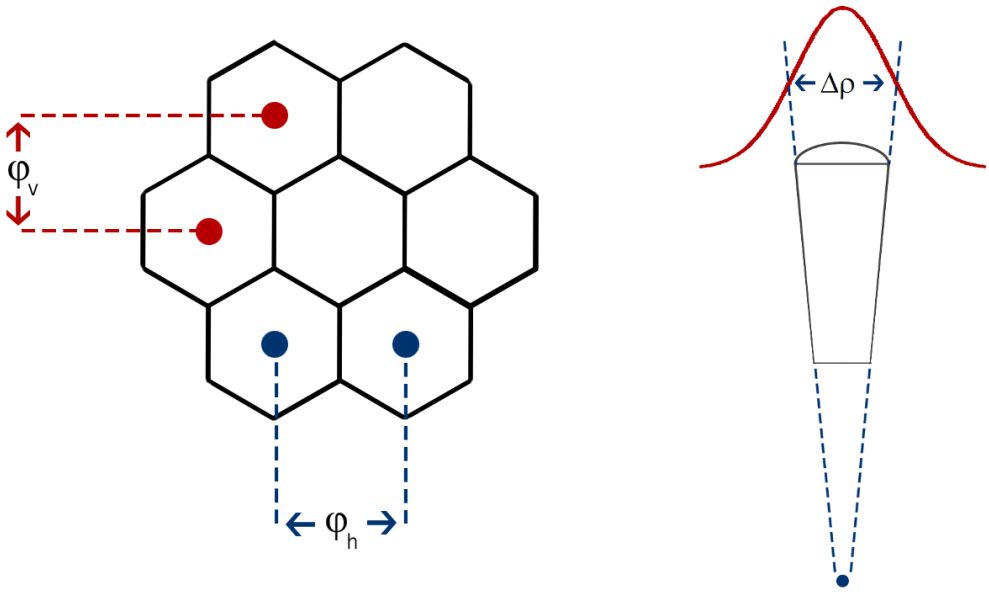
In this work we present our implementation of reconstructing the bee's view in the virtual world. We provide a detailed description of how our system performs with respect to runtime and imaging accuracy. We provide code and map data along with this paper. The software is available online on GitHub.

Previous Work

The compound eyes of insects are made up of thousands (in the case of the honeybee about 5 500) of hexagonally shaped ommatidia facing in different directions. [19] Each ommatidium acts like a single eye with its own corneal lens and (in reference to the apposition eye) photoreceptor. But unlike the human eye, each ommatidium receives light from a very limited portion of the environment. An ommatidium thus can be thought of as one picture element, or pixel [3].

In order to mimic the visual input we need to model the spatial distribution and the field of view of each ommatidium in the compound eye. The spatial resolution of the resulting image is determined by the interommatidial angles and the acceptance angles (see Figure 8) of the ommatidia. Both have been mapped out by Seidl and coworkers [19], which forms the basis of our eye model proposed here. A similar model for generating the interommatidial angles was described in [22], for the sake of brevity

Figure 2. Left: Interommatidial angles: The interommatidial angle ($\Delta\varphi$) is the angle between neighbouring ommatidia. One can differentiate horizontal (elevation, $\Delta\varphi_h$) and vertical (azimuth, $\Delta\varphi_v$) angles. Right: The acceptance angle ($\Delta\rho$) defines the visual field of the individual ommatidia. The acceptance function describes the sensitivity of the ommatidium, in relation to the angular distance from the optical axis. The angular sensitivity function can be approximated by a two-dimensional circular Gaussian. The full width at half maximum (FWHM) of this Gaussian is the acceptance angle of the ommatidium [23].



called the "Stürzl-model" from here on. It was an extension of an earlier model proposed by Giger [9] which only covered the frontal hemisphere of the eye (180° horizontal field of view). The Stürzl-model covered the full field of view of the compound eye, taking into account the boundaries of the eye's visual field. Based on his model, Giger developed an application called the Bee Eye Optics Simulation (BEOS). It simulates how bees perceive 2D images placed at a specific distance from the bee [10]. Collins [7] developed a Monte Carlo raytracing simulation for honeybee vision. He takes the geometry of the eye as a basis for calculating viewing directions, and also simulates the spectral sensitivity of the eye.

There are also hardware implementations of how a bee perceives the world. In [25], the spectral sensitivity of the receptors of an eye is simulated by recording images with a UV-sensitive camera. The spatial resolution is simulated by reconstructing the bee views depending on the interommatidial angles and acceptance angle of the ommatidia. In [15], Neumann describes a method for mapping a cubic environment map to spherical images with the resolution of insect vision.

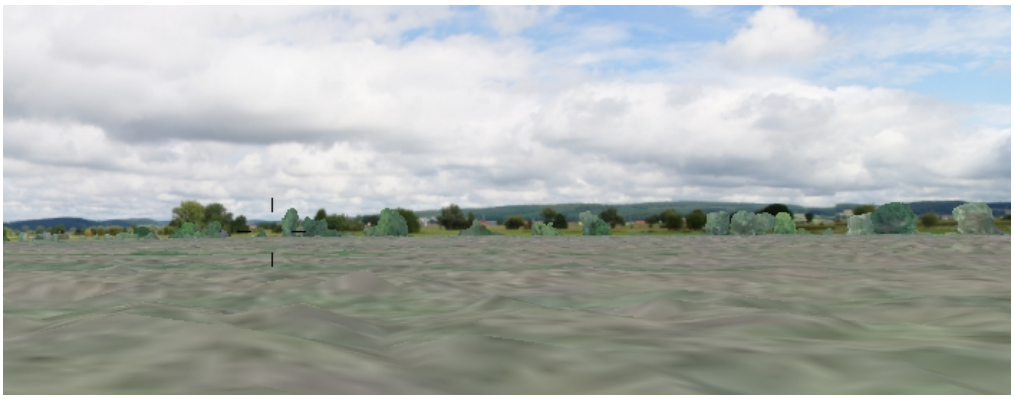
While substantial previous work has been done to reproduce the visual perception of bees, unfortunately, none of the solutions was publicly available to an interested researcher. Our motivation thus was to implement an accurate model that potentially runs in real-time, with an explicit raytracing instead of remapping. This model should be generically applicable to different models of compound eyes and different world models, and it should be freely available to the community.

Implementation

3-D World

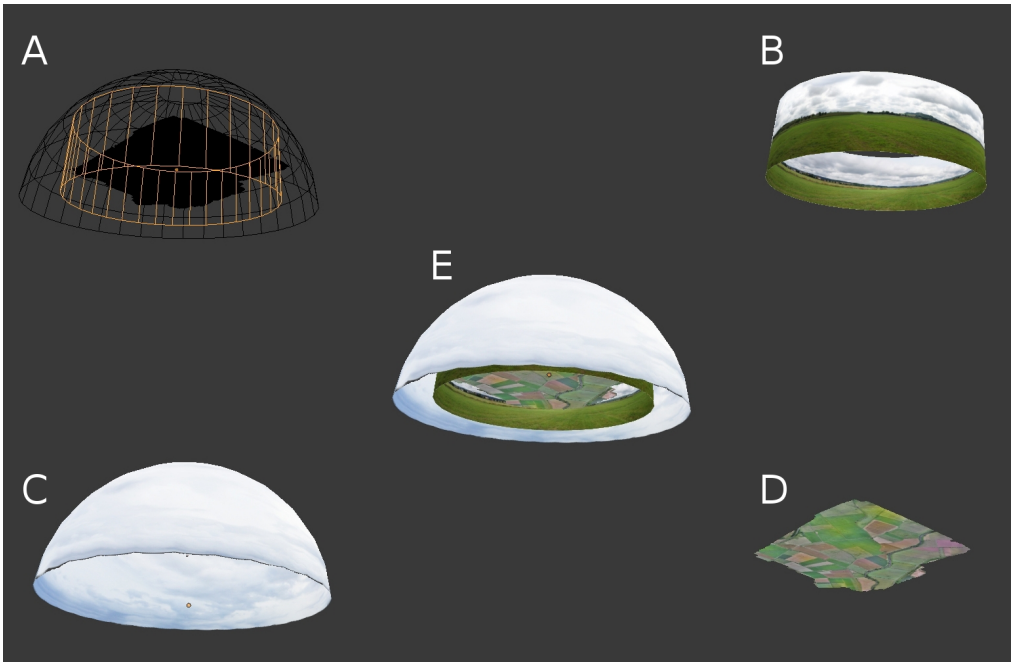
The 3-D world consists of three parts: 1) a 3-dimensional depth map of an experimental field site surrounded by 2) a cylinder that holds a panorama image and 3) a sky dome (see figure 4). The virtual world reproduced an area east of Großseelheim, a town in central Germany. It covers an area of about two square kilometers and was used for behavior experiments with bees over the last few years. The depth map was created in June 2016 from aerial images taken by a drone, using stereophotogrammetry. The resulting model has a vertical accuracy of 30 cm and a horizontal accuracy of 5 cm–10 cm. Therefore small bushes and trees appear with their respective shapes in the depth map but smaller objects such as fences and small plants are only visible in the texture of the model. The environment was highly structured and exhibits panoramic features that were too far away to be depth mapped by our drone. Hence, the model was extended by mapping a high resolution panoramic image onto a cylinder.

Figure 3. Here the problem of duplicated objects is illustrated. Objects in the 3D model also appear on the panorama. The bushes in the background are duplicated.



Objects within the drone-captured area appear in this panorama texture irrespective of the camera position and a duplicate would be imaged to the bee eye, one from the actual 3-D object and one from the panorama texture. To solve this problem, duplicated objects were identified in the 3-D world and removed manually from the panorama with an image editing program. Larger objects (such as trees) were replaced with parts of other panoramas, since one can not see what is behind these objects (see Figure 3). Note that only one recording was used as panorama texture. It's projection to the bee eye is correct only for positions close to the position we recorded the panorama. For all other positions in the world the projection exhibits an error proportional both to the distance of the original object's position to the camera and the distance of the camera to the original panorama recording position. When moving closer to objects, the objects' projection grows larger, when moving away they appear smaller, when moving parallel to the objects they shift. In a 360° panorama such as our virtual world, all of these effects can be observed in any one move. However, since the area mapped by our drone is fairly large, the errors introduced by having only a static panorama are negligible. The resulting model has 1 000 294 faces and 499 116 vertices. The resolution of the texture of the 3D terrain is 8000 px × 8000 px, the resolution of the texture of the cylinder is 24 000 px × 3000 px. The model needs 472 MiB of disk space.

Figure 4. Illustration of how the 3D model is extended. A: Wireframe of the three Components. B: The cylinder with the panorama projected onto it. C: The hemisphere with the sky texture. Since the sky in this model is static it may introduce bias to the subsequent image analysis. We therefore also created a 3D model with solid white sky. D: The 3D Terrain captured by the drone. E: The resulting 3D model with skydome and panorama.



Raycasting

In order to generate a realistic projection of the world's object to our model of a bee eye, we cast rays from each ommatidium into the world. While "ray tracing" methods follow rays of light over multiple bounces off of scene objects, "ray casting" only takes into account the primary ray, i.e. only the light rays between camera and object are simulated. To achieve this, rays are generated from the camera. For each pixel of the image to be rendered, ray directions are calculated from the eye model. The rays are "shot" in the calculated directions. Then, every object in the scene is tested whether it intersects with the ray. This is computationally expensive since there can be millions of objects in a scene. After an intersection is found, the colour for the pixel is sampled from the object's texture at the intersection point.

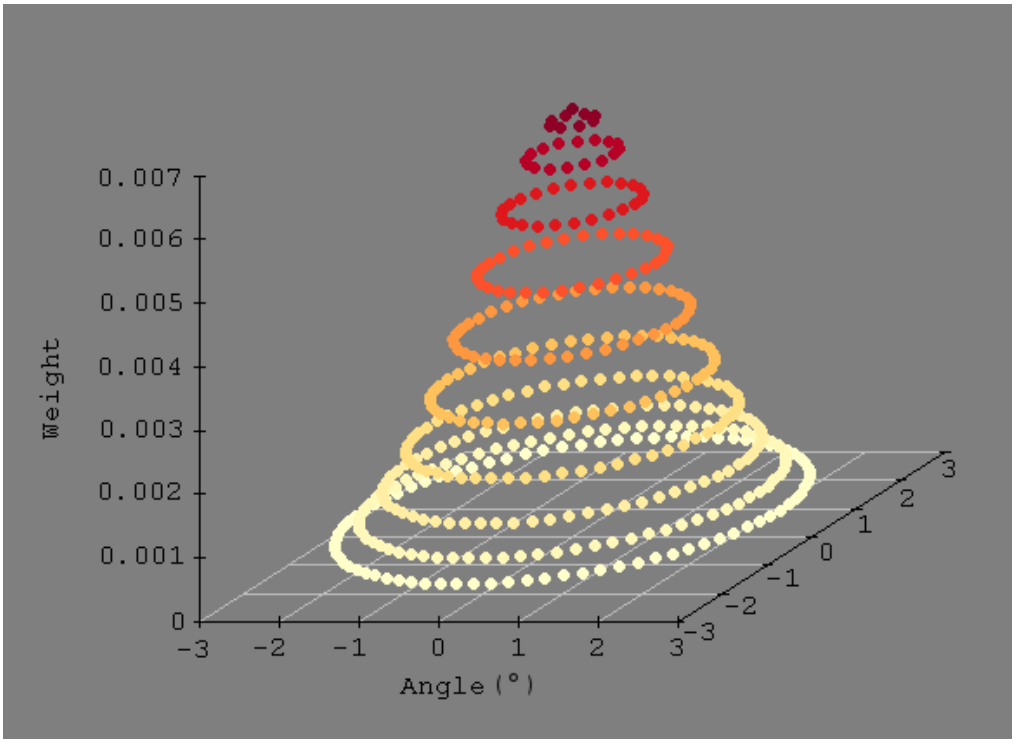
Model of the Bee's Compound Eye

In the honeybee eye, the interommatidial angles vary across the bee's compound eye, with a minimum at the equator (elevation = 0) and gradual increments towards the borders of the eye. This means, ommatidia have a smaller spacing, i.e. a higher resolution at the equator. Vertically, inter-ommatidial angles range from 1.5° to 4.5° and horizontally they range from 2.4° to 4.6°. For calculating the interommatidial angles, a routine described by Stürzl et al [22] was implemented. The routine is based on a formula from Giger [9]. Giger approximates the measurements of interommatidial angles determined by Seidl [19] for all ommatidia in the frontal hemisphere. Stürzl et al. [22] extend this model to cover the full bee's eye FOV. They also take into account the border of the visual field. Since the authors of [22] did not provide source code, we re-implemented the routine as an R script. This model produces angles for a total of 5440 ommatidia per eye. These are precomputed and stored in a comma separated file for later use by the renderer. This way, the subsequent parts of the rendering pipeline can be used for updated models of ommatidium distribution or different animal models. These angles in 3-D space define the direction of the rays to be cast. Since individual ommatidia do not just register light coming in from this exact direction, but rather collect light from a field around this average vector, we need to define how much of the scene can be sampled by one ommatidium and with how much weight samples from differing directions are integrated into the output of an ommatidium. Stürzl et al chose to use an acceptance angle that varies depending on the elevation and azimuth of the ommatidia. Since the interommatidial angles also vary, a static acceptance angle may lead to oversampling in areas of high resolution (e.g. the center of the eye) and undersampling (at the edge of the eye). The lens diameter also varies between 17 µm and 24 µm over the surface area of the eye, this has been interpreted as an indication for a dynamic acceptance angle by Stürzl et al, however as of now, there aren't any direct electro-physiological measurements available for the whole eye. The only direct measurements were conducted in the frontal region of the eye and came up with an acceptance angle of 2.6° [12]. Stürzl et al's acceptance angle is not radially symmetric since it depends on horizontal and vertical interommatidial angles. Giger uses a static acceptance angle of 2.6° with a radially symmetric acceptance function – an approach followed in our rendering engine.

Sampling from the Scene

The acceptance function is a radially symmetric Gaussian with a full width at half maximum (FWHM) that is equal to the acceptance angle. Stürzl et al use 9×9 sampling directions per ommatidium and weight the samples with a Gaussian weight matrix, whereas Giger uses a sampling array of 441 sampling points that are arranged

Figure 5. Gaussian sampling function with 462 samples showing the angular deviation from the main optical axis, the weights are shown as a third dimension.



as concentric circles around the optical axis of the ommatidium. Each sample is weighted according to its distance from the optical axis using a gaussian pdf with zero mean and a standard deviation of 1,1523 (in case of an acceptance angle of 2.6°).

Similarly to Giger, we implemented a concentric disk sampling method. This is achieved by creating a square sample matrix with coordinates ranging from -1 to 1 and then mapping the sample points to a disk. Afterwards, the coordinates are normalized to be in the range of $-\Delta\rho$ to $+\Delta\rho$. The formula maps the x, y coordinates of a point in a square to the X, Y coordinates of a point in a disk [18]:

$$(x,y) \mapsto (X,Y) = \begin{cases} \left(\frac{2y}{\sqrt{\pi}} \sin\frac{x\pi}{4y}, \frac{2y}{\sqrt{\pi}} \cos\frac{x\pi}{yx}\right), & \text{if } |x| \leq |y| \\ \left(\frac{2x}{\sqrt{\pi}} \cos\frac{y\pi}{4x}, \frac{2x}{\sqrt{\pi}} \sin\frac{y\pi}{4x}\right), & \text{otherwise} \end{cases}$$

The acceptance function that weighs the sample points depending on the distance to the main optical axis of the ommatidium is given by:

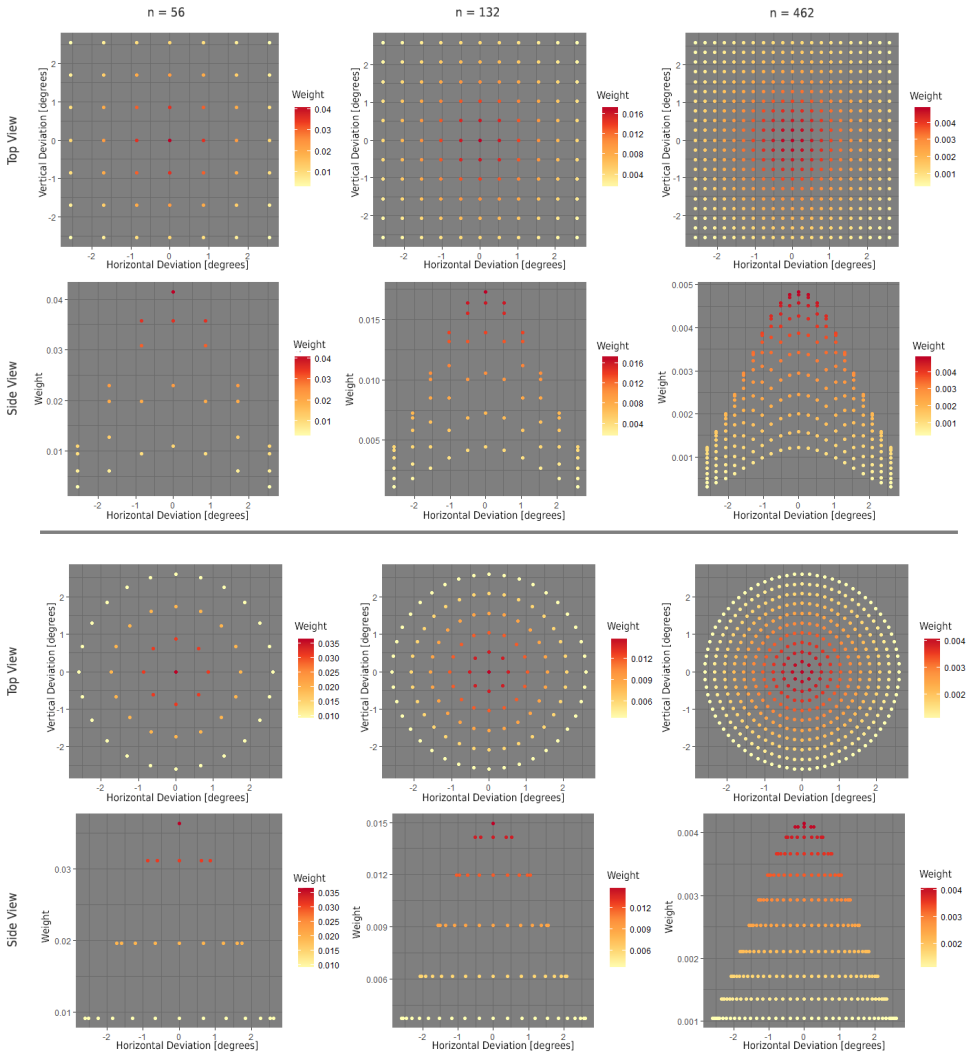
$$f(x,y) = e^{-\left(\frac{5\sqrt{x^2+y^2}}{3\Delta p}\right)^2}$$

The formula approximates a bivariate Gaussian function with FWHM $\Delta\rho$. For a $\Delta\rho$ of 2.6° this equates to:

$$f(x,y) = e^{-0.410914\left(x^2+y^2\right)}$$

The weights produced from the formula are then normalized to sum up to 1.

Figure 6. This figure compares the shape of different sampling functions viewed from the side and the top. The top view shows the angular deviation from the main optical axis of the ommatidia, the side view shows the horizontal deviation and the corresponding weights. The weights determine how much the sampled color contributes to the perceived color of the ommatidium. The Stürzl-model uses a square sampling function with 81 sampling points, and Giger uses a concentric disk sampling function with 441 sampling points. How the number of samples affect the output image is thoroughly compared in the results chapter.

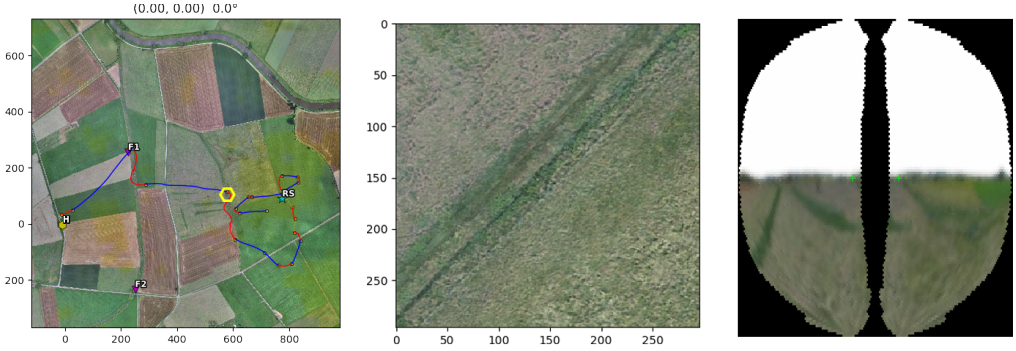


Core Technologies

The core renderer was written in C++. It uses Embree for intersecting the rays with the scene and the Eigen C++ Vector Library [11] for fast vector arithmetic. Embree is a raytracing kernel developed by Intel and offers core raytracing functionality such as intersecting rays with the scene, while hiding the underlying acceleration structures and CPU optimizations. Additionally it has a good documentation and, even though still under development, the API is stable. Furthermore it is highly optimized for CPUs, achieving good results in benchmarks. Embree is free and open source, as it is released under the Apache 2.0 license. It runs on all modern x86 and AMD64 CPUs [26].

The rendering engine uses the Wavefront Object (.obj) file format, since it is supported by most of the major 3D applications and it is an open, human readable format. The core renderer was wrapped in Python, as Python is widely used in scientific programming, so this provides an interface that can easily be used with other scientific applications. For every C++ function that should be wrapped, a corresponding Cython function was written that calls the C++ function. The result is the beeview python package that, after being built with the Cython compiler, can easily be imported to Python.

Figure 7. Left image: an example plot of a flight from the release site (RS) to the hive (H). Red is search flight, blue is linear flight and the dotted yellow lines are gaps. The yellow hexagon is the position of the bee. The central image shows the 2D bee view from that position (a 25 m × 25 m portion of the map), rendered with the Radar Track module. The right image shows the bee view rendered with the help of the beeview python package.



The source code can be found on Github [16].

Results

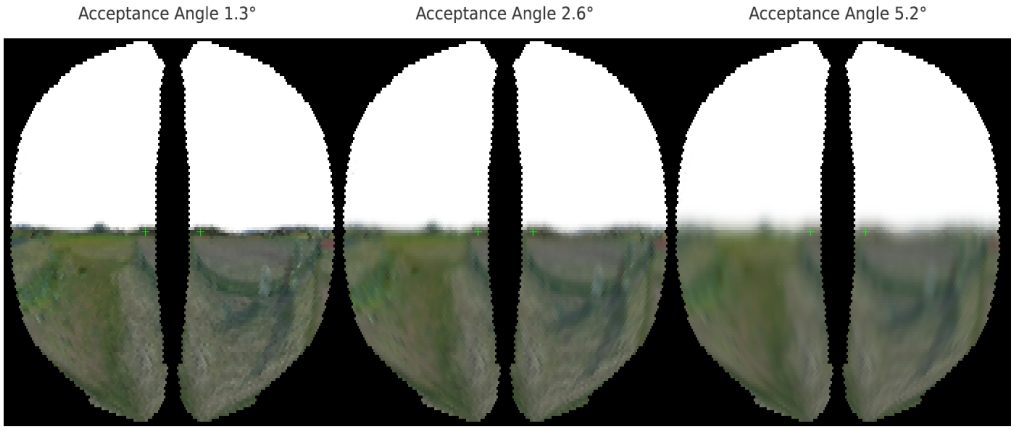
In this section we look at how different rendering parameters affect the output and the performance of the renderer. The runtime performance of raycasting directly depends on the number of rays to cast and the amount of polygons in the scene. These are given by the bee's eye model and the scene described in the section "3D World". The number of rays needed for rendering a bee view is $2N_oN_s$. Where N_o is the number of ommatidia and N_s is the number of samples per ommatidium. For 462 samples per ommatidium the renderer generates 5 026 560 rays and performs as many intersection tests. The scene has over 10^6 faces that need to be tested for intersection.

Since the performance of the renderer directly depends on the number of rays to cast, users might decide decreasing the number of rays for faster rendering. We

conducted a test series to determine the minimal sample size per ommatidium at which the renderer still yields acceptable results (see Figure 9). From visual inspection we conclude that more than 56 rays per ommatidium may not be necessary. Lower numbers decrease rendering times but produce choppier images.

The acceptance angle controls the sharpness of the rendered beeview (see Figure 8). A larger acceptance angle leads to a blurrier Image. Also, objects that are farther away are not as sharp as closer objects, since with greater distance the acceptance angle covers a larger area.

Figure 8. The effect of different acceptance angles on the output of the renderer. left: 1.3°, centre: 2.6°, right: 5.2°. A smaller acceptance angle leads to a sharper image.



We also compared if and how the rendering output is affected by using a square sampling distribution (similar to the Stürzl-model [22] or a concentric disk distribution similar to Giger [9] (see Figure 10. We find that there is no perceivable difference between the two methods, except for small sample sizes, as square sampling covers a larger area ($2\Delta\rho$ = width of the square = diameter of the disk). However for larger sample sizes the differences are less pronounced, since the samples at the edge contribute with small weights.

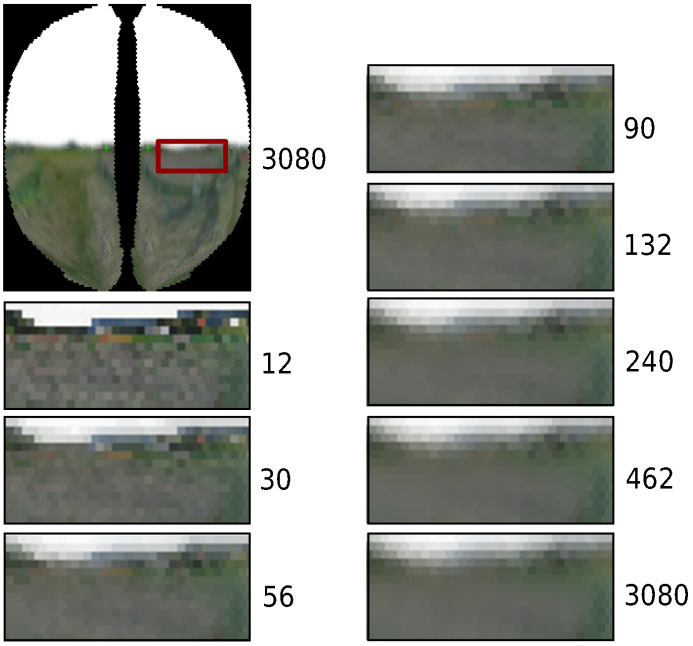
We conducted a series of tests to examine the properties of the rendered bee views (see Figure 11). The test series shows that objects in the centre of the eye appear enlarged, since the resolution of the eye is highest here. Also, the closer the object, the more it appears distorted, because the object covers a larger part of the field of view. Additionally the model confirms that only a small portion of the field of view of the eyes overlap.

Discussion

We have implemented a fast, accurate and open software package to reproduce the visual perception of honeybees. It is the first system of this kind made available as open-source package.

The renderer is not limited to rendering bee views, but can also render normal images from a pinhole or a panoramic camera. It has a simple API so it can easily be interfaced with from other applications. The beeview Python package provides bindings to the C++ functions of the renderer. The renderer is implemented in a way that all the settings of the renderer are flexible. This means that the renderer can easily be used with different interommatidial and acceptance angles for simulating the vision of other

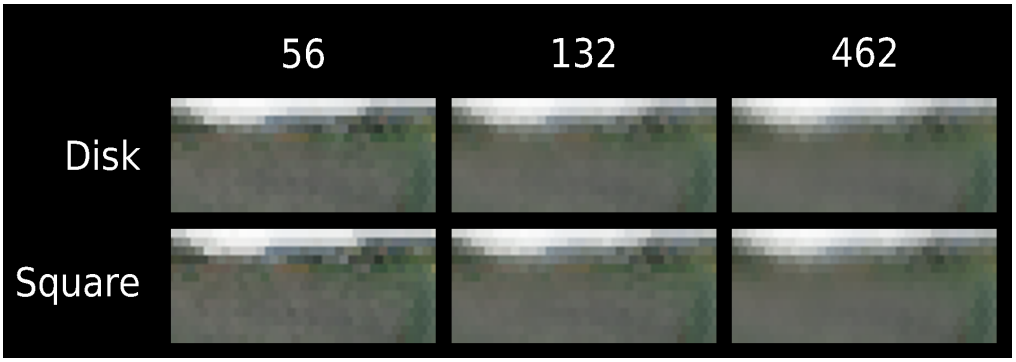
Figure 9. These renderings show a portion of the bee view (the red rectangle) for different sample sizes to determine how the number of sample points affects the simulated bee vision. For this a series of bee views was rendered, with the number of sample points per ommatidium ranging from 12 to 9900. Until n = 56 the differences between the bee views are quite large. From n = 56 to n = 462 the image still becomes smoother, but the differences are almost not noticeable. From n = 462 there is no conceivable difference between the rendered bee views.



insects. The optimal settings for rendering bee views were determined so the performance is maximized while still maintaining accuracy. With the proposed system, behavioral studies that investigate the relation of visual input to behavioral output can benefit from this system. The 3D model is also an ideal environment for synthetic studies using artificial agents. The C++ API and the beeview Python package provide all the functions needed for the movements of an agent like the one implemented by Müller [14]. Functions for moving, rotating and rolling the camera, so all the possible movements of a bee are covered. Furthermore it is possible to set the camera direction directly, or via a look_at(point) function. A render function that returns the elevation angles, the azimuth angles and the sampled colours of all ommatidia as continuous arrays for visual input of the agent. A function for measuring the distance from a point to the next object in a specific direction, that can be used for measuring the height above ground and setting the camera's position accordingly. The renderer can also be used for evaluating pattern and shape recognition experiments, as in [9], [20], [1], by placing test images at a specific distance from the camera. Additionally the renderer can be used for educational purposes to demonstrate how different parameters of the compound eye affect insect vision.

Still, a number of aspects can be improved. Loading the high resolution textures of the model has the biggest impact on the start-up time of the renderer (about 10 s). Using a faster image library instead of the simple ppm loader could speed up the process. Only supporting the ppm file format for images is not optimal, but can easily be improved by using a different image library. The model eye only takes into account the spatial resolution of honeybee's eyes. The model could be extended to include the

Figure 10. Comparison of differences between using a square sampling distribution.

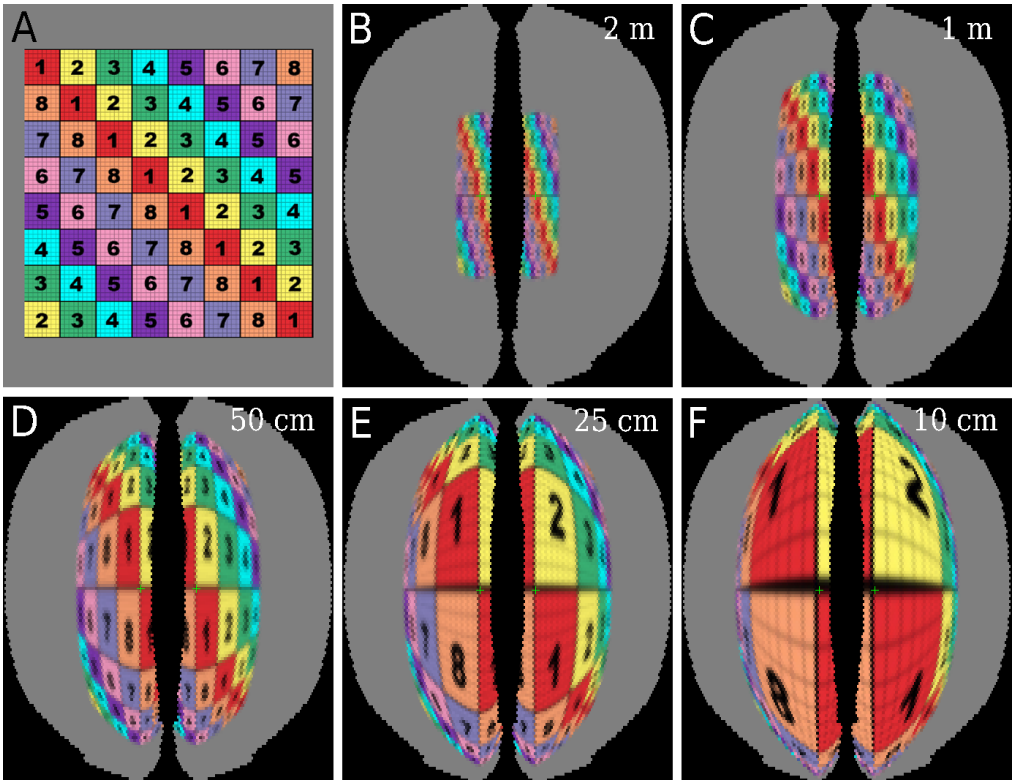


light intensity received by the ommatidia and the spectral sensitivity of the ommatidia. To achieve this, the 3D environment has to include UV emission information. This could be done by recording the scene with a camera sensitive to UV-light and storing the recorded UV-data in the red channel or alpha channel of the texture. Additionally only the compound eyes were modelled, but for a complete bee vision simulation the ocelli should also be taken into account. The 3D model could be extended by including light sources and material properties. Based on these the renderer could render shadows and other light effects by tracing the rays for multiple bounces. The polarization of light could be simulated, as some ommatidia of the bee's eye are sensitive to it. The scene could be refined by using a subdivision mesh (Embree is capable of handling subdivision meshes) [2]. In the summer of 2017, we recorded a larger 3-D model. Embedding it in a DEM would be a good method for expanding the model, since the new model covers all scene objects that are near the testing area, and the elevation data is sufficient for modelling the far away hills. The method followed in this paper (with one panorama taken from the centre of the model) would pose additional manual work to produce a sufficiently accurate panorama. Due to the larger size of the model, and therewith more extreme positions with respect to the panoramic recordings, the angular deviations of objects in the panorama would likely exceed acceptable magnitudes. We plan to provide more accurate maps of the testing grounds in the near future.

Acknowledgments

We thank Thierry Meurers for assisting in the drone-based mapping of the field. This work has, in part, been funded by the Dr.-Klaus-Tschira-Foundation, through Grant No. 00.300.2016 (Robotik in der Biologie: Ziele finden mit einem winzigen Gehirn. Die neuronalen Grundlagen der Navigation der Bienen).

Figure 11. These renderings show the simulation of how a bee sees a 2 m × 2 m image from different distances. Image A: The test scene, rendered with the pinhole camera. The test image shows squares of different colours with numbers in them. Each square has a width and height of 25 cm. Because of the square form, distortions are easily identified. The numbers and colours help distinguish the different squares. As mentioned before, the renderer only reproduces the sampling geometry and not the spectral perception of honey bees. The colors, hence, serve purely for distinguishing the squares. Image B: The test image seen from a distance of 2 m. Image C: 1 m. D: 50 cm. E: 25 cm. F: 10 cm. The settings for the bee camera used: acceptance angle 2.6°, disk sampling, 132 samples per ommatidium.



References

1. A. M. Anderson. Shape perception in the honey bee. *Animal Behaviour*, 25:67–79, Feb. 1977.

2. C. Benthin, S. Woop, M. Nießner, K. Selgrad, and I. Wald. Efficient Ray Tracing of Subdivision Surfaces using Tessellation Caching. In *Proceedings of the 7th High-Performance Graphics Conference*. ACM, 2015.

3. A. Borst. Drosophila's View on Insect Vision. *Current Biology*, 19(1):R36–R47, Jan. 2009.

4. J. F. Cheeseman, C. D. Millar, U. Greggers, K. Lehmann, M. D. Pawley, C. R. Gallistel, G. R. Warman, and R. Menzel. Way-finding in displaced clock-shifted bees proves bees use a cognitive map. *Proceedings of the National Academy of Sciences*, page 201408039, 2014.

5. A. Cheung, M. Collett, T. S. Collett, A. Dewar, F. Dyer, P. Graham, M. Mangan, A. Narendra, A. Philippides, W. Stürzl, et al. Still no convincing evidence for cognitive map use by honeybees. *Proceedings of the National Academy of Sciences*, 111(42):E4396–E4397, 2014.

6. T. S. Collett and M. Collett. Memory use in insect visual navigation. *Nature Reviews Neuroscience*, 3(7):542, 2002.

7. S. Collins. Reconstructing the Visual Field of Compound Eyes. In *Rendering Techniques '97*, Eurographics, pages 81–92. Springer, Vienna, 1997.

8. H. Cruse and R. Wehner. No need for a cognitive map: decentralized memory for insect navigation. *PLoS computational biology*, 7(3):e1002009, 2011.

9. A. Giger. *Honeybee vision: analysis of pattern orientation*. Doctoral Thesis, Australian National University, Canberra, Australia, 1996.

10. A. Giger. BEOS. http://andygiger.com/science/beye/beyehome.html, 2009.

11. G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

12. S. B. Laughlin and G. A. Horridge. Angular sensitivity of the retinula cells of dark-adapted worker bee. *Zeitschrift für Vergleichende Physiologie*, 74(3):329–335, 1971.

13. R. Menzel, A. Kirbach, W.-D. Haass, B. Fischer, J. Fuchs, M. Koblofsky, K. Lehmann, L. Reiter, H. Meyer, H. Nguyen, et al. A common frame of reference for learned and communicated vectors in honeybee navigation. *Current Biology*, 21(8):645–650, 2011.

14. J. Müller. Familiarity in Honeybee Navigation - Behavioral and Neurocomputational Investigations. Master's thesis, Ludwig-Maximilians-Universität München and Freie Universität Berlin, Munich, 2016.

15. T. R. Neumann. Modeling Insect Compound Eyes: Space-Variant Spherical Vision. In *Biologically Motivated Computer Vision*, Lecture Notes in Computer Science, pages 360–367. Springer, Berlin, Heidelberg, Nov. 2002.

16. J. Polster. Bee Vision Simulation. `https://github.com/BioroboticsLab/bee_view`, 2018.

17. J. Riley, A. Smith, D. Reynolds, A. Edwards, J. Osborne, I. Williams, N. Carreck, and G. Poppy. Tracking bees with harmonic radar. *Nature*, 379(6560):29, 1996.

18. D. Roşca. New uniform grids on the sphere. *Astronomy and Astrophysics*, 520:A63, Sept. 2010.

19. R. Seidl and W. Kaiser. Visual field size, binocular domain and the ommatidial array of the compound eyes in worker honey bees. *Journal of comparative physiology*, 143(1):17–26, Mar. 1981.

20. M. V. Srinivasan. Pattern recognition in the honeybee: Recent progress. *Journal of Insect Physiology*, 40(3):183–194, Mar. 1994.

21. M. V. Srinivasan. Going with the flow: a brief history of the study of the honeybee's navigational 'odometer'. *Journal of Comparative Physiology A*, 200(6):563–573, 2014.

22. W. Stürzl, N. Boeddeker, L. Dittmar, and M. Egelhaaf. Mimicking honeybee eyes with a 280 degrees field of view catadioptric imaging system. *Bioinspiration & Biomimetics*, 5(3):036002, Sept. 2010.

23. F. G. Varela and W. Wiitanen. The Optics of the Compound Eye of the Honeybee (Apis mellifera). *The Journal of General Physiology*, 55(3):336–358, Mar. 1970.

24. K. Von Frisch. The dance language and orientation of bees. 1967.

25. M. Vorobyev, A. Gumbert, J. Kunze, M. Giurfa, and R. Menzel. Flowers Through Insect Eyes. *Israel Journal of Plant Sciences*, 45(2-3):93–101, Jan. 1997.

26. A. T. Áfra, I. Wald, C. Benthin, and S. Woop. Embree ray tracing kernels: overview and new features. pages 1–2. ACM Press, 2016.