PARAMETER OPTIMIZATION FOR VISUAL OBSTACLE DETECTION USING A DERANDOMIZED EVOLUTION STRATEGY

THOMAS BERGENER, CARSTEN BRUCKHOFF, CHRISTIAN IGEL*

Abstract. The autonomous mobile robot ARNOLD uses information from a stereo camera system for navigating in an unknown and dynamically changing environment. A method called *Inverse Perspective Mapping* (IPM) is used for visual obstacle detection. The performance of this algorithm depends on the quality of the internal camera model. In this paper we employ an Evolutionary Algorithm (EA) to improve the parameters of this model. We use a derandomized evolution strategy called $(\mu/\mu_{\mathrm{I}}, \lambda)$-CMA, which adapts the complete covariance matrix of the mutation distribution. After descriptions of the IPM and the CMA, we show that the proposed optimization method leads to better parameter settings than adjustment by an expert.

Key words. inverse perspective mapping, evolutionary algorithm, optimization

1. Introduction. The control of an autonomous mobile robot in an unknown and dynamically changing environment using an active stereo vision system requires fast and reliable obstacle detection. For this task we employ a method called *Inverse Perspective Mapping* (IPM) [16, 3], which has been implemented on our autonomous mobile robot ARNOLD (see Fig. 1.1) developed in the framework of NEUROS (NEUral RObot Skills) [5]. The IPM provides information about obstacles in the environment of the robot. This information is used by dynamical systems for navigating ARNOLD under obstacle avoidance [11, 8]. The field of application of the IPM is not limited to robotics, e.g., the algorithm is also used for the detection of walking pedestrians in driver assistance systems [9, 10].

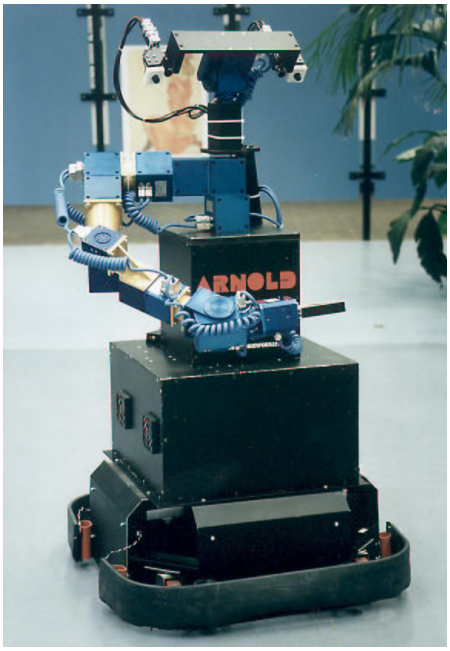


FIGURE 1.1. *Robot Arnold*

The idea of the IPM is to map the view of the one camera onto the view of the other. The mapping relies on a camera model and the corresponding internal and external parameters. Therefore the quality of the IPM depends on these hard to determine system parameters.

Evolutionary algorithms (EAs), i.e., the family of algorithms mimicking mechanisms of natural evolution, have proven to be powerful tools for solving multi-modal, non-linear and non-differentiable optimization tasks [1]. In this article we employ an elaborated *evolution strategy* (ES) [18, 19] for finding the parameters of our camera model with the required accuracy needed for the IPM. The algorithm, called covariance matrix adaption ES, implements the concept of derandomized self-adaptation of arbitrary normal mutation distributions [13, 12].

An error function that measures the quality of camera parameter settings is designed. We experimentally show that the

*Authors are in alphabetical order. Institut für Neuroinformatik, Ruhr-Universität Bochum 44780 Bochum, Germany, Tel: +49 234 32 {27969, 25570, 2558} Fax: +49 234 709 4209 {Thomas.Bergener, Carsten.Bruckhoff, Christian.Igel}@neuroinformatik.ruhr-uni-bochum.de

evolved results decrease the initial error to 7.5% on average. In comparison, even the expert who had implemented the IPM could only decrease the error to 50% by tuning the parameters manually.

In the next section, the IPM is introduced. In §3 we describe the employed EA in detail. The experimental results are shown in §4. The last section draws a conclusion.

2. Inverse Perspective Mapping.

2.1. Introduction. Visual obstacle detection for mobile robots can be achieved by observing the floor in front of the robot from two different camera perspectives and comparing the image contents. A solution of this correspondence problem by image correlation is suitable to construct a 3D representation of the scene and thus to detect obstacles. The drawback of this method is the high computational expense. Since a complete 3D representation is not required but only a segmentation into free and blocked areas of the floor plane, it is sufficient to test the hypothesis of a horizontal floor plane in front of the robot. A suitable mathematical camera model and complete knowledge of all internal and external parameters of the stereo camera system allows to reconstruct the texture of the floor from the camera image. In the same way this texture can be mapped back to the camera image by a sequence of image transformations. Therefore, it is possible to predict the contents of the left camera image from the right camera image, at least for the so called binocular range, i.e., the part of the floor that is visible in both camera images. A comparison of the predicted content of the left camera image with an acquired image by simply subtracting the grey-level values shows deviations at all points where the hypothesis of a free way is violated. Thresholding the difference image leads to a segmentation of the camera image into obstacles and free space. This technique of a purposeful change of perspective is called *Inverse Perspective Mapping* (IPM) [15, 3].

In practice it is crucial to find good values for the required camera parameters that are needed for the mapping. Especially when the system is stressed mechanically, e.g., by vibrations of the moving robot, a regular re-tuning of the parameters is necessary to achieve a small remaining error in the image mapping. There exist standard methods for estimating the internal and external camera parameters with reasonable effort [20]. Problems arise if these methods require a test pattern that is presented close to the cameras: Observing a real scene requires a different setting of aperture and focus than the test pattern. This slightly affects the internal camera parameters and leads to insufficient performance of the obstacle detection scheme. An experienced user is able to improve the mapping by observing the difference image and by fine tuning several parameters since he qualitatively knows the effect of the different parameters. This work is unsatisfactory and time-consuming. Standard optimization methods like estimated gradient descent or canonical simulated annealing turned out to be ineffective or inefficient. This makes an alternative automatic and systematic technique for finding good parameter settings very attractive.

2.2. Calculating the Mapping. The mappings between image coordinates and world coordinates contain a non-linear part, which models the radial lens distortion, and a linear part, which models the perspective image transformation assuming a pinhole camera.

The radial lens distortion of typical cameras can be modeled as a radial shift of the image content as it was projected on the camera target according to the pinhole camera model [20]. This shift from undistorted to distorted target coordinates is given by

$$\mathbf{x}_\mathrm{u} = \mathbf{x}_\mathrm{d}(1 + k_1 \left|\mathbf{x}_\mathrm{d}\right|^2 + k_2 \left|\mathbf{x}_\mathrm{d}\right|^4 \dots) \quad , \tag{2.1}$$

where $\mathbf{x}_u$ denotes the undistorted target coordinate and $\mathbf{x}_d$ is the corresponding distorted coordinate on the CCD-target. In practice it is sufficient to consider only the first term de-

pending on k_1 since this adequately models most off-the-shelf cameras. Moreover, the higher coefficients are hard to determine with good accuracy by the existing calibration mechanisms.

For this case the distorted image coordinates can be mapped to undistorted ones by

$$\mathbf{x}_{\mathrm{u}} = S_{\mathrm{du}}(\mathbf{x}_{\mathrm{d}}, k)\mathbf{x}_{\mathrm{d}} \tag{2.2}$$

with

$$S_{\mathrm{du}}(\mathbf{x}_{\mathrm{d}}, k) = k|\mathbf{x}_{\mathrm{d}}|^2 + 1 \tag{2.3}$$

and

$$|\mathbf{x}_{\mathrm{d}}| = \sqrt{x_{\mathrm{d},1}^2 + x_{\mathrm{d},2}^2} \ . \tag{2.4}$$

The inverse operation requires to solve the reduced form of a cubic polynomial of the form $x^3 + px + q = 0$. This can be done using Cardano's formula [21] and shall be given here by

$$\mathbf{x}_{\mathrm{d}} = S_{\mathrm{ud}}(\mathbf{x}_u, k)\mathbf{x}_u \ . \tag{2.5}$$

Please refer to [3] for details. Homogeneous coordinates and transformations are used to combine all linear transformations by a simple concatenation of matrix multiplications [2]. Subsequently a translation in 3D with $(x, y, z)^T$ is given by the 4×4-Matrix $\mathbf{T}(x, y, z)$, a rotation around the x-axis with angle α by $\mathbf{R}_x(\alpha)$. Scaling the cartesian dimensions with s_x, s_y and s_z is done by a multiplication with $\mathbf{S}(s_x, s_y, s_z)$. The perspective projection is defined as a mapping of cartesian points onto a plane that is perpendicular to the z-axis with distance f from the origin. This transformation is given by a multiplication with $\mathbf{P}(f)$.

In order to describe points in the world an arbitrarily chosen world coordinate system $(x_w, y_w, z_w)^T \in \mathbb{R}^3$ is defined. The 3D camera coordinate system $(x_c, y_c, z_c)^T \in \mathbb{R}^3$ is defined such that the z_c-axis and the optical axis are identical and the x_c-axis is collinear with the rows of the CCD (fig. 2.1). World coordinates are transformed into camera coordinates by applying a translation $\mathbf{T}(d_x, d_y, d_z)$ and three rotations $\mathbf{R}_x(-\alpha_x)$, $\mathbf{R}_y(-\alpha_y)$, $\mathbf{R}_z(-\alpha_z)$. Once the world points have been transformed into the camera coordinate system they can be projected onto the target using the perspective projection $\mathbf{P}(f)$, where f is the camera's focal length. Thus the mapping

$$\mathbf{M}_1 = \mathbf{T}(d_x, d_y, d_z)\mathbf{R}_z(-\alpha_z)\mathbf{R}_y(-\alpha_y)\mathbf{R}_x(-\alpha_x)\mathbf{P}(f) \tag{2.6}$$

transforms a world point $(x_w, y_w, z_w)^T$ to undistorted target coordinates $(x_{\mathrm{tu}}, y_{\mathrm{tu}})^T$ by

$$\mathbf{T}_h = (x_w, y_w, z_w, 1)\mathbf{M}_1 \tag{2.7}$$

$$(x_{\mathrm{tu}}, y_{\mathrm{tu}})^T = (T_{h,1}/T_{h,4}, T_{h,2}/T_{h,4})^T \ . \tag{2.8}$$

Applying the radial lens distortion using equation (2.5) to calculate the distorted target coordinates $(x_{\mathrm{td}}, y_{\mathrm{td}})^T$ gives

$$(x_{\mathrm{td}}, y_{\mathrm{td}})^T = S_{\mathrm{ud}}((x_{\mathrm{tu}}, y_{\mathrm{tu}})^T, k)(x_{\mathrm{tu}}, y_{\mathrm{tu}})^T. \tag{2.9}$$

The last part of the model is the acquisition process of the CCD and the frame-grabber to transform the target coordinates to pixel positions in the acquired image. The required parameters are the dimensions of the physical sensor elements d_{sx} and d_{sy}, the horizontal scaling factor s_x, and the optical image center $(c_x, c_y)^T$. Since in analog camera/frame grabber

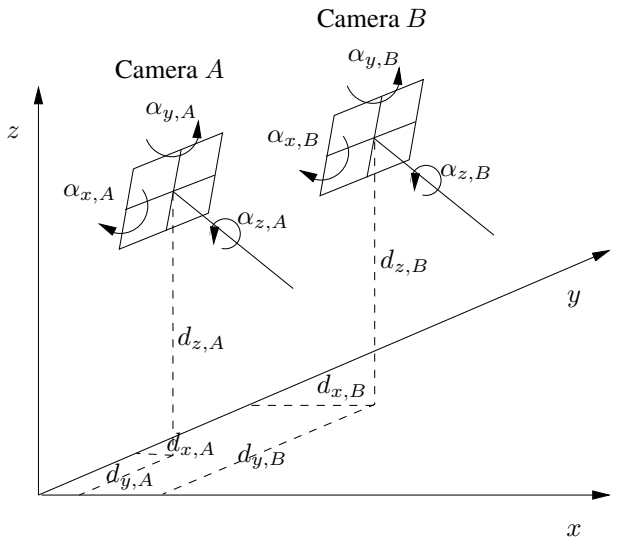


FIGURE 2.1. *World- and camera coordinate systems for two cameras A and B.*

setups the image content of the CCD's rows are converted into an analog signal and are resampled by the frame grabber the images may be scaled horizontally. Thus the horizontal scaling factor is the ratio of the camera's pixel clock frequency and the frame grabber's scan frequency. The optical image center denotes the pixel coordinate lying on the optical axis and which is usually not identical with the geometric image center. Thus the pixel vector corresponding to the world coordinate $(x_w, y_w, z_w)^T$ is

$$(I_x, I_y)^T = \begin{pmatrix} x_{\text{td}} s_x / d_{\text{sx}} + c_x \\ -y_{\text{td}} / d_{\text{sy}} + c_y \end{pmatrix}. \tag{2.10}$$

A second mapping transforms pixel vectors into world coordinates. The mapping described above does not require any assumption about the scene since it maps arbitrary world points to the image plane. Such a model of the world is required now as a target for the reverse mapping. Here the plane $z = 0$ of the world coordinate system is taken as the projection target.

Most of the following calculation steps invert the steps described above. First the pixel positions are mapped to distorted target coordinates:

$$(x_{\text{td}}, y_{\text{td}})^T = \begin{pmatrix} (I_x - c_x) d_{\text{sx}} / s_x \\ -(I_y - c_y) d_{\text{sy}} \end{pmatrix} \tag{2.11}$$

Using equation (2.2) the radial lens distortion with

$$(x_{\text{tu}}, y_{\text{tu}})^T = S_{\text{du}}((x_{\text{td}}, y_{\text{td}})^T, k)(x_{\text{td}}, y_{\text{td}})^T \tag{2.12}$$

is undone. The origin will later be the center of the perspective projection so the camera target must be shifted along the optical axis for the focal length f by $\mathbf{T}(0, 0, -f)$. The camera target is now rotated such that it is parallel with the plane $z = 0$ by applying the rotations $\mathbf{R}_x(\alpha_x)$, $\mathbf{R}_y(\alpha_y)$ and $\mathbf{R}_z(\alpha_z)$. The perspective projection $\mathbf{P}(-d_z)$ then maps the image onto the target plane $z = 0$. Finally, the translation $\mathbf{T}(-d_x, -d_y, -d_z)$ shifts the image to the world coordinate system. The resulting mapping from undistorted target coordinates to

world coordinates is

$$\mathbf{M}_2 = \mathbf{R}_x(\alpha_x)\mathbf{R}_y(\alpha_y)\mathbf{R}_z(\alpha_z)\mathbf{P}(-d_z)\mathbf{T}(-d_x, -d_y, -d_z) \tag{2.13}$$

with

$$\mathbf{W}_h = (x_{\text{tu}}, y_{\text{tu}}, 0, 1)\mathbf{M}_2 \tag{2.14}$$

and

$$(x_w, y_w, z_w)^T = (W_{h,1}/W_{h,4}, W_{h,2}/W_{h,4}, W_{h,3}/W_{h,4})^T . \tag{2.15}$$

To compare two images that were taken from different perspectives with reference to the floor plane, the mappings described above must be parameterized for the two cameras. This will be indicated by the additional indices A and B in the following formulas. For instance $(x_{\text{tu},A}, y_{\text{tu},A})^T$ is the undistorted target vector of the source camera and $(x_{\text{tu},B}, y_{\text{tu},B})^T$ is the undistorted target vector of the target camera. $\mathbf{M}_{2,A}$ is the matrix $\mathbf{M}_2$ parameterized for camera A and $\mathbf{M}_{1,B}$ is the matrix $\mathbf{M}_1$ parameterized for camera B.

Fig. 2.2 shows the steps to determine the corresponding pixel position $(I_{x,B}, I_{y,B})^T$ in the image of camera B for a given pixel position $(I_{x,A}, I_{y,A})^T$ in the image of camera A. Fig. 2.3 shows an example of an image acquired with camera A and an image the content of which was mapped from camera B to the perspective of camera A.

To optimize the parameters of the IPM it is crucial to avoid pairs of parameters that are redundant with respect to the error function chosen. For instance the quality of the mapping is independent of the camera's absolute position in the world. Instead of the six spatial coordinates of the cameras we only choose the reference vector from camera A to camera B and the height of camera A over the floor as candidate parameters for the optimization. Furthermore we exclude parameters that are known with good accuracy. The 16 parameters that are to be optimized are

- the pan, tilt and roll angles of both cameras ($\alpha_{x,A}$, $\alpha_{y,A}$, $\alpha_{z,A}$, $\alpha_{x,B}$, $\alpha_{y,B}$, $\alpha_{z,B}$),
- the image center of both cameras in pixel coordinates ($c_{x,A}$, $c_{y,A}$, $c_{x,B}$, $c_{y,B}$),
- the focal lengths of both cameras (f_A, f_B),
- the coefficients of radial distortion for both lenses (k_A k_B),
- and the height and base width of the camera system (d_z, $d_{y,A} - d_{y,B}$).

3. Evolution Strategy with Derandomized Self-Adaptation.

3.1. Evolutionary Algorithms. Evolutionary Algorithms (EAs) are a class of direct optimization methods inspired by natural evolution. Like artificial neural networks applied to technical tasks, EAs—and in particular the sophisticated *evolution strategy* used in this investigation—only make use of some basic biological concepts and combine them with classical mathematical and engineering approaches.

A typical EA starts with a parent population of individuals each representing a trial solution of the problem at hand. Each individual is assigned a fitness that is determined by the quality of the solution it represents. A so called offspring population of new individuals is generated by stochastically altering individuals from the parent population. Then the quality of each new solution is determined. A selection mechanism that prefers solutions with better fitness values chooses the individuals that constitute the next parent population. This (easily to parallelize) loop of creating new individuals from the parents, fitness evaluation, and selection is iterated until a termination criterion is fulfilled, e.g., a suitable solution is found or a certain amount of computation time has been consumed. Each pass through the loop is called a generation.

Repeat the following steps for every pixel position in image 1. Create a lookup table which holds the pixel correspondences that can be evaluated for obstacle detection.

1. Map the pixel coordinates of image A to distorted coordinates on the CCD target of camera A:

$$(x_{\text{td},A}, y_{\text{td},A})^T = \begin{pmatrix} (I_{x,A} - c_{x,A}) d_{\text{sx},A} / s_{x,A} \\ -(I_{y,A} - c_{y,A}) d_{\text{sy},A} \end{pmatrix} \tag{2.16}$$

2. Undistort the image on the CCD's target plane:

$$(x_{\text{tu},A}, y_{\text{tu},A})^T = S_{\text{du}}((x_{\text{td},A}, y_{\text{td},A})^T, k_A)(x_{\text{td},A}, y_{\text{td},A})^T \tag{2.17}$$

3. Map the undistorted target coordinates from the target of camera A to undistorted target coordinates on the CCD of camera B using homogeneous coordinates:

$$\mathbf{T}_h = (x_{\text{tu},A}, y_{\text{tu},A}, 0, 1)\mathbf{M}_{2,A}\mathbf{M}_{1,B} \tag{2.18}$$

4. Calculate the corresponding cartesian coordinates:

$$(x_{\text{tu},B}, y_{\text{tu},B})^T = (T_{h,1}/T_{h,4}, T_{h,2}/T_{h,4})^T \tag{2.19}$$

5. Map the undistorted coordinates to distorted ones on the target plane of camera 2:

$$(x_{\text{td},B}, y_{\text{td},B})^T = S_{\text{ud}}((x_{\text{tu},B}, y_{\text{tu},B})^T, k_B)(x_{\text{tu},B}, y_{\text{tu},B}). \tag{2.20}$$

6. Calculate the resulting pixel position in the image of camera 2:

$$(I_{x,B}, I_{y,B})^T = \begin{pmatrix} x_{\text{td},B} s_{x,B} / d_{\text{sx},B} + c_{x,B} \\ -y_{\text{td},B} / d_{\text{sy},B} + c_{y,B} \end{pmatrix}. \tag{2.21}$$

FIGURE 2.2. *Calculation of pixel correspondences with respect to the floor plane.*

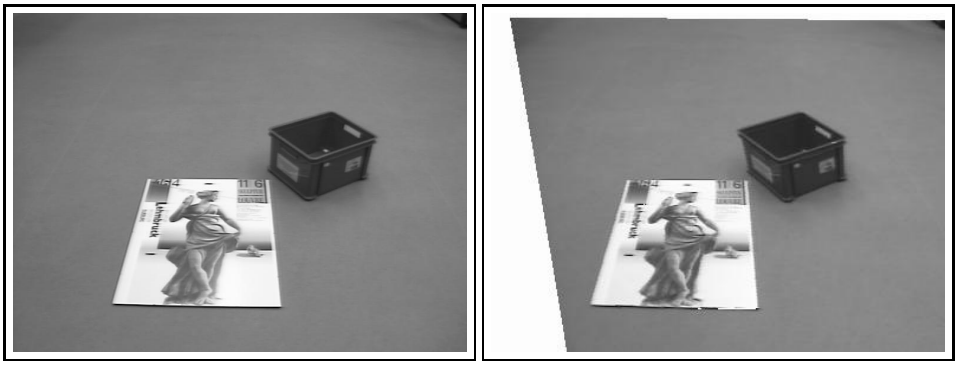


FIGURE 2.3. *Image taken from camera A and mapped image, the content of which was mapped from camera B to the perspective of camera A. The box was distorted by the mapping since its surface violates the assumption of a flat surface in front of the cameras.*

3.2. Derandomized Evolution Strategy. A variety of EAs for parameter optimization exists [1], in our application an evolution strategy (ES) is employed [18, 19]. Let μ and λ denote the parent and offspring population size, respectively. Each offspring l at generation g represents a real-valued vector $\boldsymbol{x}_l^{(g)}$, here the n parameters of the camera model. These so called object variables are altered by recombination and mutation. Intermediate recombination is used, i.e., the recombined individual is the center of mass of the parent population denoted by $\langle \boldsymbol{x} \rangle_\mu^{(g)}$, see [7] for an analysis of this recombination scheme. Mutation is realized by adding a normally distributed random vector with zero mean. Thus, for $l = 1, \ldots, \lambda$ we have

$$\boldsymbol{x}_l^{(g+1)} = \langle \boldsymbol{x} \rangle_\mu^{(g)} + \mathrm{N}_l \left(\mathbf{0}, \boldsymbol{C}^{(g)} \right) \quad . \tag{3.1}$$

The covariance matrix $\boldsymbol{C}$ of the mutation vectors is adapted to improve the search process, similar to the adaptation of mutation rates in biological evolution. The employed algorithm can produce arbitrary normal mutation distributions, so for n object variables $n(n+1)/2$ parameters are needed to describe the covariance matrix. These so called *strategy parameters*, which determine the mutation distribution, are updated online using a self-adaptation method called covariance matrix adaptation (CMA) [13, 12]. Apart from the initialization of the population and the initial strategy parameters, this algorithm is invariant towards linear transformations of the object parameter space, e.g., the scaling of the parameters. The CMA implements important concepts for strategy (and object) parameter adjustment [17, 13], e.g., the notion of *derandomization*. The change of the strategy parameters is based on the same realization of random variables as the mutations of object variables that have created fit offspring; the mutation distribution is changed in a way that the probability to reproduce steps in the search space that have led to the actual population is increased. Another important concept is *cumulation*. In order to use the information from previous generations more efficiently, not only single mutation steps are considered in the self-adaptation of the strategy parameters, but the whole search path the population has taken over a number of generations, the so called evolution path, is taken into account. This cumulation of history information is similar to momentum terms in gradient based optimization of artificial neural networks. Rank-based (μ, λ)-selection is used. This means, the new parent population consists of the μ best of the λ offspring.

These principles are not all inspired by biological evolution, but they allow an efficient use of the information gathered during the search process. Therefore, μ and λ can be chosen very small, whereas natural evolution reaches similar effects by comparatively huge populations.

3.3. Details of the algorithm.

3.3.1. The CMA-ES. The employed evolution strategy, termed a $(\mu/\mu_\mathrm{I}, \lambda)$-CMA-ES, is described in detail in [13, 12, 14]. There are different ways to specify the algorithm, here we use equations that are close to our implementation. This description should not be a substitute for reading the original work. The ES can be outlined as follows:

1. An initial parent population of μ equal individuals is randomly created.
2. From the parent population λ feasible offspring are created by applying intermediate recombination and afterwards mutation.
3. The offspring are evaluated.
4. The μ best of the λ offspring are selected to form the new parent population.
5. The strategy parameters are adapted.
6. If the termination criterion is not fulfilled go to step 2.

In the remainder of this section we explain steps 2 and 5 in more detail.

Each individual corresponds to an n-dimensional object variable vector. The offspring vectors at generation g are denoted $\boldsymbol{x}_l^{(g+1)}$, $l = 1, \ldots, \lambda$. Let $\boldsymbol{x}_{i:\lambda}^{(g)}$ be the ith best individual in the offspring population at generation g; thus under (μ, λ)-selection the individuals $\boldsymbol{x}_{i:\lambda}^{(g)}$, $i = 1, \ldots, \mu$, are the parent population at generation $g + 1$.

Intermediate recombination means calculating the center of mass of the parent population (see [7] for an analysis):

$$\langle \boldsymbol{x} \rangle_\mu^{(g)} := \frac{1}{\mu} \sum_{i=1}^{\mu} \boldsymbol{x}_{i:\lambda}^{(g)} \quad . \tag{3.2}$$

The offspring object variable vectors $\boldsymbol{x}_l^{(g+1)}, l = 1, \ldots, \lambda$, are generated according to

$$\boldsymbol{x}_l^{(g+1)} = \langle \boldsymbol{x} \rangle_\mu^{(g)} + \mathrm{N}_l \left(\boldsymbol{0}, \boldsymbol{C}^{(g)} \right) = \langle \boldsymbol{x} \rangle_\mu^{(g)} + \sigma^{(g)} \boldsymbol{B}^{(g)} \boldsymbol{D}^{(g)} \boldsymbol{z}_l^{(g+1)} \quad , \tag{3.3}$$

where

- $\sigma^{(g)}$ $\in \mathbb{R}^+$, global step size in generation g, initialized to $\sigma^{(0)} = 0.01$ in our experiments.
- $\boldsymbol{B}^{(g)}$ $n \times n$ matrix which reorients the axis parallel mutation distribution $\boldsymbol{D}^{(g)} \boldsymbol{z}_l^{(g+1)}$. Columns of $\boldsymbol{B}^{(g)}$ are normalized eigenvectors of the covariance matrix. $\boldsymbol{B}$ is orthogonal, i.e., $\boldsymbol{B}^{-1} = \boldsymbol{B}^T$.
- $\boldsymbol{D}^{(g)}$ $n \times n$ diagonal matrix where the diagonal elements are the square roots of eigenvalues of the matrix $\boldsymbol{C}'^{(g)} = \boldsymbol{C}^{(g)} / (\sigma^{(g)})^2$. The ith column of $\boldsymbol{B}^{(g)}$ is the corresponding eigenvector to the squared ith diagonal element.
- $\boldsymbol{z}_l^{(g+1)}$ $\in \mathbb{R}^n$, for $l = 1, \ldots, \lambda$ an independent realization of a normally distributed random vector with zero mean and unity matrix as covariance matrix.

This means the covariance matrix is decomposed into

$$\boldsymbol{C}^{(g)} = \overbrace{\left(\sigma^{(g)}\right)^2}^{\substack{\text{squared} \\ \text{global step size}}} \cdot \underbrace{\overbrace{\boldsymbol{B}^{(g)}}^{\substack{\text{normalized} \\ \text{rotation matrix}}} \overbrace{\boldsymbol{D}^{(g)}}^{\substack{\text{diagonal} \\ \text{step size matrix}}} \left(\boldsymbol{B}^{(g)} \boldsymbol{D}^{(g)}\right)^T}_{\boldsymbol{C}'^{(g)}} \tag{3.4}$$

The mutation (hyper)ellipsoids, i.e., the surfaces of equal probability density to place an offspring, of the random vector $\boldsymbol{D}^{(g)} \boldsymbol{z}_l^{(g+1)}$ are oriented along the coordinate axes and rotated by $\boldsymbol{B}^{(g)}$ yielding an arbitrarily oriented normally distributed random vector, see Fig. 3.1. The matrices $\boldsymbol{D}^{(g)}$ and $\boldsymbol{B}^{(g)}$ are determined by the matrix $\boldsymbol{C}'^{(g)}$. The factorization of the matrix $\boldsymbol{C}'^{(g)}$ into an orthogonal and a diagonal matrix is needed for the adaptation of the global step size, see Eqs. (3.7) and (3.8).

Then the strategy parameters are updated:

$$\boldsymbol{s}^{(g+1)} = (1 - c) \cdot \boldsymbol{s}^{(g)} + c_\mathrm{u} \cdot \underbrace{\sqrt{\mu}\, \boldsymbol{B}^{(g)} \boldsymbol{D}^{(g)} \langle \boldsymbol{z} \rangle_\mu^{(g+1)}}_{\frac{\sqrt{\mu}}{\sigma^{(g)}} \left(\langle \boldsymbol{x} \rangle_\mu^{(g+1)} - \langle \boldsymbol{x} \rangle_\mu^{(g)} \right)} \tag{3.5}$$

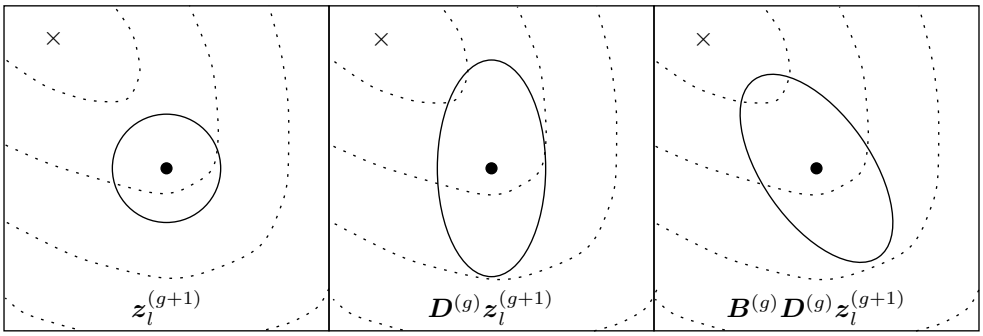


FIGURE 3.1. *The solid lines in the plots exemplarily show the mutation (hyper)ellipsoids, i.e., the surfaces of equal probability density to place an offspring, of the random vectors after the different transformations. Evolution strategies that adapt only one global step size can only produce mutation ellipsoids as shown in the left plot. Algorithms that adapt n different step sizes, one for each object variable, can produce mutation ellipsoids like the one in the center plot. Only if the complete covariance matrix is adapted, arbitrary normal distributions can be realized as shown in the right picture. The dashed lines schematically visualize an error surface, where each line represents points of equal fitness and the $\times$ symbol indicates the optimum. This example shows the benefits of an adapted covariance matrix that tends to produce offspring in the direction of the optimum with higher probability.*

$$\boldsymbol{C}'^{(g+1)} = (1 - c_{\text{cov}}) \cdot \boldsymbol{C}'^{(g)} + c_{\text{cov}} \cdot \boldsymbol{s}^{(g+1)} \left(\boldsymbol{s}^{(g+1)}\right)^{\text{T}} \quad , \tag{3.6}$$

where

$\boldsymbol{s}^{(g+1)}$ $\in \mathbb{R}^n$, cumulated evolution path, sum of weighted differences of population centers, initialized to $\boldsymbol{s}^{(0)} = \boldsymbol{0}$.

c $\in]0, 1]$, determines the cumulation time for $\boldsymbol{s}$ (roughly $1/c$). In our experiments, c is set to $c := 1/\sqrt{n}$

c_{u} $:= \sqrt{c(2-c)}$, normalizes the variance of $\boldsymbol{s}$, because $1^2 = (1-c)^2 + c_{\text{u}}^2$.

$\langle \boldsymbol{z} \rangle_\mu^{(g+1)}$ $:= \frac{1}{\mu} \sum_{i=1}^{\mu} \boldsymbol{z}_{i:\lambda}^{(g+1)}$, where $i : \lambda$ is the index of the ith best offspring $\boldsymbol{x}_{i:\lambda}^{(g+1)}$. $\langle \boldsymbol{z} \rangle_\mu^{(g+1)}$ is the average of the random variable realizations that lead to the new parent population.

c_{cov} $\in [0, 1[$, change rate of the matrix $\boldsymbol{C}'^{(g)}$. We use $c_{\text{cov}} = 2/(n^2 + n)$.

$\boldsymbol{C}'^{(g)}$ positive definite symmetrical $n \times n$ matrix that determines $\boldsymbol{B}^{(g)}$ and $\boldsymbol{D}^{(g)}$, initialized to the unity matrix.

In Eq. (3.6) the covariance matrix of the mutation distribution (apart from the global step size) is adapted in a way that the step $\boldsymbol{s}^{(g+1)}$ becomes more likely. The vector $\boldsymbol{s}^{(g+1)}$ represents not only the last evolutionary step of the (center of mass of the) population, but it additionally considers (accumulates) information about previous steps. This so called evolution path is updated in Eq. (3.5). The parameter c controls the influence of the previous steps compared to the actual step.

The adaptation of the additional global step size σ takes place on a shorter time scale. A second evolution path $\boldsymbol{s}_\sigma$ is calculated, where the scaling with $\boldsymbol{D}$ is omitted:

$$\boldsymbol{s}_\sigma^{(g+1)} = (1 - c_\sigma) \cdot \boldsymbol{s}_\sigma^{(g)} + c_{\text{u}_\sigma} \cdot \underbrace{\sqrt{\mu}\, \boldsymbol{B}^{(g)} \langle \boldsymbol{z} \rangle_\mu^{(g+1)}}_{\boldsymbol{B}^{(g)}\left(\boldsymbol{D}^{(g)}\right)^{-1}\left(\boldsymbol{B}^{(g)}\right)^{-1} \frac{\sqrt{\mu}}{\sigma^{(g)}}\left(\langle \boldsymbol{x} \rangle_\mu^{(g+1)} - \langle \boldsymbol{x} \rangle_\mu^{(g)}\right)} \tag{3.7}$$

$$\sigma^{(g+1)} = \sigma^{(g)} \cdot \exp\left(\frac{\|\boldsymbol{s}_\sigma^{(g+1)}\| - \hat{\boldsymbol{\chi}}_n}{d\hat{\boldsymbol{\chi}}_n}\right) , \tag{3.8}$$

where

$\hat{\boldsymbol{\chi}}_n$	expectation of the length of a standard normally distributed random vector of dimension n, can be approximated by $\hat{\boldsymbol{\chi}}_n :\approx \sqrt{n}\left(1 - \frac{1}{4n} + \frac{1}{21n^2}\right)$.
d	≥ 1, damping parameter for the change rate of the global step size. We set $d := 1/\sqrt{n}$.
$\boldsymbol{s}_\sigma^{(g+1)}$	$\in \mathbb{R}^n$, evolution path not scaled by $\boldsymbol{D}$, initialized to $\boldsymbol{s}_\sigma^{(0)} = \mathbf{0}$.
c_σ	$\in]0, 1]$, determines the cumulation time for $\boldsymbol{s}_\sigma$. We have used $c_\sigma := c$.
c_{u_σ}	$:= \sqrt{c_\sigma(2 - c_\sigma)}$, normalizes the variance of $\boldsymbol{s}_\sigma$.

The vector $\boldsymbol{s}_\sigma^{(g+1)}$ is a weighted sum of rotated standard normally distributed random vectors. The coefficient c_{u_σ} ensures that this sum also results in a standard normally distributed, rotated random vector, cf. [12]. The expectation of the length of such a random vector is given by $\hat{\boldsymbol{\chi}}_n$. This means, that the step size $\sigma^{(g+1)}$ in Eq. (3.8) is increased if the cumulated evolution path $\boldsymbol{s}_\sigma^{(g+1)}$ is larger than expected, if $\boldsymbol{s}_\sigma^{(g+1)}$ is shorter than expected, $\sigma^{(g+1)}$ is decreased.

3.3.2. Constraint handling. Constraint handling is not a problem in our application of the algorithm. There exist sensible upper and lower bounds for the region M_{init} where we can expect good camera parameters, as well as boundaries for valid camera settings that define the search space M_{feasible}. The center of mass of the population is initialized within $M_{\text{init}} \subset M_{\text{feasible}} \subset \mathbb{R}^n$. If an offspring is created that is not regarded as feasible, it is discarded and a new one is generated. This does not happen very often (only a few times in the early explorative stage of the ES), so a sophisticated constraint handling technique is not necessary.

4. Experiments.

4.1. Experimental setup. The experiments are based on the camera setup of the autonomous service robot ARNOLD [4]. The sensor head is shown in detail on Fig. 4.1.

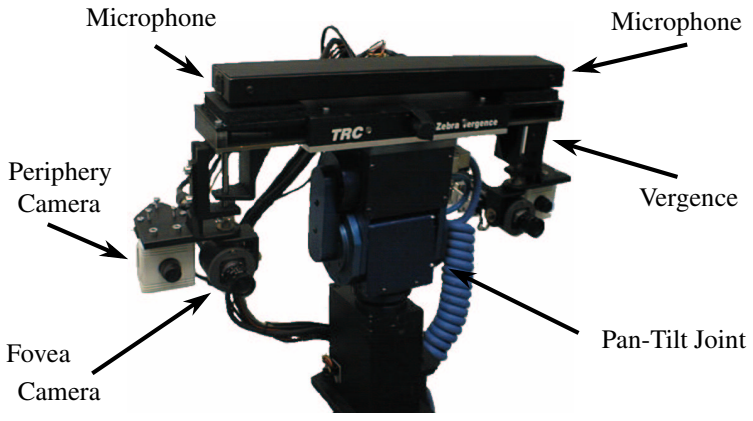
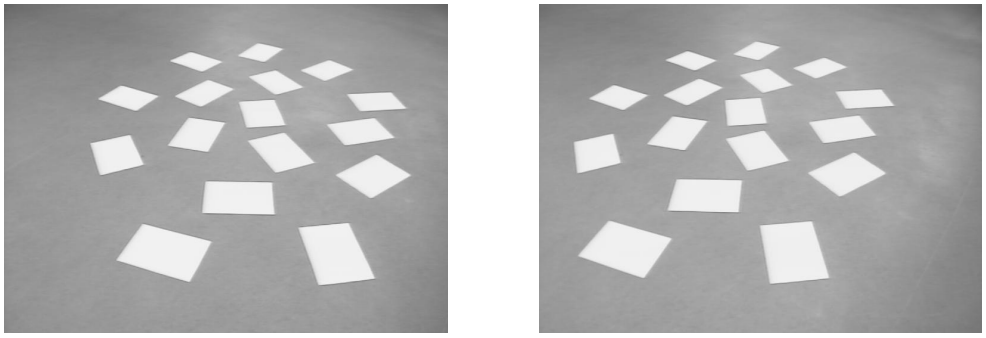


FIGURE 4.1. *Sensor head of the autonomous robot* ARNOLD

The internal camera parameters were determined using an implementation of the calibration algorithm described in [20]. Knowing the internal parameters we could mechanically adjust the cameras on the head with simple calibration patterns making the nodal lines and image rows parallel for a vergence angle of zero. This gives a good estimation of the three external camera parameters pan, tilt and yaw. Since we map the image of one camera to the perspective of the other, we have to optimize the parameters of both cameras. We have chosen the 7 most important parameters for optimization, which are the radial lense distortion, the focal length, the image center and the three external rotation parameters of the cameras, pan, tilt and yaw. Furthermore, we optimize two more parameters, the distance between both cameras (base length) and the height of the cameras. Altogether we optimize 16 parameters. The initial values of the parameter set are calculated by the calibration scheme described above.



(a) Image of the left camera

(b) Image of the right camera

FIGURE 4.2. *Structured horizontal plane to generate an error measure*

To generate an error measure needed by the EA, we put sheets of paper on the floor to get a structured horizontal plane $z = 0$ in the world coordinate system (see Fig. 4.2). After calculating the mapping, the number of pixels exceeding the threshold should be close to zero, since the assumption of an obstacle free plane is fulfilled. We compute the number of pixels exceeding the threshold for six different pairs of stereo images. The average of these values serves as the error measure which equals the fitness, where lower fitness values are considered to be better than higher fitness values, i.e., we perform a minimization task. We would like to stress that the error cannot vanish, because the pinhole-camera model, although enhanced by the radial lense distortion [20], can only approximate real cameras up to a certain extend. Furthermore not all parameters of the system are optimized.

We performed 10 independent trials of the evolution strategy described in the previous section. We used a parent population size of $\mu = 2$ and the number of offsprings in each generation was set to $\lambda = 10$. The evolutionary process was stopped after 1000 generations.

4.2. Results. After calibration, the initial error was 15282.5 averaged over the six test cases. This parameter setting can be tuned manually by an expert. By this time consuming procedure the initial parameters can be improved by roughly 50 %.

The error trajectories of the evolutionary optimization are depicted in Fig. 4.3. The final error of the best individual of the best trial was 373.3, i.e., less than 2.5 % of the initial value. On average, the evolutionary adaptation of the parameters reduced the error to 1158.72 in the final generation, this is 7.5 % of the initial error. Even the worst optimization process

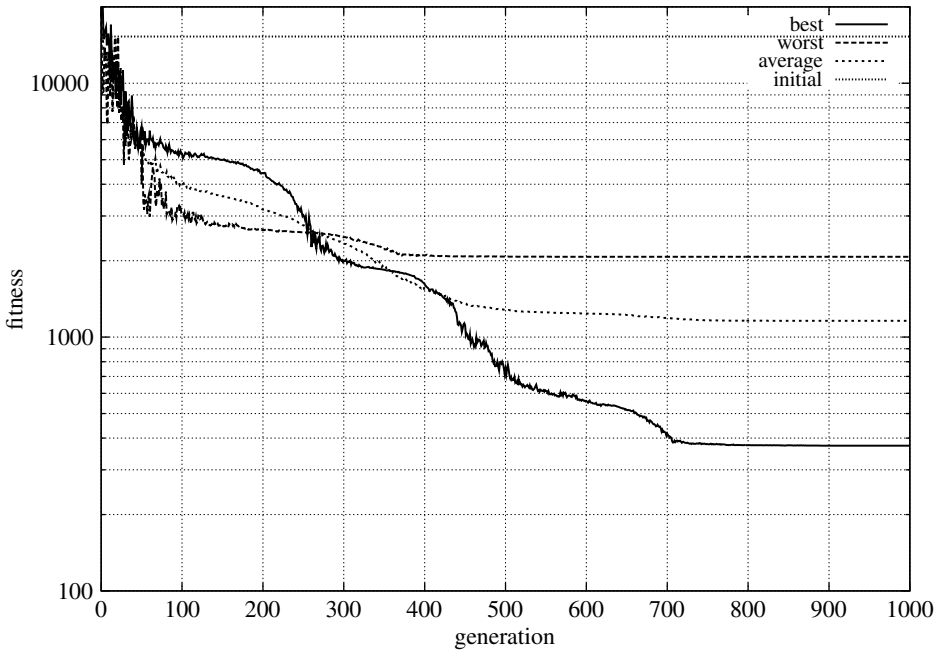
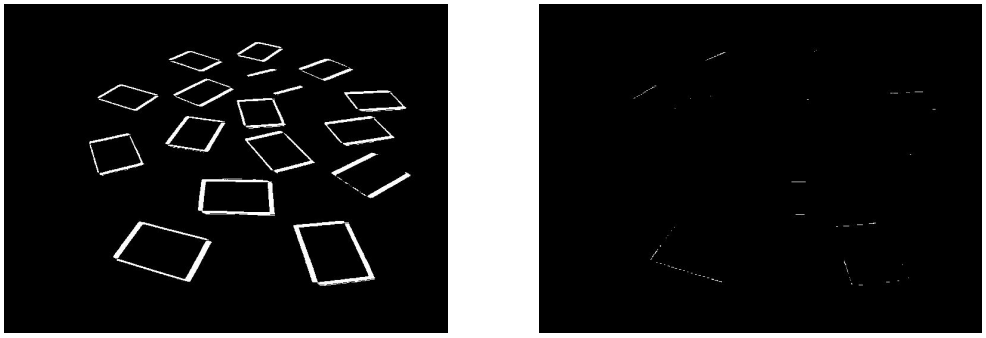


FIGURE 4.3. *For each generation, this plot shows the fitness values of the best individual for the best and worst out of 10 evolutionary optimization trials, as well as the average over the best individuals of the 10 trials. As a reference, the fitness, i.e., the average error measured in pixels per test picture, is plotted. Note the logarithmic scaling of the ordinate.*

reached a fitness of 2071.67, still a reduction to 13.6 % of the initial error. This indicates the robustness of the proposed optimization method.

The experiments show that it is not necessary to wait 1000 generations to get a satisfactory solution. All trials had almost completely converged after 800 iterations, and 500 generations appear to be sufficient for evolving good solutions, see Fig. 4.3.



(a) Initial mapping error

(b) Mapping error after optimization

FIGURE 4.4. *Result of the evolutionary optimization of the parameters. The white pixels indicate mismatches. Fig. (a) show the initial error and Fig. (b) the error after optimization.*

Figure 4.4 demonstrates the results of the parameter optimization. The left image shows the results of IPM with the initial parameter set applied to the test images shown in Fig. 4.2. The right image shows the error after optimization, which is decreased to about 2.5 % of the initial value.

The obtained results are considerably better than the ones presented by us before [6]. This has two reasons. First, we optimize more parameters of the camera model, so a higher accuracy can be achieved. Second, we use a different evolutionary algorithm improving both convergence speed and quality of the obtained solutions. This was verified be employing the CMA-ES to the same optimization task as described in [6]. Although the dimension of the problem has increased, the improved ES converges faster, i.e., needs less fitness evaluations to achieve a desirable result, than the algorithm employed in [6]. Please note that the CMA-ES needs more generations but has a much smaller population size, and that the results cannot be compared quantitatively because different test pictures were used.

5. Conclusion. Evolutionary optimization is suitable for optimizing the parameters of camera models used for visual obstacle detection. It can be successfully employed for automating the setting of the parameters for the IPM, which is used to control the mobile robot ARNOLD in an unknown and dynamically changing environment [11, 8]. After optimization the obstacle detection leads to a better performance, enabling the robot to move faster and more reliably.

The results shown here were on average by far superior to the settings suggested by an expert. The best solution found produces only 2.5 % of the errors in relation to the initial calibration. Compared to previous work [6], the proposed algorithm shows better performance both in terms of convergence speed and accuracy of the obtained solutions. This is due to the increased number of parameters optimized and the improved optimization algorithm. The CMA-ES [14] implementing completely derandomized self-adaptation of arbitrary normal mutation distributions has proven to be a powerful tool for solving real-world problems and should be regarded as the first choice of evolutionary computation methods for real-valued parameter optimization.

Acknowledgments. We would like to thank Martin Kreutz for his work on the EALib and Nikolaus Hansen and Andreas Ostermeier for helpful discussions.

REFERENCES

[1] T. BÄCK AND H.-P. SCHWEFEL, *An overview of evolutionary algorithms for parameter optimization*, Evolutionary Computation, 1 (1993), pp. 1–23.
[2] D. BALLARD AND J. BROWN, *Computer Vision*, Prentice Hall, 1985.
[3] T. BERGENER AND C. BRUCKHOFF, *Compensation of non-linear distortions in inverse-perspective mappings*, Tech. Report IRINI99-04, Institut für Neuroinformatik, Lehrstuhl für Theoretische Biologie, Ruhr-Universität Bochum, 1999.
[4] T. BERGENER, C. BRUCKHOFF, P. DAHM, H. JANSSEN, F. JOUBLIN, AND R. MENZNER, *Arnold: An anthropomorphic autonomous robot for human environments*, in Selbstorganisation von adaptivem Verhalten (SOAVE'97), 1997, pp. 25–34.
[5] T. BERGENER, C. BRUCKHOFF, P. DAHM, H. JANSSEN, F. JOUBLIN, R. MENZNER, A. STEINHAGE, AND W. VON SEELEN, *Complex behavior by means of dynamical systems for an anthropomorphic robot*, Neural Networks, 12 (1999), pp. 1087–1099.
[6] T. BERGENER, C. BRUCKHOFF, AND C. IGEL, *Evolutionary parameter optimization for visual obstacle detection*, in Advanced Concepts for Intelligent Vision Systems (ACIVS'99), J. Blanc-Talon and D. Popescu, eds., The International Institute for Advanced Studies in Systems Research and Cybernetics, 2000, pp. 104–109.
[7] H.-G. BEYER, *Toward a Theory of Evolution Strategies: On the Benefit of Sex – the* $(\mu/\mu, \lambda)$*-Theory*, Evolutionary Computation, 3 (1995), pp. 81–111.

[8] C. BRUCKHOFF AND P. DAHM, *Neural fields for local path planning*, in Proceedings of the International Conference on Intelligent RObotic Systems (IROS 98), 1998, pp. 1431–1436.

[9] C. CURIO, J. EDELBRUNNER, T. KALINKE, C. TZOMAKAS, C. BRUCKHOFF, T. BERGENER, AND W. VON SEELEN, *Walking pedestrian detection and classification*, in Proceedings of the 20. DAGM-Symposium Mustererkennung, Springer, 1999, pp. 78–85.

[10] C. CURIO, J. EDELBRUNNER, T. KALINKE, C. TZOMAKAS, AND W. VON SEELEN, *Walking Pedestrian Recognition*, IEEE Transactions on Intelligent Transportation Systems, 1 (2000), pp. 155–163.

[11] P. DAHM AND C. BRUCKHOFF, *Autonomous decision making in local navigation*, in From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (SAB 98), MIT Press, 1998, pp. 229–233.

[12] N. HANSEN, *Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie: Eine Untersuchung zur entstochastisierten, koordinatenunabhängigen Adaptation der Mutationsverteilung*, Mensch & Buch Verlag, Berlin, 1998.

[13] N. HANSEN AND A. OSTERMEIER, *Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The* $(\mu/\mu, \lambda)$*-CMA-ES*, in EUFIT'97, 5th Europ. Congr. on Intelligent Techniques and Soft Computing, Verlag Mainz, Wissenschaftsverlag, 1997, pp. 650–654.

[14] ———, *Completely derandomized self-adaptation in evolution strategies*, Evolutionary Computation, Special Issue on Self-Adaptation, (2000). to appear.

[15] J. LITTLE, S. BOHRER, H. MALLOT, AND H. BÜLTHOFF, *Inverse Perspective Mapping Simplifies Optical Flow Computation and Obstacle Detection*, Biological Cybernetics, 64 (1991), pp. 177–185.

[16] H. A. MALLOT, H. H. BÜLTHOFF, J. J. LITTLE, AND S. BOHRER, *Inverse perspective mapping simplifies optical flow computation and obstacle detection*, Biological Cybernetics, 64 (1991), pp. 177–185.

[17] A. OSTERMEIER, A. GAWELCZYK, AND N. HANSEN, *A derandomized approach to self-adaptation of evolution strategies*, Evolutionary Computation, 2 (1995), pp. 369–380.

[18] I. RECHENBERG, *Evolutionsstrategie '94*, Werkstatt Bionik und Evolutionstechnik, Frommann-Holzboog, Stuttgart, 1994.

[19] H.-P. SCHWEFEL, *Evolution and Optimum Seeking*, John Wiley & Sons, New York, 1995.

[20] R. TSAI, *A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses*, IEEE Journal Robotics and Automation, (1987), pp. 323–344.

[21] E. W. WEISSTEIN, *CRC Concise Encyclopedia of Mathematics*, CRC Press, 1998.