A Benchmark Data Set for Data Association

Jörg Kurlbaum, Udo Frese

SFB/TR 8 Report No. 017-02/2009
Report Series of the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition
Universität Bremen / Universität Freiburg

Contact Address:

Dr. Thomas Barkowsky
SFB/TR 8
Universität Bremen
P.O.Box 330 440
28334 Bremen, Germany

Tel +49-421-218-64233
Fax +49-421-218-64239
barkowsky@sfbtr8.uni-bremen.de
www.sfbtr8.spatial-cognition.de

A Benchmark Data Set for Data Association

Jörg Kurlbaum Udo Frese

February 2, 2009

Abstract

In this technical report we present a data set which can be used by researchers to test and compare their algorithms related to feature-based SLAM and in particular data association tasks. The data set covers a large path mostly through an indoor office environment with artificial and natural landmarks. It provides odometry and geometrical results from computer vision-based landmark detection, as well as raw images and camera calibration data. The artificial landmarks are white discs on the floor where the position relative to the robot is provided as measurement. The natural landmarks are vertical lines where only the bearing relative to the robot is provided and data association is remarkably difficult.

This technical report provides full information about experimental setup, model and measurement functions. The focus of this data set is on data association and for comparison of different approaches we provide full ground truth by annotating all features.

1 Introduction

In the research field of mapping and localization working online with real data is on the one hand essential and on the other hand laborious. So most researchers develop their algorithms on a few recorded data sets. A lot of data sets are available on the internet, for instance on the Radish repository [4]. Remarkably, however, for Simultaneous Localization and Mapping (SLAM) most data sets provide raw laser scan data only, while many researchers work on feature based algorithms. An exception is the well known Victoria Park data set [3] [7], where trees in a park are natural features and tree detection code is provided with the data set. As a consequence, Victoria Park is more or less the only data set used in comparing feature based SLAM algorithms.

The data set we present here evolved also from real experiments in feature based SLAM, which in the first place have been recorded for a specific use. But we thought it interesting to polish it, make it easy to use and document it for a broader community.

In particular we wanted to provide fully preprocessed data, first, to relieve you, as a user, from implementing feature detection and second, to make the core SLAM or data association algorithm more comparable. Evaluating different algorithms on one well prepared data set can eliminate effects that an algorithm may be specially *tuned* to one kind of data set and therefore it's performance is not comparable.

Our proposed data set is based on two different sparse features detected by a vision system. The data set is already preprocessed. You can directly use the features as measurements for your algorithm, without implementing your own vision or feature extraction algorithms. If you want, you can still implement own feature detectors, all necessary information is given in this documentation.

As features you can choose between artificial landmarks, namely circles on the ground, and the natural features, arbitrary vertical lines. While the circles were already used in experiments and can be considered as a solvable problem and more as an additional set to the Victoria park sequence, we believe the vertical lines are a major challenge for data association. They are typical, natural and important features in an office environment. They are relatively easy to detect and there are many of them. This, on the other hand is the major challenge of vertical lines that there are so many, it is hard to say which one is which. We combined near adjacent lines, but they still may be ambiguous, for example a line far away may split up in two or more lines, when the robots approaches a certain corner. The bearing-only information, the number of lines, and the phenomenon of splitting lines make the data association difficult.

All information about measurement and odometry is represented in geometric quantities, ready as input for your algorithms. Our data set focuses on the data association problem therefore we added ground truth values for data association to be used to evaluate your data association algorithms relative to others.

The data set also meets other requirements for an interesting data set: The path the robot travels is long. The field of view is small conditioned by the office building. There are some small and one big loop that make data association particularly challenging.

1.1 Related Work and Data Sets

The idea of having publicly available data sets for research is, of course, not new. We will present just a few sets and sources and will then try to analyze the differences to our set.

1.1.1 Victoria Park

One of the most popular data sets for simultaneous localization and mapping is the Victoria Park Sequence recorded by Eduardo Nebot et al. at the Australian Centre for Field Robotics and introduced in [3]. This data set describes a path through a park at the University of Sydney. It covers a region from about 350 meters in square and uses mostly trees as landmarks. The data is collected from laser range finders and velocity sensors attached to a car driven by a person. In addition, GPS data is recorded and can serve as ground truth for the path.

The data set contains sensor readings from steering and rear-axis wheel (odometry) and laser range finder (one 360 degrees scan per second) along with the ground truth position data from GPS. All inputs are fused in a Sensor Manager which simply sorts the sensor input in their arriving chronology.

The data set is preprocessed for Matlab in the way that all sensor readings are already in Matlab-compatible formats and you can directly access each sensor reading for a certain time step. For the laser range data a tree detector provides features with bearing and distance information.

If you cannot or don't want to use Matlab, then you have to build a feature extractor (usually trees) for the laser range data yourself.

From the data association point of view the Victoria Park sequence is relatively easy. The trees that are used as feature landmark usually have a large distance to each other and can be separated or uniquely identified with commonly used algorithms or tests, like the Mahalanobis distance.

The travelled path of the car doesn't have any big loops and in the small loops you can usually see all the landmarks in that region because the range finder has a large coverage (maybe 30m). For SLAM and especially for the data association problem loops are an important factor. Closing big loops usually increases the requirements to the underlying algorithms.

The Victoria park sequence contains also ground truth data obtained from a GPS sensor. From the nature of GPS the path has blackouts as you can see in the Figure 1.

Pros and cons The Victoria park sequence is a well documented and already widely used data set. It's used for feature based SLAM methods even when it actually a laser scan based data set. SLAM on this set is already solved including the data association problem (see e.g.[9]). It lacks bigger loops and bigger difficulties in general.

By comparision, our proposed data set is based on genuine feature landmarks. It focuses more on the data association problem, which is particular difficult if you use the line features.

1.1.2 Radish

Beside the very popular Victoria park data set many researcher provide some data from their typical testing environment. Many such data sets are published at the Radish Repository. The Radish Project (Robotics Data Set Repository) [4] started to collect useable data for the robotics community in general but until now it's mostly focused on localization and mapping.

You will find mostly laser scans from different buildings, often even only the generated map. At time of this writing (mid 2008) Radish contains about 40 entries with about two third laser range based, a few maps and also some simulated data. One data set is based on wireless strength measurement. A data set from University Alberta is one of the few that is feature based.

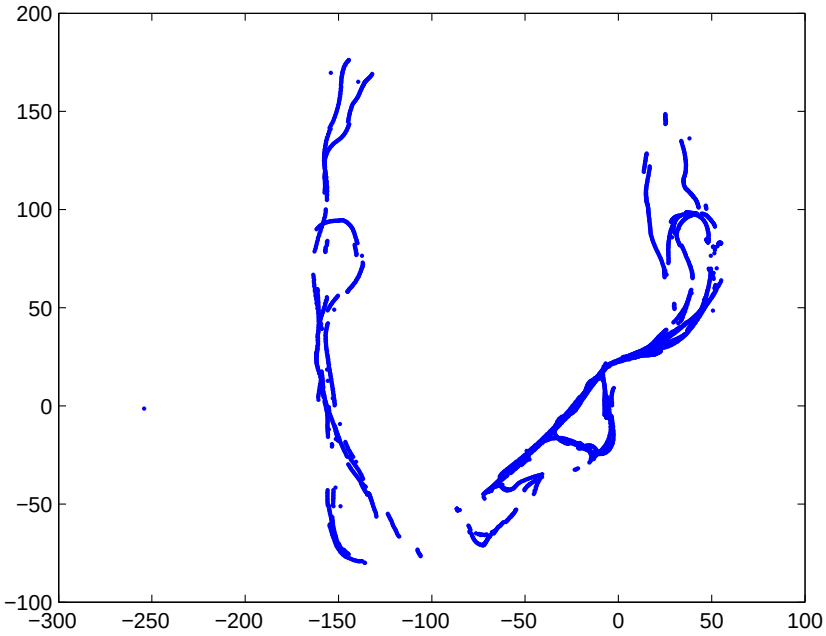


Figure 1: Victoria Park: GPS data

ualberta-csc-flr3-vision

While most are based on laser range finder sensors, we picked one which is based on vision and features using the SIFT algorithm, which is more similar to our data set because it uses the same main sensor.

The data contains images (512) of a small rectangular run through some hall balcony. For each image an extra file provides SIFT features[5], which are computed with the SIFT Demo from Lowe[6]. As some kind of ground truth for own localization experiments a localizer using the ceiling tiles was running in parallel augmenting each image with a global robot state (2D, 3DOF) and a covariance matrix. There is no odometry from the robot aside computing it from the global robot positions. The information about the camera is not suitable for defining a measurement model. It misses the extrinsic and intrinsic camera parameters. It is not easily possible to extract landmarks from the SIFT-features and add them to a map in global coordinates without a defined measurement equation.

The path travelled does not include any special difficulties. It's a round trip through the upper corridor of the Computing Science Centre of Alberta. It closes only one loop.

The data set looks interesting but lacks a lot of information needed to start right away with implementing own algorithms on this set. The SIFT features are a nice add-on, but you could compute them yourself easily and they are not processed further to actually extract track-able landmarks from each image.

2 Experimental Setup

The data set we like to propose evolved from the data used for a PhD Thesis [1] on efficient SLAM. It was recorded at the DLR (Deutsches Zentrum für Luft und Raumfahrt) Institute of Robotics and Mechatronics building using a mobile robot controlled by hand. The building covers a region of 60m x 45m and the robot path consists of three large loops within the building (plus a small outside path) with a total length of 505 meters. On the way the robot visits 29 rooms. You can see an architectural map in Figure 2.

The robot is equipped with four wheels, each with one motor for steering and another one for moving. Since wheel axis and steering axis intersect, the robot is able to move omni-directionally but is not holonomic. Internal robot control estimates the robot's velocity with corresponding covariance. The estimate is based on the wheels'

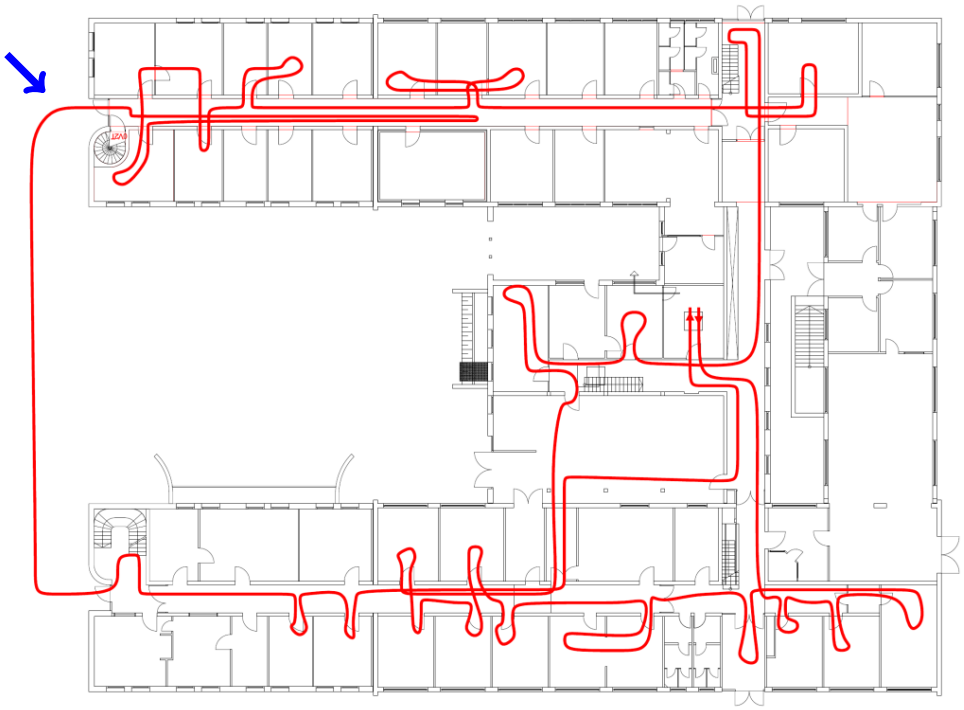


Figure 2: Sketch of the path the robot travelled through the DLR building

angular velocities measured by incremental encoders and the nonholonomic constraint that the wheel moves perpendicularly to its axis. From the estimated velocity, the robot's odometric position is integrated with

$$\begin{pmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_\theta \end{pmatrix} = \begin{pmatrix} \cos p_\theta & -\sin p_\theta & 0 \\ \sin p_\theta & \cos p_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_\theta \end{pmatrix} \tag{1}$$

The derivation of the covariance is also found in [1](p. 50ff). The Y-axis robot coordinate system points contrary to straight forward movement of the robot and the X-Axis points to the left. The robot is equipped with a stereo

Figure 3: The DLR Robot

camera[1] mounted on a pan-tilt unit. For the experiments only one camera was used and the pan-tilt unit stayed fixed. The camera parameters where calibrated with respect to the floor and it's position on the robot's pan-tilt

[1] A standard PAL camcorder zoom/iris/auto focus camera module: Sony EVI-371DG

unit. The robot's coordinate system and the camera coordinate system are translated as you can see in Figure 4. Calibration results and a reference implementation of the camera model (pinhole with radial distortion) are included in the data set. The camera images are interlaced to avoid problems with shifted scan lines, only the even lines from each scan are stored in the image.

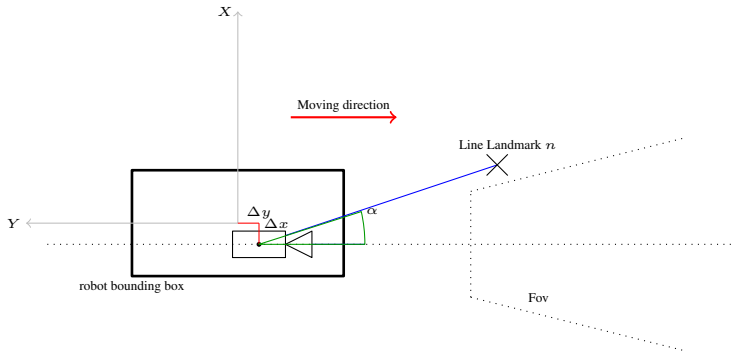


Figure 4: The robot coordinate system, the translation of the camera and the FOV of the camera on the ground

The whole path is discretized in 3297 single steps, each with full information on odometry and all measurements of landmarks in robot-centric coordinates. One particular challenge exists at the left upper corner in the floor plan (arrow in Figure 2). The short outside path has a different surface than inside the building and therefore the calculated odometry error is incorrect for the outside path. This is especially important because there are no overlapping landmark from inside and outside the building and the outside path closes a big loop. This circumstance makes it extra difficult to match landmark observations when re-visiting the corridor again.

The same data set comes with two kinds of landmarks. One can choose between circular disks (fiducial) and vertical lines (natural features). Each landmark has a unique label (identifier) manually annotated to provide data association ground truth. Of course, since this information would not be available in reality, your own solution is not supposed to use it except for evaluation.

2.1 Circular Discs

The first class of features are artificial circular disks that were outlaid on the floor throughout the building. A vision algorithm detects these landmarks and transforms them with the underlying camera model to pure geometric measurements in robot coordinates. Each circular disk has then a x, y-coordinate and a corresponding covariance matrix describing the measurement error. (Figure 5)

Figure 5: One step from the circular disk data set

2.2 Vertical Lines

The second class of features are vertical lines in the office environment which can be door frames, images at the wall, lockers and so on. Because offices are highly dynamic good heuristic to only detect usable and reliable landmarks is to require each line to have a certain length and begin above the horizon. This way moveable table chairs etc. are ignored. A vision algorithm detects such features from the natural unaltered environment. Lines are often very close to each other or they even touch. The algorithm then assembles such a bulk of lines to one feature and adds a probability distribution for the resulting line feature. In Figure 6 you can see an example of detected lines from the data set. The small red horizontal bars at the bottom of each line are an indicator for the probability distribution. After this step it is no longer possible to decide which line you really see in the data.

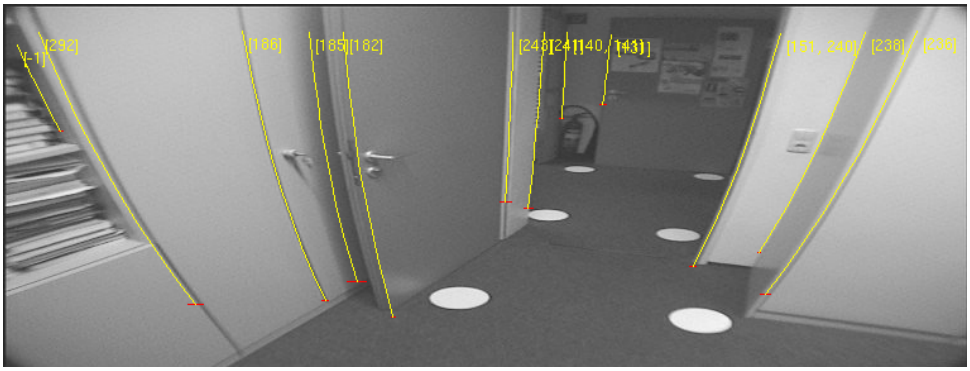


Figure 6: One step from the natural line features data set

In contrast to circles the line landmarks are not artificial. The detection of vertical lines is easily possible with today's vision systems and should be considered as an alternative to other approaches. Especially in indoor environments you can find plenty of these vertical lines, that are usable and also permanent, even when the environment is dynamic otherwise. The main problem in using vertical lines as landmarks is a robust data association. While the circular disks were actually used in experiments, the use of lines as landmarks is still a challenging subject. Having bearing-only data is already more difficult than full relative coordinates but ambiguous annotations are especially difficult. They are necessary since distinct lines observed from far away appear in the same direction and only split up later, when the robot is close enough. So if A and B are very close, from a certain distance no algorithm can decide whether it is A or B. But you would still want the information, that a feature could be A *or* B and not only A if it splits up in A and B later. Information would be lost, since we cannot decide which feature you would expect.

3 The DLR-Spatial Cognition Data Set

Using foreign or unknown data sets needs integration work. You need all sorts of information regarding the setup and calibration of all hardware used. We give you an (complete) overview on how the data sets were generated. And how to interpret certain parts.

You can use the data in three ways, each modality has it's own section because the format of the data may differ.

When you download the data set from [2] and extract the archive, you will find a tree of directories and files. Included are the four runs (c,d,e,g – a,b,f were broken and removed) with a directory for the single images (if you downloaded the big archive) and some files. The main part of the data has the endings `.circles` or `.lines` for the run with circular features and respectively with lines landmarks. In the standard setup the files are called `map.circles` and `map.lines`. In these files for each odometry step a list of measurements are associated. The file `dlr-spatial_cognition-X.cam` contains the camera parameters describing the camera model. The file `dlr-spatial_cognition-X.txt` contains odometry steps only and should usually not be used, because the information is also contained in the `map`-files. The file with suffix `assoziation` and `assoziation_lines` are results of our internal tool for labeling the features and can be ignored. If you find that you need to re-label some features, we can make the tool available.

Beside the data itself we provide some reference code that shows how you can work with the data, like reading the data, interpreting the camera model configuration et cetera. A helper script adds noise to the data sets and several other files give examples how to handle the data sets by implementing for examples a full least-square calculation and EKF-SLAM. For more detail see section 3.4.

3.1 Circular Landmarks

As described above for this set circular disks were used as artificial features.

3.1.1 Data Format

The files `map.circles` contain the information about odometry and measurements. The format is line-based. Lines starting with # are comments, white space lines should be ignored. A odometry step is initiated with the keyword `STEP`. After a `STEP`-Line follow a varying number of `LANDMARK_C` lines $(0 - n)$ each to indicate a measurement of a feature (landmark). The measurements are made after the `STEP` took place.

A `STEP` line is arranged in white space separated parts. After the `STEP` keyword follow ten more entries.

Field	**Meaning**	**Example**	**Units**
Keyword	First word in the line	`STEP`	
Image prefix	how the corresponding image is named in `images` directory	dlr-spatial_cognition-e.0002	
dX	Pose of the robot after movement in the coordinate system before moving	0.00088	m
dY		-0.15647	m
$d\phi$		0.01153	rad
$C_{x,x}$	The entries in the covariance matrix of $(dX, dY, d\phi)^T$	0.000016397281	m^2
$C_{x,y}$		-0.000000002137	m^2
$C_{y,y}$		0.000014026487	m^2
$C_{x,\phi}$		0.000021731727	$mrad$
$C_{y,\phi}$		-0.000000042656	$mrad$
$C_{\phi,\phi}$		0.000261252446	rad^2

A measurement line consists of seven entries and is arranged as followed:

Field	**Meaning**	**Example**	Units
Keyword	First word in the line	`LANDMARK_C`	
p_x	coordinate of the landmark observation in respect to current robot position (measurement)	-0.775408	m
p_y		-1.89416	m
Quality	A measure for how good the detection was $(0 - 1.0)$	0.7294	
$C_{x,x}$	The entries in the covariance matrix of $(p_x, p_y)^T$	0.00173244	m^2
$C_{x,y}$		-0.000000002137	m^2
$C_{y,y}$		0.000014026487	m^2
ID	The unique identifier for this landmark (ground truth)	123	

An example of the data file format is shown in Figure 7.

```
STEP dlr-spatial_cognition-e.0067  0.00391 -0.15602 -0.00218 0.000014424957 \
    0.000000041534 0.000012538057 0.000018179117 0.000000374277 0.000233656364
LANDMARK_C -0.302229 -2.66488 0.962567 0.00169895 0.000127857 0.00293675 26
LANDMARK_C 1.0116 -2.15034 0.951867 0.00175878 -0.000250459 0.00238307 25
LANDMARK_C -0.96104 -1.69011 0.993179 0.00171485 0.000154117 0.00202176 24
STEP dlr-spatial_cognition-e.0068  0.00234 -0.15677 -0.00241 0.000014516278 \
    0.000000028754 0.000012599615 0.000018361033 0.000000288615 0.000234812837
LANDMARK_C -0.176652 -3.54651 0.756458 0.00174355 0.000174856 0.00500673 28
LANDMARK_C -0.302829 -2.53982 0.978849 0.0016918 0.000113261 0.00275804 26
LANDMARK_C 1.01062 -2.01621 0.96499 0.00174479 -0.00021812 0.00226073 25
```

Figure 7: Example of the data file

3.1.2 Measurement Function/Model

To estimate a global position $p(t_1)$ at time t_1 from the current position $p(t_0)$ and odometry d_r a model function is used. To describe the robot motion model for this experiment you can use:

$$\begin{pmatrix} p_x(t_1) \\ p_y(t_1) \\ p_\Phi(t_1) \end{pmatrix} = f_1\left(\begin{pmatrix} p_x(t_0) \\ p_y(t_0) \\ p_\Phi(t_0) \end{pmatrix}, \begin{pmatrix} d_x \\ d_y \\ d_\Phi \end{pmatrix}\right) = \begin{pmatrix} p_x(t_0) \\ p_y(t_0) \\ p_\Phi(t_0) \end{pmatrix} + \begin{pmatrix} \cos p_\Phi & -\sin p_\Phi & 0 \\ \sin p_\Phi & \cos p_\Phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} d_x \\ d_y \\ d_\Phi \end{pmatrix} \tag{2}$$

For convenience we also provide the Jacobian for the model function (see also Section 3.4):

$$J_1 := \frac{\partial p(t_1)}{\partial p(t_0)} = \begin{pmatrix} 1 & 0 & -\sin(p_\Phi)d_x - \cos(p_\Phi)d_y \\ 0 & 1 & \cos(p_\Phi)d_x - \sin(p_\Phi)d_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$$J_2 := \frac{\partial p(t_1)}{\partial d_r} = \begin{pmatrix} \cos(p_\Phi) & -\sin(p_\Phi) & 0 \\ \sin(p_\Phi) & \cos(p_\Phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

For two given global robot positions $p(t_0)$ and $p(t_1)$ the expected measurement (odometry) is modeled as follows:

$$\begin{pmatrix} d_x \\ d_y \\ d_\Phi \end{pmatrix} := f_2\left(\begin{pmatrix} p_x(t_0) \\ p_y(t_0) \\ p_\Phi(t_0) \end{pmatrix}, \begin{pmatrix} p_x(t_1) \\ p_y(t_1) \\ p_\Phi(t_1) \end{pmatrix}\right) = \begin{pmatrix} \cos(p_\Phi(t_0)) & \sin(p_\Phi(t_0)) & 0 \\ -\sin(p_\Phi(t_0)) & \cos(p_\Phi(t_0)) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x(t_1) - p_x(t_0) \\ p_y(t_1) - p_y(t_0) \\ p_\Phi(t_1) - p_\Phi(t_0) \end{pmatrix} \tag{5}$$

With the following Jacobian:

$$J_3 := \frac{\partial d_r}{\partial p(t_0)} = \begin{pmatrix} -\cos(p_\Phi(t_0)) & -\sin(p_\Phi(t_0)) & -\sin(p_\Phi(t_0))(p_x(t_1) - p_x(t_0)) + \cos(p_\Phi(t_0))(p_y(t_1) - p_y(t_0)) \\ \sin(p_\Phi(t_0)) & -\cos(p_\Phi(t_0)) & -\cos(p_\Phi(t_0))(p_x(t_1) - p_x(t_0)) - \sin(p_\Phi(t_0))(p_y(t_1) - p_y(t_0)) \\ 0 & 0 & -1 \end{pmatrix} \tag{6}$$

$$J_4 := \frac{\partial d_r}{\partial p(t_1)} = \begin{pmatrix} \cos(p_\Phi(t_0)) & \sin(p_\Phi(t_0)) & 0 \\ -\sin(p_\Phi(t_0)) & \cos(p_\Phi(t_0)) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7}$$

To transform a global estimated landmark coordinate $\begin{pmatrix} l_x \\ l_y \end{pmatrix}$ to a local landmark observation coordinate $\begin{pmatrix} m_x \\ m_y \end{pmatrix}$ a measurement model is needed. For the circular disks the following model is applicable:

$$\begin{pmatrix} m_x \\ m_y \end{pmatrix} := f_3\left(\begin{pmatrix} l_x \\ l_y \end{pmatrix}, \begin{pmatrix} p_x \\ p_y \\ p_\Phi \end{pmatrix}\right) = \begin{pmatrix} \cos(p_\Phi) & \sin(p_\Phi) \\ -\sin(p_\Phi) & \cos(p_\Phi) \end{pmatrix} \begin{pmatrix} l_x - p_x \\ l_y - p_y \end{pmatrix} \tag{8}$$

With the Jacobian J_5:

$$J_5 := \frac{\partial m}{\partial p_r} = \begin{pmatrix} -\cos p_\Phi & -\sin(p_\Phi) & -\sin(p_\Phi)(l_x - p_x) + \cos(p_\Phi)(l_y - p_y) \\ \sin p_\Phi & -\cos(p_\Phi) & -\cos(p_\Phi)(l_x - p_x) - \sin(p_\Phi)(l_y - p_y) \end{pmatrix} \tag{9}$$

and J_6

$$J_6 := \frac{\partial m}{\partial l} = \begin{pmatrix} \cos p_\Phi & \sin p_\Phi \\ -\sin p_\Phi & \cos p_\Phi \end{pmatrix} \tag{10}$$

3.1.3 Detection algorithm

The detection algorithm is sketched in Figure 8. It is based on Hough transform and a gray-level variance criterion similar to Otsu automatic thresholding[8].

For a given image pixel it is necessary to determine whether this pixel is the center of a circular landmark: The key observation is that there is only a single possible circle with that center, since the radius is known and circles are invariant under rotation in the circle's plane. Since the camera is calibrated with respect to robot coordinates and floor plane, the image outline of this circle can be computed: The optical ray corresponding to the considered pixel is computed in robot coordinates using camera calibration parameters. This ray is intersected with the floor plane. The result defines the center for two circles, one with a radius slightly smaller than the radius of the landmark and one with a radius which is slightly larger. Both circles are mapped back into the image. If, in fact,

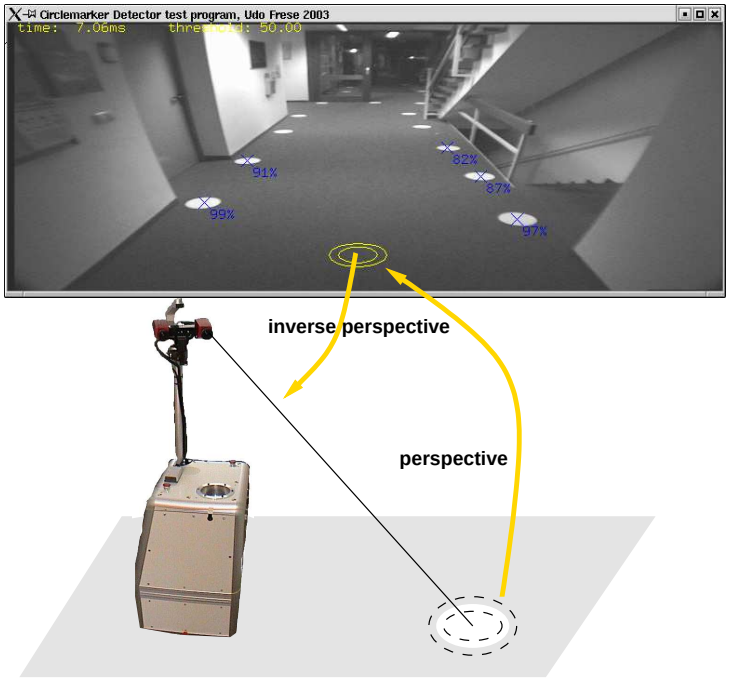


Figure 8: Landmark detection: For each image pixel the perspective outline of two circles centered at the corresponding point on the floor is computed and stored in a lookup table. Detected landmarks are shown by dark crosses together with the computed quality measure.

there is a circular landmark located at the pixel considered, the projection of the smaller circle will be inside the landmarks image and the projection of the larger circle will be outside the landmarks image. Thus, along the circumference of each circle the gray-level is basically constant, whereas it is different comparing both circles.

A threshold for checking gray-level variance statistic is computed along the circumference of the smaller and larger circle as well as along both circles together. The gray-level variance for both circles is a sum differentiated into parts: The variance along the inner circle, the variance along the outer circle (*intra class variances*), and a term proportional to the squared difference of both means (*inter class variance*). For an ideal image the intra class variance should be zero and the overall variance equal to inter class variance. Thus, according to Otsu inter class variance divided by overall variance is a non dimensional quality measure $\in [0 \ldots 1]$, invariant under affine illumination changes.

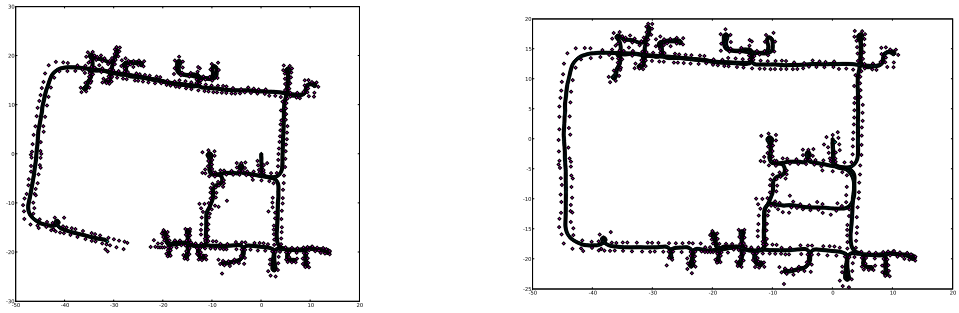
The projected circles' outlines for each pixel being the hypothetical center of a landmark are precomputed and stored. The radius of inner circle, landmarks and outer circle is $6cm$, $10cm$ and $14cm$ respectively. Each outline is represented by 16 equally spaced samples to reduce computation time. A pixel is accepted as a landmark if the quality exceeds a pre-defined threshold (60% in the experiments) and is higher than the quality of all neighbor pixels. For each accepted pixel the corresponding location relative to the robot is passed as a landmark to the landmark identification algorithm. Computation time is $8ms$ for a 768×288 image on a Intel Xeon, 2.67 GHz.

The implementation is available from the authors upon request in case you plan to use circular fiducials in your own experiments. It does however need a calibration of the camera with respect to the ground plane.

3.1.4 Experimental Results

The circle-based part of the data set was already used in the experiments of [1] for the *treemap*-algorithm. We a added some simple experiments to test the data set and it's annotation by computing a full least-square estimated map. We used the Levenberg-Marquardt method on the full information and generate a map. In Figure 9 the result is shown. The outside path has a relatively large odometry error (see Figure 9(a)), so that closing the loop when the robot enters the building again is one of the main difficulties in this set.

To quantify the quality of the provided covariance matrices for landmark measurements and the odometry error we use the χ^2 test. A good measure could be that the minimum χ^2 error should be nearly equal to the difference of the dimension of the measurement vector ($\dim(z)$) and the size of the state vector ($\dim(x)$). With n landmarks



(a) Map before loops closure

(b) Final estimated map

Figure 9: Estimated map based on least-square approach (Newton)

and p poses in the state vector and m measurements of landmarks and t odometry steps this evaluates to

$$\chi^2 \approx \dim(z) - \dim(x) \tag{11}$$

with

$$\dim(x) = 2n + 3p,\ \dim(z) = 2m + 3t \tag{12}$$

Values and Results

p	=	3298
n	=	576
t	=	3298
m	=	14309
χ^2-error	=	56871.4
Estimated χ^2-error	=	27466

The χ^2-error is the result of nonlinear optimization with Levenberg-Marquardt, whereas the Estimated χ^2-error is the result one should get on average if the covariances were precise. So apparently the dataset is overconfident by a factor of $56871/27466 = 2.07$ on the covariance level or 1.44 on the sigma level. That's not too bad and is likely to happen pretty often.

3.2 Line Landmarks

3.2.1 Data Format

The files `map.lines` is in similar format as the circle-based data set. A odometry step is also initiated with the keyword `STEP` after a `STEP`-Line follow a varying number of `LANDMARK_L` lines $(0 - n)$ each to indicate a measurement of a feature (landmark). The measurement are made after the `STEP` took place.

The `STEP` entries are the same as already described in section 3.1.1.

A `LANDMARK_L` line has at least nine entries after the keyword. The last item in this line is the (ground truth) ID of the Line, because of the ambiguous nature of the lines (see above) all IDs that correspond to this line in this particular image this may be several entries. The format is then as follows

Field	**Meaning**	**Example**
Keyword	First word in the line	`LANDMARK_L`
x_{top} y_{bottom} x_{top} y_{top}	These values are only for visualization issues. The y described a normalized line in image coordinates to draw the line for example in the viewer	
α	Angle of observation in robot coordinates in radians (the measurement)	0.600694
β	Angle of observation with respect to the vertical	0.0251159
Quality	Old quality (unused)	always 0
σ_α	The error of the line (standard deviation of α)	0.00291193
$ID[ID..]$	One or more unique identifier(s) for this landmark (ground truth)	123 34 56

An example of the data file format is shown in Figure 10.

```
STEP dlr-spatial_cognition-e.0000  0.00088 -0.15647  0.01153 0.000016397281 -0.000000002137\
          0.000014026487 0.000021731727 -0.000000042656 0.000261252446
LANDMARK_L -0.80286 -0.625259 -0.694207 -0.273645 0.600694 0.0251159 0 0.00291193 1
LANDMARK_L -0.474484 -0.583026 -0.397523 -0.148873 0.39069 0.0302675 0 0.00574867 4
LANDMARK_L -0.300722 -0.586752 -0.220978 0.0441848 0.255339 0.00867793 0 0.0027596 9
LANDMARK_L -0.169833 -0.58019 -0.147315 -0.254943 0.14735 0.0130423 0 0.00289818 14 681
LANDMARK_L -0.0350638 -0.571825 -0.0321665 -0.22555 0.0318137 0.0194733 0 0.002966 20
LANDMARK_L 0.144026 -0.576694 0.118327 -0.0896802 -0.122526 0.00451754 0 0.00954949 17 18 19
STEP dlr-spatial_cognition-e.0001 -0.00470 -0.18707  0.06820 0.000019834605 -0.000000084074 \
          0.000016337711 0.000028153298 -0.000000708950 0.000304450153
LANDMARK_L -0.73896 -0.618702 -0.641319 -0.255913 0.5639 0.040412 0 0.00277288 1
LANDMARK_L -0.518443 -0.592225 -0.469403 -0.325537 0.421207 0.0414013 0 0.00273231 2
```

Figure 10: Example of the data file

3.2.2 Measurement Function/Model

The measurement model for vertical line feature is not as complete as for the circle case. We only have bearing information and no distance. It is not possible to convert the bearing information from the camera center to a bearing information related to the robot frame without knowing the distance. At most you can make an assumption to get a better initial estimate. In this data set the lines that are detected are usually more then $1m$ away and closer than $5m$. But an algorithm that exploits this knowledge isn't generally applicable. If you work with the data set the following measurement model could be used. This model defines the calculation of an angle of any landmark seen by the robot from its global estimated coordinate. The translation Δx and Δy between the robot frame and the camera coordinates (as seen in Fig. 4) is part of the camera calibration (shown in detail in the next section in Figure 11) in the last column of the matrix sensor2World.

The robot motion model from the previous section applies also for the line landmark set and can be re-used.

To transform a global estimated landmark coordinate $\begin{pmatrix} l_x \\ l_y \end{pmatrix}$ to a local landmark observation angle α, the following formula must be applied with $\begin{pmatrix} p(t)_x \\ p(t)_y \\ p(t)_\theta \end{pmatrix}$ being the robot pose:

$$\begin{aligned} \alpha = \text{atan2}\left(-s \cdot (l_x - p(t)_x) + c \cdot (l_y - p(t)_y) - c_y, c \cdot (l_x - p(t)_x) + s \cdot (l_y - p(t)_y) - c_x\right) \\ \text{with } s = \sin(p(t)_\theta) \text{ and } c = \cos(p(t)_\theta) \end{aligned} \tag{13}$$

Here (c_x, c_y) is the position of the camera in robot coordinates (0.0179777m, -0.0808609m). The orientation of the camera is already included in the measurement α provided in the data-set. However, the derivation of the Jacobian of this formula is rather complicated and you can re-use the formulas from the circle case to describe the transformation. Using the measurement model for $\begin{pmatrix} m_x \\ m_y \end{pmatrix}$ for the relative location of one circle landmark from the previous chapter, α can be rewritten as:

$$\alpha = atan2(m_y - c_y, m_x - c_x) \tag{14}$$

We assume that the line landmarks intersect the ground plane at some point. We use this point to determine the angle α. The transformation to the camera frame is a simple operation.
The Jacobian of α with respect to m, l and $p(t)$ are:

$$\frac{\partial \alpha}{\partial m_x} = \frac{-(m_y - c_y)}{(m_y - c_y)^2 + (m_x - c_x)^2} \tag{15}$$

$$\frac{\partial \alpha}{\partial m_y} = \frac{m_x - c_x}{(m_y - c_y)^2 + (m_x - c_x)^2} \tag{16}$$

$$J_7 := \frac{\partial \alpha}{\partial m} = \left(\frac{c_y - m_y}{(m_y - c_y)^2 + (m_x - c_x)^2}, \frac{m_x - c_x}{(m_y - c_y)^2 + (m_x - c_x)^2} \right) \tag{17}$$

$$J_8 := \frac{\partial \alpha}{\partial l} = \frac{\partial \alpha}{\partial m} \cdot \frac{\partial m}{\partial l} = J_7 \cdot J_6 \tag{18}$$

$$J_9 := \frac{\partial \alpha}{\partial p} = \frac{\partial \alpha}{\partial m} \cdot \frac{\partial m}{\partial p} = J_7 \cdot J_5 \tag{19}$$

The Jacobian J_5 and J_6 are equations 9 and 10, respectively.

3.2.3 Algorithms

The vertical line detection algorithm[2] described in this section detects lines vertical in 3D space from a camera perspective that is roughly calibrated relative to the ground plane. In particular, the lines need not be vertical and not even straight in the image, so the algorithm can handle camera distortion and tilted perspectives. Purposely it detects only lines passing through the horizon. First, this gives more reliable landmarks, because objects as least as tall as the robot are less likely to be moved around than smaller objects, in particular if the robot is human-sized. Second, it reduces computation time, since the algorithm first looks on the horizon and only if it finds a line there tracks the line up and down.

The algorithm computes a lookup table beforehand using the camera calibration that specifies the shape of the different vertical lines that are searched for in the image, so for the actual detection algorithm no computations using the calibration data are necessary. Specifically, the table contains for each image x-coordinate the y-coordinate of the corresponding point on the horizon and the vertical line passing through that point. The line in turn is defined by an array giving the x-coordinate $x(y)$ for each y-coordinate. This format makes searching for lines very efficient, but requires, that the lines are very roughly ($\pm 45^\circ$) vertical in the image.

When one of these lines is tracked, first the algorithm extracts a small stripe of image data around the precomputed line. This is done by taking in each row y of the stripe the point x(y) stored in the precomputed table as well as some surrounding to the left and right. The effect of this operation is that the hypothesized line is now exactly vertical in the stripe. Line tracking is based on the response from a contrast normalized differential of Gaussian filter. The filter is elongated ($\sigma_x = 2.5$ pixel and $\sigma_y = 7$ pixel), making it selective to vertical lines. The purpose of contrast normalization described in the the following is to be able to detect edges over a wide range of intensity difference. This is not possible with a hard threshold on the DoG response, because a small DoG response could be caused by an edge with low intensity difference, but also by any high contrast image patch where some part of the contrast is in the direction detected by the DoG filter.

The squared DoG response is devided by a Gaussian weighted image variance. This result indicates, which percentage of the local image contrast looks like an edge. So this is a illumination invariant indicator of edge quality $\in [0 \dots 1]$. The number 0.4 is chosen as a threshold for an valid edge. The Gaussian weighted image contrast is computed from the response of the Gaussian filter on the image and the response on the squared image accoring to the well known formula $V(X) = E(X^2) - (E(X))^2$. Additionally a regularizer roughly corresponding to the image noise variance is added to prevent a $\frac{0}{0}$ situation for constant image patches.

The algorithm described so far would require the vertical lines to be exactly vertical within pixel precision. This is not realistic, since robots pitch and roll when moving. So, while proceeding with the filter along the precomputed line, the algorithm tracks a horizontal offset from the nominal line. This is done, by evaluating the filters on 3 adjacent x-coordinates and determining a subpixel optimum from an interpolation parabola through the filter responses.

The tracking process starts on the horizon going up and down, tolerating some gaps, where the normalized DoG response falls below the threshold. From the set of y-values and x-offset values, a eigenvalue decomposition is computed, where the smaller eigenvalue gives an indication of how straight the edge is and the larger eigenvector gives the direction of the edge. Of course, this direction is mostly vertical, but finding the direction of slightly non-vertical edges allows to compensate for robot tilt later.

The last step consists of some postprocessing. The algorithm tries to find pitch and roll angles for the robot that are consistent with the largest number of lines being vertical. Lines not compatible with these angles are discarded. The horizontal bearing angle derived from the lines image position is annotated with an uncertainty sigma corresponding initially to one pixel. Images can contain a lot of vertical linessomtimes very close to each other which are then almost impossible to distinguish. So the detector joins lines closer than a threshold deriving mean and std. deviation of that line according to the mixture of Gaussian formulas. The effect is, that when lines are joint the resulting line has an uncertainty corresponding to the extend of the joint lines. This corresponds to an observation such as "there is a line somewhere in this area".

The motivation of this step is that the raw vertical lines are sometimes so close, that even visually it is impossible to decide, which one is which.

[2]The implementation is available upon request.

3.3 Raw Data

Many researchers from the vision community work on pure Visual SLAM without predefined geometric features. For this scenario the data set is still usable but you cannot benefit from the ground truth annotation of circles and lines. We provide in this section all important information for the use of the raw data.

The data set has 3297 steps. For each step the robot took an image. To avoid problems with the interlaced PAL image of the camera only the even scan lines of each image are used. This leads to an image size of 768×288 pixel. The images contained are black and white compressed JPEG files (the original files are available upon request).

3.3.1 Measurement Function/Model

The camera model is based on pinhole-camera with added radial distortion. It provides the mapping from 3D robot to 2D image coordinates. The four steps of this mapping are transformation to 3D camera coordinates, perspectivity, distortion and scaling/offset.

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x_{\text{scale}} \\ y_{\text{scale}} \end{pmatrix} \cdot \text{distort} \left(\text{persp} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} \right) + \begin{pmatrix} x_{\text{offset}} \\ y_{\text{offset}} \end{pmatrix} \tag{20}$$

with $(x_i, y_i)^T$ being the image coordinates and $p = (p_x, p_y, p_z)^T$ the coordinates in the camera coordinate system. The distortion equation is defined as follows:

$$\text{distort} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \cdot \left(1 + \frac{a_2 r^2 + a_3 r^3}{1 + b_1 r + b_2 r^2 + b_3 r^3} \right), r = \sqrt{x^2 + y^2} \tag{21}$$

and perspective equation

$$\text{persp} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix} \tag{22}$$

With a matrix describing offset, scale and transformation (called sensor2World here for software reasons but actually being camera2Robot) in respect to the world coordinates, which is a result from the calibration process, the equation can be rewritten as:

$$\text{distort}(\ \text{persp}(\ \text{sensor2World}^{-1} \cdot [p_x, p_y, p_z, 1]^T)\ ) \tag{23}$$

All parameters (a_2, a_3, b_1, b_2, b_3, scale, offset, transformation matrices) can be retrieved from the camera calibration file for each run in the sets directories (e.g. `dlr-spatial_cognition-e.cam`). The format should be self-explanatory (see Figure 11) the `coef` line lists the distortion parameters $(a_2, a_3, b_1, b_2, b_3)$. The matrix `sensor2World` describes the matrix to transform a point from camera coordinates to the robot coordinate system which is located on the floor in the center of the robot with X pointing left, Y backward and Z up (see Figure 4). Since `sensor2World` is 4×3 you need to add the row [0 0 0 1]. Example code on how to use the files and how to implement the camera model is also included (see section 3.4).

```
{ SmRadialDistortedCamera
sensor2Tcp: [ 0.99992, -0.00994383, -0.00778315, -0.0820676 ]
[ 0.00718218, -0.0591054, 0.998226, 0.142944 ]
[ -0.0103862, -0.998202, -0.0590291, 0.0522051 ]
sensor2World: [ -0.99992, 0.00994383, 0.00778315, 0.0179777 ]
[ -0.00211987, 0.475426, -0.879753, -0.0808609 ]
[ -0.0124484, -0.879699, -0.475367, 1.50354 ]
mountIndex: 0
width: 768 height: 288 scaleX: 464.762 scaleY: 232.381 offsetX: 384 offsetY: 144
coef: { -0.0620672 -0.324903 0.314764 1.52524 -0.480806 }
}
```

Figure 11: The camera calibration file

If needed the raw odometry data is available from the already introduced files. Or alternatively, from the file in each directory `dlr-spatial_cognition-x.txt`. The `STEP` line is the same as already presented, the difference is that no measurements are included in this file.

3.4 Additional material

In addition to the data and information about measurement models, which is all needed to effectively work with this data set, we also want provide useful example code to show how you could use the data set.

The example code is also a reference on how to load and interpret the data. To show how easy it is to actually use the data set and implement your own algorithms, we already included a least-square optimizer based on the Levenberg-Marquardt method, a EKF localizer using the calculated map and an EKF SLAM algorithm. These small programs may also be used for educational use as the whole set may be useful for teaching and courses.

We also provide a small script to add noise to the data set. The real data per run consists of only one file, which you can feed to the tool. It then outputs a new file with added noise on all measurements. You can influence the noise with commandline switches.

Of couse, if code evolves from the community, we are happy to include it in the data set, which will be maintained on the corresponding web site[2], where you can also get videos, calculated maps and plots.

4 Recommendations

When using a special data set like the one we presented here you develop an algorithm to compute a solution for example for the data association problem while constantly checking results on the set. This co-evolution may lead to a specialized algorithm that works fine with the given set but fails (or gives bad results) with other data. To avoid this problem we recommend that you try out all included runs to have different data for a similar scenario. For the different runs the path may change and the measurements will be different but the environment will be the same.

To improve robustness even further you can add (a lot) noise to the data and re-run your algorithm to compare the results. As described in section 3.4 we already have a small script that adds random gaussian white noise to the measurements.

Acknowledgement

Thanks to Christoph Brachmann for annotating the data set and the German Aerospace Center (DLR) for providing the raw data on which the data set is based.

References

[1] U. Frese. *An O(log n) Algorithmm for Simultaneous Localization and Mapping of Mobile Robots in Indoor Environments*. PhD thesis, Technische Fakultät Universität Erlangen-Nürnberg, 2004.

[2] U. Frese and J. Kurlbaum. A data set for data association. Website, `http://www.sfbtr8.spatial-cognition.de/insidedataassociation/`, 6 2008.

[3] J. Guivant, E. Nebot, and S. Baiker. Autonomous navigation and map building using laser range sensors in outdoor applications. *Journal of Robotic Systems*, 17:3817–3822, 2000.

[4] A. Howard and N. Roy. Robotics data set repository (radish). Website, `http://radish.sourceforge.net`.

[5] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[6] D. G. Lowe. Keypoint detector. Website, `http://www.cs.ubc.ca/~lowe/keypoints/`, 6 2008.

[7] E. Nebot. Victoria park data set. Website, `http://www-personal.acfr.usyd.edu.au/nebot/dataset.htm`, 6 2008.

[8] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62 – 66, 1979. VISION: describes a graylevel variance based thresholding scheme.

[9] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, Cambridge, 2004.