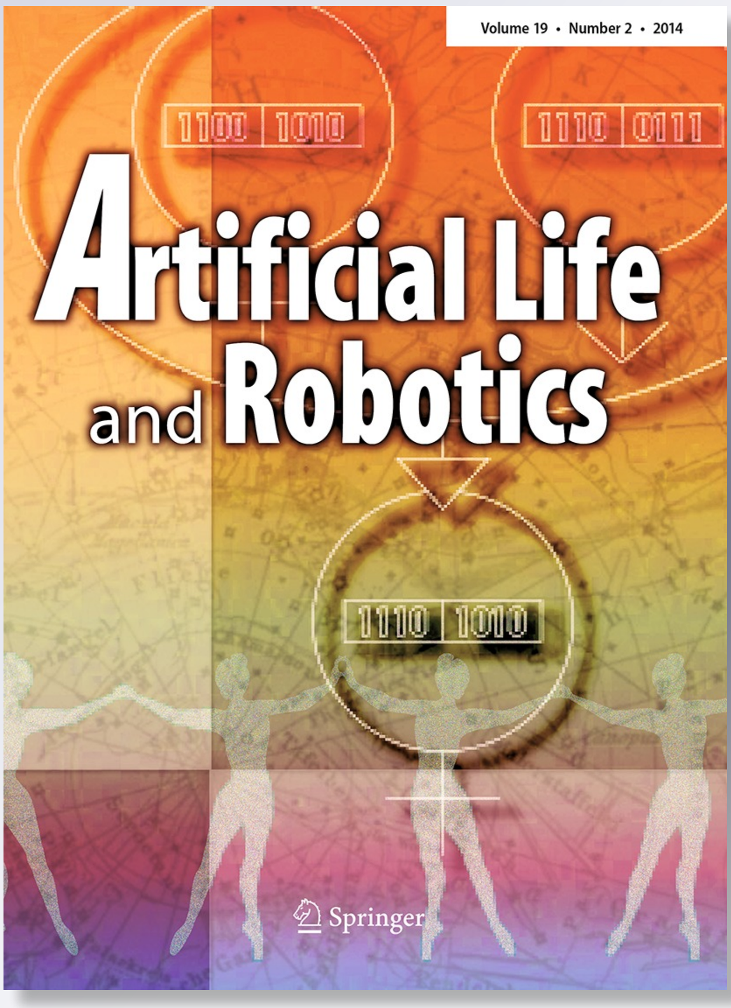
Learning to navigate in a virtual world using optic flow and stereo disparity signals

Florian Raudies, Schuyler Eldridge, Ajay Joshi & Massimiliano Versace

Artificial Life and Robotics

ISSN 1433-5298
Volume 19
Number 2

Artif Life Robotics (2014) 19:157-169
DOI 10.1007/s10015-014-0153-1



Artif Life Robotics (2014) 19:157–169
DOI 10.1007/s10015-014-0153-1

ORIGINAL ARTICLE

Learning to navigate in a virtual world using optic flow and stereo disparity signals

Florian Raudies · Schuyler Eldridge · Ajay Joshi · Massimiliano Versace

Received: 23 December 2013 / Accepted: 29 March 2014 / Published online: 20 August 2014
© ISAROB 2014

Abstract Navigating in a complex world is challenging in that the rich, real environment provides a very large number of sensory states that can immediately precede a collision. Biological organisms such as rodents are able to solve this problem, effortlessly navigating in closed spaces by encoding in neural representations distance toward walls or obstacles for a given direction. This paper presents a method that can be used by virtual (simulated) or robotic agents, which uses states similar to neural representations to learn collision avoidance. Unlike other approaches, our reinforcement learning approach uses a small number of states defined by discretized distances along three constant directions. These distances are estimated either from optic flow or binocular stereo information. Parameterized templates for optic flow or disparity information are compared against the input flow or input disparity to estimate these distances. Simulations in a virtual environment show learning of collision avoidance. Our results show that learning with only stereo information is superior to learning with only optic flow information. Our work motivates the usage of abstract state descriptions for the learning of visual navigation. Future work will focus on the fusion of optic flow and stereo information, and transferring these models to robotic platforms.

F. Raudies (✉) · M. Versace
Center for Computational Neuroscience and Neural Technology (CompNet) at Boston University, 677 Beacon Street, Boston, MA 02215, USA
e-mail: fraudies@bu.edu

S. Eldridge · A. Joshi
Department of Electrical and Computer Engineering at Boston University, 8 St. Mary Street, Boston, MA 02215, USA

Keywords Learning of navigation · Optic flow · Stereo disparity · Virtual world

1 Introduction

Estimating the distance from perceived objects in the environment, either target or obstacles, and the ability to learn their position and strategies to approach or avoid them, are crucial skills for humans, animals, and robots alike. Recently, memory structures that encode geometrical constraints of the environment have been discovered in rats [17, 29]. Cells encode the distance of walls for allocentric direction in their patterns of spatial firing, fostering the idea of using a distances as states of a reinforcement learner.

In general, various cues can be used to infer distance information, to detect the ground, or to determine traversiblity [14]. Relative depth cues learned from monocular images have been used to learn obstacle avoidance [19]. Other work focused on the unique encoding and fast recall of views from omnidirectional cameras for visual navigation and self-localization [2, 11]. Another approach is the extraction of image primitives, e.g. edges or Gabor filter responses, to encode views [15, 22, 30]. Yue et al. [35] directly take the video input to simulate the lobula giant movement detector neuron of locusts trained to signal collisions. Martinez-Marin and Duckett [18] use a color-based segmentation to learn a docking-task based on the orientation of the robot, the orientation of the table, and the distance of the object, which is placed at the edge of the table. Gaskett et al. [12] infer drivable space by a cross-correlation with a pre-loaded picture of the carpet texture, which is linked to actions of the robot by reinforcement learning.

In contrast to these methods, we use optic flow or stereo information to estimate distances along three constant directions instead of using a view-based approach. Our approach is invariant to the structure in the scene and works as long as the scene provides image features for a reliable detection of stereo or flow information. The invariance with respect to a view is provided by our template-matching approach that uses geometric constraints to infer distances regardless of the visual appearance of objects such as their texture.

Reinforcement learning provides a tool to map a potentially large state space capturing sensory data immediately preceding a collision with appropriate avoidance actions to be taken by a virtual or robotic agent. In addition, this approach avoids using neural networks to extract or interpolate between states from a large space that is defined, e.g. by low-level visual features, which encode views.

To test our flow-driven learning of visual navigation, we developed a simulation environment that provides analytical disparity and flow information. This information is passed to biologically inspired template models that estimate the distance of walls and, thus, derive the state used by the learner.

One key to the success of learning is the robustness and accuracy of the proposed template models in estimating distances. Learning appears when distance estimates are accurate enough to describe a given state within the chosen discretization correctly. An accurate state description depends largely on the sampling for this discretization. A coarse sampling promotes correct estimates, but leads to insufficient information for the learner. A dense sampling puts higher demands on the template models and slows down learning in terms of computation time and "convergence". This leads to a tradeoff between dense and coarse sampling.

Our contributions for learning of visual navigation using either optic flow or stereo information are:

1. We suggest biologically inspired algorithms that
   (a) estimate self-motion,
   (b) segment the scene into ground and walls,
   (c) estimate distances of walls from optic flow knowing the height of the camera above ground,
   (d) estimate distances from stereo disparity information, and
   (e) learn to avoid obstacles based on these sensory cues.
2. We quantify learning by evaluating the path length for one epoch, which increases from a few centimeters to several meters.

We continue in Sect. 2 with a description of the flow and stereo template model that we use within the simulation framework. In Sect. 3, we show examples of distance estimates using these algorithms and how these estimates are embedded into a reinforcement learning framework. We discuss our approach in the context of visual navigation and contrast it to alternative proposals in Sect. 4. Section 5 contains broader conclusions regarding our model simulations.

2 Methods

Figure 1 shows our system architecture. This architecture includes the modeling of the environment's geometry, a simple massless rigid dynamic for the robot, and a two-degree of freedom movement model as we show in the lower half of Fig. 1. The key idea of our approach is the explicit description of the states of a learner by three distances measured along $-60°$, $0°$, and $+60°$ from the optical axis, which are estimated from optic flow or stereo information. Combined with the three actions of moving forward, making left or right turns, these states describe the memory structure formalized by the matrix sketched in Fig. 1 in the robot box.

The simulation loops over multiple individual steps where a single step is a full action-perception cycle. One such cycle includes moving the robot according to its

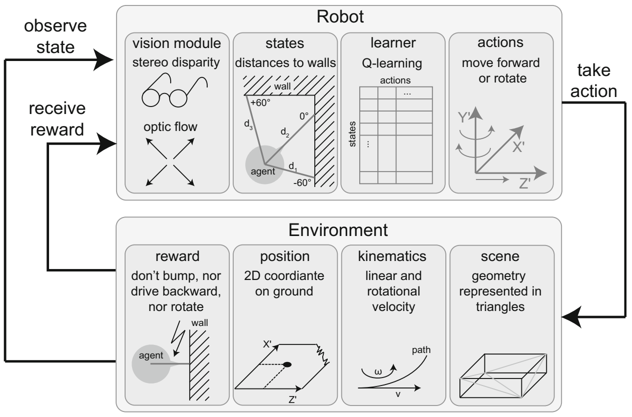


Fig. 1 Our system architecture. The robot is composed of a vision module that processes stereo disparity and optic flow, a state space, which encodes three distances to walls, and a Q-learning module adapting a matrix, which represents states and actions. Based on the observed state and the received reward (*input arrows*) the agent decides upon an action to take (*output arrow*). The simulated environment updates the 2D position based on a kinematics model with linear and rotational velocity and computes the new viewpoint of the robot in the scene defined by a triangular mesh. A reward is given depending on a collision into the wall and the agent's action

chosen action, generating the reward, acquiring flow and disparity information, passing this information along to the vision module that extracts the three distances to walls, and finally updating the memory structure.

In the next paragraphs, we describe the roles and computations performed to simulate this loop. We start with the environment and its kinematics. We briefly describe models of stereo and optic flow information. These models are restricted to curvilinear self-motion and to scenes modeled as planes. We then formulate template models that segment the scene, which is required to estimate distances toward walls. Finally, we embed the described vision module that captures the state into the reinforcement learning framework.

2.1 Simulator for the environment and its kinematics

To reduce the complexity of simulations and focus on the relevant modeling effort, we compute analytical disparity and flow that can be accomplished by knowing the geometry of the environment. No textures, material properties, or advanced physics of light transport has to be taken into account. No sophisticated ray tracing has to be performed. Intensive computations for the detection of optic flow from an image sequence and the detection of disparity signals from stereo image pairs have been sidestepped for this study. Instead the geometry of the environment, a rectangular box with optional additional walls, is defined by a triangular mesh. Non-hierarchical and non-recursive ray tracing is applied to compute the depth image (z-Buffer) for a pinhole camera model. This depth image is used by a model for optic flow or stereo disparity to define the input values for our proposed models. For the ray-triangle intersection, we use barycentric coordinates [28]. For the kinematics of the robot, we use a rigid massless model without accelerations. Instantaneous linear and rotational velocities are simulated by multiplying them with the temporal sample interval to define angles and shift values that are used in a view-port transform. In our simulation, we use the generic sample interval of one frame.

2.2 Stereo disparity and optic flow model

We give a brief explanation of model equations to introduce the constraints used in our proposed template models. Our stereo setup assumes two cameras with parallel optical axes that have a horizontal offset b, also called the inter-camera distance. We use the horizontal disparity δ introduced by the offset between the cameras. According to the geometrical constraints illustrated in Fig. 2a, the horizontal disparity is

$$\delta = f \cdot \frac{b}{Z}. \tag{1}$$

The variable Z denotes the depth of the sample point and f the focal length of a pinhole camera model.

Optic flow, the change of structured light patterns in the image plane over time, can be expressed by a model of visual image motion. For a pinhole camera moving through a rigid environment with the 3D linear velocity $\mathbf{v} = (v_x, v_y, v_z)$ and the 3D rotational velocity $\omega = (\omega_x, \omega_y, \omega_z)$, which references the point (X_0, Y_0, Z_0), the 2D velocities on the image plane are [16, 34]:

$$\begin{aligned}\dot{\mathbf{p}} &= \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \\ &= \frac{1}{Z}\begin{pmatrix} -f & 0 & x \\ 0 & -f & y \end{pmatrix}\left(\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} - \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} \times \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}\right) \\ &\quad + \frac{1}{f}\begin{pmatrix} x \cdot y & -(f^2 + x^2) & f \cdot y \\ (f^2 + y^2) & -x \cdot y & -f \cdot x \end{pmatrix}\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}\end{aligned} \tag{2}$$

Uppercase letters, e.g. X_0, denote points in the 3D world with respect to the camera's frame of reference that has its origin (0, 0, 0) at the nodal point. Lowercase letters, here x and y, denote locations in the image plane. Due to the offset of the rotation center from the camera's nodal point a rotation introduces a depth-dependent translation. That depth-dependent part is denoted by the vector cross-product (symbol $\times$) in the first term on the right-hand side of Eq. 2. The visual image motion model is visualized in Fig. 2b. The Eqs. 1 and 2 are used to compute the analytical disparity and flow information given the depth values $Z(x, y)$ from the ray-tracer together with the parameters a, b, $\mathbf{v}$, and ω, the latter two describing the self-motion of the robot with its two pinhole cameras attached.

2.3 Optic flow model for curvilinear self-motion using planes

In this section, we constrain the general model from Eq. 2 to (1) a sampling from planes and (2) curvilinear self-motion $\mathbf{v} = (0,\ 0, v_z)$, $\omega = (0,\ \omega_y,\ 0)$. A plane in Hessian form $n_x \cdot X + n_y \cdot Y + n_z \cdot Z - d = 0$ is described by its normal $\mathbf{n} = (n_x, n_y, n_z)$ and its distance d to the origin. This distance is measured along the normal. Substituting this definition in the projection $(x, y) = f/Z \cdot (X, Y)$ for a pinhole camera model gives the depth constraint $Z(x, y) = f \cdot d / (n_x \cdot x + n_y \cdot y + n_z \cdot f)$. In addition, we assume the offset of the rotation center from the nodal point to be $(X_0, Y_0, Z_0) = (a,\ 0,\ 0)$. Plugging the planar constraint, the curvilinear self-motion constraint, and the

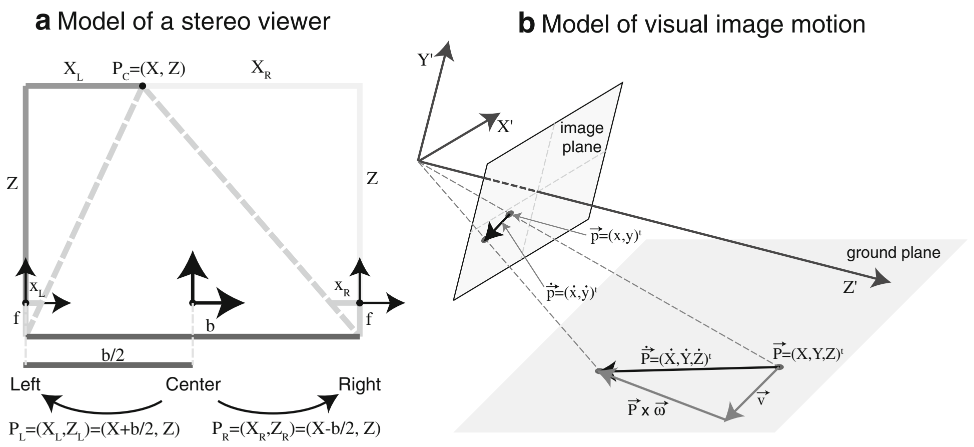


Fig. 2 Model of a stereo viewer and visual image motion using a pinhole camera. **a** Due to the horizontal displacement b between the two cameras with parallel optical axes the point of the vector $\mathbf{P} = (X, Y, Z)$ is seen in the image at location (x_L, y) from the left camera and (x_R, y) from the right camera and in general $x_L \neq x_R$. **b** If the camera is moving with the linear velocity $\mathbf{v}$ and the rotational velocity ω, a point $\mathbf{P} = (X, Y, Z)$ moves by $\dot{\mathbf{P}} = (\dot{X}, \dot{Y}, \dot{Z})$ in 3D space and its projection moves by $\dot{\mathbf{p}} = (\dot{x}, \dot{y})$ in the 2D image plane. These two models are used to describe stereo disparity and optic flow, respectively

offset constraint into Eq. 2 results in the visual motion model

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \left(v_z - a \cdot \omega_y\right) \frac{n_x \cdot x + n_y \cdot y + f \cdot n_z}{d} \begin{pmatrix} x \\ y \end{pmatrix} - \frac{\omega_y}{f} \begin{pmatrix} f^2 + x^2 \\ x \cdot y \end{pmatrix} \tag{3}$$

The plane can be further constrained. For a ground plane, the parameters are $\mathbf{n} = (\mathbf{0},\ \mathbf{1},\ \mathbf{0})$ and $d = h$. The parameter h denotes the height of the camera above the ground. For a wall-plane in direction α the normal is $\mathbf{n} = (\sin\alpha,\ \mathbf{0},\ \cos\alpha)$ and d its distance.

In the next paragraphs, we derive template models for the estimation of self-motion, the segmentation of walls from the ground, and the estimation of distances in arbitrary directions. The idea of a template model is to create prototype model responses using Eq. 1 for disparity or Eq. 3 for flow for all parameters that typically occur in the robot's environment. All these templates are compared or matched against the incoming disparity or flow information. The template with the largest similarity is selected and the parameters of that template are the result of the estimation. Neurons in various brain areas have been characterized as templates, responding to a specific set of stimulus parameters in the visual input signal, e.g. for a disparity or motion pattern [7, 23].

2.4 Template models for scene segmentation, estimation of self-motion and distances of walls

The goal is to estimate the distances of walls with the derived models. However, to apply these model Eqs. 1 and 3, the scene has to be segmented into wall and ground regions. For visual image motion model, the parameters of the curvilinear self-motion have to be estimated too. In the following paragraphs, we will first develop a template model that segments and estimates parameters of walls based on *stereo disparity*. Second, we will develop a template model for the estimation of curvilinear self-motion, segmentation of walls from the ground, and finally the estimation of walls' distances from *optic flow*.

Distances of walls estimated from stereo disparity The segmentation of walls from ground using binocular disparity is based on the observation that depth gradients from the ground appear only along the vertical axis and those of walls appear only along the horizontal axis. This gives the following constraints:

$$\begin{cases} (x, y) \in \text{Wall} & \text{if} \quad \partial_x \delta(x, y) \neq 0 \ \wedge \ \partial_y \delta(x, y) = 0 \\ (x, y) \in \text{Ground} & \text{if} \quad \partial_x \delta(x, y) = 0 \ \wedge \ \partial_y \delta(x, y) \neq 0 \end{cases}. \tag{4}$$

This formulation uses the partial derivative in vertical direction ∂_y (y-axis) and horizontal direction ∂_x (x-axis). We embed the 'equals zero'-constraints into the Gaussian tuning functions

$$f_{d,wl}(x, y) = \exp(-\left(\partial_x \hat{\delta}(x, y)\right)^2 / (2\sigma_{d,x}^2)) \text{ and} \tag{5}$$

$$f_{d,gr}(x, y) = \exp(-\left(\partial_y \hat{\delta}(x, y)\right)^2 / (2\sigma_{d,y}^2)), \tag{6}$$

with $\hat{\delta}$ denoting the sensed or input disparity. These functions are used to make the following decisions for segmentation:

Table 1 Lists model parameters with a description and their assigned values for the simulation

Description of parameter Pinhole camera model	Value
Horizontal and vertical field of view	$\alpha_{\text{fov}} = 140° \times \beta_{\text{fov}} = 140°$
Horizontal and vertical resolution	50×50 samples[a]
Dimensions of the image plane[b]	$[-0.5, +0.5] \times [-0.5, +0.5]$ cm
Range of visibility	$[0,\ 10^3]$ cm
Height of camera above ground	$h = 10$ cm
Template model for stereo disparity	
Inter-camera distance	$b = 10$ cm
Threshold value for min activity in segmentation	$\eta_{\text{disp}} = 0.4$
SD for segmentation	$\sigma_{d,x} = 0.01,\ \sigma_{d,y} = 0.1$
SD for binocular disparity	$\sigma_d = 0.1$ cm
Interval for wall's distance[c]	$d_k \in [1,\ 100]$ cm, $N_d = 99$
Sampled visual directions	$\beta_l \in [-60°,\ 0°,\ +60°]$
Template model for optic flow	
Offset between nodal point and center of rotation	$a = -10$ cm
SD for rotational velocity tuning	$\sigma_\omega = 0.573°$/frame
SD for scaled reciprocal depth tuning	$\sigma_V = 1$/frame
SD for linear velocity tuning	$\sigma_v = 0.1$cm/frame
SD for angular direction tuning	$\sigma_\alpha = 0.573°$
SD for distance tuning	$\sigma_d = \sqrt{\log(2)} = 0.8326$cm
SD for segmentation	$\sigma_{f,x} = 0.01,\ \sigma_{f,y} = 0.1$
Interval for rotational vertical velocities (yaw)	$\omega_k \in [-100, 100]°$/frame, $N_\omega = 101$
Interval for scaled, reciprocal depth	$V_k \in [10^{-5}, 10^2]$ 1/frame, $N_V = 245$
Interval for linear velocity	$v_k \in [-10^2, 10^2]$ cm/frame, $N_v = 180$
Interval for distance[c]	$d_k \in [1,\ 100]$ cm, $N_d = 99$
Sampled visual directions	$\beta_l \in [-60°,\ 0°,\ +60°]$
Reinforcement learning (Q-learning)	
Discount rate for reward	$\gamma = 0.98$
ε-Greedy, choose greedy action in ε cases	$\varepsilon = 0.7$
Learning rate	$\alpha = 0.05$
Number of epochs	$N_{\text{epochs}} = 500$
Maximum number of steps per epoch	$N_{\text{steps}} = 10{,}000$
Interval for distance values of states	$\{0,\ 10,\ 20,\ 40\}$ cm
Actions as tuple (translation, yaw rotation)[d]	$\{(+10,\ 0), (0,\ 30), (0,\ -30)\}$
Reward signal that can be accumulated, e.g. if colliding while rotating the reward is -105	$r = \begin{cases} -100 & \text{if collision} \\ -5 & \text{if rotating} \\ 0 & \text{otherwise} \end{cases}$

[a] In Fig. 3, we chose a lower resolution of 30×30 pixels to avoid too much clutter in the example plots

[b] The focal length is $f = \arctan(h_I/(2 \cdot \alpha_{\text{fov}}))$ assuming h_I being the height of the image plane

[c] We sample in log-space using $[u_1, u_N] = \exp([\log(v_1), \log(v_N)])$, applying the exponential function element-wise and using a linear equidistant sampling between $\log(v_1)$ and $\log(v_N)$

[d] Values for the translational velocities are in cm per frame and for the rotational velocities are in degrees per frame

$$\begin{cases} (x,y) \in \text{Wall} & \text{if} \quad f_{d,wl}(x,y) > f_{d,gr}(x,y) \wedge f_{d,wl}(x,y) > \eta_d \\ (x,y) \in \text{Ground} & \text{if} \quad f_{d,gr}(x,y) > f_{d,wl}(x,y) \wedge f_{d,gr}(x,y) > \eta_d \\ (x,y) \in \text{Junk} & \text{otherwise} \end{cases} \tag{7}$$

Table 1 reports the threshold value η_d. The next step uses only those image locations (x,y) that are in the visual direction β, this is the distance to be estimated. The disparity-based tuning function is

$$f_d(d_k, \beta_l) = \frac{1}{N_{\text{Wall}}} \sum_{y \in \text{Wall}} \exp\left(-\frac{\left(\log(f \cdot b/\hat{\delta}(-f\tan(\beta_l), y)) - \log(d_k)\right)^2}{2 \cdot \sigma_d^2} \right) \tag{8}$$

for sampled distances d_k in the visual directions β_l. Values for these two parameters are reported in Table 1. Applying the logarithm makes the tuning with respect to distance "log-linear" in the sense that $\log(1/d) = -\log(d)$.

Estimated values are determined by selecting the maximum and computing a locally weighted sum around the maximum location. This read-out technique enables interpolation between discrete samples of distance and direction. The solution of this estimation is denoted by $\hat{d}$ and $\hat{\alpha}$. A pseudo-code of the function is given in the Appendix Table 3. Our choice of sampling and intervals for distances d_k and directions α_l is adapted to the environment and the camera configuration in terms of the precision of *optic flow* and *stereo disparity*.

Distance and direction of walls estimated from optic flow In the following, we will develop the template model that uses optic flow to segment walls from the ground and to estimate distance and direction of walls. Equation 3 depends on the rotational and translational velocity. To eliminate the dependency on the translational velocity, we multiply by $(-y, x)$ from both sides. This results in a constraint that only depends on rotational velocity. This constraint is used to define the tuning function

$$f_\omega(\omega_{y,k}) = \frac{1}{N}\sum_{x,y} \exp\left(-\frac{\left((\hat{\dot{y}} + x\cdot y\cdot \omega_{y,k}/f)\cdot x - (\hat{\dot{x}} + (f^2+x^2)\cdot\omega_{y,k}/f)\cdot y\right)^2}{2\cdot\sigma_\omega^2}\right) \tag{9}$$

for rotational velocities $\omega_{y,k}$ and N denotes the number of flow samples. Variables with a hat-symbol are known. In this case, the flow $(\hat{\dot{x}}, \hat{\dot{y}})$ is known. We apply the same read-out method as described above for the stereo template model. This yields the estimate $\hat{\omega}_y$. For further calculations, we introduce the auxiliary variable $V = (v_z - a\cdot\omega_y)/Z$, which is the reciprocal value of the distance Z scaled by the sum the total linear velocity. With this variable and the already computed rotational velocity, the Gaussian template function is defined

$$f_V(V_k) = \exp\left(-\frac{\left(\log\left(\left\|\begin{pmatrix}\hat{\dot{x}} + (f^2+x^2)\cdot\hat{\omega}_y/f \\ \hat{\dot{y}} + x\cdot y\cdot\hat{\omega}_y/f\end{pmatrix}\right\|\right) - \log\left(\left\|V_k\cdot\begin{pmatrix}x\\y\end{pmatrix}\right\|\right)\right)^2}{2\cdot\sigma_V^2}\right) \tag{10}$$

This tuning function in Eq. 10 uses a log-tuning of the length of the target translational flow (first vector) and the template flow (second vector). To compute the length of flow vectors, we use the usual Euclidian distance norm. Again, we use the "log-linear" formulation described in the context of Eq. 8. In addition, we sample distances on a log scale in the interval from 0 to 1 m. The tuning function is defined for every image location x and y, unlike to the tuning functions from Eqs. 8 and 9 because the depth might be different at every sample point, compare with Eq. 2. To compute the maximum responding V we take the argument $\hat{V}(x,y) = \arg\max_k f_V(V_k, x, y)$ for each sample. This $\hat{V}(x,y)$ is used to segment the scene into wall and ground with the same tuning functions as given in Eqs. 4 and 7 replacing $\hat{\delta}$ by $\hat{V}(x,y)$. Furthermore, the parameters $\sigma_{d,x}$, $\sigma_{d,y}$, and η_{disp} are replaced by the parameters $\sigma_{fl,x}$, $\sigma_{fl,y}$, and η_f, respectively. Table 1 lists the values for these parameters.

With the segmentation result, we continue to resolve the scaling invariance between depth and linear velocity. Therefore, we assume the distance h of the camera above the ground to be constant and known as well as the optical axis to be parallel to the ground. These assumptions are used to formulate the following tuning function:

$$f_v(v_{z,k}) = \frac{1}{N_{\text{ground}}} \sum_{(x,y)\in\text{Ground}} \left[\cos\left(\angle\left(\begin{pmatrix}\hat{\dot{x}}_{\text{trans}} \\ \hat{\dot{y}}_{\text{trans}}\end{pmatrix}, -\frac{v_{z,k}}{h\cdot f}\begin{pmatrix}x\cdot y\\ y^2\end{pmatrix}\right)\right)\right]^+ \cdot \exp\left(-\frac{\left(\log\left(\left\|\begin{pmatrix}\hat{\dot{x}}_{\text{trans}} \\ \hat{\dot{y}}_{\text{trans}}\end{pmatrix}\right\|\right) - \log\left(\left\|-\frac{v_{z,k}}{h\cdot f}\begin{pmatrix}x\cdot y\\ y^2\end{pmatrix}\right\|\right)\right)^2}{2\cdot\sigma_v^2}\right) \tag{11}$$

with the translational input flow components $\hat{\dot{y}}_{\text{trans}} = \hat{\dot{y}} + x\cdot y\cdot\hat{\omega}/f$ and $\hat{\dot{x}}_{\text{trans}} = \hat{\dot{x}} + (f^2+x^2)\cdot\hat{\omega}/f$, N_{ground} the number of samples from the ground, $\angle(\mathbf{a}, \boldsymbol{b})$ the angular difference between the vectors **a** and **b**, and the half-wave rectification $[\,x\,]^+ = \max(0, x)$. The overall tuning uses a polar representation of flow vectors that is split in two separate functions: The rectified cosine function, the first factor on the right-hand side of Eq. 11, and the Gaussian function, the second factor. To evaluate the tuning function in Eq. 11, the same read-out technique is applied as described above. This results in the velocity estimate $\tilde{v}_z$. To compute the pure translational velocity, the rotational velocity has to be scaled by the parameter a and this product is added to the formerly estimated translational velocity. This results in $\hat{v}_z = \tilde{v}_z + a\cdot\hat{\omega}$. With this step, the estimation of the self-motion parameters is complete and, next, we will show how to estimate the distances of walls.

We estimate the distances of walls based on a model for the translational flow that depends inversely on distance. With the already estimated rotational velocity $\hat{\omega}$, we reconstruct the translational flow $(\hat{\dot{x}}_{\text{trans}}, \hat{\dot{y}}_{\text{trans}})$ from the input flow by subtracting the model rotational flow. With the total translational velocity $\tilde{v}_z$, we resolve the scaling ambiguity between depths and translational velocity. We define the tuning function for distances as

$$f_d(d_k, \beta_l) = \frac{1}{N_{\text{wall}}} \sum_{y \in \text{Wall}} \exp\left(- \frac{\left(\log\left(\left\| \begin{pmatrix} \dot{\hat{x}}_{\text{trans}}(-f\tan(\beta_l), y) \\ \dot{\hat{y}}_{\text{trans}}(-f\tan(\beta_l), y) \end{pmatrix} \right\| \right) - \log\left(\left\| \frac{\tilde{v}}{d_k} \begin{pmatrix} -f\tan(\beta_l) \\ y \end{pmatrix} \right\| \right) \right)^2}{2 \cdot \sigma_d^2} \right) \quad (12)$$

Eq. 12 has a similar structure compared to the tuning function in Eq. 10. Applying the logarithm to the length of the flow vectors makes the tuning "log-linear" with respect to the unknown distance value (compare also with the explanation given after Eq. 10). The distance $\hat{d}$ itself is estimated by the same read-out technique as used before. A summary of all steps gives Appendix Table 4.

2.5 Reinforcement learner

We chose Q-learning [32, 33] combined with an ε-Greedy strategy for learning. This assumes that the current state fully describes the agent's situation in the environment for our problem specification (Markov property) and it assumes a finite state and action space (finite Markov decision process). In total, there are 3^3=27 states sampling each of three distances by three values. The robot is able to move along the optical axis or rotate clockwise or counterclockwise around the vertical axis (yaw rotation). A discretization of these variables is reported in Table 1. For a collision the agent receives -100 as reward and -5 when rotating. These rewards accumulate, e.g. a reward of -105 is received if a collision occurs while rotating. As metric, we report the path length of an episode, which we limit to a maximum of 10,000 steps to define a termination criterion. An outline of the applied reinforcement algorithm can be found in Appendix Table 5.

3 Results

In the first part, we show an example for the proposed template methods and their representations. The second part takes these algorithms and embeds them into a reinforcement learning method (Q-learning) and evaluates their performance based on the path length with no collision.

3.1 Flow and stereo information are used as cues to segment walls from the ground and to estimate the distance of walls

To illustrate the above template models for analytical flow and stereo information we applied them to the example scene shown in Fig. 3a. The robot is positioned close to the right wall in a rectangular box (30 × 60 cm) and its optical axes are rotated by 30° with respect to the horizontal. Figure 3b, c shows maps of the depth and disparity, respectively. These maps illustrate the applied constraint, which is used for the segmentation of the scene into wall and ground. For the ground, a gradient in only the vertical dimension exists. This gradient is encoded by the varying gray-value profile in Fig. 3c, which is visible in the lower half of the image. Thus, partial derivatives in the vertical direction are non-zero and partial derivatives in the horizontal direction are zero. For the wall, a gradient in only the horizontal direction is present as visible by the gray-value gradient in the depth map in Fig. 3b. In this case, the partial derivative in the vertical direction is zero and the derivative in the horizontal direction is non-zero. These constraints are used in the above template model to achieve a segmentation based on stereo or flow information as shown in Fig. 3d, f, respectively.

In both cases, the segmentation into wall and ground is quantitatively correct leaving some points at the boundary unclassified. To compute partial derivatives boundary values are not evaluated as seen by the ring of non-classified flows in Fig. 3f. The gap between the two areas of wall and ground is explained by applying thresholds (η_d or η_f) to the output values of the tuning functions that integrate the above constraints about partial derivatives being zero-valued.

Once the segmentation is achieved, the three distances are estimated using samples from walls. Figure 3e shows the 1D tunings that include the disparity information of walls. In this example, three distances from two walls are estimated.

For flow, the distances of walls are not directly accessible. Instead the constraint, Eq. 3 depends on the self-motion of the robot besides these distances. In our template model, we split the problem of estimating self-motion and the wall's distance into three template match problems. First, the rotational velocity is estimated by template matching. Figure 3g shows the activation of templates in its first sub-panel. Second, the linear velocity scaled by depth is estimated using the already estimated rotational velocity in a template match problem. This relative velocity information $V(x, y)$ scaled by the reciprocal depth is used to segment the scene. Next, samples from the ground are used to estimate the absolute linear velocity, knowing the distance of the camera above ground and

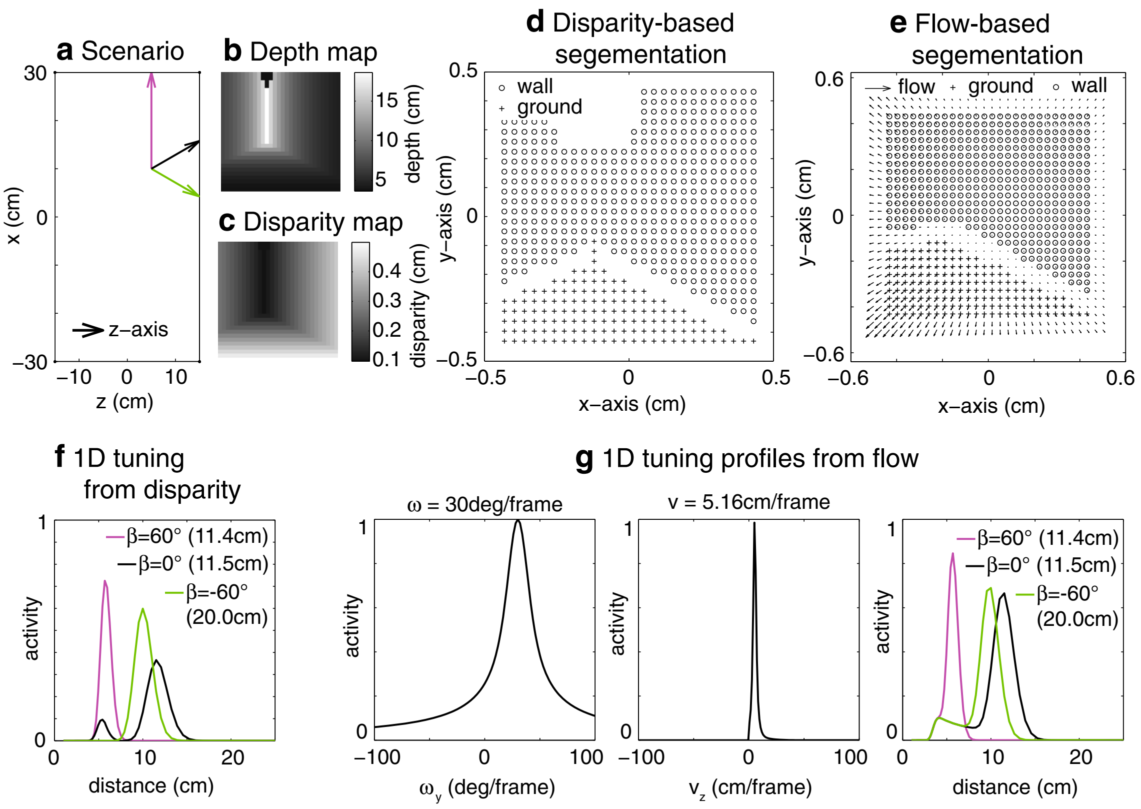


Fig. 3 Segmentation of walls from the ground and estimation of distances of walls based on flow or stereo information. **a** Shows the scenario with the robot at the location $x_0 = 10$ cm, $z_0 = 5$ cm, and orientation $\varphi = 30°$ that moves with the linear velocity $v_{z,gt} = 0$ cm/frame and rotational velocity $\omega_{y,gt} = 30°/\text{frame}$. For this position the ground-truth values for the wall distances are $d_1 = 20$ cm and $d_2 = d_3 = 11.55$ cm for the visual directions β=60°, β=0°, and $\beta = -60°$, respectively. **b** Shows the depth map for a simulated pinhole camera with a 140°× 140° field of view and a 30 × 30 pixel resolution. **c** The disparity map is computed assuming an inter-camera distance of 10 cm. **d** Shows the disparity-based segmentation. Points of the wall are indicated by circles and points from the ground are indicated by crosses. **e** Shows the flow-based segmentation into wall versus ground together with the input flow. **f** Shows three 1D tuning functions for disparity. A local vector sum read-out of the three 1D functions gives the indicated estimates. **g** The estimation of self-motion and distance is split into three 1D template-matching problems. Their 1D response activations are shown in the three sub-panels. Self-motion velocities are estimated as $\tilde{v}_{z,est} = 5.16$ cm/frame, $\omega_{y,est} = 30°/\text{frame}$, and the distance estimates are given in the third panel

assuming that the camera's optical axes are parallel to the ground. The template responses for the linear velocity show the second sub-panel of Fig. 3g. With both velocities and the segmentation result the distances toward walls are estimated in another template match problem. The activity of templates shows the third sub-panel of Fig. 3g. The distances along the visual directions $\beta = -60°$, $\beta = 0°$, and $\beta = +60°$ are estimated with high accuracy with an error less than one centimeter in this example. These estimates are plotted in Fig. 3a by magenta, black, and green colored arrows, respectively.

3.2 The number of collisions decreases by learning based on distance and direction estimates from stereo and flow information

Our model learns to avoid collisions with walls by applying Q-learning with 27 states, which represent distances discretized into three bins along three visual directions. In each state, three actions are possible: Moving forward or rotating clockwise or counterclockwise. Thus, the Q-matrix has in total 81 entries which predict future reward for each state-action pair in 500 epochs each a maximum of 10,000 steps long. An epoch ends earlier if a collision occurs and the robot is randomly re-positioned and re-oriented in the rectangular box. The agent has to learn to avoid collisions. We choose path length as error metric for the learning of collision avoidance. This is, the distance driven between the start and end of an epoch. This error metric rates trivial solutions poorly, like rotating in one location. Such solutions are avoided by giving negative reward whenever the robot decides to rotate. We tested this learning in six different experiments choosing either a simple box model or a more complex environment that has several walls as additional barriers, see Fig. 4a, e. For each of these environments, we run the learning algorithm with ground-truth

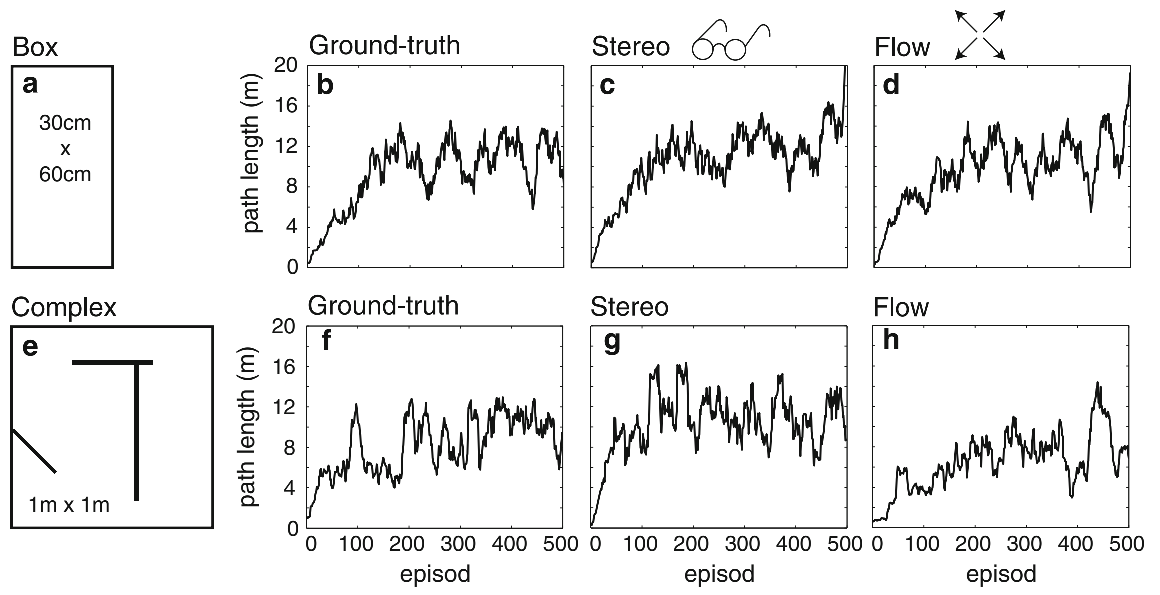


Fig. 4 Learning of collision avoidance based on the distance estimates of walls from flow or stereo information. **a** Shows a top–down view of the simple box environment. The path length is the driven distance between start and end of an epoch. The first row shows the path length for the simple box environment when **b** using ground-truth distance, **c** distance estimates from stereo information, or **d** distance estimates from flow information. **e** The complex environment has several walls and is larger than the simple one. Note **e** is not drawn to scale to match **a**. Path lengths for **f** ground-truth distances, **g** distances estimated from stereo information, and **h** distances estimated from flow information

Table 2 The greedy policy for learned collision avoidance with ground-truth distance input when trained in the simple box

d_1	d_2	d_3	Action	d_1	d_2	d_3	Action	d_1	d_2	d_3	Action
10	10	10	CW	10	10	20	CCW	10	10	40	CCW
20	10	10	CW	20	10	20	→	20	10	40	CCW
40	10	10	CW	40	10	20	CW	40	10	40	→
10	20	10	CCW	10	20	20	→	10	20	40	CCW
20	20	10	→	20	20	20	→	20	20	40	→
40	20	10	→	40	20	20	→	40	20	40	→
10	40	10	→	10	40	20	→	10	40	40	→
20	40	10	→	20	40	20	→	20	40	40	CW
40	40	10	→	40	40	20	CW	40	40	40	CW

The symbol → encodes the forward moving action, CW a clockwise rotation, and CCW a counterclockwise rotation

distance information and Fig. 4b, f shows the results, respectively. Or we run the algorithm with distances estimated from stereo information. Results show Fig. 4c, g, respectively. Or we run the algorithm with distances estimated from flow information. Results show Fig. 4d, h, respectively. For the simple box environment, the path length increases rapidly from a few centimeters to about 8–10 m within the first 100 to 200 epochs. In case of the complex environment, the path length increases too; however not as rapid and also stays at a lower level, in particular when using flow information, see Fig. 4h.

Table 2 shows the greedy policy for learning with ground-truth distances in the simple box environment. For each state, we show the most rewarding future action. For small distances of approximately 10 cm the robot learned to make turns. See the first four rows in the Table 2. For instance, for the state $(d_1, d_2, d_3) = (40, 10, 10\ \text{cm})$ the robot learned to make a CW turn, which is the turn into the direction where $d_1 = 40$ cm, which measures a larger distance than $d_3 = 10$ cm. In the "complementary" case of $d_1 = 10$ cm and $d_3 = 40$ cm the robot makes a CCW turn. See the first row in the last of the three blocks in Table 2.

4 Discussion

We showed successful learning of visual navigation using a small finite state space (27 values) representing distances toward walls along three visual directions, which are estimated by biologically inspired models using stereo, flow information, and learning. To estimate these distances, a segmentation of walls from ground has to be performed. We developed a model for such segmentation. When flow information is used, the model estimates self-motion, in addition. Distance estimates of our model are accurate enough to learn collision avoidance in a virtual environment. Our model, which uses stereo input, performs slightly better than our model with flow input. For the model with flow input, we showed that estimation problems for rotational velocity, linear velocity, and wall distances can be separated into 1D parameter search problems. These 1D problems have been solved using a voting-based method looking for the best matching flow or disparity given parameterized models.

More generally, our results show that learning of collision avoidance is possible using a small state space whereas the state information is extracted from sensory input. Prescott and Mayhew [21] used the same definition for a state space choosing distances along three visual directions and trained with a variant of the adaptive critic algorithm. More recent approaches use larger state spaces. For instance, Huang et al. [13] used eight simulated infrared ray sensors with maximum range and continuous output to represent the state of the agent in the environment. To reduce the number of possible states they deploy a three-layer back-propagation network as a mapping between the sensory output and a smaller state space. An alternative approach clustered states together with prototypical actions to reduce the search effort (Milan, 1995). This reflects the traditional view of using large state spaces that are than mapped to smaller representations using function approximation [4, 31], e.g. by deploying neural networks. Here, we suggest a small state space with the number of states largely decreased by choosing them task related, here to avoid collisions.

Our model can be categorized as a map-less indoor navigation approach integrating either stereo or flow information. Other approaches for navigation could be categorized as either indoor or outdoor, using a map or not, operating in structured or unstructured environments. Unstructured environments are defined by their absence of track-able features. Aside from the fact of generating a map-based representation of the environment [11, 30], localization is achieved by detecting and tracking landmarks [2]. Approaches that used optic flow for visual navigation did not include a learning component [9, 26]. Extensive surveys on visual navigation can be found in DeSouza and Kak [8] as well as in Bonin-Font et al. [6].

Our model bears some similarities to existing flow template models and the Hough transform, in general. Perrone suggested a template model for the estimation of self-motion [24] and fixating self-motion [25] from optic flow. The idea of setting up flow templates for parameters of self-motion and matching them to the observed input flow motivated us to generalize this approach and extend it to non-self-motion parameters such as the geometry of the scene. Furthermore, we applied the same idea of a template model to binocular disparity to estimate distances. A similar idea using structural models such as the plane model has been proposed by Adiv [1] for the segmentation of object motions in the flow. More generally, all these methods are voting schemes, like the Hough transform [10].

Our template model makes several assumptions. For the camera we use a pinhole model assuming an optical axis parallel to the ground. The environment consists of planes that are orthogonal to the ground. The robot has access to the height of the camera above ground. This height stays constant over time. Self-motions of the robot are reduced to curvilinear motion, moving with a translational velocity along the optical axis and rotating around the vertical axis (yaw). Nevertheless, all these assumptions are reasonable in the sense that such requirements could be met when setting up a robot. The most critical assumption is keeping the optical axis parallel to the ground.

For the learner, we used an indoor environment at the scale of a few tens of centimeters that has walls orthogonal to the planar ground and the robot goes at a speed of 2 cm/frame. In our setup, 40 cm is the upper bound for the distance toward a wall measured along the optical axis. This 40 cm distance translates into 20 steps before a collision with a wall occurs, assuming a straight path. This gives the robot enough frames to make several rotations to avoid a collision with the wall.

We used two abstractions to accelerate simulations. The simulated environment is reduced to the geometry of walls and ground and does not include textures, material properties, etc. This allows for the direct computation of depth by ray tracing without using recursions. The second abstraction is the use of analytical flow and disparity information, which is directly provided to the suggested template models. We call this information analytical disparity or flow, respectively. This excludes the detection of flow from image sequences and stereo disparity from image pairs. Both tasks are computationally expensive and have been studied extensively [3, 5, 27].

Successful visual navigation includes, but is not limited to collision avoidance. For instance, Michels et al. [19] proposed the reconstruction of relative depths from monocular images that are fed into a reinforcement learning policy to learn steering for autonomous driving. Others focused on navigating toward a target [36]. Their approach is most similar to ours in terms of the employed state space. This space is defined by pseudo-disparity, the difference between the horizontal positions of the target in left and right images, and the targets image coordinate in the left image. However, due to a high-resolution discretization of these variables Zhu and Levinson [36] end up with 90,000 states. This large number of states is reduced by working with a conceptual state space. That conceptual state space starts with a coarse discretization of the above variables and adaptive refinement (state splitting). Our approach is working with a coarse state space defined by the distances along three visual directions, which are estimated from stereo or flow information using bio-inspired template models.

5 Conclusion

Our model learns to avoid collisions with walls by applying Q-learning with 27 states, which represent distances discretized into three bins along three visual directions. In each state, three actions are possible: Moving forward or rotating clockwise or counterclockwise. Thus, the Q-matrix has in total 81 entries which predict future reward for each state-action pair in 500 epochs each a maximum of 10,000 steps long. An epoch ends earlier if a collision occurs and the robot is randomly re-positioned and re-oriented in the rectangular box. The agent has to learn to avoid collisions. We choose path length as error metric for the learning of collision avoidance. This is the distance driven between the start and end of an epoch. This error metric rates trivial solutions poorly, like rotating in one location. Such solutions are avoided by giving negative reward whenever the robot decides to rotate. We tested this learning in six different experiments choosing either a simple box model or a more complex environment that has several walls as additional barriers, see Fig. 4a, e. For each of these environments, we run the learning algorithm with ground-truth distance information and Fig. 4b, f shows the results, respectively. Or we run the algorithm with distances estimated from stereo information. Results show Fig. 4c, g, respectively. Or we run the algorithm with distances estimated from flow information. Results show Fig. 4d, h, respectively. For the simple box environment, the path length increases rapidly from a few centimeters to about eight to 10 m within the first 100–200 epochs. In case of the complex environment, the path length increases too; however, not as rapid and also stays at a lower level, in particular when using flow information, see Fig. 4h.

Acknowledgments All authors are supported in part by CELEST, an NSF Science of Learning Center (SMA-0835976). FR acknowledges support from the Office of Naval Research (ONR N00014-11-1-0535 and ONR MURI N00014-10-1-0936). MV acknowledges support from the National Aeronautics and Space Administration (NASA NNX12AH31G).

Appendix

We provide pseudo-code for the bio-inspired proposed models that estimate distances of walls from stereo or flow information (Tables 3, 4, 5).

Table 3 Pseudo-code for the algorithm stereo-based template model

```
Dist = distDirFromDisparity(D, X, b, f, Beta)
// D – disparity map.
// X – horizontal image coordinates.
// b – inter-camera distance.
// f – focal length of the camera.
// Beta – visual directions for distance estimates.
// Dist – distances of walls along visual direction.
DiffX = [1 0 -1; 2 0 -2; 1 0 -1];
DiffY = -DiffX';
TuneDiffXOfD = exp(-filter(D,DiffX)^2/(2$\sigma$^2));
TuneDiffYofD = exp(-filter(D,DiffY)^2/(2$\sigma$^2));
If (TuneDiffXOfD > TuneDiffYofD) & (TuneDiffXOfD > eta) then
    samples from the ground are X_g and D_g.
Elseif TuneDiffYOfD > eta then
    samples from the wall are X_w and D_w.
End
TuneWall = zeros(betaNum,distNum);
For beta from Beta do
    xImg = -f*tan(beta);
    Dw = interp1(D(-f*tan(beta),allY)); // interpolate
    For dist from minDist to maxDist,
        For Y from minW to maxW
            TuneWall(beta,dist) += exp(-(log(f*b/Dw)-log(dist))^2/(2$\sigma$^2));
        End
    End
Readout by taking the argument of max response for each beta.
```

Table 4 Pseudo algorithm for flow-based template model

```
[v, omega, Dist] = velDistDirFromFlow(Dx, Dy, X, Y, f, h, a, Beta)
// Dx – horizontal component of flow.
// Dy – vertical component of flow.
// X – horizontal image coordinate.
// Y – vertical image coordinate.
// f – focal length of pinhole camera.
// h – height of camera above ground.
// a – distance between nodal point and center of rotation.
// v – linear velocity along the optical axis.
// omega – rotational velocity around the vertical axis (yaw).
// ang – angular direction of the detected wall.
// Dist – Distances along visual directions Beta.
Estimate omega from Dx, Dy, X, and Y.
Estimate auxiliary V = (v-a*omega)/Z from Dx, Dy, X, and Y.
Compute the gradients Vx = dV/dx and Vy = dV/dy.
TuneVx = exp(-Vx^2/(2σ^2));
TuneVy = exp(-Vy^2/(2σ^2));
If (TuneVx > TuneVy) & (TuneVx > eta) then
Define ground samples as Dx_g, Dy_g, X_g, and Y_g.
Elseif TuneVy > eta then
Define wall samples as Dx_w, Dy_w, X_w, and Y_w.
End
Estimate vel from Dx_g, Dy_g, X_g, and Y_g, knowing that the depth is given by Z = f*h/y. Then set v = vel + a*omega.
Estimate Dist from Dx_w, Dy_w, X_w, and Y_w, knowing omega and vel that have been estimated.
```

Table 5 Q-learning algorithm with ε-Greedy action selection

```
Initialize Q with zeros.
For all episodes do
Initialize s1 randomly.
While s1 is not terminal or max step is not reached do
If rand < eps then
a1 = arg max_a Q(s1,a);
// For multiple maxima chose one at random.
Else
Choose a1 randomly.
End
// Take action a1 and observe reward r and state s2.
Q(s1,a1) <- Q(s1,a1)
+ alpha * (r + gamma * max_a2(Q(s2,a2))-Q(s1,a1));
s1 <- s2;
End
End
```

References

1. Adiv G (1985) Determining three-dimensional motion and structure from optical flow generated by several moving objects. IEEE Trans Pattern Anal Mach Intell PAMI–7(4):384–401
2. Adorni G, Cagnoni S, Enderle S, Kraetzschmar GK, Mordonini M, Plagge M, Ritter M, Sablatng S, Zell A (2001) Vision-based localization for mobile robots. Robot Auton Syst 36:103–119
3. Baker S, Scharstein D, Lewis JP, Roth S, Black MJ, Szeliski R (2011) A database and evaluation methodology for optical flow. Int J Comput Vis 92(1):1–31
4. Baird LC (1995) Residual algorithms: Reinforcement learning with function approximation. In: Prieditis A, Russell S (eds) Proceedings of the twelfth international conference on machine learning. Morgan Kaufmann, San Francisco, pp 30–37
5. Barron J, Fleet D, Beauchemin S (1994) Performance of optical flow techniques. Int J Comput Vis 12(1):43–77
6. Bonin-Font F, Ortiz A, Oliver G (2008) Visual navigation for mobile robots: a survey. J Intell Robot Syst 53:263–296
7. Cumming BG, DeAngelis GC (2001) The physiology of stereopsis. Annu Rev Neurosci 24:203–238
8. DeSouza GN, Kak AC (2002) Vision for mobile robot navigation: a survey. IEEE Trans Pattern Anal Mach Intell 24(2):237–269
9. Dev A, Krose B, Groen F (1997) Navigation of mobile robot on the temporal development of the optic flow. Proc Intell Robots Syst (IROS) 2:558–563
10. Duda RO, Hart PE (1972) Use of the Hough transformation to detect lines and curves in pictures. Commun ACM 15(1):11–15
11. Franz MO, Schlkopf B, Mallot HA, Blthoff H (1998) Learning view graphs for robot navigation. Auton Robots 5:111–125
12. Gaskett C, Fletcher L, Zelinsky (2000) A Reinforcement learning for a vision based mobile robot. In: Proceedings of the IEEE conference on intelligent robots and systems (IROS), pp 403–409
13. Huang B-Q, Cao G-Y Guo M (2005) Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance. In: Proceedings of the 4th international conference on machine learning and cybernetics, pp 85–89
14. Kim D, Sun J, Oh SM, Rehg JM, Bobick AF (2006) Traversibility classification using unsupervised on-line visual learning for outdoor robot navigation. In: Proceedings of IEEE international conference on robotics and automation, Orlando, Florida, pp 518–525
15. Lemaire T, Berger C, Jung I-K, Lacroix S (2007) Vison-based SLAM: stereo and monocular approaches. Int J Comput Vis 74(3):343–364
16. Longuet-Higgins HC, Prazdny K (1980) The interpretation of a moving retinal image. Proc R Soc Lond Ser B Biol Sci 208:385–397
17. Lever C, Burton S, Jeewajee A, O'Keefe J, Burgess N (2009) Boundary vector cells in the subiculum of the hippocampal formation. J Neurosci 29(31):9771–9777
18. Marinez-Marin T, Duckett T (2005) Fast reinforcement learning for vision-guided mobile robots. In: Proceedings of the IEEE conference on robotics and automation (IROS), Spain, pp 4170–4175
19. Michels J, Saxena A, Ng AY (2005) High speed obstacle avoidance using monocular vision and reinforcement learning. In: Proceedings of 22nd international conference on machine learning, Bonn, Germany, pp 593–600
20. Millan JR (1995) Reinforcement learning of goal-directed obstacle-avoiding reaction strategies in an autonomous mobile robot. Robot Auton Syst 15:275–299
21. Prescott TJ, Mayhew JEW (1992) Obstacle avoidance through reinforcement learning. In: Moody JE, Hanson SJ, Lippman RP (eds) Advances in neural information processing systems 4. Morgan Kaufmann, San Mateo, pp 523–530
22. Ohya A, Kosaka A, Kak A (1998) Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. Robot Autom 14(6):969–978
23. Pack CC, Born RT (2008) Cortical mechanisms for the integration of visual motion. In: Masland RH, Albright T (eds) The senses: a comprehensive reference, vol 2, pp 189–218
24. Perrone J (1992) Model for the computation of self-motion in biological systems. J Opt Soc Am A 9(2):177–192
25. Perrone J, Stone L (1994) A model of self-motion estimation within primate extrastriate visual cortex. Vis Res 34(21):2917–2938

26. Santos-Victor J, Sandini G, Curotto F, Garibaldi S (1993) Divergence stereo for robot navigation: Learning from bees. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 434–439
27. Scharstein D, Szeliski R (2002) A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int J Comput Vis 47(1/2/3):7–42
28. Shirley P, Marschner S (2009) Fundamentals of computer graphics, 3rd edn. A K Peters Natick, Massachusetts
29. Solstad T, Boccara CN, Kropff E, Moser M-B, Moser EI (2008) Representation of geometric borders in the entorhinal cortex. Science 332:1865–1868
30. Strsslin T, Sheynikhovich D, Chavarriaga R, Gerstner W (2003) Robust self-localization and navigation based on hippocampal place cells. Neural Netw 18:1125–1140
31. Sutton RS (1996) Generalization in reinforcement learning: successful examples using sparse coarse coding. In: Touretzky DS, Mozer MC, Hasselmo ME (eds) Advances in neural information processing systems: proceedings of the 1995 conference. MIT Press, Cambridge, pp 1038–1044
32. Sutton RS, Barto AG (1998) Reinforcement Learning—an Introduction. MIT Press, Cambridge
33. Watkins CJCH, Dayan P (1992) Q-learning. Mach Learn 8:279–292
34. Waxman A, Duncan JH (1986) Binocular image flows: steps toward stereo-motion fusion. IEEE Trans Pattern Recognit Mach Intell PAMI–8(6):715–729
35. Yue S, Rind C, Keil M, Cuadri J, Stafford R (2006) A bio-inspired visual collision detection mechanism for cars: optimisation of a model of a locust neuron to a novel environment. Neurocomputing 69:1591–1598
36. Zhu W, Levinson S (2001) Vision-based reinforcement learning for robot navigation. In: Proceedings of international joint conference on neural networks, Washington DC, pp 1025–1030