SEVENTH FRAMEWORK PROGRAMME PRIORITY:

ICT FET Open

CURVED ARTIFICIAL COMPOUND EYES

Deliverable D3.1

Methods for optic-flow extraction from arbitrary curved eyes

Fabian Recktenwald
Chunrong Yuan
Hanspeter A. Mallot

Eberhard Karls University of Tübingen

Project number FP7-2009-ICT-237940

Abstract

This report describes theoretical work on algorithmic solutions for measuring optic flow. It discusses the applicability of these solutions on curved compound eyes with fixed curvature as well as other constrained sensor geometries.

Furthermore, it presents a tool for the evaluation of the presented methods with arbitrary sensor geometries. This tool consists of a simulation of the sensor output together with ground truth data of motion and optic flow.

Executive Summary

After an introduction to the problem of optic flow computation with respect to the CURVACE sensor we present an overview of algorithmic solutions to this problem. Each of these methods are discussed with respect to their applicability to the CURVACE sensor.

There are three important issues to be considered for the CURVACE sensor: it has a non-uniform geometry, a low spatial and a high temporal resolution. Additionally, we have different performance constraints depending on the application scenario. For on-chip processing of optic flow we need high performance methods with low memory consumption that usually yield low quality optic flow. For this purpose we review existing optic flow algorithms and discuss novel modifications of these algorithms for application to the CURVACE sensor. Particularly we propose a new method for optic flow estimation by integrating normal flow with a Kalman filter utilising the high temporal resolution of the CURVACE sensor. For offline processing of sensor data, e.g. for calibration purposes, we need high quality optic flow and can afford complex algorithmic solutions since performance is not a main factor in this case.

The second part of this report demonstrates the CURVACE simulation tool which is used for the evaluation of optic flow methods. There are several reasons to create a simulation for this purpose. First, the CURVACE sensor is not available during the initial phase of the project, second it is possible to test different sensor layouts without the need to build them. Third, in a simulation it is easier to set up specific test environments and to provide ground truth data for evaluation purposes.

The simulation software can be used for other purposes apart from optic flow evaluation. Basically all application scenarios such as using the sensor on a flying vehicle, which can be tested using simulation in a controlled environment.

The simulation itself consists of a rendering system for 3D environments and a system for the computation of the sensor output from the rendered scene. It will be extensible through a plugin system thus allowing a high flexibility.

Reference Documents

TA – Technical Annex of the project (Annex I – Description of Work), v2 /07/09/2010

D1.1 – Deliverable 1.1. Ommatidia technical specifications, March 2010

Table of Contents

1 Introduction

The main task of WP3 (Visual Processing) is the development of methods to process visual information gathered with the CURVACE sensor. Those methods will optimally exploit the specificities of CURVACE such as wide field of view and high temporal frequency. They also have to deal with low spatial resolution and distributed processing on microcontrollers.

1.1 Measuring optic flow

Optic flow is the apparent motion of brightness patterns in an image sequence ([HS81],[Gi50], [Gi66]). It arises mainly from the relative motion of objects and the viewer but also from illumination changes in the scene. Methods for optic flow computation usually impose the so called brightness constancy assumption. The brightness of a point in the scene is assumed to stay constant over time and optic flow is only related to motion. Thus the brightness constancy assumption can be used to extract the motion of the viewer as well as the motion of other objects in the field of view.

A sequence of images is described as the intensity of each pixel at position (x,y) and time t as I(x,y,t). The problem of optic flow estimation consists of recovering the motion field (u(x,y),v(x,y)). Applying the brightness constancy assumption yields the equation

$$I(x,y,t)=I(x+u,y+v,t+1) \tag{1}$$

which is the basis of many differential methods for optic flow estimation. From this equation we have two unknowns u and v and it is not possible to compute both from the single equation above. To solve this problem, additional constraints are required. These usually consist of some smoothness constraints imposed on the motion field.

2 Optic flow on the CURVACE sensor

There already exist many approaches to the problem of computing optic flow from an image sequence. These algorithms are usually applied to perspective images with medium resolutions (e.g. 640x320 pixel) and the results are used for further processing like motion estimation or image segmentation. To identify a suitable method for optic flow extraction on the CURVACE sensor, we need to analyse how the specialities of the CURVACE sensor will influence the processing of optic flow.

2.1 Cylindrical CURVACE sensor

First, the cylindrical CURVACE sensor has a special imaging geometry which differs from a perspective projection. Instead it provides a cylindrical projection of the environment. This projection can be unwrapped to a planar image. While perspective images usually yield distortions at the image borders for large fields of view (especially fish-eye lenses have large distortions), the projection of the CURVACE sensor is homogeneous over the whole horizontal field of view. Thus, we can expect better optic flow results than with perspective images when employing existing algorithms.

Second, the CURVACE sensor has a high temporal resolution. This is another advantage which simplifies the application of existing flow algorithms. Estimation of optic flow becomes more reliable the smaller the motion vectors are. With a high temporal resolution the motion between consecutive frames is comparatively small. Due to these small motion vectors, approaches like pyramidal image coarsening are not necessary.

Optic flow algorithms dedicated to the CURVACE device will of course take into account the general theory of optic flow. On top of this, they will have to deal with and exploit the special advantages such as homogeneous but sparse sampling of large visual fields and high temporal resolution. In Section 2.3, we will analyse existing methods and discuss modifications needed for application to the CURVACE device.

2.2 Other CURVACE sensors

In the case of other CURVACE sensors the approach to optic flow computation depends on the specific shape of the sensor. Consider first a free form linear strip of ommatidia. If N is the number of ommatidia on the strip, we need at least 2N+1 variables to describe the geometry (3 coordinate values for the first ommatidium and a 3D unit vector for each additional one). From the intensity values in two time frames, we may obtain 2N measurements from which we would need to recover the local motion vectors (2 variables per ommatidium) plus the 2N+1 geometry variables. Since the number of unknowns thus exceeds the number of measurements by 4N+1 to 2N, a solution of this problem will not be possible. In order to tackle this problem, additional constraints must therefore be imposed. First, the sensor geometry will be constrained by limited curvature or, similarly, to surfaces described globally by a small number of parameters. Second, small patches of the CURVACE sensor might remain connected rigidly, in which case the geometry parameters would have to be specified only once for each patch. Third, the speed of surface change can be limited so that the geometry parameters remain constant over a larger set of image frames.

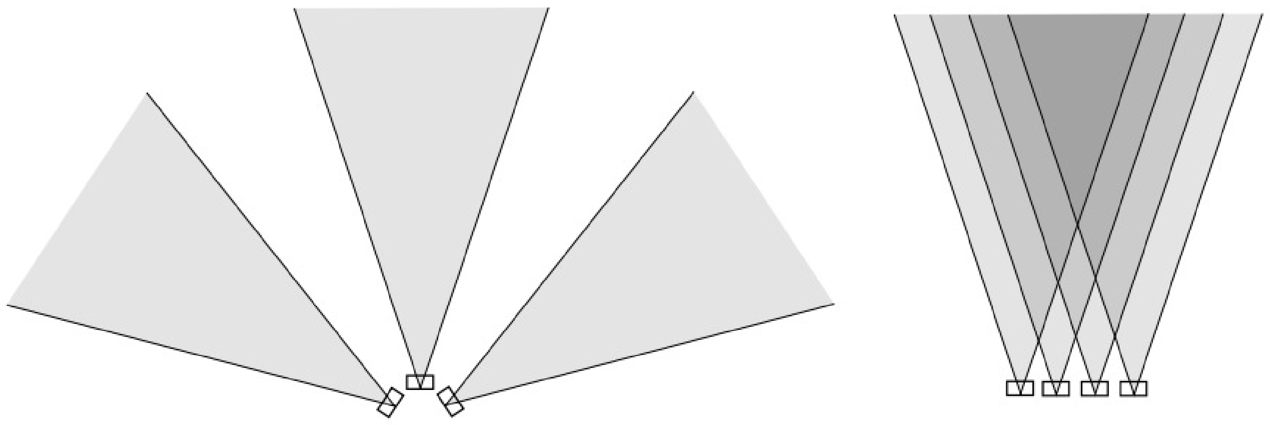


Fig. 1: ommatidia layouts with large curvature (left) and zero curvature (right)

The curvature of the CURVACE sensor should be confined to a specified range also for another reason: Zero curvature is not desirable because it results in a flat sensor layout yielding a kind of parallel projection where the field of view is only determined by the ommatidia acceptance angle. A curvature which leads to gaps between the viewfield of adjacent ommatidia (Fig. 1) is also problematic, since this would create aliasing artefacts. Thus the maximum curvature is defined by the acceptance angle of the ommatidia.

If these requirements are met and a layout of ommatidia similar to the cylindrical CURVACE sensor is used, the same optic flow algorithms can be used for both types of sensors. A calibration step is still necessary for this constrained CURVACE sensor, which means that the position and orientation of all ommatidia must be determined. This is needed to relate the optic flow, that is calculated in image space, to the motion of the sensor in world space (e.g. for egomotion estimation). This step can however be separated from the optic flow computation itself.

Finally we have to consider CURVACE sensors which do not have a regular and rectangular pixel arrangement. This would be the case for CURVACE strips, or for spherical CURVACE sensors, where a rectangular pattern of ommatidia is not given. A different distribution of pixels only means that the image space is sampled at different positions. Resampling the sensor output with a regular pattern would be one option which allows to use the presented optic flow algorithms without further adjustments, but this would be time consuming. However the main difference is that image gradients can not be computed in the usual way. Thus it is sufficient to develop a method to compute the image gradients from arbitrary positioned pixels (See also 2.3.2 Minimal Normal Flow).

2.3 Motion algorithms

The following section presents a selection of algorithms for the computation of optic flow. This selection is based on the discussed properties of the CURVACE sensor. As for processing of optic flow on a microcontroller we applied two main criteria. First, the methods should exploit the high temporal resolution of the sensor. Thus they must be efficiently computable on a microcontroller. Second, they must be computable with local information. This means that the number of pixels contributing to a single flow result should be as small as possible. The second point is crucial since many algorithms imply some smoothness assumptions. In the case of the small spatial resolution of the CURVACE sensor those assumptions are much weaker.

Global optic flow methods ([An89], [Pa06]) satisfy neither of these restrictions due to iterative propagation of local results to neighbouring pixels. Methods that rely not only on first and second order spatial derivatives ([Ur88]) also require a larger pixel neighbourhood for the computation of these derivatives. Many other methods provide high computational complexity, involving higher order matrix operations, image warping or Gabor filters (e.g. [Sr94], [He88]) and are also excluded from this survey. A more general introduction to optical flow methods can be found in the standard literature ([Te95]).

Detailed performance evaluations of optic flow algorithms are given by Galvin and Barron ([Ga98], [Ba92]) and more recently by the work of Baker et al. ([BS07]) using image sequences similar to those generated by classical camera systems. In contrast to these articles we focus first on a theoretical analysis of algorithms since a comparable evaluation of the presented methods requires an existing CURVACE sensor or a sensor simulation which are both under development. The final evaluation will also consider the acceptable tradeoff between accuracy and efficiency ([Li98]).

2.3.1 The Lucas-Kanade algorithm

Due to its efficiency and its straightforward implementation the standard Lucas-Kanade algorithm for optical flow is widely used ([LK81]). Using Taylor series expansion and omitting higher order terms equation (1) can be rewritten as:

$$\frac{dI}{dx}u+\frac{dI}{dy}v+\frac{dI}{dt}=0=I_x u+I_y v+I_t \tag{2}$$

By applying a smoothness constraint over a small neighbourhood W we get a set of linear equations to solve the otherwise underdetermined system. The new overdetermined system can be solved by least squares minimization yielding

$$\begin{bmatrix} u \\ v \end{bmatrix} = G^{-1} b \tag{3}$$

with

$$G = \sum_{(x,y)\in W} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \quad b = \sum_{(x,y)\in W} \begin{pmatrix} I_x I_t \\ I_y I_t \end{pmatrix} . \tag{4}$$

The imposed smoothness constraint entails a high sensitivity to depth discontinuities for this algorithm and it requires that the matrix G be invertible. If G is not invertible only a single gradient direction is present in the selected neighbourhood W which is known as the aperture problem.

With respect to computational complexity the Lucas-Kanade algorithm is well suited for application to the CURVACE sensor. The algorithm involves a single matrix inversion for each flow vector which only concerns a 2x2 matrix.

The algorithm requires a sufficiently large integration window to avoid the aperture problem. Since the spatial resolution of the CURVACE sensor is low, with a single pixel covering a field of view of about 4°, the integration window should be chosen very small. For an integration window of 5x5 pixels the algorithm would require that the smoothness assumption holds for a field of view of 12°. Thus it is necessary to find the best tradeoff for the integration window with respect to sensitivity to motion discontinuities and good image gradients.

The computed matrix G holds additional information about the gradients in the neighbourhood W given by its eigenvalues. This information can be used as a quality measure for the computed optical flow.

2.3.2 Minimal Normal Flow

The normal flow is the component of the optical flow which is normal to the local intensity contour line, i.e. aligned with the local image gradient. Normal flow (u_n, v_n) can be computed from the spatial and temporal gradients even if there is an aperture problem ([Te95] pp 81-82).

$$\begin{pmatrix} u_n \\ v_n \end{pmatrix} = -I_t \vec{n} = \frac{-I_t}{I_x^2 + I_y^2} \begin{pmatrix} I_x \\ I_y \end{pmatrix} \tag{5}$$

By this simplification the resulting equation is fully determined and the integration over a neighbourhood at the given position is not necessary and we can avoid some of the problems introduced by the Lucas-Kanade method. However it should be noticed that the computation of the spatial gradients still requires a neighbourhood around the given position.

For this reason at least three pixels are required to compute the normal flow, if spatial gradients are computed with a simple difference operator. From the pixel intensity (I(p0),I(p1),I(p2)) the image gradients, for pixels in a regular square pattern (Fig. 2 left), can be computed as

$$I_x = I(p_1) - I(p_0) \quad I_y = I(p_2) - I(p_0) \ . \tag{6}$$

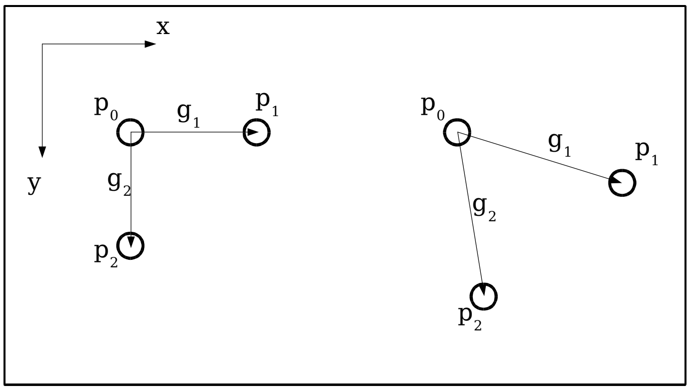


Fig. 2: regular pixel positions (left) and arbitrary pixel positions (right) in a global coordinate frame (x,y)

As was already described before, it is only necessary to adjust the computation of gradients for different pixel layouts. The gradient computation method proposed here can also be applied to other optic flow methods (e.g. Lucas-Kanade). If three pixels are positioned at arbitrary positions p_0, p_1, p_2 (Fig. 2, right) one can see that the gradients g_1 and g_2 can be calculated from the image gradient $\nabla I = (I_x, I_y)$ as

$$g_i = \vec{p_0 p_1} \cdot \nabla I \tag{7}$$

Since the gradients are given by $g_i = I(p_i) - I(p_0)$ the image gradient can be computed as

$$\begin{pmatrix} I_x \\ I_y \end{pmatrix} = \begin{pmatrix} (p_1 - p_0)^T \\ (p_2 - p_0)^T \end{pmatrix}^{-1} \begin{pmatrix} I(p_1) - I(p_0) \\ I(p_2) - I(p_0) \end{pmatrix} \ . \tag{8}$$

The advantage of the presented minimal approach is the small number of sensors needed and the low computational complexity. Thus it is very well suited for application to the CURVACE sensor as an elementary motion detector.

The drawback of this method is that it computes only normal flow and further processing might be necessary for specific application scenarios. Though motion estimation is still possible directly from normal flow ([Al94]).

2.3.3 Integrating normal flow over time

Our approach to compute 2D flow vectors from normal flow is to integrate the computed normal flow over time. We propose to use a Kalman Filter ([Ka60],[BW01]) to estimate the optic flow from the normal flow. Here we employ the high temporal resolution of the CURVACE sensor and use a temporal smoothness constraint on the optic flow. This means that we consider the optic flow vector as a constant signal and the normal flow in each time step as a measurement of this signal.

The time update and measurement equations for the Kalman filter are given by

$$\hat{x}_k = A\,x_{k-1} \quad , \quad A = \mathbf{1} \quad , \tag{9}$$

$$\hat{P}_k = A\,P_{k-1}\,A + Q = P_{k-1} + Q \quad , \tag{10}$$

$$z_k = n_k\,(n_k \cdot x_k) = H\,x_k \quad , \quad H = n_k\,n_k^t \quad , \tag{11}$$

where Q and R are the process noise and measurement noise covariance matrices, n_k is the normalized gradient and $x_k=(u_k,v_k)$ is the state vector at time-step k. The measurement z_k which is the normal flow, is calculated by projecting the optic flow on the gradient vector.

Since the optic flow, state x_k, is considered to be constant, the time update equations are simple with A being a unit matrix. The measurement update equations are then given by

$$K_k = P_k\,H^T\,(H\,P_k\,H^T + R)^{-1} \tag{12}$$

$$x_k = \hat{x}_k + K_k\,(z_k - H\,\hat{x}_k) \tag{13}$$

$$P_k = (\mathbf{1} - K_k\,H)\,\hat{P}_k \tag{14}$$

Again, these equations can be efficiently solved, since H is a special symmetric matrix and all matrices are only 2x2. For optimal performance the noise covariance matrices R and Q need to be tuned to the sensor speed.

The advantages of this method are that we only need a small spatial integration window for optic flow computation and can exploit the high temporal resolution of the CURVACE sensor. Additionally the Kalman Filter reduces the influence of noise on the resulting optic flow.

2.3.4 Correlation based optic flow

With another class of optical flow algorithms the image displacements can be inferred by finding the best matching correlation of image intensities ([Ma00]). Optical mouse chips typically detect motion by correlation, using regions of 17x17 or more pixels. Due to the small resolution of the CURVACE sensor only small pixel regions should be used for correlation since otherwise differing motion vectors might be present within one region.

Again we can take advantage of the high temporal resolution of the CURVACE sensor and use correlation over time to compute optic flow. Such a method has been proposed by [Ca95].

The method is based on computing the correlation over a region R around (x,y) with different spatial displacements d_x, d_y and temporal displacements d_t. The Match strength is given by

$$M(x,y,d_x,d_y,d_t)=\sum_{u,v\in R} C(I(u,v,t),I(u+d_x,v+d_y,t+d_t)) \tag{15}$$

where C is the correlation function. The fitting motion is selected by a winner-takes-all strategy, selecting the (d_x,d_y,d_t) with the best match strength. The resulting motion is then given by

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} \frac{1}{d_t} \tag{16}$$

The computational complexity for this method is linear w.r.t. the time window size. Since we expect optic flow vectors to be small (below one pixel) the spatial search window can be selected very small. Only two problems might arise, which is the memory consumption that increases linearly with the temporal search window, and the discretization of optic flow results through the winner-takes-all strategy.

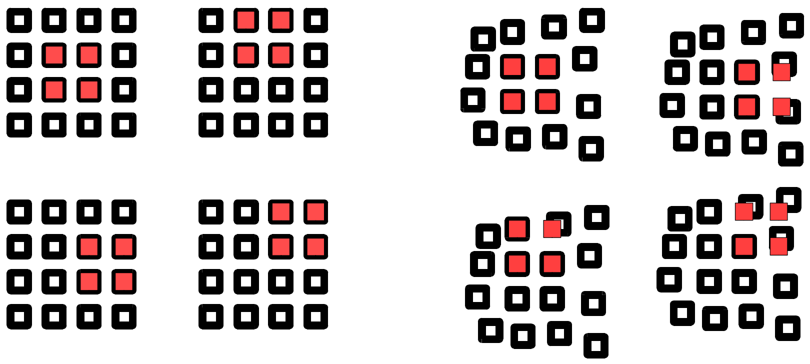


Fig. 3: possible positions of a search window for the best matching correlation. left: regular pixel grid, right: irregular pixel distribution where the search window does not match everywhere.

It is difficult to adapt such a correlation based method to a CURVACE sensor with non-regular pixel distributions since it is not necessarily possible to shift one pixel region in such a way that there is an overlap with another set of pixels. Thus it is necessary to interpolate the image brightness at the relevant positions which can introduce errors and increases the computation time.

2.3.5 Reichardt Detector

Inspired by the experiments of Werner Reichardt in the 1950's on the beetle Chlorophanus viridis, the Reichardt Detector is a biologically inspired model for motion extraction from compound eyes and still serves as basis for elementary motion detectors ([HR56]).

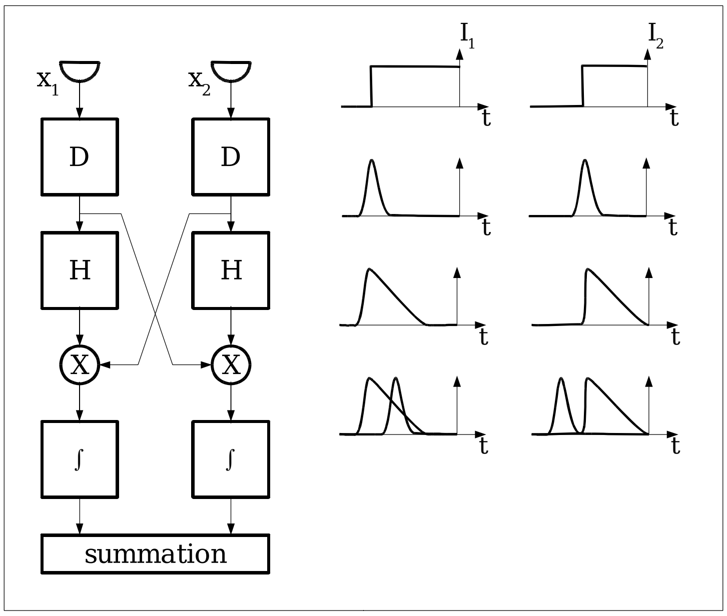


Fig. 4: Reichardt correlation detector (left) and its modifications to an input signal for a step edge moving from left to right (right) (image from [Ma00]).

D: differentiation, H: low pass filtering, X: correlation

The basic Reichardt detector consists of two point sensors. One of these sensors is connected to a delay element. The output of the filter and the second sensor are connect to a comparison unit which computes the correlation between both signals.

This detector has a preferred direction, detecting movements heading from left to right with maximal output if the speed of the movement corresponds to the delay in the respective delay element.

By adding a mirror-symmetrical unit of this half-detector we get a complete motion detector (Fig. 4). The model shown here is the most basic version of the Reichardt Detector. The hierarchical architecture allows for several refinements to be made, such as the addition of temporal filters, which essentially leave its operation unchanged. Modifications and models with a similar architecture, but different mathematical operations were proposed by [SS84], [WA83], and [AB85].

Several variations of the Reichardt motion detectors will be tested for the CURVACE sensor. Thus an optimal filter combination can be selected. Since the Reichardt Detector requires a small amount of computational effort and is based on motion models derived from insect compound eyes it is a promising choice for application to the CURVACE sensor.

2.3.6 Accurate optic flow for calibration

A large set of optical flow algorithms use variational methods to calculate optical flow, one of the first methods being the algorithm of [HS81]. A special class of this algorithms is based on total variation regularization with many different methods being recently published (see [BS07] for evaluation of recently published algorithms). Since this class of algorithms need many iterations with complex operations they are not suited for application on a microcontroller and even may not run at realtime on standard desktop PCs.

However they have proved to provide dense optical flow fields with high accuracy and therefore are best suited for the CURVACE sensor in the context of calibration. We will give a short introduction to one representative method of this kind ([WP08]).

The method is based on the minimization of the term

$$\int \left(l \cdot p(I_0(\boldsymbol{x}) - I_1(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x}))) + r(\boldsymbol{u}, \nabla \boldsymbol{u}) \right) d\,\boldsymbol{x} \tag{17}$$

where p(...) is the so called image data fidelity term, r(...) the regularization term and l weights between these terms. One can see that the first term is derived from equation (1) where u(x) is the motion field. The second term r(...) implies a smooth motion field (smoothness assumption). Selecting $p(x)=x^2$ and $r(\nabla u)=|\nabla u^2|$ results in the Horn-Schunck Model ([HS81]).
In the method from [BS07] both terms are defined differently by $p(x)=|x|$ and $r(\nabla u)=|\nabla u|$. Since both terms are not continuously differentiable a differentiable approximation is used. The Image I_1 is linearised near $x+u_0$ yielding the functional

$$\int l \left| \boldsymbol{u} I(\boldsymbol{x}+\boldsymbol{u}, t+1) + I(\boldsymbol{x}+\boldsymbol{u_0}, t+1) - \boldsymbol{u_0} I(\boldsymbol{x}+\boldsymbol{u}, t+1) - I(\boldsymbol{x}, t) \right| \tag{18}$$

The whole term is iteratively minimized by alternating between minimization with respect to the data term and minimization with respect to the regularization term yielding a smooth motion field.

2.3.7 Summary

We have presented a set of optic flow algorithms than can be used with the CURVACE sensor. Preliminary tests of the Lucas-Kanade method with the CURVACE simulator indicate that the quality of the computed optic flow is low if the integration window is chosen appropriately small. The small integration window together with the low spatial resolution of the sensor can lead to insufficient gradient information making the solution of the equation numerically unstable. Due to the small number of sensor elements it is not feasible to select only the good flow vectors.

Our approach of integrating normal flow over time with a Kalman filter represents a good tradeoff. Normal flow can be computed instantaneously as long as the local image gradient is larger than zero and integration of these measurements over time can exploit the temporal resolution of the CURVACE sensor.

The Reichardt Detector follows a similar scheme. Only two sensor elements are necessary to detect an elementary motion, but it is necessary to integrate the measurements from multiple detectors to compute an optic flow vector.

The correlation based method can only be used for CURVACE sensors with regular pixel positions and requires to store a whole sequence of sensor images. Table 1 gives an overview of the different methods. Local methods compute flow at one or more points in the image and can be adapted to performance requirements by selecting points of interest, while global methods compute optic flow over the whole image. Not every method yields a full two-dimensional optic flow vector and their output needs further processing. Realtime capability has not yet been asserted for an embedded chip thus this column is related to the performance of these algorithms on standard

PC's and can only be an estimate for embedded processors. The minimum number of sensor elements required to compute this kind of flow is an important factor, due to the low spatial resolution of the CURVACE sensors. This number includes pixels that are needed for gradient computation. This entry has no significance for the global TV-L1 method.

Optic flow method	realtime capable	full 2D flow	local/global	min. sens. elements
Lucas-Kanade	yes	yes	local	6
Normal Flow	yes	no	local	3
Kalman filtered normal flow	yes	yes	local	3
Correlation (Ted Camus)	yes	yes	local	9
TV-L1	no	yes	global	X
Reichardt Detector	yes	no	local	2

Table 1: Optic flow methods

In the next step these algorithms will be evaluated with a simulation of the CURVACE sensor with respect to accuracy, speed, and robustness. It should be noted that for the TV-L1 algorithm the main criteria will be accuracy and robustness since it will only be used for offline processing. For the other algorithms processing speed will be more important.

2.4 Further processing of optic flow

While optic flow in the image space can be computed independently from the underlying sensor geometry, further processing of this data might require to transform the computed optic flow into a geometrically meaningful representation. For the purpose of 3-dimensional motion estimation it is necessary to compute the spatial viewing directions that corresponds to each pixel.

A common representation of optic flow for this kind of problems is to inscribe the optic flow onto the surface of a unit sphere where viewing directions are represented as unit vectors and optic flow as a vector tangential to the sphere surface. This representation can be computed from the optic flow in the image space if the sensor geometry is known.

For the cylindrical CURVACE sensor the geometry is explicitly given by construction and the corresponding viewing directions can be calculated directly.

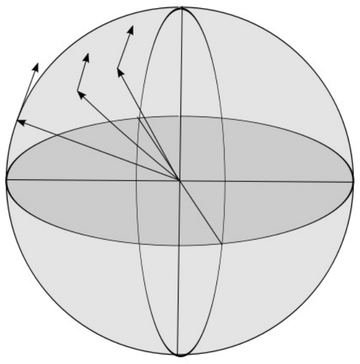


Fig. 5: spherical representation of flow vectors

In the case of other CURVACE sensors it will be necessary to recover the underlying geometry. Our first approach on calibration will be to perform predefined motions with the CURVACE sensor and compute the corresponding optic flow from the sensor output. From the optic flow predicted from the motion of the sensor and the optic flow computed from the sensor data we will be able to estimate the position of each ommatidium.

2.4.1 Egomotion and distance estimation

Some applications of the computed optic flow are the estimation of egomotion, relative distances to surrounding objects, and time to collision. Motion estimation, including the separation of egomotion from independent motion, is described in our previous work ([YS10],[YM10]). The presented methods can be applied for the CURVACE sensors by taking their specific geometry into concideration. Independent motion detection will however be less reliable due to the sensors low spatial resolution.

$$\dot{m} = -o \times m - \frac{(I - mm^T)v}{r(m)} \tag{19}$$

Equation 19 defines the derivation of the flow vector $\dot{m}$ from the rotation o and translation v for the viewing direction m, where r(m) is the distance of the point in the viewing direction m . In a general case with arbitrary motion the rotation and translation parameters and relative distance could be estimated by least squares minimization ([Ka93]). Estimating the relative distances allows to use CURVACE as proximity sensor.

In certain cases the full motion parameters don't need to be estimated from the visual input, drastically reducing the computational effort and visual ambiguities. Either additional sensors are available or only translational motion is present. The translational component of the extracted optical flow is inversely proportional to the distance to the object in relative motion, which is the base of the *motion parallax* effect ([Wh70]). The CURVACE sensor may provide the robot with the translational optic flow as input. Thus, the robot can either keep a safe distance to objects proportional to the forward speed ([Be09]) or control the speed with a close-loop system to keep constant values of the optic flow at predetermined locations of the sensor ([Ru05]). For a free flying aircraft it would be necessary to exclude the optic flow component due to rotations to estimate the proximity of obstacles, a process known as de-rotation of optic flow ([Ar04]). In CURVACE, this can be achieved with an extra processing step by predicting the optic flow generated by rotation as measured by the rate gyroscopes implemented in the sensor, and then subtracting the predicted optic flow from the measured optic flow.

In the case of visual motion it is assumed that the whole system has a single center of projection. If this does not hold for a specific CURVACE sensor, the spatial position and orientation of each single ommatidium will be recovered so that each of such ommatidium is treated as an independent camera system ([Ts97]).

3 CURVACE sensor simulation

Since the CURVACE sensor chip is not yet available at the beginning of the project, it is not possible to evaluate the performance of the presented algorithms with the real chip. Evaluation of the optic flow methods on a PC using normal image data sets is not feasible for a reliable prediction of the algorithm performance with respect to accuracy. The imaging characteristics of the CURVACE sensor differ greatly from normal cameras and the results from such an evaluation could not necessarily be transferred to the CURVACE sensor. For this reason we decided to develop a system which is able to simulate the output of the CURVACE sensor in a realistic environment. Not only can such a simulation be used for algorithm evaluation, it can also be used for testing free form CURVACE setups before assembly.

The final goal of this system development is an extensible simulation core system which provides the virtual environment and rendering for different CURVACE sensors. Extensions to the core system could e.g. consist of a flight simulation system that uses the control API to create a virtual flyer equipped with a virtual CURVACE sensor allowing to test CURVACE assisted flight control algorithms.

The simulation of an artificial compound eye is a new challenge that has not been dealt with to this extent. Methods for compound eye simulation usually focus on the image formation from the lens system but do not consider imaging of complete scenes ([Fa07]). Existing raytracing and rendering systems on the other hand can not handle the special geometry and lens system of the artificial compound eye. One approach by T. Neumann ([Ne02]) which is similar to the proposed method targets the simulation of insect eyes.

3.1 Sensor simulation with the plenoptic function

The simulation of the CURVACE sensor requires to determine the light intensity reaching each sensor. This can be done for each ommatidium by summing the amount of light coming from points within the field of view of the ommatidium and travelling through its center of projection.

The amount of light travelling through a point (x,y,z) in space in a given direction (θ, φ) is described by the plenoptic function ([AB91])

$$L(x, y, z, \theta, \varphi) \quad . \tag{20}$$

Thus we need to simulate the plenoptic function for a subspace of parameters. This set of parameters is defined by the positions (x_o, y_o, z_o) of each ommatidium and its corresponding field of view.

3.2 Partial simulation of the plenoptic function

The presented simulation system consists of two main processing units, the environment rendering unit and the sensor simulation unit.

The rendering unit loads a virtual scene and renders multiple views of the scene (Fig. 6). It allows to control the observer through mouse control, predefined motion paths

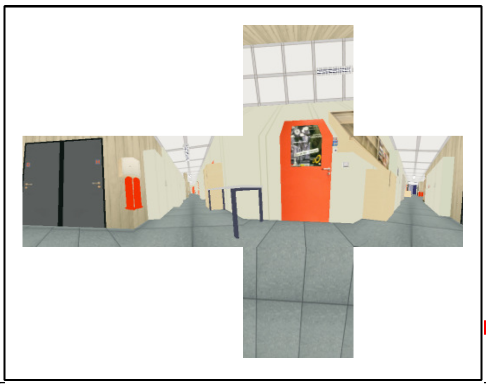


Fig. 6: unfolded environment map

or through another control extension. In this manner it is possible to simulate any motion sequence through the virtual scene.

The rendered views of the scene create a sampling of the plenoptic function in the virtual scene. Depending on the kind of CURVACE sensor that is simulated and the desired accuracy there are two methods of rendering the environment. If all ommatidia have the same center of projection (or their centers of projection lie in a sufficiently small area) only one view of the scene from a single viewpoint needs to be rendered by sampling the plenoptic function at a single position (x,y,z). Since a perspective projection can only cover less than 180° field of view and distortions tend to be large above 90° we render 6 views of the scene. This environment map allows to cover the whole scene without overlap between rendered views and creates a discrete sampling of the plenoptic function at one point in space.

For ommatidia with different centers of projection we need to sample the plenoptic function at multiple positions and consequently need to render a separate view of the scene for each ommatidium.

3.3 Computation of sensor output

The output of each CURVACE sensor element can be computed from the plenoptic function. If an ommatidium located at (x,y,z) has a field-of-view of S which is defined by two angle parameters, its output can be given by

$$\int_S L(x,y,z,\theta,\varphi)\, d\theta\, d\varphi \ . \tag{21}$$

Since the explicit reconstruction of the plenoptic function is not necessary we can directly recover the output L_o of an ommatidium from the rendered views. This is done by computing the influence (or weight) $w_o(P)$ of each rendered pixel P and computing the weighted average of the luminance of all pixels.

$$L_o = \frac{\sum_P w_o(P) * L(P)}{\sum_P w_o(P)} \tag{22}$$

Since only a few of all rendered pixels will have a weight w_o larger than zero and these weights will not change over time the performance of this algorithm can be increased by precalculating the weights and summing only over pixels that have an influence on the output.

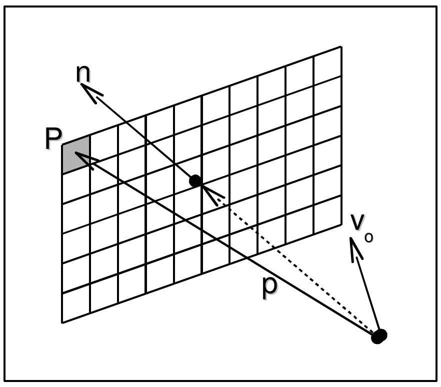


Fig. 7: calculation of pixel weight w(P)

For a simulated ommatidium with a given angular sensitivity function (ASF) the weighting function w_o can be computed from the viewing direction of the ommatidium v_o, the image plane normal to a given rendered view of the environment, and the direction vector p for each pixel P on the image plane using equation 23, as is shown in Fig. 7. The angular sensitivity depends on the angle between p and v_0.

$$w_o(P) = ASF(\measuredangle(p, v_o)) \cos(\measuredangle(n, p)) \tag{23}$$

The solid angle covered by a single pixel depends on the angle between the pixel direction p and the image plane normal n and is proportional to the cosine of this angle. The second term $\cos(\measuredangle(n, p))$ of the weighting function accounts for this difference.

The value L_o represents the amount of light incident on the photosensitive area of the respective ommatidium. In an imaging system with a linear response function this can be directly used as the *output* of the sensor element. The full simulation of the response function for the CURVACE sensors is more complex and this issue is tackled within the further developments of the simulation.

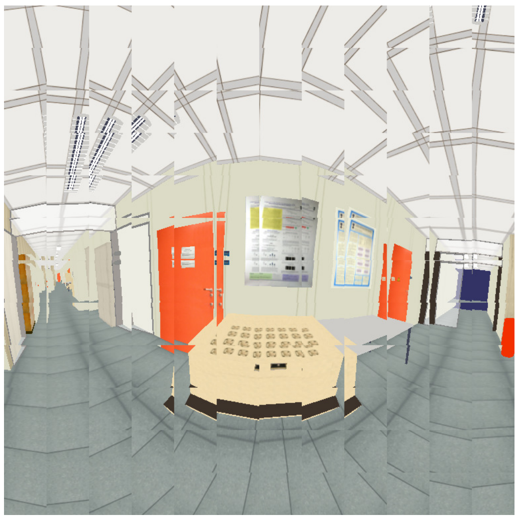
3.4 Simulation of arbitrary ommatidia

In the case of arbitrary ommatidia positions without a common centre of projection, the rendered views are created differently. For each ommatidium, a view of the environment is rendered in such a way that the corresponding image plane normal coincides with the viewing direction of the ommatidium and the opening angle of the ommatidium matches the viewfield of the scene view. Thus a much larger number of different views will be rendered while the resolution of each of these views may be significantly smaller than that of the cube environment map (Fig. 8).

The weighting function w_o then only depends on the angular sensitivity function of the ommatidium. If each ommatidium has the same ASF the weighting function needs to be computed only once.

While this kind of simulation is slower than rendering only 6 views for a cube environment it has the advantage that the direction and position of a single ommatidium can be changed during simulation without loss of performance. Changing these parameters in the first system would require recomputation of the weighting function reducing performance significantly.

Thus this simulation setup can be used to simulate dynamic CURVACE sensors or to test the effects of different sensor layouts at realtime.



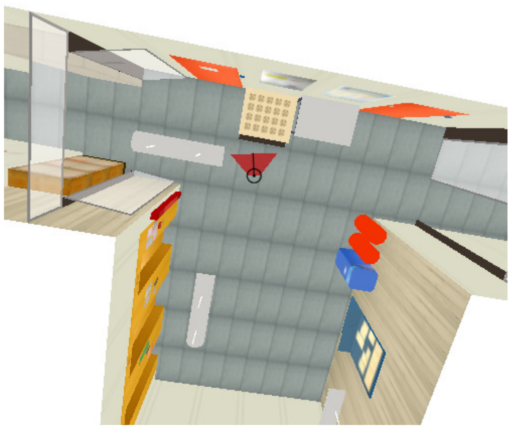


Fig. 8: example of 144 rendered views with one view per ommatidium (left), and birds eye view of the scene with sensor position (right)

3.5 Simulation system overview

A quick overview of the rendering system of the simulator is given in Fig. 9 below. The main input to the system is the CURVACE sensor layout, which contains the specifications of the simulated sensor like geometry, ommatidia type, and the angular sensitivity function. There are different options that can be used for simulation. Grey boxes indicate that we will add these options later in the simulator development. The CURVACE sensor layout is used to determine the required environment map together with the mapping from the environment map to the simulated sensor output.

To sample a single simulated sensor output, the system renders all views of the scene contained in the environment map. Then the sensor mapping is used to compute the simulated sensor output from the rendered scene views.

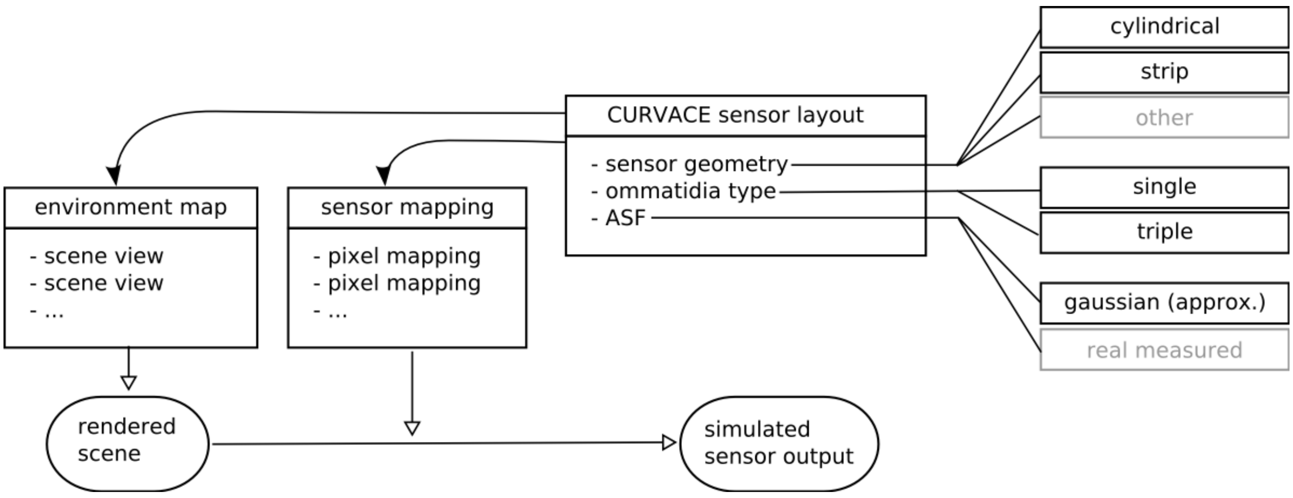


Fig. 9: Overview of the sensor simulation

3.6 System performance

For the rendering of virtual scenes we use the OpenSceneGraph library based on OpenGL. The current implementation runs on Windows but since the OpenSceneGraph library is available for Linux the core system can easily be ported.

Simulating a single frame of the cylindrical CURVACE sensor needs about 11ms while simulating the sensor with 144 views takes 120ms on a 2.2GHz Intel Core 2 Duo with NVIDIA GeForce 8600 M GS (Table 2).

	with cube map (max. 6 views)	**with 144 views**
scene rendering	**7ms**	**112ms**
output computation	**4ms**	**6m**
full time (frames/second)	**11ms (90FPS)**	**118ms (8.5FPS)**

Table 2: Performance of CURVACE simulation system

3.7 Future work

As a validation tool for optic flow algorithms the simulation will provide ground truth optic flow. In contrast to real sensor measurements, ground truth data can be directly obtained from the simulation since sensor motion and object distance are well known.

Further developments of the CURVACE simulator include different geometric sensor layouts with the possibility of changing the shape at runtime. At this time, the angular sensitivity function is modelled by a Gaussian function. While this is already a good approximation we strive to use a real measured sensitivity function for the sensor simulation to achieve more realistic sensor output. For the same reason we will implement appropriate noise models for the sensor.

Further validation of the presented optic flow algorithms will be performed using the simulated sensor output from different scenes with artificial and natural textures. By making the simulator a public tool we could allow other researchers without access to a CURVACE sensor to experiment with the sensor in simulation, making it possible to establish a community that can contribute to the public visual filter library we aim at.

4 References

[AB85] E.H. Adelson, J.R. Bergen, "Spatiotemporal energy models for the perception of motion." Opt Soc Am A 2:284-299, 1985

[AB91] E.H. Adelson, J.R. Bergen, "The plenoptic function and the elements of early vision", In 'Computation Models of Visual Processing', M. Landy and J.A. Movshon, eds., MIT Press, Cambridge, 1991, pp. 3–20.

[Al94] Y.Aloimonos, "Estimating the Heading direction using normal flow", International Journal of Computer Vision, 13:1, 33-56, 1994

[An89] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion", Int. J. Comp. Vision 2, pp. 283-310, 1989

[Ar04] Argyros, A.A., D.P. Tsakiris, and C. Groyer, *Biomimetic centering behavior [mobile robots with panoramic sensors].* Robotics & Automation Magazine, IEEE, 2004. **11**(4): p. 21-30, 68.

[Ba92] J. Barron, D. Fleet, S. Beauchemin, T. Burkitt, "Performance of optical Flow techniques", Proc. of the IEEE on CVPR, pp 236-242, 1992

[Be09] Beyeler, A., J.-C. Zufferey, and D. Floreano, *Vision-based control of near-obstacle flight.* Autonomous Robots, 2009. **27**(3): p. 201-219.

[BS07] S.Baker, D.Scharstein, J.P.Lewis, "A Database and Evaluation Methodology for Optical Flow", ICCV 2007

[BW01] G. Bishop, G. Welsh, "An introduction to the Kalman filter", SIGGRAPH, Course 8, 2001.

[Ca95] T. Camus, "Real-Time Quantized Optical Flow", Proceedings of the IEEE Computer Architectures for Machine Vision, 1995

[Fa07] H.R. Fallah, A. Karimzadeh, "Design and simulation of a high-resolution superposition compound eye", Journal of Modern Optics, 2007

[Ga98] B. Galvin, et al. "Recovering Motion Fields: An Evaluation of Eight Optical Flow algorithms", BMVC, pp 195-204, 1998

[Gi50] J.J. Gibson, "The Perception of the Visual World", Riverside Press, Cambridge 1950

[Gi66] J.J. Gibson, "The Senses Considered as Perceptual Systems", Houghton-Mifflin, Boston. MA, 1966.

[He88] D.J. Heeger, "Optical flow using spatiotemporal filters", Int. J. Comp. Vision 1, pp. 279-302, 1988

[HR56] B. Hassenstein, W. Reichardt, "Systemtheoretische Analyse der Zeit-, Reihenfolgen- und Vorzeichenauswertung bei der Bewegungsperzeption des Rüsselkäfers Chlorophanus." Naturforsch 11b:513-524, 1956

[HS81] B.K.P. Horn, B.G. Schunck, "Determining optical flow", Artificial Intelligence, vol 17, pp 185-203, 1981.

[Ka60] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems", Transactions of the ASME, Journal of Basic Engineering, pp. 35-45, 1960

[Ka93] K. Kanatani, "3D Interpretation of Optical Flow by Renormalization", IJCV, 11:3, pp. 267-282, 1993

[Li98] H. Liu, et al., "Accuracy vs Efficiency Trade-offs in Optical Flow Algorithms", Computer Vision and Image Understanding, 72:3, pp. 271-286, 1998

[LK81] B.D. Lucas, T. Kanade, "An iterative image registration technique with an application to stereo vision", IJCAI 1981.

[Ma00] H.A. Mallot, "Computational Vision: Information Processing and Visual Behavior": page 186, The MIT Press, Cambridge, Massachusetts 02142: page 186, 2000

[Ne02] T.R. Neumann, "Modeling Insect Compound Eyes: Space-Variant Spherical Vision", Proc. of 2nd Int. Workshop on Biologically Motivated Computer Vision, LNCS 2525, pp. 360-367, 2002

[Pa06] N. Papenberg, "Highly accurate optical flow computation with theoretically justified warping", Int. Journal of Computer Vision, pp 141-158, 2006

[Ru05] Ruffier, F. and N. Franceschini, *Optic flow regulation: the key to aircraft automatic guidance.* Robotics and Autonomous Systems, 2005. **50**(4): p. 177-194.

[Sr94] M. Srinivasan, "An image-interpolation technique for the computation of optic flow and egomotion", Biological Cybernetics, 71, pp. 401-416, 1994

[SS85] J.P.H. Santen, G. Sperling, "Elaborated Reichardt Detectors" J. Opt. Soc. Am. A/Vol. 2, No. 2:300-321, 1985

[Te95] A.M. Tekalp, "Digital Video Processing", Prentice Hall , 1995

[Ts97] AT. Tsao, et al., "Ego-Motion Estimation Using Optical Flow Fields Observed from Multiple Cameras", CVPR, pp.457, 1997

[Ur88] S. Uras, A. Verri, F. Girosi, and V. Torre, "A computational approach to motion perception", Biol, Cybern. 60, pp. 79-97, 1988

[WA83] A.B. Watson, A.J. Ahumada, "A look at motion in the frequency domain," NASA Tech. Memo. 84352, 1983

[Wh70] Whiteside, T.C.D. and G.D. Samuel, *Blur Zone.* Nature, 1970. **225**(5227): p. 94-95.

[WP08] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers, "An improved algorithm for TV-L1 optical flow computation", Proceedings of the Dagstuhl Visual Motion Analysis Workshop 2008.

[YM10] C. Yuan, H.A. Mallot, "Real-Time Detection of Moving Obstacles from Mobile Platforms", ICRA 2010

[YS10] C. Yuan, I. Schwab, F. Recktenwald, "Detection of Moving Objects by Statistical Motion Analysis", The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010