RAWSEEDS

Robotics Advancement through Web-publishing

of Sensorial and Elaborated Extensive Data Sets

Raw seeds can be consumed as they are...
or be the start for the growth of new results.

RAWSEEDS

WorkPackage 2

Deliverable D2.1

Raw Data (indoor)

Accompanying Document

PART 2 OF 2: GROUND TRUTH COLLECTION AND VALIDATION

due date: January 31, 2009

authors: Giulio Fontana (WP-2 Leader), Daniele Marzorati, Alessandro Giusti, Pierluigi Taddei, Davide Rizzi

contributors: Matteo Matteucci, Domenico G Sorrenti, Simone Ceriani

date of preparation of this document: March 13, 2009

Table of Contents

Introduction (from Part 1)

> *NOTE. This is a copy of the Introduction. The Accompanying Document to Deliverable D2.1, in fact, has been split into two Parts (for reasons that are explained below), and the document you are currently reading is the second of those Parts. The Introduction, therefore, has been reproduced here from Part 1.*

Deliverables D2.1 and D2.2 include the datasets collected in - respectively - indoor and outdoor environments. Such explorations have been performed with the mobile robot designed and set up by WorkPackage 1 of RAWSEEDS. D2.2 also includes the datasets recorded in mixed, i.e. outdoor+indoor, locations.

While the main content of Deliverables D2.1 and D2.2 are the datasets, additional data and information are needed to make these datasets fully usable. The additional data is attached to the datasets, while the additional information is available in the form of two *Accompanying Documents*: the present one, associated to D2.1; and a similar one, associated to D2.2.

Notwithstanding the numbering, D2.2 was due - and actually delivered - *before* D2.1. This was due to changes in the schedule of the project (subsequent to an Amendment to the Contract) and to the need to perform outdoor acquisitions during the summer to avoid weather-related problems. The collection of the data for both Deliverables and the successive processing, formatting and verification of those data have been performed in accordance with the indications of Deliverables D1.1 (*Roadmap - indoor*) and D1.2 (*Roadmap - outdoor*) and with the recommendations of WorkPackage 3 (which is dedicated to data verification and validation).

WorkPackage 3 performed a thorough analisys of the datasets and of their associated data and documentation, and generated a set of recommendations to improve and maximize their quality. The results of the work of WorkPackage 3 are collected into two Deliverables: D3.1 (*Preliminary data validation*) and D3.2 (*Final data certification*). D3.1 was the result of a validation work performed on preliminary datasets, specifically generated to make such work possible: by the time when the final datasets were collected, all the issues raised by Deliverable D3.1 had been addressed and solved, as documented by D3.2. Deliverable D3.2 states, indeed, that RAWSEEDS' final datasets reach the required quality standards; it also introduces some suggestions to improve the usability of such datasets and to correct minor imperfections. These suggestions have been implemented as well, both in the datasets and in Deliverables D2.1 and D2.2. Therefore, at the time of delivery of this document, RAWSEEDS' datasets and all associated information (including this document) are compliant with the recommendations of WorkPackage 3.

As said above, the datasets form the core and the bulk of D2.1, but additional material is required for their best use. This material comes in the form of associated data (such as calibration data for the cameras) and in the form of descriptions and explanations. For this reason, both Deliverable D2.1 and Deliverable D2.2 are actually constituted by three elements:

1. the datasets;
2. the associated data;
3. the additional descriptions and explanations.

Parts 1 and 2 (according to the list above) of D2.1 have been published electronically on a server set up by RAWSEEDS. Publication within a single document was not an option, as massive amounts of data are involved (more than one TByte). Part 3 is given by this document, which includes all the information needed to access and use parts 1 and 2. This is the reason why the title of the present document is not simply "Deliverable D2.1" but "Deliverable D2.1 - Accompanying Document".

For practical reasons, this Accompanying Document has been split into two separate documents: Part 1 (Datasets) and Part 2 (Ground Truth Collection and Validation). This has been done to facilitate consultation by the users of RAWSEEDS' datasets, as the kind of information included into the two parts are conceptually different.

Part 1 includes all the information needed to use the datasets, including a complete description of the procedures chosen to acquire the data and to calibrate the sensors.

Part 2 describes in detail how the ground truth associated to the datasets has been generated and validated. However, *this knowledge is not needed to simply use the ground truth*. Therefore, the consultation of Part 2 is only needed by people wanting to scrutinize closely the characteristics of RAWSEEDS' ground truth, or to base on RAWSEEDS' approach to ground truth collection and validation to perform further work in that field.

Notwithstanding the separation between Part 1 and Part 2, they are actually two parts of the same document. For this reason, section numbering have been made coherent between them, allowing to easily reunite them - if needed - to form a single document.

4. Ground truth

Ground truth data are recordings of (segments of) the actual trajectory of the robot while traversing the explored environment, time-synchronized with the data coming from the sensors on board the robot. This section describes the systems used to generate and validate the ground truth associated to RAWSEEDS' indoor datasets. In addition to that, it describes how synchronization was achieved.

Two ground truth collection systems were used, both completely independent from the sensors on board of the robot (in addition to being independent one from the other). Specifically, a **vision-based ground truth** was generated by a multi-camera vision system, and a **laser-based ground truth** was generated by a system using several laser range finders. In this way, two different ground truth data streams have been made available for each dataset. This was useful for different reasons, i.e.:

- because the above streams were examples of the kind of positioning information that can be provided from two widely different systems, thus helping performance comparisons between such systems;
- because the two ground truth sources could be used to validate each other;
- because a higher-precision ground truth data stream could be produced by fusing - with a suitable algorithm - the two available streams. This “fused” stream is the one that has been included into RAWSEEDS' Benchmark Problems, and will be simply called **ground truth** in the following.

Please note that *it is not necessary to know the contents of this Part of the document to* use *the ground truth provided along with RAWSEEDS' datasets.* Such knowledge is only necessary for an assessment of the properties of the ground truth.

In the following, the term "ground truth" will be often substituted with its acronym "GT".

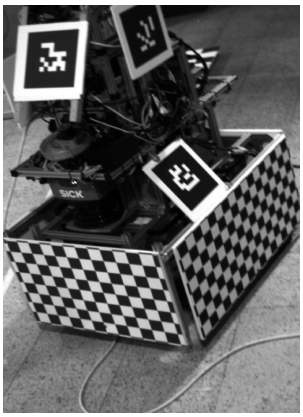
4.1 Ground truth collection

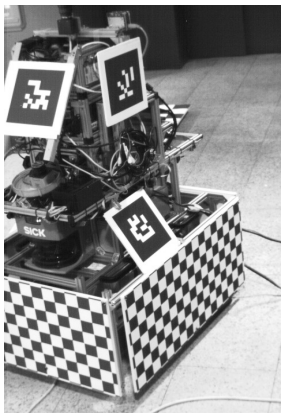
This Section describes how the ground truth data streams have been generated and how they have been fused to produce the GT stream for the Benchmark Problems. In addition to that, we will describe how we tackled the (not trivial) problem to ensure good synchronization between the mobile robot and the GT collection apparatus in presence of an unreliable network connection between the two systems (i.e., one that was subject to heavy, fluctuating delays and to abrupt connections and disconnections).

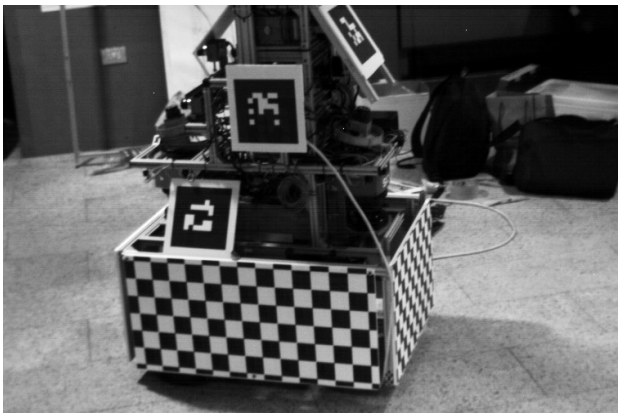
4.1.1 Vision-based GT

Vision-based ground truth (also called **GT-vision**) has been collected by attaching

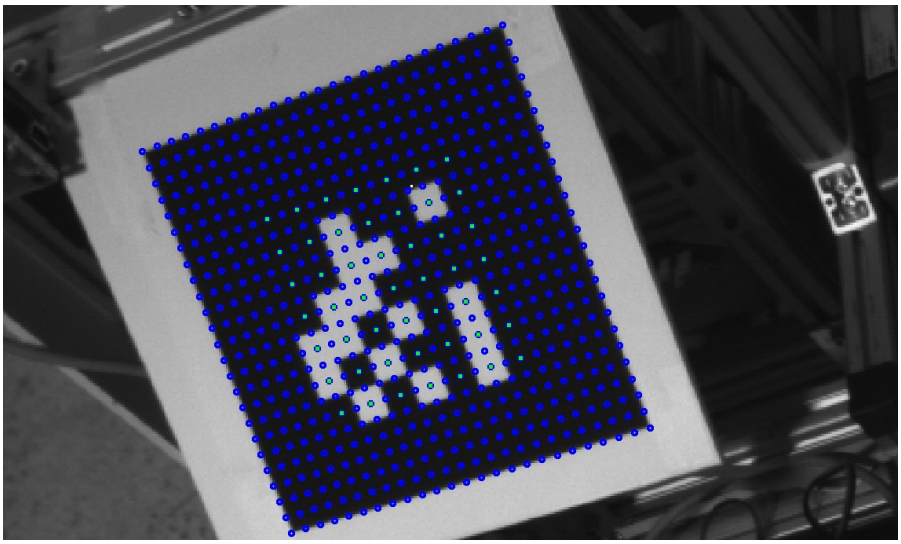
several planar visual markers to the robot, then observing these markers from external calibrated cameras fixed to the environment. Such markers are a variation of visual markers commonly used in Augmented Reality, in particular in the ARToolkit system (http://www.hitl.washington.edu/artoolkit/).







Each marker is printed on an A4 sheet of white paper, and placed on a rigid planar surface; the marker is a black square, containing internally a 6x6 matrix of black and white cells, where the marker ID is encoded; the ID is a 9-bit number, which is scrambled with a XOR mask in order to provide robustness and ensure that a minimum amount of black or white cells is always available: for example, this allows one to encode number 0 without having an “all-white” marker. We will exploit this property in the following in our enhanced marker detection code. In particular, each cell is 1/12 of the total width of the marker; the inner 6x6 matrix is not centered in the marker square, but instead displaced half a cell towards the right and top directions; this makes such markers incompatible with ARToolkit, but leads to better discrimination of the markers' rotations in our algorithms, as we detail in the following.



A high-resolution marker, with a 24x24 grid (blue points). Green dots show sampling points for the centered 7x7 matrix. Only a 6x6 submatrix represents the actual marker ID: the remaining values help in increasing resiliency to rotational ambiguities, which become an issue in smaller marker images.

Before generating ground-truth data, the relative positions of the different markers on the hull of the robot must be computed from a number of high-resolution images taken from a hand-held camera. When the relative positions of the markers on the robot frame are known, the robot can be localized from the images of the external cameras, by detecting and reconstructing the 3D position and rotation of the visual markers. Identifying the ID of each marker allows us to compute the position of the robot's frame of reference, even when few (or one) markers are visible.

The following parts of this Section are dedicated, respectively, to the description of: (i) the calibration of the camera system; (ii) the detection technique for the markers; (iii) how the robot position is reconstructed, once one or more markers are localized in the image of an external camera; (iv) how the relative 3D position and rotation of different markers is computed from the images of the handheld and external cameras.

Calibrating the vision system

The vision system is calibrated in two main stages, described below.

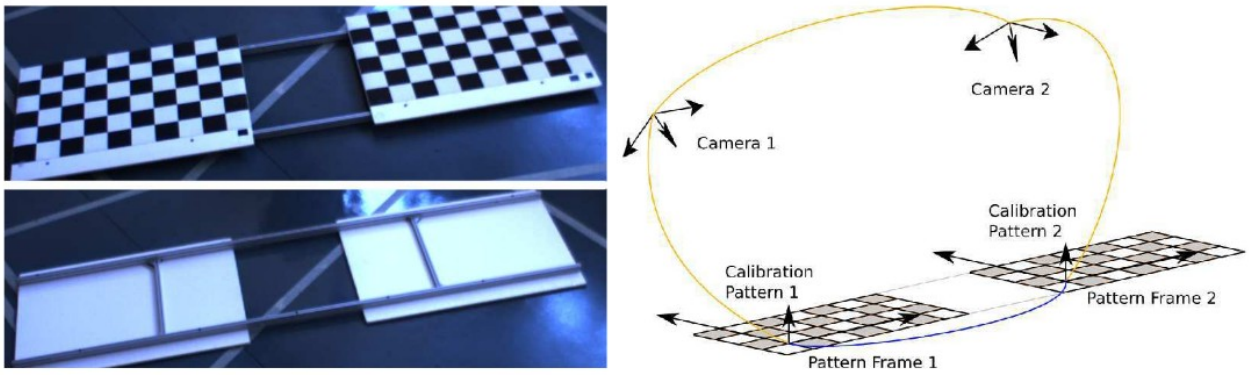
Camera setup and internal calibration. First, the cameras are placed and securely fixed in their position. Then, their fields of view are marked on the ground, in order to ease the following steps. Each camera is then independently calibrated in order to recover its internal parameters, by means of the Matlab Camera Calibration Toolbox; we use a checkerboard large enough to fill a significant part of the field of view of the camera, whose focus had been previously set to an appropriate distance.

External calibration of cameras. After all cameras are internally calibrated, we

compute the external parameters of each camera, i.e. the rototranslation which moves the world coordinates to the camera coordinates. This step is critical for a number of reasons.

- The fields of view of the cameras overlap only partially, in order to cover a larger area; this is a difficulty for the recovery of camera-to-camera rototranslations.
- Only one camera observes the “world” reference frame, and can be directly localized with respect to it.

First of all, a checkerboard is placed on the floor with a corner in the origin of the "world" reference frame, with sides parallel to the two horizontal axes of such frame; then, one of the cameras is calibrated in such a way that the "world" reference frame also becomes its own reference frame. Then, in order to connect the fields of view of adjacent cameras, we use a double calibration checkerboard mounted on a solid mechanical frame (shown in the figure below), in such a way that the relative position of the two patterns is known and stable. Then, we can place each of the two patterns in the field of view of each of the two adjacent cameras, and acquire several images with different positions and orientations of the double chessboard: this allows us to compute an accurate estimate of the relative position between the two cameras. The extrinsic parameters of each camera with respect to the “world” reference frame (which is only seen by the first camera) are finally computed by chaining the camera-to-camera rigid transformations.



Detecting and localizing the images of markers

The marker detection step analyzes every frame of the input video, detects markers, and finally outputs the accurate image coordinates of the 4 corners of every detected marker -- in a consistent order -- along with the ID of each marker.

Requirements

The requirements for this step are extremely strict, but the task is eased by the very constrained prior information about the scene. In particular, two requirements are fundamental, as their violation would originate excessive errors in the reconstructed ground truth data.

- Perfect sensibility: no marker must be detected where there is none;

- No marker misidentifications: a marker must always be correctly identified, when it is detected.

Such requirements may be relaxed only by implementing a robust 3D reconstruction step with the ability of excluding outliers in detected markers; we did not choose this possibility, therefore we must ensure that the two requirements introduced above are always met.

Moreover, in order to provide a good output quality, we also expect the following:

- Extreme localization accuracy: the corners of a marker should be localized with a very low uncertainty.
- High sensitivity: a marker should be detected, if it is visible;

Localization accuracy of the marker corners in the image is extremely important: due to the geometry involved in the 3D reconstruction technique, small errors in such data causes very large displacements in the reconstructed robot position. In particular, the angle between the viewing rays associated to the opposite corners of a marker is very often extremely narrow – less than 1 deg: therefore, small errors in the backprojection of those rays translate to macroscopic errors in reconstructing depth.

Overview

The process is divided in three phases:

1) Detection of candidate markers;

2) Basic filtering of candidate markers, and rough corner localization;

3) Corner localization refinement, marker identification of and final validation.

On some datasets, phases 1 and 2 can be substituted by ARToolkit, whose results (marker corners and IDs) are then fed to phase 3 for refinement (which is necessary for 3D reconstruction, as noted previously) and final verification. Some datasets are not immediately compatible with ARToolkit detection, and are always analyzed by means of the algorithm described below. In either case, accuracy of results does not change as it is determined by phase 3, which is always performed; instead, the number of false negatives (i.e. visible markers which are not detected), depends on the first two phases and may change depending on the specific technique.

Comparison between ad-hoc marker detection and ARToolkit

ARToolkit has problems with some of our datasets for a number of reasons:

- marker contrast is sometimes poor, and some markers are unevenly illuminated;
- markers often appear small and/or very tilted;

- there may be a significant amount of defocus or motion blur, which is especially disruptive when markers appear as a few tens of pixels wide;
- marker edges are sometimes partially occluded.

ARToolkit handles some of these difficulties, but must accommodate severe constraints of computational efforts, as it is designed to run online on a video stream. Our system, on the contrary, performs a more detailed analysis and can often outperform ARToolkit in detecting markers, at the expense of much increased computational requirements. Moreover, our system always outperforms ARToolkit in localization accuracy of the marker corners, implemented in phase 3, which is always performed.

Finally, the markers used in the final datasets are not directly compatible with ARToolkit, as they use an off-center ID matrix, which we exploit for improved robustness to rotation ambiguities.

Phase 1: Detection of Candidate Markers

This phase takes care of detecting a set *CM* of candidate markers in the given frame, providing a roughly-segmented binary mask *M* representing each marker. Since, in the following, candidate markers are going to be refined and discarded, but not added, this phase is designed to be very sensitive, in order to reduce the probability of missing valid candidates. A significant amount of invalid candidates (i.e. candidates not being actual markers) are likely to be detected in this phase and returned in set *CM*; they will be discarded and filtered in the following.

The frame is analyzed in order to detect brightness patterns compatible with the presence of a marker: in particular, high-contrast areas with the morphological characteristics of a marker are isolated, binarized, then segmented by means of connected component labeling. Every resulting connected component is considered a candidate marker with an associated binary mask *M*.

The marker is defined as the dark square, surrounded by a white border, and containing the ID matrix.

The first step is a grayscale closure on the original image I_1 with a circular kernel, which is roughly as large as half of the expected width of the marker image. Let I_2 be the resulting image.

We now explain the rationale of such operation by analyzing its effect on the marker image. The grayscale closure operation is defined as a grayscale dilation followed by a grayscale erosion.

- After the dilation, the dark areas inside the marker are completely filled with the bright areas, which expanded from the border and from the white cells in the ID matrix; the white marker support also expands outwards.
- The subsequent erosion does not significantly affect the internal part of the marker support, which stays mainly white; the external borders of the marker may contract back to the original position, depending on the surroundings of the marker itself.

The final effect of the closure operation is that, in I_2, the rectangular marker supports appear uniformly white.

The original image I_1 is now subtracted from the closed image I_2, which leads to an image I_3 where larger pixel values correspond to pixels which were both:

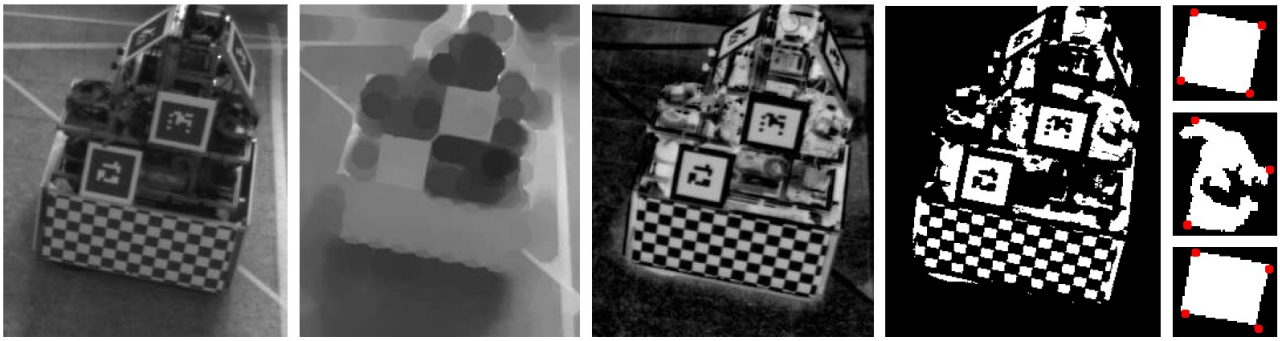
- dark in the original image I_1, and
- bright in I_2.

This strongly improves the contrast of the markers, which appear white on black supports in I_3. A thresholding operation on I_3 follows .

In our final refined implementation, the subtraction operation is optionally substituted by the inverted result of an element-by-element division between I_2 and I_1:

$I_3 = 1-(I_1./I_2)$.

Such an operation is conceptually similar to the subtraction, but provides an enhanced resiliency to lighting disuniformities between different markers, without significant side effects: in fact, using this technique the absolute brightness of the marker support does not significantly affect how bright the marker appears in I_3, which makes the subsequent binarization much easier; in other words, I_3 measures the *ratio of intensities* between the dark marker parts and the surrounding white support, which is mostly unaffected by lighting.



Steps of the marker detection phase. From left to right: original image I_1; I_2; I_3; binarized I_3; connected components in I_3 meeting constraints in Phase 2.

As previously mentioned, the last operation is a binarization, which is trivially performed using a 0.5 threshold on I_3. After binarization, connected component labeling is performed. Each connected component (blob) is a binary mask M, and an element of the set of candidate markers CM.

Phase 2: Basic filtering of candidate markers and rough corner localization

Not every candidate marker detected in the previous phase correspond to an actual

marker in the image. In this phase, all elements of *CM* are subject to a number of tests in order to discard obvious misidentifications. Tests are performed in sequence, and as soon as a test is not met, the candidate is discarded. The order in which such tests are applied is defined such that the first tests are computationally cheap and very discriminative. Computationally expensive tests are therefore only applied only to a dramatically reduced number of candidates.

In particular, for each candidate marker we compute the following measures on its binary mask:

- **Size**, which should be compatible to the expected area of a marker; although we tolerate a wide variation in the measured marker area vs. the expected area (up to 6 times bigger or smaller), this test still usually discards most of the candidates in a crowded scene. Such a large variability is justified by distance differences we must account for, but especially by possible tilt in the markers, which can dramatically reduce the area of each mask.
- **Euler number**: the Euler number of a binary blob is related to the number of holes in the marker image, which is expected to be at least one. We may reasonably expect a larger number of holes, but we are not enforcing this in order to account for problems in the marker binarization, which are very frequent when the marker image is small or blurry. We also discard candidates with too many holes, which are sometimes generated by areas with high-frequency textures.
- **Solidity**, i.e. the fraction of the marker's convex hull which overlaps with the marker itself; as the marker projects to a convex quadrangle, we expect this to be close to 1.
- **Fraction of the marker image covered by holes**. We require this is less than 3/8 but larger than 0.25/8, as holes in the marker blob are expected to cover an area of about 1/8 of the marker size (assuming perfect binarization, which is not the case).

We implement a final test by looking for well-defined corners of the filled marker; when our routine returns four corners, the candidate is accepted, the corners are sorted in counter-clockwise order w.r.t. the marker barycenter, then passed to the subsequent phase, where such approximate corner localization will initialize the refinement of the marker localization in the image. If our routine detects a different number of corners, the marker is discarded before entering phase 3.

Localizing well-defined corners from the binary marker image is not trivial, because most simple algorithms would be fooled by small errors due to imprecise binarization. We provide two alternative algorithms:

Algorithm A

The barycenter of the filled marker is computed: we create a second float-valued image I_d where every pixel belonging to the marker image is associated to its distance from the barycenter, and pixels outside the marker are set to 0.

Corners are finally detected as maxima of I_d local to a circular neighborhood few pixels wide. This is implemented in practice by dilating I_d with a binary kernel

representing such circular neighborhood, then checking where the dilated version of I_d matches I_d itself.

In order to discard couples of nearby corners with the same distances, we add a very small random noise to Id before dilating it.

Algorithm B

The perimeter pixels of the filled marker image are stored in a list, then sets of 4 points belonging to the list are evaluated with the goal of finding the set defining a quadrangle with the largest area.

The perimeter pixels are recovered by applying a single dilation step to the mask and performing the subtraction of the result with the mask itself.

We then apply a naive algorithm to sort all the pixels evaluating their euclidean distances, in order to build a chain of edge pixels. This step has a computational complexity of *O(n^2)*, where *n* is the number of pixels belonging to the contour: this may create problems with badly-binarized markers which have a large number of edge pixels. Therefore, we apply the algorithm is after a basic filtering step, which assures that the amount of perimeter pixels can be efficiently handled, even in extreme conditions.

The optimization starts from an initial solution defined by taking four evenly distributed corners from the list. Each iteration is composed of the following operations:

- we consider three of the corners as fixed and moves the fourth corner forward in the list in order to optimize the area of the defined quadrilateral;
- once a maximum is found we repeat the step for each of the other corners;
- when all four corners have been evaluated we repeat the previous two steps by moving the corners backward and finally conclude the iteration with the best candidate set found.

Since the optimization function may present multiple local minima we apply a simulated annealing strategy which consists in perturbing the maximum of the current iteration with a gaussian noise. The noise changes the corners index in the ordered list by a small amount which is decreased at each iteration.

Algorithm A is more efficient for larger markers, but consistently provides a slightly lower success rate than algorithm B, especially with noisy masks. Therefore, we used approach B but left the former as an option.

Phase 3: Localization Refinement, Validation and Identification of candidate markers

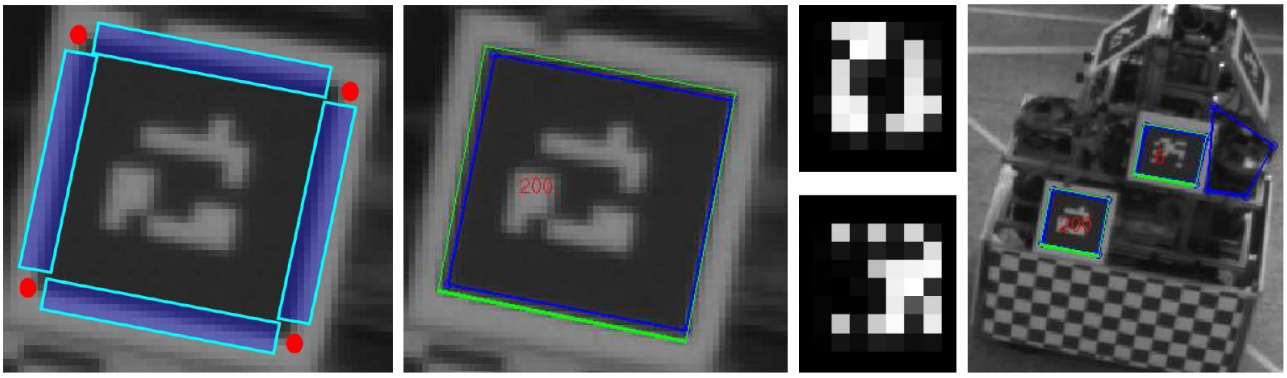
Once a candidate marker and its rough corner localization are known, the binary mask used for the previous processing is disregarded, and the original image data is

again taken into consideration in order to refine as much as possible the localization of the four corners.

Regardless on the specific techniques involved, localizing the marker corners from the neighboring pixels alone will provide a severely suboptimal accuracy; in this case, in fact, the amount of data (i.e. pixel intensity values) used for localizing each corner is limited. Moreover, due to defocus and motion blur, corners often appear smoothed, and are misplaced even by one or few pixels towards the center of the marker. This systematic error would reflect to a very noticeable systematic error in the marker 3D localization.

The technique we implemented relies instead on the straight edges connecting the marker corners. In assuming such edges straight, we disregard the effect of the camera radial distortion: we verified that this is harmless, as such edges have a limited extent in the image.

We first precisely localize those four lines by using data from all the pixels near their expected position (except those near the corners, which are deemed unreliable as previously introduced). The expected position of the edges is computed from the rough localization of the corners provided by the previous phases. This provides a precise and robust localization. Once the four lines belonging to the marker edges are defined, they are (algebraically) intersected in order to recover the *refined* corner points.



Details on marker localization refinement and ID decoding and validation. From left to right: marker with rough corners returned from Phase 2 (red dots): and localization areas for edges (blue); detected marker, with refined position (yellow) compared to rough localization from Phase 2 (blue); note that yellow square is external to blue one, as corner positions were initially displaced towards the marker center, then refined to the true, outer position.; such localization error would result in a 3D reconstruction error of more than 10cm. Marker ID matrices recovered and normalized; overview of video frame with recovered markers: note that highly slanted markers are not detected; also note late-rejected marker candidate on the right, whose localization couldn't be validated.

Any error in this process leads to a *late reject* of the candidate marker. One such possibility is the failure to find a well-fitting line to the edge in the image area where the marker border is expected. Late rejects are very limited in our final system, and amount to one every 10-20 frames.

Once the refined corner points are known they are exploited in order to rectify the

marker image and recover the binary matrix represented inside.

As we previously introduced, the 6x6 matrix is not centered in the marker, but instead displaced towards the upper-right corner by half a matrix cell. As the actual orientation of the marker is not yet known at this stage, directly recovering the 6x6 matrix is not an option. Instead, we look for a centered 7x7 matrix, and handle the marker rotation in the subsequent stages. The actual 6x6 ID matrix will be a submatrix of such 7x7 matrix.

The four corner points in the image are used in order to estimate an homography (projectivity) H. H transforms the refined corner coordinates to coordinates (0,0), (24,0),(24,24),(0,24) in the marker reference frame. H is then inverted in order to recover the image coordinates of the centerpoints of each cell of the 7x7 matrix centered w.r.t. the marker. Such points are finally sampled from the image.

Sampling from the image at fractional pixel coordinates is performed either by a linear combination of four neighboring pixels, or by just sampling the nearest neighbor. We did not notice any measurable difference in accuracy between these two methods: this can be explained as the noise levels in our datasets were way lower than the contrast between white and black marker cells, and a single marker cell often had a projection few pixels wide. In very noisy conditions with larger imaged markers, a more sophisticated technique can be implemented by appropriately weighing all the pixels (possibly fractionally) belonging to the cell projection; in practice, this may be easily implemented by appropriately using the imtransform function, transforming the whole marker to a 24x24 image with a custom resampler, then downsampling its central part to a 7x7 matrix.

The resulting 7x7 matrix is rescaled and normalized between 0 and 1. It is then compared to the ones associated with the expected marker IDs, including their three rotated versions, and matched to the one with the minimal distance; such distance is also required to be lower than a preset threshold.

Failure to find such match is another cause of *late rejection* of the candidate marker, as it is detected but not verified; in our experiments, this occurs in one every 30-40 frames. Instead, if the match is found, the marker is definitively accepted, its ID associated, and the four refined corners reordered consistently with the marker expected orientation.

Localizing the robot in 3D

For every input frame, the set of the detected markers is used in order to recover the 3D position of the robot.

First we compute the position of each corner of each detected marker in the frame of reference of the robot, by exploiting the known rototranslation between the robot frame of reference and the marker itself; in the previous steps, we have also precisely localized the the projection of each of these 3D points.

In short, if R markers are detected in a frame, we have $4*R$ image points (from a calibrated camera), paired with $4*R$ 3D points in the frame of reference of the robot.

Since the camera is calibrated, each of the image points can be backprojected to a viewing ray in world coordinates; our goal is to recover the rototranslation leading from the world frame of reference to the robot frame of reference, such that the 3D points project to the known image points (or, equivalently, such that every 3D point lies on the respective viewing ray). Such problem is just a different formulation of the well-known PNP problem (Perspective N-Point – in our case N=4*R), which is usually considered for the estimation of the extrinsic camera parameters from the images of N known 3D points.

The problem does not have a trivial solution, but it has been widely studied in the past due to its practical importance, especially for Augmented Realty applications: in fact, estimating the position and rotation of the camera with respect to the imaged scene is probably the most important geometrically-challenging problem in Augmented Reality systems.

Recent works in this field include [AP1,AP2,AP3]. Those approaches differ in their main approach to the problem, and each has its own peculiarities. For example, [AP2] is extremely efficient, as it challenges the problem with an innovative approach which does not require repeated iterations; [AP1] approaches the problem from an operational research perspective, and makes use of a specialized, highly-optimized library for optimization in convex cones. Both [AP1 and AP2] are O(N) algorithms, which however is not a fundamental requirement in our scenario as:

- we do not have strict efficiency requirements;
- we deal with relatively small problems, as no more than 4 markers (16 points) are usually visible at a time.

Our system allows one to use any of these in order to solve the position of the robot: the code is structured in a modular fashion, such that a single interface is provided for robot position reconstruction, and any of the algorithms can be used by simply switching an option. Moreover, our system separately executes the two main steps we are describing – marker localization and 3D reconstruction – as intermediate results are permanently stored. This allowed us to easily compare the different approaches without re-computing the position of the marker edges in the video frames.

A well-known issue regarding the PNP problem concerns planar point sets: in that case, in fact, most algebraic techniques incur in degeneracies. In practice, finding the position of a 3D target is usually still possible, but with much less precision, and expecially with much less robustness. In particular, the localization of planar markers has been dealt with in [AP4], where the following problem is formalized: the reprojection error of the given point set has several minima (sometimes very small) at very different poses of the target; in practice, most algorithms often return rototranslations whose rotational components are significantly wrong, sometimes up to 45-50 degrees, as those poses also have very low reprojection errors. In our setting, this problem occurs anytime we have to reconstruct the robot position from a single marker. Note that, even if only the rotational component of the marker position is wrong, the robot position may be reconstructed with a large error as the marker itself is not centered on the robot's frame of reference.

We have found the algorithm in [AP1] to be the most robust to this sort of error, up to

the point of being affected by it only in few cases, and only when the marker is very small (and therefore its corner points detected with a limited relative accuracy). Therefore, [AP1] is used by default in our system. Other techniques, especially [AP3], are instead very prone to this problem, and practically unuseable when a single marker is visible – which would also be an important problem for the mutual localization of the marker 3D positions on the robot frame, as shown in the following section.

Mutual localization of the marker 3D positions

Before the chosen approach can be applied, the relative positions and rotations of the markers on the robot frame must be known; or, in other words, the rototranslations leading from the robot's frame of reference to each marker's frame of reference must be known.

We implemented a technique for recovering these rototranslations, based on the fact that, as introduced in the preceding part of this Section, even a single marker can be usually localized in 3D, given that the suitable algorithm is used, and that high accuracy characterizes the detected image points and camera calibration data. Therefore, we created a number of images with an medium-resolution camera, whose intrinsic parameters are precisely known, but whose extrinsic parameters are not known as the camera is handheld and freely moved by the operator. A number of images has been taken of the robot with its markers, from a low distance and from several different angles.

For each frame, our system analyzes this data in order to localize the markers, as already shown. Frames where a single marker is visible are discarded. Else, the 3D position of each single detected marker is recovered.

In particular, we consider a frame of reference for each marker, centered at the center of the marker; the x and y axes aligned with the horizontal and vertical edges; and the z axis going towards the viewer.

Then, by using the algorithm in [AP1] we estimate the rototranslations leading from the camera frame of reference to the frame of reference of each marker. For each couple of markers M_1, M_2, we get two rototranslations T_1 and T_2, represented by 4x4 matrices. The relative position of M_2 w.r.t. M_1 is therefore found as a matrix $T_{12}=inv(T_2)*T_1$. Similarily, the inverse transformation is also computed as $T_{21}=inv(T_1)*T_2$.

We perform this operation on all the available frames, also considering the (lower-quality) data acquired with the external cameras. For each of the 15 couples of different markers on the robot, we finally gather all the respective transformations.

The results do not allow us to directly correlate a marker with all the other markers: in fact, for some couples no transformation is directly known; this happens when the two markers are neven seen at the same time in the same images – for example, when they lie at different sides of the robot. In these cases, the composition of two or more rototranslations must be computed, by considering a “bridge” marker.

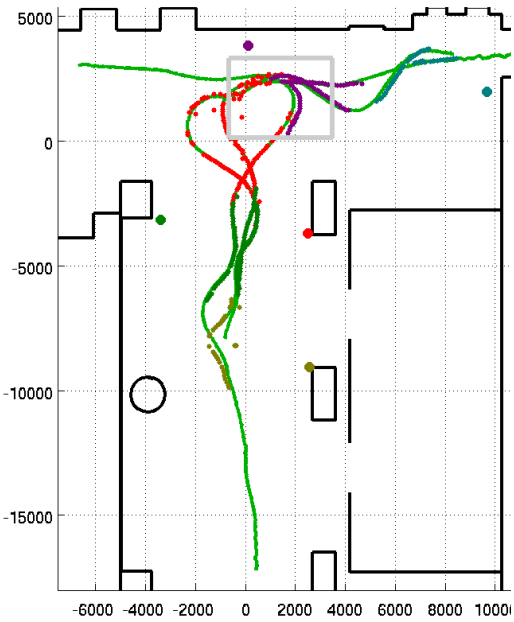
In particular, in our setting we consider one single marker as the robot frame of reference. Therefore, we need to compute the transformations leading to all the other markers, which is performed directly in 3 cases, and indirectly in two.

As many different rototranslations are detected in multiple frames for a single couple of markers, those need to be merged in order to recover the final one. A possibility of algebraically merging them exists, consisting in an averaging or median operation following by a conditioning of the matrix to make it an actual rototranslation. However, we also implemented a system for manually choosing the correct transformation given a 3D graphical display of the different possibilities. In particular, the user is presented with a 3D representation of the source marker and all the possible positions for the destination marker. In most cases, a "correct" position around which the cloud of options is aggregated is immediately visible, along with several outliers, possibly due to the robustness issues of the PNP problem with a planar target, as we highlighted in the preceding part of this Section. The user is allowed to choose one of the transformations found from the high-resolution dataset – which are usually in the center of the cloud – as the correct one, and that is saved as the actual transformation. This practically represents an user-assisted median operation.

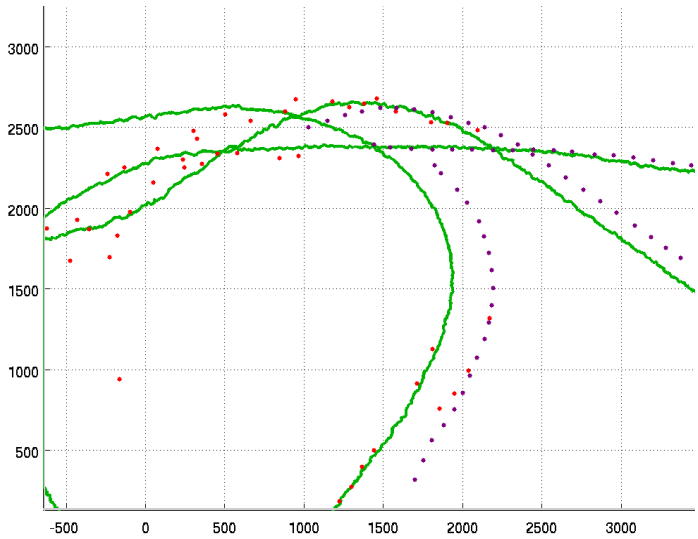
Experimental results

An assessment of both the systems used by RAWSEEDS for GT collection systems will be done in Section 4.2.7, and will include a comparison between their results. However, as a preliminary illustration of the performance of the GT-vision system, we provide here experimental results for the GT Vision reconstruction (colored dots) with respect to GT laser trajectory (green curve). The large colored circles in the maps define the camera positions. These results have been obtained by applying the GT-vision algorithms to the actual ground truth data collected during the acquisition of the Bicocca_2009-02-25a and Bicocca_2009-02-25b datasets. Multiple passages through the area covered by the GT-vision system occurred, hence the fact that more than one trajectory are present.

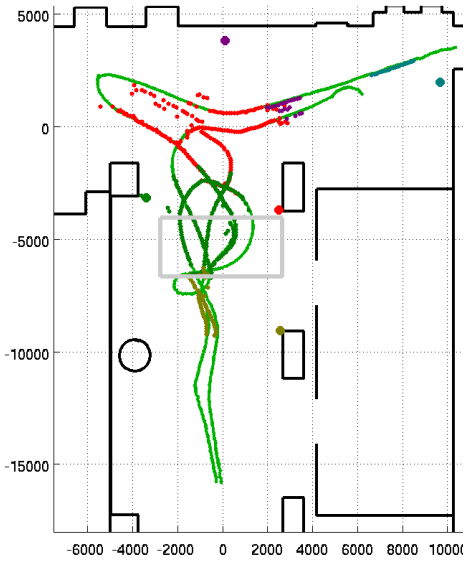
For additional details on the datasets, please see Part 1 of this Deliverable. Measurements are in millimeters.



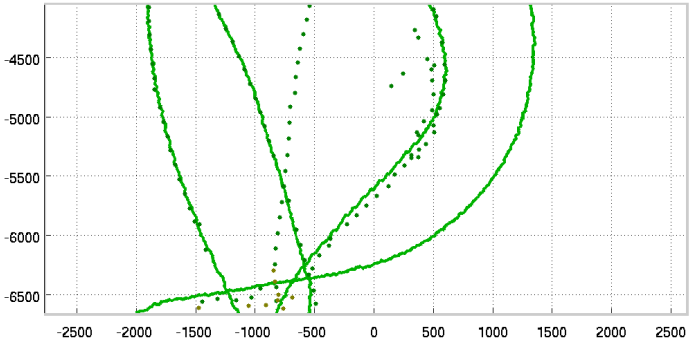
20090225a: overview



20090225a: detail of highlighted area



20090225b: overview



20090225b: detail of highlighted area

References

[AP1] Gerald Schweighofer, Axel Pinz, Global Optimal O(n) Solution to the PnP Problem for General Camera Models, in Proceedings of BMVC, 2008.

[AP2] Francesc Moreno-Noguer, Vincent Lepetit, Pascal Fua, Accurate Non-Iterative O(n) Solution to the PnP Problem, in Proceedings of ICCV, 2007.

[AP3] Chien-Ping Lu et. al., Fast and Globally Convergent Pose Estimation from Video Images, Transaction on Pattern Analysis and Machine intelligence, 1999.

[AP4] Robust Pose Estimation of a Planar Target, Schweighofer, Pinz, Graz uniTech.

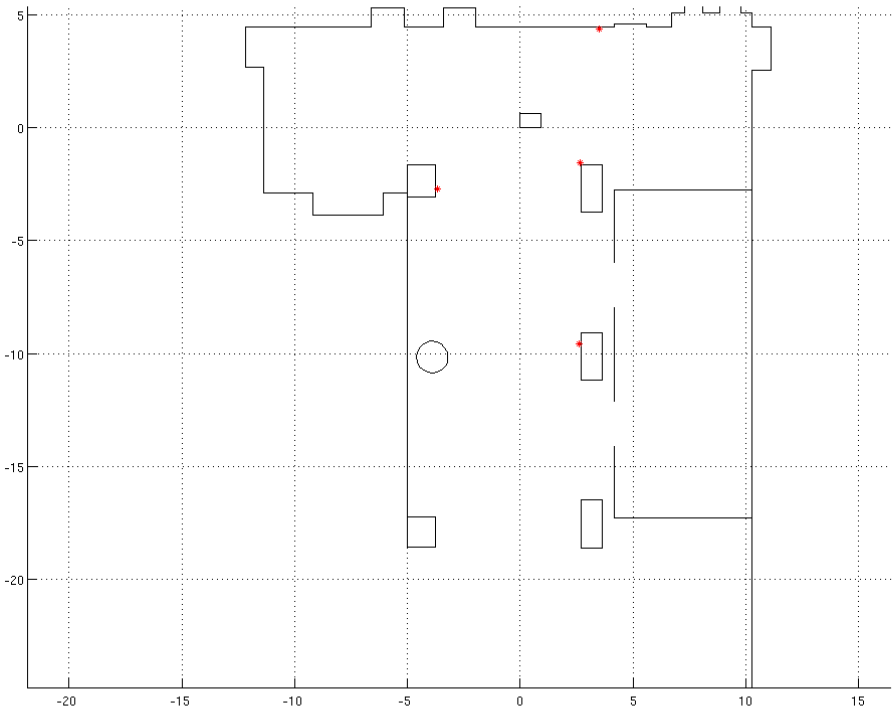
4.1.2 Laser-based GT

In this section we describe the procedures followed to generate the laser-based ground truth data (or GT-laser).

We can subdivide the whole process in three steps:

1. establish the relative positions between laser scanners;
2. collect the data;
3. estimate the position of the robot in the environment.

In the following figure we present an image extracted from the executive drawing of the explored environment where we collected the ground truth data. For doing so, we placed four laser scanners in the four red points shown in the map.



We chose the positions of these sensors in order to cover the same area covered by the GT-vision system described in the preceding Section.

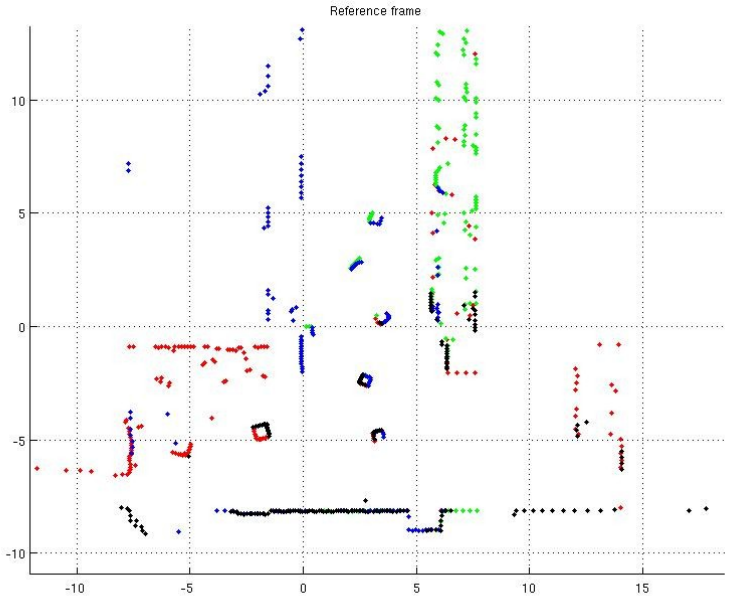
As a preliminary step, we estimated the position of each laser scanner with respect to the others by correlating their output. In fact, successive computations need to have all the data in the same reference system, therefore the relative positions between the sensors have to be known. To aid correlation, we performed this initial alignment after having inserted several objects in the field of views of the laser range scanners.

The following figure shows a panoramic image of the area covered by the ground truth collection systems, including the objects used to aid correlation. The four laser

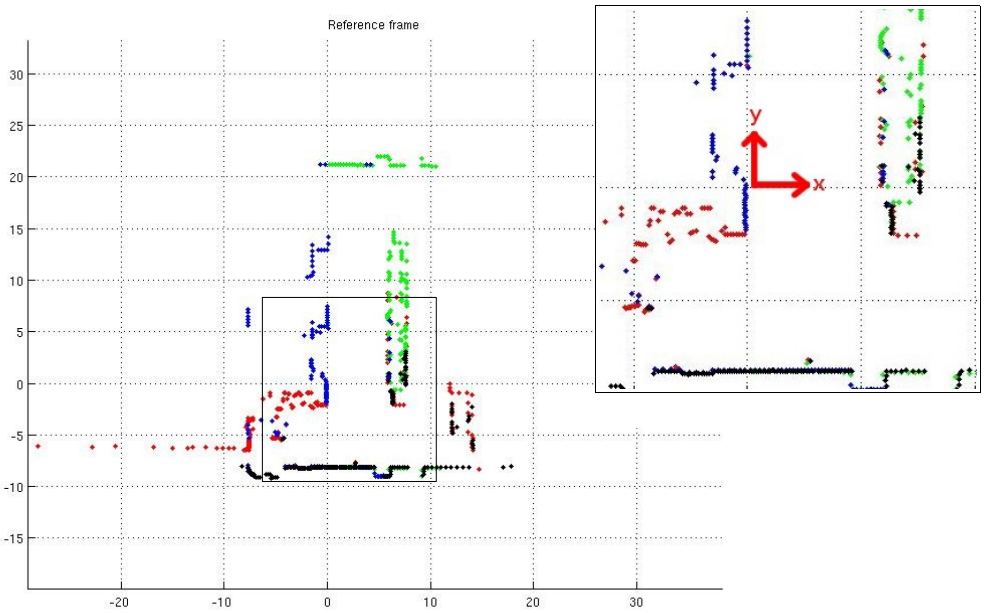
range scanners (Sick LMS200, blue) and three of the four poles supporting the cameras of the GT-vision system are visible (one of the Sick devices appears to be very distant on the background due to image perspective).

To correlate the scans from the different sensors, scans from each of them have been acquired. The results are shown by the following figure. Knowing the shape of the objects inserted in the environment, we can align these different scans to recover the correct position of one with respect to the other. This can then be used to to obtain the roto-translation between the sensors with respect to a single reference system.

Once we know the position of each laser, we can obtain each scan with respect to the same reference system. The following two figures show two examples of superimposed scans from all the sensors, first with the initial alignment setup and then in a typical situation.

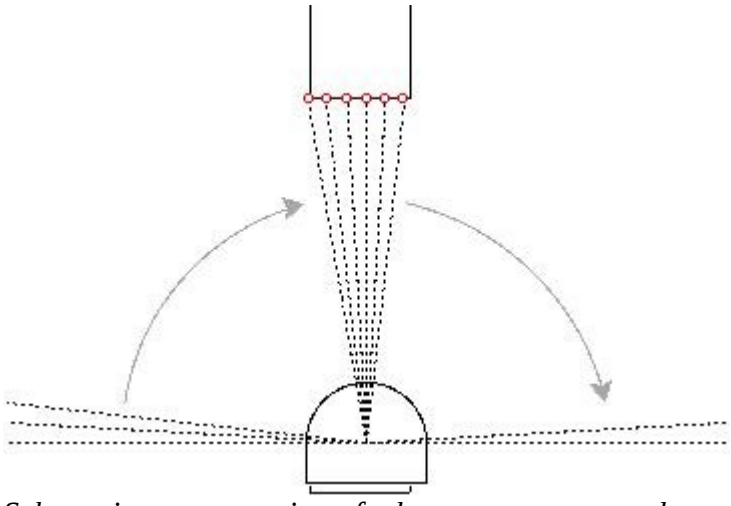


Aligned laser scans (each laser scan is represented by a different color.). Please note the shape of the objects used to align them with respect to a single reference system.



Alignment of the laser scans with respect to a single reference system (in red). Each laser scan is represented by a different color. The square on the right is a magnification of that on the left.

Now we will proceed to illustrate the actual process leading to the GT-laser data. It is therefore useful to remind the way the data has been collected. Each laser range scanner performs a set of 180 measurements at 75Hz, 1-degree spaced. Each measurement represents the distance between the laser and the first obstacle in that direction.



Schematic representation of a laser scanner at work.

The ICP algorithm

Whenever a set of four scans (one for each laser range scanner) is collected, the ICP (Iterative Closest Point) algorithm is executed on these data in order to perform their alignment. In particular, our implementation of the ICP algorithm performs an estimate of the robot position in the environment, assuming that laser data are already aligned, to recover the position of the robot inside the environment. In this section we describe this algorithm in its general form; we will present a pseudo-code of the real implementation later, while describing the overall GT-laser algorithm.

The ICP algorithm is an iterative algorithm developed to align clouds of points. The output of the algorithm is the roto-translation between these two clouds. In our case the clouds of points are defined as

- the entire scan obtained by fusing the laser scanner data;
- the shape model of the robot (formed of a set of points along the contour of the robot).

The algorithm iteratively estimates the transformation between these two raw scans. It works in three steps:

1. perform data-association (i.e., establish correspondences between the points belonging to two scans);
2. estimate the rigid transformation that best align the first scan onto the second;
3. apply this transformation to the two point clouds;
4. repeat these three steps until convergence is reached.

This algorithm works well when a good initial rigid transformation is given. In our case, we have four sets of points obtained by the four laser scanners and the shape model of the robot. The initial position is given by the previous robot position (the robot velocity is quite small with respect to the acquisition time).

Generation of the GT-laser data

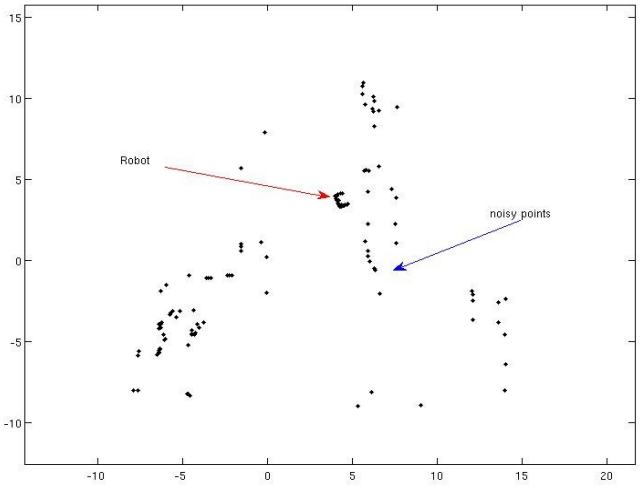
We will now proceed to describe the algorithm used for the generation of the GT-laser data. For ease of description it will be decomposed into four steps: (i) scans composition; (ii) filtering; (iii) application of the ICP algorithm; (iv) transformation refinement. These steps will be described in the following part of this section.

Scans composition

This step allows to obtain a single scan by joining the four laser scans. The scans coming from the four sensors are aligned by using the roto-translations computed by the previous steps. Output of this function is a single scan containing all the laser scans data in the same reference system.

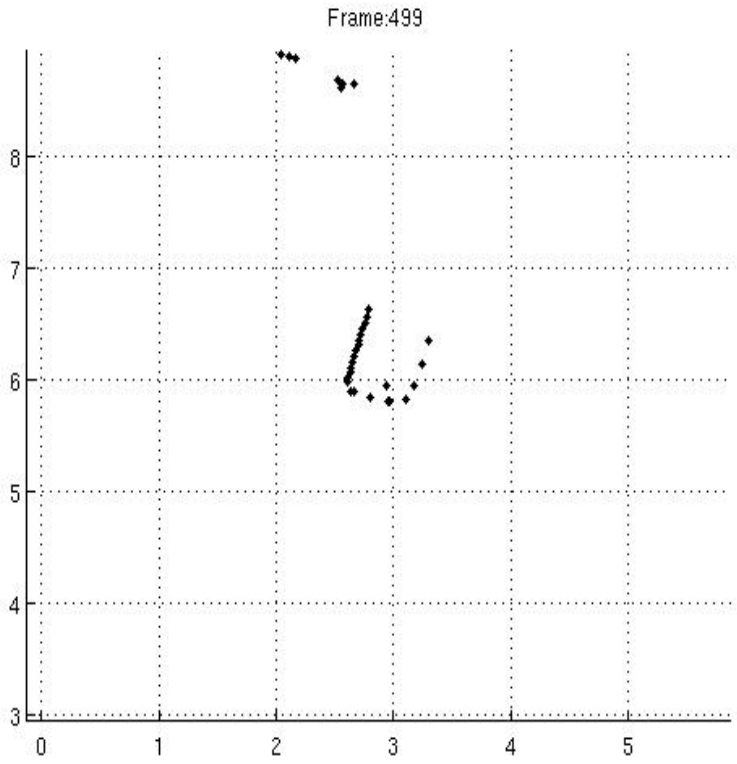
Filtering

The overall new scan is then filtered in order to remove points we are not interested in: noisy points and static environmental points. In fact, only robot points are necessary to establish the position of it in the environment. Starting from a scan of the whole environment without any other object, the algorithm removes point in the vicinity of these environmental points. The procedure is very similar to the background subtraction method in imaging. We remove the static points that belong to the “background” comparing a given scan with the reference one. Doing so, we maintain only noisy points and robot points. An example of this two different classes of measurements is shown by the following figure.



Given that we possess the estimate of the previous robot position in the environment, we can then check around it to find further noisy points. This step is performed by defining an area around the robot where that the robot cannot move out of in a single step: points outside of it will be considered noisy points and they will be removed.

The following figure shows an example of the application of such technique.

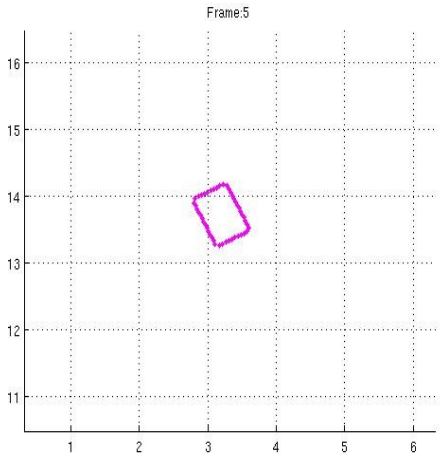


Resulting scan after the filtering processes (some noisy points remain due to rough thresholds in the example).

Application of the ICP algorithm

At this point, the ICP algorithm is applied to the filtered data. We used a classical implementation of the algorithm, adapted to our particular problem. The input parameters of this procedure are: the robot shape model, the previous robot position, the filtered scan and two thresholds (the meaning of these two thresholds will be explained in the following).

The robot shape, shown in the following figure, represents the object to track in the scan. It has to be roto-translated with respect to the previous estimate of robot position to help the algorithm to converge at the right solution.



The filtered scan is obtained by the previous step and represents the current measurements of the laser scanner (see figure 3). This is the pseudo-code of the ICP procedure used by the GT-laser ground truth collection system of RAWSEEDS:

Input:

MODEL robot shape model

DATA current laser scan

THRESHOLD_1 maximum distance between a MODEL point and a DATA point

THRESHOLD_2 error variation threshold for stop iterations

Output:

TR rotation matrix representing the estimate of the rotation between the previous robot position and the current robot position

TT translation vector representing the estimate of the translation between the previous robot position and the current robot position

```
do
    oldolderror=olderror;
    olderror = ERROR;
    %find the closest point
    ERROR = icp_closest ( MODEL, DATA, THRESHOLD_1 );
    %compute the roto-translation (TR,TT)
    icp_transformation ( );
    %apply the rototranslation to the points
    DATA = TR * DATA;
    DATA = DATA + TT;
    %check if the ERROR is above a threshold or not
while (abs ( olderror – ERROR ) > THRESHOLDS_2 )
```

Input:

MODEL robot shape model

DATA current laser scan

THRESHOLD_1 maximum distance between a MODEL point and a DATA point

Output:

ICLOSEST vector of indexes. Each record represents a MODEL point. Each entry maintains the index of the DATA point associated with that MODEL point.

```
icp_closest ( MODEL, DATA, THRESHOLD_1)
   md = sizeof ( MODEL );
   mm = sizeof ( DATA );
   ERROR=0;
   for id = 1 : md
      dist = Inf;
      for im = 1 : mm
         dista = norm ( MODEL (im) - DATA (id) );
         if dista < dist &  dista<THRESHOLD_1
                  iclosest( id ) = im;
             dist = dista;
         end
      end
      if (dist <> Inf)
         ERROR=ERROR + err (dist, id );
      end;
   end
```

This procedure performs the data-association between the MODEL points and the DATA points. For each MODEL point we associate the closest DATA point, if it exists. A threshold decides if this point exists or not with respect to its distance from the MODEL point. If the point is too far from the MODEL point it will be not associated. This distance is based on the Euclidean distance.

Also the ERROR is computed. This variable represents the sum of the all distances between the MODEL points and the associated DATA points.

Input:

MODEL robot shape model

DATA current laser scan

ICLOSEST vector of indexes representing the associations between MODEL and DATA points

Output:

TR rotation matrix representing the estimate of the rotation between the previous robot position and the current robot position

TT translation vector representing the estimate of the translation between the previous robot position and the current robot position

icp_transformation (MODEL, DATA, ICLOSEST)

%take only the associated DATA and MODEL points

for i = 1 : sizeof (ICLOSEST)

if (ICLOSEST (i) <> 0)

DATA_red (c) = DATA (i);

MODEL_red (c) = MODEL(ICLOSEST (i));

c=c+1;

end;

end;

%sort with respect the distances

V = *sum* ((DATA_red – MODEL_red) .^ 2);

in = *sort* (V);

%take the 95% of the closest points

num = round (0.95 * sizeof (DATA_red));

ind = in (1 : num);

%estimate the rototranslation

med = mean (DATA_red (ind));

mem = mean (MODEL_red (ind));

A = DATA_red (ind) - med;

B = MODEL_red (ind) - mem;

%compute the rotation

[U, S, V] = *svd* (B * A');

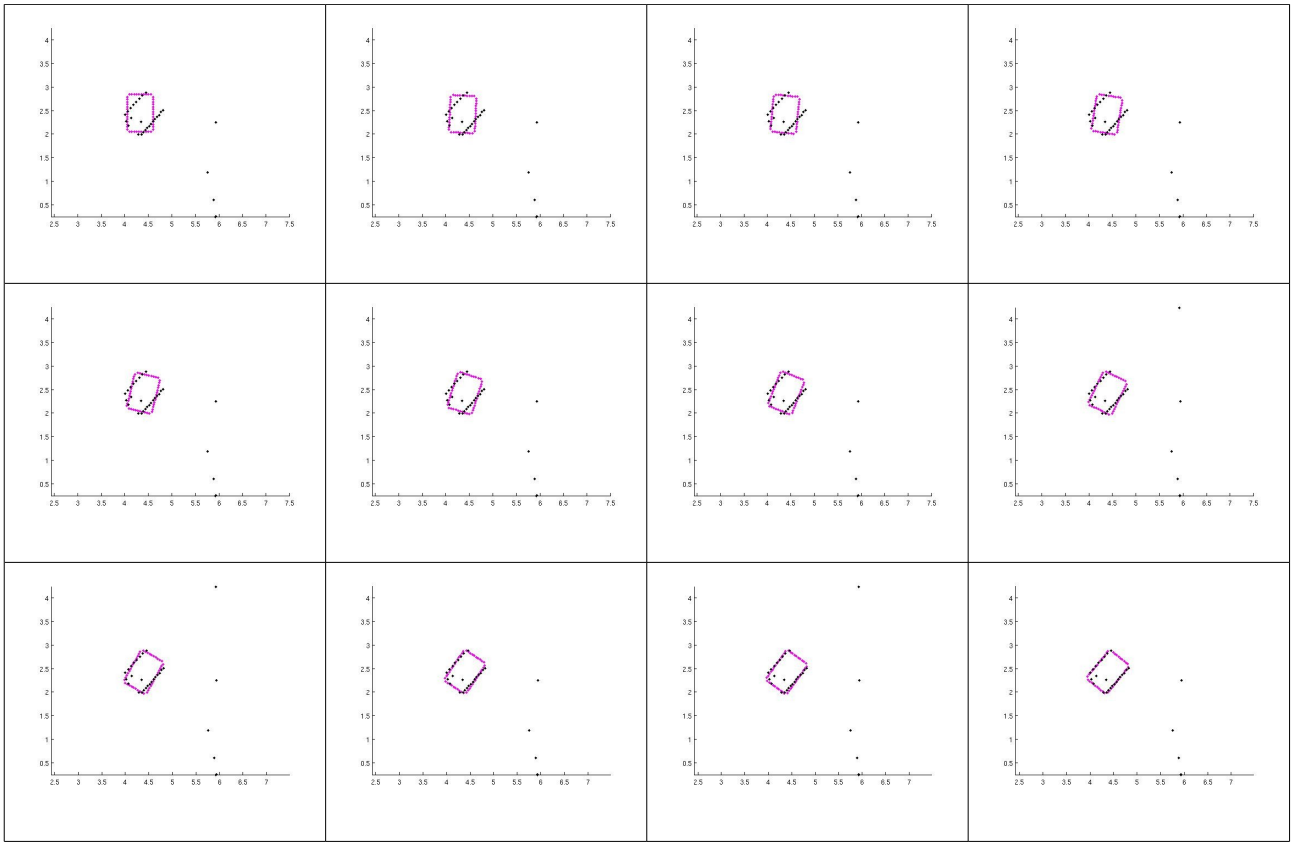
U = U * *det*(U * V');

R = U * V';

```
% Compute the translation
T=( mem – R * med );
TR = R * TR;
TT = R * TT + T;
```

This procedure estimates the roto-translation between the previous robot pose and the current robot pose basing on the associations performed in the previous step. First of all, we sort all association indexes with respect their distance from the associated MODEL points. Then, we take only the first 95% of closest points. This is an heuristic threshold that allows to remove the farthest points and so reducing the possibility to include noisy points in the estimate process. Finally, these points will be used to compute the roto-translation between two sets of points (MODEL and DATA).

The following figure shows an alignment process performed by the ICP algorithm.

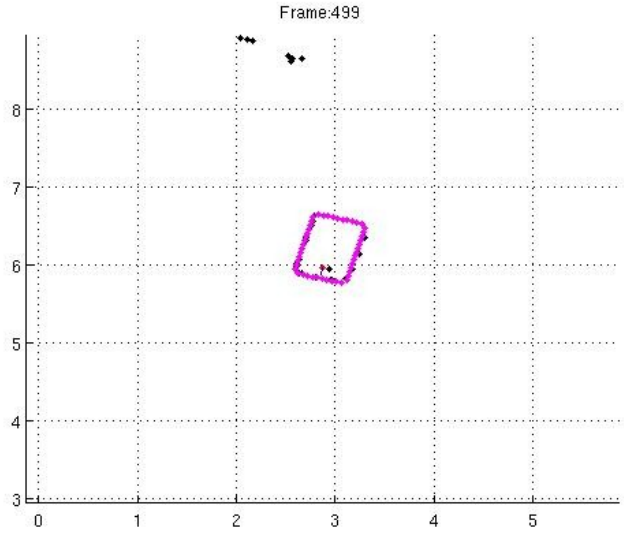


Sequence of alignments performed by the ICP algorithm: in black the laser scan data, in magenta the robot shape model

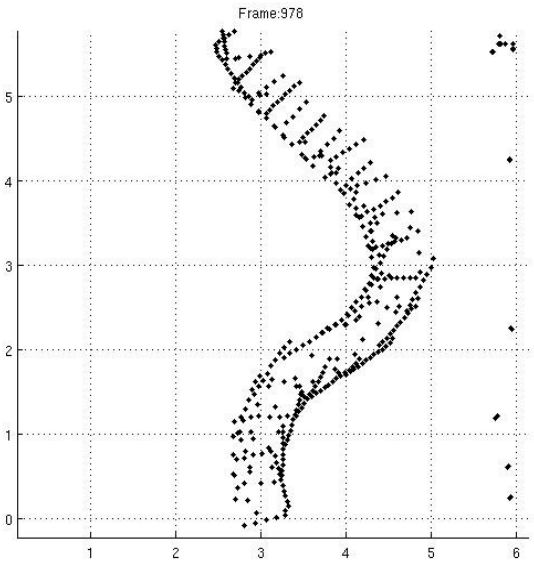
Transformation refinement

The resulting roto-translation (current robot pose with respect the previous one) is

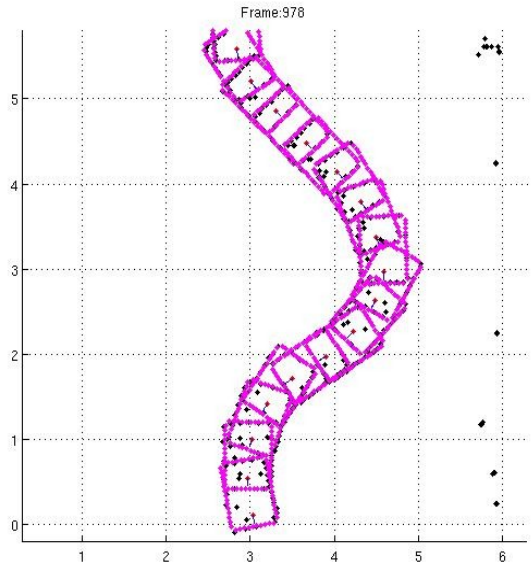
then composed with the previous estimate to compute the pose the robot with respect to the reference system of the map.



Robot shape superimposed with filtered scan data. The robot position is obtained by using the ICP algorithm.



Filtered laser scans data.



Estimated robot positions (in magenta) by using the filtered scan data.

An assessment of the performance of the GT-laser system, and a comparison with those of the GT-vision system, will be done in Section 4.2.

4.1.3 Synchronization between robot data and ground truth

Introduction

The synchronization between the Robot and the GT system used the IEEE 1588 Precision Time Protocol (http://ieee1588.nist.gov/), as implemented by the ptpd software (version 1.0rc1, available through http://ptpd.sourceforge.net/).

The setup consisted in: a Master Clock (MC) onboard the robot, i.e., the PCBrick01 machine; a Slave Clock (SC), i.e., the PC dedicated to acquisition of all GT data (both laser- and vision-based); and a wireless IEEE 802.11g link between them, obtained by radio connection between a wireless USB network adapter connected to the SC and the wireless router/switch on board of the robot. As the PC dedicated to ground truth collection was also the SC, synchronizing the MC with the SC also meant synchronizing all the systems of the robot with the GT system.

Wireless links are scarcely reliable, and have variable latency; in Rawseeds' case, the variance of the latency of the wireless link/USB adapter was much greater than the latency associated to a system composed of cable link and ethernet adapter. For this reason, during the acquisition some problems emerged: such problems were solved by performing a post-synchronization phase on the data acquired. The following part of this section describes the problems and the adopted solution.

In the PTP system, a *clock servo*[1] tries to minimize the offset between the MC and the SC by adjusting the SC tick rate. Even if the clock servo filters this offset to remove jitter, a loss of link between the MC and the SC results in a constant drift of the SC with respect to the MC. This is probably due to the fact that the noise in the offset (see Listing 1) introduced by the wireless link is too strong for the filter to handle, so when the link goes down the clock continues to drift (as there's no feedback from the MC to correct such drift). Differently from previous experiences with off-robot device synchronization, carried on in a single confined room without loss of wireless connectivity (please see Additional Deliverable AD2.3 for details), these loss of link are common when the Robot moves within a large building, where the signal path is often obstructed by massive structures.

The offsets between MC and SC, as measured by ptpd, were recorded for reference purposes in the post-synchronization phase and are showed in Listing 1. It must be noted that these offsets cannot be used for a post-synchronization, as they are contaminated with the cited strong noise. The following subsection will describe the actual post-synchronization procedure adopted.

1 The clock servo is a PI controller. Please see the documentation of ptpd for further details.

Listing 1

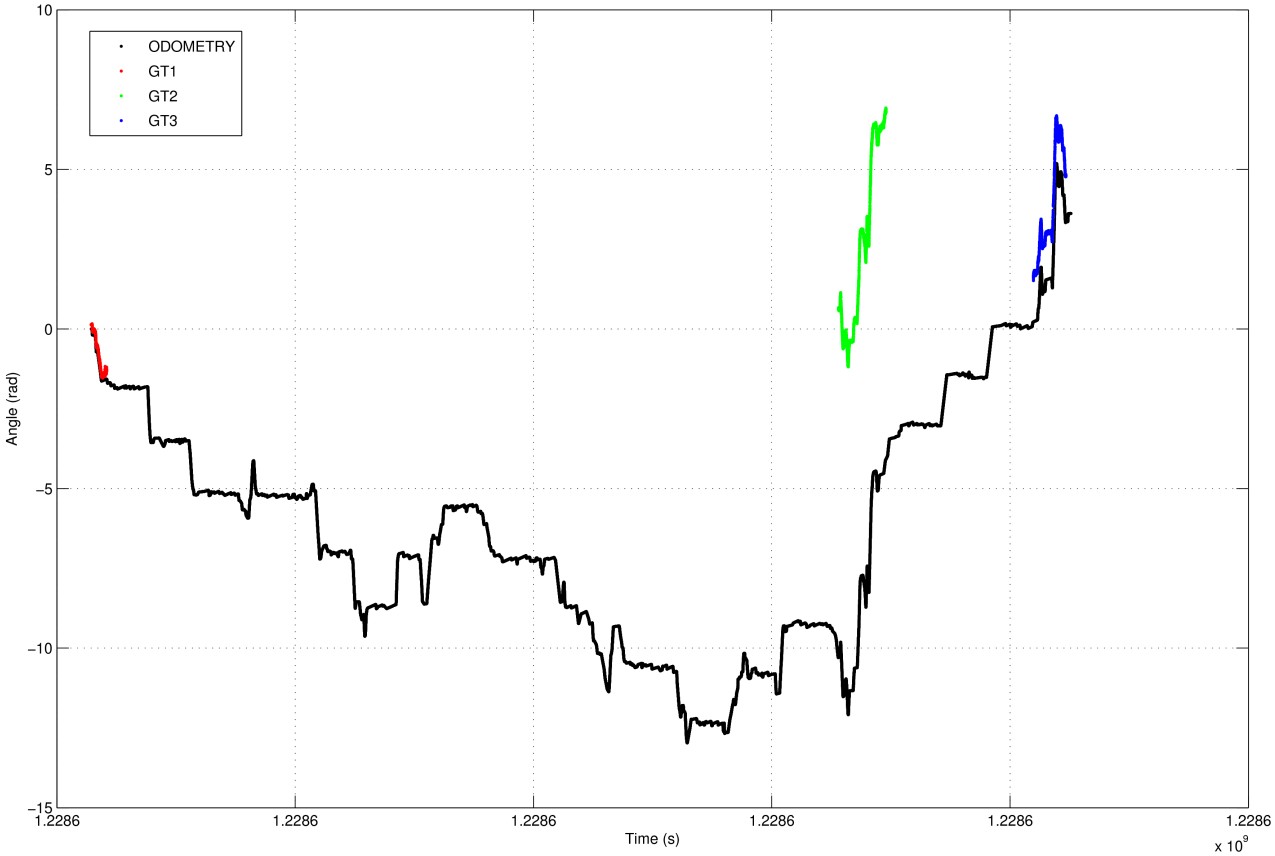
```
offset from master:                0s      591500ns
observed drift:    -255389
offset from master:                0s     -894500ns
observed drift:    -256283
offset from master:                0s     -821000ns
observed drift:    -257104
offset from master:                0s     -779500ns
observed drift:    -257883
offset from master:                0s     -955000ns
observed drift:    -258838
offset from master:                0s     -928000ns
observed drift:    -259766
offset from master:                0s     -831000ns
observed drift:    -260597
offset from master:                0s     -743500ns
observed drift:    -261340
offset from master:                0s      842500ns
observed drift:    -260498
offset from master:                0s      725000ns
observed drift:    -259773
offset from master:                0s    -1046000ns
observed drift:    -260819
offset from master:                0s    -1170500ns
observed drift:    -261989
offset from master:                0s    -1130000ns
observed drift:    -263119
offset from master:                0s     -982000ns
observed drift:    -264101
offset from master:                0s     -855500ns
observed drift:    -264956
offset from master:                0s     -775000ns
observed drift:    -265731
offset from master:                0s     -682500ns
observed drift:    -266413
offset from master:                0s     -605500ns
observed drift:    -267018
offset from master:                0s     -607500ns
observed drift:    -267625
offset from master:                0s       49000ns
observed drift:    -267576
offset from master:                0s    10556500ns
observed drift:    -257020
offset from master:                0s     9468500ns
observed drift:    -247552
offset from master:                0s    -2440500ns
observed drift:    -249992
offset from master:                0s    -3052500ns
observed drift:    -253044
offset from master:                0s    -2977000ns
observed drift:    -256021
offset from master:                0s    -3076000ns
observed drift:    -259097
offset from master:                0s    -2551500ns
observed drift:    -261648
offset from master:                0s    -2201000ns
observed drift:    -263849
offset from master:                0s    -1949500ns
```

Excerpt from a ptpd log

Synchronization procedure

To synchronize the GT system with the robot, the correlation that exists between two different streams (GT and odometry, in this case) has been exploited. In particular, the rotational part of the pose of the robot was used, as it is independent from the translation between the odometric and the GT reference frames (World), and a single degree of freedom divides and relates the 2 signals.

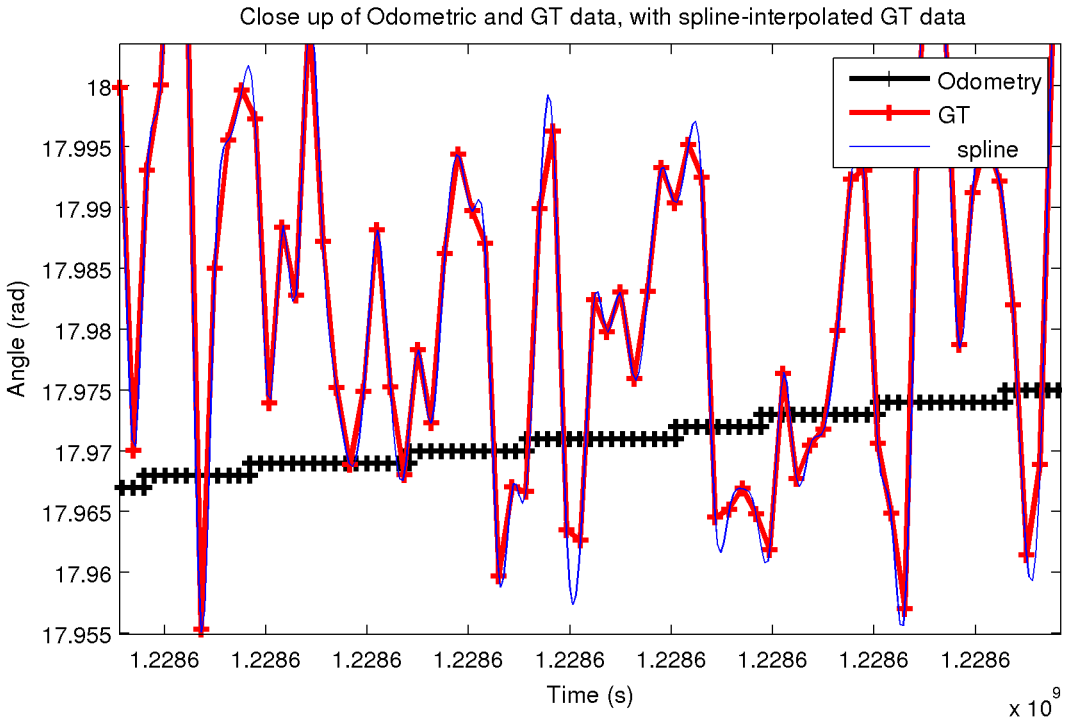
The angle estimated by the odometric system of the robot (θ_R) was compared with the angle measured by the laser GT system (θ_L), as shown in the following figure.



θR and θL for three different portions of GT data (GT1, GT2, GT3).

Then, for each segment of ground truth data (these were acquired only when the robot was in the area covered by the GT system), the two streams were aligned to recover the time offset and clock drift. For this, we chose to use the GT-laser data instead of those provided by the GT-vision system because of the higher sampling rate of the sensors (75 Hz vs. 5 Hz) and the larger area covered, two characteristics that positively influence the quality of the synchronization results. The alignment consisted in a minimization of the Sum of Squared Errors (SSE) between the GT and the odometric data, i.e., a least squares approach was taken. The minimization parameters were: Δt (time offset), d (drift) and $\Delta\theta$ (angle offset, i.e., the rotation between the Odometric and World frames).

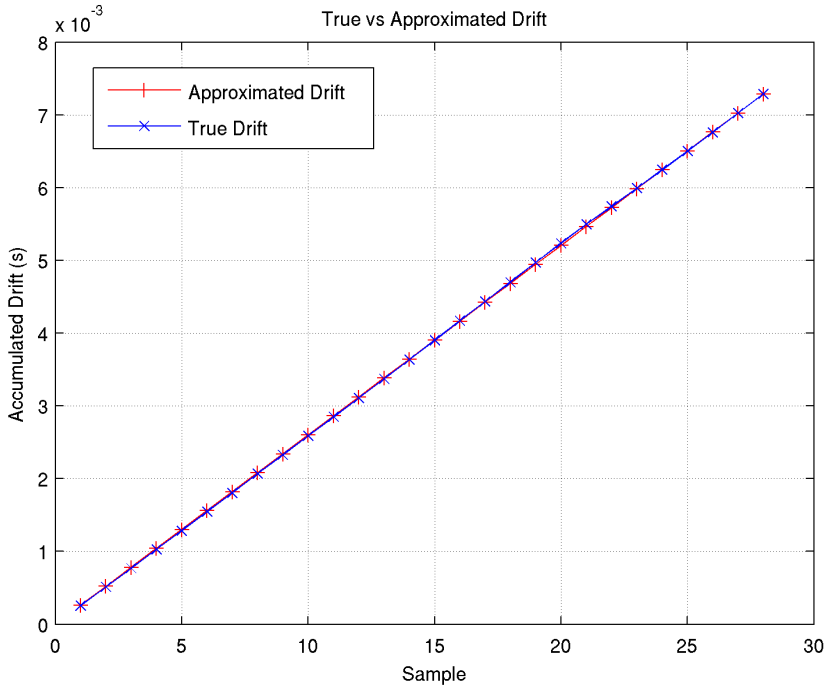
The GT data was taken as a reference, while the odometric data was translated (Δt) and stretched (d) in time, and shifted ($\Delta\theta$) in angle. This proved to be the best choice, as the GT data was noisy, and its sampling rate was less than the one of the odometric data: the interpolation needed to compute the SSE would have resulted in a poor approximation of the original signal (please see the following figure for a close up).

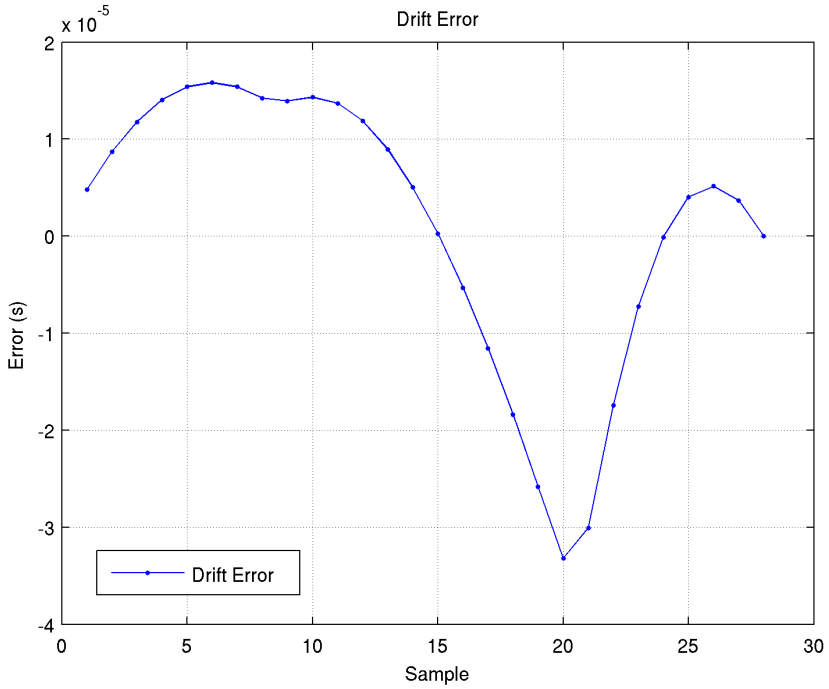


The interpolation problem

This interpolation was necessary because calculation of the error between the two streams required that the signals were sampled at the same instants. In other words, it is the sum of squared errors between isochronous samples that had to be minimized.

We made an assumption, here, that requires an explanation: we assumed that a drift exists, and that this drift is linear. This is just a hypothesis, probably simplistic: when the wireless link is active, ptpd constantly tries to minimize the offset, as explained. The variations in the speed of the clock, used to achieve its goal, make the clock drift linearly only between 2 successive ptpd updates, that occur every 2 seconds. However, on a global scale (i.e., over a period that can be considered long w.r.t. the ptpd intervention interval) this piecewise linear drift can be approximated with a single linear drift: in fact, the error is usually far below 1ms, so it's negligible for RAWSEEDS' purposes. The following figure shows and compares the drift as measured by ptpd and its linear approximation.





Example of linear approximation of drift as measured by ptpd on a 56 seconds time period (top) and the associated error (bottom).

For the minimization we used MATLAB's *Genetic Algorithm and Direct Search Toolbox* patternsearch function, which from our tests gave better results than MATLAB's fminsearch or MATLAB's *Optimization Toolbox* lsqnonlin, to cite a few. In particular it provides a stable minimum, independent from the starting point. Conversely, the other minimization algorithms we tested misbehaved badly if the starting point was not accurately set, having a very high sensitivity to the parameters, and particularly to $\Delta\theta$. Even a small difference in the starting point made the minimization reach a final point that was far away from the optimum. This was a clear suggestion for the use of a derivative-free minimization algorithm. We tried to take a two-step approach, in which an initial guess, provided by an exhaustive search on a fine pitched mesh in the parameter space, was followed by a standard minimization algorithm, like the two previously cited; however, the results were unsatisfactory. A change in the mesh pitch resulted in a different starting point, and then in a different final point, for the motivations just explained. Even an iterative approach, in which $\Delta\theta$ is optimized independently from Δt and d, gave results that were not enough close to the final point.

The solution was constrained in a small volume in the parameter space. This was necessary to avoid unrealistic solutions (e.g.: the whole odometric signal collapsed in a single point by a very large d, that would gave a SSE of 0), and to exclude many local minima. The boundaries were chosen so that the search volume included all plausible values, and are listed in the following table.

Parameter	***Lower Bound***	***Upper Bound***	***unit***
$\Delta\theta$	-3.14	3.14	rad
Δt	-1	1	s
D	0.99	1.01	%

The results of the minimization algorithm were analyzed visually, in order to exclude evidently wrong solutions and, in such cases, to correct the problems that led to such solutions. We focused on the parts of the signal where the angle rapidly changes, like the one depicted in Figure 4, being those parts the ones that convey most of the time information (i.e., where the PSD of the signal has a strong component at the higher frequencies).

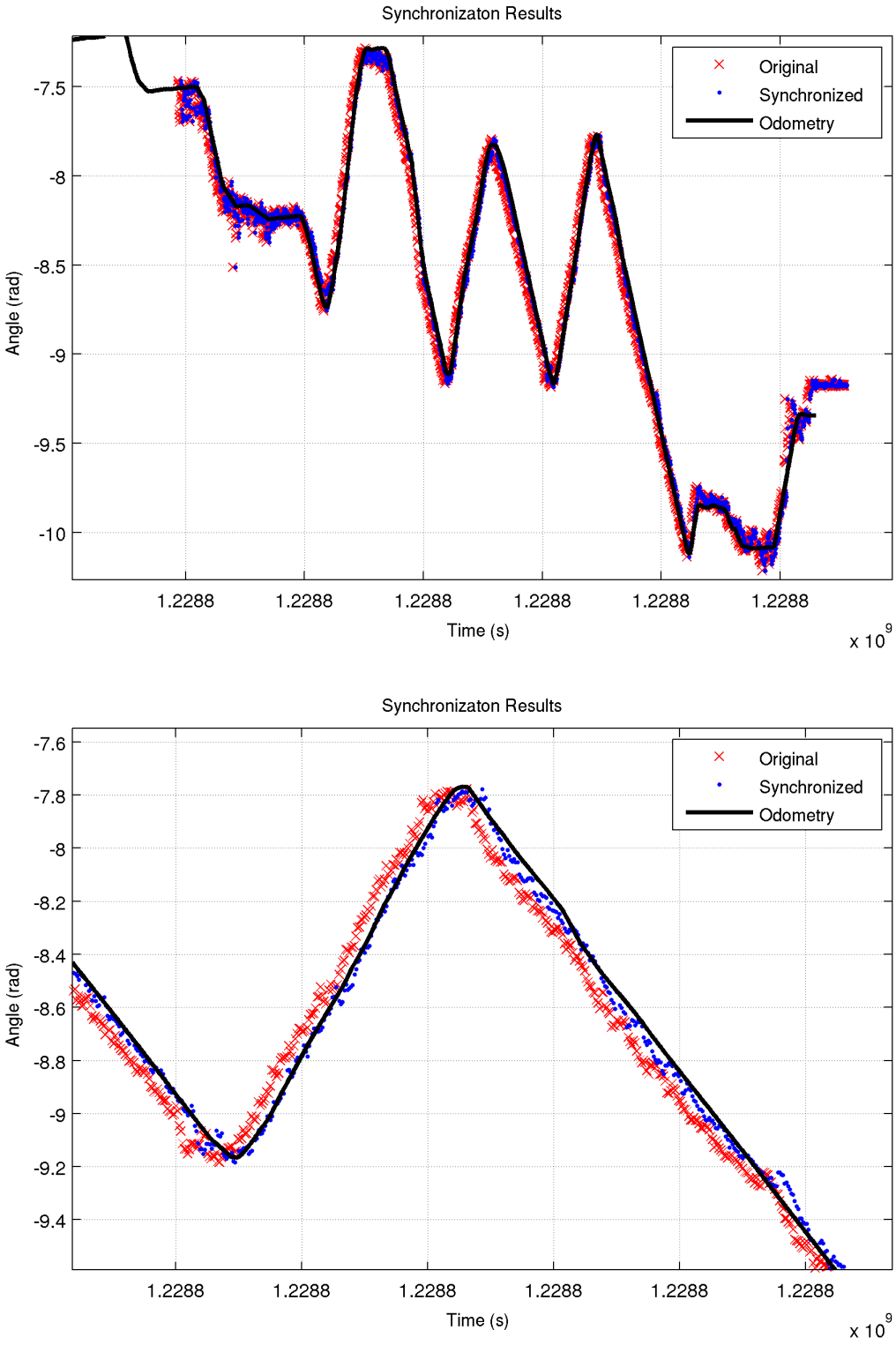


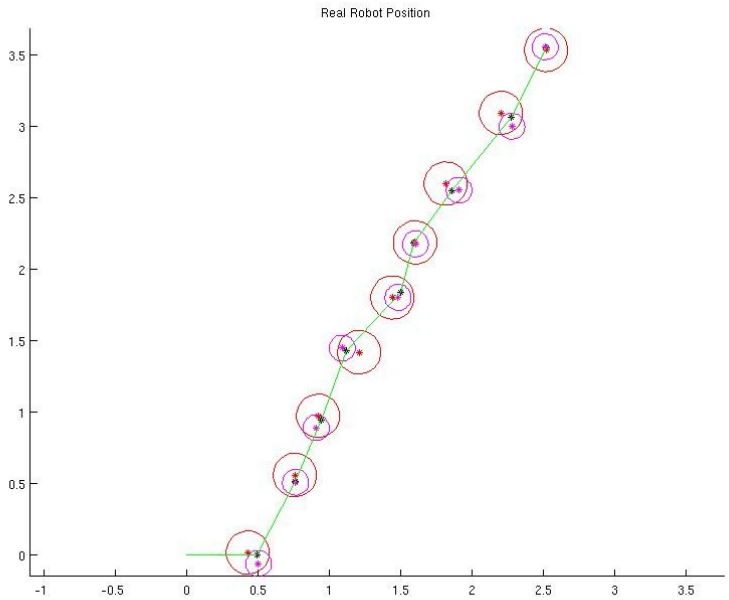
Figure 4: An example of synchronization results. A whole GT segment (i.e., the complete segment of ground truth data corresponding to a passage of the robot in the area covered by the GT collection systems) (top), and a close up (bottom).

A comparison between the recorded reference time offsets and the results obtained in the post synchronization showed that the post-synchronization method described above yielded satisfactory results. The offset as measured by ptpd were close to our estimated offset. As previously said, the recorded offsets can only be used as a reference, and in no way they can be considered exact.

4.1.4 Ground truth fusion

The purpose of this section is to describe the process used to generate a single stream of ground truth data from the information provided by the GT-laser, the GT-vision and the odometry sensor data. This is made possible by fusing all these streams into one, in order to obtain a more reliable and more usable information about the real position of the robot in the environment.

Supposing that the uncertainty associated with the measures given by the GT-laser, the GT-vision and the odometry sensor are near to be Gaussian, then we can use a classical statistical approach, as the Kalman Filter approach, to improve the correctness of the GT results. Consequently, it is necessary to know the covariance of the error associated with each measurement. This information is gathered from the validation process (please see Section 4.2 for details). Analyzing the measurements, obtained by the GT sensors, on the set of 23 robot positions used in validation, we can estimate the covariance of the error associated with each device. The Extended Kalman Smoother will make use of this information to obtain a reliable estimate of the GT stream.



Ground truth with the associated Gaussian uncertainty (+/- 3 sigma): GT-laser in red, GT-vision in magenta and true position in black. Green lines represents the path of the robot.

In fact, fusing two or more streams of information, we can reduce the uncertainties on the measures, obtaining more reliable estimates. Moreover, we are able to provide a single GT stream to which the output of RAWSEEDS' Benchmarking Solutions can be comparated with. Such result is reached by using an Extended Kalman Smoother. Unlike the Extended Kalman Filter, it allows to improve the estimate of the state by

using the whole information available (both past and future). This algorithm is composed by two phases: the first executes a forward filtering (it is equivalent to a classical Extended Kalman Filter), the second is a backward recursion which allows to smooth the past estimates integrating the future ones.

We now describe the equations used in both these phases.

Filtering

Suppose we have a nonlinear systems in the following form:

$$x(t+1) = f(t)(x(t) + u(t+1)) + w(t)$$

$$y(t+1) = h(t+1)(x(t+1)) + v(t+1)$$

where f is the state transition function, u is the control function, w is the process noise, h is the measurement equation and v is the measurement noise. Now, we suppose that v and w are random white noises with covariance Q and R respectively.

In our case, x(t) represents the robot position in the environment w.r.t. the world reference frame, f computes the transformation composition between the robot position at time t and the movement of the robot at time t+1 (we use the odometry measure as the control u), obtaining the new robot position at time t+1.

The function h is the identity matrix since we have directly the measures of the robot position in the environment from the GT streams.

The Extended Kalman Filter approach approximates the nonlinear system with a linear system using first-order Taylor expansion, where Ft represents f derived with respect to x(t|t) and H is h derived with respect to x(t|t-1).

The goal is to determine P(x(t)|y(t),y(t-1),...,y(1)) with t=1...T. This is done in two steps: prediction and update.

After the prediction phase the state will be:

$$x(t|t-1)=f(t-1)(x(t-1))$$

$$P(t|t-1)=F(t-1)\ P(t-1|t-1)\ F(t-1)^T+\ Q(t-1)$$

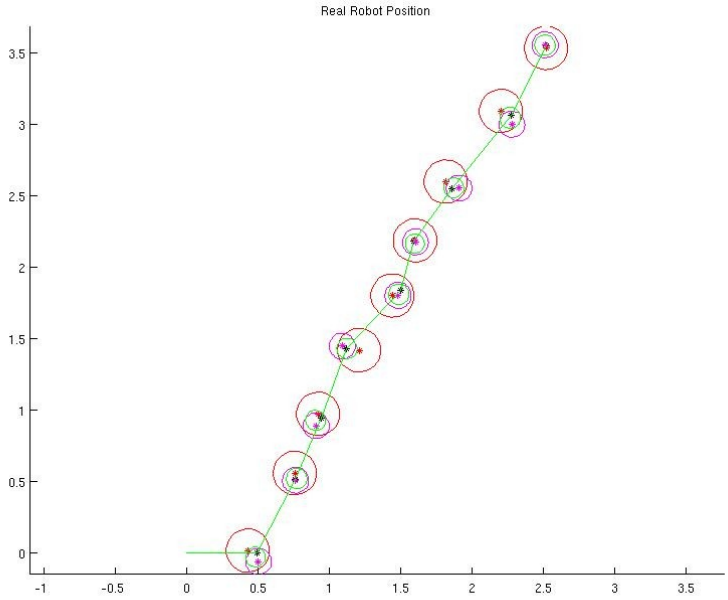
Then, we execute the update process, obtaining:

$$x(t|t)=x(t|t-1)+K(t)(y(t)-h(x(t|t-1))$$

$$P(t|t)=(1-K(t)H(t))P(t|t-1)$$

$$K(t)=P(t|t-1)*H(t)(R(t)+H(t)*P(t|t-1)*H(t)^T)^(-1)$$

The following figure shows the result of the filtering step.



Estimation result after the filtering step. The green ellipses represents the estimates obtained by using the EKF approach.

We will describe the second phase: the smoothing step.

Smoothing

The goal of the smoothing process is to obtain the probability distribution P(x(t)| y(T),y(T-1),...,y(1)) with t=1...T.

The equations are:

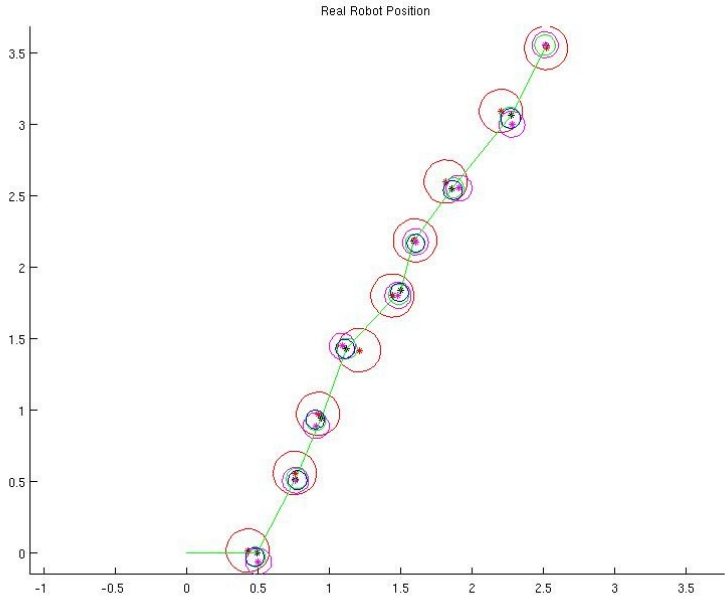
x(t-1|T)=x(t-1|t-1)+A(t-1)(x(t|T)-x(t|t-1))

A(t-1)=P(t-1|t-1)F(t-1)P(t|t-1)^(-1)

P(t-1|T)=P(t-1|t-1)+A(t-1)(P(t|T)-P(t|t-1))A(t-1)^T

Note that F(t-1), P(t|t-1) and x(t|t-1) are already calculated in the EKF prediction step and P(t-1|t-1) and x(t-1|t-1) are calculated in EKF update step.

The following figure shows the output of the smoothing step.



Estimation result after the smoothing step. The blue ellipses represents the estimates obtained by using the EKS approach

In order to use the Extended Kalman Smoother we need to resample the GT streams in input since the smoothing process requires to make the time discrete. Each data in the GT streams has a timestamp that identifies when it has been collected. Different streams have different sampling frequencies. Because of this, a resampling process is needed. This is performed by using a linear interpolation between two contiguous data in the stream. When the EKS estimator requires a new data at time t, the data in the stream at time (t1<t) and its contiguous at time (t2>t) are interpolated.

An assessment of the performance of the fusion system, when applied to the GT-laser and GT-vision data streams, will be done in Section 4.2.8.

References

Derivation of Extended Kalman Filtering and Smoothing Equations, Byron M. Yu, Krishna V. Shenoy, Maneesh Sahani

4.2 Ground truth validation

RAWSEEDS aims at providing the users of its Benchmarking Toolkit with best-quality data. This requirement, when applied to the ground truth data collected along with sensor data, means that such data must be gathered with state-of-the-art methods; but also, in addition to that, that GT data must be *validated*, i.e., that the errors and imprecisions affecting such data must be evaluated and characterized. Moreover, knowledge of the errors associated to the GT-laser and GT-vision data streams is necessary to optimally configure the smoothing algorithm used to obtain the overall GT, built by fusing such streams.

The validation of RAWSEEDS' ground truth data for indoor datasets required the definition and execution of a procedure aimed at verifying the correctness of the trajectory data generated by the GT-vision and GT-laser systems, and at evaluating the errors on such trajectory estimation. In this section we will describe the procedure used to validate RAWSEEDS' indoor ground truth data and the results obtained from the validation.

Please note that RAWSEEDS already performed a ground truth validation effort, amply documented by Additional Deliverable AD2.3 (Validation and Testing). Most of the techniques described in the following of this section correspond closely to those illustrated by AD2.3. For this reason, and because this document is not specifically dedicated to ground truth validation, the description of such techniques in the present document will not be as thorough and painstaking as it is in AD2.3. In particular, we will occasionally make references to specific procedures described by AD2.3 instead of repeating their descriptions, when such procedures were adopted without any modification. Deliverable AD2.3, of course, remains available as a reference about such matters.

4.2.1 Overview

To validate the GT collection systems, we used the following procedure:

1. put the robot in a suitable set of *known* (or "reference") robot poses within the GT area;
2. for each "reference" pose, acquire the data required by both GT-vision and GT-laser systems to compute (or "reconstruct") the postion of the robot;
3. reconstruct each of the poses of the robot with both the GT-vision and GT-laser systems;
4. evaluate the precision of the reconstructed poses by comparing them to the "reference" poses.

Actually, the "reference" poses themselves were not known *a priori*, but measured: they were, therefore, affected by unavoidable errors. This required a validation of the "reference" poses before they could be used in turn to validate the poses reconstructed by the GT-vision and GT-laser systems. The method used to measure

the "reference" poses is described by Section 4.2.4, while the validation of such poses and its results are described by Section 4.2.5. This validation was based on the use of a high-precision map of the region where ground truth data were collected, generated manually (by performing suitable measurements) on the base of the executive drawings of the building. This was done because we did not want to rely blindly on the executive drawings, as our experience shows that they are frequently inaccurate with respect to the actual interior of a building. Section 4.2.2 compares the manually reconstructed map to the executive drawing; Section 4.2.3 explains how the alignments between the frames of reference associated to the map and those associated to the GT-laser and GT-vision systems were performed.

As it happens, the precision of the first set of "reference" poses proved to be lower than expected. This outcome, given the proven success of the same procedure during the preparation of Deliverable AD2.3, was unexpected. Subsequent investigations - briefly described by Section 4.2.5 - ascertained that the professional laser range finder used to define the reference poses both for AD2.3 and for this document markedly changed its precision when large distances and angled surfaces were both involved (something that could not happen in the test environment used to generate the results described by AD2.3).

A new set of "reference" poses were then generated without the use of the manual laser range finder, and then validated with the same procedure used for the first set. The technique used to produce the new set of "reference" poses is described by Section 4.2.6. Since the validation of this new set of poses yielded good results, they qualified to be subsequently used to validate the poses reconstructed by the GT-vision and GT-laser systems.

Finally, Section 4.2.7 describes the validation of the GT collection systems (laser-based and vision-based) using the second - and accurate - set of reference poses. In addition to that, the same algorithm for fusion and smoothing used to generate the GT associated to each of RAWSEEDS' datasets (which uses the GT-vision and GT-laser data as its inputs) was applied to the data used for ground truth validation. Its output was then subjected to the same validation procedure applied separately to the GT-vison and GT-laser data. In this way, also the complete procedure used to generate the best-quality ground truth distributed along with RAWSEEDS' datasets was validated.

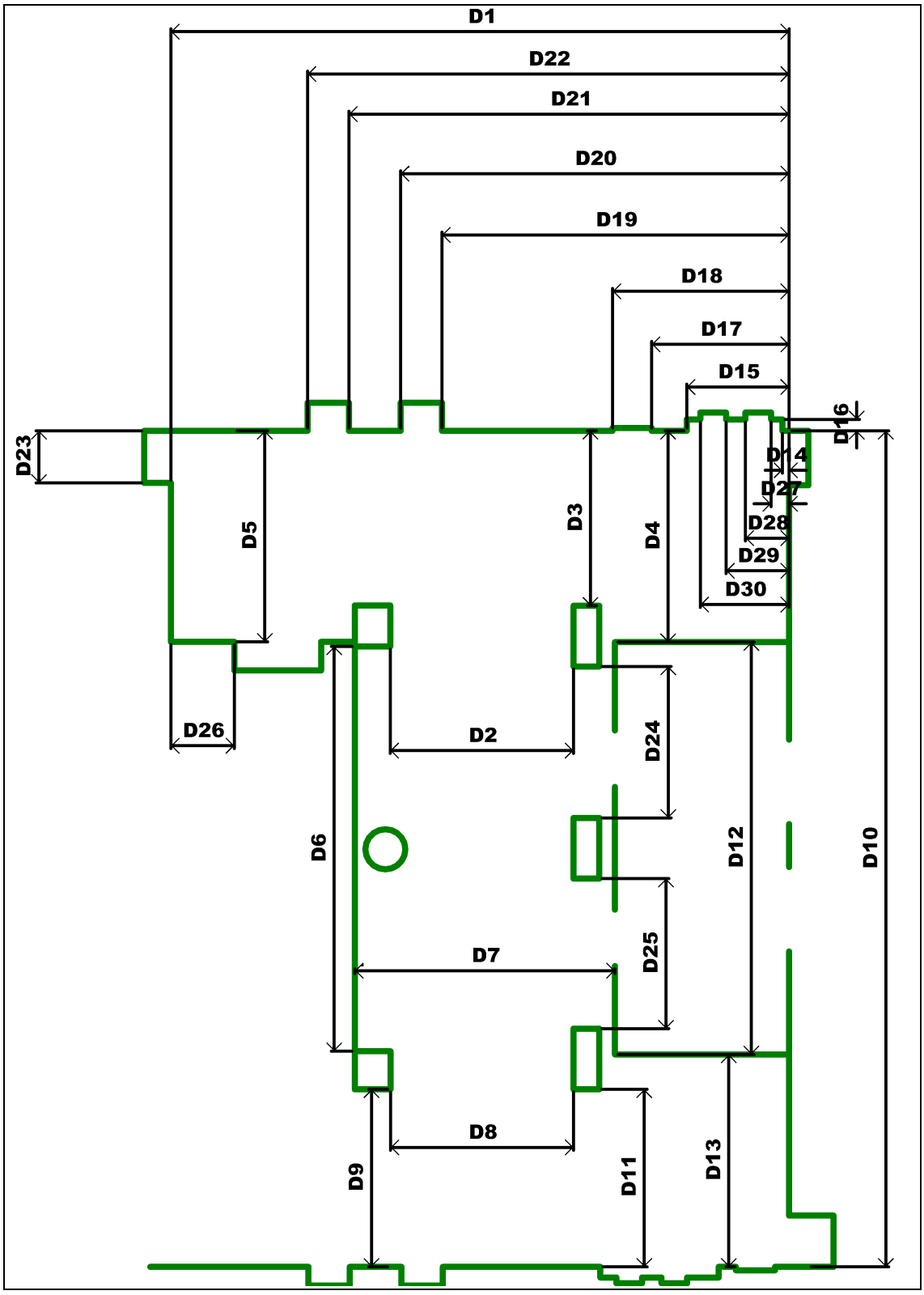
4.2.2 Map of the GT area

Two maps of the GT area were available: the executive drawings of the building (in the dxf format typical of CAD) and a new map, of the GT area only, carefully taken by hand using a manual laser range finder (Bosch DLE 50 Professional manual laser range finder, the same device used to generate the data for Additional Deliverable AD2.3 - Validation and Testing).

Although the executive drawing proved to be surprisingly adherent to the actual dimensions of the building (many factors - e.g., the finishing of surfaces - influence this correspondence, so that the drawings are often very unreliable), the manually generated map proved superior. Therefore the latter was used as a reference for all subsequent GT validation operations. From now on, unless specified otherwise, the

term "map" will refer to the one generated manually. In the following of this subsection, a brief comparison of the two maps is given.

In order to verify the correspondence between the executive drawing of the zone used for the ground-truth and the effective position of walls, door and other things, we did some measurements and comparisons. The figure below shows the executive drawing of the portion of building where the GT area was located. A set of 30 dimensions was chosen to compare the drawing to the manually generated map: such dimensions are showed in the figure.

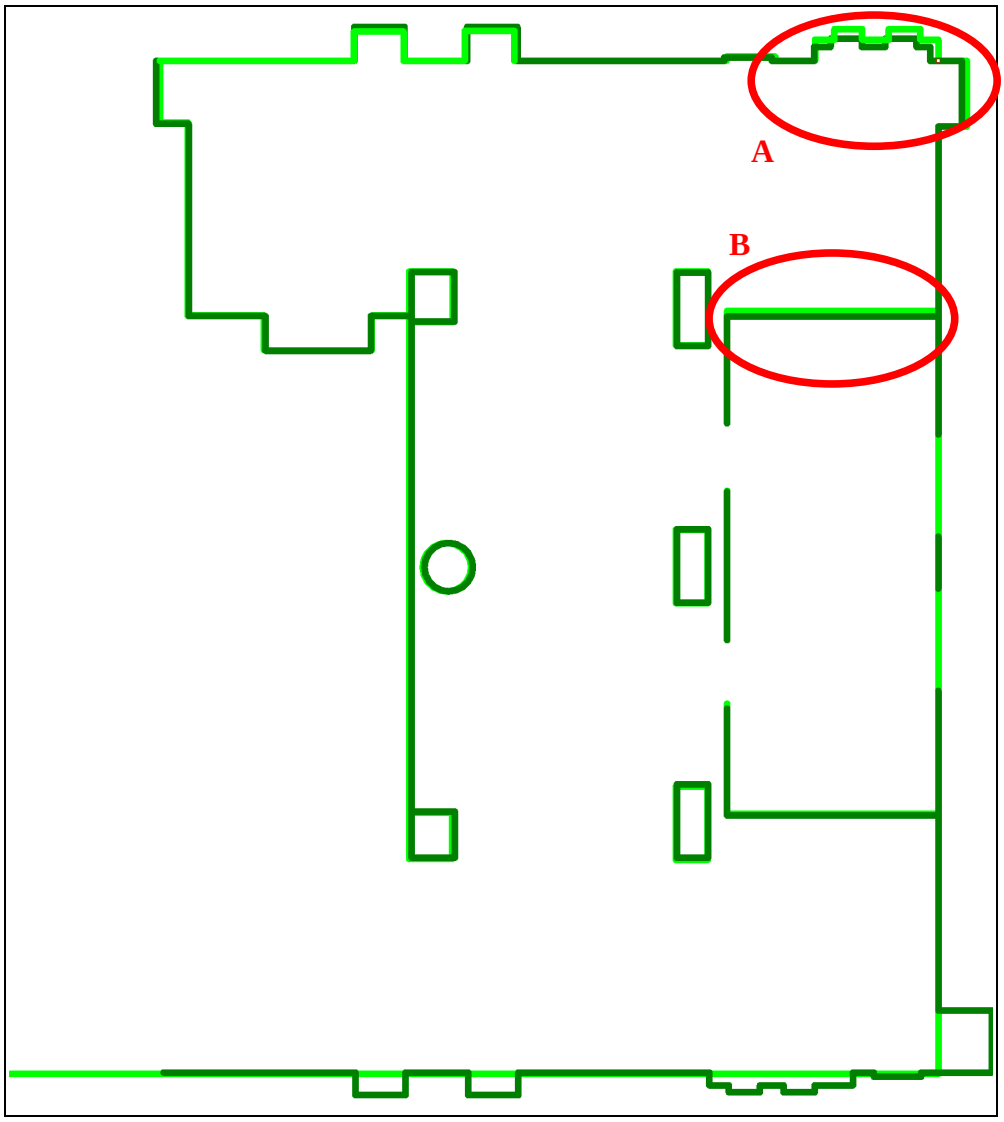


The results of the comparison of the values of the chosen dimensions between executive drawing and manually generated map are shown in the following table.

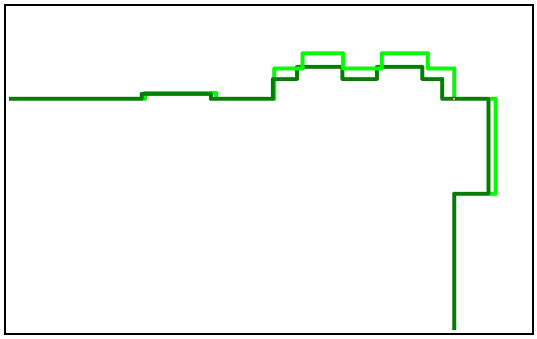
Dimension	Measure on map [mm]	Real Measure [mm]	Errore [mm]
D1	21650	21671	21
D2	6420	6445	25
D3	6110	6083	-27
D4	7380	7220	-160
D5	7370	7351	-19
D6	14140	14177	37
D7	9110	9183	73
D8	6410	6461	51
D9	6200	6208	8
D10	29200	29245	45
D11	6200	6281	81
D12	14400	14521	121
D13	7420	7504	84
D14	236	0	-236
D15	3583	3559	-24
D16	399	609	210
D17	4813	4709	-104
D18	6183	6117	-66
D19	12157	12246	89
D20	13605	13676	71
D21	15415	15442	27
D22	16863	16878	15
D23	1820	1780	-40
D24	5290	5325	35
D25	5240	5288	48
D26	2220	2204	-16
D27	630	520	-110
D28	1530	1430	-100
D29	2210	2200	-10
D30	3110	3000	-110
		Mean	0,55
		Standard Deviation	90,34
		Maimum overestimation error	210
		Maimum undestimation error	-236

The mean error is very close to zero: this means that the actual realization of the building - or, more precisely, its refurbishment: the building is, in fact, a revamped industrial site - closely followed the projects (as we said previously, this is not always the case). Nonetheless, local errors cause a standard deviation of about 100 millimetres.

The next figure shows the executive drawing (in dark green) superimposed with the manually generated map (light green). Please note that that not all features of the first map have been included into the second, as only the elements considered significant for our purpose were measured (e.g.: some doors on the lower edge of the map were not considered).

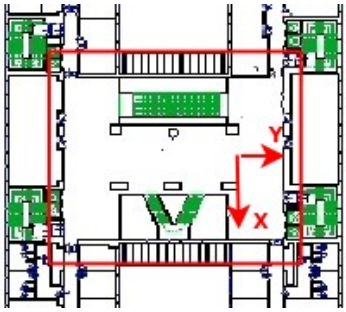


The two maps are remarkably similar. The largest errors are localized in the zones identified with red ellipses. In particular, in the zone named “A” this is due to the presence of elevators; their doors are not in the expected positions and, in fact, the maximum deviations between the two maps (both underestimation and overestimation) are located in this zone (dimensions D14 and D16). To better illustrate the local error in zone “A”, next figure shows a zoom of the area.



Another significant discrepance is located in the zone named “B”, where an entire wall is moved of about 160mm (dimension D4); however this error is not significant for ground truth validation, as such wall was not perceivable by the sensors on board of the robot while it was in the GT area.

After the decision to use the manually generated map as a reference for GT validation activities, we set up an arbitrary frame of reference in the environment by marking a Cartesian couple of orthogonal axes on the floor in a convenient position. The position of such frame is roughly shown by the following figure (please note that this image is rotated by 90° with reference to the preceding map).



Then we proceeded to localize the frame with reference to the boundaries of the area covered by the map by carefully measuring its position relative to suitable features of the map. And finally, we defined the position of all points of the map in the newly established frame of reference. In the following, such reference frame will often be called the *world reference frame*.

4.2.3 Alignment of the reference frames of the GT collection systems

To correctly evaluate the performance of the GT collection systems (or even to use the data produced by them) it was necessary to align their own frames of references with the world reference system. In this way, the trajectories generated by the GT-laser and GT-vision systems could be correctly placed in the reference frame of the map. This alignment was performed separately for the GT-laser and GT-vision systems: in the following of this subsection we will describe how.

For the GT-laser system, after positioning in the GT area the four Sick LMS200 sensors that such system is based upon, the poses of such sensors in the world reference system had to be determined. This operation was centered on the use of a Matlab script based on the same ICP alignment algorithm described in Section 4.1.2, and was structured as follows:

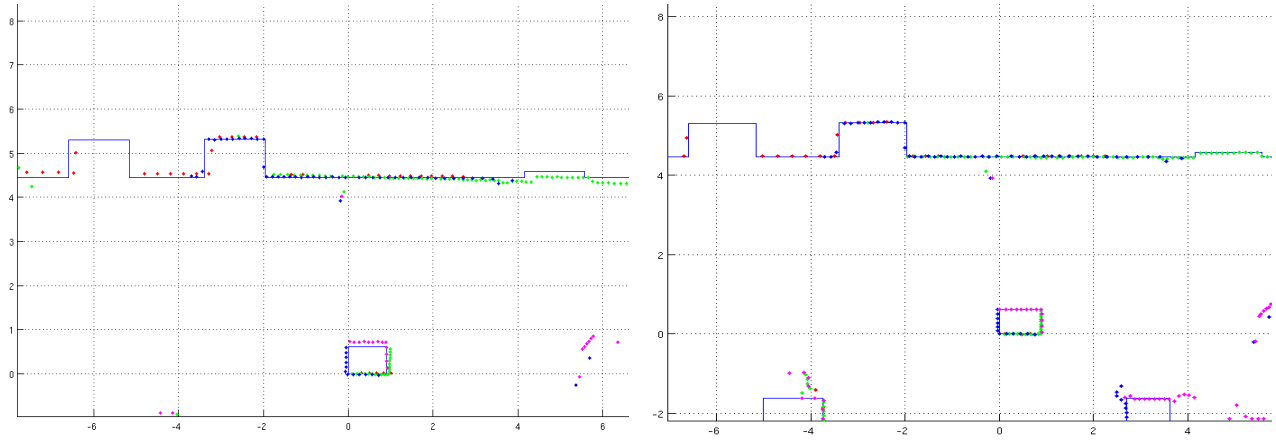
1. the position of each of the four Sick sensors (in the following they will be called Sick1, Sick2, Sick3 and Sick4) of the GT-laser ground truth collection system in

the world reference frame was measured manually (with the Bosch DLE 50 laser range finder);

2. an object of simple geometry and known dimensions (a big rectangular cardboard box with rigid, straight sides) was positioned in the environment with a corner in the origin of the world reference frame and its sides parallel to the frame's axes;
3. a scan was acquired from each of the Sick sensors;
4. the above scans were referred to the world reference frame, using the poses of the sensors measured at point 1;
5. application of the ICP algorithm to align the scans of point 4, using the poses determined at point 1 as initial poses: this yielded a modified pose of Sick2 with reference to Sick1;
6. while the pose of Sick1 in the world reference frame remained unchanged, the pose of Sick2 was updated considering the result of point 5;
7. a new scan was generated by merging the scans of Sick1 and Sick2 (please note that the two sensors were correctly aligned one with respect to the other due to the action of point 6, so the scans overlapped with good precision): this merged scan will be called Scan12;
8. the ICP algorithm was applied again to align the scan of Sick3 (starting by the position of point 1) with Scan12, leading to a modified position of Sick3 (a position that gives good alignment with Sick1 and Sick2);
9. the scans from Sick1, Sick2 and Sick3 were merged: the resulting scan will be called Scan123;
10. the ICP algorithm was applied again to align the scan of Sick4 (starting by the position of point 1) with Scan123, leading to a modified position of Sick4;
11. as the position of Sick2 on the map (i.e., in the world reference frame) was not considered fully satisfactory, a new run of the ICP algorithm was done with such initial position and the merged scans of Sick1, Sick3 and Sick4 (with the last 4 set in the poses determined at points 8 and 10 respectively);
12. the ICP algorithm was applied to the overall scan (Scan1234) obtained by merging the scans from all the sensors (set in their last computed poses), to align it to the map (the more precise, manually generated one);
13. point 12 led to a collective update of the positions of all the Sick sensors, mantaining their relative poses but modifying their poses with reference to the world reference frame;
14. finally, the ICP algorithm was applied again - separately - to each of the scans from the four sensors, considering the last computed poses of the sensors as initial poses and with the aim of better aligning each sensor to the map.

The following figure shows the portion of the GT area where the cardboard box was placed, superimposed with the map lines and the scans from the four fixed Sick LMS200 laser range scanners (plotted in different colours). On the left it is shown the

situation before the alignment, on the right the situation after the alignment. Measurement units are metres; please note that the references are not the same for the two pictures.

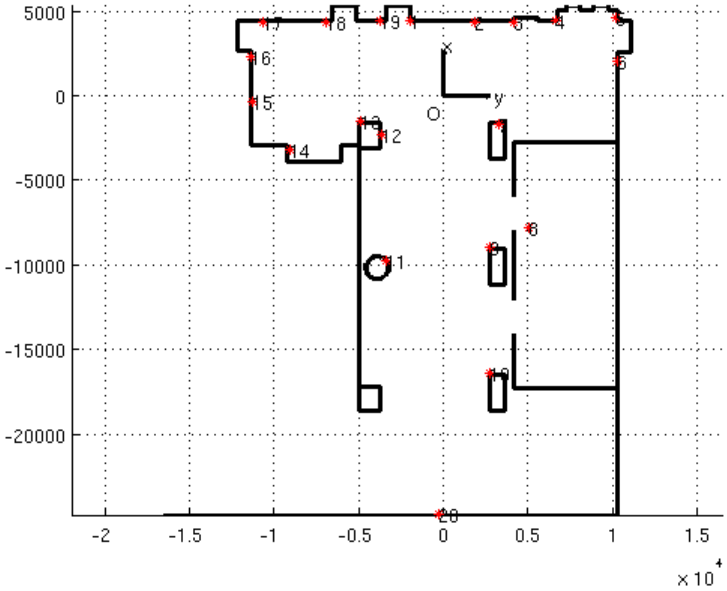


For the GT-vision system, alignment of the reference frame of the cameras to the world reference frame was performed with the procedure described in Section 4.1.1, while considering camera calibration.

4.2.4 Measurement of the "reference" poses of the robot

As we previously said, the GT validation procedure required that the robot was positioned in a suitable set of physical positions. The poses of the robot when set in each of such positions had then to be determined and localized in the world reference frame (i.e., the reference frame of the manually generated map).

We chose a set of 23 suitable robot positions within the GT area (this number of positions was sufficient to cover, with sufficient spatial sampling resolution, a suitable path within the GT area). Then we defined 3 points on the outer frame of the robot (which was made to be box-like, with vertical outer surfaces, to increase the precision of GT-laser localization) by intersecting such frame with an horizontal plane and selecting 3 of the 4 edges of the rectangle thus obtained. Finally, for each of the 23 positions the pose of the robot robot (in the reference frame of the world) was collected by measuring the distances of each of the 3 selected points on the frame of the robot from 20 fixed *fiducial points* set on the walls of the physical environment. The following figure shows the fiducial points (and the world reference frame) on the map of the GT area; measurement units are millimeters.



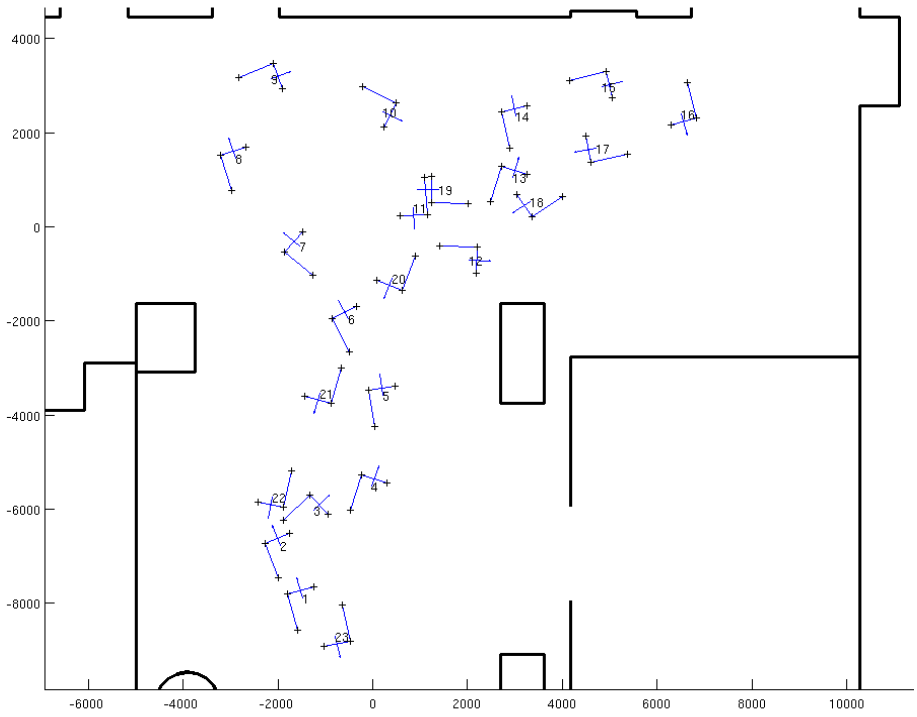
The positions of the fiducial points are not uniformly distributed over the GT area. This was a deliberate decision: in face the fiducial points have been chosen so that their visibility from the "reference" poses was maximized, while taking into account obstructions of all kinds (such as building elements or RAWSEEDS' gear and material). Finally, fiducial point number 8 appears to be in a "strange" location because it was positioned on the side balaustrade of a staircase leading down.

The positions of the fiducial points in the world reference frame were computed in the following way:

1. Given the (arbitrary) world frame of reference, defined in section 4.2.2 and specified by markers on the ground, a set of 5 additional points were defined on the ground around the origin (at a distance of 4m maximum from it). This was done by carefully positioning an L-shaped metal frame on the ground in correspondence with the horizontal axes of the world reference frame, and by marking on the ground the end points of the "L": the positions of such additional points were, therefore, associated to the dimensions of the metal frame, which in turn had been measured with great care by hand.

2. The locations of the 20 fiducial points in the world reference system were defined by measuring the distance of each of them from the 5 points defined at step 2 and from the origin, and then using the Kalman-filter-based algorithm defined in AD2.3, already cited and also used to define the position of the robot with reference to the fiducial points.

The pose of the robot was computed from the measurements of the distances of the 3 robot points from the 20 fiducial points with the estimation procedure already described in Additional Deliverable AD2.3, i.e., a Least Squares estimate of the intersection of circles, implemented by means of an extended Kalman filter (EKF). The results of such procedure is shown by the following figure. Each pose is identified by a progressive number and a portion of the robot frame (two edges) is plotted. Robot orientation is identified by an arrow, corresponding to the X axis of the robot's own frame of reference (defined in Section 3.1.2).



Distance measurements were taken with the already cited Bosch DLE 50 Professional manual laser range finder, the same used to generate the data for AD2.3. For some of the 23 poses, one or more of the three points on the robot were not visible from some of the fiducial points, due to the presence of obstacles; however, the position of the fiducial points was chosen in such a way that only a small fraction of them experienced this problem for any pose of the robot within the set of 23 "reference" poses.

4.2.5 Evaluation of the "reference" poses of the robot

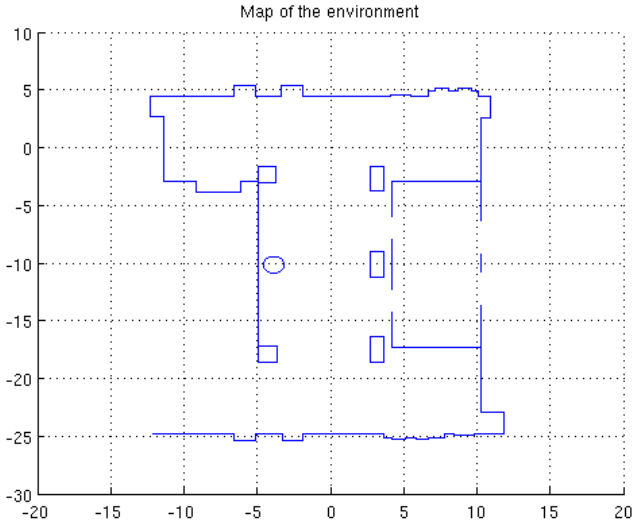
As no "perfect" values for the "reference" poses of the robot used to perform the GT validation were available, a method to evaluate the quality of a set of "reference" poses against an objective benchmark had to be devised. We decided to perform this evaluation by comparing the data acquired - in each of the "reference" poses - by the Sick laser range finders on board of the robot (one LMS200 and one LMS291) with the map of the nearby environment, i.e. with the manually measured map described by Section 4.2.2. Errors in the poses (and especially in the angle component of them) were clearly shown by this method, which has also the advantage of immediately translating such errors into maps, which are easily inspected and interpreted by a human observer.

In practice, the validation procedure for the set of 23 "reference" poses was executed as follows:

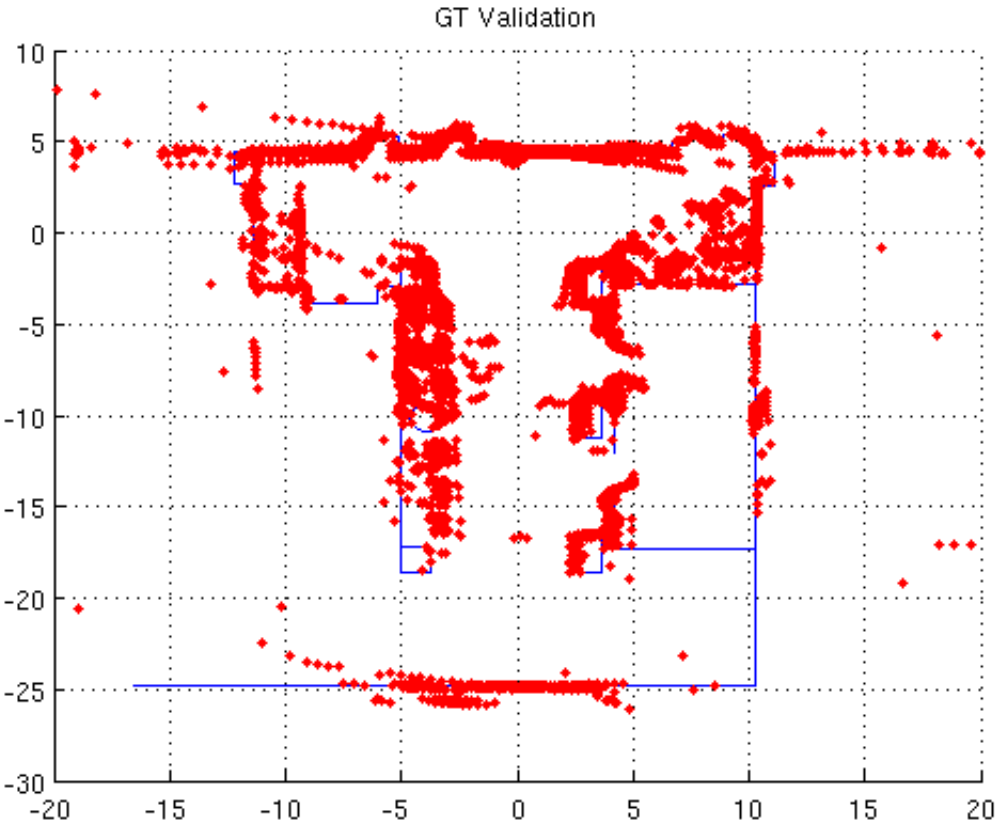
1. for each pose of the set, plot the scans of the onboard laser range finders on the map *while considering the robot as set in the pose under consideration;*

2. evaluate visually the differences between the map and the overall plot (i.e., the superimposition of the plots associated to the single poses).

At the end of such a procedure, in the absence of errors of any kind, the data from the laser range finders will align perfectly with the map. If the poses - or some of them - are affected by errors, such errors will manifest themselves as a "smearing" of the boundaries of the environment, as the scans taken from different locations of the robot will not superimpose perfectly with the map. As a reference to better evaluate the following figures, below is a copy of the map alone, without plotted scans. Measurement units are metres.



We will now show the result of the pose validation procedure described above, when it was applied to the "reference" set of poses (i.e., the one measured by hand). From previous experience in GT validation (documented by Deliverable AD2.3) we expected a good result, i.e., one where the plotted scans were well aligned between themselves and with the lines of the map; on the contrary, the result of this test contradicted our expectations, as shown by the following figure.



The "cluttered" areas on the left and on the top right of the plot were expected, as they correspond - respectively - to a set of floor-fixed, open-frame tables and chairs and to the area where RAWSEEDS' main working area was installed. What we didn't expect, however, was the overall extent of the errors between plot and map.

Three error causes could have influenced this result, i.e.:

- errors in the map describing the physical environment;
- errors in the output of the Sick sensors;
- errors in the "reference" poses (i.e., the errors that this test procedure was aimed at identifying).

In practice, the first two causes of error could be ruled out (or at least considered negligible in this context): the first, because the map was checked with the utmost care and proved reliable; the second, because Sick laser range finders are well known in the field of autonomous robotics and have satisfactory performance for this application (of course we checked that our specific items were correctly functioning). Therefore, the imprecisions in the correspondence between plots and map could be associated with certainty to imprecisions in the "reference" poses, although this outcome also contradicted similar experiences in GT validation described by AD2.3.

We hypotesized that these unexpectedly large errors in the poses were due to a lower precision of the measurements taken with the manual laser range finder and used to estimate the position of the robot. We also supposed that this effect was associated to

much larger measured distances than the ones occurring in the experiments documented by AD2.3, and/or to the fact that many of the measurements were done between a point on the robot and a surface that was not orthogonal to the measuring laser ray. In such conditions, in fact, we observed a visible smearing of the laser dot projected on the surface, which in turn could easily have led to errors in the length of the laser trajectory.

Subsequent tests allowed us to ascertain that the measuremen accuracy of the manual laser range finder was heavily dependent on the angle between the laser beam and the reflecting surface, even when the latter was optimal (light-colored, smooth, non-reflecting). This effect strongly increased with the measured distance, being negligible within a few meters but becoming very noticeable when large distances - such as the ones involved by the GT validation operations described by this document - were to be measured. Conversely, for the setup described by AD2.3 distances were limited to a few meters maximum, as in that case the experiments took place in a much smaller room. This is the likely reason why those experiments, while using the same laser range finder, were not significantly disturbed by measurement errors.

To test the manual laser range finder we performed the following test. We set up a mechanical device fitted with a vertical surface (covered with a matte material with good laser reflection) which was able to precisely rotate around a vertical axis passing for the surface plane. Then, we measured with the manual laser range finder the distance between a point of the surface belonging to the rotation axis and another - suitably distant - point. Such measurement was repeated over a range of rotation angles (and then for the initial angle again, to be sure that nothing had moved during the test). The distance between the two points (around 17m) was comparable with the typical distances to be measured when during the collection of robot poses. The following table shows the results of the test; the angle value is the one between the normal to the reflecting surface and the line of propagation of the laser.

angle [deg]	0	30	45	60	80
measure [mm]	17627	17627	17629	17636	17646

For each angle, multiple measurements were taken (small oscillations of 2-3mm were experienced during the test and are typical; it is important to note that the amplitude of such oscillations did *not* change with the angle): the values in the tables are mean values. As the table shows, when the angle exceeds 45° measurement errors rapidly increase; maximum measure error amounts to 19mm, for an angle (80°) not dissimilar from some of those associated to the measurements of the "reference" poses. Wide angles (i.e. angles exceeding 45°) were very frequent while measuring the pose of the robot, and errors of the magnitude of those listed in the above table could well have led to significant errors on the pose, especially on orientation.

The same test, when repeated at a measuring distance of 2m, showed no errors. This result confirmed that the absence of errors in the data from the manual laser range

finder associated to the experimental setup described by Deliverable AD2.3 is most likely due to the significantly smaller scale of the test environment compared to the GT area considered in this document.

Measurement errors such as those shown in the above table are compatible with the kind of misalignment of the scans that was experienced during the validation of the "reference" poses, especially considering the fact that such type of errors influenced the measured poses of the robot *two times*: first, because they occurred while measuring the distances between robot points and fiducial points; and second, because the stated positions of the fiducial points themselves in the world reference system were obtained with the same kind of measurements.

The conclusion of the above analysis was unequivocal: the set of "reference" poses generated by using manual measurements with the manual laser range finders had too low a quality to qualify - as we hoped - as ground truth in the evaluation of the same poses as reconstructed by the GT-vision and GT-laser systems. A new set of "reference" points was needed. Of course, we needed to define a new method to generate such poses that did not rely (or did not rely solely) on the use measurements output by the manual laser range finder.

It is important to note, at this point, that the kind of measurement errors described above could not have altered the map of the GT area, notwithstanding the fact that such map was based on measurements taken with the same Bosch DLE 50 laser range finder used to measure the "reference" poses. This is due to the fact that, while performing the measurements used to generate the map, we always took care of using the laser range finder in its optimal functioning conditions, i.e., with the laser beam incident on the measured surface with an angle of 90°. In conclusion, although we found out that the manual laser range finder was unsuitable for some kind of measurements (at least without compromising its remarkable precision), but that such kind of measurements was not required by the building of the map of the GT area. We could then continue to be confident in the quality of such map.

4.2.6 Generation of a more precise set of "reference" poses

As we showed in Section 4.2.5, the first set of 23 "reference" poses of the robot, to be used for GT validation, proved to be affected by excessive errors to be usable for that task. We then proceeded to devise a procedure to refine such set and produce from it a new set of substantially more precise poses. The approach we chose to solve this problem was, again, based on the application of the ICP algorithm for the alignment of clouds of points, already described in Section 4.1.2; the additional information used to refine the poses were given by the comparison between the scans of the onboard Sick laser range finders and the map of the GT area.

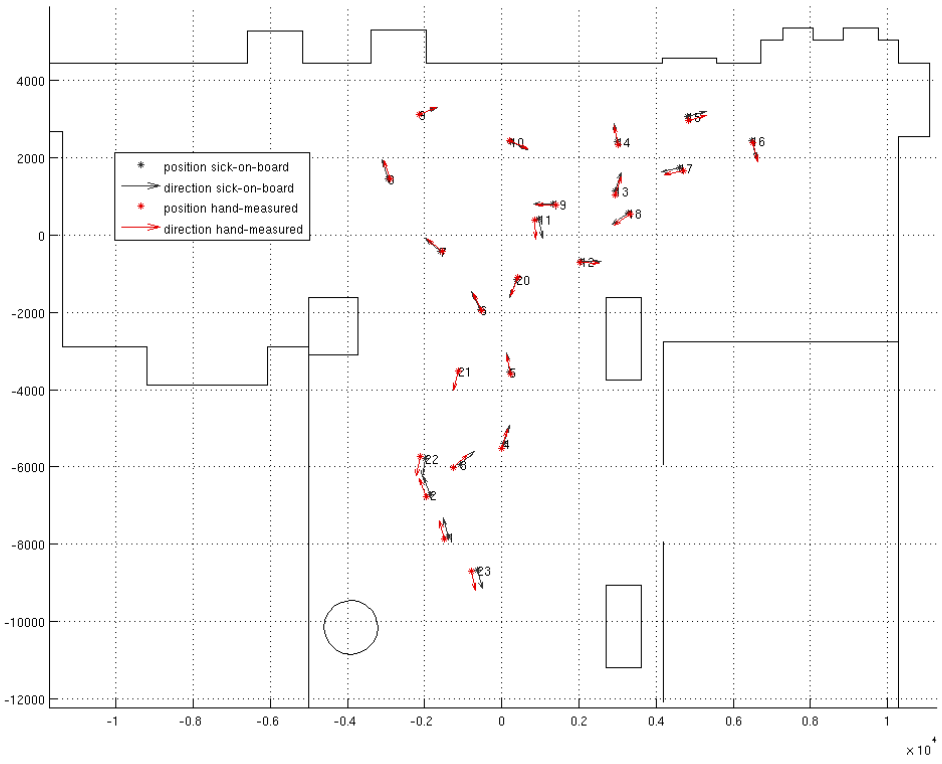
The procedure to generate the new set of "reference" poses was the following:

1. first, we found a better estimate of the relative pose of the two Sick sensors on board of the robot: this was done by applying the ICP algorithm to align the scans from the two sensors with the hand-measured map of the GT area, using

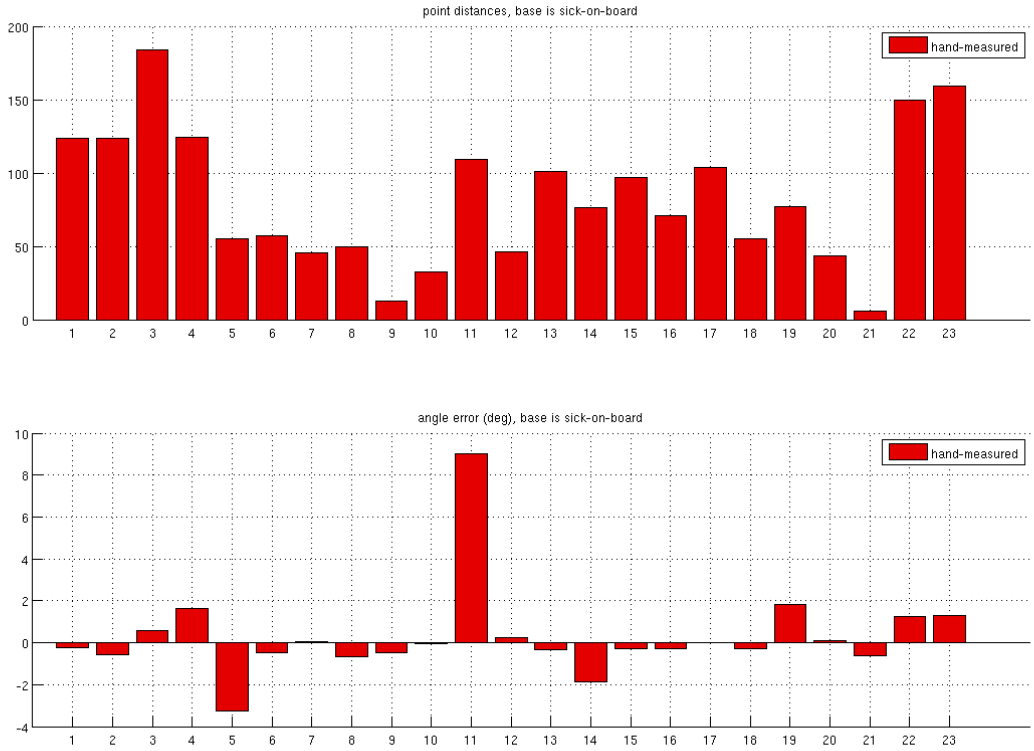
the hand-measured relative pose of the two devices as the start value (this step brought a minor refinement of such relative pose);

2. then, for each of the 23 "reference" hand-measured poses of the robot, we applied the ICP algorithm to align the scans from the Sick laser range finders (acquired with the robot in the considered pose) with the hand-measured map of the GT area: in this case we used as start value for the robot pose the "reference" pose generated from manual measurements and described in Section 4.2.4.

The above steps brought to a new set of "reference" poses of the robot, obtained by using the ICP alignment algorithm to refine the previous - and unsatisfactory - set. The following figure shows the two sets of 23 "reference" poses, represented by small arrows corresponding to the X axis of the robot's own reference frame. The starting point of such arrows corresponds to the origin of such frame. The poses coming from manual measurements are depicted in red, while the new set of poses obtained by applying the ICP algorithm to the outputs of the Sick laser range finders are depicted in black. Measurement units are millimeters.



At a first glance, the two sets of "reference" poses can appear very similar to each other. The next figure shows more clearly the differences between them, by analyzing the distances (in terms of position of the origin and of orientation of the X axis) between the two sets. Measurement units are millimeters for linear distances, degrees for angular distances.

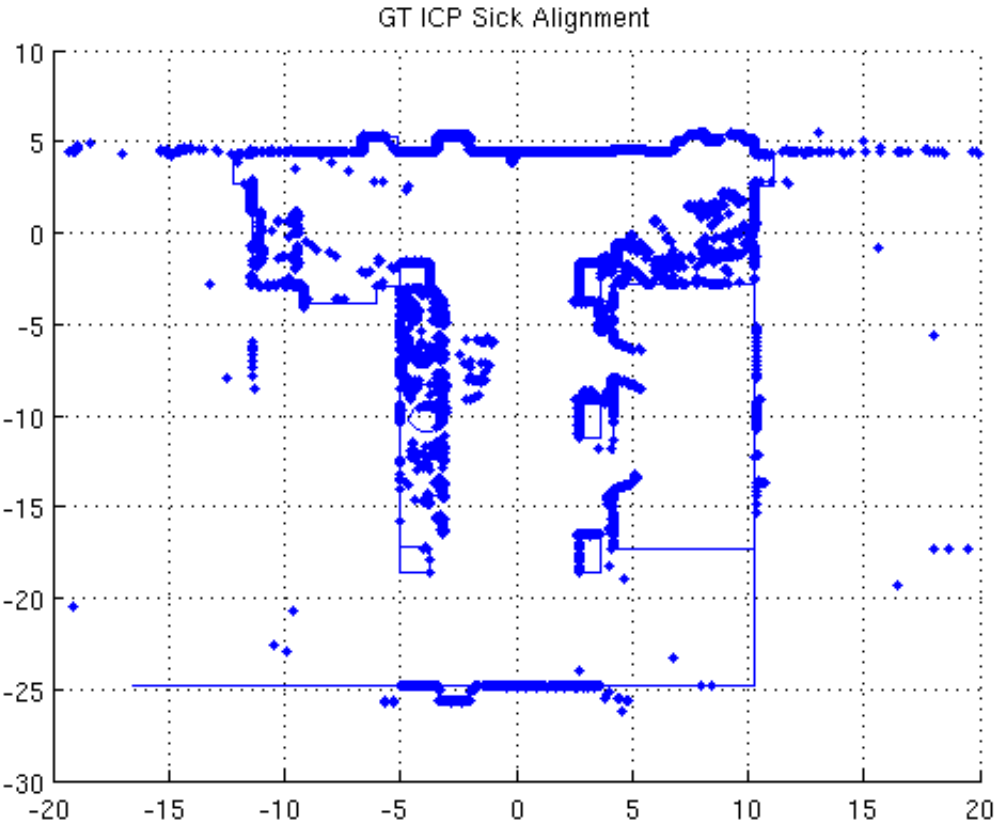


Linear distances are always smaller than 200mm, with a mean value of 83mm and a standard deviation of 47mm; angular distances are comprised in a range of about ±3 degrees, except for pose number 11 that suffers from a much larger error, and sport a mean value of 0.28 degrees and a standard deviation of 2.19 degrees.

From the above data, however, it is not immediately apparent if, and to what degree, the set of "reference" poses generated with the ICP algorithm is more accurate than the first one obtained by manual measurements. To investigate on this issue, we repeated on the new set the validation test applied to the first one, i.e.:

1. for each pose of the set, we plotted the scans of the onboard laser range finders on the map, while considering the robot as set in the pose under consideration;
2. we evaluated visually the differences between the map and the overall plot (i.e., the superimposition of the plots associated to the single poses).

The result of the test is shown by the following figure.



It is immediately apparent how the plots associated to the new set of "reference" poses are much better aligned to the map, in comparison to the plots associated to the first set. This can be considered as valid evidence that the new set is significantly better than the old one.

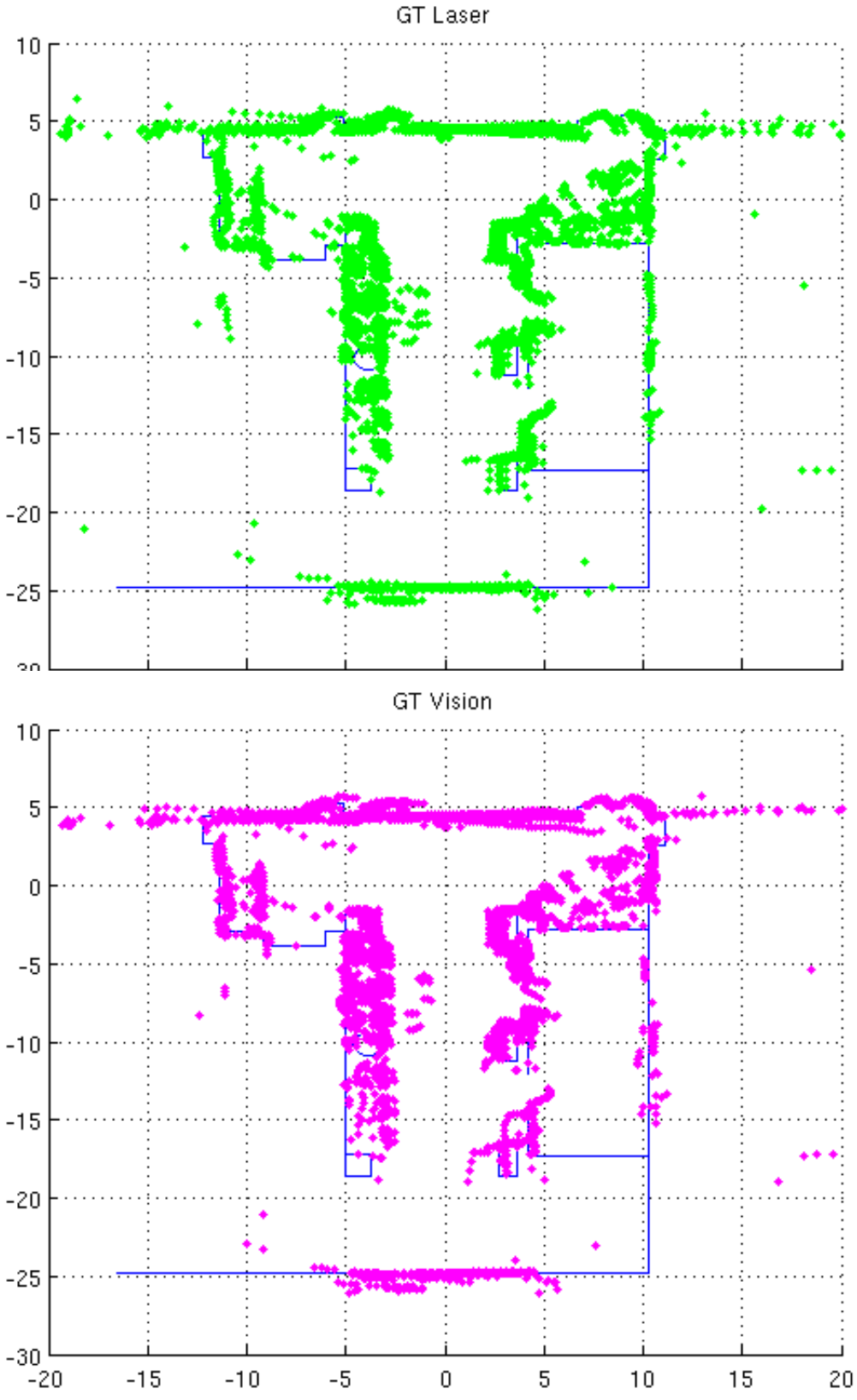
Following this analysis, we decided to adopt the new set of poses as the set of "reference" poses to be used to evaluate the precision of the GT data generated by the GT-vision and GT-laser systems. In the following of this section, therefore, the "reference" poses will be those obtained with the algorithm outlined above.

4.2.7 Assessment of the GT- collection systems

It is now possible to describe the results of the validation procedure applied to the GT-vision and GT-laser systems. For each one of the two ground truth collection systems the validation procedure, already described before, is based on the comparisons between the 23 "reference" poses generated with the algorithm described in Section 4.2.6 and the corresponding 23 *reconstructed* poses output by the system.

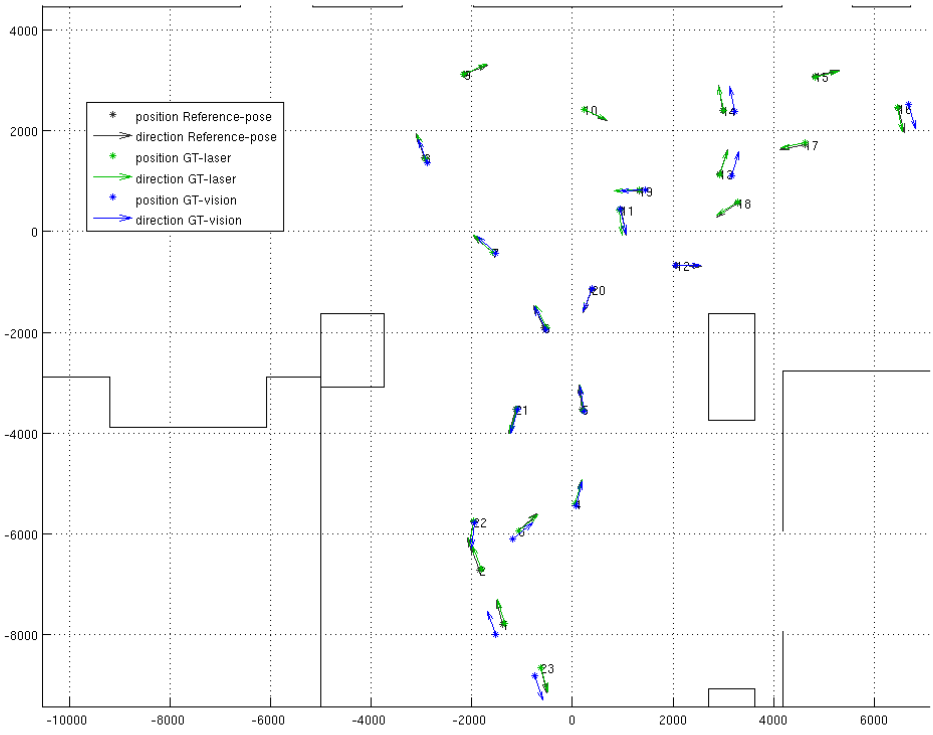
First of all, it is interesting to give an immediate perception of the performances of the two GT collection systems with plots analogous to the one used in Section 4.2.6. More specifically, for each of the "reference" poses we will plot (on a single graph

including the map of the GT area) the output of the Sick sensors on board of the robot, executing the plot by considering the robot as set in the pose reconstructed by the GT collection system under consideration (instead than in the corresponding "reference" pose). The results are shown in the following figures: first for the GT-laser system, and then for the GT-vision system.



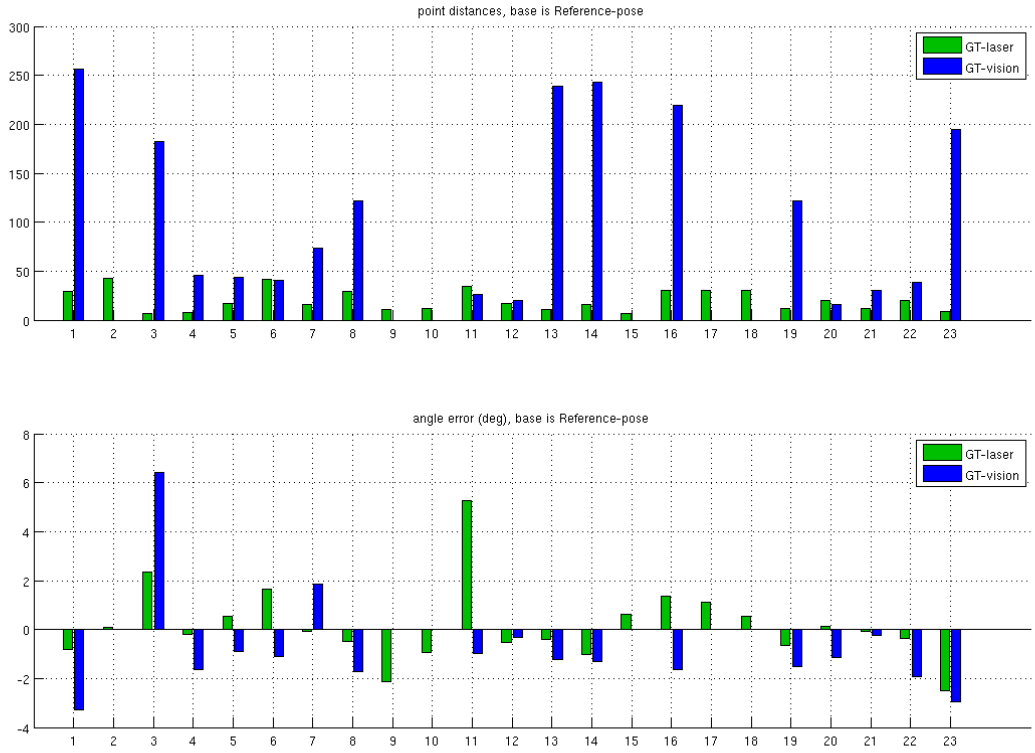
These plots show - as expected - a degradation with respect to the corresponding plot generated using the "reference" poses: in particular, they highlight the fact that some of the reconstructed poses are affected by angle errors, easily detectable with this kind of representation. However, the above figures also show a good correspondence between the plots and the map. In the following of this Section we will analyze quantitatively such correspondence.

The next figure show the 23 robot positions used for ground truth validation. For each one of them, three different poses are shown, i.e.: (i) the "reference" pose, in black; (ii) the pose reconstructed by the GT-laser system, in green; (iii) the pose reconstructed by the GT-vision system, in blue. Please note that for poses 9,5,15,17,18 the GT-vision system was unable to localize the robot. Measurement units are millimeters.



Reconstructed poses coming from the GT-laser ground truth collection system are visibly more precise than those coming from the GT-vision ground truth collection system, particularly when we consider poses that are far from the world frame of reference. This is coherent with the inner working of the GT-vision system, as in such cases the robot is observed by cameras having a reference frame that is linked to the world reference frame by a chain of multiple rototranslations.

The next figure shows more clearly the differences between the output of the GT-laser and GT-vision systems. In particular, linear and angular distances between the "reference" poses and the reconstructed poses are shown. The results associated to the GT-laser system are depicted in green, those associated to the GT-vision system are in blue; measurement units are millimeters and degrees.



Again, it is easy to notice that the GT-laser system is more precise than vision. In particular distance errors for GT-laser are always smaller than 50mm, while orientation (except for pose number 11 that suffers from a big error) are included within a range of ±2,5 degrees. For the GT-vision system, distance errors are characterized by the presence of high peaks along with much better results, for some poses better than those associated to GT-laser.

For the GT-laser system, the mean linear distance error amounts to 20mm, with a standard deviation of 11mm; the mean angular distance error amounts to 0.15 degrees with a standard deviation of 1.56 degrees.

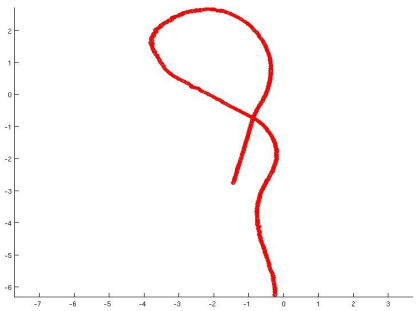
For the GT-vision system, the mean linear distance error amounts to 112mm, with a standard deviation of 90mm; the mean angular distance error amounts to -0.80 degrees with a standard deviation of 2.16 degrees.

4.2.8 Assessment of the GT data

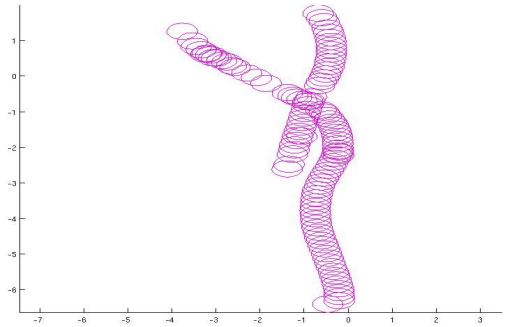
For each of RAWSEEDS' datasets, the ground truth provided along with it is the result of the application of the smoothing algorithm described by Section 4.1.4 to the data streams output by the GT-laser and GT-vision ground truth collection systems during the acquisition of the dataset under consideration. Therefore, after having assessed (in Section 4.2.7) the performance of the two GT collection systems, it is now appropriate to describe the performance of the overall GT generation system. In other words, we will now analyze the output of the smoothing algorithm when applied to the output of the GT-laser and GT-vision systems.

The evaluation of the smoothing algorithm cannot be done by using the same set of 23 poses already used to assess the GT collection systems. This is due to the particular characteristics of the validation dataset. In fact, single and independent poses constitute this stream. Each pose is stochastically independent of the other ones. There are no links between consecutive poses, and - in general - there is no temporal linking among all these poses. For this reason a filtering and smoothing process is not applicable. Moreover, being the smoothing process a statistical method, it needs to have an estimate of the errors on the input data. These errors is unknown in our case and this validation phase is just the procedure which allows to obtain these estimates. It will be done, instead, by using a segment of GT data associated to one of the datasets described in Section 3, included into Part 1 of this document.

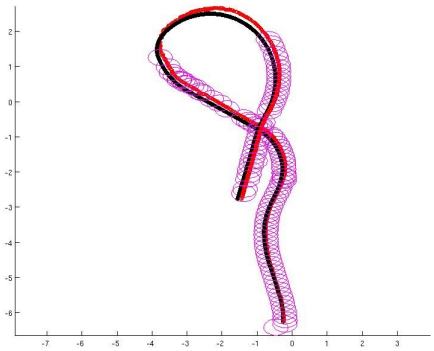
In particular, we will analyze a segment of GT data extracted from the dataset called 20090227A. In the following figures we show each stream of GT obtained by each GT sensor without any smoothing. The estimated poses are described by mean and covariance (obtained during the validation phase) and they are represented with ellipses (+/- 3 sigma) in the drawings. Measures are in metres.



GT laser



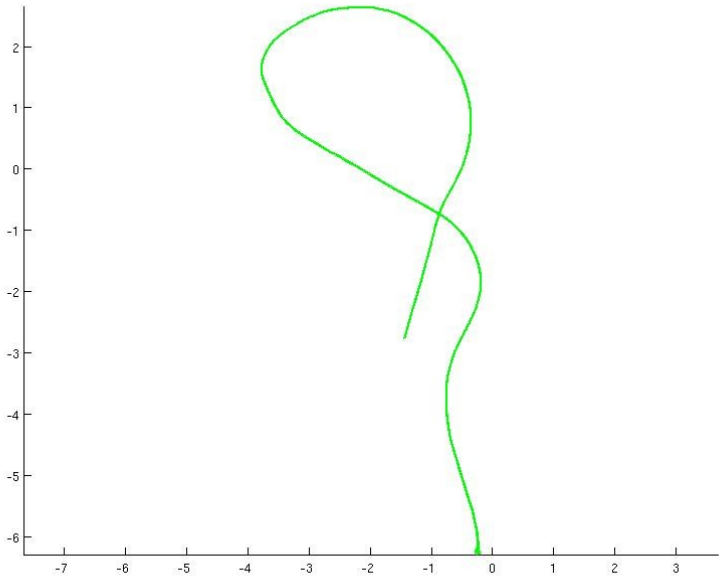
GT vision



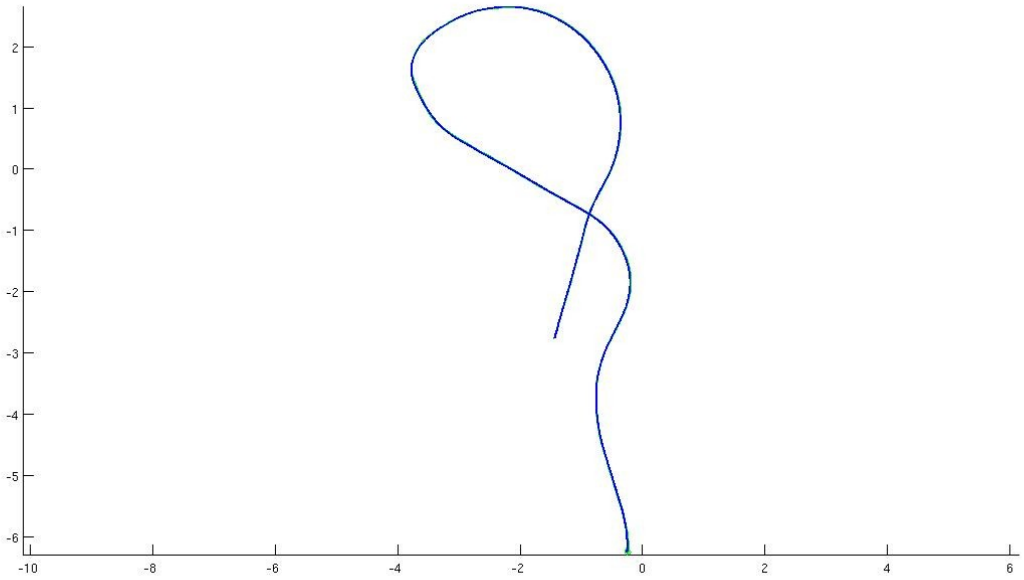
GT laser superimposed with GT vision. In black the odometry data stream.

The last figure represents all the data streams used in the smoothing process. Notice that all the +/- 3 sigma ellipses overlap, making these streams consistent.

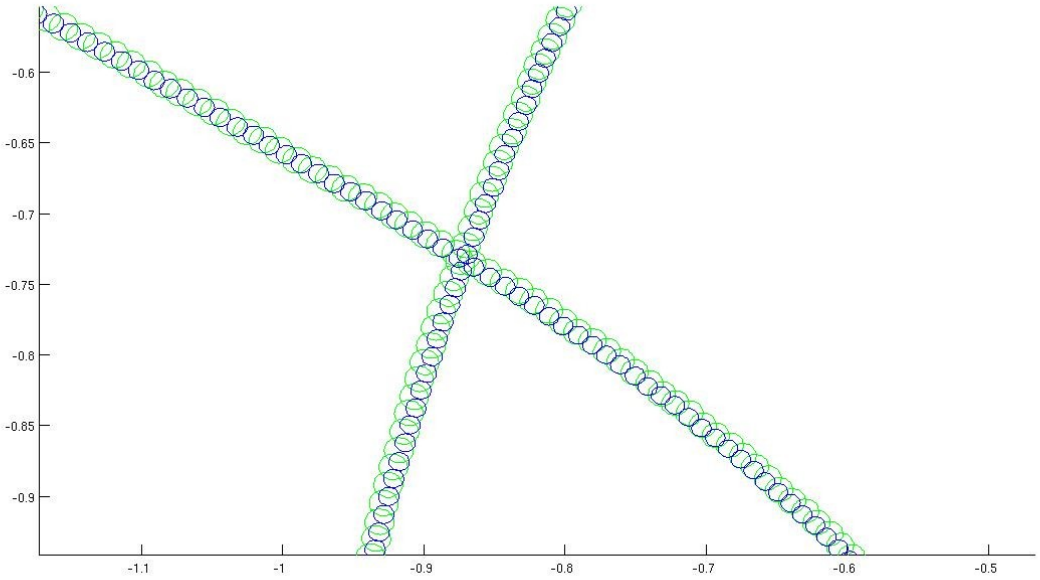
After the filtering phase of the ground truth fusion process (please see Section 4.1.2 for its description) we obtain the following result (green ellipses represents the +/- 3 sigma error in the estimated robot position); measurement are in metres.



Finally, the smoothing phase produces the final result (blue ellipses represents the +/- 3 sigma error in the estimated robot position) shown in the following figures: first in the same scale used in the preceding figure, and then zoomed to make possible a close inspection of the uncertainty ellipses).



Estimated robot positions after the smoothing phase superimposed with the result obtained by the filtering step.



Detail of the highlighted area

As can be seen, the result after the smoothing phase is better than the result after filtering alone (the blue +/- 3 sigma ellipses are smaller than the green ones). Thus, the integration of the information given by the two different and independent GT streams allows to obtain a single and more precise GT stream of robot poses.