A functional characterization of a Go-opsin and a ratio-chromatic depth gauge in *Platynereis dumerilii*

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
M.Sc. Bioinf. Martin Gühmann
aus Berlin

Tübingen
2017

Tag der mündlichen Qualifikation:	17.07.2017
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Dr. Gáspár Jékely
2. Berichterstatter:	Prof. Dr. Heinz-R. Köhler

Abstract

Light guides marine invertebrate larvae to their settlement places. A light guided behavior is phototaxis, which is mediated by opsins. Among the opsins, the Go-opsins are ancient, but poorly characterized, because they only survived in marine invertebrates. A Go-opsin is expressed with two rhabdomeric opsins in the adult eyes of the larva of *Platynereis dumerilii*. In the larva, the adult eyes mediate phototaxis.

Here, I functionally characterized this Go-opsin1, by generating a *Go-opsin1* knockout line with zinc-finger-nucleases. I designed several assays to study light guided behaviors of the larvae and to compare phototaxis of wild type and *Go-opsin1* knockout larvae. The *Go-opsin1* knockout larvae were phototactic but less phototactic to blue-cyan-green light, which is the spectral range that closely matches the *in vitro* spectrum of Go-opsin1.

Additionally, I found a new light guided behavior, which is as fast as phototaxis. When I stimulated the larvae with UV-light, the larvae swam down irrespective whether the light came from the top, the bottom or diffusely from all sides. This UV-response is a positive geotaxis induced by non-directional UV-light. The UV-response worked against phototaxis; the larvae swam down to certain ratios of UV and visible light. The ratios did not change when the light was dimmed. Therefore, the UV-response forms with phototaxis a ratio-chromatic depth gauge. The UV-response spectrum matched the absorption spectrum of c-opsin1. C-opsin1 is expressed in the ciliary photoreceptor cells, which have stacked membranes and so may be very sensitive.

Therefore, the ciliary photoreceptor cells with c-opsin1 may mediate the UV-response, while phototaxis is mediated by Go-opsin1 and the rhabdomeric opsins. Go-opsin1 seems to couple to a Gq-protein in the rhabdomeric photoreceptor cells of the adult eyes. Therein, Go-opsin1 differs from a scallop Go-opsin, which seems to couple to a Go-protein in ciliary photoreceptor cells.

Ciliary photoreceptor cells as in *Platynereis dumerilii* are common among marine invertebrate larvae so that the depth gauge may be common among those larvae, too. The depth gauge may even trace back to the last common ancestor of all bilaterians. The depth gauge helps the larvae to find the right depth for settling on a global level, while positive and negative phototaxis helps the larvae to select locally a settlement site.

Zusammenfassung

Licht lenkt die Larven mariner Wirbellosen zu ihren Siedlungsorten. Ein lichtgelenktes Verhalten ist Phototaxis, welches durch Opsine vermittelt wird. Unter den Opsinen sind die Go-Opsine sehr alt, aber schlecht charakterisiert, weil sie nur noch in marinen Wirbellosen existieren. Ein Go-Opsin ist mit zwei rhabdomerischen Opsinen in den definitiven Augen der Larve von *Platynereis dumerilii* exprimiert. In der Larve vermitteln die definitiven Augen Phototaxis.

Hier charakterisierte ich dieses Go-Opsin1 funktionell, indem ich eine *Go-Opsin1*-Knockout-Linie mit Zinkfingernukleasen erzeugte. Ich entwickelte verschiedene Versuche, um lichtgelenktes Verhalten der Larven zu untersuchen, und um Phototaxis von Wildtyp- und Go-Opsin-Knockout-Larven zu vergleichen. Die Go-Opsin-Knockout-Larven waren phototaktisch aber sie reagierten weniger phototaktisch auf blau-zyan-grünes Licht, welches den Spektralbereich abdeckt, der dem *in vitro* Absorptionsspektrum von Go-Opsin1 entspricht.

Außerdem fand ich ein neues lichtgelenktes Verhalten, das genauso schnell einsetzt wie Phototaxis. Als ich die Larven mit UV-Licht stimulierte, schwammen sie nach unten, egal ob das Licht von oben, unten oder diffus von allen Seiten kam. Diese UV-Antwort ist eine positive Geotaxis, die von UV-Licht aktiviert wird. Die UV-Antwort arbeitet gegen Phototaxis: Die Larven schwammen nach unten, wenn UV und sichtbares Licht in bestimmten Verhältnissen zueinanderstanden. Diese Verhältnisse änderten sich nicht, als das Licht gedimmt wurde. Daher bildet die UV-Antwort mit Phototaxis einen ratio-chromatischen Tiefenmesser. Das UV-Antwortspektrum stimmte mit dem Spektrum von C-Opsin1 überein. C-Opsin1 ist in den ziliären Photorezeptorzellen exprimiert, die Membranstapel besitzen, und somit sehr sensitiv sein könnten.

Daher könnten die ziliären Photorezeptorzellen mit C-Opsin1 die UV-Antwort vermitteln, während Phototaxis durch Go-Opsin1 und den rhabdomerischen Opsinen vermittelt wird. Go-Opsin1 scheint in den rhabdomerischen Photorezeptorzellen der definiten Augen an ein Gq-Protein zu koppeln. Darin unterscheidet sich Go-Opsin1 von einem Muschel-Go-Opsin, welches an ein Go-Protein in ziliären Photorezeptorzellen zu koppeln scheint.

Ziliäre Photorezeptorzellen sind verbreitet unter marinen Wirbellosenlarven, so dass auch der Tiefenmesser verbreitet sein könnte, ja sogar, dass er bereits vom letzten gemeinsamen Vorfahren aller Zweiseitentiere verwendet worden sein könnte. Der Tiefenmesser hilft den Larven, die richtige Tiefe im Allgemeinen zu finden, während positive und negative Phototaxis den Larven vor Ort hilft, einen Siedlungsort zu wählen.

Acknowledgements

I thank Gáspár Jékely my supervisor for all the support, the discussions, and the opportunity to work with him on such a wonderful project, which turned out to be more complex than we originally thought. I guess this is normal and the whole point of doing science. And obviously not everything is what it seems.

I thank Nico K. Michiels for visiting his lab, where we tinkered with the column setup to remove some reflections. However, the larvae did not care and swam down the same way as before to UV-light. Therefore, he concluded that the UV-response is not negative phototaxis but a response to non-directional light. This changed the view on the project and is the main idea that helped me to solve the puzzle of the depth gauge.

I thank Huiyong Jia and Shozo Yokoyama for measuring the absorption spectra of our opsins. This is not a trivial task and if I look at the literature with the Japanese quail and the chicken neuropsins, which should have the same absorption spectrum but for some technical reasons do not, then I am happy that we gave this task to them the experts.

I thank all the past and present members of the Jékely lab whose company I enjoyed and who helped me with the smaller and the bigger things that are too hard to enumerate here.

Especially, I thank Philipp Bauknecht and Sarah-Lena Offenburger for trying to measure the *in vitro* absorption spectra of the opsins; even so, they did not succeed. And I thank Philipp Bauknecht for cloning the *c-opsin1*. I thank Nadine Randel for purifying my Go-opsin1 antibody and trying to stain Go-opsin1 with it; even so she could not stain Go-opsin1. I thank Elizabeth A. Williams for giving me her analysis of the *Platynereis dumerilii* single cell transcriptome, so that I could estimate how many opsins are expressed in the ciliary photoreceptor cells.

I thank Sanja Jasek for the nice conversation with her about biology, programing, language, operating systems, browsers, and internet security.

I thank the inventors of the Internet, without it; I could not find so much relevant literature with so many topics and process it all.

I thank the Flying Spaghetti Monster for delivering us from the creationists.

And, I thank my parents Herbert and Christel, my sister Gabi, and my nephew Vincent for being around and supporting me.

Table of Contents

1 Introduction

If an organism moves to the light it is positively phototactic (photopositive). If it moves away from the light it is negatively phototactic (photonegative)(Jékely, 2009; Menzel, 1979). Phototaxis helps pelagic larvae of benthic marine invertebrates to spread. The larvae are first photopositive and thus move to the light at the water surface. At the surface, streams can move the larvae far away. Most larvae become photonegative later in their life, so that they move back to the bottom of the sea, ideally to a similar place as their parents have inhabited (Thorson, 1964) or other shaded places like ship hulls (Visscher, 1927). Larvae of different species differ in the time they stay photopositive and thus how long they stay in the open water column (Thorson, 1964). However, how far the larvae spread depends on their behavior: Models treating the larvae as passive particles overestimate the distance the larvae disperse (Koehl and Reidenbach, 2007; Shanks, 2009). Larvae may still settle close to their parents' place, even so they have lived long in the open water column (Shanks, 2009).

The open water column is a dangerous place, because there, the larvae are exposed to predators and ultraviolet (UV) light. UV-light can kill planktonic larvae (Thorson, 1964). The larvae could use pigments to protect their bodies from UV-light. However, pigments also make the larvae or other planktonic organisms more visible to predators and thus avoiding UV-light is preferable (Leech and Jonsen, 2002).

Some species only live in a 2 m layer of water (in any depth). To get there, their larvae must move away from the water surface, which they could do by negative phototaxis. But they must not swim too deep, which they could do by negative gravitaxis (Thorson, 1964), which brings them up.

However, some larvae must have also other possibilities to get down: The larvae of the hydroid *Clava multicornis* seem to choose a settlement site with phototaxis. They are photopositive if they crawl over inert surfaces, but become photonegative if they crawl over the surface of the brown alga *Ascophyllum nodosum*, their natural settling substrate (Williams, 1965). Similar are the larvae of the sinistral spiral tubeworm *Spirorbis borealis*. They also become photonegative when they contact their preferred brown alga *Fucus serratus*. *Fucus serratus* extract is enough to make them photonegative as long as it is on a surface, however the extract has no effect if it just dissolved in the seawater (Williams, 1964). In general, negative phototaxis seems to be associated with

settlement, because shaded parts of ship hulls are more fouled (Visscher, 1927), that means more larvae settle there.

If the larvae use negative phototaxis to choose a settlement site and do so by becoming photonegative locally, they need another mechanism to find the correct depth, which could be a ratio-chromatic depth gauge. Such a depth gauge is based on the different attenuation of different wavelengths in water and is independent of the absolute light intensity. For instance, monochromatic blue light (470 nm) penetrates water the deepest while UV-light (380 nm) or green light (540 nm) disappears before blue light (Lythgoe, 1988).

Such a depth gauge has been proposed for the polychaete worm *Torrea candida*. *Torrea candida* has an eye with a main retina and two accessory retinae. The main retina is maximally sensitive to UV-light (400 nm) and the accessory retinae are maximally sensitive to green-yellow light (560 nm). The light perceived from both retinae types needs only be compared to indicate the depth (Wald and Rayport, 1977). A ratio-chromatic depth gauge has also been proposed at the level of photoreceptor cells: Where a cell contains two types of opsins, one type hyperpolarizing and another type depolarizing the cell. Both types would use different signal cascades and work antagonistically (Nilsson, 2009, 2013). Such an antagonism works in the parietal eye of the lizard (Su et al., 2006), however it does not gauge depth, but detects dawn and dusk (Solessio and Engbretson, 1993). Such an antagonism was suggested to be the evolutionary reason, why opsins exist that couple to different signal cascades and thus hyperpolarize or depolarize their host cells, respectively (Nilsson, 2009, 2013).

1.1 Opsins

The first opsin was described by Boll (1876) from the isolated dark-adapted frog retina. Such a frog retina looks to the human eye purple. The purple becomes yellow and eventually colorless when the retina is exposed to light (Boll, 1876; Kühne, 1878). The purple can be mistaken as red. Therefore, Boll called the color visual purple first (Boll, 1876) and then visual red (Boll, 1877). Further, Boll speculated whether the visual red was a physical property of the rods or a rod pigment, which he would have called erythropsin if he could have isolated it. The pigment was then isolated and called visual purple by Kühne (1878). Kühne also called the yellow and the colorless pigment visual yellow and visual white, respectively. Later, Ewald and Kühne (1878) coined international synonyms for

visual purple, yellow, and white: Rhodopsin, xanthopsin, and leukopsin, respectively.

Rhodopsin consists of a colorless protein and a covalently bound retinal molecule (originally named retinene)(Wald, 1934), which is linked in 11-*cis*-conformation (Brown and Wald, 1956; Oroshnik, 1956; Oroshnik et al., 1956; Wald et al., 1955) via a protonated Schiff base (Collins, 1953; Pitt et al., 1955) to a Lysine (Bownds, 1967) in the seventh of seven transmembrane domains (Hargrave et al., 1983; Murakami and Kouyama, 2008; Palczewski et al., 2000). The colorless protein is called opsin since 1951 (Hubbard and Wald). Retinal was originally called retinene and renamed (Morton and Goodwin, 1944; Wald, 1968), afterwards it was found to be the aldehyde of Vitamin A (Ball et al., 1946, 1948). Xanthopsin is in fact a mixture of the colorless protein and free retinal, which is yellow. The retinal changes to vitamin A, which is colorless, and thus leukopsin is a mixture of a protein and vitamin A (Wald, 1934, 1935).

Rhodopsin gains its purple color by binding 11-*cis*-retinal, covalently. 11-*cis*-retinal isomerizes to all-*trans*-retinal upon light exposure and changes the conformation of rhodopsin via intermediates to metarhodopsin II (Wald, 1968) its active state (Choe et al., 2011; Hargrave, 2001). Metarhodopsin II releases all-*trans*-retinal and becomes colorless at or above 0°C. As long as metarhodopsin II has not released all-*trans*-retinal, it can be converted back to 11-*cis*-retinal by another photon (Wald, 1968). Below 0°C, metarhodopsin II does not release all-*trans*-retinal so that metarhodopsin II is as stable as rhodopsin. This state is called bistable.

The opsin terminology is complicated in the literature, because it evolved and at the beginning, people could only look at the superficial features like its color. Therefore, Hubbard and Wald (1951) distinguished between rhodopsin and opsin. Opsin is here only the protein component, while rhodopsin is the photoreceptor: Opsin with a covalently bound retinal. Koyanagi et al. (2002) and Terakita et al. (2004) use rhodopsin also for other opsins than for the opsins of the vertebrate rods, even if those opsins are not purple. Hofmann and Palczewski (2015) use rhodopsin to exclusively refer to the opsin of the vertebrate rods, irrespective whether retinal is actually bound or not. Porter et al. (2012) and Feuda et al. (2012) use rhodopsin exclusively for the opsin of the vertebrate rods and opsin for all other, irrespective weather retinal is bound or not. Here, I will also use opsin for all opsins, irrespective whether retinal is bound or not and use rhodopsin for the opsin of the vertebrate rods, only.

This view also focuses more on the molecular properties and not whether a pigment is purple or has another color. Pigment is also misleading: Even so, it is popular in the literature, because a pigment is a substance that gives its color to another substance. It does so by absorbing certain wavelengths of light and reflecting diffusely the other wavelengths to an observer. The observer interprets these wavelengths as color. However, biologically it is not about what wavelengths an opsin reflects but the wavelengths it absorbs, and thus can detect.

This may depend on the kind of retinal, for instance the rod-opsin of the bullfrog can contain either 11-*cis*-retinal or 11-*cis*-3-dehydroretinal in the same retina at the same time. The 11-*cis*-3-retinal-rod-opsin absorbs maximally at 502 nm and the 11-*cis*-3-dehydroretinal-rod-opsin at 522 nm (Reuter et al., 1971). Reuter et al. but already Wald (1937) used for these forms of opsin the terms rhodopsin and porphyropsin, respectively.

However, the absorption spectrum of an opsin largely depends on its amino-acid-sequence. The chicken short wavelength sensitive 1 (SWS1) opsin absorbs maximally at 415 nm and if serine 84 is replaced by a cysteine, it absorbs maximally at 369 nm (Yokoyama et al., 2000), which is a shift of 46 nm. Similar the human red and green opsins: They differ in 15 of their 364 amino acids, and only seven amino acids shift their absorption maxima by 31 nm from 532 nm to 563 nm. A single amino acid may shift the absorption maximum by 2 to 15 nm (Asenjo et al., 1994). This way, an opsin absorption spectrum can be fine-tuned across a range of wavelengths: The vertebrate middle/long wavelength sensitive (M/LWS) opsins maximally absorb between 500 and 580 nm if they contain 11-*cis*-retinal, if they contain 11-*cis*-3-dehydroretinal they absorb between 515 and 625 nm, maximally (Amora et al., 2008).

Opsins did not only diversify by absorption spectra but also in other ways. They belong to the superfamily of G-protein coupled receptors (GPCRs) (Terakita, 2005) and diversified into several families during the early evolution of marine metazoans. Opsins (of bilaterians) have been classified into four major groups, the xenopsins (Ramirez et al., 2016), the rhabdomeric (r-)opsins, the ciliary (c-)opsins, and the tetraopsins uniting Go-opsins, neuropsins, retinochromes and retinal G-protein-coupled receptor (RGR) opsins, and peropsins (Cronin and Porter, 2014; Delroisse et al., 2014; Feuda et al., 2012; Feuda et al., 2014; Porter et al., 2012). Whether the peropsins are separate from the retinochromes and RGR-opsins is so far unclear. These four groups trace back to the last common ancestor of cnidarians and bilaterians (Porter et al., 2012; Ramirez et al., 2016).

They may even trace back to the last common ancestor of cnidarians, ctenophores, and bilaterians (Feuda et al., 2014; Suga et al., 2008). Rhabdomeric opsins and ciliary opsins have been extensively studied, because they represent visual opsins from invertebrates and vertebrates, however we know less about the tetraopsins.

1.1.1 Ciliary opsins

The ciliary opsins are housed in ciliary photoreceptor cells and mediate vision in vertebrates. The first ciliary opsin was described by Boll (1876) (see section 1.1 above). Ciliary opsins were previously called vertebrate opsins, but the labels vertebrate and invertebrate opsins became meaningless, when melanopsin was discovered in the African clawed frog (Xenopus laevis), because melanopsin is more like invertebrate than vertebrate visual opsins (Provencio et al., 1998). Therefore, Arendt and Wittbrodt (2001) referred to vertebrate and invertebrate opsins as ciliary and rhabdomeric opsins, respectively. In fact, Arendt et al. (2004) found later a ciliary opsin in the brain of the invertebrate *Platynereis dumerilii*. The ciliary opsins of the vertebrate rods and cones have been studied and reviewed extensively (e.g. Arshavsky et al., 2002; Filipek et al., 2003; Luo et al., 2008; Yau and Hardie, 2009):

Ciliary opsins bind the chromophore 11-*cis*-retinal. When 11-*cis*-retinal is hit by a photon, it isomerizes to all-*trans*-retinal, and so changes the conformation of the opsin. This activates inside the cell a trimeric Gt-protein (also called transducin). The Gt-protein's alpha subunit activates a phosphodiesterase (PDE), which hydrolyzes cyclic guanine monophosphate (cGMP). Thus, the intracellular cGMP concentration decreases and cGMP is removed from non-selective cyclic nucleotide gated (CNG) cation channels. In darkness, the CNG channels are constantly kept open by cGMP, so that a steady inward current (dark current) is maintained, which keeps the membrane potential at -30 mV, which depolarizes the cell enough to release the transmitter glutamate, continuously. In light, the CNG channels close and the cell hyperpolarizes and stops releasing glutamate (Figure 1). This way, upon light, the cell switches from an activated state to a deactivated state. Therefore, this response to light has been called OFF-response (Tosches et al., 2014; Vopalensky et al., 2012). However, the term OFF-response is misleading, because it also has been used to denote the response of a cell when the light is switched off (Cornwall and Gorman, 1983; McReynolds and Gorman, 1970b).

Rod and cone opsins are intensively studied, because they mediate vision. Other ciliary opsins have been less studied, because they do not mediate vision or are not found in humans or at least in mammals. Other vertebrate ciliary opsins are the vertebrate ancient opsins (Soni and Foster, 1997), the encephalopsins/panopsins (Blackshaw and Snyder, 1999; Halford et al., 2001), the parapinopsins (Blackshaw and Snyder, 1997), the parietopsins (Su et al., 2006), and the pinopsins (Okano et al., 1994). A parietopsin and a pinopsin in the parietal eye of the lizard differ in their phototransduction cascades to vertebrate visual opsins. The parietopsin couples to a Go-protein that inhibits a phosphodiesterase, while the pinopsin couples to a Ggust-protein, which activates, like a Gt-protein, the phosphodiesterase. The photoreceptor cells of the parietal eye have also CNG channels and thus pinopsin hyperpolarizes while parietopsin depolarizes (Su et al., 2006) (Figure 1). Pinopsin and parietopsin antagonize each other chromatically to enhance the contrast between dawn and dust (Solessio and Engbretson, 1993; Su et al., 2006).

Ciliary opsins are also found in protostomes, like the honeybee and other insects, where they belong to the pteropsins, which are absent from *Drosophila melanogaster* (Feuda et al., 2016; Velarde et al., 2005). Pteropsins are thought to entrain the circadian clock in insects (Velarde et al., 2005). A ciliary opsin is also found in the marine annelid *Platynereis dumerilii*, the ciliary opsin is thought to entrain the circadian clock, as well (Arendt et al., 2004). Another ciliary opsin was thought to be found in the eyes of the brachiopod larva of *Terebratalia transversa* where it was suggested to mediate phototaxis (Passamaneck et al., 2011). However, this opsin was reclassified as a xenopsin (Ramirez et al., 2016).

The protostome ciliary opsins do not couple to Gt-proteins, because protostomes do not have Gt-proteins (Lagman et al., 2012; Wilkie and Yokoyama, 1994). Most likely ciliary opsins in vertebrate ancestors coupled to Gi-proteins (Lamb, 2013). The Gt-proteins are derived from Gi-proteins, because all three vertebrate Gi-protein genes form tandem repeats with Gt-protein genes (Wilkie et al., 1992; Wilkie and Yokoyama, 1994), with which they also share the introns (Oka et al., 2009).

Whether vertebrate and protostome ciliary opsins further differ is unknown, especially whether their phototransduction cascades are conserved. Protostome ciliary opsins may activate a Gi-protein since a mosquito ciliary opsin couples to a Gi-protein and less efficiently to a Go-protein (Koyanagi et al., 2013). Whether protostome ciliary opsins are bistable unlike rod and cone opsins is unknown. Other ciliary opsins are bistable at non-physiological low temperatures:

Parietopsin is bistable at -10°C and monostable at 20°C (Sakai et al., 2012), parapinopsin is bistable at 4°C, vertebrate ancient-long opsin is bistable at 0°C (Sato et al., 2011), teleost multiple tissue opsins, which belong to the encephalopsins, are bistable at 0°C (Sakai et al., 2015). However, whether this is physiologically relevant is unknown, since Rhodopsin is also bistable below 0°C (Wald, 1968).

1.1.2 Rhabdomeric opsins

The rhabdomeric opsins are housed in rhabdomeric photoreceptor cells and mediate vision in invertebrates. They were called invertebrate opsins, but when melanopsin was discovered (Provencio et al., 1998), the name was inappropriate, because melanopsin is a vertebrate rhabdomeric opsin. Since rhabdomeric opsins mediate vision in invertebrates, they have been studied and reviewed extensively (e.g. Hardie, 2012; Hardie and Juusola, 2015; Montell, 1999, 2012; Yau and Hardie, 2009; Zuker, 1996):

Rhabdomeric opsins are bistable at physiological temperatures, unlike vertebrate visual ciliary opsins. Rhabdomeric opsins bind 11-*cis*-retinal. When 11-*cis*-retinal is hit by a photon, it isomerizes to all-*trans*-retinal, and so changes the conformation of the opsin. This activates inside the cell a trimeric Gq-protein. The Gq-protein's alpha subunit activates a phospholipase C (PLC), which hydrolyzes phosphatidylinositol-4,5-bisphosphate (PIP_2) to inositol-1,4,5-trisphosphate ($InsP_3$), diacylglycerol (DAG) and a proton. This reaction opens transient receptor potential (TRP) channels and TRP like (TRPL) channels. The TRP and TRPL channels let in Calcium ions so that the photoreceptor cell is depolarized (Figure 1).

How the TRP and TRPL channels are exactly opened is still unclear (Hardie and Juusola, 2015). When in *Drosophila melanogaster*, rhabdomeric photoreceptor cells are illuminated, their membranes contract. This may be a physical effect of depleting PIP_2. PIP_2 is a membrane phospholipid and when its inositol head group is cleaved off, the membrane area is reduced, which may activate the TRP and the TRPL channels mechanically. At least the system also works if the TRP and the TRPL channels are replaced by gramicidin, a mechanosensitive channel that responds to changes in the physical properties of the cell membrane, however the TRP and the TRPL channels could still be regulated by acidification caused by the released protons (Hardie and Franze, 2012). The *Drosophila* TRP and TRPL channels are closely related to the mammalian canonical TRP (TRPC) channels (Christensen and Corey, 2007). In fact, TRPC6 can also be activated by

stretching the membrane mechanically (Christensen and Corey, 2007; Spassova et al., 2006). TRPC6 mediates in mouse with TRPC1, TRPC3, and TRPC5 the mechano-sensations hearing and touch (Quick et al., 2012; Sexton et al., 2016).
TRPC6 and TRPC7 mediate cell depolarization in intrinsically photosensitive retinal ganglion cells (Berson et al., 2002; Xue et al., 2011). These cells do not have rhabdomeres, but express a vertebrate rhabdomeric opsin: Melanopsin. Melanopsin also depolarizes via a phospholipase C (PLCβ4) (Xue et al., 2011), the TRPC6 and TRPC7 channels, and three Gq-type G-proteins: Gq, G11, and G14 (Hughes et al., 2015). The G-protein types can replace each other functionally, so that knocking out two of them does not yield a phenotype (Chew et al., 2014). This phototransduction cascade resembles the rhabdomeric phototransduction cascade in protostomes so that the urbilaterian, the last common ancestor of protostomes and deuterostomes, may have had it, already.
In *Xenopus laevis*, melanopsin is expressed in dermal melanocytes, the brain, and the eye. In the eye, it is expressed in the iris, the retinal pigment epithelium (RPE), and the inner retina (Provencio et al., 1998). In mammals, melanopsin is found in blood vessels (Sikka et al., 2014), the iris (Xue et al., 2011), and the inner retina (Provencio et al., 2000). In the inner retina, it is restricted to the intrinsically photosensitive retinal ganglion cells (Hattar et al., 2002). These cells entrain the circadian clock (Berson et al., 2002; Hannibal et al., 2002) and mediate the light reflex of the pupil (Hughes et al., 2015; Lucas et al., 2001; Lucas et al., 2003). Mouse and human melanopsins are most sensitive at 479 nm and 484 nm (cyan light), respectively (Bailes and Lucas, 2013). Since melanopsins are rhabdomeric opsins, they are to be expected to be bistable, too. However, melanopsins are either bistable or monostable (Davies et al., 2011).

1.1.3 Go-opsins

Among the tetraopsins, the Go-opsins only exist in lophotrochozoans and invertebrate deuterostomes; they are lost in both ecdysozoans (Hering and Mayer, 2014) and vertebrates (Porter et al., 2012). The first Go-opsin was found in the scallop mantle-edge eye (Kojima et al., 1997). The scallop retina has two layers of photoreceptor cells (Dakin, 1928; Küpfer, 1915), a layer of depolarizing rhabdomeric photoreceptor cells, and a layer of hyperpolarizing ciliary photoreceptor cells (Barber et al., 1967; McReynolds and Gorman, 1970b). The ciliary photoreceptor cells coexpress a Go-opsin and a Go-alpha subunit of trimeric G-proteins (Kojima et al., 1997) suggesting that the Go-opsin initiates hyperpolarization via the Go-alpha subunit (Gomez and Nasi, 2000; Kojima et al.,

1997). The Go-alpha subunit activates a nitric-oxide-insensitive guanulylate cyclase, which hydrolyses GTP to cGMP (Gomez and Nasi, 2000). The cGMP opens potassium selective CNG channels, leading to cell hyperpolarization (Cornwall and Gorman, 1983; Gomez and Nasi, 1995; Gomez and Nasi, 1997; Gorman and McReynolds, 1978) (Figure 1). The hyperpolarizing response is most sensitive at 500 nm (McReynolds and Gorman, 1970a), suggesting that the scallop Go-opsin is a cyan-green opsin. Only tree more Go-opsins are characterized beyond their sequence: A Go-opsin expressed in the gastrula of *Terebratalia transversa* (Passamaneck and Martindale, 2013). A Go-opsin expressed in two cells flanking the apical organ of the pluteus larva of *Strongylocentrotus purpuratus* (Valero-Gracia et al., 2016). And an amphioxus Go-opsin, which was cloned from complementary DNA (cDNA) and characterized *in vitro*: The amphioxus Go-opsin binds 11-*cis*-retinal, it absorbs maximally at 483 nm when 11-*cis*-retinal is bound, and it can convert back 11-*cis*-retinal from all-*trans*-retinal by absorbing another photon at 0°C and 20°C (Koyanagi et al., 2002; Tsukamoto et al., 2005).

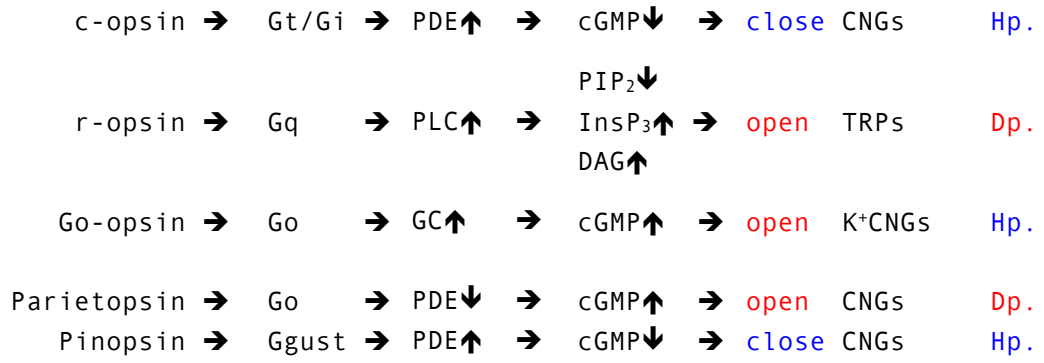


Figure 1: The different bilaterian phototransduction cascades

The different phototransduction cascades found in bilaterian animals. Top to bottom: The ciliary phototransduction cascade (as found in vertebrate vision, however in invertebrates the Gt-protein may be only replaced by a Gi-protein, but other differences are possible); the rhabdomeric phototransduction cascade; the Go-opsin mediated phototransduction cascade of the scallop eye; and the phototransduction cascades mediated by parietopsin and pinopsin of the parietal eye of the lizard. Each opsin acts on it G-protein, which then acts on its effector enzyme either by activating (↑) or deactivating (↓) it. The effector enzyme may add (↑) or remove (↓) second messenger molecules from the cell or the cell membrane. For the rhabdomeric cascade, all molecules involved are shown, because which of the molecules is the second messenger or what the exact mechanism is, is not clearly known. The second messengers open or close cation channels, which leads either to depolarization (Dp.) or hyperpolarization (Hp.) of the cell. The channels are cyclic nucleotide gated non-selective cation channels (CNGs), transient receptor potential cation channels (TRPs), or potassium selective cyclic nucleotide gated channels (K^+CNGs). The second messengers or molecules involved are cyclic guanosine monophosphate (cGMP), phosphatidylinositol-4,5-bisphosphate (PIP_2), inositol-1,4,5-trisphosphate ($InsP_3$), or diacylglycerol (DAG). The effector enzymes are phosphodiesterase (PDE), phospholipase C (PLC), or guanulylate cyclase (GC).

Go-opsins are difficult to study, because scallops and amphioxus are difficult to manipulate experimentally and they are missing in ecdysozoans and vertebrates, the classes that contain our classical genetically tractable model organisms. Therefore, their physiological functions are still unknown. However, Gáspár Jékely found a Go-opsin expressed in the nectochaete larva of Platynereis dumerilii (Figure 2A, D, F, J, L).

1.2 Platynereis dumerilii: A model organism

Platynereis dumerilii was originally described as *Nereis dumerilii* by Audouin and Milne-Edwards (1834) and was later reassigned to the *Platynereis* genus (Fauvel, 1914; Read, 2015). It is a marine annelid worm also called Dumeril's clamworm. *Platynereis dumerilii* lives in a wide range: In the waters of the Azores, the Mediterranean, the North Sea, the English Channel, and the Atlantic down to the Cape of Good Hope. It also lives in the Black Sea, the Red Sea, the Persian Gulf, the Sea of Japan, the Pacific, and the Kerguelen Islands (Fauvel, 1914).

1.2.1 *Platynereis dumerilii* ecology

Platynereis dumerilii lives in 0 to 5 m depth in tubes on the substrate. It lives on pelagic Sargassum rafts in the Sargasso Sea (Fine, 1970; Huffard et al., 2014). It has been found on a cliff in 0 to 2 m depth on algae covered hard bottoms (Giangrande, 1988), but also on sea grass (Jacobs and Pierson, 1979) and in shallow sea grass beds in 1.7 m depth (Lewis and Stoner, 1981). Additionally at sites in 1 and 3 m depth (Gambi et al., 1992), and in 5 m depth (Giangrande et al., 2003). And so, it is typical of shallow infra-littoral photophilic environments (Giangrande et al., 2003). However, *Platynereis dumerilii* was also found on a buoy in 50 m depth (Aliani and Meloni, 1999) and on rotting seaweed in 100 m depth (Cram and Evans, 1980). It may also live in less favorable habitats, like rotting plant debris (Clark and Milne, 1955), thermal vents (Giménez and Marín, 1991; Lucey et al., 2015), or polluted areas near sewer outfall pipes (Surugiu and Feunteun, 2008). In general, it dominates polluted areas (Bellan, 1980; Musco et al., 2009) or areas of pH values around 6.5 (Ricevuto et al., 2014), which also fits to the preferred pH value of a subpopulation of late *Platynereis dumerilii* nectochaete larvae (Ramanathan et al., 2015).

Platynereis dumerilii eats what it lives on: In its gut, sea lettuce, sediment, and brown algae were found (Cram and Evans, 1980). Whether the brown algae are fresh or rotting does not matter (Bedford and Moore, 1984, 1985). *Platynereis dumerilii* prefers to eat and settle on algae that are not eaten by omnivorous/herbivorous fish (Hay et al., 1988). *Platynereis dumerilii* is

considered to be herbivorous, in the field (Fauchald and Jumars, 1992). In our lab, it eats organic spinach, the green alga *Tetraselmis marinus*, and the brine shrimp *Artemia*.

1.2.2 *Platynereis dumerilii* reproduction

Platynereis dumerilii worms live three to 18 months (Fischer and Dorresteijn, 2004; Hauenschild and Fischer, 1969). Then, they metamorphose into male and female pelagic epitokes. The epitokes are the sexually mature forms. The epitokes swarm to the surface, in fact they are attracted by light and can be collected with a lamp there (Korringa, 1947). At the surface, when the males and females are close together, they start a nuptical dance. While dancing, they release sperm and eggs so that the eggs are fertilized in the water (summarized by Zeeck et al., 1998). Then, the epitokes die (Fischer and Dorresteijn, 2004; Fischer et al., 2010).

Because the epitokes die, *Platynereis dumerilii* is a mass spawner and lays batches from 2000 to 3000 (Fischer and Dorresteijn, 2004), several 1000 (Hutchinson et al., 1995), or from 20000 to 40000 eggs (Jha et al., 1995). From my experience, all these estimates are reasonable and the actual batch size depends on the size of the mother.

The epitoke metamorphosis is controlled by a circalunar body clock. The clock is entrained by the moon. The moon determines, when most worms become epitokes. The moon can be simulated with light as low as 0.02 Lux (full moon 0.2 Lux) (Hauenschild, 1955). The moon lengthens the day by additional illumination, so that periods of short and long days alternate. A long day can be entirely illuminated by light of the same intensity, so that long and short days only differ by length. Here, the relative length matters: For instance, short and long days can have 0 h and 16 h, respectively, or 16 h and 24 h, respectively (Hauenschild, 1961). When the short-day-period begins, the worms start to metamorphose. The worms need 16 to 20 days, even in an artificial month of ten short and ten long days (Hauenschild, 1956). However, if all days have the same length then the worms become epitokes on every day without a swarming maximum (Hauenschild, 1955, 1956; Zantke et al., 2013).

The swarming maximum stays at the same days, even so if the worms are not entrained by the shift from long to short days anymore. The maximum diminishes over three to four months, then the worms become epitokes on every day (Hauenschild, 1956; Zantke et al., 2013). The swarming maximum can also be shifted: The worms are set into a new artificial mooncycle that is shifted for

instance by two weeks. The worms need two months to adapt: In the first month, most worms are not shifted, yet. In the second month, most worms are shifted. And in the third month, all the worms are shifted into the new cycle (Hauenschild, 1955, 1956).

1.2.3 *Platynereis dumerilii* development and life cycle

When the *Platynereis dumerilii* epitokes have released sperm and eggs, the fertilized eggs start to develop. The eggs divide asymmetrically into a smaller AB and a bigger CD cell. The AB cell divides symmetrically into an A and a B cell. The CD cell divides asymmetrically into a C and a D cell (Dorresteijn, 1990). The D cell is the biggest cell and eventually forms the germ line (Rebscher et al., 2007; Zelada-González, 2004). Other cell lines can also be tracked, because the embryo is transparent (Dorresteijn, 1990). The cell lines develop very stereotypically between individuals (Dorresteijn, 1990; Fischer and Arendt, 2013).

Although, the *Platynereis dumerilii* embryo and larva develop very stereotypically, their developmental speed depends on the temperature, therefore the following times are given with a reference temperature of 18 °C: 24 hours after fertilization, the *Platynereis dumerilii* embryo has developed into a trochophore larva. The trochophore larva has a spherical shape and has a diameter of 180 µm. It is divided by a prototroch into an upper head and a lower trunk region. The prototroch is a band of cilia, which the larva uses for swimming. After 48 hours, the trochophore larva has developed into a metatrochophore larva. The metatrochophore larva has a longer trunk, which makes it cone shaped. The trunk also bears chaetae. Both the trochophore and the metatrochophore larva are pelagic, this means they swim in the water column. After three days, the metatrochophore larva has developed into a nectochaete larva. The nectochaete larva becomes benthic and is errant; that means it stops swimming in the open water column and settles on a suitable substrate, but still can move around. It is 160 µm wide (left to right) and 300 µm long. The nectochaete larva has three clear segments, each with a pair of parapodia and chaetae. The nectochaete larva is longish and resembles a short worm (Fischer et al., 2010). Most larvae start feeding between six and eight days after fertilization (Fischer and Dorresteijn, 2004; Fischer et al., 2010; Williams et al., 2015). Then, the larvae, depending on food intake, develop at individual speed. When the larvae have five segments, the first segment is incorporated into the head. This is called cephalic metamorphosis. After cephalic metamorphosis, *Platynereis dumerilii* worms live in tubes, which they build with their spinning

glands. The worms grow until they have 50 segments, then they start to produce immature gametes. When the worms have reached 70 segments, the gametes start to mature, the atokous worms metamorphose into the sexually mature epitokes (Fischer et al., 2010). The epitokes swarm to the water surface and release there their eggs and sperm, so that the life cycle completes.

1.2.4 *Platynereis dumerilii* as a model animal

Platynereis dumerilii lives at many places, copes with many conditions, even unfavorable conditions, and produces many relatively fast developing offspring. Therefore, *Platynereis dumerilii* is suited for the lab. It has been kept in the lab since 1953 (Fischer and Dorresteijn, 2004). It is suited for developmental studies, because the embryo is transparent and thus the cell divisions can be easily followed (Dorresteijn, 1990), and because the embryo develops very stereotypically (Fischer and Dorresteijn, 2004), until it has become a feeding nectochaete larva (Fischer et al., 2010).

Platynereis dumerilii is not only a model organism for developmental biology but also for evolution (Simakov et al., 2013): It has been considered to be ancient that means it has less derived characters than ecdysozoa, for instance (Miller and Ball, 2009). It can teach us about the last common ancestor of all bilaterians, the urbilaterian, because it has fewer group specific characters (Tessmar-Raible and Arendt, 2003), it has conserved cell types (Tessmar-Raible et al., 2007), and a conserved genome organization (Raible et al., 2005).

The *Platynereis dumerilii* genome has approximately 1 Gbp, however it is not published, yet (Zantke et al., 2014, http://4dx.embl.de/platy/, password protected). The genome is organized into 2n = 28 chromosomes (Ipucha et al., 2007; Jha et al., 1995). And compared to ecdysozoan genomes such as *Drosophila melanogaster* or *Caenorhabditis elegans*, it is intron rich like vertebrate genomes (Raible et al., 2005), and so it is less derived than those from classical invertebrate models.

1.2.5 Studying gene expression in *Platynereis dumerilii*

In *Platynereis dumerilii*, gene expression can be studied by staining proteins with antibodies (Dorresteijn et al., 1993), or by labeling mRNA with *in situ* hybridization probes, in larvae (Arendt et al., 2001; Tessmar-Raible et al., 2005), in juvenile worms with posterior regenerating segments (Backfisch et al., 2013; Prud'homme et al., 2003), or in adults (Backfisch et al., 2013). *In situ* probes can be fluorescent (Tessmar-Raible et al., 2005) or non-fluorescent (Arendt et al., 2001). Even non-fluorescent *in situ* probes can be imaged on a fluorescent

confocal microscope with a special reflection technique (Jékely and Arendt, 2007). Two *in situ* probes can be used on the same specimen to detect gene coexpression within the same cell. The two probes can be both fluorescent or one can be fluorescent and the other non-fluorescent. This technique is called double *in situ* hybridization (Tessmar-Raible et al., 2005).

Double *in situ*, however, can only detect coexpression of a few genes. For many genes, the mRNA expression patterns from many individuals can be mapped into expression atlases, because not only the development of *Platynereis dumerilii* is stereotypical but also the gene expression patterns. Gene expression atlases exist for the metatrochophore (Asadulina et al., 2012; Tomer et al., 2010) and the nectochaete larva (Asadulina et al., 2012). The atlases can be easily visualized by the software Blender so that coexpression can be easily seen (Asadulina et al., 2015).

Atlases are limited to existing *in situ* probes for known genes. Unknown genes can also be found in transcriptomes. In *Platynereis dumerilii*, transcriptomes exist for different stages of the whole body (Conzelmann et al., 2013a) and for single cells of the metatrochophore head. The single cell transcriptomes allow creating virtual *in situ* patterns if the known genes are mapped against an expression atlas (Achim et al., 2015).

In *Platynereis dumerilii*, gene expression can be studied with high temporal resolution by quantitative PCR, which is however limited to a preselected set of genes (Dray et al., 2010; Tosches et al., 2014; Zantke et al., 2013).

1.2.6 Modifying gene expression in *Platynereis dumerilii*

Gene expression in *Platynereis dumerilii* can be modified in several ways. The genes can be knocked down with morpholinos (Conzelmann et al., 2013b; Shahidi et al., 2015; Williams et al., 2015), which bind to the mRNA, and may block either the start site or a splice site so that the mRNA is not translated or is misspliced (Eisen and Smith, 2008). Ideally, missplicing creates a premature stop-codon that also triggers nonsense mediated decay (NMD reviewed by Lejeune and Maquat, 2005), which removes the mRNA. However, morpholinos are restricted to the early developmental stages, because morpholinos have to be injected into fertilized eggs and are diluted out during development (Eisen and Smith, 2008).

The genes can be modified at their mRNA sequence by A-to-I RNA editing, which may optionally be controlled by light (Hanswillemenke et al., 2015). A-to-I RNA editing can change the meaning of codons including start and stop-codons. In

principle, also splice sites can be modified (Bass, 2002; Nishikura, 2010). This can create mRNA with a premature stop-codon that triggers NMD, or without stop-codons, a condition that triggers non-stop decay (Vasudevan et al., 2002). In both cases, the mRNA is removed from the cell.

Genes, in other organisms, can be knocked out with zinc-finger-nucleases (ZFNs), transcription activator-like effector nucleases (TALENs), or the CRISPR-Cas9 system (clustered regularly interspaced short palindromic repeat; Chandrasegaran and Carroll, 2015). In *Platynereis dumerilii*, knockouts via ZFNs (Tosches, 2013) and TALENs (Bannister et al., 2014) have been reported.

Additionally, genes can be introduced transiently: For instance, to express a calcium indicator (Gühmann et al., 2015; Randel et al., 2014; Tosches, 2013; Tosches et al., 2014; Verasztó et al., 2017; Williams et al., 2015). Or permanently with germ-line transmission via transposons to express GFP for the whole life (Backfisch et al., 2014; Backfisch et al., 2013; Veedin-Rajan et al., 2013).

1.2.7 The connectome of *Platynereis dumerilii*

In the *Platynereis dumerilii* larva, the neurons and their connections can be reconstructed via serial-section transmission electron microscopy (Randel et al., 2014). The neurons and their connections are also stereotypical between individuals (Randel et al., 2015). The neurons can be immuno-gold labeled in single electron microscopy sections for neuropeptides. The neuropeptides identify a neuron by gene expression to the resolution of electron microscopy (Shahidi et al., 2015). The neuropeptide expression can be linked to expression of other genes in the same cell and possibly also to neighboring cells via a neuronal atlas (Asadulina et al., 2012). The expression could also be linked to single cell transcriptome data if single cell data were available from the same larval stage. From the neurons and their connections, circuits can be reconstructed that allow understanding behaviors like putative chemosensory behaviors (Shahidi et al., 2015), phototaxis (Randel et al., 2014) or how different ciliated fields are synchronized across the whole body (Verasztó et al., 2017).

1.2.8 Studying behaviors in *Platynereis dumerilii*

Many behaviors exist in *Platynereis dumerilii* and so, many ways to observe them:

The adults can be observed in a petridish, how they rebuilt their tube (Daly, 1973). Or in a box with video recording how they spend their time on searching for food, irrigating their tube, fighting with each other, or being idle: Either to

compare *Platynereis dumerilii* with other species (Cram and Evans, 1980) or to distinguish day from night-behavior (Zantke et al., 2013).
Nectochaete larvae can be observed under a microscope, how they bend when one eye is illuminated. The bending determines their swimming direction, which can be observed in a horizontal cuvette (Randel et al., 2014).
Trochophore larvae can also be observed under a microscope, how their cilia beat differently when one larval eye is illuminated. This allows the larvae to turn to the light, which can be observed in a horizontal cuvette (Jékely et al., 2008). The cilia also respond to neuroactive compounds supplied to the surrounding seawater (Tosches et al., 2014), including neuropeptides that regulate the depth of the larvae, which can be observed in a vertical column (Conzelmann et al., 2011); for instance, myo-inhibitory peptide brings the larvae down. It makes them settle (Conzelmann et al., 2013b) and controls their feeding behavior (Williams et al., 2015), both can be observed in a single culture dish.
The larvae can be assayed for their preferred pH-value and salt concentration in a laminar flow microfluidic device (Ramanathan et al., 2015).

1.2.9 Cell ablation – a way to study the eyes of *Platynereis dumerilii*

Platynereis dumerilii eyes and their function can be studied by cell-ablation: The eyes can be ablated with a laser and in the adult with an electrode, too. Additionally, the eyes can be chemically ablated. When the larval eyes are laser-ablated in trochophore larvae, the larvae are not phototactic anymore (Jékely et al., 2008). However, if the larval eyes are laser-ablated in nectochaete larvae, the larvae are still phototactic, but if the adult eyes are ablated, the larvae are not phototactic anymore (Randel et al., 2014). Therefore, phototaxis is mediated in the trochophore larva by the larval eyes; and in the nectochaete larva by the adult eyes. When the eyes in the juvenile worm are laser ablated, the worms still entrain their circadian body clock (Keplinger, 2010), therefore the eyes do not entrain the circadian clock, at least not alone. When the eyes of the adult atoke worms are electro-ablated, the atokes still synchronize epitoke metamorphosis to the mooncycle (Hauenschild, 1961). However, the eyes regenerate partially within a week, so that some cells still could entrain the circalunar clock. This problem can be solved by chemical ablation. Here, the target cells are made to express nitroreductase. Nitroreductase produces a cytotoxic substance from metronidazole. Metronidazole needs only be added to the seawater surrounding *Platynereis dumerilii*, to kill any nitroreductase expressing cell. The cells can be

killed anytime during development if nitroreductase is expressed permanently (Veedin-Rajan et al., 2013).

1.2.10 The photoreceptor cells and phototaxis in *Platynereis dumerilii*

The larval eyes of the *Platynereis dumerilii* trochophore larva consist of a pigment cell and a rhabdomeric photoreceptor cell (Rhode, 1992). The photoreceptor cell expresses a rhabdomeric opsin, r-opsin3 (Randel et al., 2013), and is shaded by the pigment cell, so that it only detects light from one direction. With a pair of these eyes, which innervate the prototroch directly, the larva can swim phototacticly to the light, but to do so, the larva must rotate. The trochophore larva is only positively phototactic (Jékely et al., 2008).

However, *Platynereis dumerilii* nectochaete larvae are positively and negatively phototactic. The phototaxis sign is switched within the nervous system, but the exact sensory input that switches the sign is unknown. Phototaxis in the nectochaete larvae is mediated by two pairs of adult eyes. The adult eyes do not innervate the prototroch directly, but relay the signals to a visual processing center that creates a four-pixel-image, which tells the larva from where the light is coming from, so that it does not need to rotate anymore. This visual phototaxis may be advantageous for living at the bottom of the sea (Randel et al., 2014).

The adult eyes are already developing in the metatrochophore larva (Rhode, 1992) and express two rhabdomeric opsins, r-opsin1 and r-opsin3 (Randel et al., 2013), before they become functional. When they become functional (in the 3-day-old nectochaete larva), they express at least three opsins (Figure 2A-C), which are in the same cells: The two rhabdomeric opsins and a Go-opsin called *Go-opsin1* (Figure 2D-I). *Go-opsin1* is also cellularly coexpressed in the larval eye with *r-opsin1* (Figure 2J-L), but not with *r-opsin3*, because *r-opsin3* is not expressed in the larval eye (Randel et al., 2013). The larval eye's function in the nectochaete larva is unknown. *Go-opsin1* is also expressed in another, slightly asymmetrical median cell (Figure 2A), however its function there is also unknown.

Additionally, to the rhabdomeric photoreceptor cells of the eyes, the larva has deep brain ciliary photoreceptor cells, which are very prominent in the acetylated tubulin staining (Figure 2A-C). The ciliary photoreceptor cells exist already in the 2-day-old metatrochophore larva (Arendt et al., 2004). They are found in many polychaetes. They have stacked membranes and no shading pigment (Arendt et al., 2004; Hausen, 2007; Purschke, 2005). This indicates that

they respond to non-directional light with a short integration time, which points to a UV avoidance response (Nilsson, 2009) or a shadow response.

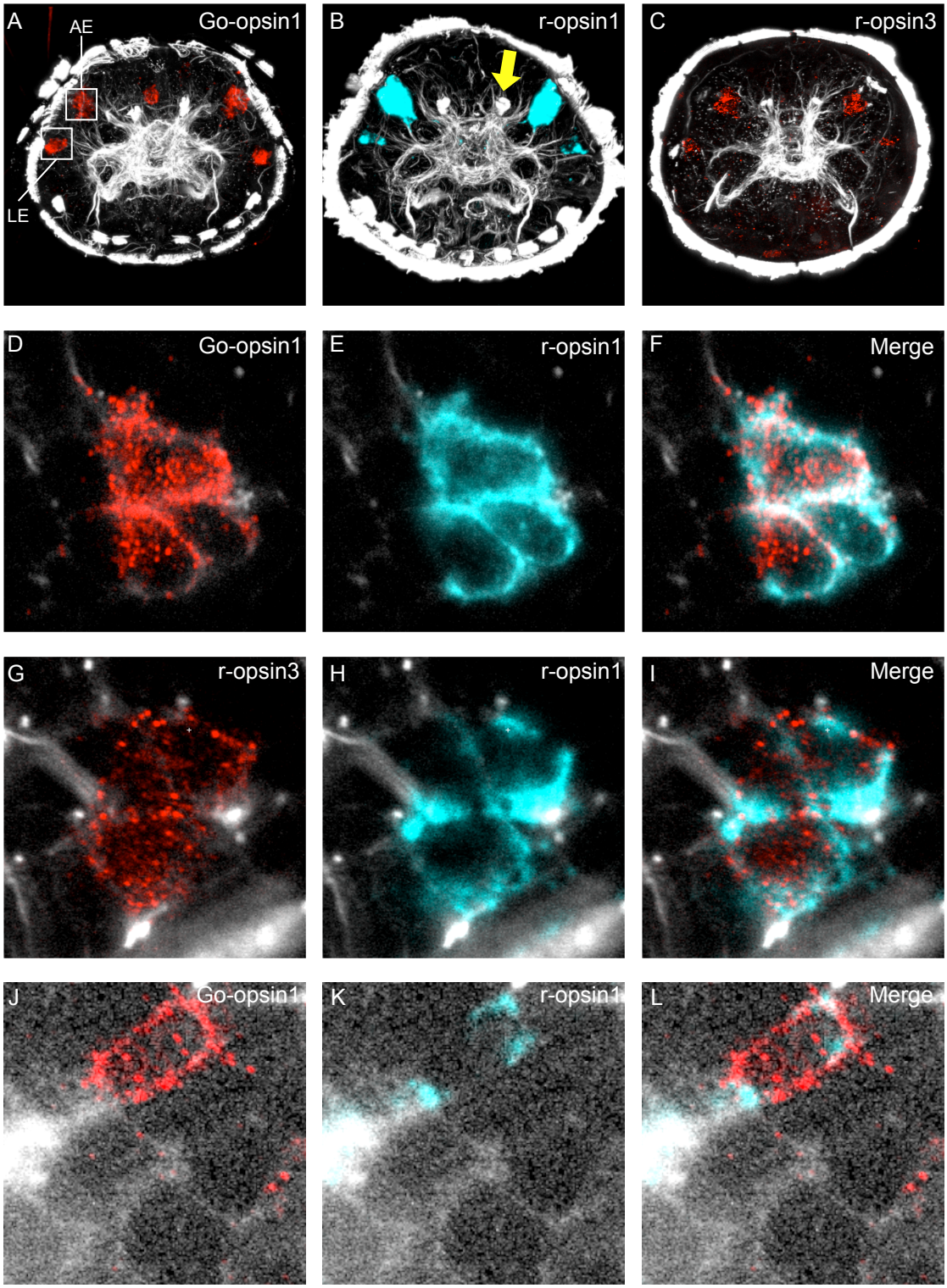


Figure 2: Opsin expression in the eyes of the nectochaete larva of *Platynereis dumerilii*

Opsin mRNA expression in 3-day-old nectochaete larvae of *Platynereis dumerilii* was stained via whole mount *in situ* hybridization. The larvae were double stained with a non-fluorescent (red) and a fluorescent (cyan) *in situ* probe. Two kinds of opsin mRNA, were stained per larva. The larvae were counter stained with an antibody against acetylated tubulin (white), which marks neurons and cilia.

1st row: Z-stack projections of the head nervous system and the ciliary band (trochophore) of different larvae with *Go-opsin1* (A), *r-opsin1* (B), and *r-opsin3* (C) expression; apical views with dorsal up and ventral down. The yellow arrow points at two ciliary photoreceptor cells, which are visible in the tubulin staining.
2nd row: A close-up from a confocal section of the adult eye of one larva. D: *Go-opsin1* is labeled with a non-fluorescent *in situ* probe. E: *R-opsin3* is labeled with a fluorescent *in situ* probe. F: Merge of D and F.
3rd row: A close-up from a confocal section of the adult eye of another larva. G: *R-opsin3* is labeled with a non-fluorescent *in situ* probe. H: *R-opsin1* is labeled with a fluorescent *in situ* probe. I: Merge of G and H.
4th row: A close-up from a confocal section of the larval eye of one larva. J: *Go-opsin1* is labeled with a non-fluorescent *in situ* probe. K: *R-opsin1* is labeled with a fluorescent *in situ* probe. L: Merge of J and K. The acetylated tubulin channel has been enhanced to visualize the cell membrane.
Abbreviations: AE, adult eye; LE, larval eye. Images from the same raw data have been published in the meantime (Gühmann et al., 2015; Randel et al., 2013). Images and raw data are courtesy to Gáspár Jékely.

However, the ciliary photoreceptor cells have been speculated to entrain the circadian body clock (Arendt et al., 2004), especially because they are surrounded by cells that express clock related genes in both the larva (Tosches et al., 2014) and the adult (Zantke et al., 2013). The ciliary photoreceptor cells resemble molecularly the photoreceptor cells of the vertebrate retina and the pineal organ (Tosches, 2013). The pineal organ entrains the circadian clock in many non-mammalian vertebrates (Vigh et al., 2002), while in mouse the circadian clock is also entrained by rod and cones (Panda et al., 2003). Therefore, entraining the circadian clock may be an ancient function of ciliary photoreceptor cells, but so far, what the function of the ciliary photoreceptor cells in *Platynereis dumerilii* is, is unknown.

1.3 The goals

Platynereis dumerilii expresses in its eyes a Go-opsin. Go-opsins are not very well studied, because they are lost in ecdysozoans and vertebrates, to which our classical model animals belong. However, since *Platynereis dumerilii* has this rich toolbox to manipulate genes and to study its behavior, I can use it to study a Go-opsin and its physiological function. *Go-opsin1* is expressed in the adult eye, which is the only place of expression with a known function. Therefore, I focused on the adult eye and its function: Visual phototaxis.
At first, I tested the larvae in a vertical column illuminated with green (520 nm) and UV (400 nm) light coming from the top. The larvae swam up to green light and swam down to UV-light (see results section 3.1). I originally thought this was positive and negative phototaxis, respectively. I hypothesized that Go-opsin1 switched the larvae between positive and negative phototaxis by forming a chromatic antagonism with the rhabdomeric opsins within the same cell. This would have implemented a chromatic depth gauge that is independent of the absolute light intensity as proposed by Nilsson (2009, 2013).

Therefore, I generated a *Go-opsin1* knockout line with zinc-finger-nucleases (ZFN). But the knockout larvae still swam up to green light and swam down to UV-light. This falsified my hypothesis, but I had found another behavior: It responded fast to UV-light and was also non-directional. So that it may be mediated by the ciliary photoreceptor cells.
Therefore, my two goals were: 1st to identify the function of Go-opsin1 in the adult eye in the context of phototaxis and 2nd to characterize and to identify the function of the UV-induced down-swimming behavior.

2 Material and Methods

2.1 *Platynereis dumerilii* culture

From our laboratory culture, I used wild type *Platynereis dumerilii* larvae and worms. The larvae and worms were handled by all members of the lab according to established breeding procedures derived from those of Hauenschild and Fischer (1969). Below, I describe the main procedures so that I can also describe how I treated my mutant worms differently.

2.1.1 *Platynereis dumerilii* batches

For *Platynereis dumerilii* batches, natural seawater was filled into a 100 ml glass beaker, which had been autoclaved, and one male and one female epitoke worm (sometimes more) were added. Once the worms had released their gametes, they were removed and the beaker was tagged with the date and time as the moment of fertilization. When the eggs had settled, most water was poured off to remove excess sperm. The remaining sperm was diluted by refilling the beaker to avoid polyspermy. The fertilized eggs develop jelly, which the larvae leave after 24 hours. 24 hours later, the empty jelly was removed with a 10 ml pipette (Eppendorf Research 10 ml), so that the jelly could not rot. The batches were kept at 18°C or 22°C until they were used for experiments or cultured.

2.1.2 Culturing batches and worm culture

The Batches were cultured four to six days after fertilization in a culture box. The box received five to eight batches and was filled up with a one-to-one mixture of natural and artificial seawater, so that 1 or 2 cm water covered the larvae and later the worms. The larvae and the worms were kept at 18°C or 22°C, in an artificial day/night cycle of 16 h light and 8 h darkness. In each month, seven

consecutive nights were illuminated by an artificial moon, which was a 15 W light bulb.

2.1.3 Water change in the culture

The water was changed the first time when the worms were two months old or big enough. Then, the water was changed every second week. The old water was poured from the culture box into the sink without caring whether a worm was lost. The boxes with older worms where cleaned with brushes to remove algae and worm feces.

2.1.4 *Platynereis dumerilii* feeding

In the boxes, the worms were fed with algae-fish-soup, spinach, and *Artemia* each week.

The algae-fish-soup was mixed from finely grinded dry flake fish food and the benthic flagellate alga *Tetraselmis marinus*, which was cultured in natural seawater. The algae-fish-soup was given in different amounts to the worms depending on their age and size. The worms got 5 ml before the first water change, and afterwards 10 ml.

The spinach was bought minced from organic farming. The spinach was only given to worms after the first water change. The smaller worms received less spinach than the bigger worms, but not too much so that the spinach would not rot.

The *Artemia* were grown in natural seawater and were given to all boxes that had a water change, the amount was 10 ml for each box.

2.1.5 *Go-opsin1* knockout mutant culture

I cared about my *Go-opsin1* knockout mutant worms, alone. The mutants in the boxes were exclusively kept at 22°C. I changed their water, gave them extra food, and crossed them.

At water change, the boxes were cleaned less vigorously so that fewer worms were brushed out accidentally and that the algae film of the box stayed intact. The algae film served as additional food source, so that the worms could grow faster. However, the worms only eat green algae, probably *Tetraselmis marinus*, therefore every biofilm that was red or black was removed. Additionally, the old water was not poured into the sink directly, but into another box first, to recover any worm that was poured out. This is very important for worms that had been injected as eggs, because every such worm could found a mutant line.

For each batch, only one female and one male were crossed. The worms that were homozygous mutants were crossed before the worms that were heterozygous, to minimize contamination by accidental sperm carryover. Contamination was also excluded by genotyping each batch. Worms that did not release their gametes were squeezed with the side of a plastic Pasteur-pipette to release their gametes. The worms were pressed against the wall of the beaker. The males were only pressed gently, so that they would not be split. The pipette was moved along the body to the tail, to release some sperm, but not too much to avoid polyspermy. The females were squeezed in the middle so much that they were split. The pieces were squeezed along the length axis to the split site to release the eggs there.
This gave good batches, even so the survival of the eggs and larvae seemed to be reduced. This is in contrast to what Hauenschild and Fischer (1969) reported. They could only retrieve viable sperm from males that were cut and could not obtain viable eggs from females.

2.1.6 Genotyping: Single worms in six-well-plates

During and after worm genotyping, I kept the worms in six-well-plates (Nunc multidish #150239, Thermo Scientific) in a mixture of one-to-one artificial and natural seawater. The worms were fed as the ones in the boxes, but with less food so that it could not rot. The worms got two to three drops of *Artemia* and algae-fish-soup from a plastic Pasteur-pipette, and got one or two leaves of spinach. The worms got fresh water every second week.
The worms were kept in the wells until they became sexually mature. The mature worms were crossed to another worm with a matching genotype. If there was no worm with a matching genotype, the lonely worm was put to 4°C in a fridge. This allowed the worm to survive one night; exceptionally a worm could survive up to five days, however this did not guarantee that a worm would survive the first night.

2.2 Opsin intron/exon annotation

The introns and exons of *Go-opsin1*, *Go-opsin2*, *r-opsin2*, and *r-opsin5* were annotated. Their complementary DNA (cDNA) sequences were searched with BLAST (Altschul et al., 1990) in a database of genomic and cDNA sequences of *Platynereis dumerilii* (http://4dx.embl.de/platy/, unpublished, password protected) for matches. The matches were assembled with Velvet (Zerbino and Birney, 2008) and viewed in the assembly viewer Tablet (Milne et al., 2009). In Tablet, the introns were identified by the part of the genomic DNA sequence that

did not match to the cDNA sequence and by the intron start GT and end AG bases. The introns of *Go-opsin1* were annotated by Gáspár Jékely; I annotated the introns of *Go-opsin2*, *r-opsin2*, and *r-opsin5*. For the translated opsin, sequences with annotated introns see the appendix section: Opsins translated with annotated introns.

2.3 *Go-opsin1* knockdown and knockout

I used morpholinos to knockdown the expression of *Go-opsin1* and zinc-finger-nucleases to generate a *Go-opsin1* knockout line. I injected the morpholinos and the zinc-finger-nucleases into freshly fertilized *Platynereis dumerilii* eggs.

2.3.1 The injection setup

The injection setup consisted of an Axiovert 40 CFL inverted microscope equipped with an A-Plan 5x/0,12 Rh0 objective for specimen location and an A-Plan 10x/0,25 Rh1 objective for injection. The setup had a temperature controllable Mini PCT chamber III (Luigs & Neumann) controlled by a temperature controller type TC05 (Badcontroller V, Luigs and Neumann) with a Cyclo II water pump (Roth) for removing the heat. The injections were done with a FemtoJet (Eppendorf) microinjector equipped with a Femtotip II (Eppendorf) needle. The needle was mounted on a Luigs and Neumann motorized micromanipulator and loaded with the injection solution.

2.3.2 The injection procedure

Several *Platynereis dumerilii* batches were set up and incubated at 18°C for 45 min. The best batch was used. Good batches have eggs that have developed a lot of jelly. The jelly also creates between each egg some distance, which is ideally the same between each egg.

The eggs were poured into a sieve within a beaker. The eggs were always covered by natural seawater, because eggs exposed to air may die. The eggs were rinsed until the water flew through the sieve without resistance, which indicated that all the jelly was gone. Usually, 0.5 l natural seawater was needed. The dejellied eggs were treated with Proteinase K (final concentration: 70 µg/ml) for 1 min to soften their cuticle. The Proteinase K was removed by rinsing the eggs with another 0.5 l natural seawater.

300 to 400 eggs were put into a groove on a 2% agarose gel made with natural seawater. The grove was 1 mm wide and was limited by an upper and a lower wall. The lower wall had scratches for removing the eggs. The gel was put into the lid of a little petridish (Nunc diam. x H 35 mm x 10 mm, Thermo Scientific)

and covered by natural seawater. The petridish was placed in the Mini PCT chamber III of the injection setup. The petridish was kept at 14.5 °C during the injection, to slow down egg development. Each egg was injected by pressing it against the higher wall of the gel until the needle went in, then it was injected, and finally striped off at the scratches of the lower wall.

The injection was started with an injection pressure of 600 hPa, a compensation pressure of 35 hPa, and a manual injection time. Injection pressure and injection time were adjusted during the injection session to compensate for partial needle clogging. During injection, the egg plasma was observed for clearance, indicating the injected volume. The injection volume was targeted to be 10 % of the total egg volume so that the egg would not be destroyed.

After injection, the injected eggs were collected from the gel and put into a well of a six-well-plate (Nunc multidish #150239, Thermo Scientific). Uninjected eggs from the gel were placed into another well of the same plate as control.

2.3.3 *Go-opsin1* expression knockdown with morpholinos

I ordered the morpholinos to knockdown *Go-opsin1* from Gene Tools. Gene Tools designed and synthesized four morpholinos: The first to block the start site, the second for the start site with five mismatches as control, the third to remove exon 5, and the fourth to keep intron 3. The morpholino sequences are given in Table 1. The underlined sequences are the reverse complement of the coding regions. The lower-case letters indicate mismatches. The morpholinos were dissolved and diluted in water before injection, their respective final concentrations are given in Table 1.

Table 1: The *Go-opsin1* morpholinos I injected

Morpholino Name	Sequence	Final Concentration
Go-opsin1 Start Site	GTGTGATTAAATTCCATGGTTACTT	0.5 mM
Go-opsin1 Start Site with five mismatches	GTGTcAaTAAATTgCATGcTTAgTT	0.5 mM
Go-opsin1 Exon 5 Skip	CCATCGTCATCTGAAAGGTCAAGAT	0.2 mM
Go-opsin1 Intron 3 Keep	AATTGTCTGGAGTGAATTACCTTAT	0.45 mM

Only one morpholino was injected per session to maximize the number of larvae per batch and condition. After injection, uninjected eggs were set apart as control. Three days later, the injected and the control larvae were used to study their behavior in a horizontal high-intensity phototaxis setup (Gühmann et al.,

2015). If the larvae were injected with a splice site morpholino, they were recovered after the experiment to check for misspliced mRNA. The mRNA was extracted with trizol, reverse transcribed, and the exon or the intron was PCR amplified from the neighboring exons. The PCR-product was sequenced.

2.3.4 *Go-opsin1* zinc-finger-nucleases

The *Go-opsin1* zinc-finger-nucleases (ZFN) were designed, produced, and validated by Sigma-Aldrich (CompoZr™ Custom Zinc Finger Nucleases). Sigma-Aldrich placed the ZFNs into the first exon of *Go-opsin1*, which lies before the first transmembrane domain of Go-opsin1. Ideally, the ZFNs should have avoided a single nucleotide polymorphism, but Sigma-Aldrich could not avoid one. Sigma-Aldrich tested the ZFNs with the MEL-1 reporter assay (Doyon et al., 2008). The *Go-opsin1* ZFNs showed 59% activity six hours after induction relative to the positive control ZFNs of Sigma-Aldrich. Sigma-Aldrich considers ZFNs that show more than 50% relative activity six hours after induction "as useful for genome editing". The *Go-opsin* ZFNs did not show any activity in the uninduced state. Sigma-Aldrich regards those ZFNs that also show activity in the uninduced state "as superior".

The ZFN recognition site was AAGGTGGAAGCTGAAatattGAATGCCATGGGTAC. The binding sites are given in upper case and the cut site in between is given in lower case. The sequence is the complement of the coding sequence.

Sigma-Aldrich shipped the ZFNs encoded by plasmids (pZFN1 (pVax-3FN-27507-FokKK) and pZFN2 (pVAX-N2A-3FN-27506-FokEL2)) and by mRNA ready for injection. I only injected the mRNA from Sigma-Aldrich in a final concentration of 40 ng/µl.

2.3.5 Genotyping of larvae and worms

For genotyping of the *Go-opsin1* locus, genomic DNA was isolated from single larvae, groups of 20 larvae, or from the tentacular cirri of adult worms. The worms were placed into a well of a six-well-plate (Nunc multidish no. 150239, Thermo Scientific) and their cirri were taken with a pair of forceps. Either the cirri were torn off by me or by the worms, which seems to be an escape response. The DNA was isolated with the dilution protocol of the Phusion Human Specimen Direct PCR Kit (Thermo Scientific). The protocol slightly differed between larvae and cirri (Table 2). The DNA was amplified by PCR (forward primer: 5'-CTGCTGAATGCCATTAGTTGACG-3', reverse primer: 5'-AACACCAATGACCATATAGACCCG-3') with the reagents (Table 3) and the instructions (Table 4) of the PCR Kit. The PCR product comprised 258 bp of the

first exon of *Go-opsin1*. The PCR product was sequenced directly in our in-house sequencing facility with a nested sequencing primer (5'-CCAAATTGGACAAGAAAAGTAACC-3') to avoid sequencing of unspecific PCR product. In a PCR-product sample, a mixture of wild type and deletion alleles gave double peaks in the sequencing chromatograms, with the relative height of the double peaks reflecting the relative allele ratio in the sample (Figure 3).

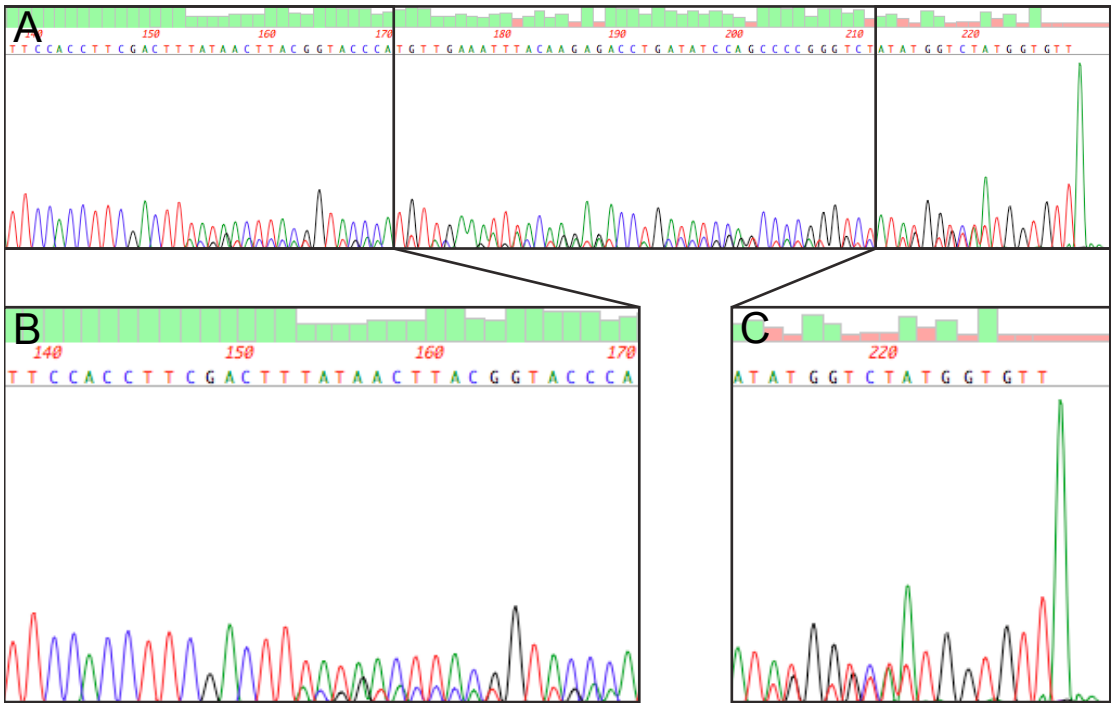


Figure 3: Sequencing chromatogram illustrates the genotyping method

This is the sequence chromatogram of 20 larvae of the *Go-opsin1* knockout founder batch. The chromatogram illustrates the method. A) shows an overview of the last 90 bp of the PCR fragment used for genotyping. B) shows a close-up on the zinc-finger-nuclease binding site, where the double peaks are starting. The minor peaks are 1/3 of the size of the major peaks, which fits to the expected allele ratio of 1/4 if the father was wild type and the mother was heterozygous or at least her complete germ line. The major peaks have the sequence TATAACTTACGGTACCC. The minor peaks have the sequence ACGGTACCC, which is identical to the major peaks except that the first 8 bp are missing. C) shows a close-up of the end of the chromatogram with the end of the double peaks including the big terminating adenine peak. The double peaks also end 8 bp before the last peak. Note that I use this figure to illustrate the method, however it also includes the result of the *Go-opsin1* 8 bp deletion, which is summed up in Figure 7B.

Table 2: Sample collection and tissue lysis for genotyping

Larvae as sample	Cirrus as sample
Take 20 larvae, remove seawater as much as possible	Add 50 µl Dilution Buffer
Add 50 µl Dilution Buffer	Add cirrus
Add 1.5 µl DNARelease Additive	Add 1.5 µl DNARelease Additive
Incubate at RT for 5 min	Incubate at RT for 5 min
Incubate at 98 °C for 2 min	Incubate at 98 °C for 2 min

Table 3: *Go-opsin* PCR reaction for genotyping

Reagent	Amount
PCR grade water	7.1 µl
2x Phusion Human Specimen PCR Buffer	10.0 µl
Forward *Go-opsin1* Primer F001 10 µM	1.0 µl
Reverse *Go-opsin1* Primer B001 10 µM	1.0 µl
Phusion Human Specimen DNA Polymerase	0.4 µl
Tissue Lysis Solution	0.5 µl
Total Volume	20.0 µl

Table 4 *Go-opsin1* PCR cycling conditions for genotyping

Cycle Step	Temperature	Time	Cycles
Initial denaturation	98 °C	5:00	1
Denaturation	98 °C	0:01	40
Annealing	65 °C	0:05	
Extension	72 °C	0:30	
Final Extension	72 °C	1:00	1

2.3.6 *Go-opsin1* mutant crossing

Larvae from eggs injected with *Go-opsin1* zinc-finger-nucleases were kept at 18°C for five to eight days in six-well-plates (Nunc multidish no. 150239, Thermo Scientific) and then cultured at 22°C until sexual maturity. The mature worms were crossed to wild type worms to produce an F1 generation batch.

From each batch, 20 larvae were pooled and genotyped. This gave one founder batch. Its sequence chromatogram showed double peaks with minor peaks one third as high as the major peaks (Figure 3). This matched the expected allele ratio of one to three in the F1 generation (Figure 4). From the founder batch, the other larvae were distributed across three culture boxes, so that the larvae had

more space to become mature worms. 96 of the worms were genotyped before they became mature and were distributed into the wells of six-well-plates and kept there until sexual maturity. The mature worms that were heterozygous were crossed with each other to produce F2 generation batches.

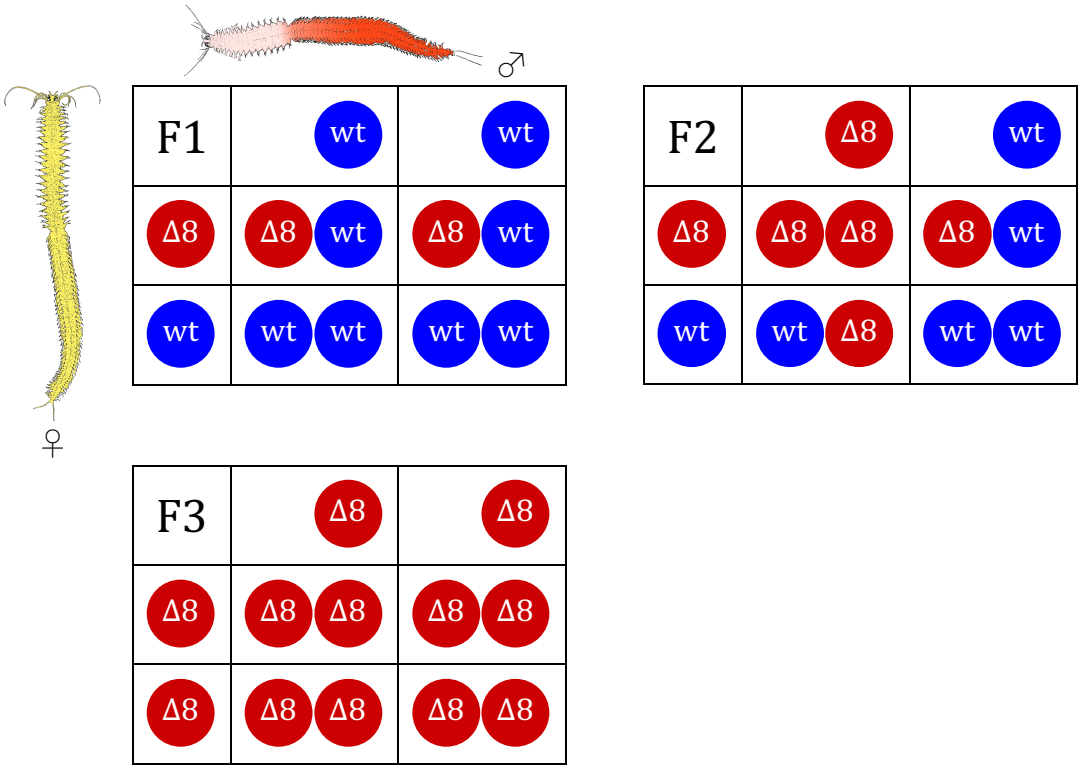


Figure 4: Mutant Crossing schemes

The worms with the *Go-opsin1*$^{\Delta 8}$ knockout allele were crossed to obtain a homozygous knockout line within three generations. The crossing started with a founder mother and a wild type father to obtain the F1 generation. In the F1 generation, half of the individuals carried the knockout allele, so that a quarter of the alleles in the F1 generation were knockout alleles. This means that the mother or at least her germ line was heterozygous for the knockout allele. From the F1 generation, two heterozygous worms were crossed to obtain the F2 generation. In the F2 generation, a quarter of the individuals were homozygous mutants, half of them were heterozygous mutants, and a quarter of them were homozygous wild types. Thus, in the F2 population half of the alleles were knockout alleles. From the F2 generation, two homozygous mutant worms were crossed to obtain the F3 generation. In the F3 generation, all individuals were homozygous for the knockout allele and thus in the population all alleles were knockout alleles. The drawings of a male and a female Platynereis epitoke worms were taken from Fischer et al. (2010), who tock and modified the drawings from Hauenschild and Fischer (1969). Fischer et al. (2010) published the drawings under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/2.0).

The F2 batches were also genotyped by pooling 20 larvae. This gave sequence chromatograms with double peaks that had equal size, matching the expected allele ratio of one to one in the F2 generation (Figure 4). The F2 larvae were also raised to adulthood and genotyped before they became sexually mature. Since the F2 generation had only half as many homozygous worms than the F1

generation had heterozygous worms (Figure 4), 192 instead of 96 worms were genotyped. The homozygous worms were crossed to each other to produce F3 generation batches. The F3 batches were also genotyped to confirm their status. The genotyping gave also data to check whether the *Go-opsin1* knockout allele was inherited in a Mendelian ratio compared to the wild type allele.

2.4 Opsin absorption spectrum measurement

Originally, we wanted to measure the *in vitro* spectra of the opsins ourselves. Therefore, I amplified full-length *Platynereis dumerilii r-opsin1*, *r-opsin3*, *r-opsin4 Go-opsin1*, and *peropsin1* from cDNA of an expressed sequence tag (EST) library by PCR. The PCR primers were designed in the following way:

The forward primers contained nine to ten spacer bases that matched the endogenous sequence 13 bases upstream of the start codon. The twelve bases in between were filled with a BamHI site and a Kozak sequence (GCCACC) (Kozak, 1987). The primers contained another 20 to 30 specific bases starting at the start codon.

The reverse primers contained 20 to 50 specific bases upstream of the endogenous stop-codon. Between the specific bases and the stop-codon a rho 1D4 tag sequence was inserted. The stop-codon was followed by an XbaI site and seven to nine unspecific spacer bases. The PCR-products were cloned into the BamHI and XbaI sites of pcDNA3.1 (+) vectors (Invitrogen).

The vectors were transfected into HEK 293 cells by Sarah-Lena Offenburger (2011) and Philipp Bauknecht (2013) to express the opsins. Philipp Bauknecht also reconstituted the opsins with retinal and tried to measure their spectra. However, he could not measure a spectrum. Therefore, we gave the opsins plus *c-opsin1*, which was cloned by Philipp Bauknecht in the same way, to experts: Huiyong Jia and Shozo Yokoyama.

They purified the opsins and measured their spectra, as Yokoyama (2000) did. However, they used their own expression system. Therefore, the opsins from the pcDNA3.1 (+) vectors were amplified by PCR with overhang primers containing EcoRI, Kosak, and SalI sequences. The PCR-products were cloned into the EcoRI and SalI restriction sites of pMT5 expression plasmids. The plasmids were transfected into COS1 cells to express the opsins. The opsins were incubated with 11-*cis*-retinal (a gift from Dr. Rosalie K. Crouch at Storm Eye Institute, Medical University of South Carolina, and National Eye Institutes) to regenerate photopigments. The pigments were purified with an immobilized rho 1D4 antibody (The Culture Center, Minneapolis, MN) in buffer W1 (50 mM N-(2-

hydroxyethyl) piperazine-N'-2-ethanesulfonic acid (HEPES) (pH 6.6), 140 mM NaCl, 3mM MgCl2, 20% (w/v) glycerol and 0.1% dodecyl maltoside) (Yokoyama, 2000). The pigments' UV/VIS spectra were recorded at 20°C with a Hitachi U-3000 dual beam spectrophotometer. The spectral data were analyzed with Sigmaplot software (Jandel Scientific, San Rafael, CA).

The last paragraph is modified and adapted from Gühmann et al. (2015) and was not originally written by me.

2.5 Photobehavior: The experimental assays

To study the photobehavior of *Platynereis dumerilii* larvae, I build several setups and designed several protocols. I build a beaker setup to measure phototaxis in a horizontal environment, so that the larvae had more space around and above them compared to previous flat cuvette setups (Gühmann et al., 2015; Jékely et al., 2008; Randel et al., 2014). I build a column setup that allowed me to study the photobehavior of the larvae in response to light coming from the top, diffusely from the side, or from the bottom. And I build a vertical cuvette setup to study how the larvae react to light coming from the side of different wavelength. For the setups, I was supported by our metal and electro-workshops. I also wrote or modified programs to analyze the data and to control a monochromator.

2.5.1 The horizontal phototaxis assay

Horizontal phototaxis of 3-day-old *Platynereis* larvae (nectochaete) was assayed in 100 ml glass beakers. Each beaker contained 50 to several 100 larvae. The larvae had been exposed to daylight in their culturing incubator before the experiment. During the experiment, the larvae were exposed to alternating monochromatic/white light stimuli from opposite directions. The white light was provided by a halogen cold-light source (Schott KL 2500 LCD, Schott AG, Mainz, Germany). The monochromatic light was provided by a monochromator (Polychrome II, Till Photonics). The monochromatic light passed a diffuser (Ø1" 20° Circle Pattern Diffuser, ED1-C20; Thorlabs) and a collimating lens (LAG-65.0-53.0-C with MgF2 Coating, CVI Melles Griot) before it hit the column. The light could be blocked with a shutter (ultra-thin shutter 04UTS201, shutter controller 04ISC850; both CVI Melles Griot) placed between the diffuser and the lens.
The larvae were stimulated with monochromatic light from 340 nm to 680 nm in 20 nm steps. The last step was followed by a dark phase, to record a dark measurement for background subtraction. The larvae were illuminated before each monochromatic light stimulus with white light from one side of the beaker

to redistribute them. Afterwards, they were stimulated with monochromatic light from the other side so that the larvae turned 180 degrees. Their displacement along the light vector was measured between 15-30 s after light direction change. All monochromatic and white light stimuli, as well as the dark phase, lasted 30 s (Table 5). The behavior was recorded at 10 Hz with an AxioCam MRm camera (Carl Zeiss AG, Jena, Germany) mounted on a Zeiss Stemi 2000-CS stereomicroscope. During recording, the beaker was illuminated from the bottom with transmitted white light filtered by a 750 nm long-pass filter (BrightLine HC 736/LP, AHF Analysentechnik). The camera, the light, and the shutter were controlled by the program AxioVision 4.8.2.0 (Carl Zeiss MicroImaging GmbH).

Table 5: Horizontal protocol: 340 nm to 680 nm (duration: 00:19)

Wavelength	From	Increment per cycle	Duration	Cycles
White	Left	-	30 s	18
340 nm to 680 nm	Right	20 nm	30 s	
Dark	-	-	30 s	1

2.5.2 The vertical column setup for measuring photoresponses

Photoresponses of larvae of different ages were assayed in a vertical Plexiglas column (31 mm x 10 mm x 160 mm water height). The column was illuminated from top with light from a monochromator (Polychrome II, Till Photonics). The monochromator was controlled by AxioVision 4.8.2.0 (Carl Zeiss MicroImaging GmbH) via analog voltage or a custom written program via the serial port. The light passed a collimator lens (LAG-65.0-53.0-C with MgF2 Coating, CVI Melles Griot) before it hit the column. The column was placed in an aquarium for temperature buffering. Under the aquarium and the column, three UV (395 nm) or three green (525 nm) light-emitting diodes (LEDs; SMB1W-395 Roithner Lasertechnik) were placed to stimulate the larvae from the bottom. The LEDs were run at 4 V (overvoltage for green LEDs) and were manually controlled. The column was hold by a scaffold. The scaffold held two arrays of light-emitting diodes (LEDs). One array contained UV (395 nm) LEDs (SMB1W-395, Roithner Lasertechnik) mounted along infrared (810 nm) LEDs (SMB1W-810NR-I, Roithner Lasertechnik). The UV LEDs were run at 4 V to stimulate the larvae in the column from the side. The infrared LEDs were run at 8 V (overvoltage) to illuminate the larvae in the column. The second LED array contained more infrared (810 nm) LEDs (ELD-810-525, Roithner Lasertechnik), which were run at 9 V (overvoltage), to illuminate the larvae for the camera (DMK 22BUC03, The

Imaging Source), which recorded videos at 15 frames per second and was controlled by IC Capture (The Imaging Source). IC Capture saved the videos in the uncompressed Y800 monochrome avi-file format. Since the camera and the monochromator were controlled by different programs, the videos and the monochromator protocols were started separately.

2.5.3 The protocols for the vertical column

I used several protocols to study the behavior of the larvae in the vertical column. Before a protocol was started, the larvae were mixed to distribute them equally. The first protocol (Table 6) tested whether the larvae could repeatedly switch between UV induced down-swimming and visible light induced up-swimming. Therefore, the larvae were stimulated six times alternatively with UV (380 nm) and green (520 nm) light. Each stimulus lasted 4.5 min. The last 3 min were analyzed for vertical displacement of the larvae. The light was provided by the monochromator, which was controlled by AxioVision.

Table 6: Vertical protocol 1: 380 nm 520 nm switching (duration: 01:05)

Wavelength	Duration	Cycles
Dark	4.5 min	1
380 nm	4.5 min	6
520 nm	4.5 min	
Dark	4.5 min	1

The second (Table 7) and third protocol (Table 8) measured the photoresponses of the larvae across the spectrum from 340 nm to 680 nm. Since the larvae were swimming down below 420 nm and swimming up above 420 nm, the protocols overlapped with their ranges of wavelengths. This way, the larvae's switching point could be determined and the results of the two protocols could be combined into one spectrum. The results of both protocols agreed about the switching point at 420 nm. The light was provided by the monochromator, which was controlled by AxioVision.

Table 7: Vertical protocol 2: 340 nm to 480 nm alternating with 520 nm (duration: 01:05)

Wavelength	Increment per cycle	Duration	Cycles
Dark	-	3.5 min	1
520 nm	-	3.5 min	8
340 nm to 480 nm	20 nm	3.5 min	
Dark	-	3.5 min	1

The second protocol (Table 7) stimulated the larvae with monochromatic light from the top between 340 nm and 480 nm in 20 nm steps. Between the stimuli, additional 520 nm stimuli were introduced to redistribute the larvae within the column. Each stimulus lasted 3.5 min. The last 2 min were analyzed for vertical displacement of the larvae.

The third protocol (Table 8) stimulated the larvae with monochromatic light from the top between 400 nm and 680 nm in 20 nm steps. Between the stimuli, additional 400 nm stimuli were introduced to redistribute the larvae within the column. Each stimulus lasted 3.5 min. The last 2 min were analyzed for vertical displacement of the larvae.

Table 8: Vertical protocol 3: 420 nm to 680 nm alternating with 400 nm (duration: 01:50)

Wavelength	Increment per cycle	Duration	Cycles
Dark	-	3.5 min	1
400 nm	-	3.5 min	14
420 nm to 680 nm	20 nm	3.5 min	
Dark	-	3.5 min	1

The fourth protocol stimulated the larvae with UV (395 nm) or green (525 nm) light from LEDs at the bottom. The LEDs were controlled manually. The larvae were stimulated for 5 min after a 5 min dark period. The larvae's horizontal displacement was analyzed in an interval from 1 to 1.5 min after stimulus onset. Weather UV or green light was given first, was randomized.

The fifth protocol (Table 9) stimulated the larvae with UV (395 nm) light coming from LEDs from the side to check whether the larvae would react to non-directional UV-light. Afterwards the larvae were stimulated with cyan (480 nm) light coming from the top to check for phototaxis. Each stimulus lasted 4 min. The LEDs were controlled manually and the monochromator was controlled via AxioVision but also manually.

Table 9: Vertical protocol 5: 395 nm from side and 480 nm from top (duration: 00:12)

Wavelength	Duration
Dark	4 min
395 nm from side LEDs	4 min
480 nm from top	4 min

The sixth protocol (Table 10) stimulated the larvae with UV-cyan (380 nm, 480 nm) ratiometric light from the top. The protocol started with 660 nm a

wavelength the larvae can hardly see. Then the larvae were exposed to UV-light, which was stepwise reduced. In each step, 10 % UV-light was replaced by cyan light. Each step was followed by a cyan (480 nm) light stimulus to redistribute the larvae. For instance, the 90 % of UV-light ratio was created by giving UV-light for 450 ms and 50 ms, which was repeated until the stimulus ended. The light was optionally dimmed by a 1 OD neutral density filter (CSND-4-100.0M, CVI Melles Griot). The light was given by the monochromator, which was controlled via the serial port by a custom java program (See section 2.5.4). Each light condition lasted 4 min.

The seventh protocol (Table 11) is a modification of the sixth protocol. The ratiometric ratio of UV-cyan light was replaced by a ratio of UV-red (380 nm, 660 nm) light.

Table 10: Vertical protocol 6: 380 nm : 480 nm ratiometric alternating with 480 nm (duration: 1:20)

Wavelengths	Ratio times	Ratio	% of 380 nm	Duration
660 nm				4 min
380 nm		10:0	100 %	4 min
480 nm				4 min
380 nm : 480 nm	450 ms : 50 ms	9:1	90 %	4 min
480 nm				4 min
380 nm : 480 nm	400 ms : 100 ms	8:2	80 %	4 min
480 nm				4 min
380 nm : 480 nm	350 ms : 150 ms	7:3	70 %	4 min
480 nm				4 min
380 nm : 480 nm	300 ms : 200 ms	6:4	60 %	4 min
480 nm				4 min
380 nm : 480 nm	250 ms : 250 ms	5:5	50 %	4 min
480 nm				4 min
380 nm : 480 nm	200 ms : 300 ms	4:6	40 %	4 min
480 nm				4 min
380 nm : 480 nm	150 ms : 350 ms	3:7	30 %	4 min
480 nm				4 min
380 nm : 480 nm	100 ms : 400 ms	2:8	20 %	4 min
480 nm				4 min
380 nm : 480 nm	50 ms : 450 ms	1:9	10 %	4 min

Table 11: Vertical protocol 7: 380 nm : 480 nm ratiometric alternating with 480 nm (duration: 1:20)

Wavelengths	Ratio times	Ratio	% of 380 nm	Duration
660 nm				4 min
380 nm		10:0	100 %	4 min
480 nm				4 min
380 nm : 660 nm	450 ms : 50 ms	9:1	90 %	4 min
480 nm				4 min
380 nm : 660 nm	400 ms : 100 ms	8:2	80 %	4 min
480 nm				4 min
380 nm : 660 nm	350 ms : 150 ms	7:3	70 %	4 min
480 nm				4 min
380 nm : 660 nm	300 ms : 200 ms	6:4	60 %	4 min
480 nm				4 min
380 nm : 660 nm	250 ms : 250 ms	5:5	50 %	4 min
480 nm				4 min
380 nm : 660 nm	200 ms : 300 ms	4:6	40 %	4 min
480 nm				4 min
380 nm : 660 nm	150 ms : 350 ms	3:7	30 %	4 min
480 nm				4 min
380 nm : 660 nm	100 ms : 400 ms	2:8	20 %	4 min
480 nm				4 min
380 nm : 660 nm	50 ms : 450 ms	1:9	10 %	4 min

2.5.4 Custom java program for controlling the monochromator

The monochromator can give light switching rapidly between two wavelengths, a feature I used to stimulate the larvae with two wavelengths simultaneously. However, this feature cannot be activated via analog voltage, which uses AxioVision, only. Therefore, I used the serial port to control the monochromator. For that, Till Photonics provides the program TILL ADC Communication, which allows setting the wavelength steps manually. This is however cumbersome and error-prone.

Therefore, I customized a java program, so that it can send commands from a list to the monochromator via the serial port. I added also a timer so that each command can be send after a certain time. The commands and their meanings can be looked up in the help file of TILL ADC Communication. The java program I

modified is BlackBox from the Java Communications API. It already, provided an interface, so that I only needed to add the timer, a way to load and run the protocol to its Transmitter.java file (Code 15) and recompile it. The protocol file to run the 380/480 nm ratios in BlackBox is given by Code 1.

The command SR connects to or disconnects from the monochromator; ID gets the monochromator's ID; WL sets the wavelengths; PD sets an alternating wavelength to the memory of the monochromator; and PC activates the wavelength protocol. The first three numbers are time intervals, the first is the dead time, which the monochromator needs to switch between wavelengths, the second is the time for the first wavelength, and the third is the time for the second wavelengths. The times are given in tenth of ms, not ms as the manual claims. The last two numbers are the first and second wavelength. The keyword Wait is for BlackBox to wait the specified seconds. The minimum number of seconds between commands is 1; otherwise, the monochromator cannot be controlled properly. Therefore, 1 s is removed from the one-wavelength only periods to set the wavelength protocol; and 1 s later, it is activated.

Code 1: Protocol file for the 380/480 nm ratio for BlackBox

```
SR,0; Wait 1
ID; Wait 1
WL,660; Wait 238
WL,380; Wait 240
WL,480; Wait 239
PD,30,500,4470,380,480; Wait 1
PC,0; Wait 240
WL,480; Wait 239
PD,30,1000,3970,380,480; Wait 1
PC,0; Wait 240
WL,480; Wait 239
PD,30,1500,3470,380,480; Wait 1
PC,0; Wait 240
WL,480; Wait 239
PD,30,2000,2970,380,480; Wait 1
PC,0; Wait 240
WL,480; Wait 239
PD,30,2500,2470,380,480; Wait 1
PC,0; Wait 240
WL,480; Wait 239
PD,30,3000,1970,380,480; Wait 1
PC,0; Wait 240
WL,480; Wait 239
PD,30,3500,1470,380,480; Wait 1
PC,0; Wait 240
WL,480; Wait 239
PD,30,4000,970,380,480; Wait 1
PC,0; Wait 240
WL,480; Wait 239
PD,30,4500,470,380,480; Wait 1
PC,0; Wait 240
SR,1;
```

2.5.5 The vertical cuvette setup for measuring photoresponses

2-day-old *Platynereis dumerilii* larvae were assayed in a vertical cuvette of 10 mm x 10 mm x 42 mm (L x W x H). The larvae were kept at 18°C and had been exposed to daylight before the experiment. The larvae were illuminated with a monochromator (Polychrome II, Till Photonics) controlled by AxioVision from one side and were illuminated from the other side with the same light reflected by a mirror (PFSQ20-03-F01, Thorlabs). The light passed a diffuser (Ø1" 20° Circle Pattern Diffuser, ED1-C20; Thorlabs) and a collimating lens (LAG-65.0-53.0-C with MgF2 Coating, CVI Melles Griot) before it hit the cuvette. The cuvette was illuminated with infrared (850 nm) light-emitting diodes (LEDs) (L2X2-I3CA, Roithner Lasertechnik) from the side. The LEDs were run at 6.0 V. The larvae were stimulated with light from 340 nm to 680 nm in 20 nm steps. Each step lasted 1 min. The steps were separated by 1 min darkness, so that the larvae could redistribute after each stimulus (Table 12). The larvae were recorded at 16 frames per second with a DMK 23GP031 camera (The Imaging Source) controlled by IC Capture. The camera was equipped with a macro objective (Macro Zoomatar 1:4/50-125 mm, Zoomar Muenchen). It was mounted with a close-up lens (+2 52 mm, Dörr close-up lens set 368052). The larvae were tracked and their vertical displacement was analyzed during the last 45 s of each stimulus period for the UV-response and for the first 30 s for phototaxis.

Table 12: Vertical cuvette protocol: 340 nm to 680 nm (duration: 00:39)

Wavelength	Increment per cycle	Duration	Cycles
Dark	-	1 min	1
Dark	-	1 min	18
340 nm to 680 nm	20 nm	1 min	
Dark	-	1 min	1
Dark	-	1 min	

2.6 Photobehavior: The data analyses

The larvae in the videos were tracked with ImageJ macros using the plugin mTRack2. The resulting tracks were analyzed with a Perl script that extracted the average displacement of the larvae along the axis parallel to the light vector, which was in the video either the x- or the y-axis. The ImageJ macros and Perl scripts were derived from previous scripts (Conzelmann et al., 2011) and heavily modified. For horizontal phototaxis in the beaker, the displacement values for each wavelength and each batch were normalized to the last dark measurement,

by subtracting the displacement value of the dark measurement from the measurements for each wavelength.

2.6.1 The ImageJ macros

I wrote three ImageJ macros, one for the horizontal beaker (Code 9), one for the vertical column (Code 10), and one for the vertical cuvette (Code 11). The macros ask either for a list of input and output directories or for one input and one output directory. The input directory should only contain video files; otherwise, the macro will stop processing. The video files can be in all formats ImageJ can read. The macros may remove the first n frames, which the user can specify. This feature allows adjusting for the delay between the manual video start and the manual start of the monochromatic light protocol. However, this feature was only used for the vertical column.

Then the macros cut the videos in intervals of 30 s (Code 9, Code 10) or 15 s (Code 11), which corresponds to 50, 450, and 240 frames, respectively. The frames are here specified, because the macros do not know the frame rates of the videos. The macros may remove additional frames at the start or the end of the cut interval. This allows cutting a video into intervals without having to cut it precisely. For instance, when the light stimulus changes then the larvae change also their behavior and when the light stimulus changes than the illumination in the videos changes, which may be a problem for further processing: The macros convert the videos into black and white only, by removing the background, filtering, and converting all the pixel values below a certain threshold to black. This results in moving black dots representing the larvae. The larvae are tracked with the ImageJ plugin mTrack2. The tracks are saved in a text file. Also, the distribution of the larvae is saved to another text file in a text image format.

The three macros differ how they convert the videos to black and white. This depends weather for instance the larvae are brighter or darker than the background. The macros also differ how mTrack2 tracks the larvae. For instance, in the vertical cuvette videos, the larvae are bigger than in the column; therefore, the particle size and their velocities, which mTrack2 receives as arguments, are also bigger.

2.6.2 ImageJ modifications

To automate the data analysis in ImageJ 1.46r, I modified three source files of ImageJ and recompiled it. For recompiling, I used the integrated development environment NetBeans 7.1.2.

The first file I modified was AVI_Reader.java. There I modified the showDialog method of the AVI_Reader class. The method is given below (Code 2) and the added code is underlined. This way, ImageJ does not ask for user input when it is opening an avi file and if it runs in automatized macro mode.

I also modified the readMovieData method (Code 3) of the AVI_Reader class by adding the actual number of frames the ImageStack object will contain so that it can allocate all the needed memory in advance instead of having to reallocate memory, which is time consumptive. For that I had to fix one of the ImageStack (Code 4) constructors in the file ImageStack.java.

The third file was LutApplier.java. There I modified the apply method of the LutApplier class (Code 5). This way, ImageJ does not throw an error, if nothing is to be done in macro mode. In fact, this is not an error; it just stops the ImageJ macro analyzing the data.

Code 2: Modified showDialog method of the AVI_Reader class of ImageJ

```
/** Parameters dialog, returns false on cancel */
    private boolean showDialog (String fileName) {
        if (IJ.isMacro()) {
            // If we run a marcro we don't want to be asked everytime
            firstFrame    = 0;
            lastFrame     = dwTotalFrames;
            isVirtual     = false;
            convertToGray = false;
            flipVertical  = false;
            return true;
        }
        if (lastFrame!=-1)
            lastFrame = dwTotalFrames;
        if (!IJ.isMacro()) {
            convertToGray = staticConvertToGray;
            flipVertical = staticFlipVertical;
            isVirtual = staticIsVirtual;
        }
        GenericDialog gd = new GenericDialog("AVI Reader");
        gd.addNumericField("First Frame: ", firstFrame, 0);
        gd.addNumericField("Last Frame: ", lastFrame, 0, 6, "");
        gd.addCheckbox("Use Virtual Stack", isVirtual);
        gd.addCheckbox("Convert to Grayscale", convertToGray);
        gd.addCheckbox("Flip Vertical", flipVertical);
        gd.showDialog();
        if (gd.wasCanceled()) return false;
        firstFrame = (int)gd.getNextNumber();
        lastFrame = (int)gd.getNextNumber();
        isVirtual = gd.getNextBoolean();
        convertToGray = gd.getNextBoolean();
        flipVertical = gd.getNextBoolean();
        if (!IJ.isMacro()) {
            staticConvertToGray = convertToGray;
            staticFlipVertical = flipVertical;
            staticIsVirtual = isVirtual;
        }
        IJ.register(this.getClass());
        return true;
```

```
    }
```

Code 3: The modified part of readMovieData method of the AVI_Reader class of ImageJ

```
/**Read from the 'movi' chunk. Skips audio ('..wb', etc.), 'LIST'
'rec' etc, only reads '..db' or '..dc'*/
    void readMovieData(long endPosition) throws Exception,
IOException {
        if (verbose)
            IJ.log("MOVIE DATA "+posSizeString(endPosition-
raFile.getFilePointer())+timeString());
        if (verbose)
            IJ.log("Searching for stream "+streamNumber+": '"+
                    fourccString(type0xdb)+"' or
'"+fourccString(type0xdc)+"' chunks");
        if (isVirtual) {
            if (frameInfos==null)                          // we might
have it already from reading the first chunk
                frameInfos = new Vector<long[]>(100);    // holds
frame positions in file (for non-constant frame sizes, should hold
long[] with pos and size)
        } else if (stack==null)
                stack = new ImageStack(dwWidth, biHeight,
lastFrameToRead);
```

Code 4: Modified ImageStack constructor of ImageJ

```
/** Creates a new, empty image stack with a capacity of 'size'.  All
         'size' slices and labels of this image stack are initially
null.*/
    public ImageStack(int width, int height, int size) {
        this.width = width;
        this.height = height;
        this.cm = null;
        stack = new Object[size];
        label = new String[size];
        nSlices = 0;
    }
```

Code 5: The first part of the modified apply method of the LutApplier class of ImageJ

```
void apply(ImagePlus imp, ImageProcessor ip) {
    if (ip.getMinThreshold()!=ImageProcessor.NO_THRESHOLD) {
        imp.unlock();
        IJ.runPlugIn("ij.plugin.Thresholder", "skip");
        return;
    }
    min = (int)ip.getMin();
    max = (int)ip.getMax();
    if (min==0 && max==255) {
        if(!IJ.isMacro())
        {
            IJ.error("Apply LUT", "The display range must first be
updated\n"
                    +"using Image>Adjust>Brightness/Contrast\n"
                    +"or threshold levels defined using\n"
                    +"Image>Adjust>Threshold.");
        }

        // For a macro we don't care, there is nothing to do, we just
want to continue
        return;
    }
```

2.6.3 The Perl script

The Perl script TrackProcessor.pl (Code 12) processes the output from the ImageJ macro. The script depends on MTrack.pm (Code 13).

MTrack.pm parses the output of mTrack2 or mTrack3, dependent on a parameter. MTrack.pm is derived from the mTrack2 parser of a previous script (Conzelmann et al., 2011), which I rewrote to modularize and tidy the code so that it can be better maintained. I added an mTrack3 parser and moved it to MTrack.pm as a library file so that the same code can be reused by different files instead of being copied to different places.

The other code stayed in the main file TrackProcessor.pl, which has twelve input parameters (Table 13). The first parameter is the input file, which was generated by one of the ImageJ macros and contains the coordinates of the tracks. The input file is accompanied by a second file that has the base name of the input file appended by "_vertical.text_image.txt", which is the distribution file in text-image format. The second parameter is the frame rate with that the analyzed video was recorded. The third and seventh parameter give the width of the field of view in pixels or mm, respectively. The frame rate and the width of the field of view are used to normalize the displacements of the larvae to the unit mm/s. The fourth and fifth parameter give the number of pixels that should be removed in the video at the top and bottom, respectively. This allows removing pixels at the top and bottom of a vertical column where mirror images of larvae were tracked. The sixth parameter gives the version of mTrack that was used. The value 2 indicates mTrack2, any other value indicates mTrack3.

The twelfth parameter determines how the tracks are plotted in the track images. Two examples are shown in Figure 5. The examples were created with a value of 2, for values of 0 and 1; more information is plotted. The eighth parameter determines whether on the track images, the tracks pointing up and down (value 0) or left and right tracks (value 1) are color-coded differently. The ninth parameter determines whether for each frame an image is created that shows the positions of the larvae with color-coded dots. These images can be used to create movies. If the value is 0, nothing is done. If the value is 1, it encodes the tracks by time. If it is 2, it encodes them by moving angel. And if it is greater than 2, it encodes the tracks by time and angle. The size of the dots is determined in pixels by the eleventh parameter. The tenth parameter determines the background color of the images; if it is 0, the background is white, if it is 1, the background is black; if it is bigger than 1, no image is saved.

Table 13: Input parameters of TrackProcessor.pl

#	Description	Example
1	File to process (input file)	/File/Path/Example.res
2	Video frame rate (fps)	15
3	Field of view width in pixels	142
4	Number of pixels above column	5
5	Number of pixels at bottom	5
6	Version of mTrack	Default: 2
7	Column width in mm	Default: 31
8	Display from left to right	Default: 0
9	Print frames	Default: 0
10	Frames background	Default: 0
11	Particle size	Default: 10
12	Track image style	Default: 0

The Perl script TrackProcessor.pl saves three files into the directory of the input file: The first shows the distribution of the larvae along a vertical axis. The vertical axis can be changed to a horizontal axis if the eighth input parameter (Display from left to right) is 1. The second file shows the tracks of the larvae in the field of view and the tracks put to a common origin. The third is a text file, which contains the results for the calculations and is called Results.txt. Results.txt is created if it does not exist with a header row and a data row. Otherwise, another data row is added. This way, the results of many input files can be compiled into one output file.

Results.txt contains many columns, because I tried many ways to process the data. Eventually, I settled on the values highlighted in cyan in Table 14. These are "#Average x Displacement" for the horizontal beaker experiments and the "#Average y Displacement" for the vertical experiments. These are like "#Average x Movement" and "#Average y Movement", which are based on the tracks as units instead on single track-pieces. A track-piece is the part of a track between consecutive frames. Whole tracks may not follow a larva completely, because mTrack2 may not track a larva continuously. Therefore, two tracks may represent one larva. Each larva however is represented by many track-pieces and therefore if some of the path of a larva has not been tracked then the mistake is smaller than adding a larva twice to the average. This is theoretically better, however practically the values for "#Average y Displacement" and "#Average y movement" only differ slightly. Additional useful values (even so not presented

here) are “#Larvae” and “#Speed (mm per sec)” for checking the number of larvae and their speed in the experiment.

Results.txt is a tab-delimited list that can be pasted into an Excel file that contains formatting and graphs so that the data can be reviewed quickly. The data can be copied to an average sheet manually, or it can be copied automatically, via another Perl script.

Table 14: Column fields in the output file Results.txt

Column	Description
File Name	Name and full path of the input file
#Vectors	Number of vectors derived from mTrack traces
#upward Vectors	Number of vectors pointing up
#downward Vectors	Number of vectors pointing down
#leftward Vectors	Number of vectors pointing left
#rightward Vectors	Number of vectors pointing right
%upward Vectors	Percentage of vectors pointing up
%downward Vectors	Percentage of vectors pointing down
%leftward Vectors	Percentage of vectors pointing left
%rightward Vectors	Percentage of vectors pointing right
#Average x Displacement	Average displacement of the larvae along the x-axis in mm/s (based on track pieces)
#Average y Displacement	Average displacement of the larvae along the y-axis in mm/s (based on track pieces)
#Average x positive Displacement	Average displacement of the rightward swimming larvae along the x-axis in mm/s (based on track pieces)
#Average y positive Displacement	Average displacement of the upward swimming larvae along the y-axis in mm/s (based on track pieces)
#Average x negative Displacement	Average displacement of the leftward swimming larvae along the x-axis in mm/s (based on track pieces)
#Average y negative Displacement	Average displacement of the downward swimming larvae along the y-axis in mm/s (based on track pieces)

Column	Description
#Average x absolute Displacement	Average displacement of the larvae along the x-axis in mm/s with ignoring the sign (based on track pieces)
#Average y absolute Displacement	Average displacement of the larvae along the y-axis in mm/s with ignoring the sign (based on track pieces)
#Average x Movement	Average displacement of the larvae along the x-axis in mm/s (based on whole tracks)
#Average y Movement	Average displacement of the larvae along the y-axis in mm/s (based on whole tracks)
#Average positive y Movement	Average displacement of the upward swimming larvae along the y-axis in mm/s (based on whole tracks)
#Average negative y Movement	Average displacement of the downward swimming larvae along the y-axis in mm/s (based on whole tracks)
#Average absolute y Movement	Average absolute displacement of the larvae along the y-axis in mm/s (based on whole tracks)
#Average Movement	Average displacement of the larvae along the x- and y-axis in mm/s (based on whole tracks)
#Average positive Movement	Average displacement of the upward swimming larvae along the x- and y-axis in mm/s (based on whole tracks)
#Average negative Movement	Average displacement of the downward swimming larvae along the x- and y-axis in mm/s (based on whole tracks)
#Average absolute Movement	Average absolute displacement of the larvae along the x- and y-axis in mm/s (based on whole tracks)
#Larvae	Maximum number of larvae that could be tracked at the same time
#Larvae Upper	Maximum number of larvae that could be tracked at the same time in the upper half of the column
#Larvae Lower	Maximum number of larvae that could be tracked at the same time in the lower half of the column
#Larvae % Upper	Percentage of larvae in the upper half of the column

Column	Description
#Larvae % Lower	Percentage of larvae in the lower half of the column
#Upward Speed (mm per sec)	Average speed of larvae swimming upwards in mm/s (based on whole tracks)
#Downward Speed (mm per sec)	Average speed of larvae swimming downwards in mm/s (based on whole tracks)
#Absolute Speed (mm per sec)	Average speed of all larvae in mm/s (based on whole tracks)
#Single Sum Speed (mm per sec)	Average speed of all larvae, downward swimming larvae weighted negatively, in mm/s (based on whole tracks)
#Sum Speed (mm per sec)	Upward Speed – Downward Speed (based on whole tracks)
#Speed (mm per sec)	Average speed of all larvae in mm/s (based on track pieces)
#Mean depth (mm from surface)	Average depth of all the larvae from the surface
#Mean depth (in %)	Average depth distribution of all the larvae from the surface in percentage, 0 % is surface and 100 % is bottom
#Mean left/right (mm from middle)	Average deviation of all larvae from the midline
#Mean left/right (in % from middle)	Average deviation of all larvae from the midline in percentage, 0 % is midline, -50 % is left, 50 % is right
#Simple X straightness	How parallel the larvae swim to the x-axis, the closer the value is to one the more parallel are the larvae swimming (based on whole tracks)
#Simple Y straightness	How parallel the larvae swim to the y-axis, the closer the value is to one the more parallel are the larvae swimming (based on whole tracks)
#Single X straightness	How parallel the larvae swim to the x-axis, the closer the value is to one the more parallel are the larvae swimming (based on track pieces)

Column	Description
#Single Y straightness	How parallel the larvae swim to the y-axis, the closer the value is to one the more parallel are the larvae swimming (based on track pieces)
#Average Angel	The average angle to the x-axis of all swimming, larvae based on #Average x Displacement and #Average y Displacement (based on whole tracks)
#Top Vectors	The number of vectors that point up and are parallel to the y-axis or less than 15° off (based on whole tracks).
#Bottom Vectors	The number of vectors that point down and are parallel to the y-axis or less than 15° off (based on whole tracks).
#Left Vectors	The number of vectors that point left and are parallel to the x-axis or less than 15° off (based on whole tracks).
#Right Vectors	The number of vectors that point right and are parallel to the x-axis or less than 15° off (based on whole tracks).
%Top Vectors	Percentage of vectors that point up and are parallel to the y-axis or less than 15° off. If the vectors are pointing randomly then 30/360 ≈ 8 % is expected (based on whole tracks).
%Bottom Vectors	Percentage of vectors that point down and are parallel to the y-axis or less than 15° off. If the vectors are pointing randomly then 30/360 ≈ 8 % is expected (based on whole tracks).
%Left Vectors	Percentage of vectors that point left and are parallel to the x-axis or less than 15° off. If the vectors are pointing randomly then 30/360 ≈ 8 % is expected (based on whole tracks).
%Right Vectors	Percentage of vectors that point right and are parallel to the x-axis or less than 15° off. If the vectors are pointing randomly then 30/360 ≈ 8 % is expected (based on whole tracks).
#Track Pieces	The number of all tracks in all frames.

Column	Description
#Upward track pieces	The number of all track pieces that point up (based on track pieces).
#Downward track pieces	The number of all track pieces that point down (based on whole tracks).
#Leftward track pieces	The number of all track pieces that point left (based on track pieces).
#Rightward track pieces	The number of all track pieces that point right (based on whole tracks).
#Top track pieces	The number of all track pieces that point up and are parallel to the y-axis or less than 15° off (based on whole tracks).
#Bottom track pieces	The number of all track pieces that point down and are parallel to the y-axis or less than 15° off (based on whole tracks).
#Left track pieces	The number of all track pieces that point left and are parallel to the y-axis or less than 15° off (based on whole tracks).
#Right track pieces	The number of all track pieces that point right and are parallel to the y-axis or less than 15° off (based on whole tracks).
%Upward track pieces	Percentage of all track pieces that point up (based on track pieces).
%Downward track pieces	Percentage of all track pieces that point down (based on whole tracks).
%Leftward track pieces	Percentage of all track pieces that point left (based on track pieces).
%Rightward track pieces	Percentage of all track pieces that point right (based on whole tracks).
%Top track pieces	Percentage of all track pieces that point up and are parallel to the y-axis or less than 15° off (based on whole tracks). If the track pieces are pointing randomly then $30/360 \approx 8$ % is expected (based on track pieces).

Column	Description
%Bottom track pieces	Percentage of all track pieces that point down and are parallel to the y-axis or less than 15° off (based on whole tracks). If the track pieces are pointing randomly then 30/360 ≈ 8 % is expected (based on track pieces).
%Left track pieces	Percentage of all track pieces that point left and are parallel to the y-axis or less than 15° off (based on whole tracks). If the track pieces are pointing randomly then 30/360 ≈ 8 % is expected (based on track pieces).
%Right track pieces	Percentage of all track pieces that point right and are parallel to the y-axis or less than 15° off. If the track pieces are pointing randomly then 30/360 ≈ 8 % is expected (based on track pieces).
#Median X Displacement	The median displacement of the larvae along the x-axis (based on track pieces)
#Median Y Displacement	The median displacement of the larvae along the y-axis (based on track pieces)

2.6.4 Perl script mass calling

The Perl script TrackProcessor.pl can be called via the bash script TrackerCaller*.sh (Code 6). The bash script only needs to be placed into the parent directory of the ImageJ macro output files, which is in the example called Dest1* matching Dest1 or any other Dest1 appended by any characters. However, there should be only be one directory matching Dest1*, otherwise the bash script breaks. This way, the ImageJ macro output files can be all processed at once. The ImageJ macro can create easily hundreds of files, which should be processed with the same input parameters and should be written to the same Results.txt file. Results.txt is deleted if it already exists when the TrackerCaller*.sh is started. Ideally, a TrackerCaller*.txt is created for each experiment with its own parameters for TrackProcessor.pl and its own subdirectory.

Code 6: Example call of TrackProcessor.pl via TrackerCaller*.sh

```
#!/bin/bash

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do # resolve $SOURCE until the file is no
longer a symlink
```

```
  DIR="$( cd -P "$( dirname "$SOURCE" )" && pwd )"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE" # if $SOURCE was a
relative symlink, we need to resolve it relative to the path where
the symlink file was located
done
DIR="$( cd -P "$( dirname "$SOURCE" )" && pwd )"

SUBDIR=Dest1*

if [ -a $DIR/$SUBDIR/Results.txt ]
      then
       rm $DIR/$SUBDIR/Results.txt
fi

for i in $DIR/$SUBDIR/*.res;
do perl     /File/Path/TrackProcessor.pl  $i    15    135   19    12
      2;
done
```

The TrackerCaller*.sh files can be called from a SuperTrackerCaller.sh bash script. This allows reanalyzing the tracks, which is useful, if there has been found a bug in TrackProcessor.pl or another way to analyze the tracks, has been added to TrackProcessor.pl. SuperTrackerCaller.sh requires that all the TrackerCaller*.sh are in subdirectories of SuperTrackerCaller.sh.

Code 7: Calling TrackerCaller*.sh via SuperTrackerCaller.sh

```
#!/bin/bash

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do # resolve $SOURCE until the file is no
longer a symlink
  DIR="$( cd -P "$( dirname "$SOURCE" )" && pwd )"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE" # if $SOURCE was a
relative symlink, we need to resolve it relative to the path where
the symlink file was located
done
DIR="$( cd -P "$( dirname "$SOURCE" )" && pwd )"

for i in $DIR/*;
do
      if [ -d $i ]
      then
       for j in $i/TrackerCaller*;
       do
             /bin/bash $j
       done
      fi

done
```

Since, extracting manually, the data from the Results.txt files, is laborious, I automatized it with the Perl script CSV_ColumnExtractor.pl (Code 14).

CSV_ColumnExtractor.pl takes three arguments, the input file that contains the data to be extracted, the output file that receives the data, and a key to access the data column to be extracted. CSV_ColumnExtractor.pl can also be called via a bash script, for instance CVSExtractorSpeed.sh (Code 8), which is an example from the UV-side column experiments (Figure 14).

To make CSV_ColumnExtractor.pl and CVSExtractorSpeed.sh work optimally, I organized the file structure of the experiments in a certain way. The experiments were put into directories that contained the date of the experiments in yyyy-mm-dd format. This way the directories sort by date if they are sorted by name.

The date directories were filled with subdirectories, which were called for instance Dest0_3d_GGxGG_0_UV-Side. Here the first 0 was the number of the video file and the second 0 was the number of experiment of that day, which in that example are identical. Then 3d_GGxGG is the type of batch that was used in the experiment; here it was a 3-day-old batch, derived from two wild type parents. The upper-case G represent a wild type allele of *Go-opsin1*, while the lower-case g represents a Go-opsin$^{\Delta 8}$ knockout allele. The UV-Side is the experiment type; here it was just UV-light from the side followed by cyan light from the top. I also tried other conditions with different drugs, light intensities, or with my Go-opsin1$^{\Delta 8/\Delta 8}$ knockout mutants. However, those conditions were only exploratory and thus are not shown here. The subdirectories must be strictly named in this way so that the scripts work. Such a strict naming also documents the data analysis better.

CVSExtractorSpeed.sh extracts the data for the "#Speed (mm per sec)" columns of Results.txt, with only changing two lines it can extract any other data field of Results.txt. Once CVSExtractorSpeed.sh generated the output files, their contents were pasted into an Excel template to extract the values for instance in an interval from 3.5 to 4 min after mixing or from 1.5 to 2 min after stimulus onset. These values were then pasted into Prism (GraphPad Software) for statistical analysis and generating figures.

Code 8: Extract data from the Speed field via CVSExtractorSpeed.sh

```
#!/bin/bash

#Encoding of this file must be UTF8, otherwise does not find file
names with µ's

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do # resolve $SOURCE until the file is no
longer a symlink
  DIR="$( cd -P "$( dirname "$SOURCE" )" && pwd )"
  SOURCE="$(readlink "$SOURCE")"
```

```
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE" # if $SOURCE was a
relative symlink, we need to resolve it relative to the path where
the symlink file was located
done
DIR="$( cd -P "$( dirname "$SOURCE" )" && pwd )"

KEY="#Speed (mm per sec)"
KEYNAME=Speed_
TYPE[0]=3d_ggxgg_
TYPE[1]=3d_GGxGG_
TYPE[2]=2d_GGxGG_
TYPE[3]=1d_GGxGG_
TYPE[4]=1.5d_GGxGG_

EXPERIMENT[0]=UV-Side
EXPERIMENT[1]=UV-Side_5µM-RGWamide-0min
EXPERIMENT[2]=UV-Side_5µM-RGWamide-30min
EXPERIMENT[3]=UV-Side_5µM-L-cis-Diltiazem-0min
EXPERIMENT[4]=UV-Side_5µM-L-cis-Diltiazem-30min
EXPERIMENT[5]=UV-Side_20µM-L-cis-Diltiazem-0min
EXPERIMENT[6]=UV-Side_20µM-L-cis-Diltiazem-30min
EXPERIMENT[7]=UV-Side_100µM-L-cis-Diltiazem-0min
EXPERIMENT[8]=UV-Side_10µM-Mecamylamide-0min
EXPERIMENT[9]=UV-Side_10µM-MLD-peptide-0min
EXPERIMENT[10]=UV-Side_10µM-MLD-peptide-30min
EXPERIMENT[11]=UV-Side_10µM-MLD-peptide-90min
EXPERIMENT[12]=UV-Side_5µM-Serotonin-0min
EXPERIMENT[13]=UV-Side_5µM-Serotonin-30min
EXPERIMENT[14]=UV-Side_Slide1
EXPERIMENT[15]=UV-Side_Slide2
EXPERIMENT[16]=UV-Side_Slide3
EXPERIMENT[17]=UV-Side_Slide4
EXPERIMENT[18]=UV-Side_Slide5
EXPERIMENT[19]=UV-Side_Slide6
EXPERIMENT[20]=UV-Side_Slide7
EXPERIMENT[21]=UV-Side_Slide8
EXPERIMENT[22]=UV-Side_Slide3_5µM-L-cis-Diltiatem-0min
EXPERIMENT[23]=UV-Side_Slide3_5µM-L-cis-Diltiatem-30min
EXPERIMENT[24]=UV-LongDark


DESTINATION=Dest
EXTENSION=.txt
OUTPUT=Output

DATE=$(date +"%Y-%m-%d_")

for e in ${EXPERIMENT[@]};
do
      for t in ${TYPE[@]};
      do
       for i in $DIR/*;
       do
             if [ -d $i ]
             then
                   for j in $i/$DESTINATION*_$t*$e;
                   do
                         if [ -e $j ]
                         then
                               perl /File/Path/CSV_ColumnExtractor.pl
$j/Results.txt $DIR/$OUTPUT/$DATE$KEYNAME$t$e$EXTENSION "$KEY"
```

```
                        fi
                    done
                fi
            done
        done
done
```

2.6.5 Repairing corrupted avi-files

IC Capture may crash sometimes at the end of recording session so that it does not finish saving the avi-files. Such avi-files cannot be read by ImageJ, but they still contain the data IC Capture was writing to the file while it was recording. These corrupted avi-files only miss a header and some other details, which can be copied from a non-corrupted file of the same or slightly smaller size with a hex-editor like Hex Fiend.

With the hex-editor, the start of the file needs to be copied. What exactly needs to be copied is in the corrupted file zero, usually the first 33776 or 66048 bytes (of up to a 20 gigabyte file) depending when IC Capture crashed. Additionally, bigger files have additional blocks to be copied. These blocks start with the bytes D001 (hexadecimal notation). They are followed by additional zeros. The blocks may differ by the number of zeros. The longest blocks need attention; they are interrupted by non-zero bytes in the non-corrupted file. The non-zero bytes must be copied into the corresponding position in the corrupted file to fix it.

3 Results

3.1 The larvae swam down to UV and up to green light

The adult eyes of 3-day-old *Platynereis dumerilii* larvae mediate phototaxis (Randel et al., 2014) and express *Go-opsin1* (Figure 2). Therefore, I wondered how *Go-opsin1* contributed to phototaxis. An opsin only covers a part of the spectrum (Lamb, 1995). Thus, I checked how the larvae responded to different wavelengths, so that I could hypothesize how Go-opsin1 contributes to phototaxis. Phototaxis in *Platynereis dumerilii* larvae was studied earlier (Jékely et al., 2008; Randel et al., 2014), but the larvae were exposed to light from the side. In the sea, light may come from the side, if for instance, one side is shaded by rocks or sea grass, but without obstacles, direct light comes from the top. Therefore, I placed collimated monochromatic stimulus light at the top of the column setup of Conzelmann et al. (2011). In that column, 3-day-old larvae were stimulated for 4.5 min with either UV-light (400 nm) or green light (520 nm).

The stimulus period was binned into 30 s intervals and the last interval was analyzed.

When the larvae were stimulated with UV-light, the larvae were swimming down (Figure 5A). And when they were stimulated with green light, they were swimming up (Figure 5B).

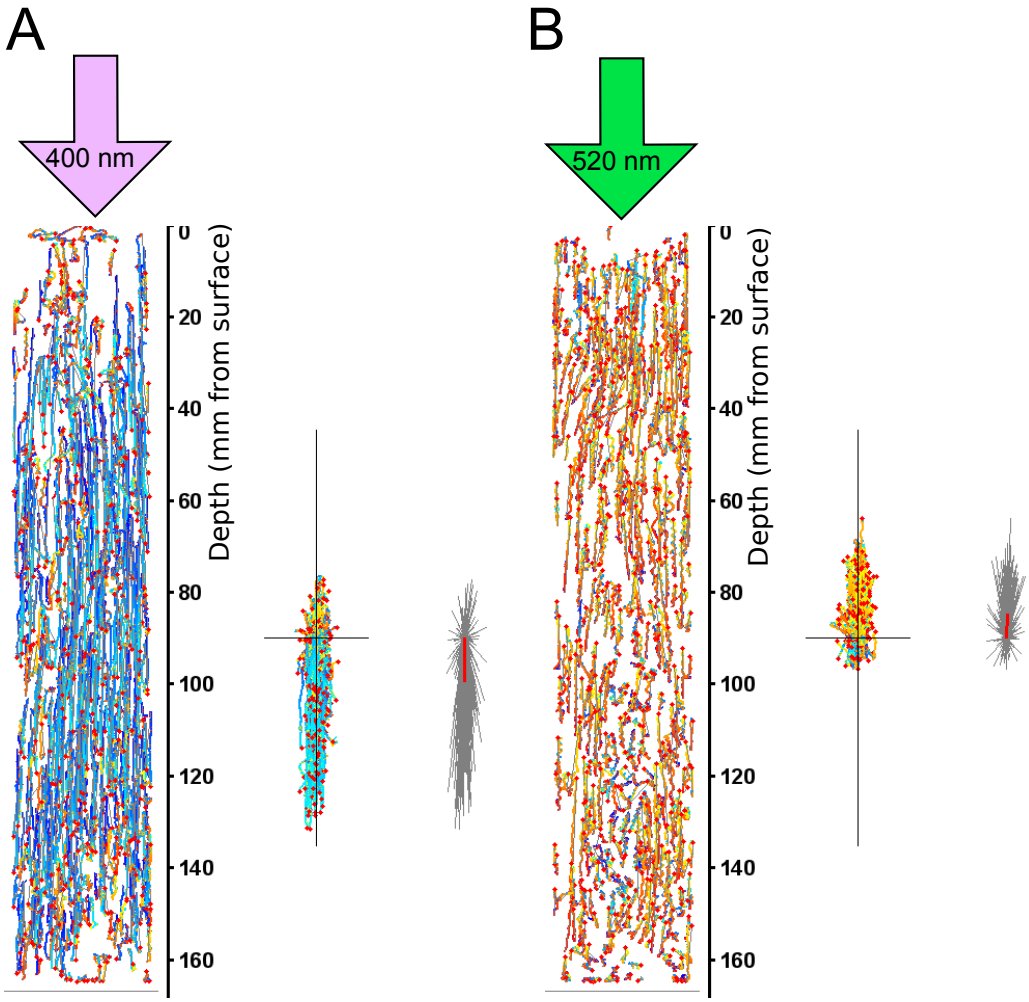


Figure 5: *Platynereis dumerilii* larvae swam down or up depending on the wavelength

A: Left: *Platynereis dumerilii* larvae swimming in a vertical column. The larvae were stimulated with UV-light (400 nm) from the top for 4.5 min. The larvae were tracked for the last 30 s with mTrack2. And the tracks were encoded for down-swimming larvae from blue to cyan, and for up-swimming larvae from red to yellow. Blue and red were used for the early frames that were analyzed, and cyan and yellow were used for the late frames. The ends of the tracks are marked with red dots. Middle: The same tracks as on the left, but moved to a common origin to clarify the main movement direction of the larvae. Right: The tracks from the middle, which were converted to vectors, by connecting their start and endpoints. The vector in red is the average vector of all the grey vectors.

B: Same as in A, except that the larvae were stimulated with green light (520 nm).

A and B: More than 300 larvae were tracked.

3.2 The larvae switched repeatedly the direction with the wavelength

Since 3-day-old *Platynereis dumerilii* larvae swam down with UV-light (400 nm) and up with green light (520 nm), I wondered how robust this behavior was and whether the larvae would adapt.

Therefore, I stimulated the larvae six times in a row with UV (380 nm) and green (520 nm) light. Before the larvae were stimulated, they were mixed in the column and left in the dark, so that they calmed down and were randomly distributed. The last stimulus was followed by a dark period. Each stimulus and dark period was 4.5 min long. Each period was binned into intervals of 30 s and the larvae were tracked for each bin with mTrack2. The tracks where averaged as shown in Figure 5.

When the larvae had calmed down, they neither swam up nor down, but they swam down with UV-light (380 nm) and up with green light (520 nm). Interestingly after the last green stimulus, the larvae were still swimming up for a while in the dark (Figure 6A). So, this behavior is robust and does not adapt.

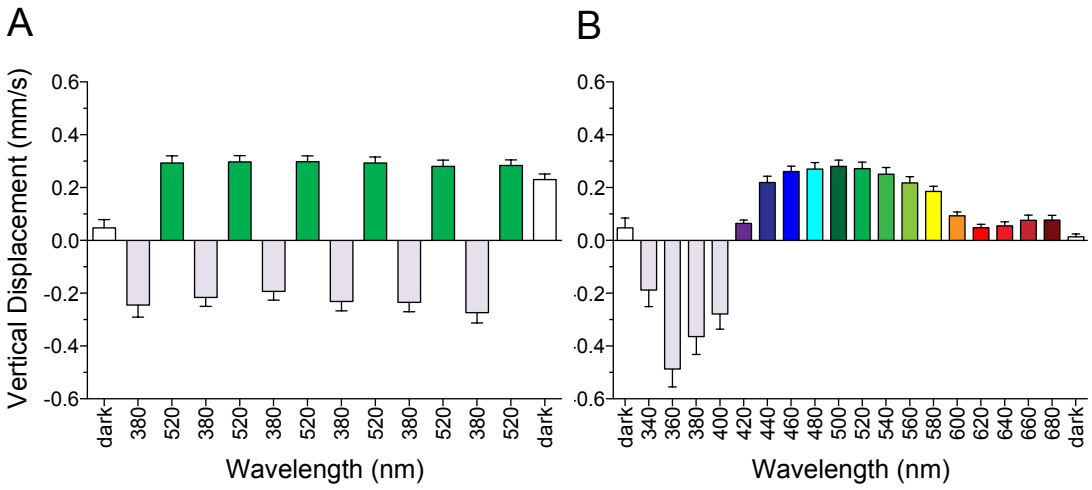


Figure 6: *Platynereis dumerilii* larvae swim down to UV-light and up to visual light

3-day-old *Platynereis dumerilii* larvae were stimulated with different wavelengths of light. They were tracked and their average displacement was calculated during the analysis period. Positive displacement indicates up-swimming and negative displacement indicates down-swimming. The wavelengths are color coded: UV wavelengths are colored grey, and the other wavelengths are colored as an average human observer would approximately perceive them.

All error bars are SEM. n = 8.

A: The larvae were alternatingly stimulated with light of 380 nm and 520 nm. The stimulus period was 4.5 min, and was binned into 30 s intervals. The last six intervals were averaged.

B: The larvae were stimulated with wavelength from 340 nm to 680 nm. From 340 nm to 400 nm, green light (520 nm) was used before to pull up the larvae. From 420 nm to 680 nm, UV-light (400 nm) was used before to push down the larvae. Each stimulus period was 3.5 min, and was binned into 30 s intervals. The last four intervals were averaged. The data have been partially published (Gühmann et al., 2015).

3.3 The larvae switched swimming direction at 420 nm

Since 3-day-old *Platynereis dumerilii* larvae swam down repeatedly with UV-light (380 nm), and up with green light (520 nm), I wondered what was the switching wavelength.

Therefore, I stimulated the larvae with different wavelengths in 20 nm steps from 340 nm to 680 nm. After each stimulus, the larvae were not randomly distributed, but depending on the wavelength at the bottom or at the top. The larvae did not redistribute even after a while. Larvae that are at the top or at the bottom do not swim up or down, respectively. Even so, they would if they could. Therefore, I had to redistribute them; however, mixing might have introduced artifacts due to mechanical stimulation.

Therefore, I split the spectrum into two ranges: From 340 nm to 480 nm, and from 400 nm to 680 nm. The two ranges overlapped, so that the switching point could be identified in both ranges. In the range from 340 nm to 480 nm, the larvae were pulled up after green light (520 nm). And from 400 to 680 nm, they were pushed down after UV-light (400 nm). Each stimulus was 3.5 min long and was binned into intervals of 30 s.

The larvae swam down when they were exposed to light between 340 and 400 nm, they switched at 420 nm, and swam up from 440 nm to 600 nm (Figure 6B).

This shows that 3-day-old *Platynereis dumerilii* larvae possess a mechanism that makes them swimming up or down depending on the wavelength. This mechanism may be a cellular chromatic antagonism.

3.4 Generating a *Platynereis dumerilii Go-opsin1* knockout line

Since the larvae were swimming up or down wavelength-specifically, I wondered how this switch worked. I hypothesized that up-swimming was positive phototaxis and down-swimming was negative phototaxis. Since phototaxis in 3-day-old larvae is mediated by the adult eyes (Randel et al., 2014), I hypothesized that the adult eyes mediated the switch by a cellular chromatic antagonism (Nilsson, 2009). A cellular chromatic antagonism could be formed by Go-opsin1 and the rhabdomeric opsins. Go-opsin1 and the rhabdomeric opsins belong to different classes of opsins. A scallop Go-opsin is thought to be hyperpolarizing (Kojima et al., 1997) and rhabdomeric opsins are depolarizing. Therefore, these opsins could antagonize each other in the same cell, like the lizard parietopsin and pinopsin (Su et al., 2006).

To test this, I tried first to knockdown *Go-opsin1* with three different morpholinos. The morpholinos targeted the start site, and two different splice sites of *Go-opsin1*. The morpholinos were injected into fertilized eggs, and three days later when the eggs had become larvae, the larvae were tested for phototaxis defects. The splice site morpholinos caused mRNA missplicing (data not shown) and so worked, but only few larvae survived the injections and those that survive seemed to be sick.

Therefore, I gave up on the morpholinos, and generated a *Platynereis dumerilii Go-opsin1* knockout line with engineered zinc-finger-nucleases (ZFNs). The ZFNs should not target splice sites and single nucleotide polymorphisms; therefore, I screened *Go-opsin1* for them. The ZFNs were generated by Sigma-Aldrich and targeted the first *Go-opsin1* exon, which encodes the N-terminus before the first transmembrane domain (Figure 7A). The ZFNs, however, still targeted a single nucleotide polymorphism; even so, Sigma-Aldrich tried several alternative designs. Sigma-Aldrich characterized the ZFNs to be not "superior" but to have cleavage activity that is "useful for genome editing experiments".

The ZFNs were injected as mRNA into fertilized *Platynereis dumerilii* wild type eggs. The injected individuals were cultured and kept until they became mature worms. The sexual mature worms were crossed to wild type worms, and the progeny was screened for inherited genetic lesions via PCR and subsequent sequencing.

For PCR, I tried primers that targeted pieces of genomic DNA longer than 600 bp. The primers would have detected longer deletions; however, they did not amplify DNA from every DNA sample I tested.

Therefore, I sequenced the PCR products I could amplify, and found upstream of the start codon many single nucleotide polymorphisms that could have prevented some of the primers to bind and amplify the affected alleles. Another allele had a 300 bp deletion removing the binding sites of a subset of my primers (data not shown). This does not seem to be uncommon; intron 2 of the *vtn* locus has a 2.5 kb length polymorphism (Bannister et al., 2014). In contrast, exon 1 of *Go-opsin1* was more conserved.

Therefore, I used PCR primers covering 258 bp of exon 1. I used a nested sequencing primer that was placed right before the *Go-opsin1* start codon (Figure 7A) to avoid sequencing of unspecific PCR product. These primers amplified DNA from all the samples I tested.

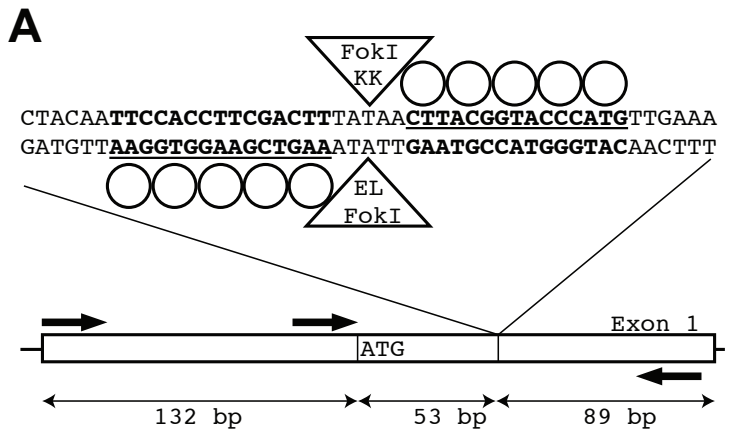
After injection, I could culture around 500 larvae that survived the first days. However, only a few larvae survived and became adults. These adults produced

51 batches. This is an approximate survival rate of 10 %, which agrees with earlier reports (Backfisch et al., 2014; Hauenschild and Fischer, 1969; Tosches, 2013).

Table 15: Ratio of mutant and wild type *Go-opsin1* genotypes in the F1 and F2 generation

Genotype	F1 obs	F1 exp	F2 obs1	F2 obs2	F2 exp
Go-opsin1$^{wt/wt}$	56	48	20	25	24
Go-opsin1$^{wt/\Delta 8}$	40	48	55	55	48
Go-opsin1$^{\Delta 8/\Delta 8}$	-	-	21	16	24

For establishing a homozygous *Go-opsin1* knockout line, F1 and F2 worms were genotyped before crossing, so that they could be selected by genotype. The F1 population was sampled once, and the F2 population was sampled twice, because more homozygous worms were needed for crossing, than those originally identified. Each sample had a size of 96 worms. The observed genotype frequencies did not diverge from the expected frequencies, according to a Chi-square test with an alpha of 0.05.



B

```
       wt CTACAATTCCACCTTCGACTTTATAACTTACGGTACCCATG
Go-ops-Δ8 CTACAATTCCACCTTCGACTT--------ACGGTACCCATG

       wt MEFNHTTEDSYNSTFDFITYGTHVEIYKRPDIQPRVY…
Go-ops-Δ8 MEFNHTTEDSYNSTFDLRYPC-NLQET-YPAPGLYGH…
```

Figure 7: *Go-opsin1* ZFN design and *Go-opsin1* mutation

A: Genomic region of *Go-opsin1* targeted by the ZFNs: Circles indicate the base pairs recognized by the individual zinc-fingers. The cleavage site of the FokI nuclease is indicated by triangles. The arrows indicate the sites of the primers I used: The PCR primers, at the start and the end of exon 1; the sequencing primer, in the middle before the start codon. The double arrows give the sizes of the different pieces of exon 1. Note that the PCR product is a little bit shorter than exon 1, because the reverse primer is placed a few base pairs before the end of exon 1.

B: 8 bp deletion in the *Go-opsin1*$^{\Delta 8}$ mutant allele. The mutation leads to a frame-shift and a premature stop-codon in *Go-opsin1*.

This figure has been published, before (Gühmann et al., 2015).

Among the 51 batches, I found one batch that carried an 8-base-pair deletion, leading to a frame-shift and a premature stop-codon in the *Go-opsin1* open reading frame (Figure 7B). Since the premature stop-codon is in the first of six exons, the *Go-opsin1* mRNA is subject to nonsense-mediated decay, and therefore this is a *Go-opsin1* knockout allele, which I call *Go-opsin1*$^{\Delta 8}$. I crossed heterozygous *Go-opsin1*$^{wt/\Delta 8}$ F1 carriers to generate a homozygous *Go-opsin1*$^{\Delta 8/\Delta 8}$ knockout line. The *Go-opsin1* genotypes were distributed in the F1 and F2 generations as expected (Table 15). This means the mutants were as viable as the wild type worms when they competed for space and food with each other under laboratory conditions. The mutants were also as fertile and showed a lunar reproductive cycle as wild type worms (Figure 8).

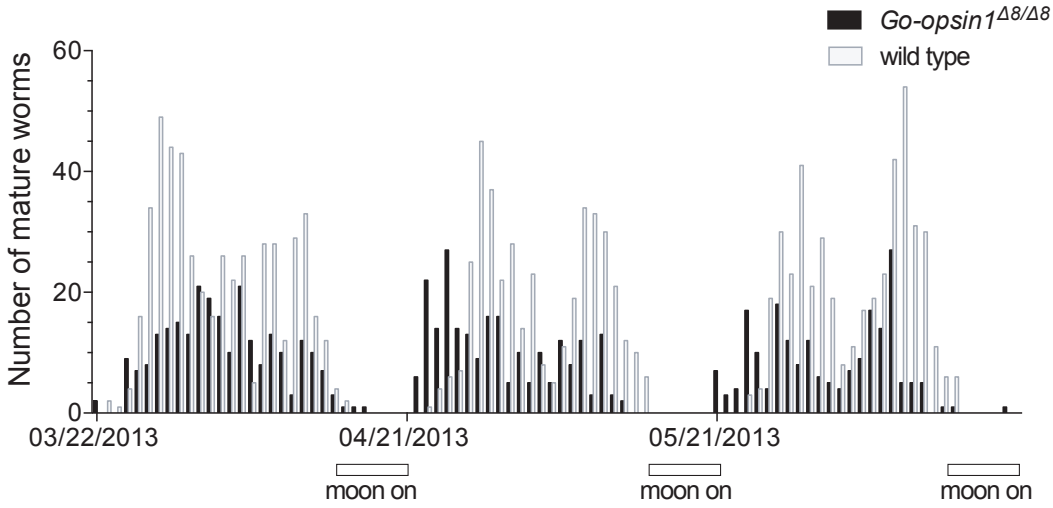


Figure 8: Lunar reproduction cycle of wild type and *Go-opsin1* knockout worms

Number of mature worms per day during a three-month period: During the full-moon phases, the nights were illuminated with 10 W light-bulb. This figure has been published, before (Gühmann et al., 2015).

3.5 *Go-opsin1* knockout larvae are less phototactic to cyan light

With the homozygous *Go-opsin1*$^{\Delta 8/\Delta 8}$ knockout larvae, I could test how Go-opsin1 contributed to phototaxis, by comparing phototaxis of mutant and wild type larvae. I used 3-day-old larvae that were kept at 22°C from fertilization to the experiment. These larvae correspond to 4.5-day-old larvae that were kept at 18°C (Fischer et al., 2010). I stimulated these larvae the same way as before (Figure 6A). In brief, the larvae were stimulated alternatingly with UV (380 nm) and green (520 nm) light from the top in the vertical column for 4.5 min for each stimulus.

In general, the homozygous *Go-opsin1*$^{\Delta 8/\Delta 8}$ knockout larvae behaved like wild type larvae. They swam down when they were stimulated with UV-light

(380 nm) and swam up with green light (520 nm). They only differed during the first two green light stimuli (520 nm) by how fast they were swimming up. The heterozygous *Go-opsin1*$^{wt/\Delta 8}$ larvae did not differ from the wild type larvae even so they behaved more like the homozygous larvae (Figure 9A).

Since the *Go-opsin1*$^{\Delta 8/\Delta 8}$ knockout larvae swam down to UV-light (380 nm) as the wild type larvae, the down-swimming is not mediated by Go-opsin1. However, Go-opsin1 may contribute to the up-swimming, as the minor defect at green light (520 nm) indicates. The defect may be just minor, because Go-opsin1 may not be maximally sensitive at 520 nm, but maximally sensitive to some wavelength close by. To check this, I measured the behavior of homozygous *Go-opsin1*$^{\Delta 8/\Delta 8}$ knockout larvae across the whole spectrum from 340 nm to 680 nm in 20 nm steps and compared them to heterozygous and wild type larvae (Figure 9B).

The homozygous knockout larvae were less swimming up in the blue-cyan-green part of the spectrum (440 nm – 520 nm), the heterozygous larvae, however, did not differ statistically from the homozygous mutant larvae or the wild type larvae, even so they behaved more like the wild type larvae. Therefore, Go-opsin1 contributes to the up-swimming.

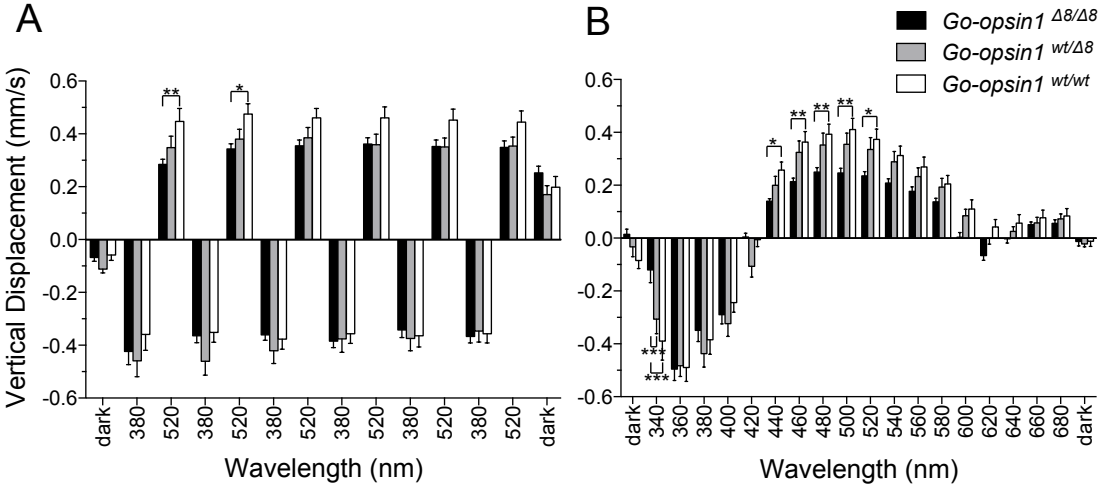


Figure 9: *Go-opsin1*$^{\Delta 8/\Delta 8}$ knockout larvae are less sensitive to blue-cyan-green light

Homozygous *Platynereis dumerilii Go-opsin1*$^{\Delta 8/\Delta 8}$ knockout larvae were compared with heterozygous *Go-opsin1*$^{wt/\Delta 8}$ mutant and wild type larvae. All larvae were three days old and kept at 22°C, before the experiment. The larvae were stimulated with different wavelength of light. They were tracked and their average displacement was calculated during the analysis period. Positive displacement indicates up-swimming and negative displacement indicates down-swimming. All error bars are SEM. Statistical significance was determined with a 2-way-ANOVA with Tukey's post hoc test. P* < 0.05, P** < 0.01, P*** < 0.001.

A: The larvae were alternatingly stimulated with light of 380 nm and 520 nm. The stimulus period was 4.5 min, and was binned into 30 s intervals. The last six intervals were averaged. *Go-opsin1*$^{\Delta 8/\Delta 8}$, n = 10; *Go-opsin1*$^{wt/\Delta 8}$, n = 13; *Go-opsin1*$^{wt/wt}$, n = 14

B: The larvae were stimulated with wavelength from 340 nm to 680 nm. From 340 nm to 400 nm, green light (520 nm) was used before to pull up the larvae. From 420 nm to 680 nm, UV-light (400 nm) was used

before to push down the larvae. Each stimulus period was 3.5 min, and was binned into 30 s intervals. The last four intervals were averaged. The data has been partially published (Gühmann et al., 2015). *Go-opsin1*$^{\Delta 8/\Delta 8}$, n = 13; *Go-opsin1*$^{wt/\Delta 8}$, n = 13; *Go-opsin1*$^{wt/wt}$, n = 14.

3.6 Go-opsin1 is a cyan opsin and c-opsin1 is a UV opsin

The *Go-opsin1*$^{\Delta 8/\Delta 8}$ knockout larvae showed that Go-opsin1 does not mediate the UV down-swimming, but instead mediates the up-swimming in the blue-cyan-green range of the spectrum (460 nm – 520 nm). To confirm this, I wanted to get an *in vitro* absorption spectrum of Go-opsin1. Additionally, I tried to get spectra of r-opsin1, r-opsin3, r-opsin4, peropsin1, and c-opsin1, because these spectra could tell which of the opsins may mediate the UV down-swimming.

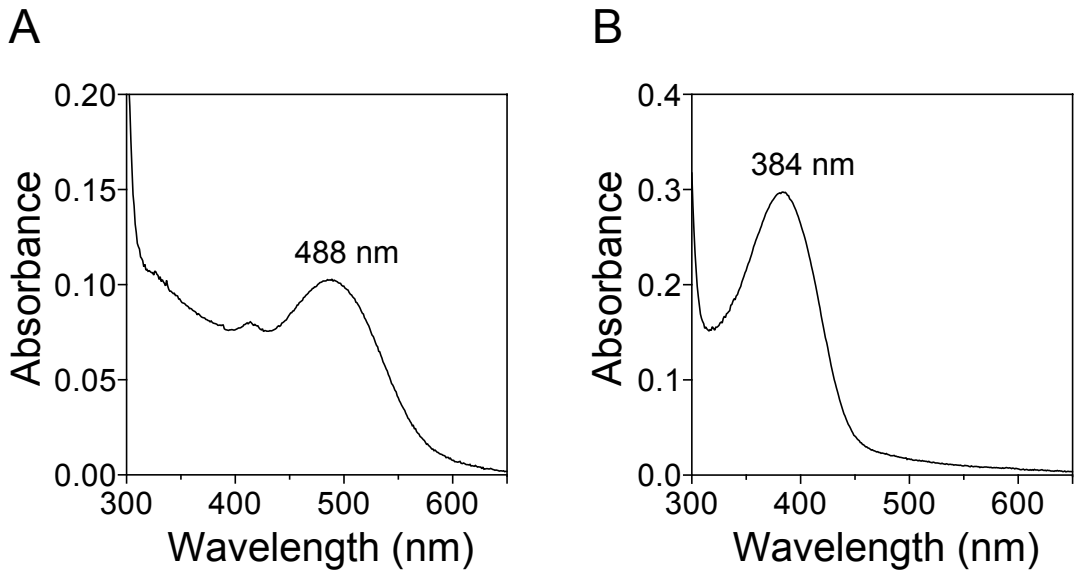


Figure 10: Absorption spectra of Go-opsin1 and c-opsin1

Absorption spectra of *Platynereis dumerilii* Go-opsin1 (A) and c-opsin1 (B) pigments in the dark: The pigments were reconstituted with 11-*cis*-retinal from opsins expressed in COS1 cells. The pigments were purified and their absorption was measured in a cuvette across the different wavelengths by Huiyong Jia and Shozo Yokoyama. The Go-opsin1 spectrum has been published, before (Gühmann et al., 2015).

First, I tried to record with Phillip Bauknecht and Sarah-Lena Offenburger the absorption spectra from purified photopigments. The idea was to express the opsins in HEK 293 cells, reconstitute them with 11-*cis*-retinal to functional pigments, and then purify them so that their absorption spectrum could be measured. For that, I cloned the opsins into a pcDNA3.1+ plasmid, except c-opsin1, which Phillip Bauknecht cloned. Then Phillip Bauknecht and Sarah-Lena Offenburger tried to purify functional photopigments, however unsuccessfully. In general, obtaining functional photopigments from cultured cells is difficult, especially invertebrate photopigments (Knox et al., 2003).

Therefore, we gave this task to experts: Huiyong Jia and Shozo Yokoyama. We sent them the cloned opsins. They expressed them in COS1 cells. However, they could not purify functional r-opsin1, r-opsin3, and r-opsin4 pigments, either. But they could purify functional Go-opsin1, c-opsin1, and peropsin1 pigments. The peropsin1 spectrum shall be published elsewhere.

The Go-opsin1 pigment absorbed in the blue-cyan-green range of the spectrum, with a λ_{max} of 488 nm (Figure 10A), and the absorption covered a range from approximately 440 nm to 560 nm, which matches the behavior defect of the *Go-opsin1*$^{\Delta 8/\Delta 8}$ knockout larvae.

The c-opsin1 pigment absorbed in the UV range of the spectrum, with a λ_{max} of 384 nm (Figure 10B). From the spectrum, c-opsin1 may mediate a photoresponse from 340 nm to 420 nm, which matches the spectral range of the UV down-swimming response.

3.7 UV-response and phototaxis can be separated

The c-opsin1 spectrum matched the UV down-swimming response. This indicates that the UV-response is mediated by c-opsin1, which is expressed in the ciliary photoreceptor cells. The ciliary photoreceptor cells are deep brain photoreceptor cells that are not shaded by any pigment. Therefore, the ciliary photoreceptor cells are non-directional photoreceptor cells and so the UV-response should be non-directional, too. This is already an idea, Nico K. Michiels pointed out to me when I was visiting his lab before I saw the c-opsin1 absorption spectrum. This non-directional UV-response is in contrast to phototaxis, which is a directional light response and is mediated by the pigment shaded adult eyes in 3-day-old *Platynereis dumerilii* larvae (Randel et al., 2014).

Thus, the UV-response and phototaxis are two different light responses and so should be experimentally distinguishable by changing the direction of the light. Irrespective of the light direction, UV-light should make the larvae swim down, while phototaxis should make the larvae swim to or away from the light. In a horizontal setup where the light comes from the side, any down-swimming component should disappear if the larvae are already at the bottom. Therefore, a horizontal setup should also allow me to study whether Go-opsin1 contributes to phototaxis in the UV range.

In a horizontal setup, I used directional, diffuse, collimated monochromatic light of different wavelengths to stimulate 3-day-old larvae swimming in a glass beaker. 3 to 4-day-old nectochaete larvae can show both positive and negative phototaxis and may switch during an experiment in a horizontal setup (Randel et

al., 2014). Commonly, initially photopositive larvae turned photonegative after light exposure, but some batches stayed photopositive during the whole experiment (data not shown). Therefore, I could consistently test negative phototaxis on several batches with many larvae (50 - 500 larvae). To quantify the turning-efficiency of phototaxis, the larvae were first illuminated for 30 s with white light from one side of the beaker to trigger negative phototaxis. Following this white light stimulus, I stimulated for 30 s with monochromatic light from the other side. After stimulus onset, the larvae turned 180 degrees and their displacement along the light vector was measured from 15 s to 30 s after stimulus onset.

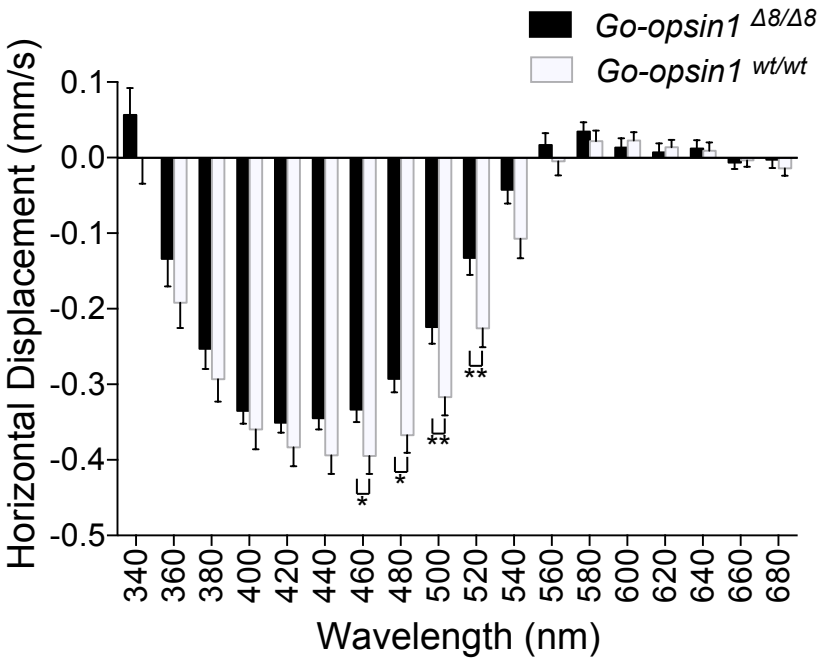


Figure 11: Reduced efficiency of phototaxis in *Go-opsin1$^{Δ8/Δ8}$* mutant larvae

Efficiency of negative phototaxis of wild type and *Go-opsin1$^{Δ8/Δ8}$* knockout larvae to different wavelengths of light: The larvae were swimming in a beaker and were illuminated from one side for 30 s. The larvae were tracked and their horizontal displacement was calculated during the analysis period. Negative values mean the larvae were swimming away from the light. n = 31 batches for wild-type and n = 25 batches for *Go-opsin1$^{Δ8/Δ8}$* mutant larvae. Each batch contained >50 larvae. All error bars are SEM. Statistical significance was determined with pairwise two tailed unpaired t-tests. P* < 0.05, P** < 0.01. The significances at 500 nm and 520 nm are left after Sidak-Bonferroni correction. The data have been partially published, before (Gühmann et al., 2015).

The wild type larvae were negatively phototactic between 360 nm - 540 nm, and showed the strongest response between 440 nm - 460 nm (Figure 11). This is a broad spectral response and indicates that multiple opsins are involved. The *Go-opsin1$^{Δ8/Δ8}$* mutant larvae were also negatively phototactic and were sensitive

across the whole spectrum, like the wild-type larvae. However, the mutant larvae showed the strongest response at 420 nm and their phototaxis was significantly reduced between 460 nm - 520 nm relative to wild type (Figure 11). The spectral range of reduced phototaxis-performance matches the *in vitro* absorption of the Go-opsin1 photopigment (Figure 10A). This confirms that Go-opsin1 contributes to phototaxis and enhances the efficiency of phototaxis in the blue-cyan-green (460 nm - 520 nm) range of the spectrum and shows that the UV-response is an independent behavior.

3.8 Larvae swim towards UV and green light from the bottom

When the light came from the side, the larvae either swam towards the light (data not shown) or swam away (Figure 11). They did not switch their direction depending on the wavelength. Therefore, the UV down-swimming response is unlikely phototaxis. However, it could still be negative phototaxis restricted to the vertical axis.

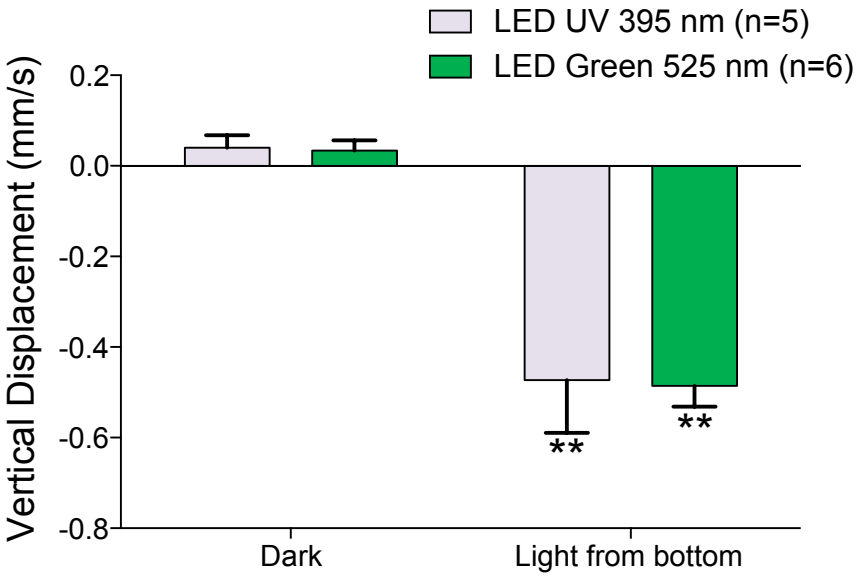


Figure 12: *Platynereis dumerilii* larvae swim down to UV and Green light from the bottom

Platynereis dumerilii larvae vertical swimming in response to UV (395 nm) or green (525 nm) light coming from LEDs at the bottom: The larvae were mixed and recorded for 10 min. After 5 min darkness, the larvae were exposed to the light for 5 min. For the dark control, the larvae's swimming was analyzed in an interval from 4.5 to 5 min after mixing. For the light stimuli, the larvae's swimming was analyzed in an interval from 1 to 1.5 min after stimulus onset. Negative values indicate down-swimming. Statistical differences are shown between dark control and the stimulus light. The statistical significance was determined with a 2-way-ANOVA and Holm-Sidak post hoc test. P** < 0.01. All error bars are SEM. For UV n = 5, for green n = 6.

Therefore, I tested how 3-day-old *Platynereis dumerilii* larvae reacted to light that came from the bottom instead from the top. For that, I put three UV

(395 nm) or three green (525 nm) LEDs under the column that contained the larvae. Before the experiment, I mixed the larvae to redistribute them and waited 5 min so that they could calm down. Then I switched on the LEDs and measured 1.5 min later, for 0.5 min the vertical displacement of the larvae.
Right before stimulus onset, the larvae had calmed down and did not move up or down anymore. 1.5 min after switching on the LEDs, the larvae swam down towards the light irrespective whether it was UV (395 nm) or green (525 nm) light (Figure 12).
Since the larvae did not switch their direction depending on the wavelength with light from the bottom, the UV down-swimming response cannot be negative phototaxis. The UV-response depends on the wavelength but not on the direction of the light.

3.9 Already 41-hour-old larvae show the UV-response

The UV-response is a non-directional response, and its spectrum fits to the spectrum of c-opsin1. Both observations favor the idea that the UV-response is mediated by the c-opsin1 expressing non-directional ciliary photoreceptor cells. The ciliary photoreceptor cells have already developed in 2-day-old larvae (Arendt et al., 2004) before the adult eyes are functional. This means that larvae younger than three days should also swim down when they are exposed to UV-light.
Therefore, I recorded 41 and 53-hour-old larvae in the vertical column with light from the top. The light alternated between 380 nm (UV) and 520 nm (green) and ranged from 340 nm to 680 nm in 20 nm steps: The same protocols as I used for the 3-day-old larvae kept at 18°C (Figure 6) and 22°C (Figure 9). The larvae kept at 22°C correspondent to 4.5-day-old larvae kept at 18°C. I compared those four sets of larvae (Figure 13).
Already, 41-hour-old larvae swam up to green (520 nm) light and swam down to UV (380 nm) light (Figure 13A). The larvae responded in all age groups to UV-light between 340 nm and 400 nm. However, the UV-response differed in strength between the age groups. It was strongest in the 3-day-old larvae kept at 18°C and 22°C (Figure 13B), weaker in the 41-hour-old larvae, but remarkably, still stronger than in the 53-hour-old larvae. This is remarkable, because the naïve expectation is that in such a developmental series, the response would follow a trend, an increasing trend.
Therefore, I looked at the variability of the UV-response with box plots, and indeed the 41-hour-old larvae are highly variable (Figure 13C), more variable

than the other groups of larvae (Figure 13C, D), so that by chance the average may be biased in one or another direction.

The 41-hour-old larvae were not so variable when they were swimming up (Figure 13C). The two youngest age groups did not differ. The older the larvae became the more sensitive they became to longer wavelength and they were swimming up faster to blue-cyan-green (460 nm - 540 nm) light (Figure 13B).

This shows that the UV-response is active whether phototaxis is mediated by the larval or the adult eyes, and that it originates before the adult eyes become active, so that it is a separate response system.

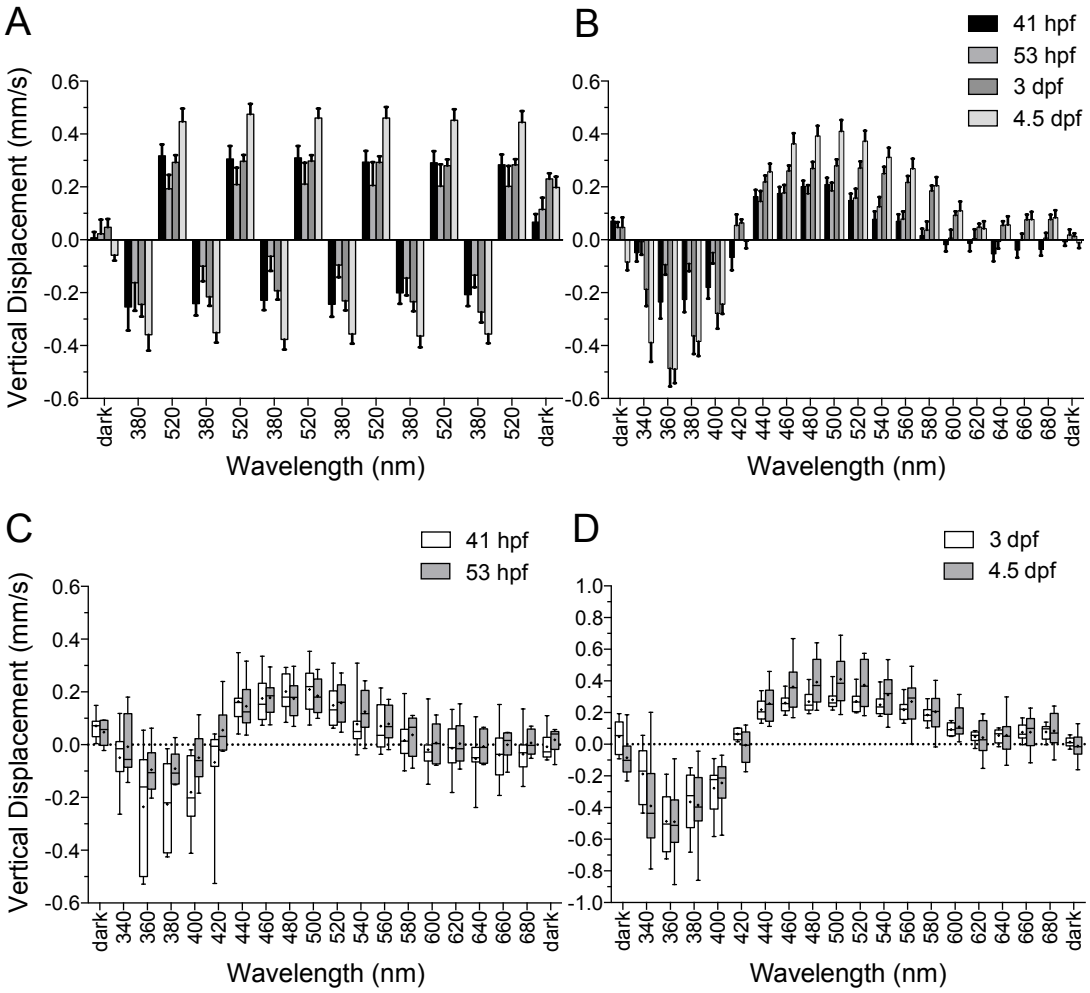


Figure 13: UV-response and phototaxis to light from top across different larval stages

41 and 53-hour-old, 3 and 4.5-day-old *Platynereis dumerilii* larvae were stimulated with different wavelength of light. They were tracked and their average displacement was calculated during the analysis period. Positive displacement indicates up-swimming and negative displacement indicates down-swimming. The 4.5-day-old larvae are in fact 3-day-old larvae kept at 22°C instead of 18°C. These larvae corresponded developmentally to 4.5-day-old larvae that were kept at 18°C (Fischer et al., 2010). These larvae are however called 4.5-day-old larvae for brevity. The data for the 3-day-old larvae and the 4.5-day-old larvae have been already displayed in Figure 6 and Figure 9. And are shown here for comparison. All error bars are SEM. Abbreviations: hpf: hours post fertilization, dpf: days post fertilization

A: The larvae were alternatingly stimulated with light of 380 nm and 520 nm. The stimulus period was 4.5 min, and was binned into 30 s intervals. The last six bins were averaged. The 4.5-day-old larvae swim faster up or down when they react to light than the younger larvae. The difference is also statistically significant for most pairs of measurement groups with p-values below 0.05 or 0.01 according to a 2-way-ANOVA with a Tukey's post hoc test. However, the test details are not shown so that the figure is not cluttered, but the test details are given in section 11.5. 41 hpf: n = 12; 53 hpf: n = 4; 3 dpf: n = 13; 4.5 dpf: n = 14.

B: The larvae were stimulated with wavelength from 340 nm to 680 nm. From 340 nm to 400 nm, green light (520 nm) was used before to pull up the larvae. From 420 nm to 680 nm, UV-light (400 nm) was used before to push down the larvae. Each stimulus period was 3.5 min, and was binned into 30 s intervals. The last four intervals were averaged. The younger larvae swim down to UV light (340 nm – 400 nm) more slowly than the older larvae. This difference is also statistically significant for most pairs of measurement groups with p-values below 0.05 or even below 0.0001 according to a 2-way-ANOVA with a Tukey's post hoc test. The 41 and 53 hour old larvae also differ for 360 nm and 380 nm with p-values below 0.05. The older larvae swim up faster to visible light (460 nm – 580 nm). This difference is also statistically significant for most of the pairs of measurement groups, with p-values below 0.05 or 0.01 according to a 2-way-ANOVA with a Tukey's post hoc test. However, the test details are not shown so that the figure is not cluttered, but the test details are given in section 11.6. 41 hpf: n = 11; 53 hpf: n = 6; 3 dpf: n = 12; 4.5 dpf: n = 14.

C: Box Plot for the data of the 41-hour-old and 53-hour-old larvae shown in B. The box plot highlights the variability in the UV range, that cannot be displayed with bar plots. However, only two data sets are given to avoid that the figure gets to crowded. The other two data sets are shown in D. The whiskers indicate the 5th and the 95th percentile; dots indicate the means, bars within the boxes indicate the sample median. 41 hpf: n = 11; 53 hpf: n = 6.

D: Box Plot for the data of the 3 and 4.5-day-old larvae shown in B. The whiskers indicate the 5th and the 95th percentile; dots indicate the means, bars within the boxes indicate the sample median. 3 dpf: n = 13; 4.5 dpf: n = 14.

3.10 From 36 hours on, larvae respond to diffuse UV-light

41-hour-old larvae already showed the UV-response. I wondered whether there were younger larvae without the UV-response. However, larvae without the UV-response cannot be measured this way, because I used the UV-response to pull them down.

Therefore, I stimulated the larvae with diffuse non-directional UV-light (395 nm) from the side. The light was emitted by LEDs. I mixed the larvae to distribute them, waited so that they could calm down, stimulated them with UV-light from the side, and then with cyan light (480 nm) from the top to check for phototaxis.

After the larvae calmed down, they swam down to diffuse UV-light, except the 27-hour-old larvae. However, all larvae swam up when they were stimulated with cyan (480 nm) light from the top (Figure 14).

This shows that the UV-response is indeed non-directional and not coupled to phototaxis, because phototaxis develops before the UV-response. And the UV-response is the same whether phototaxis is mediated by the larval or the adult eyes.

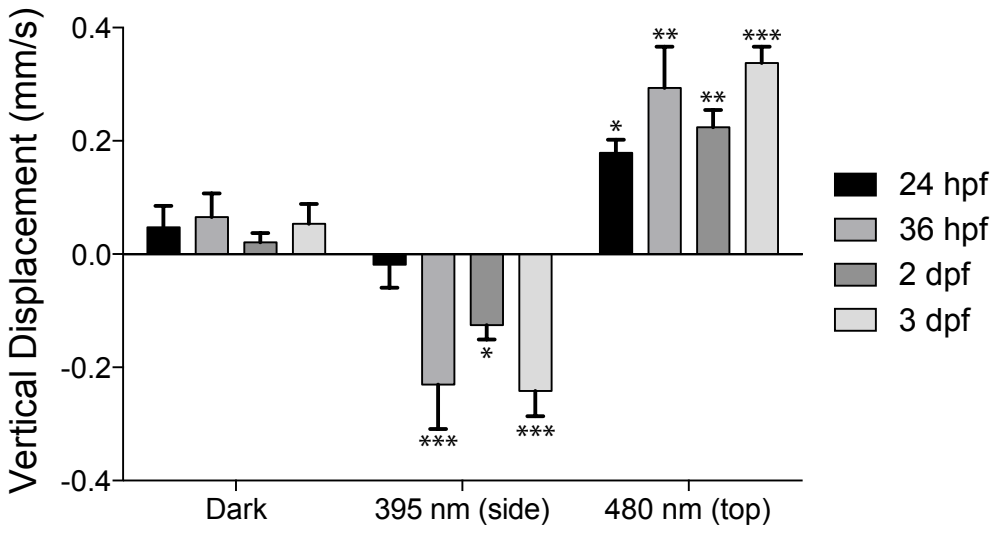


Figure 14: UV-light already made 36-hour-old larvae swam down

Platynereis dumerilii larvae of different ages were exposed to different light conditions in the vertical column: First, the larvae were mixed and kept for 4 min in the dark. Then, they were exposed to UV (395 nm) light from the site for 4 min. And finally, they were exposed to cyan (480 nm) light from the top for 4 min. For the dark control, the larvae's swimming was analyzed in an interval from 3.5 to 4 min after mixing. For the UV (395 nm) and cyan (480 nm) light stimuli, the larvae's swimming was analyzed in an interval from 1.5 to 2 min after stimulus onset. All error bars are SEM. Statistical differences are shown within the same age groups between dark control and 395 nm (side), and between dark control and 480 nm (top). The statistical significance was determined with a 2-way-ANOVA and Holm-Sidak post hoc test. P* < 0.05, P** < 0.01, P*** < 0.001. Abbreviations: hpf – hours post fertilization, dpf – days post fertilization. For 27 hpf n = 12, for 36 hpf n = 8, for 2 dpf n = 9, for 3 dpf n = 15.

3.11 The larvae swim down to diffuse UV-light in a narrow spectrum

The larvae swam down with UV-light irrespective of its direction. If the UV down-swimming response with light from top and the side is the same, then the spectral sensitivity should also be the same.

To record a response spectrum with diffuse UV-light from the side, I put 2-day-old larvae into a cuvette of 1 cm x 1 cm x 4.2 cm (L x W x H). I illuminated the larvae with a monochromator from one side and illuminated them from the other side with the same light reflected by a mirror. The larvae were stimulated with light from 340 nm to 680 nm in 20 nm steps. Each step lasted 1 min. The steps were separated by 1 min of darkness. The larvae were tracked and their horizontal displacement was analyzed for the first 30 s for phototaxis, and their vertical displacement was analyzed for the last 45 s for the UV-response.

When I put 2-day-old larvae into the cuvette, they distributed equally along its height. However, for some batches the larvae accumulated at the bottom, so that

I could not record their behavior. This was regular for 3-day-old larvae. Therefore, I only measured 2-day-old larvae.

In this setup, despite the mirror, the light is not coming equally from both sides, therefore, the larvae swam initially to the light and thus showed phototaxis (Figure 15A). The phototaxis response spectrum is like that reported by Jékely et al. (2008). It just extends more into the UV range. The larvae also showed the UV-response (Figure 15B). It had the same spectrum as c-opsin1 (Figure 10B) and matched the UV-response spectrum in the column (compare Figure 15B and Figure 13B).

So, the UV-response showed up in all the different setups, it is non-directional, and most likely it is mediated by c-opsin1 and the ciliary photoreceptor cells.

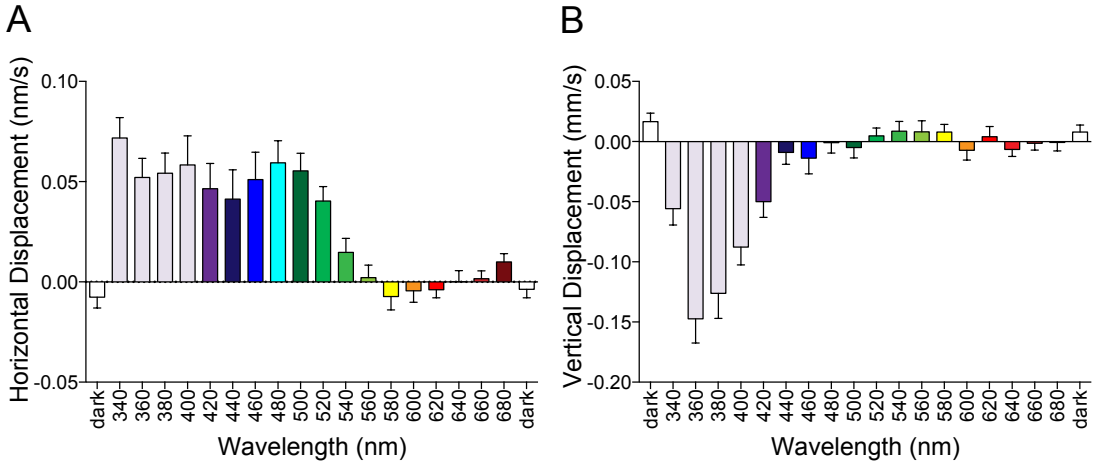


Figure 15: Separating phototaxis and the UV-response by direction

2-day-old *Platynereis dumerilii* larvae where stimulated in a cuvette with light of different wavelength. Each stimulus lasted 1 min. The light was provided by a monochromator from one side and reflected by a mirror to provide light from the other side. The wavelengths are color coded: UV wavelengths are colored grey, and the other wavelengths are colored as an average human observer would approximately perceive them. Each batch contained > 100 larvae. All error bars are SEM.

A: The horizontal displacement shows larval phototaxis to different wavelength. For phototaxis, the first 30 s of the stimulus interval were analyzed. Positive values mean swimming to the light. (n = 9)

B: The vertical displacement shows the larval UV-response. For the UV-response, the last 45 s of the stimulus interval were analyzed. Negative values mean swimming down. (n = 20)

3.12 *Platynereis dumerilii* larvae have a ratio-chromatic depth gauge

Go-opsin1 is not involved in a cellular antagonism. Nevertheless, the larvae swim up phototacticly and swim down to UV-light. Both behaviors still could work antagonistically and form a depth gauge, not on cellular level but on network level. Alternatively, the UV-response's main function is avoiding UV-light; then, it should override phototaxis.

Thus, I measured larvae in the vertical column, how they reacted to different ratios of UV (380 nm) and cyan (480 nm) light from the top. I programmed the

monochromator so that it generated the ratios by alternating the two wavelengths. For instance, the monochromator gave UV (380 nm) light for 450 ms and cyan (480 nm) light for 50 ms and then repeated it for 4 min. This is a ratio of 90 % UV-light versus 10 % cyan light. The monochromator also gave ratios of red (660 nm) and UV (380 nm) light. In the column, light of 660 nm induced phototaxis only weakly or not at all (Figure 6, Figure 9, and Figure 13), so that I could simulate darkness; even so, I could not use the shutter here.

The larvae were mixed and kept in darkness. Then they were exposed to the ratios from 100 % to 10 % in 10 % steps. Each ratio was followed by cyan (480 nm) light to redistribute the larvae. Each step lasted 4 min. I measured 2, 3, and 4-day-old larvae to determine the UV/cyan and UV/red ratios that do not make the larvae swim down anymore so that I could compare.

All larvae stopped swimming down at a certain UV/cyan ratio. However, the UV/cyan ratio differed from the UV/red ratio. The larvae swam down to less UV-light in the UV/red than in the UV/cyan ratio (Figure 16B and C). This may also be true for the 2-day-old larvae, however for them the ratios did not differ statistically (Figure 16A).

This means that phototaxis and the UV-response work antagonistically. However, for a ratio-chromatic depth gauge, the ratio must be the same at different intensities. Therefore, I repeated the UV/cyan ratio experiment with 3-day-old larvae but I placed a neutral density filter (1 OD) in front of the monochromator for lowering the intensity. At lower intensity, the larvae were neither swimming down nor up at the same ratio as at high intensity (Figure 16D). The ratio was 40 %, the larvae seem to vary so that the true ratio could also be closer to 50 %.

This shows that *Platynereis dumerilii* larvae have a ratio-chromatic depth gauge, which is independent of the light intensity. It may still be developing in 2-day-old trochophore larvae, but it is fully functional in 3-day-old nectochaete larvae.

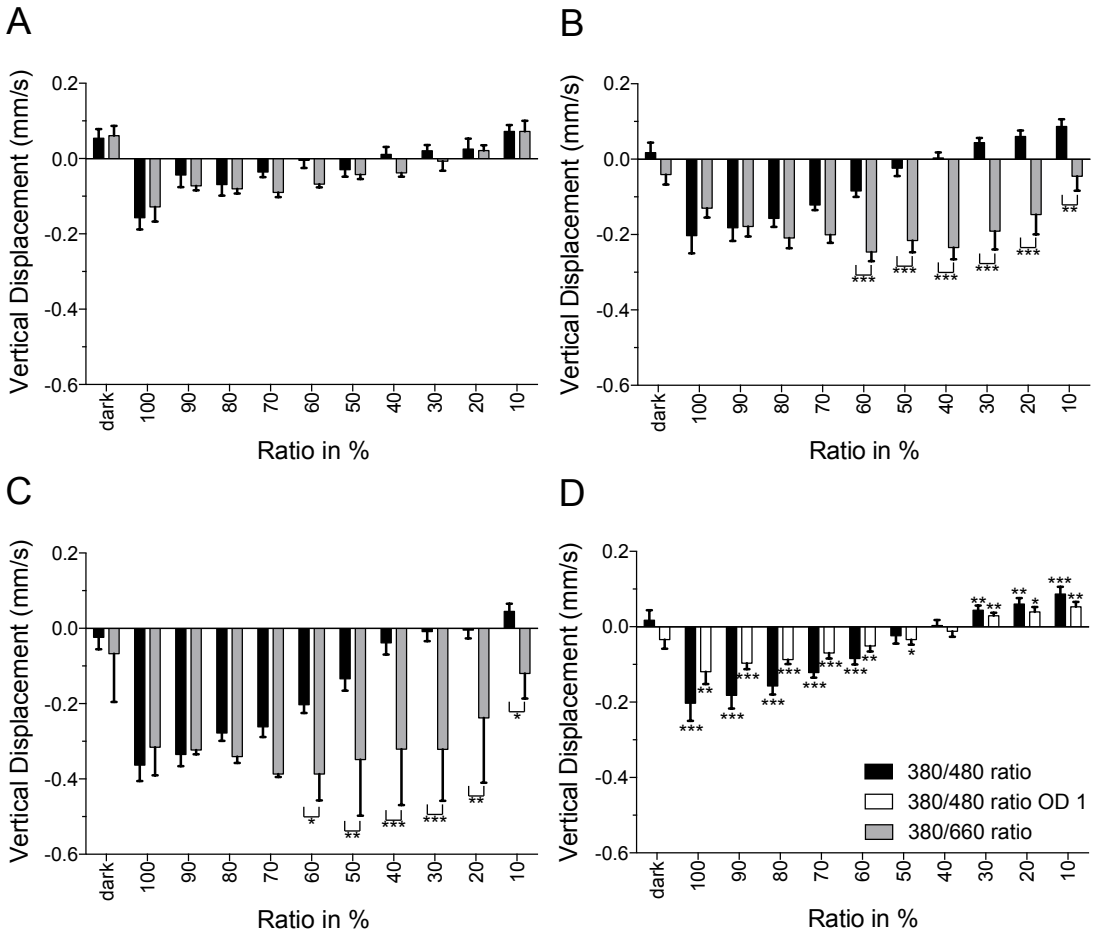


Figure 16: Platynereis dumerilii larvae have a ratio-chromatic depth gauge

2, 3, and 4-day-old *Platynereis dumerilii* larvae were stimulated with different ratios of UV/cyan (380 nm/480 nm) and UV/red (380 nm/660 nm) light coming from the top. For instance, a 90 % UV/cyan ratio was generated by giving for 450 ms UV-light and for 50 ms cyan light. This was then repeated for 4 min. Each stimulus lasted 4 min. The larvae were mixed before and kept in the dark for 4 min. The larvae's swimming was analyzed in an interval from 3.5 to 4.0 min after stimulus onset. Negative values indicate down-swimming. All error bars are SEM.

A-C: Statistical significance was determined with two tailed unpaired t-tests. P* < 0.05, P** < 0.01, P*** < 0.001.

A: 2-day-old larvae exposed to UV/cyan ratios (n = 6) and UV/red ratios (n = 4).

B: 3-day-old larvae exposed to UV/cyan ratios (n = 13) and UV/red ratios (n = 8). All differences are significant after Sidak-Bonferroni correction.

C: 4-day-old larvae exposed to UV/cyan ratios (same data as in B, n = 13) and UV/red ratios (n = 2). The differences at 40 % and 30 % are significant after Sidak-Bonferroni correction.

D: 3-day-old larvae exposed to UV/cyan ratios at full intensity (n = 13) and at a lower intensity, which was reduced by a neutral density filter of OD 1 (n = 12). Statistical significance was determined with two tailed one sample t-tests. The hypothetical mean was 0. P* < 0.05, P** < 0.01, P*** < 0.001. For 50 % the larvae at high intensity may not swim down anymore, however their vertical displacement does not differ from those larvae at lower intensity (two tailed unpaired t-tests, $\alpha = 0.05$, not shown).

4 Discussion

Animal opsins have been classified into four major groups: The rhabdomeric opsins, the ciliary opsins, the xenopsins, and the tetraopsins (Cronin and Porter, 2014; Delroisse et al., 2014; Feuda et al., 2012; Feuda et al., 2014; Porter et al., 2012; Ramirez et al., 2016). Among the tetraopsins, the Go-opsins are poorly studied because they are lost in both ecdysozoans (Hering and Mayer, 2014) and vertebrates (Porter et al., 2012), the groups that contain our classical model organisms. However, *Platynereis dumerilii* expresses a Go-opsin, *Go-opsin1*, in the eyes of its larva (Figure 2). Therefore, I could study a Go-opsin via behavior and zinc-finger-nuclease mediated knockout in Platynereis dumerilii nectochaete larvae.

The larvae swam down to UV light (400 nm) and up to green (520 nm) light. Originally, I hypothesized that these behaviors were negative and positive phototaxis, respectively. I hypothesized that negative phototaxis was mediated by Go-opsin1 and positive phototaxis by the rhabdomeric opsins. Since these opsins are expressed in the same cells and rhabdomeric opsins depolarize, while at least a scallop Go-opsin hyperpolarizes, I hypothesized that both kinds of opsins antagonized each other to form a ratio-chromatic depth gauge. Such a cellular depth gauge was proposed by Nilsson (2009, 2013). Nilsson also proposed that the reason, why different classes of opsins that either hyperpolarize or depolarize exist, are chromatic antagonisms. Here, the chromatic antagonism would have switched the larvae between positive and negative phototaxis.

However, the phototaxis sign was neither switched by a cellular chromatic antagonism nor by Go-opsin1. Go-opsin1 contributed instead with the rhabdomeric opsins to phototaxis. And the phototaxis sign was switched by an unknown mechanism not depending on the wavelength. The wavelength, however, made the larvae swimming down not by phototaxis but by a fast, non-directional response to UV-light. This UV-response and phototaxis formed a depth gauge, not on cellular level but on network level.

4.1 Go-opsin1 contributes with other opsins to phototaxis

Here, I characterized the first Go-opsin *in vivo*: *Go-opsin1* and two rhabdomeric opsins are co-expressed in the adult eyes of the nectochaete larva of *Platynereis dumerilii*. The adult eyes mediate phototaxis (Randel et al., 2014), thus phototaxis should be co-mediated by Go-opsin1. For Go-opsin1, I originally hypothesized that it formed with the rhabdomeric opsins on cellular level an

antagonistic chromatic depth gauge; a depth gauge as theoretically proposed by Nilsson (2009, 2013). Here, the rhabdomeric opsins would have responded to visible light from the top and would have made the larvae swimming up phototactically. The larvae would have been made swimming down by Go-opsin1.

However, my Go-opsin1 knockout larvae still swam down to UV-light and did not differ from the wild type larvae except at 340 nm. If this was an opsin phenotype then adjacent wavelengths should have been affected, too, because opsins cover wavelength ranges that are broader than 20 nm (Govardovskii et al., 2000; Lamb, 1995), and no opsin is known that absorbs maximally below 340 nm, but only opsins that absorb maximally above 350 nm (Govardovskii et al., 2000; Hunt et al., 2007; Townson et al., 1998). Therefore, this cannot be a specific phenotype, and it does not fit to the Go-opsin1 absorption spectrum, either.

The Go-opsin1 absorption spectrum ranges from 440 nm to 560 nm, the same wavelengths the *Go-opsin1* knockout larvae were less phototactic to. Phototaxis is mediated in nectochaete larvae by the adult eye (Randel et al., 2014), which expresses Go-opsin1. Therefore, I conclude that Go-opsin1 mediates phototaxis.

Phototaxis in the mutants is not abolished, only diminished, so that it cannot be mediated by Go-opsin1 alone (Foster and Hankins, 2002). At least two more opsins are expressed in the adult eye (Randel et al., 2013). These are, however, rhabdomeric opsins. Rhabdomeric opsins depolarize while at least a scallop Go-opsin is thought to hyperpolarize (Kojima et al., 1997). In principle, Go-opsin1 and the rhabdomeric opsins could still form a chromatic antagonism. Then the rhabdomeric opsins would mediate the UV-response. However, the UV-response is non-directional, and thus cannot be mediated by the directional light sensing adult eye. Additionally, the adult eye's photoreceptor cells do not hyperpolarize at all, but both UV (405 nm) and cyan (488 nm) light makes them depolarize (Gühmann et al., 2015), so that the rhabdomeric opsins and Go-opsin1 depolarize and mediate a broad phototaxis response.

The phototaxis response is so broad that it may be mediated by three opsins, but an opsin spectrum template (Govardovskii et al., 2000) suggests even four or five opsins. If three opsins are expressed in the adult eye, then even more opsins could be expressed there. Candidate opsins are, which have been cloned and verified by phylogenetic reconstruction (Gühmann et al., 2015; Randel et al., 2013): *R-opsin2*, *r-opsin4*, *r-opsin5*, and *Go-opsin2*. I found eight more putative opsin sequences in the Platynereis dumerilii whole body (Conzelmann et al., 2013a) and single cell (Achim et al., 2015) transcriptomes (data not shown) with

the help of Elisabeth A. Williams. I putatively identified these sequences as opsins via BLAST (Altschul et al., 1990). Among these opsin sequences were four more putative rhabdomeric opsins, which still must be confirmed by phylogenetic reconstruction. These are more opsins than needed for a broad phototaxis response with four or five opsins.

4.1.1 Go-opsin1 and the circalunar clock

Go-opsin1 has beside phototaxis also other functions, because it is not only expressed in the adult eyes, but also in the larval eyes and in a medio-lateral cell. However, what these functions are, is unknown. One of the functions may be entraining the circalunar body clock to the moon phase. However, the clock was not affected in the *Go-opsin1*$^{\Delta 8/\Delta 8}$ knockout worms. They became epitokes, when the nights were dark and did not became epitokes when the nights were illuminated. This is how wild type worms behave as reported by Zantke et al. (2013) and Hauenschild (1955).

However, Hauenschild (1956) also reported that most epitokes came between 16 and 20 days after the start of the short-day-period. At the start of the short-day-period, my mutant and the wild type worms became already epitokes. This differed, because my worms were exposed to artificial bright light at day and dim light at night, during the moon phase, while Hauenschild's worms received 24 hours artificial dim light of the same intensity, which is a bigger relative contrast.

This could help to better synchronize the spawning of mutant worms so that they can be bred better: Mature epitokes die after spawning or within 24 hours, even so if they have not spawned (Fischer and Dorresteijn, 2004; Fischer et al., 2010). My epitokes could survive longer if they were cooled to 4°C in the fridge. This way, I could gain another day or exceptionally five days, but still this did not guarantee that the epitokes would survive the night. To solve this problem of getting enough epitokes, I genotyped many worms, which is work and time intensive, so that a lot could be gained here by a stronger moon.

Since the circalunar clock was not affected in the *Go-opsin1*$^{\Delta 8/\Delta 8}$ knockout worms, Go-opsin1 may not regulate it. However, the clock can be entrained by monochromatic violet (433 nm) or red (629 nm) light (Hauenschild, 1956). This is a broad range, broader than one opsin could cover (Lamb, 1995), so that the circalunar clock is entrained by many opsins. If one of them is Go-opsin1, then I could not have detected an effect, because I used white light covering most of the visible spectrum. If I had specifically looked for a phenotype by providing cyan

(488 nm) light matching the absorption maximum of Go-opsin1, I still could have missed a phenotype, because the other opsins may also be sensitive to cyan (488 nm) light and so mask a circalunar clock phenotype.

The circalunar clock should be entrained by cells or an organ that expresses many opsins. The organ should also exist in the adult, since the circalunar clock can be entrained there (Hauenschild, 1955, 1956). Such an organ is the adult eye, which expresses at least two rhabdomeric opsins (Backfisch et al., 2013; Randel et al., 2013). However, whether it expresses also *Go-opsin1* or any other opsin beyond the larval stage is unknown. At least five rhabdomeric opsins, two Go-opsins, and two neuropsins of *Platynereis dumerilii*, which are published and confirmed by phylogenetic reconstruction (Gühmann et al., 2015), could be all expressed in the adult eyes, as well.

However, if the adult eyes are removed, the circalunar clock can still be entrained so that the adult eyes do not entrain it (Hauenschild, 1961), at least not alone. In principle, the circalunar clock could be entrained by any photoreceptor cells that express all the opsins covering a broad spectrum. Such photoreceptor cells could be the ones of the notopodium, even so, they mediate a photo-avoidance response (Backfisch et al., 2013), the photoreceptor cells of the larval eye, or the median *Go-opsin1* expressing cell. In principle, all these photoreceptor cells together could entrain the circalunar clock. In the mouse, the circadian clock is also entrained by more than one type of photoreceptor cells: The melanopsin expressing retinal ganglion cells, and the rods and cones (Panda et al., 2003).

4.1.2 What is the G-protein that Go-opsin1 activates?

In the adult eye of the *Platynereis dumerilii* larva, the opsins belong to different groups: The rhabdomeric opsins and the Go-opsins. The adult eye's photoreceptor cells only depolarize whether they respond to UV (405 nm) or cyan (488 nm) light (Gühmann et al., 2015), so that Go-opsin1 must activate a different phototransduction cascade than the scallop Go-opsin.

Go-opsin1 may depolarize via a Go-protein. For example, in the parietal eye of the lizard, parietopsin depolarizes via a Go-protein. The Go-protein inhibits a phosphodiesterase in a ciliary phototransduction cascade in ciliary photoreceptor cells (Su et al., 2006). However, in the adult eye of *Platynereis dumerilii*, the photoreceptor cells are rhabdomeric (Rhode, 1992), and thus probably use a rhabdomeric phototransduction cascade, which is conserved

between vertebrates and insects (see section 1.1.2), and likely in *Platynereis dumerilii*, too.

Therefore, if Go-opsin1 activates a Go-protein, the Go-protein may activate the phospholipase C of the rhabdomeric phototransduction cascade, either via its alpha subunit or its beta-gamma complex. For instance, the beta-gamma complex of Ggust (gustducin) activates a phospholipase C in mammalian taste receptor cells (Margolskee, 2002). Alternatively, the Go-protein may activate a completely different phototransduction cascade that depolarizes in parallel to the rhabdomeric phototransduction cascade. In fact, a G-protein coupled signal transduction cascade can be assembled in many ways (Hepler and Gilman, 1992), but the simplest way for an opsin to depolarize in a rhabdomeric photoreceptor cell is via a Gq-protein.

For G-protein coupling, opsins have a tripeptide motif. The tripeptide motif between rhabdomeric and ciliary opsins differ greatly (Arendt et al., 2004). The tripeptide motif of cattle rhodopsin interacts with the Gt-protein alpha subunit. If one of its peptides is mutated, it still interacts well. However, if all three amino acids are mutated, it activates the Gt-protein 75% less (Marin et al., 2000). The tripeptide motif of Go-opsin1 is between the motifs of rhabdomeric and ciliary opsins (Gühmann et al., 2015), so that Go-opsin1 may indeed interact with another kind of G-protein in the *Platynereis dumerilii* adult eye than in the scallop eye. However, which G-protein is unknown, but most likely it is the Gq-protein.

4.1.3 Other opsins that may couple to other G-proteins *in vivo*

Opsins couple *in vitro* to different G-proteins: For instance, human melanopsin and rhodopsin couple to Gi/o- and Gq-proteins. However, these opsins may activate non-native G-proteins less efficiently (Bailes and Lucas, 2013). Mouse melanopsin couples to a Gt-protein (Newman et al., 2003), too; and may do so in a cone, as well. Among human cones, 0.11 % to 0.55 % express exclusively melanopsin. Melanopsin cones are restricted to the peripheral retina (Dkhissi-Benyabya et al., 2006). The peripheral retina becomes tetrasensitive at high photopic light levels while the fovea stays trisensitive. Trisensitive and tetrasensitive means that the sensitivity of the retina is covered by three or four classes of photopigments, respectively (Horiguchi et al., 2013). The tetrasensitivity could be achieved by the melanopsin cone, but the melanopsin expressing retinal ganglion cells could contribute, too.

The mouse blood vessels are relaxed light-dependently by melanopsin. Melanopsin hyperpolarizes the vessel cells via a phosphodiesterase-6 and cGMP (Sikka et al., 2014), so that melanopsin may not couple here to a Gq-protein, but to another G-protein whose identity is, however, unknown.

In humans, rhodopsin and the cone opsins are not only expressed in the rods and cones, but also in keratinocytes (Haltaufderhyde et al., 2015) and melanocytes (Haltaufderhyde et al., 2015; Wicks et al., 2011). In melanocytes, rhodopsin is suspected to activate early melanin synthesis on UV-exposure (Wicks et al., 2011). There, rhodopsin would couple to a Gq/11-protein (Bellono et al., 2014) that activates phospholipase C (Wicks et al., 2011) and eventually would depolarize the cell by opening TRPA1 channels (Bellono and Oancea, 2013). This response needs retinal and is reduced if rhodopsin is knocked down, but its response spectrum of 320 nm to 400 nm does not match to rhodopsin (Wicks et al., 2011), which peaks at 500 nm (Wald and Brown, 1958).

Different peaks of *in vitro* absorption have been reported for melanopsin. The peaks were between 420 nm and 480 nm, with 480 nm fitting to the action spectrum of the melanopsin expressing retinal ganglion cells (Tu et al., 2005). The different peaks seem to depend on the cell type melanopsin is expressed in (Shirzad-Wasei and DeGrip, 2016). Therefore, a property of the melanocytes could influence the absorption of rhodopsin, but how phototransduction is affected is unclear, it may be even initiated unspecifically by UV-light (Foster and Hankins, 2002).

If phototransduction is initiated unspecifically then it may be rather thermotransduction. Thermotransduction could be initiated by UV-light, which heats up the melanocytes more than visible light. Thermotransduction exists in human and mouse spermatozoa. In the spermatozoa, thermotaxis is mediated by rhodopsin. The spermatozoa also express the cone opsins, encephalopsin, neuropsin, and melanopsin, which may also contribute to thermotaxis (Perez-Cerezales et al., 2015). In *Drosophila melanogaster* larvae, thermotaxis is mediated by dm-r-opsin1, which can be replaced by most of the other fly rhabdomeric opsins, and even by mouse melanopsin (Shen et al., 2011). In the *Drosophila melanogaster* larva, thermotransduction also involves a Gq-protein, a phospholipase C, and a TRPA1 channel (Kwon et al., 2008). Interestingly, TRPA1 is also required for UV-activated melatonin synthesis in human melanocytes (Bellono and Oancea, 2013). However, if the melanocytes are thermosensitive then they should also react to blue and green light, because it would heat them up, too. But this light does not affect the melanocytes (Wicks et al., 2011).

The melanocytes spectral response matches better to neuropsin. Neuropsin absorbs maximally at 380 nm in humans and mice, couples to Gi- and Go-proteins (Kojima et al., 2011) and is also expressed in human melanocytes (Haltaufderhyde et al., 2015), however at least the chicken neuropsin does not couple to a Gq-protein (Yamashita et al., 2010). The melanocytes also express neuropsin (Haltaufderhyde et al., 2015), however, due to expression difficulties (Terakita et al., 2004), its sensitivity and G-protein coupling is unknown. Even if neuropsin initiated the response, why the response is reduced if rhodopsin expression is reduced would still be unclear.

In human keratinocytes, rhodopsin is also suspected to regulate how differentiation markers are expressed. The expression changes systematically, when rhodopsin is knocked down or overexpressed, and a Gi-protein is involved. But the expression is regulated by UV and violet light (350 nm – 420 nm) (Kim et al., 2013), which does not match the absorption spectrum of rhodopsin either. For the expression regulation, neuropsin and the SWS1 cone opsin, which is maximally sensitive at ~425 nm (Fasick et al., 1999; Merbs and Nathans, 1992; Oprian et al., 1991), would fit better.

In the cichlid fish *Oreochromis niloticus*, the cone opsins SWS1, RH2b, and LWS (short wavelength sensitive 1, rhodopsin like 2b, and long wavelength sensitive opsin) are coexpressed in dermal erythrophores (Chen et al., 2013). The erythrophores contract their pigment granules on UV/violet (365 nm – 440 nm) or red (beyond 600 nm) light exposure. But on blue/yellow (460 nm – 580 nm) light, they expand the granules (Chen et al., 2013; Chen et al., 2015; Sato et al., 2004). The granules are contracted below 440 nm to the same wavelengths that SWS1 absorbs. And they are expanded to the same wavelengths that RH2b absorbs. This suggests that SWS1 and RH2b mediate the two responses, respectively and work antagonistically (Chen et al., 2015). LWS may mediate the pigment granule contraction above 600 nm (Ban et al., 2005). The erythrophores depolarize below 440 nm (Chen et al., 2015). However, this seems to be unrelated to the contraction, because the erythrophores neither depolarize nor hyperpolarize on the expansion (Chen et al., 2015) or contraction above 600 nm. The contraction above 600 nm is mediated by a Gi-protein that inhibits an adenylyl cyclase. The adenylyl cyclase is activated by a Gs-protein that mediates the expansion. The adenylyl cyclase regulates the intracellular level of cAMP, which regulates the pigment granule expansion and contraction via a phosphokinase A (Ban et al., 2005). The Gs-protein may couple to the RH2b opsin and the Gi-protein to the SWS1 and LWS opsins.

In the lizard parietal eye, the antagonistically working pinopsin and parietopsin are another example that an opsin changed its G-protein partner if not in the same animal but at least during evolution. The parietopsin couples to a Go-protein, depolarizes (Su et al., 2006), and is a ciliary opsin (Su et al., 2006; Wada et al., 2012). However, the proto ciliary opsin most likely hyperpolarized via a Gi-protein (Lamb, 2013), therefore parietopsin must have changed its G-protein during evolution. Parietopsin may have coupled initially to both the Gi- and the Go-protein and was later optimized to activate the Go-protein, only.

4.1.4 Why do different phototransduction cascades exist for opsins?

Nilsson (2009, 2013) speculated that opsins use different phototransduction cascades, because of ancient chromatic antagonisms. However, the antagonisms of the cichlid erythrophores and the lizard parietal eye only involve ciliary opsins, and thus are not ancient. Even so, Su et al. (2006) speculated so. Ancient chromatic antagonisms that involve rhabdomeric and ciliary opsins have so far not been found.

Animal opsins, evolved in animals (Feuda et al., 2014), but G-protein coupled receptors predate animals (Krishnan and Schioth, 2015) and are found across eukaryotes, the same is true for G-proteins. The G-proteins of animals have G-alpha subunits from five groups, which predate animals (de Mendoza et al., 2014). Therefore, the question is not why different phototransduction cascades exist for opsins, but why different signal transduction cascades exist for G-protein coupled receptors. Additionally, not only G-protein coupled receptors transduce signals, but also ionotropic receptors and gap junctions.

These are many ways to deliver signals to a cell. The cell must integrate the signals and may have to react in opposite ways to them. For instance, rod photoreceptor cells do not only hyperpolarize, but also depolarize while they are responding to synaptic input from horizontal cells activated by cones under bright light (Szikra et al., 2014). Here, an opsin is just another component in a very modular G-protein signaling system (Birnbaumer, 2007; de Mendoza et al., 2014), which can be simply modified in the lab by adding an opsin and maybe retinal to make a cell sensing light (Cao et al., 2012; Lin et al., 2008). This may not only happen in the lab but also in nature during evolution.

For instance, rhabdomeric and ciliary photoreceptor cells increase their membrane area by stacking the membrane via microvilli and cilia, respectively. Both microvilli and cilia are older than opsins and animals. Cilia are associated in eukaryotes with cAMP and cGMP signaling, which is used by ciliary

photoreceptor cells (Oakley and Speiser, 2015). Rhabdomeric photoreceptor cells may even be derived from mechanoreceptor cells, because their TRP channels are activated mechanically via membrane deformation (Christensen and Corey, 2007; Spassova et al., 2006).

Therefore, the mechanisms that diversify opsins are duplication and ectopic expression, which then becomes normal expression. An evolutionary recent example may be melanopsin in a cone (Dkhissi-Benyabya et al., 2006). Not only an opsin can be integrated into another cell type, but also whole functional modules (Arendt et al., 2016). Ectopic expression does not only diversify opsin signaling but also function. Ectopic expression may have created ciliary and rhabdomeric photoreceptor cells by adding an opsin. For instance, an opsin may have turned a chemo avoidance response into a light avoidance response.

Initially, an opsin does not need to activate its new G-protein efficiently; it only needs to activate it at all. Then the opsin could be optimized with other components of its new cell. This includes increasing the binding efficiency of the opsin to the G-protein and stacking the cell membranes, which increases the sensitivity and the temporal resolution of the photoreceptor cell, so that the cell can mediate complex tasks like spatial vision, which requires high integration time and sensitivity (Nilsson, 2009, 2013). For high sensitivity and fast response, the vertebrate ciliary photoreceptor cells, opsins and signal transduction cascade have been optimized (Lamb, 2013).

The signal transduction cascades of different G-protein coupled receptors differ, and differ even between cell types (Birnbaumer, 2007), because they serve different functions. And opsins were just plugged into one or another of the cell types. That is why opsins have different phototransduction cascades.

4.1.5 What determines the phototransduction cascade?

Opsin research has focused on visual opsins and their photoreceptor cells, because they are the most visual and most interesting to the human researcher like Boll (1876) who studied the purple frog retina. Opsins that mediate non-visual functions are mainly studied for a relative short time, starting with RGR-opsin (Jiang et al., 1993; Shen et al., 1994), and continuing with peropsin (Sun et al., 1997), melanopsin (Provencio et al., 1998; Provencio et al., 2000), encephalopsin (Blackshaw and Snyder, 1999), and neuropsin (Tarttelin et al., 2003). In humans, these opsins are accompanied by rhodopsin and the three cone-opsins. This totals to nine human opsins, most of them non-visual opsins. Since all cone opsins and rhodopsin are expressed in human spermatozoa

(Perez-Cerezales et al., 2015), all human opsins may have non-visual functions, too.

Even so, five of nine human opsins are primarily non-visual opsins; phylogenetic opsin datasets are biased to visual opsins (Nickle and Robinson, 2007; Plachetzki et al., 2010; Porter et al., 2012). The visual opsins are expressed in photoreceptor cells of image forming eyes, which are highly specialized and optimized systems (Lamb, 2013). Such systems are moved from optimum by relative small changes, so that visual systems can be reduced to base-types, for instance: The vertebrate eye, the protostome eye, the parietal eye of the lizard, which seems however only be a variation of the vertebrate eye, the scallop eye, and the cube jellyfish eye, whose opsins couple to a Gs-protein and increase the intracellular cAMP concentration (Koyanagi et al., 2008; Liegertova et al., 2015), however it cannot be generalized as Shichida and Matsuyama (2009) did that all cnidarian opsins couple to a Gs-protein. In fact, cnidarian opsins exist that couple to other G-proteins (Liegertova et al., 2015; Mason et al., 2012).

Something similar may be true for Go-opsins; they are only called Go-protein coupled opsins, because the first Go-opsin was found coexpressed with a Go-protein (Kojima et al., 1997). This was the reason to assume that all Go-opsins are Go-coupled (Shichida and Matsuyama, 2009). Go-opsins in scallop eyes and derived systems may indeed couple to Go-proteins, but in other systems they may just integrate. For instance, in a rhabdomeric photoreceptor cell, they may just couple to a Gq-protein. And melanopsin in a human cone (Dkhissi-Benyabya et al., 2006) may just couple to a Gt-protein, even so with low efficiency (Horiguchi et al., 2013). Therefore, phylogeny does not determine the phototransduction cascade of an opsin (Porter et al., 2012), but the cell it is expressed in.

4.2 The UV-response forms with phototaxis a depth gauge

I found a ratio-chromatic depth gauge, which is formed by phototaxis and a UV-response. The UV-response was fast; the larvae could already swim down after 0.5 s of UV-light (data not shown). The UV-light could come from the top, the bottom, or diffusely from the side; and the larvae swam down. This contradicts my original hypothesis that the UV-response was negative phototaxis. Instead of phototaxis, the UV-response is a non-directional light response or a UV-induced gravitaxis.

Platynereis dumerilii differs here from *Daphnia magna*. *Daphnia magna* swims up to visible light (420 nm – 600 nm) and swims down to UV-light (260 nm –

380 nm). However, if UV-light (350 nm) comes from one side, *Daphnia magna* swims away, but does neither swim up nor down. And so, *Daphnia magna* avoids UV-light by negative phototaxis (Storz and Paul, 1998). Other species, like medical leeches (Jellies, 2014) or sea urchins avoid UV-light by phototaxis, too, and may even cover themselves with shells (Sharp and Gray, 1962). In planktonic species, UV-avoidance is usually attributed to phototaxis (Leech and Jonsen, 2002), even so UV-avoidance is studied as outcome, for instance whether planktonic organisms avoid UV-exposed areas (Donahue and Schindler, 1998), how UV-transparency influences them in lakes (Kessler et al., 2008), or which depth they chose after UV-exposure (Leach et al., 2015; Rhode et al., 2001). For the depth distribution, UV-phototaxis is just assumed and a fast non-directional light response is not considered. To my knowledge, a fast non-directional UV-response has not been described in zooplankton, which is probably why Nico K. Michiels had to point this out to me while I was visiting his lab.

Both *Platynereis dumerilii* late trochophore and nectochaete larvae avoid UV-light; even so the trochophores use the larval eyes (Jékely et al., 2008) and the nectochaetes the adult eyes (Randel et al., 2014) for phototaxis. Phototaxis is already shown by the early trochophore larvae (Jékely et al., 2008), earlier than the UV-response. Thus, the UV-response is separated developmentally from the eyes and their directional photoreceptor cells. The UV-response should be mediated by photoreceptor cells that are not shaded by pigment, so that they can detect light from all sides. They should also integrate light over a short period, so that the larvae can response quickly. For that, the photoreceptor cells need stacked membranes (Nilsson, 2009, 2013).

These characters fit to the ciliary photoreceptor cells. They exist already in the metatrochophore larva, are not shaded by pigment, have stacked membranes, and express *c-opsin1* (Arendt et al., 2004). C-opsin1 absorbed light of the same wavelengths as the larvae were swimming down to. The down-swimming response spectrum fits to an opsin template of 100 nm width (Govardovskii et al., 2000; Lamb, 1995), so that the UV-response may be mediated by c-opsin1 alone.

To check this, I searched the single transcriptome data of Achim et al. (2015) for cells expressing *c-opsin1* and other opsins with the help of Elizabeth A. Williams. I found ten known opsin sequences (Gühmann et al., 2015) and eight putative opsin sequences. Among these opsins, *c-opsin1* was expressed in four cells: Two cells expressed *c-opsin1* only, and two cells coexpressed *peropsin1*, which absorbs maximally at 500 nm (data not shown). An amphioxus peropsin

preferentially binds all-*trans*-retinal and isomerizes it to 11-*cis*-retinal. Thus, peropsins are thought to be photoisomerases. Peropsins may also signal via a G-protein (Koyanagi et al., 2002), but which G-protein they activate is unknown. The *Platynereis dumerilii* peropsin1 has been reclassified as a retinochrome/RGR-opsin (Ramirez et al., 2016). RGR-opsins work as photoisomerases and are claimed not to bind any G-protein (Nagata et al., 2010; Terakita, 2005), because they have an NAxxY motif instead of an NPxxY motif. The NPxxY motif is highly conserved among G-protein coupled receptors. If it is mutated in the rat m3 muscarinic receptor to NAxxY, the receptor can be activated less efficiently (Wess et al., 1993). However, the human MT2 melatonin receptor signals via a G-protein and has an NAxxY motif natively. If that motif is mutated to NPxxY, the receptor cannot be activated, but the receptor can be rescued partially if it is mutated to NVxxY (Mazna et al., 2008). The NVxxY motif is present in peropsin1 and therefore it remains to be seen whether it and RGR-opsins in general signal via G-proteins. Independently, whether peropsin1 signals, it maximally absorbs light at 500 nm (data not shown), which does not fit to the UV-response. Therefore, the UV-response is not mediated by peropsin1, but probably by c-opsin1 alone.

The UV-response is not a UV-avoidance response alone, because it did not override phototaxis. Instead, phototaxis and the UV-response worked against each other and canceled out each other at a certain ratio. The ratio did not change when I dimmed the light. However, when I replaced the cyan (480 nm) light by red light (660 nm) in the ratio, the larvae swam down to less intense UV-light (380 nm). The intensity of the UV-light did not change, but because red light hardly induced phototaxis compared to cyan light, the perceived ratio of UV-light vs. visible light did change. Therefore, the UV-response and phototaxis form a depth gauge. It could help the larvae to avoid UV-light, however this does not seem to be the main function, because otherwise, the absolute intensity of the UV-light would matter, and the UV-response would override phototaxis.

However, the depth gauge could have started evolutionary as UV-avoidance. UV-avoidance may have worked with a long integration time (Nilsson, 2009), because it had not to compete with phototaxis. Then phototaxis was added to the system. For instance, by adding extra synaptic input to the ciliated motor cells or to interneurons, so that the input can be integrated there. However, we can expect, that initially, UV-avoidance and phototaxis were not integrated perfectly, so that they would work against each other and form a depth gauge, automatically. Then UV-avoidance could have evolved to override phototaxis.

However, avoiding UV-light was not the main function and so the UV-response did not evolve to override phototaxis. But phototaxis became more sensitive and faster so that the UV-response became also more sensitive and faster to still compete with phototaxis.
In principle, the depth gauge and UV-avoidance may not be the only functions of the ciliary photoreceptor cells, they may, as speculated by Arendt et al. (2004), entrain the circadian clock. They may do it together with other photoreceptor cells. This would be like the vertebrate rods and cones, which resemble molecularly the ciliary photoreceptor cells. They also mediate vision and entrain the circadian clock together with the melanopsin expressing retinal ganglion cells (Panda et al., 2003).

4.2.1 The function of the depth gauge

The depth gauge may still help the larvae to avoid UV-light. But then, the UV-response must be already triggered at harmless levels of UV-light, so that the UV-response can out-compete phototaxis at harmful levels. The depth gauge with the UV-response allows the larvae to swim horizontally to or away from the light and at the same time allows them to swim down. If the larvae are in the open ocean, positive phototaxis drives them to the top while the UV-response brings them down. The UV-response and phototaxis work antagonistically so that the larvae reach a certain depth, at which they are transported to shallow water. If the water is shallower than the depth, the larvae reach the ground and may have found a good settling place. There, the larvae may switch from positive to negative phototaxis, like the larvae of the hydroid *Clava multicornis* (Williams, 1965) and the sinistral spiral tubeworm *Spirorbis borealis* (Williams, 1964) do when they settle on their favorite brown alga.
Other settlement cues, for instance for barnacle larvae, are pits, grooves, and holes, which are sensed mechanically (Crisp and Barnes, 1954; Knight-Jones and Crisp, 1953). Many more different settlement cues are described (Hadfield, 2011; Pawlik, 1986, 1992; Rittschof et al., 1998; Woodin, 1991), which in principle could all switch *Platynereis dumerilii* larvae to negative phototaxis. With negative phototaxis, the larvae know precisely were the light is coming from (Randel et al., 2014) and can swim into the shade of sea grass or rock cracks, where they can hide from predators and UV-light. Therefore, the larvae use the depth gauge to find the right depth for settling and avoiding UV-light may be just a side function if it is a function at all.

The UV-response is not shown by early trochophore larvae. They swim up to the surface phototacticly; even so, they are as transparent and pigmented as the nectochaete larvae and so are harmed by UV-light as much as the nectochaete larvae. However, the larvae may repair UV-induced DNA-damage efficiently (Zagarese and Williamson, 1994) and thus would not need to avoid UV-light.
The larvae may settle in shallow water, where the adults live (see section 1.2.1). But they could also settle deeper or higher and later migrate to their final habitat, since the adult worms are mobile and can build a new tube elsewhere (lab observation; Ricevuto et al., 2014). The metatrochophore and nectochaete larvae react phototacticly to many wavelengths that occur in shallow water. Therefore, shallow water might be their habitat, because the sensitivity hypothesis predicts that the sensitivity of an animal reflects the spectral distribution of light in its environment (Cohen and Forward, 2002; Munz, 1958). For larvae, however, that may differ and their spectral sensitivity reflects the spectral distribution of the light in the adult habitat (Forward and Cronin, 1979), which the larvae should eventually reach to settle there (Pawlik, 1992). To reach the habitat, the larvae use the depth gauge; and with phototaxis, they select a specific site.

4.2.2 Why a UV down-swimming response instead of negative phototaxis?

Platynereis dumerilii larvae use positive phototaxis with the UV-response as a depth gauge. However, the water flea *Daphnia magna* regulates its depth by switching between positive and negative phototaxis: They are photopositive for visible light above 420 nm and photonegative for UV-light below 380 nm (Storz and Paul, 1998). In principle, *Platynereis dumerilii* larvae could do the same. So why does *Platynereis dumerilii* use the UV-response with positive phototaxis as depth gauge? What is the advantage?
Daphnia lives in the open water column for its whole life (Ebert, 2005), while *Platynereis dumerilii* larvae leave the open water column and settle at the bottom of the sea. In these two taxa, UV induced down-swimming has different functions: *Daphnia* uses it to avoid UV-light (Storz and Paul, 1998) and visual predators (Ebert, 2005); *Platynereis dumerilii* may use it to find the right settling depth.
Daphnia magna needs only to avoid UV-light or visual predators. Therefore, it just needs to swim down by negative phototaxis until the UV-light falls below a certain threshold. Then, *Daphnia magna* can switch to positive phototaxis and swim up again. Practically, positive and negative phototaxis will balance at the

switching point so that *Daphnia magna* stays at a certain depth. Negative phototaxis may drive *Daphnia magna* into hiding places. But once the UV-light is gone, *Daphnia magna* may leave by positive phototaxis and enter the water column, again. This behavior is useful for diel-vertical migration, as *Daphnia* shows (Ebert, 2005).

Platynereis dumerilii larvae, however, should settle once they found a good place, which may be a hiding place in a rock crack or under sea grass, where shading may reduce UV-light exposure. In principle, this may be fine, as light of all the other wavelengths is dimmed there too, and so the depth gauge still indicates the same depth. However, a pure phototaxis depth gauge would have two functions: Bring down the larvae to their depth and push them into a hiding place. However, if a larva rejects a place, it must leave and is trapped there if it does not switch from negative to positive phototaxis, which however brings the larva back to the surface.

Already a few centimeters above the bottom, the larvae encounter currents that are faster than they could swim (Woodin, 1991) and that differ in speed at microscale (Koehl and Hadfield, 2010; Koehl and Reidenbach, 2007), so that a larva that swims up a few centimeters is moved several meters away. This is inefficient, because good and bad habitat patches are small and adjacent (Woodin, 1991). A patch's quality may be indicated by an odor. An odor may be poisonous (Walters et al., 1996) or indicate a competitor for space and food (Pawlik, 1992; Woodin, 1991; Woodin et al., 1997). Odors come in small plumes and not in concentration gradients (Koehl and Hadfield, 2010; Koehl and Reidenbach, 2007). Thus, a better place may be just a few centimeters away.

Therefore, the larvae should stay on the bottom and swim a few centimeters further to find a better place. They can do this with the depth gauge: The UV-response keeps them at the bottom; and with phototaxis, they can leave a bad place and swim to a good settlement place.

4.3 Outlook and evolutionary context of ciliary and Go-opsins

I found that *Platynereis dumerilii* larvae show positive gravitaxis when they are illuminated with non-directional UV-light. The spectrum of this UV-response matched the absorption spectrum of c-opsin1, and the other characters of the UV-response matched to the ciliary photoreceptor cells, so that most likely c-opsin1 mediates the UV-response. However, the UV-response could still be mediated by another opsin in other cells. To be sure, c-opsin1 should be removed from the larvae. Since, c-opsin1 seems to be the only relevant opsin

expressed by the ciliary photoreceptor cells, I expect that the larvae do not swim down anymore to UV-light if c-opsin1 is eliminated.

C-opsin1 can be eliminated by knocking it out with zinc-finger-nucleases (ZFNs), transcription activator-like effector nucleases (TALENs), or the CRISPR-Cas9 system (clustered regularly interspaced short palindromic repeat; Chandrasegaran and Carroll, 2015). Alternatively, *c-opsin1* can be knocked down with morpholinos. However, morpholinos were not useful in my hands, because for my experiments, I needed more larvae than I could realistically inject with the morpholinos. Additionally, even uninjected larvae tended to settle instead of swimming phototacticly in the cuvette. This may not be too surprising, since also the larvae of some barnacle species settle in grooves, which they sense mechanically (Crisp and Barnes, 1954; Knight-Jones and Crisp, 1953). These grooves may range for some species from 1 to 10 mm (Lemire and Bourget, 1996), which is similar to the containers I used. Additionally, the larvae may have been damaged by the injection and so do not show their natural behavior.

Therefore, I do not recommend using morpholinos for behavior experiments that require many larvae. For experiments, however, that only require a few larvae, morpholinos are useful (Conzelmann et al., 2013b). Morpholinos can also be used in calcium imaging experiments, since they also require just a few larvae (Gühmann et al., 2015; Randel et al., 2014; Tosches, 2013; Tosches et al., 2014; Verasztó et al., 2017; Williams et al., 2015). In calcium imaging experiments, I expect that *c-opsin1* morpholinos abolish UV-light induced calcium signals from the ciliary photoreceptor cells.

For experiments that require many larvae, I recommend gene-knockout, since only a knockout line can produce many healthy larvae. With a knockout line, the adults can also be checked for a phenotype, because a knockout does not rely on morpholinos that dilute out during development. For *c-opsin1* knockout larvae, I expect that they do not show the UV-response, because c-opsin1 is the only opsin expressed in the ciliary photoreceptor cells with a matching absorption spectrum.

Interesting will be the depth, to which the depth gauge brings the larvae in nature; however, this can only be shown by experiments in the field or in a deep swimming pool. Probably, the depth will differ between wild type larvae and my *Go-opsin1* knockout larvae. The depth-gauge and so the UV-response may be widespread across polychaete larvae, because the larvae of many polychaetes have ciliary photoreceptor cells (Arendt et al., 2004; Hausen, 2007; Purschke, 2005).

The depth gauge between species may only differ in the target depth. The depth gauge may be tuned by the opsins expressed by the ciliary photoreceptor cells and the eyes. If, for instance, the eyes only express one type of opsin that senses maximally cyan light (e.g. 480 nm), then the fraction of the spectrum sensed is smaller and the UV-light in the ratio increases, so that the depth gauge would be tuned to deeper water. The depth gauge could be tuned by other factors too, like the relative opsin expression between the eyes and the ciliary photoreceptor cells. The ciliary photoreceptor cells could also be differently connected with the eyes, so that the relative input from both components is altered. Such differences could be reflected in the morphology of the ciliary photoreceptor cells, which differs between species (Purschke, 2005).

Since, the ciliary photoreceptor cells are common among polychaete larvae, the UV-response and the depth gauge could be ancestral features of polychaetes. In fact, such a depth gauge is useful for all marine larvae of species with a pelagic-benthic live cycle, as they all need to descent from the open water column to the bottom of the sea, and all have the problem with phototaxis to select a certain site.

Since, the ciliary photoreceptor cells resemble molecularly the vertebrate photoreceptor cells of the retina and the pineal organs (Tosches, 2013), the urbilaterian, the last common ancestor of all bilateral symmetrical animals, may have had already ciliary photoreceptor cells. Whether they already had the UV-avoidance function, depends on the life style of the urbilaterian. If the urbilaterian had a pelagic-benthic life-style with a larva, which is commonly assumed (Budd and Jensen, 2000), then such a depth gauge would have been useful. If the urbilaterian were holobenthic then the depth gauge would not have been useful, because the urbilaterian would have been on the bottom of the sea anyway. If the urbilaterian were holopelagic then the depth gauge could have kept it at a certain depth during the day and could have served as a mechanism for diel vertical migration.

Phototaxis in *Platynereis dumerilii* is also mediated by Go-opsin1. Go-opsin1 seems to have been recruited by the rhabdomeric photoreceptor cells. Therefore, the ancestral state of Go-opsins is hard to determine. Go-opsins may be associated with a certain cell type that is not a rhabdomeric photoreceptor cell. Go-opsins may thus preferentially couple to a certain type of G-proteins. Or they may be very promiscuous and found in many different cell types so that it will be hard to reconstruct what was their ancestral function. However, to answer these

questions, more Go-opsins must be characterized by their expression, their G-protein binding partner, and their function.

4.4 Beyond phototaxis and opsins

Opsins and phototaxis are not the only things to study in *Platynereis dumerilii*: Two questions occurred in the discussion. One is: How do the larvae know where is down? This question is about sensing gravity. Gravity can influence unicellular organisms and planktonic larvae by some passive mechanisms. But how it works exactly in the trochophore and nectochaete larva of *Platynereis dumerilii* is unknown; especially whether passive mechanisms are enough or whether gravity is also sensed and the body is actively steered towards the desired direction.

The other question is: What switches the sign of phototaxis in the larvae? I observed that nectochaete larvae of *Platynereis dumerilii* could be positively or negatively phototactic. What switches the sign and how the nervous system controls the switch, is unknown. The larvae seem to react on mechanical cues, but other cues are possible for instance the sign of phototaxis can be switched by chemical cues in some hydroid larvae. Here the cues are linked to settlement, so that in principle all kinds of settlement cues could switch the sign of phototaxis.

4.4.1 What switches the sign of phototaxis in the larvae?

What exactly makes *Platynereis dumerilii* nectochaete larvae switch from positive to negative phototaxis, is unknown. The nauplius larva of the barnacle *Balanus perforatus* switches between negative and positive phototaxis depending on wavelength, light intensity, temperature, and oxygen, salt and ion concentrations (Ewald, 1912). Also, the larva of the polychaete *Pseudopolydora pulchra* switches from negative to positive phototaxis when the salinity is increased above that of natural seawater (Ranade, 1957). And the larva of the crab *Rhithropanopeus harrisi* switches between positive and negative phototaxis depending on the light intensity (Forward, 1974).

I also tested nectochaete larvae in a high intensity horizontal setup (Gühmann et al., 2015), weather they were switched between positive and negative phototaxis by temperature. Most of the larvae were negatively phototactic at 20 °C, 10 °C, and 4 °C (data not shown). In the same setup, I also dimmed the light with neutral density filters from OD 0 to OD 4 the larvae did not change the sign of phototaxis at any wavelength I tested (data not shown). Instead, I had in that setup batches of larvae that were either positively or negatively phototactic. And in the setup with the beaker, some batches were positively phototactic, but

switched during the experiment. The larvae did not switch depending on the wavelength, but to my impression after prolonged contact with the bottom of the beaker.

In the beaker, there were three kinds of nectochaete larvae: Larvae that were negatively phototactic from the beginning, larvae that became negatively phototactic during the experiment, and those that stayed positively phototactic. The larvae that were negatively phototactic swam very closely above the bottom of the beaker. The larvae that were initially positively phototactic were initially swimming in the full water column of the beaker, but during the experiment, they swam down induced by UV-light and when they were swimming for a while at the bottom, they became negatively phototactic. The larvae that stayed positively phototactic did not move to the bottom, but stayed in the water column.

In the horizontal column, nectochaete larvae were always positively phototactic; they only swam down when they were exposed to UV-light or when the light came from the bottom. However, when I put them into a vertical cuvette (10 mm x 10 mm x 40 mm), these larvae went down to the bottom and stayed there. I could only push them up via negative phototaxis when a diode emitted light from the bottom (data not shown). Weather they are positively or negatively phototactic seems to depend on the size of the container and how close they are to the bottom. They seem to need to contact the bottom and the sides of the container to switch the sign of phototaxis.

Thus, the switch could be triggered by a mechanical stimulus. In principle, mechanical stimuli can be settlement cues, larvae can sense different surface textures (Price, 2010): The mussel *Mytilus edulis* prefers to settle on rough surfaces (Petraitis, 1990). The barnacle *Chthamalus anisopoma* prefers granite over basalt. However, if the granite is cut and has a smooth artificial surface, it is not attractive anymore (Raimondi, 1990). The barnacle species *Balanus balanoides*, *Balanus crenatus*, and *Elminius modestus* prefer to settle in grooves, which they sense mechanically (Crisp and Barnes, 1954; Knight-Jones and Crisp, 1953). The preferred groove diameters range from 1 to 10 mm for some species (Lemire and Bourget, 1996). This is like the cuvette (10 mm x 10 mm x 42 mm) I used to determine the spectrum of the UV-response. The cuvette resembles a 10 mm diameter hole with a depth of 42 mm, in which *Platynereis dumerilii* nectochaete larvae seem to settle readily. These nectochaete larvae also settle at the bevel of 100 ml glass beakers. The larvae prefer the bevel at the rim, while

the center of the beaker is empty (personal observation). Here the larvae may also sense the bevel, mechanically.

A mechanical stimulus could be created by wall drag (Winet, 1973). Wall drag arises when a fluid moves relatively to a surface and the fluid does not slip from the surface so that a boundary layer forms between the surface and the free-stream flow. The flow within the boundary layer is laminar; this means it is free of turbulences. Wall drag arises either if an object moves along a wall or if a wall moves past an object. Wall drag influences an object up to a distance *Y*, which can be calculated with this formula:

$$Y > 20\nu/U$$

U is the velocity of the wall or the object, and ν is the kinematic viscosity of the fluid, which is $1x10^{-6}m^{-2}s^{-1}$ for water (Loudon et al., 1994). The velocity for *Platynereis dumerilii* metatrochophore larvae was around 1.4 mm/s; and was for nectochaete larvae swimming up 1.1 mm/s (data not shown). These velocities yield roughly 14 mm and 18 mm for *Y*, respectively. This means the larvae can feel a wall or a bottom without touching it in the vertical cuvette (10 mm x 10 mm x 42 mm), the horizontal cuvette (20 mm x 9 mm x 5 mm), the column (32 mm x 10 mm x 160 mm), and the 100 ml glass beaker (48 mm diameter, 42 mm water-height or filled with 65 ml). The larvae may feel the different flows caused by wall drag mechanically via their cilia (Khayyeri et al., 2015) or chaetae (Loudon et al., 1994; Merz and Woodin, 2006; Woodin et al., 2003). Since the larvae were positively phototactic in the vertical column, they may only switch to negative phototaxis when they feel the bottom. In practice, a container that avoids wall drag and meets all the experimental constraints is difficult to find (Mann et al., 1991). The container must be illuminated so that the larvae cannot only be stimulated, but also be recorded by a camera that needs the larvae also in its focus and field of view. This rather favors small containers.

Besides mechanical cues, also chemical cues could make the larvae switch to negative phototaxis. However, the cuvettes and the beakers consist of glass, which is chemically inert. And so, should not release any chemicals. The cuvettes where just rinsed after each use and so could have contained chemicals or bacteria that were introduced by the larvae from the experiment before. Also, the glass beakers, even so, they were autoclaved, may have contained chemicals and bacteria. The glass beakers did not only house the larvae during the experiments, but also during their whole life from fertilization on, when their

parents released eggs and sperm there, including their bacteria. So, even if everything were sterile until then, the bacteria contaminated the beakers, and might have produced settlement cues. This cannot be fixed by changing the beakers, since the bacteria and the substances are in the water, too.

Substances can indeed serve as settlement cues and change the sign of phototaxis. Positive phototaxis is shown by the larvae of the hydroid *Clava multicornis*, when they are crawling over inert surfaces. However, they become photonegative, when they are crawling over the surface of the brown alga *Ascophyllum nodosum*, their natural settling substrate (Williams, 1965). The sinistral spiral tubeworm *Spirorbis borealis* settles on another brown alga: *Fucus serratus*. It also settles on surfaces that are coated with a *Fucus serratus* extract. However, it does not react to the extract if the extract is just dissolved in the seawater. But when it contacts the extract on a surface, it also becomes photonegative (Williams, 1964).

Not all larvae have to contact their substrate: The larvae of the nudibranch mollusc *Phestilla sibogae* settle and metamorphose when they sense substances dissolved in the water from corals of the genus *Porites*, which the adults prey on (Hadfield and Pennington, 1990). In other species, substances also from conspecific adults like chemicals, pheromones, and peptides may indicate a good settlement site or even chemicals from predators if these indicate a good habitat (Pawlik, 1992; Rittschof et al., 1998; Rodriguez et al., 1993). Another chemical cue are unsaturated free fatty acids, which are found in the tubes of the reef-building tubeworm *Phragmatopoma californica*. When its larvae contact these fatty acids in the tubes, they settle and metamorphose (Pawlik, 1986). Other settlement cues are physical factors: Beside light, contour, and texture, these are gravity, pressure, temperature, salinity (Pawlik, 1992), and vibrations from waves (Rittschof et al., 1998).

Also, bacterial biofilms can be used as cues, the biofilm compositions differ between subtidal and intertidal zones, and depending on the preferences, a larva may accept or reject a biofilm from one of these zones (Hadfield, 2011). Other negative settlement cues exist: Larvae and juveniles of infaunal species avoid sediments that are contaminated by bromophenols secreted by capitellid polychaetes (Woodin, 1991; Woodin et al., 1997). Many other species avoid settling near competitive species, too, so that probably many chemical substances exist including metabolites that indicate competitors (Pawlik, 1992) or even predators (Welch et al., 1997). Odors of visual predators activate

negative phototaxis in diel-vertically migrating brine shrimp (*Artemia franciscana*) nauplii (McKelvey and Forward, 1995).

These settlement cues may vary between species and the list here presented is not exhaustive. Among these cues are pits, grooves, and holes, which can be felt mechanically, even without touch. And settlement cues exist that can induce negative phototaxis. Therefore, negative phototaxis may be induced in *Platynereis dumerilii* larvae, because they feel to be in a small compartment. In principle, the larvae could become photonegative on any positive settlement cue, so that they swim to the substrate, the site where the light does not come from. And the larvae could become photopositive on any negative settlement cue so that they swim to the light and away from the rejected substrate.

4.4.2 How do the larvae know where is down?

Platynereis dumerilii metatrochophore and nectochaete larvae swim down when they are exposed to UV-light. How do they know where is down? The UV-response here is a UV-induced positive gravitaxis. How gravitaxis works in *Platynereis dumerilii* larvae, is unknown. As far as we know, the larvae have no statocysts or statoliths. This is also true for many other planktonic larvae (Chan, 2012), even so exceptions exists, for instance the tadpole larva of *Ciona intestinalis* has a pigment cell that works as a statocyte (Tsuda et al., 2003). In general, invertebrate larvae (Chan, 2012) and unicellular microorganisms are denser than their surrounding medium but still swim up preferentially (Häder, 1999). Small organisms can use gravity-buoyancy, drag-gravity, propulsion-gravity, or the special statocyst to find the direction of gravity.

The special statocysts is relevant for unicellular organisms (Häder, 1997) like the giant ciliate *Bursaria truncatella* where the cell mass presses onto the bottom cell membrane and stretch-activates ion channels, there (Krause and Braucker, 2009).

Propulsion-gravity predicts that negative gravitaxis (up-swimming) arises from spiral swimming (axial gyration) if the body is propelled anterior of the center of mass. Spiral swimming is shown by many marine larvae (Chia et al., 1984; Winet and Jahn, 1974) including *Platynereis dumerilii* (Jékely et al., 2008).

Gravity-buoyancy is caused by unequal density distribution within a body. If the tail is denser than the head, the tail is less buoyant and sinks faster so that an animal sinks with the tail first to the bottom (Mogami et al., 2001). This way the animal has only to swim forward to swim up. The density in nectochaete larvae is unequally distributed, because they have lipid droplets (Chia et al., 1984).

Drag-gravity is caused by asymmetry between head and tail in a uniform-dense body. The thicker end of a body sinks faster than the thinner end, so that the body sinks with the thicker end down (Mogami et al., 2001). One end of the body can become thinner by adding protruding cilia to the apical tuft as in some trochophore larvae or adding setae like in nectochaete larvae (Chia et al., 1984).
It can be tested whether an organism uses gravity-buoyancy or drag-gravity by comparing its behavior in hypo- and hyper-dense medium. If the organism turns up with the same side while it is sinking in hyper-dense and floating in hypo-dense medium, then it uses gravity-buoyancy otherwise it uses drag-gravity. The organism can also be paralyzed; however, it should not change its shape, otherwise extra drag could be added. For instance, cells of *Paramecium caudatum* and the gastrulae of the sea urchin *Hemicentrotus pulcherrimus* orientate up while sinking in hypo-dense medium, but orientate down while they are floating in hyper-dense medium. However, *Hemicentrotus pulcherrimus* pluteus larvae orient up while they are sinking or floating in hypo- and hyper-dense medium, respectively. Therefore, *Hemicentrotus pulcherrimus* changes its orientation mode during its development from drag-gravity to gravity-buoyancy (Mogami et al., 2001).
Hemicentrotus pulcherrimus pluteus larvae do not only orient passively, but also seem to sense their orientation as they swim with equal speed in all directions, and so would adjust their speed depending on gravity (Mogami et al., 1988). However, there could still be a passive mechanism behind it. In comparison, *Platynereis dumerilii* trochophore and metatrochophore larvae are as fast as they swim up phototacticly to cyan (480 nm) light or swim down to non-directional UV (395 nm) light, but nectochaete larvae are slower if they swim up to cyan light than if they swim down to UV-light (data not shown). Living metatrochophore larvae orient up while they are sinking (Conzelmann et al., 2011), possibly because the head is more buoyant since it contains lipid yolk droplets, which are anterior of the prototroch. The prototroch itself lies anterior of the midline and consists of the cilia that propel the body (Fischer et al., 2010).
Therefore, gravity-buoyancy could orient the larvae passively upward; also, propulsion-gravity if all the cilia are beating equally. However, these mechanisms make the larvae swimming up but not swimming down. For swimming down the larvae need to turn and then stay on track. To turn down, the larvae could slow down (or speed up) the beating of some cilia as they do when they are steering towards the light (Jékely et al., 2008). There may be even

a way to activate the cilia in a way that the larvae may swim down automatically, so that the larvae do not even need to know where down is to swim down.
However, if no such ciliary activity pattern exists then the larvae need to know how much they are tilted compared to where is down. The tilt could be indicated by buoyancy. Buoyancy rotates the larvae until they are pointing upwards, unless the larvae steer into the opposite direction. The rotation creates a flow around the body. The flow has a direction, which could be measured by mechano-sensitive cilia on the body surface. If the surface is without flow caused by buoyancy, then the larvae can steer in any direction they want to get down. They need only to keep the direction against the buoyancy rotation until they turned by more than 180°. Then, they start to rotate into the other direction and must steer against it. This predicts that the larvae can sink head up in a straight line, but cannot swim down in a straight line with head down without further cues. In fact, larvae can be made swimming down by treating them with MIP (myoinhibitory peptide). These larvae do not swim down in a straight line but in a zigzag line (Conzelmann et al., 2013b).
Therefore, gravity-buoyancy is a plausible mechanism to let the larvae know where is up and thus also where is down.

4.5 Conclusion

Here, I found a new type of light induced behavior, which is as fast as phototaxis. It is a positive geotaxis induced by UV-light coming from all sides. The light may be detected by c-opsin1 expressed by the ciliary photoreceptor cells. The positive gravitaxis works against positive phototaxis and both form this way a ratio-chromatic depth gauge, which could be common among invertebrate larvae. The depth gauge helps the larvae to find the right depth for settling on a global level, while positive and negative phototaxis helps the larvae to select a local settlement site, which gives some shelter.
Phototaxis is mediated by Go-opsin1 and other rhabdomeric opsins. Go-opsin1 seems to couple to a Gq-protein in the rhabdomeric photoreceptor cells. This is surprising from a textbook point of view, but many examples exist where phototransduction cascades may differ and that depends rather on the host cell then phylogeny. Since Go-opsins seem to bind only in one example to a Go-protein, it remains to be seen whether Go-opsins indeed deserve the name Go-opsins.

5 Contributions

My previous publication (Gühmann et al., 2015) contributed to this thesis, I included all my contributions I regarded as relevant from it. And I cite it for the things I did not include here or others have contributed. Except I included the contribution I regard as important, which was the Go-opsin spectrum, contributed by Huiyong Jia and Shozo Yokoyama, who also contributed the c-opsin1 spectrum.

6 List of Figures

7 List of Tables

8 List of Program Codes

9 List of Abbreviations

AE ..... adult eye
AMP ..... adenosine monophosphate
BLAST ..... basic local alignment search tool
cAMP ..... cyclic AMP
cDNA ..... complementary DNA
cGMP ..... cyclic GMP
CNG ..... cyclic nucleotide-gated (cation channel)
c-opsin ..... ciliary opsin
CRISPR ..... clustered regularly interspaced short palindromic repeat
DAG ..... diacylglycerol
DNA ..... deoxyribonucleic acid
dpf ..... days post fertilization
EST ..... expressed sequence tag
GMP ..... guanine monophosphate
GPCR ..... G-protein coupled receptor
G-protein ..... guanine-nucleotide-binding-protein
hpf ..... hours post fertilization
$InsP_3$ inositol-1,4,5-trisphosphate
LE ..... larval eye
LED ..... light-emitting diode
LWS ..... long wavelength sensitive (opsin)
mRNA ..... messenger RNA
MWS ..... middle wavelength sensitive (opsin)
NMD ..... nonsense mediated decay
OD ..... optical density
PCR ..... polymerase chain reaction
PDE ..... phosphodiesterase
PIP_2 ..... phosphatidylinositol-4,5-bisphosphate
PLC ..... phospholipase C
RGR-opsin ..... retinal G-protein-coupled receptor
RH2b ..... rhodopsin like 2b
RNA ..... ribonucleic acid
r-opsin ..... rhabdomeric opsin

RPE ..... retinal pigment epithelium
SWS1 ..... short wavelength sensitive 1 (opsin)
TALEN ..... activator-like effector nuclease
TRP ..... transient receptor potential (channel)
TRPA1 ..... TRP (channel) A1
TRPC ..... Canonical TRP (channel)
TRPL ..... TRP like (channel)
UV ..... ultraviolet
ZFN ..... zinc-finger-nuclease

10 Literature

Achim, K., Pettit, J.B., Saraiva, L.R., Gavriouchkina, D., Larsson, T., Arendt, D., and Marioni, J.C. (2015). High-throughput spatial mapping of single-cell RNA-seq data to tissue of origin. Nature Biotechnology *33*, 503-U215.

Aliani, S., and Meloni, R. (1999). Dispersal strategies of benthic species and water current variability in the Corsica Channel (Western Mediterranean). Sci Mar *63*, 137-145.

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. (1990). Basic Local Alignment Search Tool. Journal of Molecular Biology *215*, 403-410.

Amora, T.L., Ramos, L.S., Galan, J.F., and Birge, R.R. (2008). Spectral tuning of deep red cone pigments. Biochemistry-Us *47*, 4614-4620.

Arendt, D., Musser, J.M., Baker, C.V., Bergman, A., Cepko, C., Erwin, D.H., Pavlicev, M., Schlosser, G., Widder, S., Laubichler, M.D., and Wagner, G.P. (2016). The origin and evolution of cell types. Nat Rev Genet *17*, 744-757.

Arendt, D., Technau, U., and Wittbrodt, J. (2001). Evolution of the bilaterian larval foregut. Nature *409*, 81-85.

Arendt, D., Tessmar-Raible, K., Snyman, H., Dorresteijn, A.W., and Wittbrodt, J. (2004). Ciliary photoreceptors with a vertebrate-type opsin in an invertebrate brain. Science (New York, N.Y.) *306*, 869-871.

Arendt, D., and Wittbrodt, J. (2001). Reconstructing the eyes of Urbilateria. Philos T Roy Soc B *356*, 1545-1563.

Arshavsky, V.Y., Lamb, T.D., and Pugh, E.N. (2002). G proteins and phototransduction. Annu Rev Physiol *64*, 153-187.

Asadulina, A., Conzelmann, M., Williams, E.A., Panzera, A., and Jékely, G. (2015). Object-based representation and analysis of light and electron microscopic volume data using Blender. Bmc Bioinformatics *16*.

Asadulina, A., Panzera, A., Verasztó, C., Liebig, C., and Jékely, G. (2012). Whole-body gene expression pattern registration in Platynereis larvae. Evodevo *3*.

Asenjo, A.B., Rim, J., and Oprian, D.D. (1994). Molecular Determinants of Human Red/Green Color Discrimination. Neuron *12*, 1131-1138.

Audouin, J.V., and Milne-Edwards, H. (1834). Néréide de Dumeril. Nereis Dumerilii. Recherches pour servir a l'histoire naturelle du littoral de la France, ou, Recueil de mémoires sur l'anatomie, la physiologie, la

classification et les moeurs des animaux des nos côtes : ouvrage accompagné de planches faites d'après nature *2*, 196-199.
Backfisch, B., Kozin, V.V., Kirchmaier, S., Tessmar-Raible, K., and Raible, F. (2014). Tools for gene-regulatory analyses in the marine annelid Platynereis dumerilii. Plos One *9*, e93076.
Backfisch, B., Veedin Rajan, V.B., Fischer, R.M., Lohs, C., Arboleda, E., Tessmar-Raible, K., and Raible, F. (2013). Stable transgenesis in the marine annelid Platynereis dumerilii sheds new light on photoreceptor evolution. Proc Natl Acad Sci U S A *110*, 193-198.
Bailes, H.J., and Lucas, R.J. (2013). Human melanopsin forms a pigment maximally sensitive to blue light (λmax ≈ 479 nm) supporting activation of G(q/11) and G(i/o) signalling cascades. Proc Biol Sci *280*, 20122987.
Ball, S., Goodwin, T.W., and Morton, R.A. (1946). Retinene1-vitamin A aldehyde. Biochem J *40*, lix.
Ball, S., Goodwin, T.W., and Morton, R.A. (1948). Studies on vitamin A: 5. The preparation of retinene(1)-vitamin A aldehyde. Biochem J *42*, 516-523.
Ban, E., Kasai, A., Sato, M., Yokozeki, A., Hisatomi, O., and Oshima, N. (2005). The signaling pathway in photoresponses that may be mediated by visual pigments in erythrophores of Nile tilapia. Pigm Cell Res *18*, 360-369.
Bannister, S., Antonova, O., Polo, A., Lohs, C., Hallay, N., Valinciute, A., Raible, F., and Tessmar-Raible, K. (2014). TALENs mediate efficient and heritable mutation of endogenous genes in the marine annelid Platynereis dumerilii. Genetics *197*, 77-89.
Barber, V.C., Evans, E.M., and Land, M.F. (1967). The fine structure of the eye of the mollusc Pecten maximus. Z Zellforsch Mikrosk Anat *76*, 25-312.
Bass, B.L. (2002). RNA editing by adenosine deaminases that act on RNA. Annual review of biochemistry *71*, 817-846.
Bauknecht, P. (2013). Heterologous Expression and Spectral Characterization of Platynereis dumerilii Opsins. (Tübingen, Eberhard Karls University, Tübingen).
Bedford, A.P., and Moore, P.G. (1984). Macrofaunal Involvement in the Sublittoral Decay of Kelp Debris - the Detritivore Community and Species Interactions. Estuar Coast Shelf S *18*, 97-111.
Bedford, A.P., and Moore, P.G. (1985). Macrofaunal Involvement in the Sublittoral Decay of Kelp Debris - the Polychaete Platynereis-Dumerilii (Audouin and Milne-Edwards) (Annelida, Polychaeta). Estuar Coast Shelf S *20*, 117-134.
Bellan, G. (1980). Relationship of Pollution to Rocky Substratum Polychaetes on the French Mediterranean Coast. Mar Pollut Bull *11*, 318-321.
Bellono, N.W., Najera, J.A., and Oancea, E. (2014). UV light activates a G alpha(q/11)-coupled phototransduction pathway in human melanocytes. Journal of General Physiology *143*, 203-214.
Bellono, N.W., and Oancea, E. (2013). UV light phototransduction depolarizes human melanocytes. Channels *7*, 243-248.
Berson, D.M., Dunn, F.A., and Takao, M. (2002). Phototransduction by retinal ganglion cells that set the circadian clock. Science (New York, N.Y.) *295*, 1070-1073.
Birnbaumer, L. (2007). Expansion of signal transduction by G proteins - The second 15 years or so: From 3 to 16 alpha subunits plus beta gamma dimers. Bba-Biomembranes *1768*, 772-793.

Blackshaw, S., and Snyder, S.H. (1997). Parapinopsin, a novel catfish opsin localized to the parapineal organ, defines a new gene family. Journal of Neuroscience *17*, 8083-8092.

Blackshaw, S., and Snyder, S.H. (1999). Encephalopsin: A novel mammalian extraretinal opsin discretely localized in the brain. Journal of Neuroscience *19*, 3681-3690.

Boll, F. (1876). Zur Anatomie und Physiologie der Retina. Monatsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin aus dem Jahre 1876, 783-787.

Boll, F. (1877). Zur Anatomie und Physiologie der Retina. Aus dem Laboratorium für vergleichende Anatomie und Physiologie zu Rom, Achte Mittheilung. Archiv für Anatomie und Physiologie, Physiologische Abtheilung, 4-35.

Bownds, D. (1967). Site of attachment of retinal in rhodopsin. Nature *216*, 1178-1181.

Brown, P.K., and Wald, G. (1956). The neo-b isomer of vitamin A and retinene. J Biol Chem *222*, 865-877.

Budd, G.E., and Jensen, S. (2000). A critical reappraisal of the fossil record of the bilaterian phyla. Biol Rev *75*, 253-295.

Cao, P.X., Sun, W.Y., Kramp, K., Zheng, M.H., Salom, D., Jastrzebska, B., Jin, H., Palczewski, K., and Feng, Z.Y. (2012). Light-sensitive coupling of rhodopsin and melanopsin to G(i/o) and G(q) signal transduction in Caenorhabditis elegans. Faseb J *26*, 480-491.

Chan, K.Y.K. (2012). Biomechanics of Larval Morphology Affect Swimming: Insights from the Sand Dollars Dendraster excentricus. Integrative and Comparative Biology *52*, 458-469.

Chandrasegaran, S., and Carroll, D. (2015). Origins of Programmable Nucleases for Genome Engineering. J Mol Biol.

Chen, S.C., Robertson, R.M., and Hawryshyn, C.W. (2013). Possible Involvement of Cone Opsins in Distinct Photoresponses of Intrinsically Photosensitive Dermal Chromatophores in Tilapia Oreochromis niloticus. Plos One *8*.

Chen, S.C., Xiao, C.F., Troje, N.F., Robertson, R.M., and Hawryshyn, C.W. (2015). Functional characterisation of the chromatically antagonistic photosensitive mechanism of erythrophores in the tilapia Oreochromis niloticus. J Exp Biol *218*, 748-756.

Chew, K.S., Schmidt, T.M., Rupp, A.C., Kofuji, P., and Trimarchi, J.M. (2014). Loss of G(q11) Genes Does Not Abolish Melanopsin Phototransduction. Plos One *9*.

Chia, F.S., Bucklandnicks, J., and Young, C.M. (1984). Locomotion of Marine Invertebrate Larvae - a Review. Can J Zool *62*, 1205-1222.

Choe, H.W., Kim, Y.J., Park, J.H., Morizumi, T., Pai, E.F., Krauss, N., Hofmann, K.P., Scheerer, P., and Ernst, O.P. (2011). Crystal structure of metarhodopsin II. Nature *471*, 651-655.

Christensen, A.P., and Corey, D.P. (2007). TRP channels in mechanosensation: direct or indirect activation? Nat Rev Neurosci *8*, 510-521.

Clark, R.B., and Milne, A. (1955). The Sublittoral Fauna of 2 Sandy Bays on the Isle of Cumbrae, Firth of Clyde. J Mar Biol Assoc Uk *34*, 161-180.

Cohen, J.H., and Forward, R.B. (2002). Spectral sensitivity of vertically migrating marine copepods. Biol Bull *203*, 307-314.

Collins, F.D. (1953). Rhodopsin and indicator yellow. Nature *171*, 469-471.

Conzelmann, M., Offenburger, S.L., Asadulina, A., Keller, T., Munch, T.A., and Jékely, G. (2011). Neuropeptides regulate swimming depth of Platynereis larvae. Proc Natl Acad Sci U S A *108*, E1174-1183.

Conzelmann, M., Williams, E.A., Krug, K., Franz-Wachtel, M., Macek, B., and Jékely, G. (2013a). The neuropeptide complement of the marine annelid Platynereis dumerilii. Bmc Genomics *14*.

Conzelmann, M., Williams, E.A., Tunaru, S., Randel, N., Shahidi, R., Asadulina, A., Berger, J., Offermanns, S., and Jékely, G. (2013b). Conserved MIP receptor-ligand pair regulates Platynereis larval settlement. Proc Natl Acad Sci U S A *110*, 8224-8229.

Cornwall, M.C., and Gorman, A.L. (1983). The cation selectivity and voltage dependence of the light-activated potassium conductance in scallop distal photoreceptor. J Physiol *340*, 287-305.

Cram, A., and Evans, S.M. (1980). Stability and Lability in the Evolution of Behavior in Nereid Polychaetes. Anim Behav *28*, 483-490.

Crisp, D.J., and Barnes, H. (1954). The Orientation and Distribution of Barnacles at Settlement with Particular Reference to Surface Contour. J Anim Ecol *23*, 142-162.

Cronin, T.W., and Porter, M.L. (2014). The Evolution of Invertebrate Photopigments and Photoreceptors. In Evolution of Visual and Non-visual Pigments, pp. 105-135.

Dakin, W.J. (1928). The eyes of Pecten, Spondylus, Amussium and allied Lamellibranchs, with a short discussion on their Evolution. P R Soc Lond B-Conta *103*, 355-365.

Daly, J.M. (1973). Behavioral and Secretory Activity during Tube Construction by Patynereis-Dumerilii Aud and M Edw [Polychaeta - Nereidae]. J Mar Biol Assoc Uk *53*, 521-529.

Davies, W.I.L., Zheng, L., Hughes, S., Tamai, T.K., Turton, M., Halford, S., Foster, R.G., Whitmore, D., and Hankins, M.W. (2011). Functional diversity of melanopsins and their global expression in the teleost retina. Cell Mol Life Sci *68*, 4115-4132.

de Mendoza, A., Sebe-Pedros, A., and Ruiz-Trillo, I. (2014). The Evolution of the GPCR Signaling System in Eukaryotes: Modularity, Conservation, and the Transition to Metazoan Multicellularity. Genome Biology and Evolution *6*, 606-619.

Delroisse, J., Ullrich-Luter, E., Ortega-Martinez, O., Dupont, S., Arnone, M.I., Mallefet, J., and Flammang, P. (2014). High opsin diversity in a non-visual infaunal brittle star. BMC Genomics *15*, 1035.

Dkhissi-Benyabya, O., Rieux, C., Hut, R.A., and Cooper, H.M. (2006). Immunohistochemical evidence of a melanopsin cone in human retina. Invest Ophth Vis Sci *47*, 1636-1641.

Donahue, W.F., and Schindler, D.W. (1998). Diel emigration and colonization responses of blackfly larvae (Diptera : Simuliidae) to ultraviolet radiation. Freshwater Biol *40*, 357-365.

Dorresteijn, A.W.C. (1990). Quantitative-Analysis of Cellular-Differentiation during Early Embryogenesis of Platynereis-Dumerilii. Roux Arch Dev Biol *199*, 14-30.

Dorresteijn, A.W.C., Ogrady, B., Fischer, A., Porchethennere, E., and Boillymarer, Y. (1993). Molecular Specification of Cell-Lines in the Embryo of Platynereis (Annelida). Roux Arch Dev Biol *202*, 260-269.

Doyon, Y., McCammon, J.M., Miller, J.C., Faraji, F., Ngo, C., Katibah, G.E., Amora, R., Hocking, T.D., Zhang, L., Rebar, E.J., Gregory, P.D., Urnov, F.D., and Amacher, S.L. (2008). Heritable targeted gene disruption in zebrafish using designed zinc-finger nucleases. Nat Biotechnol *26*, 702-708.

Dray, N., Tessmar-Raible, K., Le Gouar, M., Vibert, L., Christodoulou, F., Schipany, K., Guillou, A., Zantke, J., Snyman, H., Behague, J., Vervoort, M., Arendt, D., and Balavoine, G. (2010). Hedgehog Signaling Regulates Segment Formation in the Annelid Platynereis. Science (New York, N.Y.) *329*, 339-342.

Ebert, D. (2005). Introduction to Daphnia Biology.In Ecology, Epidemiology, and Evolution of Parasitism in Daphnia [Internet], D. Ebert, ed. (Bethesda (MD), National Library of Medicine (US), National Center for Biotechnology).

Eisen, J.S., and Smith, J.C. (2008). Controlling morpholino experiments: don't stop making antisense. Development *135*, 1735-1743.

Ewald, A., and Kühne, W. (1878). Untersuchungen über den Sehpurpur. Untersuchungen aus dem Physiologischen Institute der Universität Heidelberg *1*, 139-218.

Ewald, W.F. (1912). On artificial modification of light reactions and the influence of electrolytes on phototaxis. J Exp Zool *13*, 591-612.

Fasick, J.I., Lee, N., and Oprian, D.D. (1999). Spectral tuning in the human blue cone pigment. Biochemistry-Us *38*, 11593-11596.

Fauchald, K., and Jumars, P.A. (1992). Diet of Worms - a Citation-Classic Commentary on the Diet of Worms - a Study of Polychaete Feeding Guilds by Fauchald,K., and Jumars,P.A. Cc/Agr Biol Environ, 8-8.

Fauvel, P. (1914). Annélides polychètes non-pélagiques provenant des campagnes de l'Hirondelle et de la Princesse-Alice (1885-1910). Résultats des campagnes scientifiques accompliés par le Prince Albert I *46*, 193.

Feuda, R., Hamilton, S.C., McInerney, J.O., and Pisani, D. (2012). Metazoan opsin evolution reveals a simple route to animal vision. Proc Natl Acad Sci U S A *109*, 18868-18872.

Feuda, R., Marletaz, F., Bentley, M.A., and Holland, P.W. (2016). Conservation, Duplication, and Divergence of Five Opsin Genes in Insect Evolution. Genome Biol Evol *8*, 579-587.

Feuda, R., Rota-Stabelli, O., Oakley, T.H., and Pisani, D. (2014). The comb jelly opsins and the origins of animal phototransduction. Genome Biol Evol *6*, 1964-1971.

Filipek, S., Stenkamp, R.E., Teller, D.C., and Palczewski, K. (2003). G protein-coupled receptor rhodopsin: A prospectus. Annu Rev Physiol *65*, 851-879.

Fine, M.L. (1970). Faunal Variation on Pelagic Sargassum. Mar Biol *7*, 112-&.

Fischer, A., and Dorresteijn, A. (2004). The polychaete Platynereis dumerilli (Annelida): a laboratory animal with spiralian cleavage, lifelong segment proliferation and a mixed benthic/pelagic life cycle. Bioessays *26*, 314-325.

Fischer, A.H., and Arendt, D. (2013). Mesoteloblast-like mesodermal stem cells in the polychaete annelid Platynereis dumerilii (Nereididae). J Exp Zool B Mol Dev Evol *320*, 94-104.

Fischer, A.H., Henrich, T., and Arendt, D. (2010). The normal development of Platynereis dumerilii (Nereididae, Annelida). Front Zool *7*, 31.

Forward, R.B. (1974). Negative Phototaxis in Crustacean Larvae - Possible Functional Significance. J Exp Mar Biol Ecol *16*, 11-17.

Forward, R.B., and Cronin, T.W. (1979). Spectral Sensitivity of Larvae from Inter-Tidal Crustaceans. J Comp Physiol *133*, 311-315.

Foster, R.G., and Hankins, M.W. (2002). Non-rod, non-cone photoreception in the vertebrates. Prog Retin Eye Res *21*, 507-527.

Gambi, M.C., Lorenti, M., Russo, G.F., Scipione, M.B., and Zupo, V. (1992). Depth and Seasonal Distribution of Some Groups of the Vagile Fauna of the Posidonia-Oceanica Leaf Stratum - Structural and Trophic Analyses. Pszni Mar Ecol *13*, 17-39.

Giangrande, A. (1988). Polychaete Zonation and Its Relation to Algal Distribution down a Vertical Cliff in the Western Mediterranean (Italy) - a Structural-Analysis. J Exp Mar Biol Ecol *120*, 263-276.

Giangrande, A., Delos, A.L., Fraschetti, S., Musco, L., Licciano, M., and Terlizzi, A. (2003). Polychaete assemblages along a rocky shore on the South Adriatic coast (Mediterranean Sea): patterns of spatial distribution. Mar Biol *143*, 1109-1116.

Giménez, F., and Marín, A. (1991). Los Anelidos poliquetos de una solfatara submarina en el Golfo de Napoles. Anales de Biología *17*, 143-151.

Gomez, M., and Nasi, E. (1995). Activation of light-dependent K+ channels in ciliary invertebrate photoreceptors involves cGMP but not the IP3/Ca2+ cascade. Neuron *15*, 607-618.

Gomez, M.P., and Nasi, E. (1997). Antagonists of the cGMP-gated conductance of vertebrate rods block the photocurrent in scallop ciliary photoreceptors. J Physiol *500 (Pt 2)*, 367-378.

Gomez, M.P., and Nasi, E. (2000). Light transduction in invertebrate hyperpolarizing photoreceptors: possible involvement of a Go-regulated guanylate cyclase. The Journal of neuroscience : the official journal of the Society for Neuroscience *20*, 5254-5263.

Gorman, A.L., and McReynolds, J.S. (1978). Ionic effects on the membrane potential of hyperpolarizing photoreceptors in scallop retina. J Physiol *275*, 345-355.

Govardovskii, V.I., Fyhrquist, N., Reuter, T., Kuzmin, D.G., and Donner, K. (2000). In search of the visual pigment template. Visual Neurosci *17*, 509-528.

Gühmann, M., Jia, H., Randel, N., Verasztó, C., Bezares-Calderón, L.A., Michiels, N.K., Yokoyama, S., and Jékely, G. (2015). Spectral Tuning of Phototaxis by a Go-Opsin in the Rhabdomeric Eyes of Platynereis. Curr Biol *25*, 2265-2271.

Häder, D.P. (1997). Oben oder unten - Schwerkraftperzeption bei dem einzelligen Flagellaten Euglena gracilis. Mikrokosmos *86*, 351-356.

Häder, D.P. (1999). Gravitaxis in unicellular microorganisms. Adv Space Res-Series *24*, 843-850.

Hadfield, M.G. (2011). Biofilms and Marine Invertebrate Larvae: What Bacteria Produce That Larvae Use to Choose Settlement Sites. Annu Rev Mar Sci *3*, 453-+.

Hadfield, M.G., and Pennington, J.T. (1990). Nature of the Metamorphic Signal and Its Internal Transduction in Larvae of the Nudibranch Phestilla-Sibogae. B Mar Sci *46*, 455-464.

Halford, S., Freedman, M.S., Bellingham, J., Inglis, S.L., Poopalasundaram, S., Soni, B.G., Foster, R.G., and Hunt, D.M. (2001). Characterization of a novel human opsin gene with wide tissue expression and identification of embedded and flanking genes on chromosome 1q43. Genomics *72*, 203-208.

Haltaufderhyde, K., Ozdeslik, R.N., Wicks, N.L., Najera, J.A., and Oancea, E. (2015). Opsin Expression in Human Epidermal Skin. Photochemistry and Photobiology *91*, 117-123.

Hannibal, J., Hindersson, P., Knudsen, S.M., Georg, B., and Fahrenkrug, J. (2002). The photopigment melanopsin is exclusively present in pituitary adenylate cyclase-activating polypeptide-containing retinal ganglion cells of the retinohypothalamic tract. Journal of Neuroscience *22*.

Hanswillemenke, A., Kuzdere, T., Vogel, P., Jékely, G., and Stafforst, T. (2015). Site-Directed RNA Editing in Vivo Can Be Triggered by the Light-Driven Assembly of an Artificial Riboprotein. J Am Chem Soc *137*, 15875-15881.

Hardie, R.C. (2012). Phototransduction mechanisms in *Drosophila* microvillar photoreceptors. WIREs Membr Transp Signal *1*, 162–187.

Hardie, R.C., and Franze, K. (2012). Photomechanical responses in Drosophila photoreceptors. Science (New York, N.Y.) *338*, 260-263.

Hardie, R.C., and Juusola, M. (2015). Phototransduction in Drosophila. Curr Opin Neurobiol *34*, 37-45.

Hargrave, P.A. (2001). Rhodopsin structure, function, and topography - The Friedenwald Lecture. Invest Ophth Vis Sci *42*, 3-9.

Hargrave, P.A., Mcdowell, J.H., Curtis, D.R., Wang, J.K., Juszczak, E., Fong, S.L., Rao, J.K.M., and Argos, P. (1983). The Structure of Bovine Rhodopsin. Biophys Struct Mech *9*, 235-244.

Hattar, S., Liao, H.W., Takao, M., Berson, D.M., and Yau, K.W. (2002). Melanopsin-containing retinal. ganglion cells: Architecture, projections, and intrinsic photosensitivity. Science (New York, N.Y.) *295*, 1065-1070.

Hauenschild, C. (1955). Photoperiodizität als Ursache des von der Mondphase abhängigen Metamorphose-Rhythmus bei dem Polychaeten Platynereis Dumerilii. Z Naturforsch Pt B *10*, 658-662.

Hauenschild, C. (1956). Neue experimentelle Untersuchungen zum Problem der Lunarperiodizität. Naturwissenschaften *43*, 361-363.

Hauenschild, C. (1961). Die Schwarmperiodizität von Platynereis Dumerilii im DD/LD-Belichtungszyklus und nach Augenausschaltung. Z Naturforsch Pt B *B 16*, 753-756.

Hauenschild, C., and Fischer, A. (1969). Platynereis dumerilii: Mikroskopische Anatomie, Fortpflanzung, Entwicklung (Stuttgart: Gustav Fischer Verlag).

Hausen, H. (2007). Ultrastructure of presumptive light sensitive ciliary organs in larvae of Poecilochaetidae, Trochochaetidae, Spionidae, Magelonidae (Annelida) and its phylogenetic significance. Zoomorphology *126*, 185-201.

Hay, M.E., Renaud, P.E., and Fenical, W. (1988). Large Mobile Versus Small Sedentary Herbivores and Their Resistance to Seaweed Chemical Defenses. Oecologia *75*, 246-252.

Hepler, J.R., and Gilman, A.G. (1992). G-Proteins. Trends Biochem Sci *17*, 383-387.

Hering, L., and Mayer, G. (2014). Analysis of the opsin repertoire in the tardigrade Hypsibius dujardini provides insights into the evolution of opsin genes in panarthropoda. Genome Biol Evol *6*, 2380-2391.

Hofmann, L., and Palczewski, K. (2015). The G protein-coupled receptor rhodopsin: a historical perspective. Methods in molecular biology (Clifton, N.J.) *1271*, 3-18.

Horiguchi, H., Winawer, J., Dougherty, R.F., and Wandell, B.A. (2013). Human trichromacy revisited. P Natl Acad Sci USA *110*, E260-E269.

Hubbard, R., and Wald, G. (1951). The mechanism of rhodopsin synthesis. Proc Natl Acad Sci U S A *37*, 69-79.

Huffard, C.L., von Thun, S., Sherman, A.D., Sealey, K., and Smith, K.L. (2014). Pelagic Sargassum community change over a 40-year period: temporal and spatial variability. Mar Biol *161*, 2735-2751.

Hughes, S., Jagannath, A., Hickey, D., Gatti, S., Wood, M., Peirson, S.N., Foster, R.G., and Hankins, M.W. (2015). Using siRNA to define functional interactions between melanopsin and multiple G Protein partners. Cell Mol Life Sci *72*, 165-179.

Hunt, D.M., Carvalho, L.S., Cowing, J.A., Parry, J.W.L., Wilkie, S.E., Davies, W.L., and Bowmaker, J.K. (2007). Spectral tuning of shortwave-sensitive visual pigments in vertebrates. Photochemistry and Photobiology *83*, 303-310.

Hutchinson, T.H., Jha, A.N., and Dixon, D.R. (1995). The Polychaete Platynereis-Dumerilii (Audouin and Milne-Edwards) - a New Species for Assessing the Hazardous Potential of Chemicals in the Marine-Environment. Ecotox Environ Safe *31*, 271-281.

Ipucha, M.a.C., Santos, C.G., Lana, P.d.C., and Sbalqueiro, I.J. (2007). Cytogenetic characterization of seven South American species of nereididae (annelida: polychaeta): Implications for the karyotypic evolution. BAG *18*, 27-38.

Jacobs, R.P.W.M., and Pierson, E.S. (1979). Zostera-Marina Spathes as a Habitat for Platynereis-Dumerilii (Audouin and Milne-Edwards, 1834). Aquat Bot *6*, 403-406.

Jékely, G. (2009). Evolution of phototaxis. Philos T R Soc B *364*, 2795-2808.

Jékely, G., and Arendt, D. (2007). Cellular resolution expression profiling using confocal detection of NBT/BCIP precipitate by reflection microscopy. Biotechniques *42*, 751-755.

Jékely, G., Colombelli, J., Hausen, H., Guy, K., Stelzer, E., Nédélec, F., and Arendt, D. (2008). Mechanism of phototaxis in marine zooplankton. Nature *456*, 395-399.

Jellies, J. (2014). Detection and selective avoidance of near ultraviolet radiation by an aquatic annelid: the medicinal leech. J Exp Biol *217*, 974-985.

Jha, A.N., Hutchinson, T.H., Mackay, J.M., Elliott, B.M., Pascoe, P.L., and Dixon, D.R. (1995). The Chromosomes of Platynereis-Dumerilii (Polychaeta, Nereidae). J Mar Biol Assoc Uk *75*, 551-562.

Jiang, M., Pandey, S., and Fong, H.K.W. (1993). An Opsin Homolog in the Retina and Pigment-Epithelium. Invest Ophth Vis Sci *34*, 3669-3678.

Keplinger, S. (2010). Influence of the adult eyes on circadian and lunar rhythms in Platynereis dumerilii. (Vienna, Universität Wien).
Kessler, K., Lockwood, R.S., Williamson, C.E., and Saros, J.E. (2008). Vertical distribution of zooplankton in subalpine and alpine lakes: Ultraviolet radiation, fish predation, and the transparency-gradient hypothesis. Limnol Oceanogr *53*, 2374-2382.
Khayyeri, H., Barreto, S., and Lacroix, D. (2015). Primary cilia mechanics affects cell mechanosensation: A computational study. J Theor Biol *379*, 38-46.
Kim, H.J., Son, E.D., Jung, J.Y., Choi, H., Lee, T.R., and Shin, D.W. (2013). Violet Light Down-Regulates the Expression of Specific Differentiation Markers through Rhodopsin in Normal Human Epidermal Keratinocytes. Plos One *8*.
Knight-Jones, E.W., and Crisp, D.J. (1953). Gregariousness in Barnacles in Relation to the Fouling of Ships and to Anti-Fouling Research. Nature *171*, 1109-1110.
Knox, B.E., Salcedo, E., Mathiesz, K., Schaefer, J., Chou, W.H., Chadwell, L.V., Smith, W.C., Britt, S.G., and Barlow, R.B. (2003). Heterologous expression of limulus rhodopsin. J Biol Chem *278*, 40493-40502.
Koehl, M.A.R., and Hadfield, M.G. (2010). Hydrodynamics of Larval Settlement from a Larva's Point of View. Integrative and Comparative Biology *50*, 539-551.
Koehl, M.A.R., and Reidenbach, M.A. (2007). Swimming by microscopic organisms in ambient water flow. Exp Fluids *43*, 755-768.
Kojima, D., Mori, S., Torii, M., Wada, A., Morishita, R., and Fukada, Y. (2011). UV-Sensitive Photoreceptor Protein OPN5 in Humans and Mice. Plos One *6*.
Kojima, D., Terakita, A., Ishikawa, T., Tsukahara, Y., Maeda, A., and Shichida, Y. (1997). A novel Go-mediated phototransduction cascade in scallop visual cells. J Biol Chem *272*, 22979-22982.
Korringa, P. (1947). Relations between the Moon and Periodicity in the Breeding of Marine Animals. Ecol Monogr *17*, 347-381.
Koyanagi, M., Takada, E., Nagata, T., Tsukamoto, H., and Terakita, A. (2013). Homologs of vertebrate Opn3 potentially serve as a light sensor in nonphotoreceptive tissue. P Natl Acad Sci USA *110*, 4998-5003.
Koyanagi, M., Takano, K., Tsukamoto, H., Ohtsu, K., Tokunaga, F., and Terakita, A. (2008). Jellyfish vision starts with cAMP signaling mediated by opsin-G(s) cascade. P Natl Acad Sci USA *105*, 15576-15580.
Koyanagi, M., Terakita, A., Kubokawa, K., and Shichida, Y. (2002). Amphioxus homologs of Go-coupled rhodopsin and peropsin having 11-cis- and all-trans-retinals as their chromophores. FEBS Lett *531*, 525-528.
Kozak, M. (1987). An analysis of 5'-noncoding sequences from 699 vertebrate messenger RNAs. Nucleic acids research *15*, 8125-8148.
Krause, M., and Braucker, R. (2009). Gravitaxis of Bursaria truncatella: Electrophysiological and behavioural analyses of a large ciliate cell. Eur J Protistol *45*, 98-111.
Krishnan, A., and Schioth, H.B. (2015). The role of G protein-coupled receptors in the early evolution of neurotransmission and the nervous system. J Exp Biol *218*, 562-571.
Kühne, W. (1878). Ueber den Sehpurpur. Untersuchungen aus dem Physiologischen Institute der Universität Heidelberg *1*, 15-104.

Küpfer, M. (1915). Entwicklungsgeschichtliche und neuro-histologische Beiträge zur Kenntnis der Sehorgane am Mantelrande der Pecten-Arten : mit anschliessen vergleichend-anatomischen Betrachtungen / von Max Küpfer (Jena: Gustav Fischer).

Kwon, Y., Shim, H.S., Wang, X.Y., and Montell, C. (2008). Control of thermotactic behavior via coupling of a TRP channel to a phospholipase C signaling cascade. Nat Neurosci *11*, 871-873.

Lagman, D., Sundstrom, G., Daza, D.O., Abalo, X.M., and Larhammar, D. (2012). Expansion of transducin subunit gene families in early vertebrate tetraploidizations. Genomics *100*, 203-211.

Lamb, T.D. (1995). Photoreceptor Spectral Sensitivities - Common Shape in the Long-Wavelength Region. Vision Res *35*, 3083-3091.

Lamb, T.D. (2013). Evolution of phototransduction, vertebrate photoreceptors and retina. Prog Retin Eye Res *36*, 52-119.

Leach, T.H., Williamson, C.E., Theodore, N., Fischer, J.M., and Olson, M.H. (2015). The role of ultraviolet radiation in the diel vertical migration of zooplankton: an experimental test of the transparency-regulator hypothesis. J Plankton Res *37*, 886-896.

Leech, D.M., and Jonsen, S. (2002). Behavioral responses–UVR avoidance and Vision. In UV Effects in Aquatic Organisms and Ecosystems, E.W. Helbling, ed., pp. 455-481.

Lejeune, F., and Maquat, L.E. (2005). Mechanistic links between nonsense-mediated mRNA decay and pre-mRNA splicing in mammalian cells. Curr Opin Cell Biol *17*, 309-315.

Lemire, M., and Bourget, E. (1996). Substratum heterogeneity and complexity influence micro-habitat selection of Balanus sp and Tubularia crocea larvae. Mar Ecol Prog Ser *135*, 77-87.

Lewis, F.G., and Stoner, A.W. (1981). An Examination of Methods for Sampling Macrobenthos in Seagrass Meadows. B Mar Sci *31*, 116-124.

Liegertova, M., Pergner, J., Kozmikova, I., Fabian, P., Pombinho, A.R., Strnad, H., Paces, J., Vlcek, C., Bartunek, P., and Kozmik, Z. (2015). Cubozoan genome illuminates functional diversification of opsins and photoreceptor evolution (vol 5, 11885, 2015). Sci Rep-Uk *5*.

Lin, B., Koizumi, A., Tanaka, N., Panda, S., and Masland, R.H. (2008). Restoration of visual function in retinal degeneration mice by ectopic expression of melanopsin. P Natl Acad Sci USA *105*, 16009-16014.

Loudon, C., Best, B.A., and Koehl, M.A.R. (1994). When Does Motion Relative to Neighboring Surfaces Alter the Flow-through Arrays of Hairs. J Exp Biol *193*, 233-254.

Lucas, R.J., Douglas, R.H., and Foster, R.G. (2001). Characterization of an ocular photopigment capable of driving pupillary constriction in mice. Nat Neurosci *4*, 621-626.

Lucas, R.J., Hattar, S., Takao, M., Berson, D.M., Foster, R.G., and Yau, K.W. (2003). Diminished pupillary light reflex at high irradiances in melanopsin-knockout mice. Science (New York, N.Y.) *299*, 245-247.

Lucey, N.M., Lombardi, C., DeMarchi, L., Schulze, A., Gambi, M.C., and Calosi, P. (2015). To brood or not to brood: Are marine invertebrates that protect their offspring more resilient to ocean acidification? Sci Rep-Uk *5*.

Luo, D.G., Xue, T., and Yau, K.W. (2008). How vision begins: An odyssey. P Natl Acad Sci USA *105*, 9855-9862.

Lythgoe, J.N. (1988). Light and Vision in the Aquatic Environment. In Sensory Biology of Aquatic Animals, R.R. Fay, A.N. Popper, and W.N. Tavolga, eds. (Springer-Verlag), pp. 57-82.

Mann, R., Campos, B.M., and Luckenbach, M.W. (1991). Swimming Rate and Responses of Larvae of 3 Mactrid Bivalves to Salinity Discontinuities. Mar Ecol Prog Ser *68*, 257-269.

Margolskee, R.F. (2002). Molecular mechanisms of bitter and sweet taste transduction. Journal of Biological Chemistry *277*, 1-4.

Marin, E.P., Krishna, K.G., Zvyaga, T.A., Isele, J., Siebert, F., and Sakmar, T.P. (2000). The amino terminus of the fourth cytoplasmic loop of rhodopsin modulates rhodopsin-transducin interaction. Journal of Biological Chemistry *275*, 1930-1936.

Mason, B., Schmale, M., Gibbs, P., Miller, M.W., Wang, Q., Levay, K., Shestopalov, V., and Slepak, V.Z. (2012). Evidence for Multiple Phototransduction Pathways in a Reef-Building Coral. Plos One *7*.

Mazna, P., Grycova, L., Balik, A., Zemkova, H., Friedlova, E., Obsilova, V., Obsil, T., and Teisinger, J. (2008). The role of proline residues in the structure and function of human MT2 melatonin receptor. J Pineal Res *45*, 361-372.

McKelvey, L.M., and Forward, R.B. (1995). Activation of brine shrimp nauplii photoresponses involved in diel vertical migration by chemical cues from visual and non-visual planktivores. J Plankton Res *17*, 2191-2206.

McReynolds, J.S., and Gorman, A.L. (1970a). Membrane conductances and spectral sensitivities of Pecten photoreceptors. J Gen Physiol *56*, 392-406.

McReynolds, J.S., and Gorman, A.L. (1970b). Photoreceptor potentials of opposite polarity in the eye of the scallop, Pecten irradians. J Gen Physiol *56*, 376-391.

Menzel, R. (1979). Spectral sensitivity and color vision in invertebrates. In Handbook of Sensory Physiology: Vision in Invertebrates, H. Autrum, ed. (Berlin, Heidelberg, New York: Springer-Verlag), pp. 503–580.

Merbs, S.L., and Nathans, J. (1992). Absorption spectra of human cone pigments. Nature *356*, 433-435.

Merz, R.A., and Woodin, S.A. (2006). Polychaete chaetae: Function, fossils, and phylogeny. Integrative and Comparative Biology *46*, 481-496.

Miller, D.J., and Ball, E.E. (2009). The gene complement of the ancestral bilaterian - was Urbilateria a monster? J Biol *8*, 89.

Milne, I., Bayer, M., Cardle, L., Shaw, P., Stephen, G., Wright, F., and Marshall, D. (2009). Tablet--next generation sequence assembly visualization. Bioinformatics *26*, 401-402.

Mogami, Y., Ishii, J., and Baba, S.A. (2001). Theoretical and experimental dissection of gravity-dependent mechanical orientation in gravitactic microorganisms. Biol Bull *201*, 26-33.

Mogami, Y., Oobayashi, C., Yamaguchi, T., Ogiso, Y., and Baba, S.A. (1988). Negative Geotaxis in Sea-Urchin Larvae - a Possible Role of Mechanoreception in the Late Stages of Development. J Exp Biol *137*, 141-156.

Montell, C. (1999). Visual transduction in Drosophila. Annu Rev Cell Dev Bi *15*, 231-268.

Montell, C. (2012). Drosophila visual transduction. Trends Neurosci *35*, 356-363.
Morton, R.A., and Goodwin, T.W. (1944). Preparation of retinene in vitro. Nature *153*, 405-406.
Munz, F.W. (1958). The Photosensitive Retinal Pigments of Fishes from Relatively Turbid Coastal Waters. Journal of General Physiology *42*, 445-459.
Murakami, M., and Kouyama, T. (2008). Crystal structure of squid rhodopsin. Nature *453*, 363-U333.
Musco, L., Terlizzi, A., Licciano, M., and Giangrande, A. (2009). Taxonomic structure and the effectiveness of surrogates in environmental monitoring: a lesson from polychaetes. Mar Ecol Prog Ser *383*, 199-210.
Nagata, T., Koyanagi, M., and Terakita, A. (2010). Molecular Evolution and Functional Diversity of Opsin-Based Photopigments
Newman, L.A., Walker, M.T., Brown, R.L., Cronin, T.W., and Robinson, P.R. (2003). Melanopsin forms a functional short-wavelength photopigment. Biochemistry-Us *42*, 12734-12738.
Nickle, B., and Robinson, P.R. (2007). The opsins of the vertebrate retina: insights from structural, biochemical, and evolutionary studies. Cell Mol Life Sci *64*, 2917-2932.
Nilsson, D.E. (2009). The evolution of eyes and visually guided behaviour. Philos T R Soc B *364*, 2833-2847.
Nilsson, D.E. (2013). Eye evolution and its functional basis. Vis Neurosci *30*, 5-20.
Nishikura, K. (2010). Functions and Regulation of RNA Editing by ADAR Deaminases. Annual Review of Biochemistry, Vol 79 *79*, 321-349.
Oakley, T.H., and Speiser, D.I. (2015). How Complexity Originates: The Evolution of Animal Eyes. Annu Rev Ecol Evol S *46*, 237-+.
Offenburger, S.L. (2011). Examiniation of the Larval Phototactic Response of Platynereis dumerilii. (Tübingen, Eberhard Karls Univertity, Tübingen).
Oka, Y., Saraiva, L.R., Kwan, Y.Y., and Korsching, S.I. (2009). The fifth class of Galpha proteins. Proc Natl Acad Sci U S A *106*, 1484-1489.
Okano, T., Yoshizawa, T., and Fukada, Y. (1994). Pinopsin Is a Chicken Pineal Photoreceptive Molecule. Nature *372*, 94-97.
Oprian, D.D., Asenjo, A.B., Lee, N., and Pelletier, S.L. (1991). Design, chemical synthesis, and expression of genes for the three human color vision pigments. Biochemistry-Us *30*, 11367-11372.
Oroshnik, W. (1956). The synthesis and configuration of neo-b vitamin A and neoretinene b. J Am Chem Soc *78*, 2651-2652.
Oroshnik, W., Brown, P.K., Hubbard, R., and Wald, G. (1956). HINDERED CIS ISOMERS OF VITAMIN A AND RETINENE: THE STRUCTURE OF THE NEO-b ISOMER. Proc Natl Acad Sci U S A *42*, 578-580.
Palczewski, K., Kumasaka, T., Hori, T., Behnke, C.A., Motoshima, H., Fox, B.A., Le Trong, I., Teller, D.C., Okada, T., Stenkamp, R.E., Yamamoto, M., and Miyano, M. (2000). Crystal structure of rhodopsin: A G protein-coupled receptor. Science (New York, N.Y.) *289*, 739-745.
Panda, S., Provencio, I., Tu, D.C., Pires, S.S., Rollag, M.D., Castrucci, A.M., Pletcher, M.T., Sato, T.K., Wiltshire, T., Andahazy, M., Kay, S.A., Van Gelder, R.N., and Hogenesch, J.B. (2003). Melanopsin is required for non-image-forming photic responses in blind mice. Science (New York, N.Y.) *301*, 525-527.

Passamaneck, Y.J., Furchheim, N., Hejnol, A., Martindale, M.Q., and Luter, C. (2011). Ciliary photoreceptors in the cerebral eyes of a protostome larva. Evodevo *2*.

Passamaneck, Y.J., and Martindale, M.Q. (2013). Evidence for a phototransduction cascade in an early brachiopod embryo. Integr Comp Biol *53*, 17-26.

Pawlik, J.R. (1986). Chemical Induction of Larval Settlement and Metamorphosis in the Reef-Building Tube Worm Phragmatopoma-Californica (Sabellariidae, Polychaeta). Mar Biol *91*, 59-68.

Pawlik, J.R. (1992). Chemical Ecology of the Settlement of Benthic Marine-Invertebrates. Oceanogr Mar Biol *30*, 273-335.

Perez-Cerezales, S., Boryshpolets, S., Afanzar, O., Brandis, A., Nevo, R., Kiss, V., and Eisenbach, M. (2015). Involvement of opsins in mammalian sperm thermotaxis. Sci Rep-Uk *5*.

Petraitis, P.S. (1990). Direct and Indirect Effects of Predation, Herbivory and Surface Rugosity on Mussel Recruitment. Oecologia *83*, 405-413.

Pitt, G.A., Collins, F.D., Morton, R.A., and Stok, P. (1955). Studies on rhodopsin. VIII. Retinylidenemethylamine, an indicator yellow analogue. Biochem J *59*, 122-128.

Plachetzki, D.C., Fong, C.R., and Oakley, T.H. (2010). The evolution of phototransduction from an ancestral cyclic nucleotide gated pathway. P Roy Soc B-Biol Sci *277*, 1963-1969.

Porter, M.L., Blasic, J.R., Bok, M.J., Cameron, E.G., Pringle, T., Cronin, T.W., and Robinson, P.R. (2012). Shedding new light on opsin evolution. Proc Biol Sci *279*, 3-14.

Price, N. (2010). Habitat selection, facilitation, and biotic settlement cues affect distribution and performance of coral recruits in French Polynesia. Oecologia *163*, 747-758.

Provencio, I., Jiang, G.S., De Grip, W.J., Hayes, W.P., and Rollag, M.D. (1998). Melanopsin: An opsin in melanophores, brain, and eye. P Natl Acad Sci USA *95*, 340-345.

Provencio, I., Rodriguez, I.R., Jiang, G.S., Hayes, W.P., Moreira, E.F., and Rollag, M.D. (2000). A novel human opsin in the inner retina. Journal of Neuroscience *20*, 600-605.

Prud'homme, B., de Rosa, R., Arendt, D., Julien, J.F., Pajaziti, R., Dorresteijn, A.W.C., Adoutte, A., Wittbrodt, J., and Balavoine, G. (2003). Arthropod-like expression patterns of engrailed and wingless in the annelid Platynereis dumerilii suggest a role in segment formation. Current Biology *13*, 1876-1881.

Purschke, G. (2005). Sense organs in polychaetes (Annelida). Hydrobiologia *535*, 53-78.

Quick, K., Zhao, J., Eijkelkamp, N., Linley, J.E., Rugiero, F., Cox, J.J., Raouf, R., Gringhuis, M., Sexton, J.E., Abramowitz, J., Taylor, R., Forge, A., Ashmore, J., Kirkwood, N., Kros, C.J., Richardson, G.P., Freichel, M., Flockerzi, V., Birnbaumer, L., and Wood, J.N. (2012). TRPC3 and TRPC6 are essential for normal mechanotransduction in subsets of sensory neurons and cochlear hair cells. Open Biol *2*.

Raible, F., Tessmar-Raible, K., Osoegawa, K., Wincker, P., Jubin, C., Balavoine, G., Ferrier, D., Benes, V., de Jong, P., Weissenbach, J., Bork, P., and Arendt, D.

(2005). Vertebrate-type intron-rich genes in the marine annelid Platynereis dumerilii. Science (New York, N.Y.) *310*, 1325-1326.

Raimondi, P.T. (1990). Patterns, Mechanisms, Consequences of Variability in Settlement and Recruitment of an Intertidal Barnacle. Ecol Monogr *60*, 283-309.

Ramanathan, N., Simakov, O., Merten, C.A., and Arendt, D. (2015). Quantifying Preferences and Responsiveness of Marine Zooplankton to Changing Environmental Conditions using Microfluidics. Plos One *10*.

Ramirez, M.D., Pairett, A.N., Pankey, M.S., Serb, J.M., Speiser, D.I., Swafford, A.J., and Oakley, T.H. (2016). The last common ancestor of bilaterian animals possessed at least 7 opsins. biorXiv.

Ranade, M.R. (1957). Reversal of Phototaxis in the Larvae of Polydora-Pulchra, Carazzi (Polychaeta, Spionidae). Nature *179*, 151-152.

Randel, N., Asadulina, A., Bezares-Calderón, L.A., Verasztó, C., Williams, E.A., Conzelmann, M., Shahidi, R., and Jékely, G. (2014). Neuronal connectome of a sensory-motor circuit for visual navigation. Elife *3*.

Randel, N., Bezares-Calderón, L.A., Gühmann, M., Shahidi, R., and Jékely, G. (2013). Expression dynamics and protein localization of rhabdomeric opsins in Platynereis larvae. Integr Comp Biol *53*, 7-16.

Randel, N., Shahidi, R., Verasztó, C., Bezares-Calderón, L.A., Schmidt, S., and Jékely, G. (2015). Inter-individual stereotypy of the Platynereis larval visual connectome. Elife *4*, e08069.

Read, G. (2015). Nereis dumerilii Audouin & Milne Edwards, 1834. By: Read, G.; Fauchald, K. (Ed.) (2015) World Polychaeta database. Accessed through: World Register of Marine Species at http://www.marinespecies.org/aphia.php?p=taxdetails&id=339282 on 2017-02-09.

Rebscher, N., Zelada-Gonzalez, F., Banisch, T.U., Raible, F., and Arendt, D. (2007). Vasa unveils a common origin of germ cells and of somatic stem cells from the posterior growth zone in the polychaete Platynereis dumerilii. Dev Biol *306*, 599-611.

Reuter, T.E., White, R.H., and Wald, G. (1971). Rhodopsin and porphyropsin fields in the adult bullfrog retina. J Gen Physiol *58*, 351-371.

Rhode, B. (1992). Development and Differentiation of the Eye in Platynereis-Dumerilii (Annelida, Polychaeta). J Morphol *212*, 71-85.

Rhode, S.C., Pawlowski, M., and Tollrian, R. (2001). The impact of ultraviolet radiation on the vertical distribution of zooplankton of the genus Daphnia. Nature *412*, 69-72.

Ricevuto, E., Kroeker, K.J., Ferrigno, F., Micheli, F., and Gambi, M.C. (2014). Spatio-temporal variability of polychaete colonization at volcanic CO2 vents indicates high tolerance to ocean acidification. Mar Biol *161*, 2909-2919.

Rittschof, D., Forward, R.B., Cannon, G., Welch, J.M., McClary, M., Holm, E.R., Clare, A.S., Conova, S., McKelvey, L.M., Bryan, P., and Van Dover, C.L. (1998). Cues and context: Larval responses to physical and chemical cues. Biofouling *12*, 31-44.

Rodriguez, S.R., Ojeda, F.P., and Inestrosa, N.C. (1993). Settlement of Benthic Marine-Invertebrates. Mar Ecol Prog Ser *97*, 193-207.

Sakai, K., Imamoto, Y., Su, C.Y., Tsukamoto, H., Yamashita, T., Terakita, A., Yau, K.W., and Shichida, Y. (2012). Photochemical Nature of Parietopsin. Biochemistry-Us *51*, 1933-1941.

Sakai, K., Yamashita, T., Imamoto, Y., and Shichida, Y. (2015). Diversity of Active States in TMT Opsins. Plos One *10*.

Sato, K., Yamashita, T., Ohuchi, H., and Shichida, Y. (2011). Vertebrate Ancient-Long Opsin Has Molecular Properties Intermediate between Those of Vertebrate and Invertebrate Visual Pigments. Biochemistry-Us *50*, 10484-10490.

Sato, M., Ishikura, R., and Oshima, N. (2004). Direct effects of visible and UVA light on pigment migration in erythrophores of Nile tilapia. Pigm Cell Res *17*, 519-524.

Sexton, J.E., Desmonds, T., Quick, K., Taylor, R., Abramowitz, J., Forge, A., Kros, C.J., Birnbaumer, L., and Wood, J.N. (2016). The contribution of TRPC1, TRPC3, TRPC5 and TRPC6 to touch and hearing. Neurosci Lett *610*, 36-42.

Shahidi, R., Williams, E.A., Conzelmann, M., Asadulina, A., Verasztó, C., Jasek, S., Bezares-Calderón, L.A., and Jékely, G. (2015). A serial multiplex immunogold labeling method for identifying peptidergic neurons in connectomes. Elife *4*.

Shanks, A.L. (2009). Pelagic Larval Duration and Dispersal Distance Revisited. Biol Bull *216*, 373-385.

Sharp, D.T., and Gray, I.E. (1962). Studies on Factors Affecting Local-Distribution of Two Sea-Urchins, Arbacia-Punctulata and Lytechinus-Variegatus. Ecology *43*, 309-&.

Shen, D.W., Jiang, M.S., Hao, W.S., Tao, L., Salazar, M., and Fong, H.K.W. (1994). A Human Opsin-Related Gene That Encodes a Retinaldehyde-Binding Protein. Biochemistry-Us *33*, 13117-13125.

Shen, W.L., Kwon, Y., Adegbola, A.A., Luo, J.J., Chess, A., and Montell, C. (2011). Function of Rhodopsin in Temperature Discrimination in Drosophila. Science (New York, N.Y.) *331*, 1333-1336.

Shichida, Y., and Matsuyama, T. (2009). Evolution of opsins and phototransduction. Philos T R Soc B *364*, 2881-2895.

Shirzad-Wasei, N., and DeGrip, W.J. (2016). Heterologous expression of melanopsin: Present, problems and prospects. Prog Retin Eye Res *52*, 1-21.

Sikka, G., Hussmann, G.P., Pandey, D., Cao, S.Y., Hori, D., Park, J.T., Steppan, J., Kim, J.H., Barodka, V., Myers, A.C., Santhanam, L., Nyhan, D., Halushka, M.K., Koehler, R.C., Snyder, S.H., Shimoda, L.A., and Berkowitz, D.E. (2014). Melanopsin mediates light-dependent relaxation in blood vessels. P Natl Acad Sci USA *111*, 17977-17982.

Simakov, O., Larsson, T.A., and Arendt, D. (2013). Linking micro- and macro-evolution at the cell type level: a view from the lophotrochozoan Platynereis dumerilii. Brief Funct Genomics *12*, 430-439.

Solessio, E., and Engbretson, G.A. (1993). Antagonistic Chromatic Mechanisms in Photoreceptors of the Parietal Eye of Lizards. Nature *364*, 442-445.

Soni, B.G., and Foster, R.G. (1997). A novel and ancient vertebrate opsin. Febs Letters *406*, 279-283.

Spassova, M.A., Hewavitharana, T., Xu, W., Soboloff, J., and Gill, D.L. (2006). A common mechanism underlies stretch activation and receptor activation of TRPC6 channels. P Natl Acad Sci USA *103*, 16586-16591.

Storz, U.C., and Paul, R.J. (1998). Phototaxis in waterfleas (Daphnia magna) is differently influenced by visible and UV light. J Comp Physiol A *183*.

Su, C.Y., Luo, D.G., Terakita, A., Shichida, Y., Liao, H.W., Kazmi, M.A., Sakmar, T.P., and Yau, K.W. (2006). Parietal-eye phototransduction components and their potential evolutionary implications. Science (New York, N.Y.) *311*, 1617-1621.

Suga, H., Schmid, V., and Gehring, W.J. (2008). Evolution and functional diversity of jellyfish opsins. Current Biology *18*, 51-55.

Sun, H., Gilbert, D.J., Copeland, N.G., Jenkins, N.A., and Nathans, J. (1997). Peropsin, a novel visual pigment-like protein located in the apical microvilli of the retinal pigment epithelium. P Natl Acad Sci USA *94*, 9893-9898.

Surugiu, V., and Feunteun, M. (2008). The structure and distribution of polychaete populations influenced by sewage from the Romanian Coast of the Black Sea. Analele Ştiinţifice ale Universităţii „Al. I. Cuza" Iaşi, s. Biologie animală *LIV*.

Szikra, T., Trenholm, S., Drinnenberg, A., Juttner, J., Raics, Z., Farrow, K., Biel, M., Awatramani, G., Clark, D.A., Sahel, J.A., da Silveira, R.A., and Roska, B. (2014). Rods in daylight act as relay cells for cone-driven horizontal cell mediated surround inhibition. Nat Neurosci *17*, 1728-1735.

Tarttelin, E.E., Bellingham, J., Hankins, M.W., Foster, R.G., and Lucas, R.J. (2003). Neuropsin (Opn5): a novel opsin identified in mammalian neural tissue. Febs Letters *554*, 410-416.

Terakita, A. (2005). The opsins. Genome Biol *6*, 213.

Terakita, A., Koyanagi, M., Tsukamoto, H., Yamashita, T., Miyata, T., and Shichida, Y. (2004). Counterion displacement in the molecular evolution of the rhodopsin family (vol 11, pg 284, 2004). Nat Struct Mol Biol *11*, 384-384.

Tessmar-Raible, K., and Arendt, D. (2003). Emerging systems: between vertebrates and arthropods, the Lophotrochozoa. Curr Opin Genet Dev *13*, 331-340.

Tessmar-Raible, K., Raible, F., Christodoulou, F., Guy, K., Rembold, M., Hausen, H., and Arendt, D. (2007). Conserved sensory-neurosecretory cell types in annelid and fish forebrain: Insights into hypothalamus evolution. Cell *129*, 1389-1400.

Tessmar-Raible, K., Steinmetz, P.R., Snyman, H., Hassel, M., and Arendt, D. (2005). Fluorescent two-color whole mount in situ hybridization in Platynereis dumerilii (Polychaeta, Annelida), an emerging marine molecular model for evolution and development. Biotechniques *39*, 460, 462, 464.

Thorson, G. (1964). Light as an ecological factor in the dispersal and settlement of larvae of marine bottom invertebrates. Ophelia *1*, 167-208.

Tomer, R., Denes, A.S., Tessmar-Raible, K., and Arendt, D. (2010). Profiling by Image Registration Reveals Common Origin of Annelid Mushroom Bodies and Vertebrate Pallium. Cell *142*, 800-809.

Tosches, M.A. (2013). Development and function of brain photoreceptors in the annelid Platynereis dumerilii.In Combined Faculties for the Natural Sciences and for Mathematics (Ruperto-Carola University of Heidelberg).

Tosches, M.A., Bucher, D., Vopalensky, P., and Arendt, D. (2014). Melatonin Signaling Controls Circadian Swimming Behavior in Marine Zooplankton. Cell *159*, 46-57.

Townson, S.M., Chang, B.S., Salcedo, E., Chadwell, L.V., Pierce, N.E., and Britt, S.G. (1998). Honeybee blue- and ultraviolet-sensitive opsins: cloning, heterologous expression in Drosophila, and physiological characterization. The Journal of neuroscience : the official journal of the Society for Neuroscience *18*, 2412-2422.

Tsuda, M., Sakurai, D., and Goda, M. (2003). Direct evidence for the role of pigment cells in the brain of ascidian larvae by laser ablation. J Exp Biol *206*, 1409-1417.

Tsukamoto, H., Terakita, A., and Shichida, Y. (2005). A rhodopsin exhibiting binding ability to agonist all-trans-retinal. Proc Natl Acad Sci U S A *102*, 6303-6308.

Tu, D.C., Zhang, D.Y., Demas, J., Slutsky, E.B., Provencio, I., Holy, T.E., and Van Gelder, R.N. (2005). Physiologic diversity and development of intrinsically photosensitive retinal ganglion cells. Neuron *48*, 987-999.

Valero-Gracia, A., Petrone, L., Oliveri, P., Nilsson, D.-E., and Arnone, M.I. (2016). Non-directional photoreceptors in the pluteus of Strongylocentrotus purpuratus. Frontiers in Ecology and Evolution *4*.

Vasudevan, S., Peltz, S.W., and Wilusz, C.J. (2002). Non-stop decay - a new mRNA surveillance pathway. Bioessays *24*, 785-788.

Veedin-Rajan, V.B., Fischer, R.M., Raible, F., and Tessmar-Raible, K. (2013). Conditional and Specific Cell Ablation in the Marine Annelid Platynereis dumerilii. Plos One *8*.

Velarde, R.A., Sauer, C.D., Walden, K.K.O., Fahrbach, S.E., and Robertson, H.M. (2005). Pteropsin: A vertebrate-like non-visual opsin expressed in the honey bee brain. Insect Biochem Molec *35*, 1367-1377.

Verasztó, C., Ueda, N., Bezares-Calderón, L.A., Panzera, A., Williams, E.A., Shahidi, R., and Jékely, G. (2017). Ciliomotor circuitry underlying whole-body coordination of ciliary activity in the Platynereis larva. bioRxiv.

Vigh, B., Manzano, M.J., Zadori, A., Frank, C.L., Lukats, A., Rohlich, P., Szel, A., and David, C. (2002). Nonvisual photoreceptors of the deep brain, pineal organs and retina. Histol Histopathol *17*, 555-590.

Visscher, J.P. (1927). Nature and extent of fouling of ships' bottoms. Bulletin of the Bureau of Fisheries *43*, 193-252.

Vopalensky, P., Pergner, J., Liegertova, M., Benito-Gutierrez, E., Arendt, D., and Kozmik, Z. (2012). Molecular analysis of the amphioxus frontal eye unravels the evolutionary origin of the retina and pigment cells of the vertebrate eye. P Natl Acad Sci USA *109*, 15383-15388.

Wada, S., Kawano-Yamashita, E., Koyanagi, M., and Terakita, A. (2012). Expression of UV-Sensitive Parapinopsin in the Iguana Parietal Eyes and Its Implication in UV-Sensitivity in Vertebrate Pineal-Related Organs. Plos One *7*.

Wald, G. (1934). Carotenoids and the vitamin A cycle in vision. Nature *134*, 65-65.

Wald, G. (1935). Pigments of the bull frog retina. Nature *136*, 832-833.

Wald, G. (1937). Visual purple system in fresh-water fishes. Nature *139*, 1017-1018.

Wald, G. (1968). Molecular basis of visual excitation. Science (New York, N.Y.) *162*, 230-239.
Wald, G., and Brown, P.K. (1958). Human Rhodopsin. Am J Ophthalmol *45*, 286-286.
Wald, G., Brown, P.K., Hubbard, R., and Oroshnik, W. (1955). Hindered Cis Isomers of Vitamin a and Retinene: The Structure of the Neo-B Isomer. Proc Natl Acad Sci U S A *41*, 438-451.
Wald, G., and Rayport, S. (1977). Vision in Annelid Worms. Science (New York, N.Y.) *196*, 1434-1439.
Walters, L.J., Hadfield, M.G., and Smith, C.M. (1996). Waterborne chemical compounds in tropical macroalgae: Positive and negative cues for larval settlement. Mar Biol *126*, 383-393.
Welch, J.M., Rittschof, D., Bullock, T.M., and Forward, R.B. (1997). Effects of chemical cues on settlement behavior of blue crab Callinectes sapidus postlarvae. Mar Ecol Prog Ser *154*, 143-153.
Wess, J., Nanavati, S., Vogel, Z., and Maggio, R. (1993). Functional-Role of Proline and Tryptophan Residues Highly Conserved among G-Protein-Coupled Receptors Studied by Mutational Analysis of the M3-Muscarinic-Receptor. Embo J *12*, 331-338.
Wicks, N.L., Chan, J.W., Najera, J.A., Ciriello, J.M., and Oancea, E. (2011). UVA Phototransduction Drives Early Melanin Synthesis in Human Melanocytes. Current Biology *21*, 1906-1911.
Wilkie, T.M., Gilbert, D.J., Olsen, A.S., Chen, X.N., Amatruda, T.T., Korenberg, J.R., Trask, B.J., Dejong, P., Reed, R.R., Simon, M.I., Jenkins, N.A., and Copeland, N.G. (1992). Evolution of the Mammalian G-Protein Alpha-Subunit Multigene Family. Nat Genet *1*, 85-91.
Wilkie, T.M., and Yokoyama, S. (1994). Evolution of the G-Protein Alpha-Subunit Multigene Family. Soc Gen Phy *49*, 249-270.
Williams, E.A., Conzelmann, M., and Jékely, G. (2015). Myoinhibitory peptide regulates feeding in the marine annelid Platynereis. Frontiers in Zoology *12*.
Williams, G.B. (1964). The Effect of Extracts of Fucus Serratus in Promoting the Settlement of Larvae of Spirorbis Borealis [Polychaeta]. J Mar Biol Assoc Uk *44*, 397-414.
Williams, G.B. (1965). Observations on the behaviour of the planulae larvae of Clava squamata. J Mar Biol Assoc Uk *45*, 257-273.
Winet, H. (1973). Wall drag on free-moving ciliated micro-organisms. J Exp Biol *59*, 753-766.
Winet, H., and Jahn, T.L. (1974). Geotaxis in protozoa I. A propulsion-gravity model for Tetrahymena (Ciliata). J Theor Biol *46*, 449-465.
Woodin, S.A. (1991). Recruitment of Infauna - Positive or Negative Cues. Am Zool *31*, 797-807.
Woodin, S.A., Lindsay, S.M., and Lincoln, D.E. (1997). Biogenic bromophenols as negative recruitment cues. Mar Ecol Prog Ser *157*, 303-306.
Woodin, S.A., Merz, R.A., Thomas, F.M., Edwards, D.R., and Garcia, I.L. (2003). Chaetae and mechanical function: tools no Metazoan class should be without. Hydrobiologia *496*, 253-258.
Xue, T., Do, M.T., Riccio, A., Jiang, Z., Hsieh, J., Wang, H.C., Merbs, S.L., Welsbie, D.S., Yoshioka, T., Weissgerber, P., Stolz, S., Flockerzi, V., Freichel, M., Simon,

M.I., Clapham, D.E., and Yau, K.W. (2011). Melanopsin signalling in mammalian iris and retina. Nature *479*, 67-73.
Yamashita, T., Ohuchi, H., Tomonari, S., Ikeda, K., Sakai, K., and Shichida, Y. (2010). Opn5 is a UV-sensitive bistable pigment that couples with Gi subtype of G protein. P Natl Acad Sci USA *107*, 22084-22089.
Yau, K.W., and Hardie, R.C. (2009). Phototransduction Motifs and Variations. Cell *139*, 246-264.
Yokoyama, S. (2000). Phylogenetic analysis and experimental approaches to study color vision in vertebrates. Method Enzymol *315*, 312-325.
Yokoyama, S., Radlwimmer, F.B., and Blow, N.S. (2000). Ultraviolet pigments in birds evolved from violet pigments by a single amino acid change. P Natl Acad Sci USA *97*, 7366-7371.
Zagarese, H.E., and Williamson, C.E. (1994). Modeling the Impacts of UV-B Radiation on Ecological Interactions in Freshwater and Marine Ecosystems. In Stratospheric Ozone Depletion/UV-B Radiation in the Biosphere, R.H. Biggs, and M.E.B. Joyner, eds. (Springer Berlin Heidelberg), pp. 315-328.
Zantke, J., Bannister, S., Rajan, V.B.V., Raible, F., and Tessmar-Raible, K. (2014). Genetic and Genomic Tools for the Marine Annelid Platynereis dumerilii. Genetics *197*, 19-31.
Zantke, J., Ishikawa-Fujiwara, T., Arboleda, E., Lohs, C., Schipany, K., Hallay, N., Straw, A.D., Todo, T., and Tessmar-Raible, K. (2013). Circadian and Circalunar Clock Interactions in a Marine Annelid. Cell Rep *5*, 99-113.
Zeeck, E., Harder, T., and Beckmann, M. (1998). Uric acid: The sperm-release pheromone of the marine polychaete Platynereis dumerilii. J Chem Ecol *24*, 13-22.
Zelada-González, Y.F. (2004). Germline development in Platynereis dumerilii and its connection to
embryonic patterning.In Combined Faculties for Natural Sciences and for Mathematics (Heidelberg, Germany, Ruperto-Carola University).
Zerbino, D.R., and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res *18*, 821-829.
Zuker, C.S. (1996). The biology of vision in Drosophila. P Natl Acad Sci USA *93*, 571-576.

11 Appendix

The files in the appendix are available at:
https://github.com/JekelyLab/PhDThesis_Martin_Guehmann.

11.1 The ImageJ macros for larva tracking

Code 9: The ImageJ macro file Horizontal-Track-Extractor.ijm

```
////////////////////////////////////////////////////////
// This file is best viewed with a monospaced font.    //
////////////////////////////////////////////////////////

////////////////////////////////////////////////////////
// ImageJ macro to extract the tracks and the          //
// distribution of the larvae from the raw videos.     //
////////////////////////////////////////////////////////

////////////////////////////////////////////////////////
// Asks the user where the input video files are, and  //
// into which directory the output files should go.    //
// The user can give the input directory first and     //
// then the output directory. Or the user can give a   //
// text file that contains on each line an input       //
// directory and an output directory separated by a    //
// space, for batch processing.                        //
////////////////////////////////////////////////////////
macro "Extract Tracks"
{
      if(getBoolean("Choose an input and output directory otherwise
give a text file containing a list of input and output directories"))
      {
        inputDir = getDirectory("Choose the input directory (where the
files are)");
       outputDir = getDirectory("Choose the output directory (where
the files should go)");
       extractTracks(inputDir, outputDir);
      }
      else
      {
       fileList = File.openDialog("Open a text file containing a list
of input and output directories");
       lines = split(File.openAsString(fileList), "\n");
       for(i = 0; i < lines.length; i++)
       {
             dirs = split(lines[i], " ");
             extractTracks(dirs[1], dirs[0]);
       }
      }
}

////////////////////////////////////////////////////////
// Extracts the tracks from the videos that are all in //
// the folder inputDir via mTrack2. All files in the   //
// folder inputDir must be video files that ImageJ can //
// read. Otherwise this macro aborts with an error     //
// message. And will not analyze more videos.          //
```

```
////////////////////////////////////////////////////////
function extractTracks(inputDir, outputDir)
{
	// These three parameters can be adjusted
	// depending on video length and frame rate
	numFramesToProcess		= 50; // Cut the video into pieces
of 50 frames
	startFrame			= 1;
	lastFrame			= numFramesToProcess;

	print (inputDir);
	print (outputDir);

	// Do not show all the calculation steps on the
	// ImageJ user interface, this saves time and memory.
	setBatchMode(true);

	// Get all the files in the input directory.
	list=getFileList(inputDir);
	Array.sort(list);
	print(list.length);

	for (k=0; k<list.length; k++)
	{
	 print(list[k]);
	 open(inputDir + list[k]);
	 imageTitle = getTitle();
	 imageTitle = replace(imageTitle, " ", "_"); // Replace spaces
by underscores to avoid problems with file writing

	 run("8-bit");
	 rename("video");

	 // Cut the video into smaller parts and give each part an
index.
	 // Start with 100 to avoid problems with file sorting
	 m=100;
	 while (nSlices > 1)
	 {
		// Process the first n frames of the video so that
different points in time can be checked.
		nSlices
		run("Duplicate...", "title=stack duplicate range=" +
startFrame + "-" + lastFrame);
		selectWindow("stack");
		processImage();
		selectWindow("stack");
		threshold();

		createDistributionImage(m);
		trackParticles2(m);
		close();

		// Delete the first n frames of the video so that the
next n frames can be processed.
		selectWindow("video");
		if(nSlices > numFramesToProcess)
		{
			run("Slice Remover", "first=1 last=" +
numFramesToProcess + " increment=1");
		}
```

```
            else
            {
                  run("Slice Remover", "first=1 last=" + (nSlices-1)
+ " increment=1");
            }
            selectWindow("video");

            m++;
      }
      close();

      }

      setBatchMode(false);
}

////////////////////////////////////////////////////////
// Creates an image of the distribution of the larvae  //
// and saves it to a file in the text image format.    //
////////////////////////////////////////////////////////
function createDistributionImage(laneNumber)
{
      selectWindow("stack");
      run("Z Project...", "start=1 stop=-1 projection=[Average
Intensity]");
      selectWindow("AVG_stack");

      outputFilename = imageTitle + "_lane_" + laneNumber
+"_vertical"+".text_image";
      fullPathResults = outputDir + outputFilename;
      saveAs("Text image", fullPathResults);
      close();
}

////////////////////////////////////////////////////////
// Removes the background so that the moving larvae    //
// are left as dot in the video that can be tracked.   //
// These thing can be adjusted according to the        //
// contrast in the video.                              //
////////////////////////////////////////////////////////
function processImage()
{
      // Adjust the contrast
      run("Brightness/Contrast...");
      run("Enhance Contrast", "saturated=0.5");
      run("Apply LUT", "stack");
      run("Invert", "stack");

      // Subtract the average projection (Remove background)
      run("Z Project...", "start=1 stop=-1 projection=[Average
Intensity]");
      imageCalculator("Subtract stack", "stack","AVG_stack");
      selectWindow("AVG_stack");
      close();
      selectWindow("stack");

      // Apply some filters
      run("Unsharp Mask...", "radius=20 mask=0.90 stack");
      run("Invert", "stack");

      run("Despeckle", "stack");
```

```
}

////////////////////////////////////////////////////////
// Thresholds an 8 bit grayscale image, and converts  //
// all pixel above the threshold to 255 (i.e. white). //
// Otherwise it converts all values to 0 (i.e. black). //
////////////////////////////////////////////////////////
function threshold()
{
      setThreshold(0, 180);
      run("Convert to Mask", " ");
}

////////////////////////////////////////////////////////
// Tracks the larvae with mTrack2, and writes the     //
// output to a file, with .res extension.             //
////////////////////////////////////////////////////////
function trackParticles2(laneNumber)
{
      run("Clear Results");
      outputFilename = imageTitle + "_lane_" + laneNumber + ".res";
      fullPathResults = outputDir + outputFilename;
      run("MTrack2 ", "minimum=1 maximum=200 maximum_=3 minimum_=10
display save save=" + fullPathResults);

      run("Clear Results");
}
```

Code 10: The ImageJ macro file Vertical-Track-Extractor.ijm

```
////////////////////////////////////////////////////////
// This file is best viewed with a monospaced font.   //
////////////////////////////////////////////////////////

////////////////////////////////////////////////////////
// ImageJ macro to extract the tracks and the         //
// distribution of the larvae from the raw videos.    //
////////////////////////////////////////////////////////

////////////////////////////////////////////////////////
// Asks the user where the input video files are, and //
// into which directory the output files should go.   //
// The user can give the input directory first and    //
// then the output directory. Or the user can give a  //
// text file that contains on each line an input      //
// directory and an output directory separated by a   //
// space, for batch processing.                       //
////////////////////////////////////////////////////////
macro "Extract Tracks"
{
      if(getBoolean("Choose an input and output directory otherwise
give a text file containing a list of input and output directories"))
      {
        inputDir = getDirectory("Choose the input directory (where the
files are)");
       outputDir = getDirectory("Choose the output directory (where
the files should go)");
       extractTracks(inputDir, outputDir);
      }
      else
```

```
	{
	 fileList = File.openDialog("Open a text file containing a list
of input and output directories");
	 lines = split(File.openAsString(fileList), "\n");
	 for(i = 0; i < lines.length; i++)
	 {
		dirs = split(lines[i], " ");
		extractTracks(dirs[1], dirs[0]);
	 }
	}
}

////////////////////////////////////////////////////////
// Extracts the tracks from the videos that are all in //
// the folder inputDir via mTrack2. All files in the   //
// folder inputDir must be video files that ImageJ can //
// read. Otherwise this macro aborts with an error     //
// message. And will not analyze more videos.          //
////////////////////////////////////////////////////////
function extractTracks(inputDir, outputDir)
{
	// These three parameters can be adjusted
	// depending on video length and frame rate
	numFramesToProcess = 452;  // 30s with 15fps
	startFrame                    = 12;
	lastFrame                     = numFramesToProcess - 15;

	print (inputDir);
	print (outputDir);

	// Do not show all the calculation steps on the
	// ImageJ user interface, this saves time and memory.
	setBatchMode(true);

	// Get all the files in the input directory.
	list=getFileList(inputDir);
	Array.sort(list);
	print(list.length);

	// Track the number of files that have been read
	files = 0;

	for (k=0; k<list.length; k++)
	{
	 print(list[k]);
	 open(inputDir + list[k]);
	 imageTitle = getTitle();
	 imageTitle = replace(imageTitle, " ", "_"); // Replace spaces
by underscores to avoid problems with file writing

	 // Remove the first n frames, to synchronize with the
phototaxis protocol start.
	 // Has to be adjusted for each video.
	 // Can be done for several input files individually.
	 // This is needed to synchronize the start of the AxioVision
	 // protocol and the start of the video, which was started a few
	 // seconds before the start of the protocol.
	 // For now all set to 0.
	 if(files == 0)
	 {
		run("Slice Remover", "first=1 last=0 increment=1");
```

```
	}
	else if(files == 1)
	{
		run("Slice Remover", "first=1 last=0 increment=1");
	}
	else if(files == 2)
	{
		run("Slice Remover", "first=1 last=0 increment=1");
	}
	else if(files == 3)
	{
		run("Slice Remover", "first=1 last=0 increment=1");
	}
	else if(files == 4)
	{
		run("Slice Remover", "first=1 last=0 increment=1");
	}
	else
	{
		run("Slice Remover", "first=1 last=0 increment=1");
	}

	files++;

	run("8-bit");
	rename("video");

	// Cut the video into smaller parts and give each part an
index.
	// Start with 100 to avoid problems with file sorting
	m=100;
	while (nSlices > 1)
	{
		// Process the first n frames of the video so that
different points in time can be checked.
		nSlices
		run("Duplicate...", "title=stack duplicate range=" +
startFrame + "-" + lastFrame);
		selectWindow("stack");
		processImage();
		selectWindow("stack");
		threshold();

		createDistributionImage(m);
		trackParticles2(m);
		close();

		// Delete the first n frames of the video so that the
next n frames can be processed.
		selectWindow("video");
		if(nSlices > numFramesToProcess)
		{
			run("Slice Remover", "first=1 last=" +
numFramesToProcess + " increment=1");
		}
		else
		{
			run("Slice Remover", "first=1 last=" + (nSlices-1)
+ " increment=1");
		}
		selectWindow("video");
```

```
            m++;
       }
       close();

      }

      setBatchMode(false);
      print("End");
}

////////////////////////////////////////////////////////
// Creates an image of the distribution of the larvae  //
// and saves it to a file in the text image format.    //
////////////////////////////////////////////////////////
function createDistributionImage(laneNumber)
{
      selectWindow("stack");
      run("Z Project...", "start=1 stop=-1 projection=[Average
Intensity]");
      selectWindow("AVG_stack");

      outputFilename = imageTitle + "_lane_" + laneNumber
+"_vertical"+".text_image";
      fullPathResults = outputDir + outputFilename;
      saveAs("Text image", fullPathResults);
      close();
}

////////////////////////////////////////////////////////
// Removes the background so that the moving larvae    //
// are left as dot in the video that can be tracked.   //
// These thing can be adjusted according to the        //
// contrast in the video.                              //
////////////////////////////////////////////////////////
function processImage()
{
      // Subtract the average projection (Remove background)
      run("Z Project...", "start=1 stop=-1 projection=[Average
Intensity]");
      imageCalculator("Subtract stack", "stack","AVG_stack");
      selectWindow("AVG_stack");
      close();
      selectWindow("stack");

      // Apply some filters
      run("Unsharp Mask...", "radius=20 mask=0.90 stack");
      run("Invert", "stack");

      // Adjust the contrast
      run("Brightness/Contrast...");
      run("Enhance Contrast", "saturated=0.5");
      run("Apply LUT", "stack");
}

////////////////////////////////////////////////////////
// Thresholds an 8 bit grayscale image, and converts   //
// all pixel above the threshold to 255 (i.e. white).  //
// Otherwise it converts all values to 0 (i.e. black). //
////////////////////////////////////////////////////////
function threshold()
```

```
{
	setThreshold(0, 200);
	run("Convert to Mask", " ");
}

////////////////////////////////////////////////////////
// Tracks the larvae with mTrack2, and writes the     //
// output to a file, with .res extension.             //
////////////////////////////////////////////////////////
function trackParticles2(laneNumber)
{
	run("Clear Results");
	outputFilename = imageTitle + "_lane_" + laneNumber + ".res";
	fullPathResults = outputDir + outputFilename;
	run("MTrack2 ", "minimum=2 maximum=100 maximum_=3 minimum_=50
display save save=" + fullPathResults);

	run("Clear Results");
}
```

Code 11: The ImageJ macro file Vertical-Cuvette-Track-Extractor.ijm

```
////////////////////////////////////////////////////////
// This file is best viewed with a monospaced font.   //
////////////////////////////////////////////////////////

////////////////////////////////////////////////////////
// ImageJ macro to extract the tracks and the         //
// distribution of the larvae from the raw videos.    //
////////////////////////////////////////////////////////

////////////////////////////////////////////////////////
// Asks the user where the input video files are, and //
// into which directory the output files should go.   //
// The user can give the input directory first and    //
// then the output directory. Or the user can give a  //
// text file that contains on each line an input      //
// directory and an output directory seperated by a   //
// space, for batch processing.                       //
////////////////////////////////////////////////////////
macro "Extract Tracks"
{
	if(getBoolean("Choose an input and output directory otherwise
give a text file containing a list of input and output directories"))
	{
	 inputDir = getDirectory("Choose the input directory (where the
files are)");
	 outputDir = getDirectory("Choose the output directory (where
the files should go)");
	 extractTracks(inputDir, outputDir);
	}
	else
	{
	 fileList = File.openDialog("Open a text file containing a list
of input and output directories");
	 lines = split(File.openAsString(fileList), "\n");
	 for(i = 0; i < lines.length; i++)
	 {
		dirs = split(lines[i], " ");
		extractTracks(dirs[0], dirs[1]);
```

```
      }
     }
}

////////////////////////////////////////////////////////
// Extracts the tracks from the videos that are all in //
// the folder inputDir via mTrack2. All files in the   //
// folder inputDir must be video files that ImageJ can //
// read. Otherwise this macro aborts with an error     //
// message. And will not analyse more videos.          //
////////////////////////////////////////////////////////
function extractTracks(inputDir, outputDir)
{
     // These three parameters can be adjusted
     // depending on video length and frame rate
     numFramesToProcess          = 240;  // 15s with 16fps
     startFrame                  = 1;
     lastFrame                   = numFramesToProcess;

     print (inputDir);
     print (outputDir);

     // Do not show all the calculation steps on the
     // ImageJ user interface, this saves time and memory.
     setBatchMode(true);

     // Get all the files in the input directory.
     list=getFileList(inputDir);
     Array.sort(list);
     print(list.length);

     // Track the number of files that have been read
     files = 0;

     for (k=0; k<list.length; k++)
     {
      print(list[k]);
      open(inputDir + list[k]);
      imageTitle = getTitle();
      imageTitle = replace(imageTitle, " ", "_"); // Replace spaces
by underscores to avoid problems with file writing

      // Remove the first n frames, to synchronize with the
phototaxis protocol start.
      // Has to be adjusted for each video.
      // Can be done for several input files individually.
      // This is needed to synchronize the start of the AxioVision
      // protocol and the start of the video, which was started a few
      // seconds before the start of the protocol.
      // For now all set to 0.
      if(files == 0)
      {
            run("Slice Remover", "first=1 last=0 increment=1");
      }
      else if(files == 1)
      {
            run("Slice Remover", "first=1 last=0 increment=1");
      }
      else if(files == 2)
      {
            run("Slice Remover", "first=1 last=0 increment=1");
```

```
	}
	else if(files == 3)
	{
		run("Slice Remover", "first=1 last=0 increment=1");
	}
	else if(files == 4)
	{
		run("Slice Remover", "first=1 last=0 increment=1");
	}
	else
	{
		run("Slice Remover", "first=1 last=0 increment=1");
	}

	files++;

	run("8-bit");
	rename("video");

	// Cut the video into smaller parts and give each part an
index.
	// Start with 100 to aviod problems with file sorting
	m=100;
	while (nSlices > 1)
	{
		// Process the first n frames of the video so that
different points in time can be checked.
		nSlices
		run("Duplicate...", "title=stack duplicate range=" +
startFrame + "-" + lastFrame);
		selectWindow("stack");
		processImage();
		selectWindow("stack");
		threshold();

		createDistributionImage(m);
		trackParticles2(m);
		close();

		// Delete the first n frames of the video so that the
next n frames can be processed.
		selectWindow("video");
		if(nSlices > numFramesToProcess)
		{
			run("Slice Remover", "first=1 last=" +
numFramesToProcess + " increment=1");
		}
		else
		{
			run("Slice Remover", "first=1 last=" + (nSlices-1)
+ " increment=1");
		}
		selectWindow("video");

		m++;
	}
	close();

     }

     setBatchMode(false);
```

```
	print ("End");
}

////////////////////////////////////////////////////////
// Creates an image of the distribution of the larvae  //
// and saves it to a file in the text image format.    //
////////////////////////////////////////////////////////
function createDistributionImage(laneNumber)
{
	selectWindow("stack");
	run("Z Project...", "start=1 stop=-1 projection=[Average
Intensity]");
	selectWindow("AVG_stack");

	outputFilename= imageTitle + "_lane_" + laneNumber
+"_vertical"+".text_image";
	fullPathResults = outputDir + outputFilename;
	saveAs("Text image", fullPathResults);
	close();
}

////////////////////////////////////////////////////////
// Removes the background so that the moving larvae    //
// are left as dot in the video that can be tracked.   //
// These thing can be adjusted according to the        //
// contrast in the video.                              //
////////////////////////////////////////////////////////
function processImage()
{
	// Subtract the average projection (Remove background)
	run("Z Project...", "start=1 stop=-1 projection=[Average
Intensity]");
	imageCalculator("Subtract stack", "stack","AVG_stack");
	selectWindow("AVG_stack");
	close();
	selectWindow("stack");

	// Apply some filters
	run("Invert", "stack");

	// Adjust the contrast
	run("Max...", "value=253 stack");
	run("Enhance Contrast", "saturated=0.35");
	run("Apply LUT", "stack");
	run("Remove Outliers...", "radius=4 threshold=0 which=Dark
stack");
}

////////////////////////////////////////////////////////
// Thresholds an 8 bit grayscale image, and converts   //
// all pixel above the threshold to 255 (i.e. white).  //
// Otherwise it converts all values to 0 (i.e. black). //
////////////////////////////////////////////////////////
function threshold()
{
	setThreshold(0, 230);
	run("Convert to Mask", " ");
}

////////////////////////////////////////////////////////
// Tracks the larvae with mTrack2, and writes the      //
```

```
// output to a file, with .res extension.                    //
////////////////////////////////////////////////////////////
function trackParticles2(laneNumber)
{
      run("Clear Results");
      outputFilename = imageTitle + "_lane_" + laneNumber + ".res";
      fullPathResults = outputDir + outputFilename;
      run("MTrack2 ", "minimum=3 maximum=100 maximum_=15 minimum_=15
display save save=" + fullPathResults);

      run("Clear Results");
}
```

11.2 The Perl files for track analysis

Code 12: The Perl file TrackProcessor.pl

```
#!/usr/bin/perl -w

############################################################
# This script takes as input two files from the ImageJ #
# macro. The first file contains the coordinates for   #
# each track: x, y, and time encoded as frame number.  #
# The second file contains the vertical distribution   #
# of the larvae in text-image format.                  #
# The second file is implicitly given by the name of   #
# the first file, which is appended by                 #
# _vertical.text_image.txt to form the name of the     #
# second file.                                         #
# This script takes additional input parameters, see   #
# below or start the script with insufficient input    #
# parameters.                                          #
# This script generates several output file. One is a  #
# larval distribution image as png file. It also       #
# generates a track plot of the single tracks, which   #
# is useful for debugging and trouble shooting, for    #
# more details see below where the code generates the  #
# image.                                               #
# Another file contains the summary of various         #
# measurements, like the average displacement of the   #
# larvae across the vertical axis. This file can be    #
# appended when more measurements come in.             #
############################################################

use strict;
use warnings;

# Get the directory of this script:
use FindBin qw($Bin);
use lib $FindBin::Bin;
FindBin::again(); # Just to be sure that it hasn't been called in
another script before

use Statistics::Descriptive;
use Math::BigFloat;
use Math::Trig;
use MTrack;
use File::Basename;

use Storable qw<dclone>;
```

```
#############################################################
# NormalizeToCount divides a value by count, which          #
# should be positive. Returns zero if count is not          #
# bigger zero.                                              #
#############################################################
sub NormalizeToCount
{
      my $value = $_[0];
      my $count = $_[1];

      if($count > 0)
      {
       return $value / $count;
      }
      else
      {
       return 0;
      }
}

#############################################################
# Determine the best distance between the single            #
# ticks, so that the plots are the most readable and        #
# scalable.                                                 #
#############################################################
sub BestTick
{
    my $largest   = $_[0];
    my $mostticks = $_[1];
    my $minimum = $largest / $mostticks;
    my $magnitude = 10 ** floor(log($minimum) / log(10));
    my $residual = $minimum / $magnitude;

    if    ($residual > 5) { return 10 * $magnitude;}
    elsif ($residual > 2) { return  5 * $magnitude;}
    elsif ($residual > 1) { return  2 * $magnitude;}
    else                  { return      $magnitude;}
}

#############################################################
# Min, Max, and Round functions                             #
#############################################################
sub max ($$) { $_[$_[0] < $_[1]] }
sub min ($$) { $_[$_[0] > $_[1]] }

# Perl doesn't have round, so let's implement it
sub round
{
    my($number) = shift;
    return int($number + .5 * ($number <=> 0));
}

use POSIX qw(ceil floor);
use GD::Simple;

#############################################################
# Deal with script input                                    #
#############################################################
my $usage  = "usage: perl program.pl infile.res fps
column_width_in_px air_at_top_in_px not_visible_bottom_in_px
(mTrackVersion), (column_width_in_mm), (display_left_right),
```

```
(printFramesMode), (printFramesBackground), (particleSize),
(trackImageStyle) \n\n"
            . "perl:                      Start perl\n"
            . "program.pl:                This script\n"
            . "infile.res:                mTrack output file: Do not
use files containing spaces, unless you escape the spaces\n"
            . "fps:                       The frame rate the video was
recorded with\n"
            . "column_width_in_px:        The width of the view field
on screen in pixels\n"
            . "air_at_top_in_px:          Number of pixels that should
be removed from the top\n"
            . "not_visible_bottom_in_px: Number of pixels that should
be cut from the bottom\n"
            . "mTrackVersion:             The version of mTrack that
was used to generate the tracks, default value 2\n"
            . "                           Use 2 for mTrack2, and any
other value for mTrack3.\n"
            . "column_width_in_mm:        The width of the view field
on screen in mm, default value 31 mm\n"
            . "                           If not default value is used,
isMTrack3 must be specified.\n"
            . "display_left_right:        If 0 distinguishes upward and
downward tracks by color,\n"
            . "                           Otherwise distinguishes
leftward and rightward tracks by color.\n"
            . "                           Default value is 0.\n"
            . "printFramesMode:           If bigger 0, it generates for
each frame points and track images.\n"
            . "                           If 1 it encodes the tracks
and dots by time.\n"
            . "                           If 2 it encodes the tracks
and dots by angel.\n"
            . "                           If bigger 2, it encodes the
tracks and dots by up and down, and time.\n"
            . "                           Default value is 0.\n"
            . "printFramesBackground:     If printFramesMode bigger 0,
it makes the background white.\n"
            . "                           If it is 1 it makes the
background black. Otherwise does not create frames.\n"
            . "                           Default value is 0.\n"
            . "particleSize:              If printFramesMode bigger 0,
it specifies the size of the particles to be printed.\n"
            . "                           The default size is 10.\n"
            . "trackImageStyle:           Defines how the images for
the tracks are layouted:\n"
            . "                           0 is debug style and
default,\n"
            . "                           1 is presentation style,\n"
            . "                           otherwise is thesis style.\n"
            . "\n";


# Check that everything is okay, sometimes tabs, spaces, and newlines
make trouble. The number of arguments could hint about this.
print scalar(@ARGV), "\n";

die $usage unless (@ARGV > 4 and @ARGV < 13);

my $infile = $ARGV[0];
print $infile, "\n";
```

```
my $mTrackVersion = (@ARGV > 5) ? $ARGV[5] : 2;
print "mTrack version used: ", $mTrackVersion, "\n";

# Variable parameters fetched from the script arguments:
my $frame_rate                          = $ARGV[1];
my $columnWidth_in_px                   = $ARGV[2];
my $columnWidth_in_mm                   = (@ARGV > 6) ? $ARGV[6] : 31;
my $mm_per_pixel                        =
$columnWidth_in_mm/$columnWidth_in_px;
my $pixels_air                          = $ARGV[3];
my $pixels_bottom                       = $ARGV[4];
my $is_left_right                       = (@ARGV > 7) ? $ARGV[7] != 0 :
0;
my $printFramesMode                     = (@ARGV > 8) ? $ARGV[8] : 0;
my $printFramesBackground               = (@ARGV > 9) ? $ARGV[9] : 0;
my $particleSize                        = (@ARGV > 10) ? $ARGV[10] :
10;
my $trackImageStyle                     = (@ARGV > 11) ? $ARGV[11] : 0;

# Print input parameters on screen, to display that everything is
right.
print "Frame Rate: ",               $frame_rate,
      "\nColumn Width in pixel: ",  $columnWidth_in_px,
      "\nColumn Width in mm: ",     $columnWidth_in_mm,
      "\nmm per pixel: ",           $mm_per_pixel,
      "\nPixels air: ",             $pixels_air,
      "\nPixels bottom: ",          $pixels_bottom,
      "\nIs left right: ",          $is_left_right,
      "\nPrint Frame Mode: ",       $printFramesMode,
      "\nPrint Frame Background: ", $printFramesBackground,
      "\nParticle Size: ",          $particleSize,
      "\nTrack Image Style: ",      $trackImageStyle,
      "\n";

############################################################
# Initialize variables                                     #
############################################################

my $delta                                = 0;
my @Array_of_Scalar_Products             = ();
my @Array_of_All_Track_Scalar_Products = ();
my $size                                 = 0;
my $sum_cosine_tracks                    = 0;
my $avg_cosine_tracks                    = 0;
my @All_X_Moves                          = ();
my @All_Y_Moves                          = ();
my $avg_Y_move_tracks                    = 0;
my $sum_Y_move_tracks                    = 0;
my @positions                            = ();
my $pos_move_counter                     = 0;
my $neg_move_counter                     = 0;
my $Total_Y_Pos_Move                     = 0;
my $Total_Y_Neg_Move                     = 0;
my $Total_Y_Abs_Move                     = 0;
my $Total_Y_Move                         = 0;
my $Total_X_Move                         = 0;
my $Total_Move                           = 0;
my $Total_Pos_Move                       = 0;
my $Total_Neg_Move                       = 0;
my $Total_Abs_Move                       = 0;
my $Total_Plot_X_Move                    = 0;
```

```
my $Total_Plot_Y_Move                   = 0;
my @full_vector_angles                  = ();
my @trackStart                          = ();
my @averageDistances                    = ();
my @average_X_Moves                     = ();
my @average_X_Moves_Counter             = ();
my @average_Y_Moves                     = ();
my @average_Y_Moves_Counter             = ();
my @average_distances                   = ();
my @average_distances_Counter           = ();
my $median_X_Moves                      = 0;
my $median_Y_Moves                      = 0;
my $Total_Single_X_Move                 = 0;
my $Total_Single_X_Move_Counter         = 0;
my $Total_Single_X_Pos_Move             = 0;
my $Total_Single_X_Pos_Move_Counter     = 0;
my $Total_Single_X_Neg_Move             = 0;
my $Total_Single_X_Neg_Move_Counter     = 0;
my $Total_Single_X_Abs_Move             = 0;
my $Total_Single_X_Abs_Move_Counter     = 0;
my $Total_Single_X_Straightness         = 0;
my $Total_Single_Y_Move                 = 0;
my $Total_Single_Y_Move_Counter         = 0;
my $Total_Single_Y_Pos_Move             = 0;
my $Total_Single_Y_Pos_Move_Counter     = 0;
my $Total_Single_Y_Neg_Move             = 0;
my $Total_Single_Y_Neg_Move_Counter     = 0;
my $Total_Single_Y_Abs_Move             = 0;
my $Total_Single_Y_Abs_Move_Counter     = 0;
my $Total_Single_Y_Straightness         = 0;
my $Total_Single_Distance               = 0;
my $Total_Single_Distance_Counter       = 0;
my $all_track_pieces                    = 0;
my $upward_track_pieces                 = 0;
my $downward_track_pieces               = 0;
my $rightward_track_pieces              = 0;
my $leftward_track_pieces               = 0;
my $top_track_pieces                    = 0;
my $bottom_track_pieces                 = 0;
my $right_track_pieces                  = 0;
my $left_track_pieces                   = 0;

#########################################################
# Create Results file if there is none.                 #
#########################################################

# Get input file name without suffix
my($filename, $path, $suffix) = fileparse($infile, qr/\.[^.]*/);
my $basefile = File::Spec->catfile($path, $filename);

# Put the output result file into the same folder as the input files
my $ResultsFile = File::Spec->catfile($path, "Results.txt");

# Create the results file if it does not exist and write a header to
it
if(! -e $ResultsFile)
{
     open (RESULTS, ">$ResultsFile") or die "Error: ", print
"\nCannot open file!\n$ResultsFile! \n";
     print RESULTS  "File Name",
                    "\t#Vectors",
```

```
                    "\t#upward Vectors",
                    "\t#downward Vectors",
                    "\t#leftward Vectors",
                    "\t#rightward Vectors",
                    "\t%upward Vectors",
                    "\t%downward Vectors",
                    "\t%leftward Vectors",
                    "\t%rightward Vectors",
                    "\t#Average x Displacement", # Horizontal
displacement
                    "\t#Average y Displacement", #   Vertical
displacement
                    "\t#Average x positive Displacement",
                    "\t#Average y positive Displacement",
                    "\t#Average x negative Displacement",
                    "\t#Average y negative Displacement",
                    "\t#Average x absolute Displacement",
                    "\t#Average y absolute Displacement",
                    "\t#Average x Movement",
                    "\t#Average y Movement",
                    "\t#Average positive y Movement",
                    "\t#Average negative y Movement",
                    "\t#Average absolute y Movement",
                    "\t#Average Movement",
                    "\t#Average positive Movement",
                    "\t#Average negative Movement",
                    "\t#Average absolute Movement",
                    "\t#Larvae", # Number of larvae
                    "\t#Larvae Upper",
                    "\t#Larvae Lower",
                    "\t#Larvae % Upper",
                    "\t#Larvae % Lower",
                    "\t#Upward Speed (mm per sec)",
                    "\t#Downward Speed (mm per sec)",
                    "\t#Absolute Speed (mm per sec)",
                    "\t#Single Sum Speed (mm per sec)",
                    "\t#Sum Speed (mm per sec)",
                    "\t#Speed (mm per sec)", # Speed of the larvae
                    "\t#Median depth (mm from surface)",# Median
depth
                    "\t#Median depth (in %)",# Median depth
percentage from top to bottom
                    "\t#Median left/right (mm from middle)",
                    "\t#Median left/right (in % from middle)",
                    "\t#Simple X straightness",
                    "\t#Simple Y straightness",
                    "\t#Single X straightness",
                    "\t#Single Y straightness",
                    "\t#Average Angel",
                    "\t#Top Vectors",
                    "\t#Bottom Vectors",
                    "\t#Left Vectors",
                    "\t#Right Vectors",
                    "\t%Top Vectors",
                    "\t%Bottom Vectors",
                    "\t%Left Vectors",
                    "\t%Right Vectors",

                    "\t#Track Pieces",
                    "\t#Upward track pieces",
                    "\t#Downward track pieces",
```

```
                        "\t#Leftward track pieces",
                        "\t#Rightward track pieces",
                        "\t#Top track pieces",
                        "\t#Bottom track pieces",
                        "\t#Left track pieces",
                        "\t#Right track pieces",

                        "\t%Upward track pieces",
                        "\t%Downward track pieces",
                        "\t%Leftward track pieces",
                        "\t%Rightward track pieces",
                        "\t%Top track pieces",
                        "\t%Bottom track pieces",
                        "\t%Left track pieces",
                        "\t%Right track pieces",

                        "\t#Median X Movement",
                        "\t#Median Y Movement",

                        "\n";

      close RESULTS;
}

############################################################
# Calculate the vertical and horizontal distributions. #
############################################################
open (RESULTS_IN, "<$basefile"."_vertical.text_image.txt") or die
"Error: ", print "\nCannot open file!\n$! \n";

# Open and run through, to get the number of lines
while (<RESULTS_IN>) {}

my $no_of_lines = $.; # Get the number of lines
my $lineNumber = 0;
my @vertical_distribution = ();
my @horizontal_distribution = ();
my $sum_of_all_vertical_values = 0;
my $column_width = 0;

# Open the text-image from mTrack output
close RESULTS_IN;
open (RESULTS_IN, "<$basefile"."_vertical.text_image.txt") or die
"Error: ", print "\nCannot open file!\n$! \n";

# Calculate normalized distribution from the text-image
while (defined (my $line = <RESULTS_IN>))    # Sums up the pixel
values in the text_image file for each line separately
{
      my $sum=0;
      my @pixel_values = split (/\t/, $line);

      if($column_width == 0){ $column_width = scalar(@pixel_values);
}

      if
        (
             $lineNumber >= $pixels_air
          && $lineNumber <= $no_of_lines - $pixels_bottom
        )
      {
```

```
        my $i = 0;
        foreach (@pixel_values)
        {
                $sum += $_;
                $horizontal_distribution[$i] += $_;
                $i++;
        }

        $sum_of_all_vertical_values+=$sum;
       }

       $lineNumber++;

       push (@vertical_distribution, $sum);
}

# Normalize the vertical distribution
foreach (@vertical_distribution)
{
       # $_ is a reference
       if($sum_of_all_vertical_values > 0)
       {
        $_=$_/$sum_of_all_vertical_values;
       }
}

# Normalize the horizontal distribution
foreach (@horizontal_distribution)
{
       # The sum of all vertical values is identical to the sum of all
horizontal values
       if($sum_of_all_vertical_values > 0)
       {
        $_=$_/$sum_of_all_vertical_values;
       }
}

#########################################################
# Read in tracks and distances from mTrack output       #
#########################################################

# Read in all the tracks from the mTRack output file.
my @tracks = MTrack::ReadTracks($infile, $mTrackVersion);

# Figure out the length of the longest track in the dataset
my $track_length                        = 0;

for(my $t = 0; $t < scalar(@tracks); $t++)
{
       if($track_length < $#{$tracks[$t]}+1)
       {
        $track_length = $#{$tracks[$t]}+1;
       }
}

#########################################################
# Invalidate all the parts of the tracks that reach     #
# beyond the cutoff pixel values at the top and the     #
# bottom of the column. Split the tracks if necessary. #
#########################################################
for(my $t = scalar(@tracks)-1; $t >= 0; $t--)
```

```
{
      my $frameCounter = 0;
      my $continiousCounter = 0;
      my $continuity = 0;
      for(my $f = $#{$tracks[$t]}; $f >= 0; $f--)
      {
       # Save this into a local variable,
       # and use that for the testing. Otherwise
       # we run out of memory with big data sets.
       # Bug in perl.
       my $pos = $tracks[$t][$f];
       if
       (
            $pos->{isValid}
         &&
            (
                 $pos->{y} < $pixels_air
              || $pos->{y} > $no_of_lines - $pixels_bottom
            )
       )
       {
             $pos->{isValid} = 0;
       }

       if($pos->{isValid})
       {
             $frameCounter++;
             $continiousCounter++;
       }
       elsif(!$pos->{isValid} && $continiousCounter == 1)
       {
             my $pos2 = $tracks[$t][$f+1];
             $pos2->{isValid} = 0;
             $continiousCounter = 0;
             $frameCounter--;
       }
       elsif(!$pos->{isValid})
       {
             $continuity = max($continuity, $continiousCounter);
             $continiousCounter = 0;
       }
      }

      $continuity = max($continuity, $continiousCounter);

      if($frameCounter <= 1)
      {
       splice @tracks, $t, 1;
       next;
      }

      if($continuity < $frameCounter)
      {
       # Copy subarray, including its elements and not only the
references to those elements
       my @tracks_copy = @{dclone($tracks[$t])};

       $frameCounter      = 0;
       $continiousCounter = 0;

       for(my $f = $#{$tracks[$t]}; $f >= 0; $f--)
```

```
        {
                my $pos      = $tracks[$t][$f];
                my $pos_copy = $tracks_copy[$f];

                if($pos->{isValid})
                {
                        $frameCounter++;
                        $continiousCounter++;

                        if($frameCounter == $continiousCounter)
                        {
                                $pos_copy->{isValid} = 0;
                        }
                        else
                        {
                                $pos->{isValid} = 0;
                        }
                }
                else
                {
                        $continiousCounter = 0;
                }
        }
        unshift(@tracks, \@tracks_copy); # Insert at the beginning
        $t++; # Correct index, after insert
       }
}

# Exit if we have no tracks, but record the current input file in the
results file.
if(scalar(@tracks) == 0)
{
      open (RESULTS, ">>$ResultsFile") or die "Error: ", print
"\nCannot open file!\n$! \n";

      # Write the data to the results file.
      print RESULTS $infile, "\n";
      close RESULTS;

      print "Exit: No tracks found\n";
      exit;
}

############################################################
#                                                          #
# Create an image. It contains:                            #
# - The larval tracks in the column                        #
#   - The tracks are colored depending on time             #
#     - Red to yellow for upward tracks                    #
#     - Blue to cyan for downward tracks                   #
#   - The start and end points are connected by            #
#     vectors                                              #
# - The larval tracks aligned to a common origin           #
# - The larval vectors aligned to a common origin and      #
#   an average vector                                      #
# - The tracks broken into frame by frame pieces           #
#   multiplied by 10, with the end points plotted          #
#   around a common origin.                                #
# - The y component multiplied by 10 of the tracks         #
#   broken into frame by frame pieces plotted in           #
#   reference to a time axis.                              #
```

```
# - Average of the tracks broken into frame by frame   #
#   pieces over time per frame and multiplied by 10.   #
# - Culminated average multiplied by 10of the tracks   #
#   broken into frame by frame pieces over time.       #
# - Collect movement information for quantification.   #
#                                                      #
########################################################
my $no_of_tracks                        = scalar(@tracks);

# Scale the output image size according to the original video
dimensions, so that everything has space on it
my $img;
if($trackImageStyle == 0 || $trackImageStyle == 1)
{
      # For standart styles
      $img = GD::Simple->new(max($column_width*5 +50, $track_length +
50 + $column_width), $lineNumber*2 + 50, 1);
}
else
{
      # Use for thesis
      $img = GD::Simple->new($column_width*3.2 + 50, $lineNumber, 1);
}

# Scale the elements on the picture relatively to each other
my $averageOffset  = ({x => 0.5*$column_width+10, y =>
$lineNumber*1.8});
my $speedOffset    = ({x => 0.5*$column_width+10, y =>
$lineNumber*1.5});
my $scaleOffset    = ({x => 1.0*$column_width+10, y => 0});

my $centerOffset;
my $vectorOffset;

if($trackImageStyle == 0)
{
      # Use for debugging and trouble shooting:
      $centerOffset   = ({x => 2.0*$column_width+20, y =>
$lineNumber+25});
      $vectorOffset   = ({x => 3.5*$column_width+30, y =>
$lineNumber+25});
}
elsif($trackImageStyle == 1)
{
      # Use for presentations:
      $centerOffset   = ({x => 2.7*$column_width+20, y =>
$lineNumber/2+25});
      $vectorOffset   = ({x => 4.4*$column_width+30, y =>
$lineNumber/2+25});
}
else
{
      # Use for thesis:
      $centerOffset   = ({x => 1.9*$column_width+20, y =>
$lineNumber/2+25});
      $vectorOffset   = ({x => 2.8*$column_width+30, y =>
$lineNumber/2+25});
}

print "Number of tracks: ", $no_of_tracks, "\n";
print "Track length: ", $track_length, "\n";
```

```
# Print the axes crossing at the center of the tracks starting from a
common origin
$img->bgcolor(0,0,0);
$img->fgcolor(0,0,0);
$img->penSize(1);
$img->moveTo($speedOffset->{x},     $speedOffset->{y}+200);
$img->lineTo($speedOffset->{x},     $speedOffset->{y}-200);
$img->moveTo($speedOffset->{x}-200, $speedOffset->{y});
$img->lineTo($speedOffset->{x}+200, $speedOffset->{y});


$img->moveTo($averageOffset->{x}*2,     $averageOffset->{y});
$img->lineTo(max($column_width*4 +50, $track_length + 50 +
$column_width),     $averageOffset->{y});

# Draw temporal axes with ticks and labels after every 30 seconds
for(my $i = 0; $i < $track_length; $i += $frame_rate * 30)
{
      $img->moveTo($averageOffset->{x}*2 + $i,     $averageOffset-
>{y}+200);
      $img->lineTo($averageOffset->{x}*2 + $i,     $averageOffset-
>{y}-200);
      $img->moveTo(  $speedOffset->{x}*2 + $i,     $speedOffset-
>{y}+200);
      $img->lineTo(  $speedOffset->{x}*2 + $i,     $speedOffset->{y}-
200);
      $img->moveTo(  $speedOffset->{x}*2 + $i +10, $speedOffset-
>{y}+200);
      $img->string($i . " frame");
      $img->moveTo(  $speedOffset->{x}*2 + $i +10, $speedOffset-
>{y}+180);
      $img->string($i/$frame_rate . "\'");
      $img->moveTo(  $speedOffset->{x}*2 + $i +10, $speedOffset-
>{y}+160);
      $img->string($i/$frame_rate/60 . "\'\'");
}

# Set the size before, saves expansive resizing
$averageDistances[scalar(@tracks)-1] = 0;
       $trackStart[scalar(@tracks)-1] = 0;

$average_X_Moves          [$track_length-1] = 0;
$average_X_Moves_Counter  [$track_length-1] = 0;
$average_Y_Moves          [$track_length-1] = 0;
$average_Y_Moves_Counter  [$track_length-1] = 0;
$average_distances        [$track_length-1] = 0;
$average_distances_Counter[$track_length-1] = 0;

# Init values
for(my $f = 0; $f < $track_length; $f++)
{
      $average_X_Moves          [$f] = 0;
      $average_X_Moves_Counter  [$f] = 0;
      $average_Y_Moves          [$f] = 0;
      $average_Y_Moves_Counter  [$f] = 0;
      $average_distances        [$f] = 0;
      $average_distances_Counter[$f] = 0;
}

# Draw the tracks and the vectors
```

```
for(my $t = 0; $t < scalar(@tracks); $t++)
{
      my $lastPos           = ({x => 0, y => 0});
      my $initPos           = ({x => 0, y => 0});
      my $gotFirstFrame     = 0;
      my $distanceCounter   = 0;
      $averageDistances[$t] = 0;

      for(my $f = 0; $f < $#{$tracks[$t]}+1; $f++)
      {
       # Put this into a local variable, so that perl does not freak
out at big data sets.
       my $pos = $tracks[$t][$f];
       # Get the start frame of the current track
       if($pos->{isValid} && !$gotFirstFrame)
       {
             $trackStart[$t] = $f;

             $gotFirstFrame = 1;
             $lastPos = ({x => $pos->{x}, y => $pos->{y} -
$pixels_air});
             $initPos = ({x => $pos->{x}, y => $pos->{y} -
$pixels_air});

             # Go to next loop iteration
             next;
       }

       # Last frame with track reached, so leave the inner loop
       if(!$pos->{isValid} && $gotFirstFrame)
       {
             # Leave loop
             last;
       }

       # Paint the tracks in the column and from a common origin
       if($gotFirstFrame)
       {
             my $thisPos = ({x => $pos->{x}, y => $pos->{y} -
$pixels_air});

             my $x = $thisPos->{x} - $lastPos->{x};
             my $y = $thisPos->{y} - $lastPos->{y};

             my $gamma = atan2(-$y, $x);
             $all_track_pieces++;

             # Score the number of up and down, and right and left
track pieces from the angles
             if($gamma < 7*pi/12 && $gamma > 5*pi/12)
             {
                   $top_track_pieces++;
             }
             if($gamma > -7*pi/12 && $gamma < -5*pi/12)
             {
                   $bottom_track_pieces++;
             }
             if(abs($gamma) - pi/2 < 7*pi/12 && abs($gamma) - pi/2 >
5*pi/12)
             {
                   $left_track_pieces++;
```

```
            }
            if(abs($gamma) - pi/2 > -7*pi/12 && abs($gamma) - pi/2 <
-5*pi/12)
            {
                  $right_track_pieces++;
            }

            # Score the number of upward and downward pointing track
pieces from the vector angles
            if($gamma > 0)
            {
                  $upward_track_pieces++;
            }
            else
            {
                  $downward_track_pieces++;
            }

            # Correct way to turn the angle:
            # If $_ >= 0:   $_ - pi/2
            # If $_ <  0: -($_ + pi/2)
            # But the result is the same
            if(abs($gamma) - pi/2 > 0)
            {
                  $leftward_track_pieces++;
            }
            else
            {
                  $rightward_track_pieces++;
            }

            my $distance = sqrt($x**2 + $y**2);

            # Average the distances the larvae traveled
            $averageDistances[$t] += $distance;
            $distanceCounter++;

            # Average the x component the larvae traveled
            $average_X_Moves[$f] += $x;
            $average_X_Moves_Counter[$f]++;
            # Average the y component the larvae traveled
            $average_Y_Moves[$f] += $y;
            $average_Y_Moves_Counter[$f]++;
            # Average the distances the larvae traveled
            $average_distances[$f] += $distance;
            $average_distances_Counter[$f]++;
            push (@All_X_Moves, $x);
            push (@All_Y_Moves, $y);

            if($x >= 0)
            {
                  $Total_Single_X_Pos_Move += $x;
                  $Total_Single_X_Pos_Move_Counter++;
            }
            else
            {
                  $Total_Single_X_Neg_Move += $x;
                  $Total_Single_X_Neg_Move_Counter++;
            }
```

```
            # Up is positive, but the y-axis is reversed on screen.
So change sign.
            if($y <= 0)
            {
                  $Total_Single_Y_Pos_Move -= $y;
                  $Total_Single_Y_Pos_Move_Counter++;
            }
            else
            {
                  $Total_Single_Y_Neg_Move -= $y;
                  $Total_Single_Y_Neg_Move_Counter++;
            }

            $Total_Single_X_Abs_Move += abs($x);
            $Total_Single_X_Abs_Move_Counter++;
            $Total_Single_Y_Abs_Move += abs($y);
            $Total_Single_Y_Abs_Move_Counter++;

            $Total_Single_X_Straightness += NormalizeToCount($x,
$distance);
            $Total_Single_Y_Straightness -= NormalizeToCount($y,
$distance);
            $Total_Single_Distance += $distance;
            $Total_Single_Distance_Counter++;

            # This disturbs the vector plotting with some tracks
            # But this seems to be a problem of the GD::Simple
library
            # Color tracks pointing upwards from red to yellow
depending on time
            # Color tracks pointing downwards from blue to cyan
depending on time
            my $red;
            my $green;
            my $blue;

            if($is_left_right)
            {
                  $red   = ($x >= 0) ? 255 : 0;
                  $green = round($f*(255/($track_length)));
                  $blue  = ($x <  0) ? 255 : 0;
            }
            else
            {
                  $red   = ($y <= 0) ? 255 : 0;
                  $green = round($f*(255/($track_length)));
                  $blue  = ($y >  0) ? 255 : 0;
            }

            $img->penSize(2);
            $img->fgcolor($red, $green, $blue);
            $img->bgcolor($red, $green, $blue);

            # Draw the frame by frame vector ends from a common
origin multiplied by 10
            $img->moveTo($speedOffset->{x} + $x*10, $speedOffset->{y}
+ $y*10);
            $img->ellipse(2,2);
            # Draw the frame by frame vector ends over time
multiplied by 10
```

```
            $img->moveTo($speedOffset->{x}*2 + $f, $speedOffset->{y}
+ $y*10);
            $img->ellipse(2,2);

            # Draw the tracks in the column, multiply by 10 for
debugging
            $img->moveTo($lastPos->{x}, $lastPos->{y});
            $img->lineTo($thisPos->{x}, $thisPos->{y});
     #            $img->moveTo($lastPos->{x}*10, $lastPos->{y}*10);
     #            $img->lineTo($thisPos->{x}*10, $thisPos->{y}*10);

            # Prepare to draw the tracks starting from a common point
            $thisPos->{x} -= $initPos->{x};
            $thisPos->{y} -= $initPos->{y};
            $lastPos->{x} -= $initPos->{x};
            $lastPos->{y} -= $initPos->{y};

            # Multiply by 10 for debugging
     #            $thisPos->{x} *= 10;
     #            $thisPos->{y} *= 10;
     #            $lastPos->{x} *= 10;
     #            $lastPos->{y} *= 10;

            $thisPos->{x} += $centerOffset->{x};
            $thisPos->{y} += $centerOffset->{y};
            $lastPos->{x} += $centerOffset->{x};
            $lastPos->{y} += $centerOffset->{y};

            # Draw the tracks starting from a common point
            $img->moveTo($lastPos->{x}, $lastPos->{y});
            $img->lineTo($thisPos->{x}, $thisPos->{y});

            $lastPos = ({x => $pos->{x}, y => $pos->{y} -
$pixels_air});
      }
     }

     $averageDistances[$t] /= $distanceCounter;

     # Note y coordinate 0 is on top of the screen, and next pixel
line down is 1
     if($lastPos->{y} - $initPos->{y} > 0)
     {
      $averageDistances[$t] = -$averageDistances[$t];
     }

     # Plot the vectors in the column
     $img->penSize(1);
     $img->bgcolor('gray'); # British spelling!!!
     $img->fgcolor('gray'); # British spelling!!!
     $img->moveTo($initPos->{x}, $initPos->{y}); #we move to the
beginning of each track
     $img->lineTo($lastPos->{x}, $lastPos->{y}); #draw a line to the
end of each track

     my $X_move  = $lastPos->{x};
     my $Y_move  = $lastPos->{y};
        $X_move -= $initPos->{x};
        $Y_move -= $initPos->{y};

     # Plot the vectors starting from a common origin
```

```
      $img->moveTo($vectorOffset->{x}, $vectorOffset->{y});
      $img->lineTo($vectorOffset->{x} + $X_move, $vectorOffset->{y} +
$Y_move);

      $Total_Plot_X_Move     += $X_move;
      $Total_Plot_Y_Move     += $Y_move; # See note below

      $X_move /= $distanceCounter;
      $Y_move /= $distanceCounter;

      # Calculate the angle of the vector with respect to a
horizontal (0,-1) line
      # Take the negative of $Y_move, since bigger y values mean on a
screen go down, while in geometry mean go up
      my $gamma = atan2(-$Y_move, $X_move) - atan2(0,1); # atan2(0,1)
actually 0
      push (@full_vector_angles, $gamma);
      #here we calculate the total movement
      $Total_X_Move     += $X_move;
      $Total_Y_Move     -= $Y_move; # See note below
      $Total_Y_Abs_Move += abs($Y_move);

      my $move = sqrt($X_move**2 + $Y_move**2);
      $Total_Abs_Move += $move;

      # Collect movement information
      # Note the line of pixels at the top has y value 0, next line
below has 1, so the signs must be revered
      if($Y_move <= 0)
      {
       $Total_Move       += $move;
       $Total_Pos_Move   += $move;
       $Total_Y_Pos_Move -= $Y_move;
       $pos_move_counter++;
      }
      else
      {
       $Total_Move       -= $move;
       $Total_Neg_Move   -= $move;
       $Total_Y_Neg_Move -= $Y_move;
       $neg_move_counter++;
      }

      # Print a red dot at the end of the current track
      $img->bgcolor(255,0,0);
      $img->fgcolor(255,0,0);
      $img->moveTo($lastPos->{x}, $lastPos->{y});
      $img->ellipse(4,4);

      # Print a red dot at the end of the current track starting from
a common point
      $lastPos->{x} -= $initPos->{x};
      $lastPos->{y} -= $initPos->{y};
      $lastPos->{x} += $centerOffset->{x};
      $lastPos->{y} += $centerOffset->{y};

      $img->moveTo($lastPos->{x}, $lastPos->{y});
      $img->ellipse(4,4);
}
```

```
#here we print the axes crossing at the center of the tracks starting
from a common point
$img->bgcolor(0,0,0);
$img->fgcolor(0,0,0);
$img->penSize(1);
$img->moveTo($centerOffset->{x},     $centerOffset->{y}+200);
$img->lineTo($centerOffset->{x},     $centerOffset->{y}-200);

if($trackImageStyle == 0 || $trackImageStyle == 1)
{
      # Use for debugging or presentation
      $img->moveTo($centerOffset->{x}-200, $centerOffset->{y});
      $img->lineTo($centerOffset->{x}+200, $centerOffset->{y});
}
else
{
      # Use for thesis
      $img->moveTo($centerOffset->{x}-50, $centerOffset->{y});
      $img->lineTo($centerOffset->{x}+50, $centerOffset->{y});
}

# Draw the bottom of the column
$img->penSize(1);
$img->bgcolor('gray'); # British spelling!!!
$img->fgcolor('gray'); # British spelling!!!
$img->moveTo(0   , $no_of_lines - $pixels_bottom);
$img->lineTo($column_width, $no_of_lines - $pixels_bottom);

# Plot the average movement vector
$img->penSize(3);
$img->bgcolor('red');
$img->fgcolor('red');
$img->moveTo($vectorOffset->{x},$vectorOffset->{y});

if($no_of_tracks == 0){ $no_of_tracks = 1;}
$img->lineTo($vectorOffset->{x} + $Total_Plot_X_Move / $no_of_tracks,
$vectorOffset->{y} + $Total_Plot_Y_Move / $no_of_tracks);


# Calculate the number of ticks needed
my $numLabels       = 10;
my $depth_mm        = $lineNumber * $mm_per_pixel;
my $tick_interval   = BestTick($depth_mm, $numLabels);
my $max_tick        = floor($depth_mm / $tick_interval);
my $pixels_per_tick = $tick_interval/$mm_per_pixel;

# Draw the axes
$img->penSize(3);
$img->bgcolor('black');
$img->fgcolor('black');
$img->moveTo($scaleOffset->{x},$scaleOffset->{y});
$img->lineTo($scaleOffset->{x},$lineNumber);

$img->penSize(3);
$img->font('Arial Bold');
$img->fontsize(16);

# Draw the ticks
for(my $i = 0; $i <= $max_tick; $i++)
{
      my $ypos = $scaleOffset->{y} + $pixels_per_tick*$i;
```

```
      $img->moveTo($scaleOffset->{x}+5, $ypos);
      $img->lineTo($scaleOffset->{x},   $ypos);
      $img->moveTo($scaleOffset->{x}+10,$ypos + (($i==0)?14:8));
      $img->string($i*$tick_interval);
}

# Draw averages by frame and cumulative average by frame, and save
file final average for later use.
for(my $f = 0; $f < $track_length; $f++)
{
      $Total_Single_X_Move         += $average_X_Moves[$f];
      $Total_Single_X_Move_Counter += $average_X_Moves_Counter[$f];
      $Total_Single_Y_Move         -= $average_Y_Moves[$f]; # 0 on y-
axis is on top, so reverse sign
      $Total_Single_Y_Move_Counter += $average_Y_Moves_Counter[$f];

      my $y  = $average_Y_Moves[$f];
      if($average_Y_Moves_Counter[$f] > 0)
      {
       $y /= $average_Y_Moves_Counter[$f];
      }
      else
      {
       $y = 0;
      }

      my $red   = ($y <= 0) ? 255 : 0;
      my $green = round($f*(255/($track_length)));
      my $blue  = ($y >  0) ? 255 : 0;

      $img->penSize(2);
      $img->fgcolor($red, $green, $blue);
      $img->bgcolor($red, $green, $blue);

      $img->moveTo($averageOffset->{x}*2 + $f, $averageOffset->{y} +
$y*10);
      $img->ellipse(2,2);

      $y = -$Total_Single_Y_Move;
      if($Total_Single_Y_Move_Counter > 0)
      {
       $y /= $Total_Single_Y_Move_Counter;
      }
      else
      {
       $y = 0;
      }

      $red   = ($y <= 0) ? 255 : 0;
      $green = round($f*(255/($track_length)));
      $blue  = ($y >  0) ? 255 : 0;

      $img->penSize(2);
      $img->fgcolor($red, $green, $blue);
      $img->bgcolor($red, $green, $blue);

      $img->moveTo($averageOffset->{x}*2 + $f, $averageOffset-
>{y}+150 + $y*10);
      $img->ellipse(2,2);
```

```
      $average_X_Moves  [$f]         = NormalizeToCount(
$average_X_Moves[$f],           $average_X_Moves_Counter[$f]);
      $average_Y_Moves  [$f]         = NormalizeToCount(-
$average_Y_Moves[$f],           $average_Y_Moves_Counter[$f]);
      $average_distances[$f]         = NormalizeToCount(
$average_distances[$f],         $average_distances_Counter[$f]);
}

my $stats = Statistics::Descriptive::Full->new();
$stats->add_data(@All_X_Moves);
$median_X_Moves = $stats->median();
$stats = Statistics::Descriptive::Full->new();
$stats->add_data(@All_Y_Moves);
$median_Y_Moves = $stats->median();

$Total_Single_X_Move         = NormalizeToCount($Total_Single_X_Move,
$Total_Single_X_Move_Counter);
$Total_Single_Y_Move         = NormalizeToCount($Total_Single_Y_Move,
$Total_Single_Y_Move_Counter);
$Total_Single_X_Pos_Move     =
NormalizeToCount($Total_Single_X_Pos_Move,
$Total_Single_X_Pos_Move_Counter);
$Total_Single_Y_Pos_Move     =
NormalizeToCount($Total_Single_Y_Pos_Move,
$Total_Single_Y_Pos_Move_Counter);
$Total_Single_X_Neg_Move     =
NormalizeToCount($Total_Single_X_Neg_Move,
$Total_Single_X_Neg_Move_Counter);
$Total_Single_Y_Neg_Move     =
NormalizeToCount($Total_Single_Y_Neg_Move,
$Total_Single_Y_Neg_Move_Counter);
$Total_Single_X_Abs_Move     =
NormalizeToCount($Total_Single_X_Abs_Move,
$Total_Single_X_Abs_Move_Counter);
$Total_Single_Y_Abs_Move     =
NormalizeToCount($Total_Single_Y_Abs_Move,
$Total_Single_Y_Abs_Move_Counter);
$Total_Single_Distance       =
NormalizeToCount($Total_Single_Distance,
$Total_Single_Distance_Counter);
$Total_Single_X_Straightness =
NormalizeToCount($Total_Single_X_Straightness,
$Total_Single_Distance_Counter);
$Total_Single_Y_Straightness =
NormalizeToCount($Total_Single_Y_Straightness,
$Total_Single_Distance_Counter);

# Write the image data just created to a file
open (IMAGE_OUT, ">$basefile"."_tracks.png") or die "Error: ", print
"\nCannot open file!\n$! \n";
print IMAGE_OUT $img->png;
close IMAGE_OUT;

#########################################################
# Count the upward and downward pointing vectors and    #
# write the results to a file.                          #
#########################################################
my $upward_vectors    = 0;
my $downward_vectors  = 0;
my $rightward_vectors = 0;
my $leftward_vectors  = 0;
```

```
my $top_vectors        = 0;
my $bottom_vectors     = 0;
my $right_vectors      = 0;
my $left_vectors       = 0;

foreach (@full_vector_angles)
{
      if($_ < 7*pi/12 && $_ > 5*pi/12)
      {
       $top_vectors++;
      }
      if($_ > -7*pi/12 && $_ < -5*pi/12)
      {
       $bottom_vectors++;
      }
      if(abs($_) - pi/2 < 7*pi/12 && abs($_) - pi/2 > 5*pi/12)
      {
       $left_vectors++;
      }
      if(abs($_) - pi/2 > -7*pi/12 && abs($_) - pi/2 < -5*pi/12)
      {
       $right_vectors++;
      }

      # Score the number of upward and downward pointing vectors from
the vector angles
      if($_ > 0)
      {
       $upward_vectors++;
      }
      else
      {
       $downward_vectors++;
      }

      # Correct way to turn the angle:
      # If $_ >= 0:   $_ - pi/2
      # If $_ <  0: -($_ + pi/2)
      # But the result is the same
      if(abs($_) - pi/2 > 0)
      {
       $leftward_vectors++;
      }
      else
      {
       $rightward_vectors++;
      }
}

############################################################
# Find the depth corresponding to the median of the    #
# distribution                                             #
############################################################
my $density_till_mean=0;
my $mean_depth_counter=0;
for (my $i=0; $i<$no_of_lines; $i++)
{
      $density_till_mean+=$vertical_distribution[$i];
      $mean_depth_counter++;
      if ($density_till_mean>=0.5)
      {
```

```
      last;
      }
}

#############################################################
# Find the center corresponding to the median of the        #
# distribution                                              #
#############################################################
$density_till_mean=0;
my $mean_left_right_counter=0;
for (my $i=0; $i < scalar(@horizontal_distribution); $i++)
{
      $density_till_mean+=$horizontal_distribution[$i];
      $mean_left_right_counter++;
      if ($density_till_mean>=0.5)
      {
       last;
      }
}

#############################################################
# Create an image of the larval depth distribution          #
# with a median distribution bar. The image is scaled       #
# according to the original dimensions of the input         #
# video. The distribution image is mainly useful for        #
# debugging.                                                #
#############################################################
my $depthScaleOffset    = ({x => 50, y => 50});

# Crate a drawing object
$img = GD::Simple->new(325, $depthScaleOffset->{y}*2 + $lineNumber);

# Draw the distribution
$img->penSize(3);
$img->bgcolor('red');
$img->fgcolor('red');

my $y_offset = $depthScaleOffset->{y} - $pixels_air;
my $counter  = 0;

# Go trough the image line by line and draw
foreach (@vertical_distribution)
{
      $counter++;
      unless ($_==0)
      {
       $img->moveTo($depthScaleOffset->{x}         , $y_offset +
$counter);
       $img->lineTo($depthScaleOffset->{x}+$_*10000, $y_offset +
$counter);
      }
}

# Draw the line corresponding to the median depth
$img->penSize(3);
$img->bgcolor('black');
$img->fgcolor('black');
$img->moveTo($depthScaleOffset->{x}+ 70,$y_offset +
$mean_depth_counter);
$img->lineTo($depthScaleOffset->{x}+150,$y_offset +
$mean_depth_counter);
```

```
# Draw the bottom of the column
$img->penSize(1);
$img->bgcolor('gray'); # British spelling!
$img->fgcolor('gray'); # British spelling!
$img->moveTo($depthScaleOffset->{x} , $y_offset + $no_of_lines -
$pixels_bottom);
$img->lineTo($depthScaleOffset->{x} + 200, $y_offset + $no_of_lines -
$pixels_bottom);

# Draw of the axes
$img->penSize(3);
$img->bgcolor('black');
$img->fgcolor('black');
$img->moveTo($depthScaleOffset->{x},$depthScaleOffset->{y});
$img->lineTo($depthScaleOffset->{x},$depthScaleOffset-
>{y}+$lineNumber);
$img->moveTo($depthScaleOffset->{x},$depthScaleOffset-
>{y}+$lineNumber);
$img->lineTo($depthScaleOffset->{x}+200,$depthScaleOffset-
>{y}+$lineNumber);

# Draw the ticks
$img->font('Arial Bold');
$img->fontsize(30);
for(my $i = 0; $i <= $max_tick; $i++)
{
      my $ypos = $depthScaleOffset->{y} + $pixels_per_tick*$i;
      $img->moveTo($depthScaleOffset->{x}-5, $ypos);
      $img->lineTo($depthScaleOffset->{x},   $ypos);
      $img->moveTo($depthScaleOffset->{x}-49,$ypos+14);# +
(($i==0)?14:8));
      $img->string($i*$tick_interval);
}

$img->moveTo($depthScaleOffset->{x}+100,$depthScaleOffset-
>{x}+$lineNumber);
$img->lineTo($depthScaleOffset->{x}+100,$depthScaleOffset-
>{x}+5+$lineNumber);
$img->moveTo($depthScaleOffset->{x}+200,$depthScaleOffset-
>{x}+$lineNumber);
$img->lineTo($depthScaleOffset->{x}+200,$depthScaleOffset-
>{x}+5+$lineNumber);

# Draw the x-axis labels, not scalable
$img->font('Arial Bold');
$img->fontsize(20);
$img->moveTo($depthScaleOffset->{x}-8,$depthScaleOffset-
>{y}+27+$lineNumber);
$img->string("0");
$img->moveTo($depthScaleOffset->{x}-8+80,$depthScaleOffset-
>{y}+27+$lineNumber);
$img->string("0.01");
$img->moveTo($depthScaleOffset->{x}-8+180,$depthScaleOffset-
>{y}+27+$lineNumber);
$img->string("0.02");

# Write the image data to a file
open (IMAGE_OUT, ">$basefile"."_distr".".png") or die "Error: ",
print "\nCannot open file!\n$! \n";
print IMAGE_OUT $img->png;
```

```
close IMAGE_OUT;

if($printFramesMode > 0)
{
      open (RESULTS_FRAMES, ">$basefile"."_displacement.txt") or die
"Error: ", print "\nCannot open file!\n$! \n";

      $img = GD::Simple->new($column_width, $lineNumber, 1);
      $img->penSize(3);

      my $img_points = GD::Simple->new($column_width, $lineNumber,
1);
      $img_points->penSize(3);

      if($printFramesBackground)
      {
       $img->bgcolor('black');
       $img->fgcolor('black');
      }
      else
      {
       $img->bgcolor('white');
       $img->fgcolor('white');
      }

      $img->rectangle(0, 0, $column_width, $lineNumber);

      for(my $f = 0; $f < $track_length; $f++)
      {
       if($printFramesBackground)
       {
             $img_points->bgcolor('black');
             $img_points->fgcolor('black');
       }
       else
       {
             $img_points->bgcolor('white');
             $img_points->fgcolor('white');
       }

       $img_points->rectangle(0, 0, $column_width, $lineNumber);

       for(my $t = 0; $t < scalar(@tracks); $t++)
       {
             my $pos = $tracks[$t][$f];
             if($pos->{isValid})
             {
                   my $x = $pos->{x};
                   my $y = $pos->{y};

                   if($particleSize < 5)
                   {
       #                  $x = round($x);
       #                  $y = round($y);
                   }

                   my $red   = 0;
                   my $green = 0;
                   my $blue  = 0;

                   # Temporal encoding
```

```
                    if($printFramesMode == 1)
                    {
                          my $numOpts = 5;
                          my $denom = $track_length/$numOpts;
                          my $variableValue = round(($f %
$denom)*(255/$denom));

                          if($f < $denom)
                          {
                                $red   = 255;
                                $green = $variableValue;
                                $blue  = 0;
                          }
                          elsif($f < 2*$denom)
                          {
                                $red   = 255 - $variableValue;
                                $green = 255;
                                $blue  = 0;
                          }
                          elsif($f < 3*$denom)
                          {
                                $red   = 0;
                                $green = 255;
                                $blue  = $variableValue;
                          }
                          elsif($f < 4*$denom)
                          {
                                $red   = 0;
                                $green = 255 - $variableValue;
                                $blue  = 255;
                          }
                          elsif($f < 5*$denom)
                          {
                                $red   = $variableValue;
                                $green = 0;
                                $blue  = 255;
                          }

                          #if($f < $denom)
                          #{
                          #     $red   = 0;
                          #     $green = 255 - $variableValue;
                          #     $blue  = 255;
                          #}
                          #elsif($f < 2*$denom)
                          #{
                          #     $red   = $variableValue;
                          #     $green = 0;
                          #     $blue  = 255;
                          #}
                          #elsif($f < 3*$denom)
                          #{
                          #     $red   = 255;
                          #     $green = 0;
                          #     $blue  = 255 - $variableValue;
                          #}
                          #elsif($f < 4*$denom)
                          #{
                          #     $red   = 255;
                          #     $green = $variableValue;
                          #     $blue  = 0;
```

```
                        #}
                        #elsif($f < 5*$denom)
                        #{
                        #       $red   = 255 - $variableValue;
                        #       $green = 255;
                        #       $blue  = 0;
                        #}
                }
                # Angular encoding
                elsif($printFramesMode == 2)
                {
                        my $x1 = 0;
                        my $y1 = 0;

                        if($f > 0)
                        {
                                my $lastPos = $tracks[$t][$f-1];
                                if($lastPos->{isValid})
                                {
                                        $x1 = $lastPos->{x};
                                        $y1 = $lastPos->{y};
                                }
                        }

                        my $gamma = 0;
                        if($is_left_right == 0)
                        {
                                $gamma = round((atan2(($y-$y1), $x-$x1)
+ pi) * 1000000000);
                        }
                        else
                        {
                                $gamma = round((atan2(-($y-$y1), $x-
$x1) + pi) * 1000000000)
                        }
        #               my $gamma = round((atan2(-($y-$y1), $x-$x1) +
pi) * 1000000000); # Integerize
        #               my $gamma = round((atan2(($y-$y1), $x-$x1) +
pi) * 1000000000); # Integerize
                        my $numOpts = 6;
                        my $denom = round(2*pi / $numOpts *
1000000000); # Integerize
                        my $variableValue = round(($gamma %
$denom)*(255/$denom));

        #               print $f, "\t";
        #               print $x-$x1, "\t";
        #               print $y-$y1, "\t";
        #               print $gamma, "\t";
        #               print $numOpts, "\t";
        #               print $denom, "\t";
        #               print $variableValue, "\n";

                        if($gamma < $denom)
                        {
                                $red   = 255;
                                $green = $variableValue;
                                $blue  = 0;
                        }
                        elsif($gamma < 2*$denom)
                        {
```

```
                                $red   = 255 - $variableValue;
                                $green = 255;
                                $blue  = 0;
                        }
                        elsif($gamma < 3*$denom)
                        {
                                $red   = 0;
                                $green = 255;
                                $blue  = $variableValue;
                        }
                        elsif($gamma < 4*$denom)
                        {
                                $red   = 0;
                                $green = 255 - $variableValue;
                                $blue  = 255;
                        }
                        elsif($gamma < 5*$denom)
                        {
                                $red   = $variableValue;
                                $green = 0;
                                $blue  = 255;
                        }
                        elsif($gamma < 6*$denom)
                        {
                                $red   = 255;
                                $green = 0;
                                $blue  = 255 - $variableValue;
                        }
                }
                # Temporal up and down encoding
                elsif($printFramesMode > 2)
                {
                        my $x1 = 0;
                        my $y1 = 0;

                        if($f > 0)
                        {
                                my $lastPos = $tracks[$t][$f-1];
                                if($lastPos->{isValid})
                                {
                                        $x1 = $lastPos->{x};
                                        $y1 = $lastPos->{y};
                                }
                        }

                        if($is_left_right)
                        {
                                $red   = ($x-$x1 >= 0) ? 255 : 0;
                                $green =
round($f*(255/($track_length)));
                                $blue  = ($x-$x1 <  0) ? 255 : 0;
                        }
                        else
                        {
                                $red   = ($y-$y1 <= 0) ? 255 : 0;
                                $green =
round($f*(255/($track_length)));
                                $blue  = ($y-$y1 >  0) ? 255 : 0;
                        }
                }
                # Crate a drawing object
```

```
                    $img->moveTo($x, $y);

                    $img->fgcolor($red, $green, $blue);
                    $img->bgcolor($red, $green, $blue);

                    $img->ellipse($particleSize, $particleSize);

                    $img_points->moveTo($x, $y);

                    $img_points->fgcolor($red, $green, $blue);
                    $img_points->bgcolor($red, $green, $blue);

                    $img_points->ellipse($particleSize, $particleSize);

                    # Write the image data just created to a file
               }
          }

          if($printFramesBackground < 2)
          {
               open (IMAGE_OUT, ">$basefile"."_tracks_$f.png") or die
"Error: ", print "\nCannot open file!\n$! \n";
               print IMAGE_OUT $img->png;
               close IMAGE_OUT;

               open (IMAGE_OUT, ">$basefile"."_track_points_$f.png") or
die "Error: ", print "\nCannot open file!\n$! \n";
               print IMAGE_OUT $img_points->png;
               close IMAGE_OUT;
          }

          print RESULTS_FRAMES $f, "\t",
                               $average_X_Moves[$f]   * $mm_per_pixel *
$frame_rate, "\t",
                               $average_Y_Moves[$f]   * $mm_per_pixel *
$frame_rate, "\t",
                               $average_distances[$f] * $mm_per_pixel *
$frame_rate, "\n";
          }
          close RESULTS_FRAMES;
}

#########################################################
# Estimate the number of larvae in the column. The      #
# maximum number of tracks in any frame gives the       #
# minimum number of larvae in the column.               #
#########################################################
my $number_of_larvae_in_column = 0;

for(my $f = 0; $f < $track_length; $f++)
{
          my $validCounter = 0;
          for(my $t = 0; $t < scalar(@tracks); $t++)
          {
           # That's now really ridiculous, do you really need
           # to use up all the memory for dereferencing?
           my $pos = $tracks[$t][$f];
           if
           (
                $pos->{isValid}
           )
```

```
        {
                $validCounter++;
        }
       }

       if($validCounter > $number_of_larvae_in_column)
       {
        $number_of_larvae_in_column = $validCounter;
       }
}

#########################################################
# Calculate the number of larvae in the upper half of  #
# the column and the lower half, write the results to  #
# a file.                                               #
#########################################################

# Calculate the larvae in the upper half of the chamber, using the
$no_of_lines/2 and the number of larvae as estimated by the short
tracks
my $larvae_in_upper_half=0;
for (my $i=$pixels_air; $i < round(($no_of_lines - $pixels_bottom -
$pixels_air)/2 + $pixels_air); $i++)
{
      $larvae_in_upper_half+=$vertical_distribution[$i];
}

my $larvae_in_upper_half_percent = $larvae_in_upper_half;
$larvae_in_upper_half=round($larvae_in_upper_half*$number_of_larvae_i
n_column);

# Calculate the larvae in the upper lower of the chamber, using the
$no_of_lines/2 and the number of larvae as estimated by the short
tracks
my $larvae_in_lower_half=0;
for (my $i=round(($no_of_lines - $pixels_bottom - $pixels_air)/2 +
$pixels_air); $i<$no_of_lines - $pixels_bottom; $i++)
{
      $larvae_in_lower_half+=$vertical_distribution[$i];
}

my $larvae_in_lower_half_percent = $larvae_in_lower_half;
$larvae_in_lower_half=round($larvae_in_lower_half*$number_of_larvae_i
n_column);

#########################################################
# Calculate means for later saving to a results file.  #
#########################################################
my $positiveDistance = 0;
my $negativeDistance = 0;
my $absoluteDistance = 0;
my $sumDistance      = 0;
my $posCounter       = 0;
my $negCounter       = 0;
my $absCounter       = 0;

for(my $t = 0; $t < scalar(@tracks); $t++)
{
      if($averageDistances[$t] >= 0)
      {
       $positiveDistance += $averageDistances[$t];
```

```
        $posCounter++;
      }
      else
      {
       $negativeDistance += $averageDistances[$t];
       $negCounter++;
      }

      $absoluteDistance += abs($averageDistances[$t]);
      $sumDistance      += $averageDistances[$t];
      $absCounter++;
}

if($posCounter > 0) { $positiveDistance /= $posCounter; }
if($negCounter > 0) { $negativeDistance /= $negCounter; }
if($absCounter > 0) { $absoluteDistance /= $absCounter; }
if($absCounter > 0) {      $sumDistance /= $absCounter; }
if($pos_move_counter > 0) { $Total_Y_Pos_Move /= $pos_move_counter;
$Total_Pos_Move /= $pos_move_counter;}
if($neg_move_counter > 0) { $Total_Y_Neg_Move /= $neg_move_counter;
$Total_Neg_Move /= $neg_move_counter;}
if($no_of_tracks     > 0)
{
      $Total_Move       /= $no_of_tracks;
      $Total_Abs_Move   /= $no_of_tracks;
      $Total_X_Move     /= $no_of_tracks;
      $Total_Y_Move     /= $no_of_tracks;
      $Total_Y_Abs_Move /= $no_of_tracks;
}

############################################################
# Write the data to the results file; data from            #
# different calls of this script can go to the same        #
# results file                                             #
############################################################

# We made sure that the file exists
open (RESULTS, ">>$ResultsFile") or die "Error: ", print "\nCannot
open file!\n$! \n";
print "Column width: ",  $column_width, "\n";

my $numOfVectors     = scalar(@full_vector_angles);
if($numOfVectors     == 0){ $numOfVectors     = 1;}
if($all_track_pieces == 0){ $all_track_pieces = 1;}

# Write the data to the results file.
print RESULTS       $infile,
# File Name
               "\t", scalar(@full_vector_angles),
# #Vectors
               "\t", $upward_vectors,
# #upward Vectors
               "\t", $downward_vectors,
# #downward Vectors
               "\t", $leftward_vectors,
# #leftward Vectors
               "\t", $rightward_vectors,
# #rightward Vectors
               "\t", $upward_vectors / $numOfVectors,
# %upward Vectors
```

```
                "\t", $downward_vectors / $numOfVectors,
# %downward Vectors
                "\t", $leftward_vectors / $numOfVectors,
# %leftward Vectors
                "\t", $rightward_vectors / $numOfVectors,
# %rightward Vectors
                "\t", $Total_Single_X_Move * $mm_per_pixel *
$frame_rate,                                    # #Average x
Displacement
                "\t", $Total_Single_Y_Move * $mm_per_pixel *
$frame_rate,                                    # #Average y
Displacement
                "\t", $Total_Single_X_Pos_Move * $mm_per_pixel *
$frame_rate,                                # #Average x positive
Displacement
                "\t", $Total_Single_Y_Pos_Move * $mm_per_pixel *
$frame_rate,                                # #Average y positive
Displacement
                "\t", $Total_Single_X_Neg_Move * $mm_per_pixel *
$frame_rate,                                # #Average x negative
Displacement
                "\t", $Total_Single_Y_Neg_Move * $mm_per_pixel *
$frame_rate,                                # #Average y negative
Displacement
                "\t", $Total_Single_X_Abs_Move * $mm_per_pixel *
$frame_rate,                                # #Average x absolute
Displacement
                "\t", $Total_Single_Y_Abs_Move * $mm_per_pixel *
$frame_rate,                                # #Average y absolute
Displacement
                "\t", $Total_X_Move * $mm_per_pixel * $frame_rate,
# #Average x Movement
                "\t", $Total_Y_Move * $mm_per_pixel * $frame_rate,
# #Average y Movement
                "\t", $Total_Y_Pos_Move * $mm_per_pixel * $frame_rate,
# #Average positive y Movement
                "\t", $Total_Y_Neg_Move * $mm_per_pixel * $frame_rate,
# #Average negative y Movement
                "\t", $Total_Y_Abs_Move * $mm_per_pixel * $frame_rate,
# #Average absolute y Movement
                "\t", $Total_Move * $mm_per_pixel * $frame_rate,
# #Average Movement
                "\t", $Total_Pos_Move * $mm_per_pixel * $frame_rate,
# #Average positive Movement
                "\t", $Total_Neg_Move * $mm_per_pixel * $frame_rate,
# #Average negative Movement
                "\t", $Total_Abs_Move * $mm_per_pixel * $frame_rate,
# #Average absolute Movement
                "\t", $number_of_larvae_in_column,
# #Larvae
                "\t", $larvae_in_upper_half,
# #Larvae Upper
                "\t", $larvae_in_lower_half,
# #Larvae Lower
                "\t", $larvae_in_upper_half_percent,
# #Larvae % Upper
                "\t", $larvae_in_lower_half_percent,
# #Larvae % Lower
                "\t", $positiveDistance * $mm_per_pixel * $frame_rate,
# #Upward Speed (mm per sec)
```

```
                "\t", $negativeDistance * $mm_per_pixel * $frame_rate,
# #Downward Speed (mm per sec)
                "\t", $absoluteDistance * $mm_per_pixel * $frame_rate,
# #Absolute Speed (mm per sec)
                "\t",      $sumDistance * $mm_per_pixel * $frame_rate,
# #Single Sum Speed (mm per sec)
                "\t", ($positiveDistance + $negativeDistance) *
$mm_per_pixel * $frame_rate,                # #Sum Speed (mm per sec)
                "\t", $Total_Single_Distance * $mm_per_pixel *
$frame_rate,                                 # #Speed (mm per sec)
                "\t", ($mean_depth_counter - $pixels_air) *
$mm_per_pixel,                                   # #Median depth (mm
from surface)
                "\t", ($mean_depth_counter - $pixels_air) /
($no_of_lines - $pixels_bottom - $pixels_air), # #Median depth (in %)
                "\t", $mean_left_right_counter - ($column_width/2) *
$mm_per_pixel,                          # #Median left/right (mm from
middle)
                "\t", ($mean_left_right_counter - ($column_width/2)) /
$column_width,                       # #Median left/right (in % from
middle)
                "\t", $Total_Single_X_Move / $Total_Single_Distance,
# #Simple X straightness
                "\t", $Total_Single_Y_Move / $Total_Single_Distance,
# #Simple Y straightness
                "\t", $Total_Single_X_Straightness,
# #Single X straightness
                "\t", $Total_Single_Y_Straightness,
# #Single Y straightness
                "\t", atan2($Total_Single_X_Move,
$Total_Single_Y_Move),                                  # #Average
Angel
                "\t", $top_vectors,
# #Top Vectors
                "\t", $bottom_vectors,
# #Bottom Vectors
                "\t", $left_vectors,
# #Left Vectors
                "\t", $right_vectors,
# #Right Vectors
                "\t", $top_vectors / $numOfVectors,
# %Top Vectors
                "\t", $bottom_vectors / $numOfVectors,
# %Bottom Vectors
                "\t", $left_vectors / $numOfVectors,
# %Left Vectors
                "\t", $right_vectors / $numOfVectors,
# %Right Vectors

                "\t", $all_track_pieces,
# #Track Pieces
                "\t", $upward_track_pieces,
# #Upward track pieces
                "\t", $downward_track_pieces,
# #Downward track pieces
                "\t", $rightward_track_pieces,
# #Leftward track pieces
                "\t", $leftward_track_pieces,
# #Rightward track pieces
                "\t", $top_track_pieces,
# #Top track pieces
```

```
                "\t", $bottom_track_pieces,
# #Bottom track pieces
                "\t", $left_track_pieces,
# #Left track pieces
                "\t", $right_track_pieces,
# #Right track pieces

                "\t", $upward_track_pieces / $all_track_pieces,
# %Upward track pieces
                "\t", $downward_track_pieces / $all_track_pieces,
# %Downward track pieces
                "\t", $rightward_track_pieces / $all_track_pieces,
# %Leftward track pieces
                "\t", $leftward_track_pieces / $all_track_pieces,
# %Rightward track pieces
                "\t", $top_track_pieces / $all_track_pieces,
# %Top track pieces
                "\t", $bottom_track_pieces / $all_track_pieces,
# %Bottom track pieces
                "\t", $left_track_pieces / $all_track_pieces,
# %Left track pieces
                "\t", $right_track_pieces / $all_track_pieces,
# %Right track pieces

                "\t", $median_X_Moves * $mm_per_pixel * $frame_rate,
# #Median X Movement
                "\t", $median_Y_Moves * $mm_per_pixel * $frame_rate,
# #Median Y Movement

                "\n";
close RESULTS;
```

Code 13: The Perl file MTrack.pm

```
############################################################
# MTrack.pm: Perl module file to parse the output of       #
# mTrack2 or mTrack3.                                      #
############################################################

package MTrack;

use strict;
use warnings;

use File::Basename;
use File::Spec;

############################################################
# ReadTracks reads tracks from an mTrack output file.      #
# infile is the file that it reads the tracks from.        #
# mTrackVersion gives the version of mTrack with that      #
# the output file was created with.                        #
# If mTrackVersion equals 3 then mTrack3 is assumed.       #
# Otherwise mTrack2 is assumed.                            #
# Returns an array of tracks. Each track is an array       #
# of positions. Each position contains x and y             #
# coordinates and an isValid flag indicating whether       #
# the position is a real position or just a                #
# placeholder.                                             #
############################################################
sub ReadTracks
```

```
{
      my $infile        = shift;
      my $mTrackVersion = shift;

      if($mTrackVersion == 3)
      {
       return MTrack::ReadTracks3($infile);
      }
      else
      {
       return MTrack::ReadTracks2($infile);
      }
}

############################################################
# ReadDistances reads the distances for each track         #
# from mTrack output.                                      #
# infile is the file that it reads the distances from.     #
# mTrackVersion gives the version of mTrack with that      #
# the output file was created with.                        #
# If mTrackVersion equals 3 then mTrack3 is assumed.       #
# Otherwise mTrack2 is assumed.                            #
############################################################
sub ReadDistances
{
      my $infile        = shift;
      my $mTrackVersion = shift;

      if($mTrackVersion == 3)
      {
       return MTrack::ReadDistances3($infile);
      }
      else
      {
       return MTrack::ReadDistances2($infile);
      }
}

############################################################
# ReadTracks2 reads tracks from an mTrack2 output          #
# file.                                                    #
# infile is the file that it reads the tracks from.        #
# Returns an array of tracks. Each track is an array       #
# of positions. Each position contains x and y             #
# coordinates and an isValid flag indicating whether       #
# the position is a real position or just a                #
# placeholder.                                             #
############################################################
sub ReadTracks2
{
      my $infile = shift;

      open (TRACKS_IN, "<$infile") or die "Error: ", print "\nCannot
open file!\n$! \n";

      my @tracks       = ();
      my $blockCounter = 0;

      # We fill the data into an array of arrays
      while (defined (my $line = <TRACKS_IN>))
      {
```

```
	# We exclude the first line that starts with "Frame" and the
lines with the track lengths
	unless ($line =~ m/^Track/ or $line =~ m/^Frame/ or $line =~
m/\:/)
	{
		my @tmp = split('\t', $line);

		for (my $i=1; $i < scalar(@tmp); $i+= 3)
		{
			my $valid = ($tmp[$i] ne " ") ? 1 : 0;
			my $position = ({x => $tmp[$i], y => $tmp[$i+1],
isValid => $valid});
			my $j = (($i - 1) / 3) + $blockCounter;

			$tracks[$j][$tmp[0]-1] = $position;
		}
	}

	# A new block starts
	if($line =~ m/^Track/)
	{
		$blockCounter = scalar(@tracks);
	}

	# Stop looking for data if we encounter the distance traveled
section:
	if($line =~ m/Nr of Frames$/)
	{
		last;
	}
	}

	close TRACK_IN;

	return @tracks;
}

############################################################
# ReadTracks2 reads tracks from an mTrack3 output	   #
# file.                                                    #
# infile is the file that it reads the tracks from.	   #
# Returns an array of tracks. Each track is an array	   #
# of positions. Each position contains x and y		   #
# coordinates and an isValid flag indicating whether	   #
# the position is a real position or just a		   #
# placeholder.                                             #
############################################################
sub ReadTracks3
{
	my $infile = shift;

	open (TRACKS_IN, "<$infile") or die die "Error: ", print
"\nCannot open file!\n$! \n";
	my @tracks       = ();
	my $blockCounter = 0;

	# We fill the data into an array of arrays
	while (defined (my $line = <TRACKS_IN>))
	{
	 # We exclude the first line that starts with "Frame" and the
lines with the track lengths
```

```
      unless ($line =~ m/^ \tTrack/ or $line =~ m/^Frame/ or $line =~
m/\:/)
      {
            my @tmp = split('\t', $line);

            my $position = ({x => $tmp[3], y => $tmp[5], isValid =>
1});
            $tracks[$tmp[1]-1][$tmp[2]-1] = $position;
      }
      }

      close TRACK_IN;

      return @tracks;
}

############################################################
# ReadDistances2 reads the distances for each track        #
# from mTrack output.                                      #
# infile is the file that it reads the distances from.     #
############################################################
sub ReadDistances2
{
      my $saveDistances = 0;
      my $infile = shift;

      open (TRACKS_IN, "<$infile") or die "Error: ", print "\nCannot
open file!\n$! \n";

      # For debugging the distances can be saved to a file
      if($saveDistances){ open (DISTANCE_PER_LENGTH,
">$infile"."_distance_per_length.txt") or die "Error: ", print
"\nCannot open file!\n$! \n";}

      my @distance_per_length = ();
      my $hasStarted          = 0;

      # We check the number of track blocks, the track lengths and
number of tracks by looking at the second number in the last line
"Tracks"
      while (defined (my $line = <TRACKS_IN>))
      {
       if
          (
              $line =~ /\:/ # These are the lines with the track
lengths, but only if you do not directly save the output to a file.
           || $hasStarted   # The above issue is fixed with that
          )
       {
            my @length              = split (/\t/, $line);
            my $length              = $length[1];
            my $distance_travelled  = $length[2];

            if($length != 0.0)
            {
                  my $distance_per_length =
$distance_travelled/$length; # Calculate the distance/length
parameter for each track
                  push (@distance_per_length, $distance_per_length);
                  if($saveDistances){ print DISTANCE_PER_LENGTH
$distance_per_length, "\n";}
```

```
            }
            else
            {
                  # For debugging the distances can be saved to a
file
                  if($distance_travelled >= 0)
                  {
                        if($saveDistances){ print DISTANCE_PER_LENGTH
Math::BigFloat->binf(), "\n";}
                  }
                  else
                  {
                        if($saveDistances){ print DISTANCE_PER_LENGTH
Math::BigFloat->binf('-'), "\n";}
                  }
            }
      }
      else
      {
            # Scan the file until we reach the distance traveled
section and then we parse the distances
            if($hasStarted == 0 && $line =~ m/Track/ && $line =~
m/Length/ && $line =~ m/Distance traveled/)
            {
                  $hasStarted = 1;
            }
      }
     }

     close TRACKS_IN;
     if($saveDistances){ close DISTANCE_PER_LENGTH;}

     return @distance_per_length;
}

#########################################################
# ReadDistances2 reads the distances for each track     #
# from mTrack output.                                   #
# infile is the file that it reads the distances from.  #
#########################################################
sub ReadDistances3
{
     my $saveDistances = 0;
     my $infile = shift;

     my($filename, $path, $suffix) = fileparse($infile,
qr/\.[^.]*/);

     my $prefixed_infile = File::Spec->catfile($path, "Summary_" .
$filename . $suffix);
     my $distance_file   = File::Spec->catfile($path, $filename .
"_distance_per_length.txt");


     open (TRACKS_IN, "<$prefixed_infile") or die "Error: ", print
"\nCannot open file!\n$! \n", $prefixed_infile, "\n";

     # For debugging the distances can be saved to a file
     if($saveDistances){ open (DISTANCE_PER_LENGTH,
">$distance_file") or die "Error: ", print "\nCannot open file!\n$!
\n", $distance_file, "\n";}
```

```
      my @distance_per_length = ();

      # We check the number of track blocks, the track lengths and
number of tracks by looking at the second number in the last line
"Tracks"
      while (defined (my $line = <TRACKS_IN>))
      {
       unless ($line =~ m/Track/)
       {
             my @length              = split (/\t/, $line);
             my $length              = $length[2];
             my $distance_travelled  = $length[3];

             if($length != 0.0)
             {
                   my $distance_per_length =
$distance_travelled/$length; # Calculate the distance/length
parameter for each track
                   push (@distance_per_length, $distance_per_length);
                   if($saveDistances){ print DISTANCE_PER_LENGTH
$distance_per_length, "\n";}
             }
             else
             {
                   # For debugging, the distances can be saved to a
file
                   if($distance_travelled >= 0)
                   {
                         if($saveDistances){ print DISTANCE_PER_LENGTH
Math::BigFloat->binf(), "\n";}
                   }
                   else
                   {
                         if($saveDistances){ print DISTANCE_PER_LENGTH
Math::BigFloat->binf('-'), "\n";}
                   }
             }
       }
      }

      close TRACKS_IN;
      if($saveDistances){ close DISTANCE_PER_LENGTH;}

      return @distance_per_length;
}

1; # Return that everything is fine
```

Code 14: The Perl file CSV_ColumnExtractor.pl

```
#!/usr/bin/perl -w

############################################################
# This script extracts data columns identified by      #
# their title from a tab delimetered list.             #
# It takes three arguments: The input file that        #
# contains the data to be extracted, the output file   #
# that receives the data, and a key to access the data #
# column to be extracted.                              #
```

```
############################################################

use strict;
use warnings;

use Text::CSV;
use Carp;
use File::Basename;
use File::Spec;

############################################################
# Min and Max functions                                    #
############################################################
sub max ($$) { $_[$_[0] < $_[1]] }
sub min ($$) { $_[$_[0] > $_[1]] }

#set command line arguments
my ($infi, $outfile, $idcol) = @ARGV;

# Get date, larval age, and genotype from input file
$infi =~ m/([0-9]{4}-[0-9]{2}-[0-9]{2}).*((\d+(\.{1}\d+){1})[A-Za-
z]+_[A-Za-z0-9\-_]+_[0-9]+)/;
my $title = "";

if(defined $1)
{
      $title = $1 . "_" . $2;
}
else
{
      # For some reason I cannot make it match a float and an integer
at the same time.
      $infi =~ m/([0-9]{4}-[0-9]{2}-[0-9]{2}).*([0-9]+[A-Za-z]+_[A-
Za-z0-9\-_]+_[0-9]+)/;
      $title = $1 . "_" . $2;
}

print "Add ", $idcol, " of ", $title, " to ", $outfile, "\n";

# Read in data from whatever seperated values format, use tab as
seperator
my $csv = Text::CSV->new({
  sep_char => "\t"
});

open(my $fh, "<:encoding(UTF-8)", $infi) or die "Can't open $infi:
$!";

# Get column names
$csv->column_names($csv->getline($fh));

my @column = ();

while(my $hr = $csv->getline_hr($fh))
{
      # Read data by certain column ID
      push(@column, $hr->{$idcol})
}

close $fh;
```

```
if(-e $outfile)
{
	my @outtable = ();

	open($fh, "<:encoding(UTF-8)", $outfile) or die "Can't open
$outfile: $!";
	while (<$fh>)
	{
	 $_ =~ s/[\r\n]+//g; # Clean line endings
	 push(@outtable, $_);
	}
	close $fh;

	open($fh, ">:encoding(UTF-8)", $outfile) or die "Can't open
$outfile: $!";
	print $fh $outtable[0], "\t", $title, "\n";
	for(my $i = 0; $i < max(scalar(@column), scalar(@outtable)-1);
$i++)
	{
	 if(defined($column[$i]))
	 {
		if(defined($outtable[$i+1]))
		{
			print $fh $outtable[$i+1], "\t", $column[$i], "\n";
		}
		else
		{
			my $countTabs = ($outtable[0] =~ tr/\t//);
			print $fh "\t" x $countTabs, "\t", $column[$i],
"\n";
		}
	 }
	 elsif(defined($outtable[$i+1]))
	 {
		print $fh $outtable[$i+1], "\t\n";
	 }
	}
	close $fh;
}
else
{
	open($fh, ">:encoding(UTF-8)", $outfile) or die "Can't open
$outfile: $!";
	print $fh $title, "\n";
	for(my $i = 0; $i < scalar(@column); $i++)
	{
	 if(defined($column[$i]))
	 {
		print $fh $column[$i], "\n";
	 }
	 else
	 {
		print $fh "\n";
	 }
	}
	close $fh;
}
```

11.3 Controlling the monochromator via the serial port

Code 15: Modified Transmitter.java of BlackBox

```
/*
 * @(#)Transmitter.java 1.12 98/06/25 SMI
 *
 * Author: Tom Corson
 *
 * Copyright (c) 1998 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Sun grants you ("Licensee") a non-exclusive, royalty free, license
 * to use, modify and redistribute this software in source and binary
 * code form, provided that i) this copyright notice and license
appear
 * on all copies of the software; and ii) Licensee does not utilize
the
 * software in a manner which is disparaging to Sun.
 *
 * This software is provided "AS IS," without a warranty of any kind.
 * ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES,
 * INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN
AND
 * ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY
 * LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE
 * SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS
 * BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT,
 * INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES,
 * HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING
 * OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS
BEEN
 * ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
 *
 * This software is not designed or intended for use in on-line
control
 * of aircraft, air traffic, aircraft navigation or aircraft
 * communications; or in the design, construction, operation or
 * maintenance of any nuclear facility. Licensee represents and
 * warrants that it will not use or redistribute the Software for
such
 * purposes.
 */

import java.lang.Thread;

import java.io.IOException;

import java.awt.Panel;
import java.awt.Label;
import java.awt.TextArea;
import java.awt.Checkbox;
import java.awt.Color;
import java.awt.BorderLayout;
import java.awt.FlowLayout;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.TextListener;
import java.awt.event.TextEvent;
import java.awt.event.ItemListener;
```

```
import java.awt.event.ItemEvent;
import java.nio.file.Files;
import java.io.BufferedReader;
import java.awt.Button;

import javax.swing.JFileChooser;
import javax.swing.Timer;
import javax.comm.SerialPort;

public class Transmitter extends Panel implements TextListener,
ItemListener, ActionListener, Runnable
{
      private Panel             p;
      private Panel             p1;
      private Panel             p2;
      private Panel             useProtocolPanel;
      private Panel             timerPanel;
      private Label             timerLabel;
      private TextArea          text;
      private Checkbox          auto;
      private Checkbox          sendBreak;
      private Checkbox          useProtocolBox;
      private Button            runProtocolButton;
      private ByteStatistics    counter;
      private SerialPortDisplay owner;
      private      Thread            thr;
      private Color             onColor;
      private Color             offColor;
      private boolean           first;
      private boolean           modemMode;
      private Timer             timer;
      private int               counterValue;
      static  String            testString =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ\nabcdefghijklmnopqrstuvwxyz\n1234567890\n
";

      public Transmitter(SerialPortDisplay owner,
                int                 rows,
                int                 cols)
      {
       super(new BorderLayout());

       this.first = true;
       this.modemMode = false;

       this.owner = owner;

       p  = new Panel(new FlowLayout());

       p1 = new Panel(new BorderLayout());
       p1.add("West", new Label("Auto Transmit"));
       auto = new Checkbox();
       auto.addItemListener(this);
       p1.add("East", auto);
       p.add(p1);

       p2 = new Panel(new BorderLayout());
       p2.add("West", new Label("Send Break"));
       sendBreak = new Checkbox();
       sendBreak.addItemListener(this);
       p2.add("East", sendBreak);
```

```
        p.add(p2);

        useProtocolPanel = new Panel(new BorderLayout());
        useProtocolPanel.add("West", new Label("Use Protocol"));
        useProtocolBox = new Checkbox();
        useProtocolBox.addItemListener(this);
        useProtocolPanel.add("East", useProtocolBox);
        p.add(useProtocolPanel);

        timerPanel = new Panel(new BorderLayout());
        timerLabel = new Label("Next: 0");
        timerPanel.add("West", timerLabel);
        p.add(timerPanel);

        runProtocolButton = new Button();
        runProtocolButton.setLabel("Run Protocol");
        runProtocolButton.addActionListener(this);
        p.add(runProtocolButton);

        this.add("North", p);

        this.text = new TextArea(rows, cols);
        this.text.append("Type here");
        this.text.addTextListener(this);
        this.add("Center", text);

        this.counter = new ByteStatistics("Bytes Sent", 10,
                                  owner.port, false);
        this.add("South", this.counter);

        this.thr = null;

        this.onColor = Color.green;
        this.offColor = Color.black;

        this.timer = new Timer(1000, this);
      }

      public Transmitter(SerialPortDisplay owner,
                int                   rows,
                int                   cols,
                boolean         modemMode)
      {
        this(owner, rows, cols);

        this.modemMode = modemMode;
      }

      public void setPort(SerialPort port)
      {
        this.counter.setPort(port);
      }

      public void showValues()
      {
        this.counter.showValues();
      }

      public void clearValues()
      {
        this.counter.clearValues();
```

```
    }

    public void setBitsPerCharacter(int  val)
    {
     this.counter.setBitsPerCharacter(val);
    }

    /*
     *     Handler for transmit text area events
     */

    public void textValueChanged(TextEvent     ev)
    {
     if(this.useProtocolBox.getState())
     {
           return;
     }

     if (first && (this.text.getCaretPosition() > 0))
     {
           first = false;

           this.text.replaceRange("",
                                  0,
                                  this.text.getCaretPosition()
                                  - 1);
     }

     if (!first && this.text.getText().endsWith("\n"))
     {
           this.sendData();
     }

    }

    public void run()
    {
     this.sendData();
    }

    public void sendString(String str)
    {
     int   count;

     count = str.length();

     if (count > 0)
     {
           try
           {
                 owner.out.write(str.getBytes());

                 counter.incrementValue((long) count);

                 owner.ctlSigs.BE = false;

                 owner.ctlSigs.showErrorValues();
           }

           catch (IOException ex)
           {
```

```
                    if (owner.open)
                    {
                          System.out.println(owner.port.getName()
                                      + ": Cannot write to output
stream");

                          this.auto.setState(false);
                    }
              }
       }
      }

      private void sendData()
      {
       if (this.owner.open && this.auto.getState())
       {
              while (this.owner.open && this.auto.getState())
              {
                    sendString(testString);
              }
       }

       else
       {
              String str = this.text.getText();

              sendString(str);

              this.text.setText("");
       }
      }

      /*
       *     Handler for checkbox events
       */

      public void itemStateChanged(ItemEvent     ev)
      {
       if (this.auto.getState() && (thr == null) && this.owner.open)
       {
              startTransmit();
       }
       else
       {
              stopTransmit();
       }

       if (this.sendBreak.getState())
       {
              if (this.owner.open)
              {
                    this.sendBreak.setForeground(this.onColor);

                    /*
                     *  Send a 1000 millisecond break.
                     */

                    owner.port.sendBreak(1000);
              }

              this.sendBreak.setState(false);
```

```
            this.sendBreak.setForeground(this.offColor);
      }

      if(this.useProtocolBox.getState())
      {
            //Create a file chooser
            final JFileChooser fc = new JFileChooser();

            //In response to a button click:
            int returnVal = fc.showOpenDialog(this);

            if(returnVal == JFileChooser.APPROVE_OPTION)
            {
                  this.text.setEnabled(false);

                  try (BufferedReader reader =
Files.newBufferedReader(fc.getSelectedFile().toPath()))
                  {
                        String line = null;
                        StringBuilder builder = new StringBuilder();
                        while ((line = reader.readLine()) != null)
                        {
                              builder.append(line);
                              builder.append('\n');
                        }
                        first = false;
                        this.text.setText(builder.toString());
                  }
                  catch (IOException x)
                  {
                        System.err.format("IOException: %s%n", x);
                  }
            }
            else
            {
                  this.useProtocolBox.setState(false);
            }
      }
      else
      {
            this.text.setEnabled(true);
      }
     }

     private void startTransmit()
     {
      if (thr == null);
      {
            this.auto.setForeground(this.onColor);
            counter.resetRate();

            thr = new Thread(this, "Xmt " + owner.port.getName());

            thr.start();
      }
     }

     public void stopTransmit()
     {
      thr = null;
```

```
        this.auto.setState(false);
        this.auto.setForeground(this.offColor);
      }

      public void actionPerformed(ActionEvent e)
      {
        if(this.useProtocolBox.getState())
        {
              if(e.getActionCommand() == runProtocolButton.getLabel())
              {
                    if(runProtocolButton.getLabel() == "Run Protocol")
                    {
                          this.timer.start();
                          this.runProtocolButton.setLabel("Stop
Protocol");

                          this.processNextProtocolItem();
                    }
                    else
                    {
                          this.stopTimer();
                    }
              }
              else
              {
                    counterValue--;
                    timerLabel.setText("Next: " +
String.valueOf(this.counterValue));

                    if(counterValue < 1)
                    {
                          if(this.text.getText().trim().isEmpty())
                          {
                                this.stopTimer();
                          }
                          else
                          {
                                this.processNextProtocolItem();
                          }
                    }
              }
        }
      }

      private void processNextProtocolItem()
      {
        while(!this.text.getText().isEmpty())
        {
              String[] lines = this.text.getText().split("\n", 2);
              String[] instructions = lines[0].split(" Wait ");
              this.text.setText(lines[1]);

              this.sendString(instructions[0] + "\n");

              if(instructions.length > 1)
              {
                    this.counterValue =
Integer.parseInt(instructions[1]);
                    this.timerLabel.setText("Next: " +
String.valueOf(this.counterValue));
```

```
                    break;
              }
              else
              {
                    this.counterValue = 1;
                    this.timerLabel.setText("Next: " +
String.valueOf(this.counterValue));
              }
       }
      }
      private void stopTimer()
      {
       this.timer.stop();

       this.counterValue = 0;
       this.timerLabel.setText("Next: " +
String.valueOf(this.counterValue));
       this.text.setText("");
       this.runProtocolButton.setLabel("Run Protocol");
       this.useProtocolBox.setState(false);
       this.text.setEnabled(true);
      }
}
```

11.4 Opsins translated with annotated introns

```
>Pdu-Go-opsin1 translated with introns annotated
0 MEFNHTTEDSYNSTFDFITYGTHVEIYKRPDIQPRVYMVIGVYLTIA 1
2
GIISTVGNSVVIGVVVKNKELRKQGHNILLLNLAICDLGFTFVGYPLTASSAFAQRW
LFGHLGCVIYGFCCTVLALTDINILMALSIYRYIVICKPHI 1
2
RHILHRRTVAAAMVTSCWVYSLLWGVAALVGWNRYTNEAFGTSCSIDWTARGASDLS
YTILMIFFCYISHIIVMTFCYYK 0
0 IKQRSSLMLSRLRNHHKFSAEDAVLINNIRNEKRLTV 0
0
MTMVMVGGFILVWSPYAWVAVWKIVVPDGVPDWLTTFPTMFAKATPMLNPLIYVSTN
RKFRREARGMLRRWCCCFSAKVDDIVASAIRNRQA 2
1 SPKEKRVYFVNMTKEGIYTGKRRVSLPTIETIF 0
```

```
>Pdu-Go-opsin2 translated with introns annotated:
0 MTTESHGPPDVVALGRTGYIVAGTYLCVI 1
2
ATVATLGNSLVVVTFVRNSSTRKKCHNILLLNLAIADLGISFFGYPLVTVSTLSGRW
MSR 0
0 DYGCKIYAFCTFFFSLVSLNTLVFLSIYRYVIVCRPSY 1
2
KHHLNKRVTTWSIISSWIYGFFWAILPFFGWSHYTYEKFGTSCTIDWVDQSLSAITY
DVTVIVTCFLIHVAIMIFCYQK 0
0 IIKRARNLIFDHGISVAEEQKNGGGFSEGFNVKYMRKQSRISF 0
0 MCCIMVFSFIICWTPYTVMSCVTIFTQVPSTLSTIPTFFAK 0
0 AAPMSNSIIYFFMNKKFREAFFRTFCCCQQNIILPRNTGAAHLAK 2
1
YYPNWNLWSLFHARPLCNKEGVYVGAEHQDDTGHSRGQLEYEMAANQMAPQVEGDVD
AKNQDEDISPKASTSSTSFRMERLTNFKAEQRNKSKNELPLLNLIKQESFVKSQRTS
RVVTSNVDSDTPTPNETEDQRRSSNTDALPSTSDSLKYRQTHFGGRDSNHAKNYGHK
VMLHTVHPEKPGKSPRRKTSTSKMSNKSATSKRRNSQSSDQSQKLSPATKTPPSRSC
SSSQQSNSNNDSGIEVLPDSGLLDPSDKNPAQSSVAKYSRHMAQDNLDEMFSWL 0
```

```
>Pdu-r-opsin2 translated with introns annotated
0 M 0
0 MFFQSE
MAQDDSESFTAYPEEGDTNNITLGDLDLDSTLTVPYLENGLFFHPHWRKYRQMLENV
PDSVHYILGIYITFVGFAGVIGNAIVIFVFTA
TKSLRTPSNMFIVNLAMSDLGFSLVNGFPLM 2
1 SVSSFMRKWYFGRV 1
2 GCILYGTLSGVFGLTSINTLALIAFDRFYVIQFPLRAIRTVTRTR 2
1
SFVQICLVWIWATFWSCPPLFGWGRYIPEGLQTSCGFDFLSQDPLNRAFNYCIFSCG
FVLPVTFAICSYCGILATVSMQAKHMEKIKQTKGGQLNDDKEKEKKQQIRLAKIAAG
TISLFIISWMPYALLVILSTSGYRHIMTPYVCQIPSVFAKASAIWNPFVYSISHPKY
RQALQERFPWLLCNKKDTDDVIELGDKKTRKQSLDSENLSDSTISESSKDSPKPRKA
NVTPTPASKKVVSQAAFGTSNKSQVTSNTL 0
```

```
>r-opsin5 translated with introns annotated
0
MAGPCESGCPALDYRSDNITNVANVTNVVQVLLHNNSETTSATTDVLAMHSGIHHHY
WTKFSPPPHEIHITIGFAMATIGVLAVAGNTFVIFVFLR 2
1 FRSLRTPGNLLMINLAVSDLLMAVTGFPLYSISSFYGRWVLPDA 1
2
VCLFYGACGATFGLLSINSLAAIAVDRYLVIAHSYAVTKRTNRRQAIVMIVLSWINS
LCWAIPPLLGWNRYLLEGFGTTCTFDYLSRTKSDRLFVMLMFCCGFCLPLLLIIGSY
AYIYSVVHRHERMFRNMSQNLNARIMHGGKEATQRTEMKTARTVILAVLFYCISWVP
YATIALIGIYGNYQLLTPLVTAVPGILAKMSTIYNPLLYTFSHPRFHKKVMLLLFKR
SMVLDKNTSNMDHMGGGKSQCHTQCLPKVNGSEPQAVSTLSSTSSWSGDTLPGDQRN
AFHLRDASDHTDTISLASTARLSNQASFSSKSRQPRFMQKGKKDQNRSGRRGKDSAN
NSIEERHTFLKQKEGKESKPKKFLKDLPVKPPVETVV 0
```

11.5 Test statistics details for Figure 13A

Within each row, compare columns (simple effects within rows)		
Number of families	14	
Number of comparisons per family	6	
Alpha	0.05	
Tukey's multiple comparisons test	Significance	Adjusted P Value
dark		
41 hpf vs. 53 hpf	ns	0.9976
41 hpf vs. 3 dpf	ns	0.8985
41 hpf vs. 4.5 dpf	ns	0.6323
53 hpf vs. 3 dpf	ns	0.9909
53 hpf vs. 4.5 dpf	ns	0.7381
3 dpf vs. 4.5 dpf	ns	0.2122
380		
41 hpf vs. 53 hpf	ns	0.6799
41 hpf vs. 3 dpf	ns	0.9984
41 hpf vs. 4.5 dpf	ns	0.2376
53 hpf vs. 3 dpf	ns	0.7434
53 hpf vs. 4.5 dpf	ns	0.0707
3 dpf vs. 4.5 dpf	ns	0.1566
520		
41 hpf vs. 53 hpf	ns	0.4279
41 hpf vs. 3 dpf	ns	0.9733
41 hpf vs. 4.5 dpf	ns	0.0931
53 hpf vs. 3 dpf	ns	0.6075

53 hpf vs. 4.5 dpf	**	0.009
3 dpf vs. 4.5 dpf	*	0.0253
380		
41 hpf vs. 53 hpf	ns	0.3059
41 hpf vs. 3 dpf	ns	0.9677
41 hpf vs. 4.5 dpf	ns	0.2013
53 hpf vs. 3 dpf	ns	0.4803
53 hpf vs. 4.5 dpf	**	0.0098
3 dpf vs. 4.5 dpf	ns	0.0629
520		
41 hpf vs. 53 hpf	ns	0.645
41 hpf vs. 3 dpf	ns	0.9989
41 hpf vs. 4.5 dpf	*	0.013
53 hpf vs. 3 dpf	ns	0.7007
53 hpf vs. 4.5 dpf	**	0.0055
3 dpf vs. 4.5 dpf	**	0.0065
380		
41 hpf vs. 53 hpf	ns	0.1789
41 hpf vs. 3 dpf	ns	0.9248
41 hpf vs. 4.5 dpf	*	0.0412
53 hpf vs. 3 dpf	ns	0.3727
53 hpf vs. 4.5 dpf	***	0.0006
3 dpf vs. 4.5 dpf	**	0.0047
520		
41 hpf vs. 53 hpf	ns	0.6192
41 hpf vs. 3 dpf	ns	0.9966
41 hpf vs. 4.5 dpf	*	0.0359
53 hpf vs. 3 dpf	ns	0.7047
53 hpf vs. 4.5 dpf	*	0.0106
3 dpf vs. 4.5 dpf	*	0.016
380		
41 hpf vs. 53 hpf	ns	0.2685
41 hpf vs. 3 dpf	ns	0.9958
41 hpf vs. 4.5 dpf	ns	0.1838
53 hpf vs. 3 dpf	ns	0.3403
53 hpf vs. 4.5 dpf	**	0.0068
3 dpf vs. 4.5 dpf	ns	0.1006

520		
41 hpf vs. 53 hpf	ns	0.7052
41 hpf vs. 3 dpf	ns	> 0.9999
41 hpf vs. 4.5 dpf	*	0.0153
53 hpf vs. 3 dpf	ns	0.7048
53 hpf vs. 4.5 dpf	**	0.0087
3 dpf vs. 4.5 dpf	*	0.012
380		
41 hpf vs. 53 hpf	ns	0.903
41 hpf vs. 3 dpf	ns	0.9347
41 hpf vs. 4.5 dpf	*	0.0183
53 hpf vs. 3 dpf	ns	0.6865
53 hpf vs. 4.5 dpf	*	0.0328
3 dpf vs. 4.5 dpf	ns	0.0819
520		
41 hpf vs. 53 hpf	ns	0.7078
41 hpf vs. 3 dpf	ns	0.9975
41 hpf vs. 4.5 dpf	*	0.0215
53 hpf vs. 3 dpf	ns	0.7797
53 hpf vs. 4.5 dpf	*	0.0113
3 dpf vs. 4.5 dpf	**	0.0096
380		
41 hpf vs. 53 hpf	ns	0.803
41 hpf vs. 3 dpf	ns	0.6518
41 hpf vs. 4.5 dpf	*	0.0398
53 hpf vs. 3 dpf	ns	0.3116
53 hpf vs. 4.5 dpf	*	0.0297
3 dpf vs. 4.5 dpf	ns	0.4307
520		
41 hpf vs. 53 hpf	ns	0.7578
41 hpf vs. 3 dpf	ns	> 0.9999
41 hpf vs. 4.5 dpf	*	0.021
53 hpf vs. 3 dpf	ns	0.7519
53 hpf vs. 4.5 dpf	*	0.0147
3 dpf vs. 4.5 dpf	*	0.0176
dark		

41 hpf vs. 53 hpf	ns	0.9367
41 hpf vs. 3 dpf	*	0.0215
41 hpf vs. 4.5 dpf	ns	0.0898
53 hpf vs. 3 dpf	ns	0.484
53 hpf vs. 4.5 dpf	ns	0.7318
3 dpf vs. 4.5 dpf	ns	0.9322

11.6 Test statistics details for Figure 13B

Within each row, compare columns (simple effects within rows)		
Number of families	20	
Number of comparisons per family	6	
Alpha	0.05	
Tukey's multiple comparisons test	Significance	Adjusted P Value
dark		
41 hpf vs. 53 hpf	ns	0.9795
41 hpf vs. 3 dpf	ns	0.9739
41 hpf vs. 4.5 dpf	**	0.006
53 hpf vs. 3 dpf	ns	> 0.9999
53 hpf vs. 4.5 dpf	ns	0.0993
3 dpf vs. 4.5 dpf	ns	0.0559
340		
41 hpf vs. 53 hpf	ns	0.8982
41 hpf vs. 3 dpf	ns	0.0538
41 hpf vs. 4.5 dpf	****	< 0.0001
53 hpf vs. 3 dpf	*	0.0235
53 hpf vs. 4.5 dpf	****	< 0.0001
3 dpf vs. 4.5 dpf	***	0.0007
360		
41 hpf vs. 53 hpf	ns	0.087
41 hpf vs. 3 dpf	****	< 0.0001
41 hpf vs. 4.5 dpf	****	< 0.0001
53 hpf vs. 3 dpf	****	< 0.0001
53 hpf vs. 4.5 dpf	****	< 0.0001
3 dpf vs. 4.5 dpf	ns	> 0.9999
380		
41 hpf vs. 53 hpf	ns	0.1083

41 hpf vs. 3 dpf	ns	0.0556
41 hpf vs. 4.5 dpf	**	0.0048
53 hpf vs. 3 dpf	***	0.0001
53 hpf vs. 4.5 dpf	****	< 0.0001
3 dpf vs. 4.5 dpf	ns	0.9812
400		
41 hpf vs. 53 hpf	ns	0.1285
41 hpf vs. 3 dpf	ns	0.2715
41 hpf vs. 4.5 dpf	ns	0.5373
53 hpf vs. 3 dpf	**	0.0019
53 hpf vs. 4.5 dpf	**	0.0043
3 dpf vs. 4.5 dpf	ns	0.9068
420		
41 hpf vs. 53 hpf	ns	0.1775
41 hpf vs. 3 dpf	ns	0.079
41 hpf vs. 4.5 dpf	ns	0.5727
53 hpf vs. 3 dpf	ns	0.9988
53 hpf vs. 4.5 dpf	ns	0.7191
3 dpf vs. 4.5 dpf	ns	0.5364
440		
41 hpf vs. 53 hpf	ns	0.9913
41 hpf vs. 3 dpf	ns	0.7378
41 hpf vs. 4.5 dpf	ns	0.1938
53 hpf vs. 3 dpf	ns	0.6569
53 hpf vs. 4.5 dpf	ns	0.2101
3 dpf vs. 4.5 dpf	ns	0.882
460		
41 hpf vs. 53 hpf	ns	> 0.9999
41 hpf vs. 3 dpf	ns	0.3987
41 hpf vs. 4.5 dpf	***	0.0005
53 hpf vs. 3 dpf	ns	0.5617
53 hpf vs. 4.5 dpf	**	0.0072
3 dpf vs. 4.5 dpf	ns	0.2028
480		
41 hpf vs. 53 hpf	ns	0.9687
41 hpf vs. 3 dpf	ns	0.5958
41 hpf vs. 4.5 dpf	***	0.0004

53 hpf vs. 3 dpf	ns	0.4369
53 hpf vs. 4.5 dpf	***	0.0009
3 dpf vs. 4.5 dpf	ns	0.0888
500		
41 hpf vs. 53 hpf	ns	0.9776
41 hpf vs. 3 dpf	ns	0.5682
41 hpf vs. 4.5 dpf	***	0.0002
53 hpf vs. 3 dpf	ns	0.4428
53 hpf vs. 4.5 dpf	***	0.0006
3 dpf vs. 4.5 dpf	ns	0.0611
520		
41 hpf vs. 53 hpf	ns	0.9988
41 hpf vs. 3 dpf	ns	0.1157
41 hpf vs. 4.5 dpf	****	< 0.0001
53 hpf vs. 3 dpf	ns	0.2846
53 hpf vs. 4.5 dpf	**	0.0011
3 dpf vs. 4.5 dpf	ns	0.2031
540		
41 hpf vs. 53 hpf	ns	0.8626
41 hpf vs. 3 dpf	**	0.0092
41 hpf vs. 4.5 dpf	****	< 0.0001
53 hpf vs. 3 dpf	ns	0.1983
53 hpf vs. 4.5 dpf	**	0.0064
3 dpf vs. 4.5 dpf	ns	0.6383
560		
41 hpf vs. 53 hpf	ns	0.999
41 hpf vs. 3 dpf	*	0.0369
41 hpf vs. 4.5 dpf	***	0.0002
53 hpf vs. 3 dpf	ns	0.1303
53 hpf vs. 4.5 dpf	**	0.0052
3 dpf vs. 4.5 dpf	ns	0.7492
580		
41 hpf vs. 53 hpf	ns	0.9858
41 hpf vs. 3 dpf	*	0.0112
41 hpf vs. 4.5 dpf	***	0.0004
53 hpf vs. 3 dpf	ns	0.0919
53 hpf vs. 4.5 dpf	*	0.0186

3 dpf vs. 4.5 dpf	ns	0.9814
600		
41 hpf vs. 53 hpf	ns	0.9754
41 hpf vs. 3 dpf	ns	0.1721
41 hpf vs. 4.5 dpf	*	0.0351
53 hpf vs. 3 dpf	ns	0.5201
53 hpf vs. 4.5 dpf	ns	0.2745
3 dpf vs. 4.5 dpf	ns	0.9892
620		
41 hpf vs. 53 hpf	ns	0.9903
41 hpf vs. 3 dpf	ns	0.6687
41 hpf vs. 4.5 dpf	ns	0.6331
53 hpf vs. 3 dpf	ns	0.9006
53 hpf vs. 4.5 dpf	ns	0.9092
3 dpf vs. 4.5 dpf	ns	0.9996
640		
41 hpf vs. 53 hpf	ns	0.8635
41 hpf vs. 3 dpf	ns	0.1966
41 hpf vs. 4.5 dpf	ns	0.0923
53 hpf vs. 3 dpf	ns	0.7664
53 hpf vs. 4.5 dpf	ns	0.682
3 dpf vs. 4.5 dpf	ns	> 0.9999
660		
41 hpf vs. 53 hpf	ns	0.9146
41 hpf vs. 3 dpf	ns	0.1537
41 hpf vs. 4.5 dpf	ns	0.0724
53 hpf vs. 3 dpf	ns	0.6306
53 hpf vs. 4.5 dpf	ns	0.5455
3 dpf vs. 4.5 dpf	ns	> 0.9999
680		
41 hpf vs. 53 hpf	ns	0.8911
41 hpf vs. 3 dpf	ns	0.1609
41 hpf vs. 4.5 dpf	ns	0.0547
53 hpf vs. 3 dpf	ns	0.6792
53 hpf vs. 4.5 dpf	ns	0.5288
3 dpf vs. 4.5 dpf	ns	0.9992

dark		
41 hpf vs. 53 hpf	ns	0.972
41 hpf vs. 3 dpf	ns	0.9793
41 hpf vs. 4.5 dpf	ns	0.9997
53 hpf vs. 3 dpf	ns	0.9999
53 hpf vs. 4.5 dpf	ns	0.9518
3 dpf vs. 4.5 dpf	ns	0.9601