RAWSEEDS

Robotics Advancement through Web-publishing of Sensorial and Elaborated Extensive Data Sets

WorkPackage 2

Deliverable AD2.3
Validation of the Ground Truth collection systems

Project no. 045144

Instrument: Specific Support Action

Thematic Priority: IST-2005-2.6.1 Advanced Robotics

date due: end of month 21 (July 31st, 2008)
authors: Domenico G. Sorrenti, Matteo Matteucci

contributors:

Simone Ceriani, Davide Migliore, Giulio Fontana, Davide Rizzi, POLIMI
Alessandro Cirillo, Daniele Marzorati, Davide Tantillo, UNIMIB
Giorgio Grisetti, Cyrill Stachniss, ALUFR

Date of completion of this document: Wednesday, August 6, 2008

Table of Contents

Executive Summary

This document is the description of the validation activities performed in order to assess, in an objective way, the quality of the Ground Truth acquisition procedure in the RAWSEEDS project. This evaluation has been required by the reviewers after the first year activity reports and it is mainly focused on indoor Ground Truth acquisition.

Indoor executive drawings are reasonable as Ground Truth for mapping, but robot pose requires an original solution to be developed. In a recent project meeting it was decided to evaluate two different procedures for indoor Ground Truth acquisition: an external camera network providing an independent robot pose estimate, and the manual alignment (followed by an automatic optimization procedure) of laser scans as acquired by the robot itself. The first should be preferred whenever we are interested in an independent measurement of SLAM performance wit respect to on board robot sensors, the second one is supposed to give better performance, but it has the drawback of manual inspection of all scan alignments and it is not applicable to algorithms using the on-board laser scanners. The validation procedure has been performed in a controlled indoor scenario comparable with the real one; robot pose estimates computed by the Ground Truth systems have been objectively compared with (hand) laser measurements of the real robot pose.

Outdoor Ground Truth is provided by the use of an RTK-GPS. It is known to give a few centimeters accuracy in areas covered by at least 5 satellites and requires a radio link with an RTK-GPS base station; the 5 required satellites have to be contemporaneously in the view of the RTK-GPS base station and also of the RTK-GPS on the robot. We acquired the two apparatus and verified the satellites coverage in the worst case scenario: the mixed dataset acquisition. In this scenario we have both outdoor and indoor areas and we checked the RTK-GPS coverage for the outdoor parts of the acquisition path.

For what concerns the presentation of the document, we decided that the best way to convince a user of our datasets about the accuracy of our Ground Truth collection systems, is to make she/he pass through the very same experience she/he would have if trying to do the same task personally. This document is therefore organized not only to provide the final quantitative data, but also, by means of a large amount of pictures, to allow the reader to eventually spot some mistake of ours. It goes without saying, that in the unlikely case that some mistake will actually be found, such that it heavily affects the results here presented, we will run again this activity.

In the rest of the document "Ground Truth" is shortened in "GT".

Indoor validation

Validation procedure

Introduction to indoor validation procedure

The validation of the GT systems is the activity that allows to validate whether the proposed GT systems are actually capable to perform their task with the required accuracy or not; the procurement of an accurate GT is a key point to prepare a trusted dataset. A trusted dataset is, in turn, the key for its widespread usage.

For the validation of the GT systems we implemented a limited experimentation of the GT systems, altogether with an evaluation of the performance of the GT systems, by means of independent measurement systems. This evaluation is based on manual measurements, which one might expect to be less accurate, but we believe these measurements are, at least, able to convince about the quality of the GT system. With "manual measurements" we actually include the manual usage of instruments, like the laser range finders in normal use in civil engineering (i.e., those that measure the range along a single line, in opposition to the scanning ones, in normal use in robotics).

In order to perform a convincing validation we, unfortunately, had to avoid the places where the actual data-collections will take place, at least for the indoor activities, because of their openness to the general public: this would have implied the unacceptable risk of people moving the cameras because of the long time interval required for the validation. This is to be avoided as it severely degrades the accuracy of the GT system; it also might mean having instrumentation moved or even stolen, too many people asking questions, etc. Being so impractical, we therefore decided to limit the validation to a room with no public access.

In order to validate the quality also of the vision-based GT, we have to include the verification that the cameras do not move, from the moment of acquisition of the data used for setting up the GT system, to the end of the GT collection; we devised a simple procedure for such verification.

The room had to be large enough to replicate the viewing conditions of the real GT system(s), including openings allowing direct sunlight, a sufficiently high number of cameras, and all the effects affecting the performance of the GT systems (e.g., the laser beam can be reflected away from the receiver, the sun-blades can make undetected some markers, etc.).

In the room there are some points, whose coordinates are supposed known. On the robot we also have points, whose coordinates are known with respect to the robot, i.e., the extrema of the robot-frame axes. We manually moved the robot around the room, to the poses where we will validate the GT. Then, for each such robot pose, we collected the validation data. This means to draw on the floor the robot-points and then, for each robot-point, to measure the distance to the known room points: these data allow to compute the robot-frame pose. The robot pose estimates are then compared to the output of the GT systems, on order to evaluate their accuracy. The validation system is the set of room points, their coordinates, the set of robot-point, the practical method to take the measurements of the distances, the software used to compute the robot pose from the distances, etc. The validation system should be accurate enough to allow the appreciation of the errors in the GT systems.

As the validation data are data that allow to evaluate the quality of the GT systems; they are homogeneous in nature to the ones provided by the GT systems, although they are obtained with different approaches. In the case of the robot pose, the validation data should hence represent the parameters of the roto-translation from a reference frame to the robot frame. GT validation is mainly based on human and manual work. To clarify the reasoning behind such, apparently, error prone solution we first summarize the overall picture.

1. When using the datasets produced by the project, for a user it will be similar to being receiving the data from a real robot, moving in its working space, and collecting data with its sensors; the only not perceivable difference it will be that the data-collection did happen some time before. Each research group will propose a different algorithm, i.e., a benchmarking solution (BS) in the terms of the project. Each BS is outputting data about the robot state, and we want to enable the comparative evaluation of its output, with respect to the output of other BS. These estimates of the robot state represent the first set that we will need to consider. We call this set **Robot-Pose-BS**.
2. We therefore need another trust-able source for the same robot state, to be used as a reference for the comparisons. This is the so-called GT; these data cannot be, in principle, the ones obtained by the robot sensors, otherwise the comparison would not be fair. Notice here the relevant exception of the BSs using sensor streams other than the LRFs, e.g., vision-based approaches. These BSs can use a GT system based on the accurate LRFs streams. Whatever the source of the GT, i.e., the GT system, we have here a second set of estimates of the robot-state. We call this set **Robot-Pose-GT**.
3. Lastly, whatever the GT system, we need to validate it, i.e., to perform a quantitative evaluation of the GT system, and to publish the procedure as well as the outcome of this activity, in order to gain a wide acceptance of our datasets. Therefore, we need an approach for the quantitative evaluation of the GT system, which is another independent measure of the robot state. We call this extra set of estimates **Robot-Pose-Validation**.

Of course, the requirements for these different estimates of the robot-state are not the same:

1. **Robot-Pose-BS** is based on the robot sensors and is computed, during the usage of the dataset, by the BSs;
2. **Robot-Pose-GT** is part of the benchmark problem (BP), and has to be provided by means of a source independent on the sensors used by the BSs; it can be provided only for some limited part of the robot workspace;
3. **Robot-Pose-Validation** is not part of the BP, and aims at convincing about the accuracy of the GT system; therefore it might be built around theoretical, and heuristic considerations. It also has to be provided by means of a third independent measurement system. It can be provided off-line, with respect to the functioning of the GT system(s).

In the case of indoor GT, we thought that we had no alternatives to base the validation on quantitative hand-measuring, to convince our users about the accuracy of the GT system(s). We proceeded in the validation with the following approach: to determine the position in the world of a few points fixed on the robot, basing on manual measurements, and then combining these manual estimates into a robot pose, which is then compared to the GT system output.

Hand-laser distance measuring device

In order to reach the best accuracy for the validation, we based all distance measures on the usage of an instruments that is in normal use in civil engineering: a laser range finder measuring along a single line, i.e., not scanning. The model used is a "Bosch DLE50 Professional", see pictures below. We always used it with its origin set in the back right corner. As the position of the emitter and receiver are slightly shifted left with respect to the back right corner, we even checked that its output was corrected for that. We also verified its accuracy, comparing its output on a distance similar to the ones we will measure with it. As we had the need, see below the explanation, of measuring at a certain fixed height, we mounted the range finder on a simple structure. We also had the need, to speed up the measurements, to rotate the structure without translating it, which is not an easy task when the rotating object has no fixed point, so we assembled a sharp piece of metal

to the structure, allowing the structure to rotate around its contact point, see pictures below The technical specifications about the accuracy of the instrument are:

- 5cm - 50m measuring range
- ±1.5mm/<30m / ±5mm/50m accuracy
- 0.5-4.0 seconds measuring time

(Left) The hand-laser used for the validation. (Right) Hand-laser mounted on a mechanical structure, to take measurements at the same height.

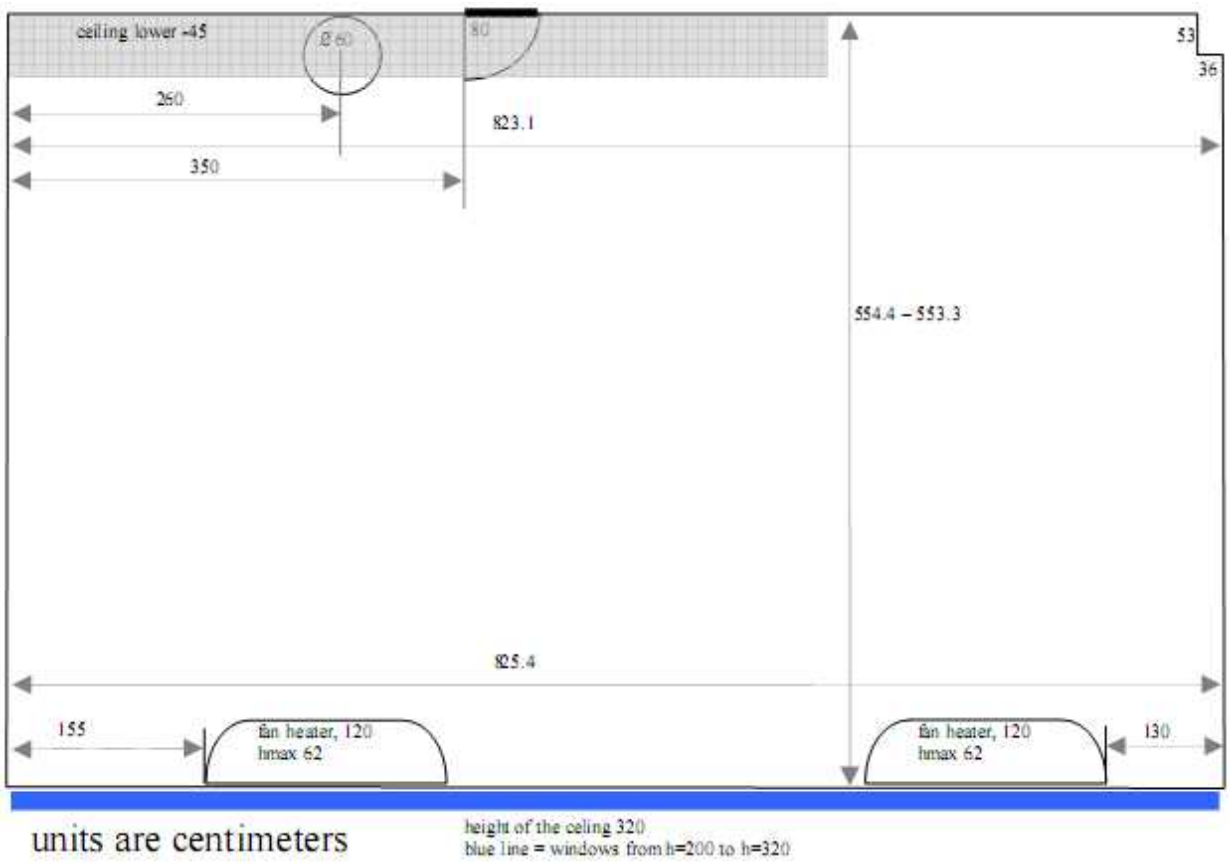
(Left) Particular of the mechanical structure, easing a pure rotation about the floor points. (Right) Usage of the hand-laser.

Definition of the validation frame (Vframe)

The validation frame (Vframe) is the frame with respect to which all validation data are referred. As such, during the planning of the validation activities we decided to base on a corner and the walls of the GT room, see photo and drawing below, expecting the walls of the room to be approximatively orthogonal. In such a way we would have had all room data referenced to this room Vframe.

Pictures of the GT room. (Top left) The corner in view is the bottom-left corner of the bird's eye view. (Top right) The corner in view, on the right of the robot, the top-right corner in the bird's eye view. (Bottom left) The corner in the view, with a desk encircled by carton boxes and sheets, is the top-right corner in the bird's eye view. (Bottom right) The corner in the view is the bottom-right corner in the bird's eye view.

Unfortunately, it turned out that the two shortest wall, see the room map below, are not parallel by an amount we considered not negligible, more than 2cm of difference between 2 points 3m away each other, along the shortest wall.

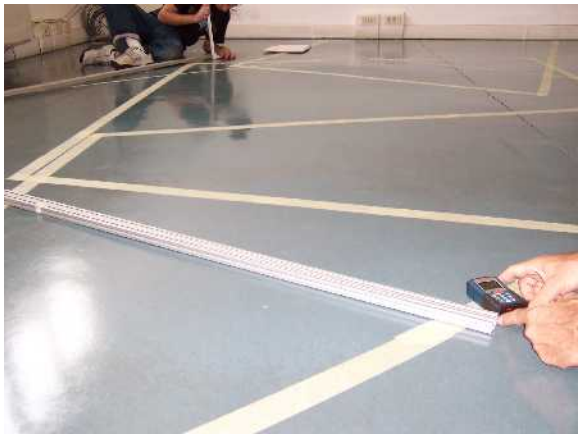


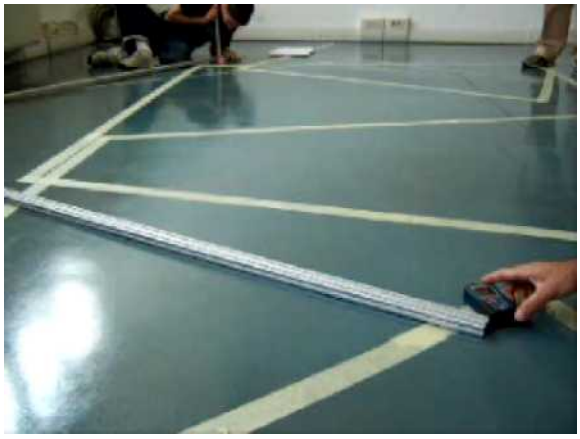
Bird's eye view of the GT room, reporting the measures showing that the 2 short walls are not parallel.

We therefore had to introduce an explicit Vframe in the room, and to refer to it all points, see photo below. This Vframe has been physically built with two 2.5m long poles; they are kept together with a $\pi/2$ fixture, but, given the relative length of the poles with respect to the fixture, we had to verify their orthogonality by hand, see photos below. The origin as well as the endpoints of the poles, defining the axes of the Vframe, have been drawn on the floor with a felt-tip pen, see photos below. For the subsequent referencing needs, introduced below, we also drawn the point symmetric to the origin, i.e., the fourth corner of the square built by flipping the two poles, i.e., the Vframe, along the diagonal of the square, see photo below.

The data read from the hand-laser were both confirming the length of the poles, and also confirming the orthogonality of the Vframe in both poses (origin and 4th point): the actual measure was 3.536m, while sqrt(2) * 2.5m = 3.535533.

Vframe, x axis is the one from middle-right to top-center; y axis is from middle-right to bottom-left; the bottom-right hands in view are at about (0, 2.5), the laser spot is at (2.5, 0).

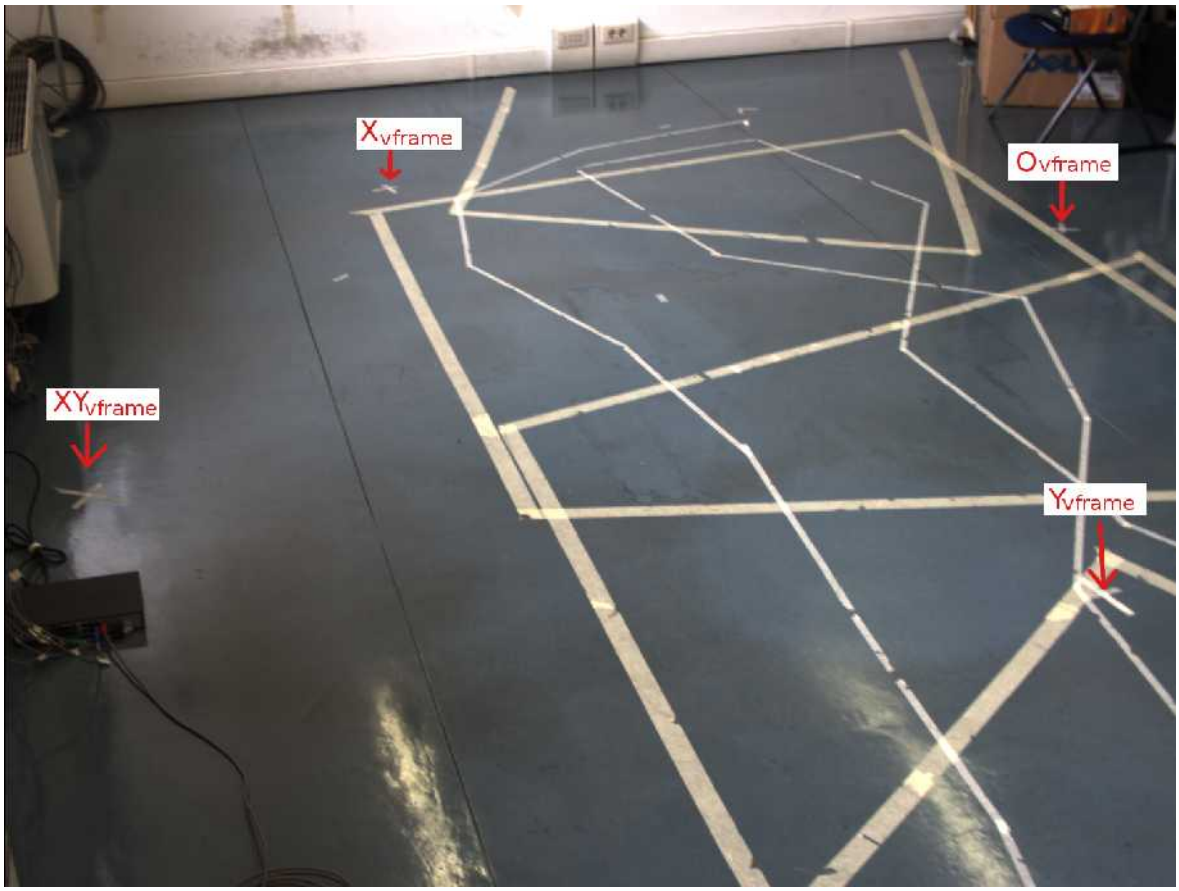




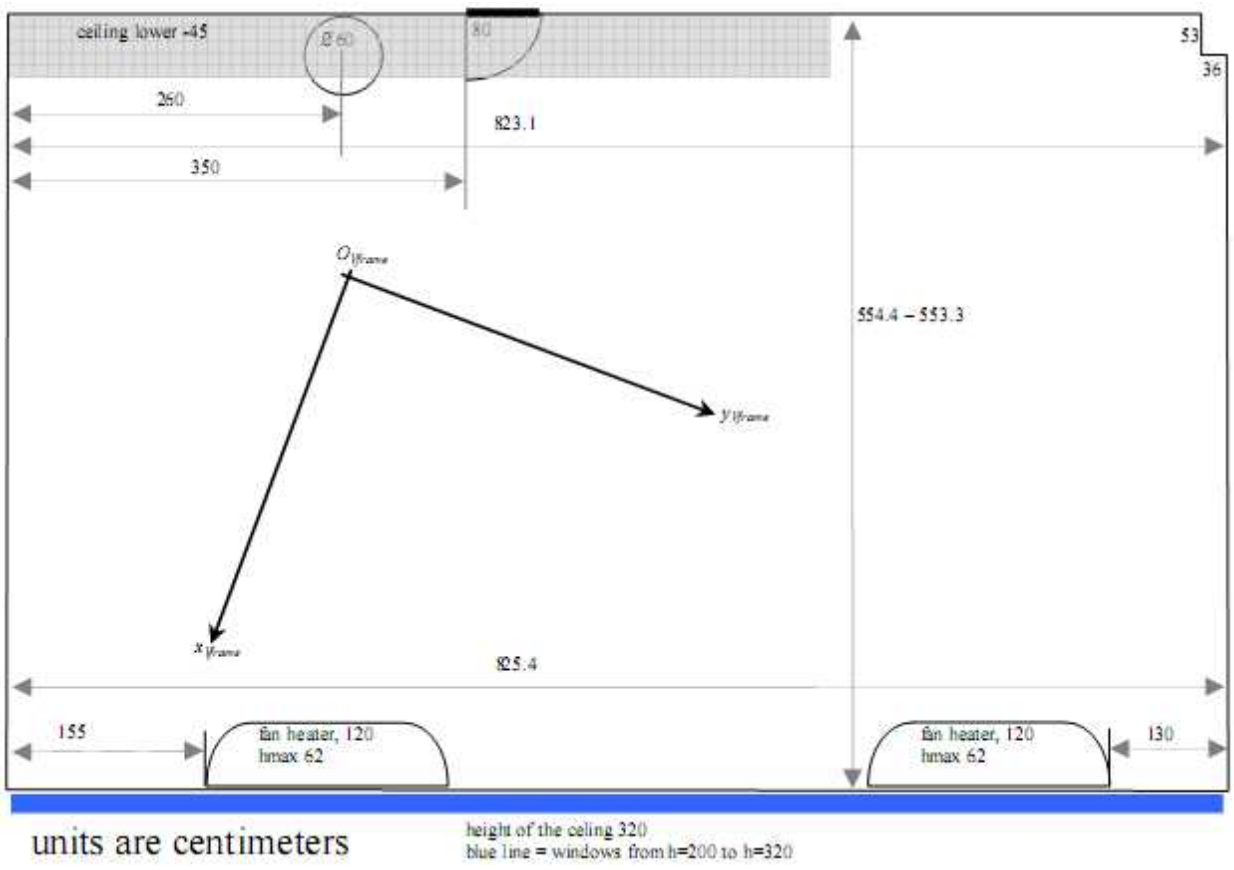
(Left and right) Verification of the orthogonality of the Vframe, this has been done both when the corner was the Vframe origin, and when it was flipped, to define the 4th point.

Pictures of the Vframe points from a very near distance, with paper-tape on the floor and pen markings on it.

Pictures of the Vframe points from a very near distance, with paper-tape on the floor and pen markings on it.



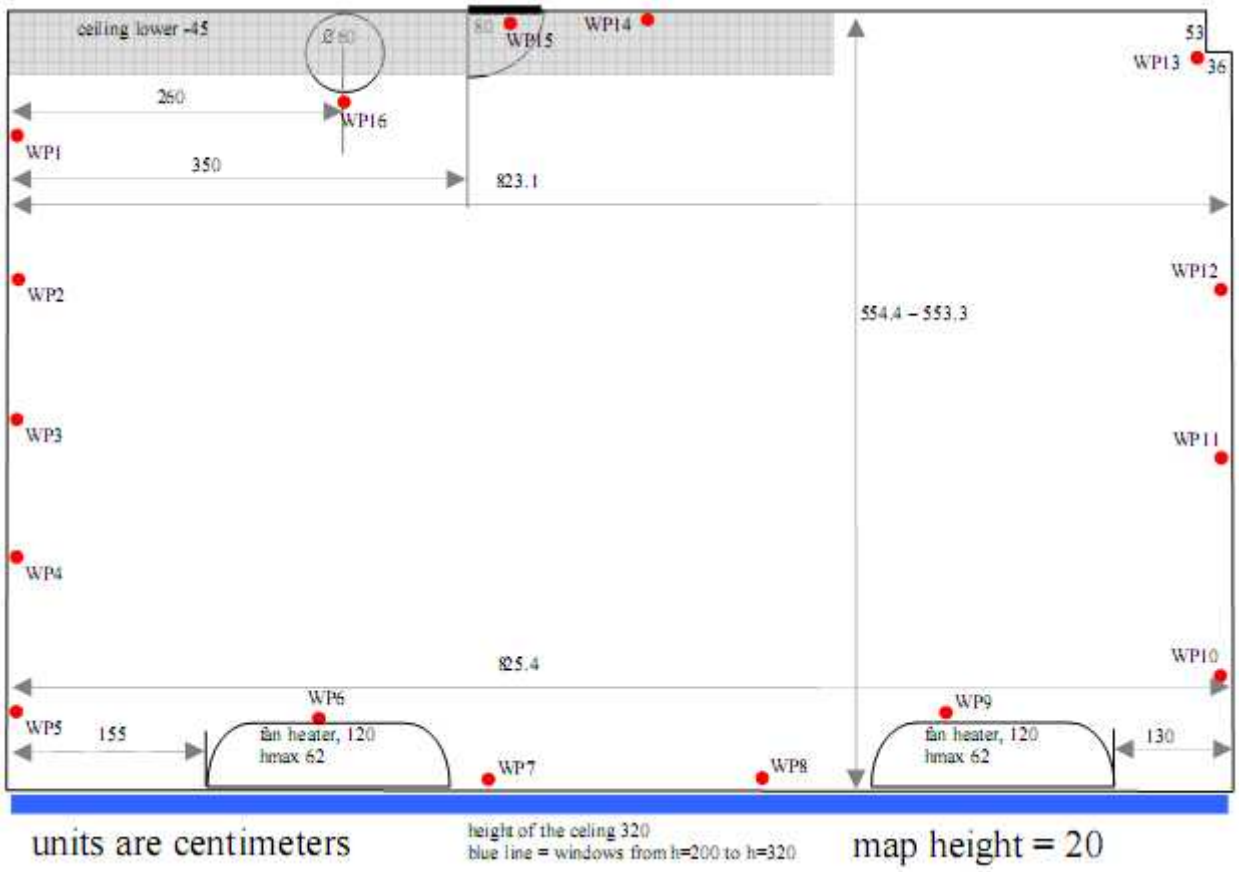
Overview of the Vframe points.



The GT room from above, with the approximate position of the Vframe.

Definition of world-points

Given that the room drawings are not reliable, we decided to reduce the world to just a few points, which we will reference to the above defined Vframe. These points in the GT room are called world-points from now on. They are currently 16, from 1 to 16; they were 18 in origin, but then we had to reduce them, see below. The world-points are roughly distributed all around the room (see figure below), in order to avoid any directional bias during their usage, in validation. If we didn't distribute the points all around the room, then, being the validation of the robot pose is more or less similar to a triangulation, the errors in such computation, because of its geometry, is much larger in depth than in other directions; see e.g., the classical paper by L. Matthies and S. Shafer, for further details; therefore a more or less uniform distribution of the world-points in the horizon reduces the errors.



GTroom bird's eye, with the approximate position of the world-points

(Left) Average distance from two world points. (Right) A zoom on world-point n. 12.

Determination of the coordinates of the world-points

In order to reference the world-points with respect to the Vframe, we measured the distances between all world-points and the Vframe points. We recall that we defined 4 Vframe points, to obtain a reliable determination of the Vframe coordinates of the world-points; the 4 Vframe points are: the Vframe origin

(O_{Vframe}), a point along the Vframe x-axis 2.5m away from the origin (X_{Vframe}), a point along the Vframe y-axis 2.5m away from the origin (Y_{Vframe}), and a point at 2.5m on both Vframe x and y (XY_{Vframe}).

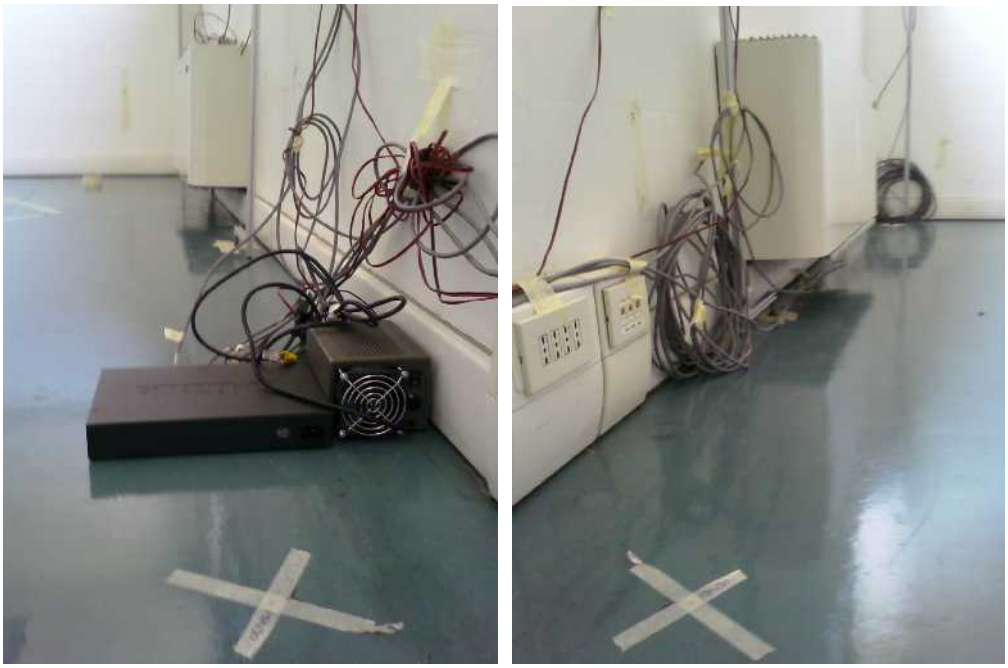
During the measurements, in order to avoid errors due to the baseboards, signal and power cables, etc. (see picture below) we decided to measure points at a certain heights, and prepared also the structure for keeping the hand-laser at that height, see the picture above for such structure. The height was a little bit more than 0.2m, so to match the height of the robot LRFs; therefore the map of the world-points is defined at this height (were there are interesting features for the laser-based GT system).

Pictures of camera cables, baseboards, power sockets, and other things, potentially disturbing when measuring at a smaller height than that of the robot LRFs.

For each world-point we therefore can have a maximum of 4 measures. During the measuring activity we discovered that 2 world-points, out of the original set of 18, were not visible from all the 4 Vframe points (1 had 3 measures, the other just 2); we simply discarded these 2 points, therefore reducing the cardinality of the set of world-points to 16, see picture below.

After measuring all distances, we determined, for each world-point, a Least Squares estimate of the intersection of the 4 circles corresponding to the 4 measures (one world-point to each of the 4 Vframe points) to compute their position with respect to the Vframe. For speeding up the development we implemented such estimate by means of an extended Kalman filter (EKF), which basically performs a gradient descent over the non-linear measurement equation.

Being this a local method it might not converge, if not properly initialized, and indeed it happened when we were initializing it in the origin of the Vframe. To speed up the process, instead of developing an approach less prone to the local minima problem, we just computed the intersection of the first 2 circles (which is more or less like to select randomly), each one built on one of the 4 measures. This gives out 2 solutions; we selected the first one (which is again more or less like to select randomly) and then gave the filter this solution as initialization. We were expecting troubles, for some world-point, as we could have generated a bad initial guess, which might have led the filter to diverge. We therefore checked all the outcomes of the filter, and we can say that the filter never diverged, for all our world-points.



Pictures showing that some world-points were not visible from all the four Vframe-points, e.g. in the view from XY_{Vframe}, a point between WP5 and WP6 (right picture) and a point between WP9 and WP10 (left picture), were hidden by the fan heaters.

The state equation used in this filter just says that the unknowns, i.e., the coordinates of the world-point with respect to the Vframe, are constant. The measurement equation is stating that the measure is a noisy observation of the radius of a circle; the center of the circle is in the *i*-th Vframe-point and the circle is passing through the point defined by the unknowns. The filter is iterated 20 times on the same data, each time restarting the sigma[0] to its a priori value and taking the previous estimate as x[0], as it is usually done in EKF for gradient-descent-like zeroing of linearization errors. The other settings are as follows:

1. σ-extrema-axes-Vframe (both for *x* and *y*)= 0.002m;
2. σ-hand-laser-measure = 0.008;
3. σ[0] = 100, for both *x* and *y*.

The Matlab code is:

```
for i = 1:20
            p=p_old;

for k = 1:nmeas

%d = distance from the world-point and the Vframe-point
%xw1 = x coordinate of the Vframe-point
%yw1 = y coordinate of the Vframe-point
            d=input(1,k);
            xw1=input(2,k);
            yw1=input(3,k);

%R = covariance of d, xw1, yw1
            R=diag([input(4,k),input(5,k),input(6,k)]).^2;

%x1 = x coordinate of the world-point in the Vframe
%x2 = y coordinate of the world-point in the Vframe
            x1=x(1,1);
            x2=x(2,1);

%h = measurement equation
            h=(xw1-x1)^2+(yw1-x2)^2-d^2;

%H Jacobian of the measurement equation w.r.t. the state (x1,x2)
            H=[-2*(xw1-x1) -2*(yw1-x2)];

%W Jacobian of the measurement equation w.r.t. the measures (d,xw1,yw1)
            W=[-2*d 2*(xw1-x1) 2*(yw1-x2)];

            S=H*P*H'+W*R*W';
            K=(P*full(H)')*inv(S);

            P=P-K*S*K';
            x=x+K*(-h);
end
end
```

In the following we report tables with hand-laser readings a well as with world-point coordinates.

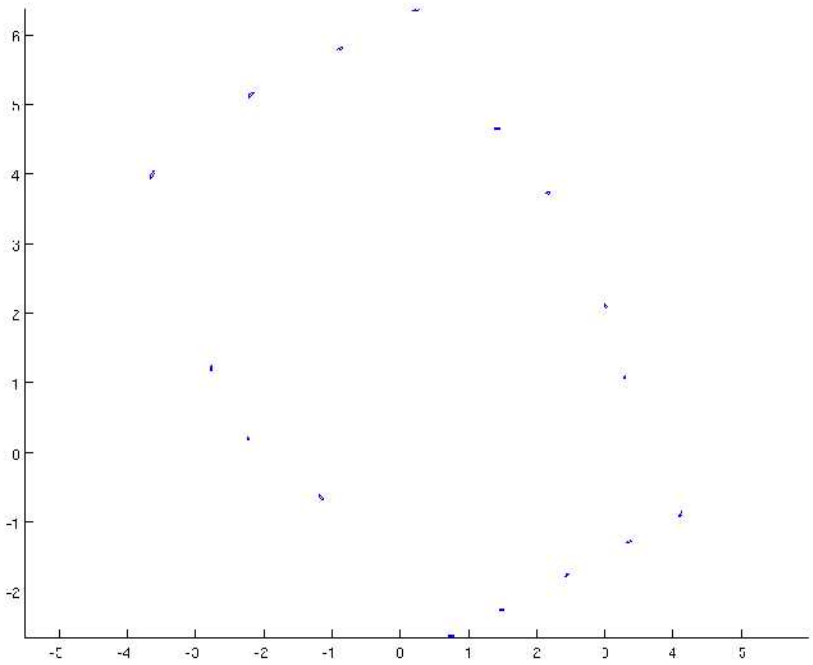
#	***dist to O_{Vframe}***	***dist to X_{Vframe}***	***dist to Y_{Vframe}***	***dist to XY_{Vframe}***
1	2.740	5.190	3.165	5.430
2	2.698	4.978	2.466	4.862
3	3.008	4.904	1.755	4.259
4	3.587	5.051	1.533	3.876
5	4.200	5.321	1.828	3.750
6	3.465	3.578	1.341	1.615
7	3.678	3.037	2.162	0.640
8	4.315	2.488	3.743	1.273
9	4.872	2.584	4.789	2.410
10	6.372	3.875	6.758	4.483
11	5.868	3.416	6.715	4.731
12	5.581	3.427	6.952	5.378
13	5.392	3.920	7.308	6.301
14	3.012	3.045	5.401	5.415
15	2.231	3.191	4.732	5.248
16	1.322	3.345	3.716	4.824

Table with the measures of each world-point from the 4 Vframe points.

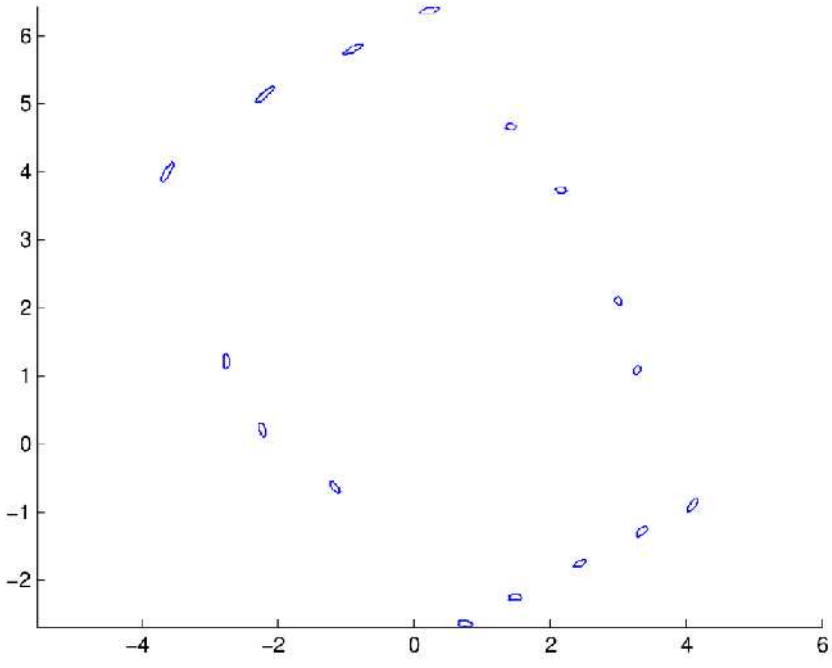
#	x_{Vframe}	y_{Vframe}
1	0.746335	-2.636647
2	1.487105	-2.251523
3	2.440868	-1.755774
4	3.349824	-1.279703
5	4.101903	-0.889099
6	3.287522	1.088277
7	3.010098	2.106208
8	2.165534	3.728941
9	1.420934	4.660137
10	0.234251	6.367675
11	-0.889505	5.798842
12	-2.188851	5.134402
13	-3.622194	3.993617
14	-2.760034	1.212409
15	-2.226985	0.209837
16	-1.160521	-0.637959

Table with the coordinates of each world-point in the Vframe, the coordinates of these world-points will be extensively used in the subsequent validation activities.

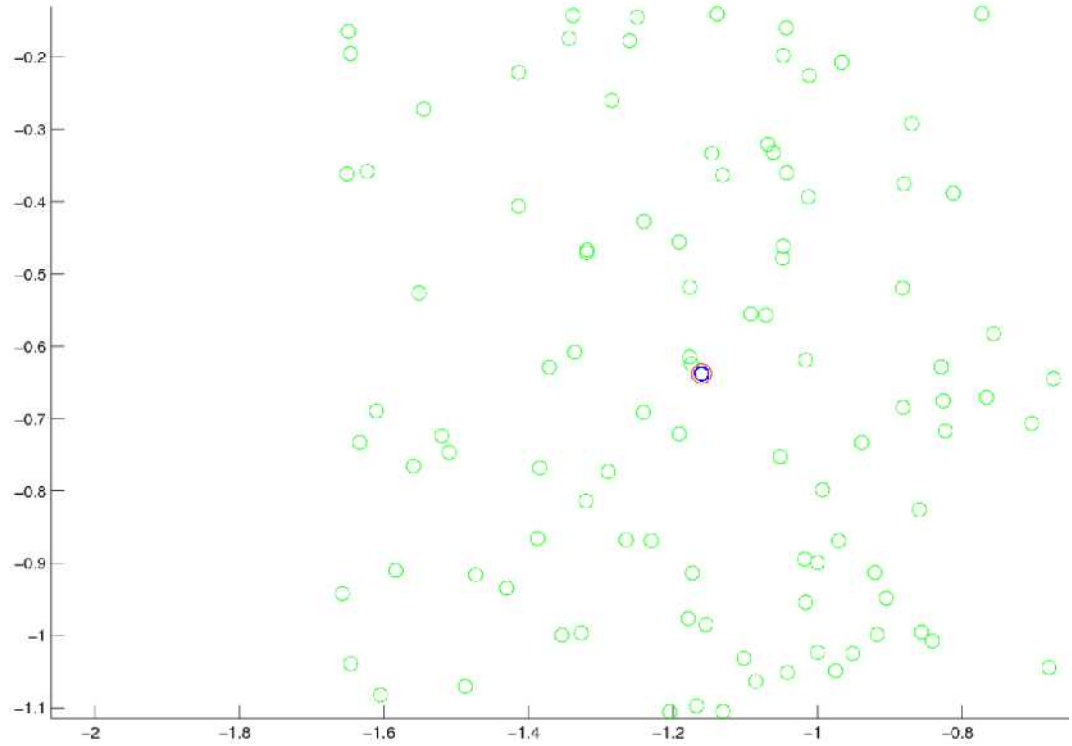
The accuracy on the coordinates of the world-points are represented by the covariance output by the EKF; as it can be noticed in the figures below, such ellipses are quite small: about 1 cm.



Uncertainly ellipses (3 sigma) of world-points



Uncertainly ellipses (3 sigma) of world-points, covariances were multiplied by ten in order to make them more visible.



Plot showing the absence of other solutions for the filter: the blue circle is the real position (one of the world-points) and the green circles are the initial guesses that correctly output the estimate position (the one in the red circle). Scale is in meters.

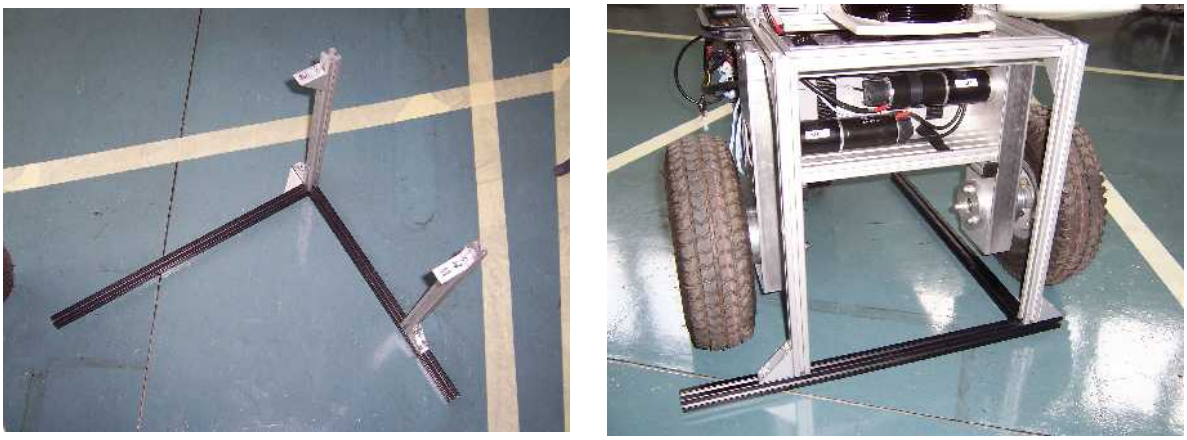
In order to increase the trust a generic user of our datasets would give to our GT, we need to convince she/he that our world-points are accurately estimated by our method. In the Figure above we show (heuristically) that all (i.e., a lot of) initializations about the used one turns into the same solution, i.e., the filter always converges to the same solution. Notice that we had both the initial solutions converging to the correct one. Moreover, this happens for all the world-points. This "proves" (actually, being the filter an extended KF, there is no theoretic proof about its convergence) that there is only one local minimum in that "region of attraction", which is then "the correct one", unless one believes the measurement equation to be questionable.

Robot-frame

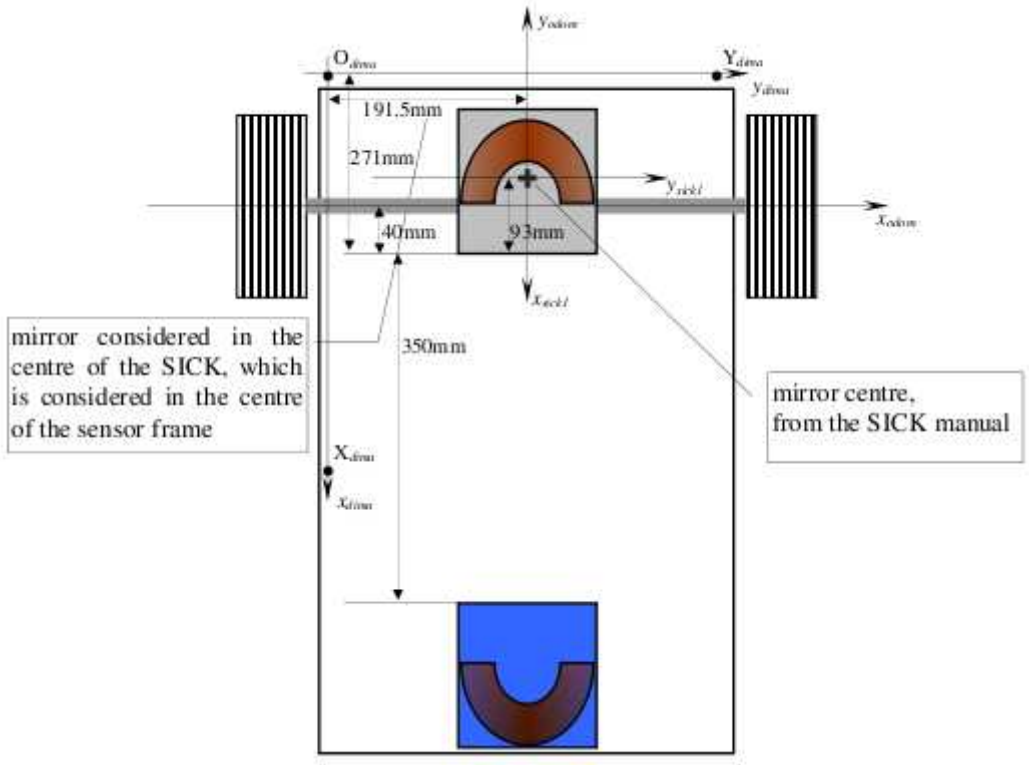
The robot-frame is the frame whose pose is given in output by the validation; its pose is given with respect to the Vframe. For what concerns the robot-frame, we proceeded in an a bit unusual way: instead of using the so-called odometric frame, i.e., the frame to which odometry is referred, we decided to mount on the robot a mechanical frame that we called "dima" (dima it's just the italian word for it), see pictures below. The odometric-frame is at the centre of the wheel baseline, usually with the y axis pointing in the forward direction and the x axis to the right of the robot, and it is therefore difficult to use as a practical frame for performing measurements.

The introduction of the dima aims at easing the collection of the validation data, by allowing to use the dima for drawing a frame on the floor with a felt-tip pen, which is the robot-frame for that robot pose. We mark also the pose number, and then we move the robot to somewhere else, e.g., the next pose. At this point we can start the manual measurements for the current pose. For each pose we need to measure 3 robot-frame

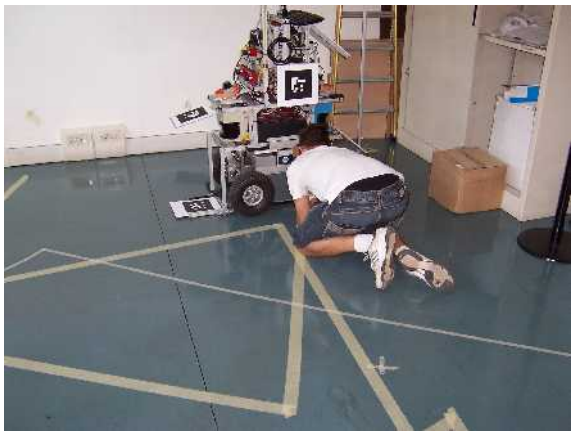
points, with respect to the world-points: O_{dima}, X_{dima}, Y_{dima}, see the pictures below. X_{dima} is about 0.6m along the *x* axis, Y_{dima} is about 0.4m along the *y* axis. The distances between O_{dima}, X_{dima}, Y_{dima} and the world-points will be the input of the determination of the robot-frame pose; these distances will be measured with the hand-laser.



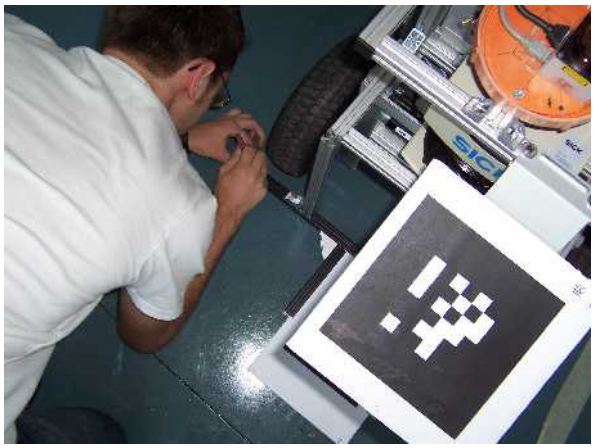
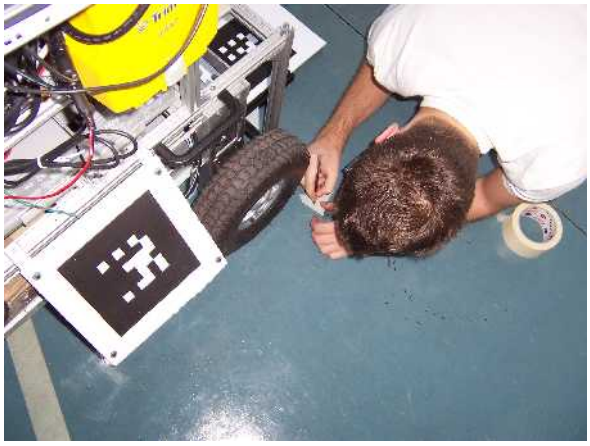
Picture of the dima, before (left) and after (right) being mounted on the robot.



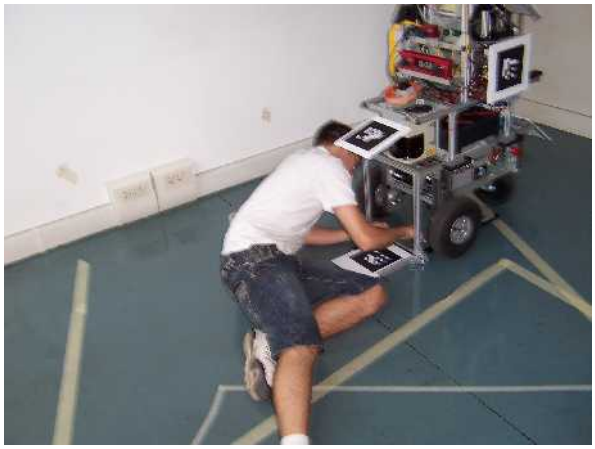
Drawing of the robot-frame and other frames.

(Left) An extremum of the dima x axis, nearby the hand in view. (Right) writing down the on the floor the x axis extremum.

(Left and right) Writing down on the floor the extremum of the dima y axis.

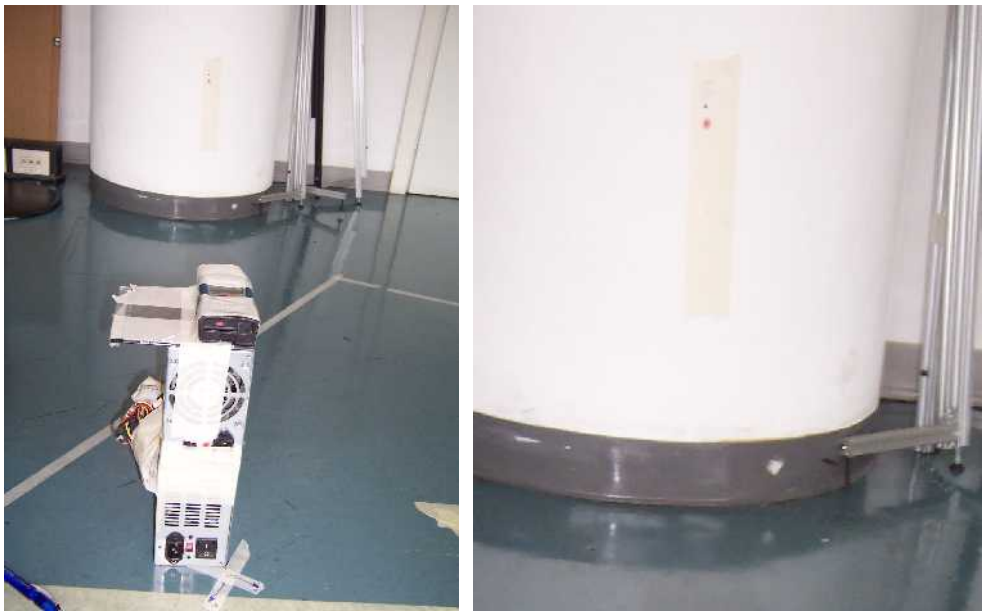



(Left, and, right) Writing down on the floor the origin of the dima frame.

(Left and centre) Taking distance measurements from the 3 robot-frame points to the world-points, with the hand-laser. (Right) Zoom of the central picture, to show the hand-laser spot and the world-point n. 5.

Collection of the validation data

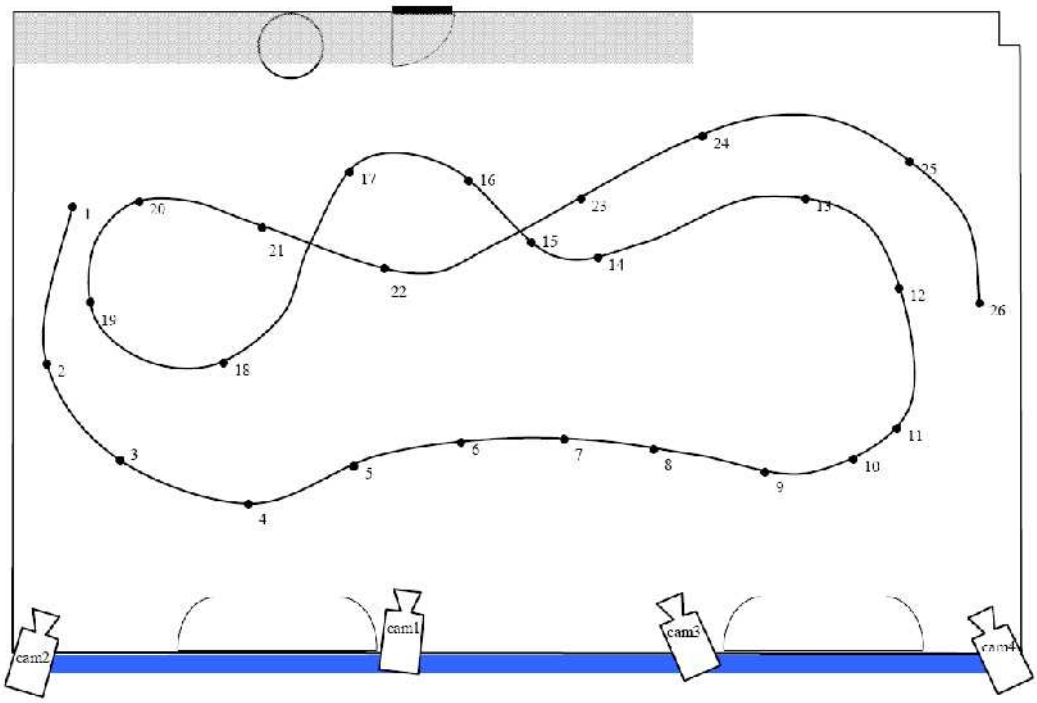
During the measuring of the distances from the robot-frame points to the world-points, we pointed the hand-laser on the wall, and aligned the spot only horizontally with the world-point, see pictures above and below. This was the result of an explicit decision, as it would have been possible to tilt manually the hand-laser, and align the laser spot also vertically. Notice that leaving the structure on the floor usually did not result in the vertical alignment of the spot and world-point, as the floor was not perfectly flat. The decision about not caring about the vertical alignment came from the observation that, on one hand, correcting the tilt by hand was error prone, on the other it was not giving significantly more accurate distance measure; moreover, tilting by hand was making the measurement process much longer.



Picture showing how the actual measurements of the robot-frame point to the world-point were taken. Notice that only the horizontal (and not the vertical) alignment is reached.

When inputting the hand-laser readings into the programs, we were quite attentive to avoid typos: we had always two person for the data entry, one actually reading the hand-laser display, the other typing-in the number, and repeating it to the first one, for double checking.

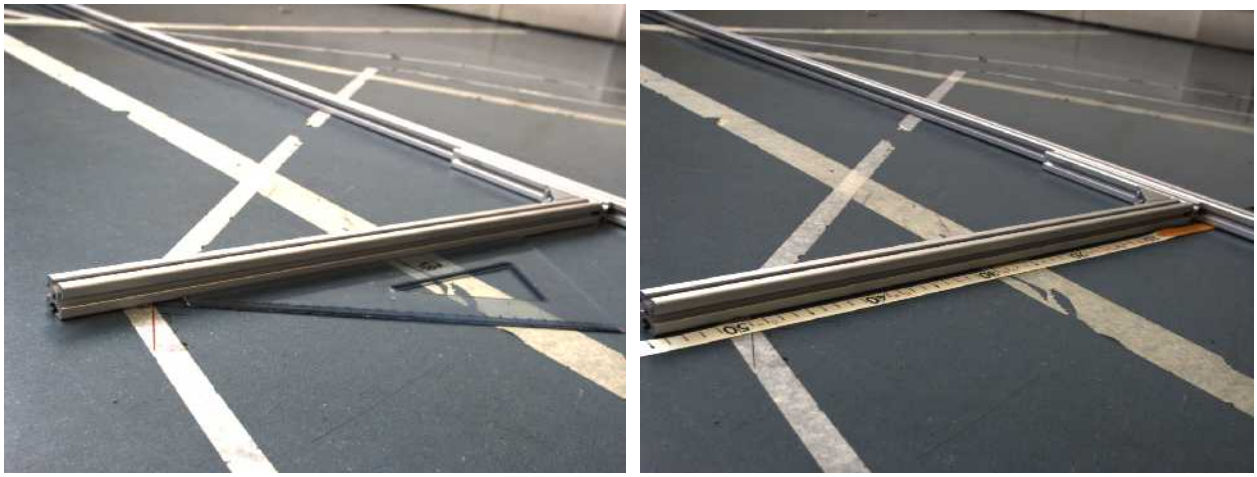
We collected validation data for 26 poses along a path in the GT room. The path is roughly depicted in the picture below.



The GT room and the approximate path and poses used for validation.

For what concerns the determination of the validation data, i.e., the robot poses, which are represented by the roto-translation of the robot-frame with respect to the Vframe, we developed another non-linear Least Squares estimator, again by means of an EKF, whose state equation is again expressing the time constancy of the unknowns, while the output equation states that the noisy measurement is the distance between the robot-point from the world-point. Both are expressed in the Vframe, so that the robot-point includes the unknowns.

Also this filter might in principle diverge, as we did not know in advance how large was the region of attraction of the solutions. Again, we took the fastest to implement approach, i.e., we provided a reasonable initial guess and then checked the outcomes. The initial guess was determined in one of two ways, depending on the distance of the robot-frame from the Vframe. One way was by counting the steps and / or using a flexible meter, and the alignments were obtained by checking perpendicularity by eye. The other way was different only in that it was basing on a mechanical frame (see pictures below) in order to determine the perpendicular to the x axis of the Vframe, passing through the origin of the robot-frame. Then the steps were counted on both axes (or the flexible meter was used) and turned in a distance in meters. The orientation was always determined roughly "by eye measure". Again, we were concerned that the initial guess might have been too poor to allow the filter to converge, but it always worked well, i.e., the filter never diverged.



Pictures showing how the initial guesses based on the mechanical frame were taken; other initial guesses were taken by "eye determination" of the intersection, with the x axis, of a line parallel to the y axis and passing through the origin of the robot-frame; such intersection determined, along the x axis, the x coordinate of the origin of the robot-frame, while the length along the line was the y coordinate; the orientation was also the result of "eye determination".

The filter is iterated 20 times on the same data, each time restarting the sigma[0] to its a priori value and taking the previous estimate as x[0], as it is usually done in EKF for gradient-descent-like zeroing of linearization errors. The other filter settings are:

1. σ-hand-laser-measure = 0.008;

2. Covariance robot-frame points = |0.010 0.000|
   |0.000 0.010|

3. Covariance world-points = |0.000004 0.000000|
   |0.000000 0.000004|

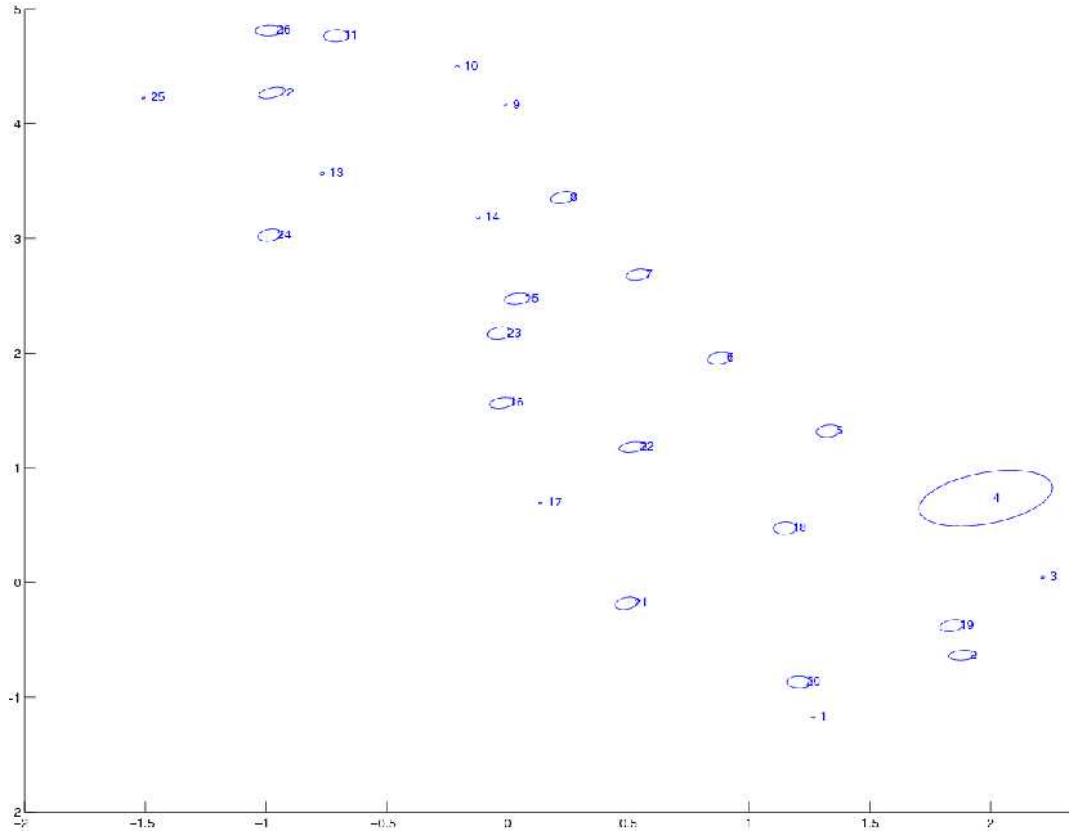
4. Covariance robot pose [0] = |1000.000 0.000 0.000|
   | 0.000 1000.000 0.000|
   | 0.000 0.000 100.000|

The Matlab code is:

```
for i = 1:20

    p=p_old;

for k = 1:nmeas

%d = distance between the Vframe and the robot-point
        d=input(1,k);

%xw1 = x coordinate of the Vframe point
%yw1 = y coordinate of the Vframe point
        xw1=input(2,k);
        yw1=input(3,k);

%xr1 = x coordinate of the robot-frame point
%yr1 = x coordinate of the robot-frame point
        xr1=input(4,k);
        yr1=input(5,k);
```

```
%R = covariance of d, xw1, yw1, xr1,yr1
        R=diag([input(6,k),input(7,k),input(8,k),input(9,k),input(10,k)]);

%x1 = x coordinate of roto-translation between Vframe and robot-frame
%x2 = y coordinate of roto-translation between Vframe and robot-frame
%x3 = theta coordinate of roto-translation between Vframe and robot-frame
        x1=x(1,1);
        x2=x(2,1);
        x3=x(3,1);

%h = measurement equation
        h=(xw1-(xr1*cos(x3)-yr1*sin(x3)+x1))^2+(yw1-(xr1*sin(x3)+yr1*cos(x3)+x2))^2-d^2;

%H Jacobian of the measurement equation w.r.t. state (x1,x2,x3)
                                        H=[-2*xw1+2*xr1*cos(x3)-2*yr1*sin(x3)+2*x1,-
2*yw1+2*xr1*sin(x3)+2*yr1*cos(x3)+2*x2,(2*(xw1-xr1*cos(x3)+yr1*sin(x3)-
x1))*(xr1*sin(x3)+yr1*cos(x3))+(2*(yw1-xr1*sin(x3)-yr1*cos(x3)-x2))*(-
xr1*cos(x3)+yr1*sin(x3))];

%W Jacobian of the measurement equation w.r.t. measures (d,xw1,yw1,xr1,yr1)
                      W=[-2*d,2*xw1-2*xr1*cos(x3)+2*yr1*sin(x3)-2*x1,2*yw1-2*xr1*sin(x3)-
2*yr1*cos(x3)-2*x2,-(2*(xw1-xr1*cos(x3)+yr1*sin(x3)-x1))*cos(x3)-(2*(yw1-xr1*sin(x3)-
yr1*cos(x3)-x2))*sin(x3),(2*(xw1-xr1*cos(x3)+yr1*sin(x3)-x1))*sin(x3)-(2*(yw1-
xr1*sin(x3)-yr1*cos(x3)-x2))*cos(x3)];

        S=H*P*H'+W*R*W';
        K=(P*H')*inv(S);

        P=P-K*S*K';
        x=x+K*(-h);

end
end
```

The accuracy on the robot pose will be represented by the covariance output by the EKF above.



Uncertainly ellipses (3 sigma) of world-points, covariances were multiplied by ten in order to make them more visible.

In order to increase the trust of a generic user of our datasets would give to our GT, through our validation, we would like to proof that our robot poses are accurately estimated by our method. We verified that the filter always converges to the same solution, i.e., all of the initializations in a radius of about half-meter about the used one, turns into the same final solution, which means that there is only one local minimum in that "region of attraction", which is then "the correct one", unless one decides to question the measurement equation.

The results are presented hereafter.

pose 1

	hand robot-frame	validation robot-frame
x	1.276	1.2653
y	-1.169	-1.174
th	3.4034	3.4456

pose 2

	hand robot-frame	validation robot-frame
x	1.89	1.8838
y	-0.64	-0.6341
th	4.014	4.00131

pose 3

	hand robot-frame	validation robot-frame
x	2.227	2.2171
y	0.024	0.0456
th	4.4506	4.4645

pose 4

	hand robot-frame	validation robot-frame
x	1.97	1.9801
y	0.73	0.7345
th	5.1487	5.16

pose 5

	hand robot-frame	validation robot-frame
x	1.35	1.3285
y	1.33	1.3213
th	5.497	5.5105

pose 6

	hand robot-frame	validation robot-frame
x	0.89	0.8781
y	1.957	1.9556
th	5.235	5.1869

pose 7

	hand robot-frame	validation robot-frame
x	0.54	0.5397
y	2.67	2.6828
th	5.41	5.1143

pose 8

	hand robot-frame	validation robot-frame
x	0.23	0.229
y	3.347	3.357
th	5.1487	5.1503

pose 9

	hand robot-frame	validation robot-frame
x	0	-0.0079
y	4.164	4.1598
th	5.0615	4.9086

pose 10

	hand robot-frame	validation robot-frame
x	-0.14	-0.2102
y	4.502	4.4968
th	5.759	5.488

pose 11

	hand robot-frame	validation robot-frame
x	-0.74	-0.7092
y	4.77	4.7668
th	6.1087	6.068

pose 12

	hand robot-frame	validation robot-frame
x	-0.91	-0.9756
y	4.24	4.2708
th	1.57	1.7449

pose 13

	hand robot-frame	validation robot-frame
x	-0.73	-0.7689
y	3.57	3.5655
th	1.9199	1.9824

pose 14

	hand robot-frame	validation robot-frame
x	-0.232	-0.1228
y	3.237	3.1824
th	2.8798	2.7651

pose 15

	hand robot-frame	validation robot-frame
x	0.04	0.039
y	2.471	2.4743
th	1.483	1.4604

pose 16

	hand robot-frame	validation robot-frame
x	-0.02	-0.0228
y	1.574	1.5651
th	1.5359	1.515

pose 17

	hand robot-frame	validation robot-frame
x	0.14	0.1349
y	0.7	0.694
th	2.1817	2.074

pose 18

	hand robot-frame	validation robot-frame
x	1.145	1.1491
y	0.47	0.47468
th	2.618	2.6219

pose 19

	hand robot-frame	validation robot-frame
x	1.84	1.841
y	-0.37	-0.3749
th	2.0595	1.9983

pose 20

	hand robot-frame	validation robot-frame
x	1.2	1.207
y	-0.85	-0.8692
th	6.108	6.06

pose 21

	hand robot-frame	validation robot-frame
x	0.5	0.49296
y	-0.181	-0.18139
th	5.2011	5.1983

pose 22

	hand robot-frame	validation robot-frame
x	0.51	0.5174
y	1.18	1.1805
th	4.5378	4.4479

pose 23

	hand robot-frame	validation robot-frame
x	-0.05	-0.0333
y	2.153	2.1745
th	5.235	5.5533

pose 24

	hand robot-frame	validation robot-frame
x	-0.97	-0.9847
y	2.97	3.0299
th	5.3756	5.3534

pose 25

	hand robot-frame	validation robot-frame
x	-1.52	-1.5081
y	4.221	4.2249
th	4.9742	4.9689

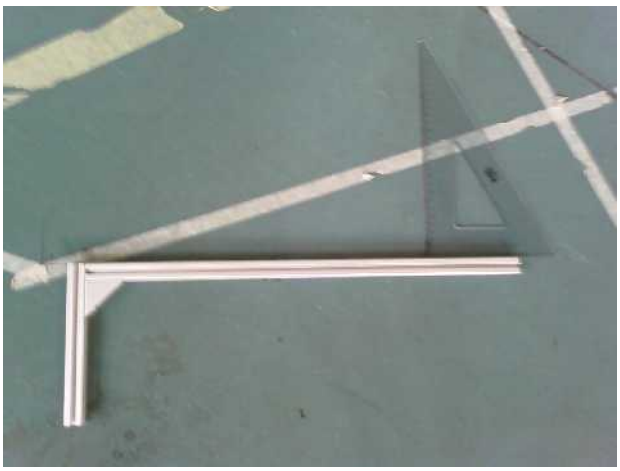
pose 26

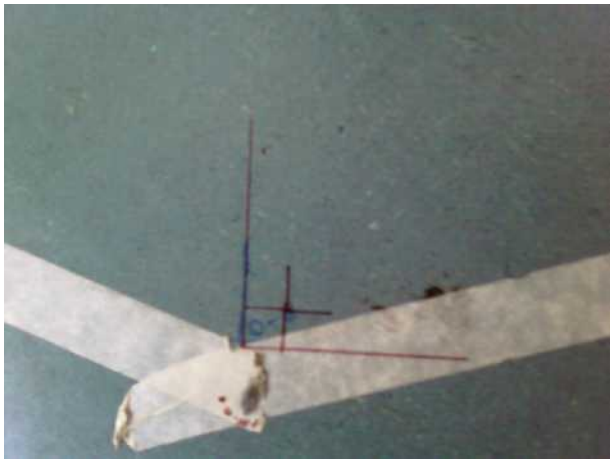
	hand robot-frame	validation robot-frame
x	-0.97	-0.9872
y	4.77	4.8136
th	3.4907	3.5275

Tables with the initial (hand robot-frame) and final estimates of the robot poses.

Accuracy experiments

For what is concerning the accuracy of the validation, we measured the position of two robot poses, one slightly moved with respect to the other, to show that the validation system can actually perceive such small differences, and it is therefore suitable for appreciating the GT errors. The moved robot pose was translated by 1cm on both *x* and *y*, and turned about *z* of 1deg, which means 8.7mm of motion of the extremum of the *x* axis, from its previous position, see pictures below. The results are presented in the following table and allow us to claim that the validation system is capable to appreciate errors of about 1cm.





From top to bottom: drawing of the dima on the floor with a felt-tip pen in pose 18, measuring a movement of 1cm on x-axis and y-axis and one degree on theta, drawing of the new pose 18', close to the previous one.

real ΔX motion	real ΔY motion	real $\Delta\theta$ motion	validation ΔX	validation ΔY	validation $\Delta\theta$	error in X	error in Y	error in θ
0.01	0.01	1.0	0.013192	0.010923	1.013083	0.003192	0.000923	0.013083

Table with results showing the accuracy experiment. Units are meters.

In order to be more than sure about the validation system, we considered another way to "verify" our validation system, and we came out with the idea of using, for a point whose coordinates in the Vframe are known, the vision-based GT system. The idea is to use it just for a verification of the agreement between the two estimates. Notice that we are just testing the agreement, as our trust-able source will be the validation, which we will use to validate the GT system(s).

Therefore, we positioned a chessboard parallel to the *x*-axis of the Vframe, with the top-right corner aligned with the origin. We then took a picture of the chessboard using a calibrated camera (it was camera n. 2 of our set of cameras, see details beyond, in the section on the vision-based GT system). The coordinate of a point of the chessboard were then computed using the calibrated intrinsic parameters of the camera and the same Matlab Toolbox used for camera calibration, these coordinates being relative to the Vframe. We then combined the coordinates of this point with the known (hand-measured) pure translation from the point on the chessboard to the origin of the Vframe. The difference between the estimate obtained from the vision-based GT system and the estimate obtained by the validation system was under the centimeter. In other words the two systems agreed. This result allow us to claim the absence of bias, due to the unlikelihood of the two independent systems (validation and vision-based GT) to provide the same data, i.e., to be subject to the same errors.

"Verification" of the validation by means of a pose estimation computed from an independent system (the vision-based GT system).

Validation of the vision-based GT system

Validation of a GT system means to obtain the GT values from the system, and then to compare them with the values obtained from the validation activities.

In the vision-based GT system there are cameras; the field-of-view (FOV) of these cameras will likely have a narrow superimposition, to increase the part of the robot workspace where GT will be provided. Of course, in the real data-collection scenarios, it will also be possible to have cameras with large FOV superimpositions, but, during the validation of the GT system, this event should be considered as a not worst-case situation. Therefore, our setting replicated, in the GT room, a small chain of cameras with small superimpositions in the FOV of consecutive cameras. The depth of the FOV has been roughly set from 2m to 5m, measured in an horizontal plane, from the camera pin-hole. The tilt angle of the cameras was about $-\pi/4$ from the horizontal, i.e., pointing downward, see pictures below.

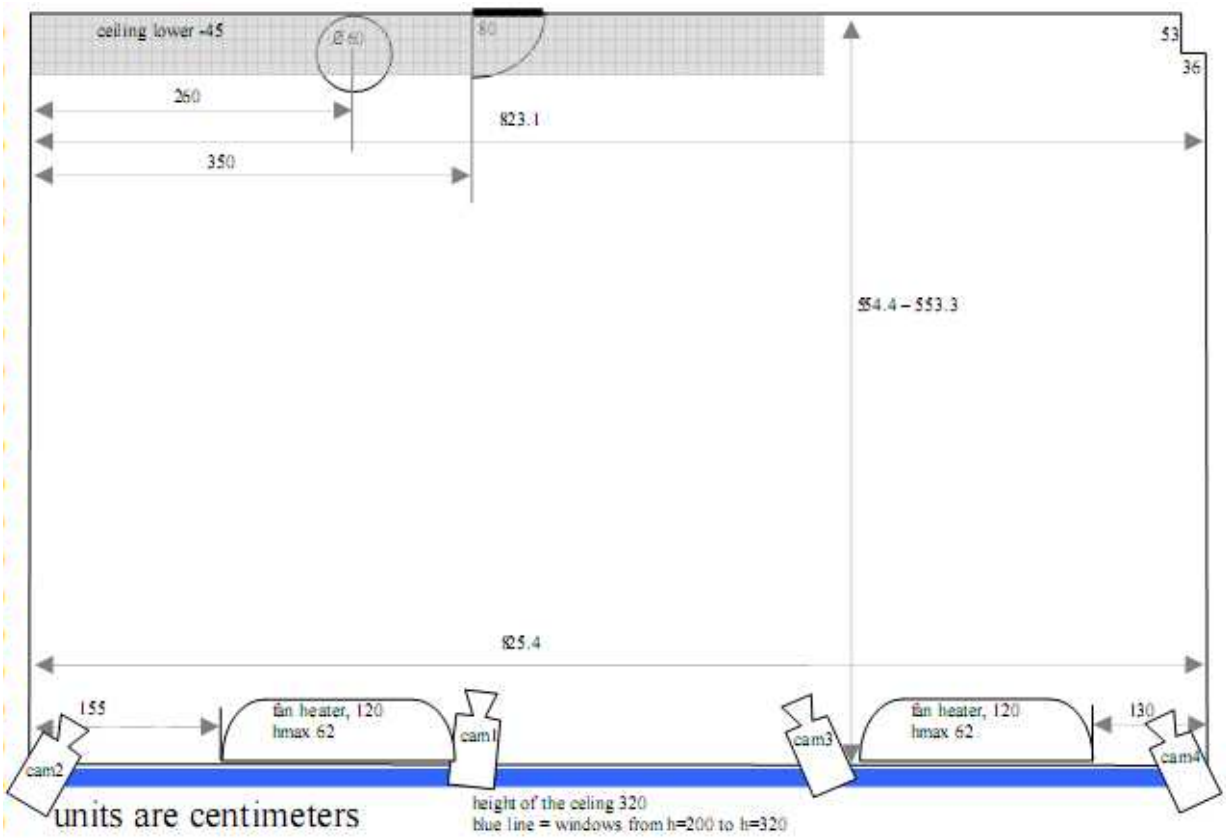
As the GT system has to provide its output in a single reference frame, the GTframe, there will be a roto-translation matrix from each camera to the first one in the chain, and then from the first camera to the GTframe. These matrices will be obtained by properly chaining the roto-translations between adjacent cameras along the chain, up to the first one, and then composing them with the first-camera-to-GTframe matrix.

In order to compare the output of the GT system with the validation, we also need to refer these two estimates of the robot state to the very same reference system. We decided to put the GTframe in coincidence with the Vframe, and this can be done, see below for the explanation. Therefore, from now on, the two systems will be mentioned with the same name, i.e., GTVframe.

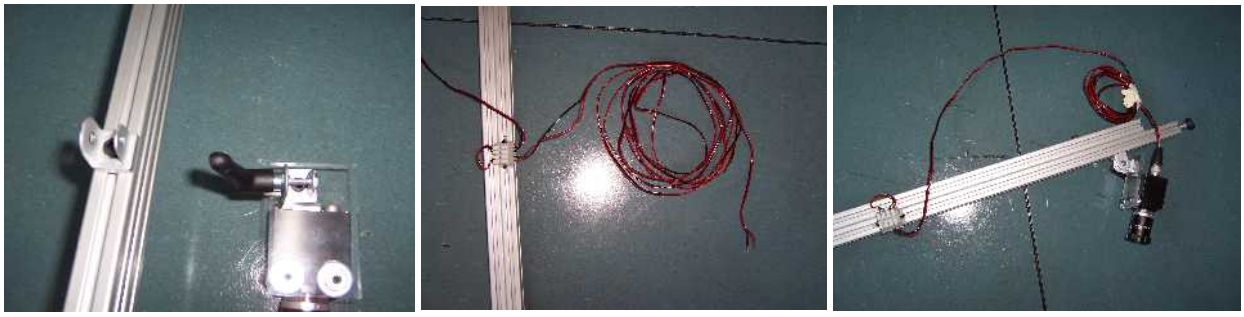
Physical preparation of the system

In order to have the vision-based GT system ready for the work we have to:

1. define the position of the cameras so that there is a narrow common field of view between adjacent cameras; this is to mimic the real GT setup; see picture below;
2. mount all cameras on a rigid post structure, see pictures below showing fixtures and the mounting process;
3. connect every camera and the GT workstation to a Gigabit switch (we bought cameras of the Gige technology), and also to the power supply, see pictures below;
4. check that every point in the GT area is visible at least from one camera, see pictures above for an overall view of the GT room, and notice on the floor the paper-tape representing the FOV of each camera;



GT room with the approximate position of the 4 cameras, replicating, at a small scale, the vision-based GT system and working conditions.

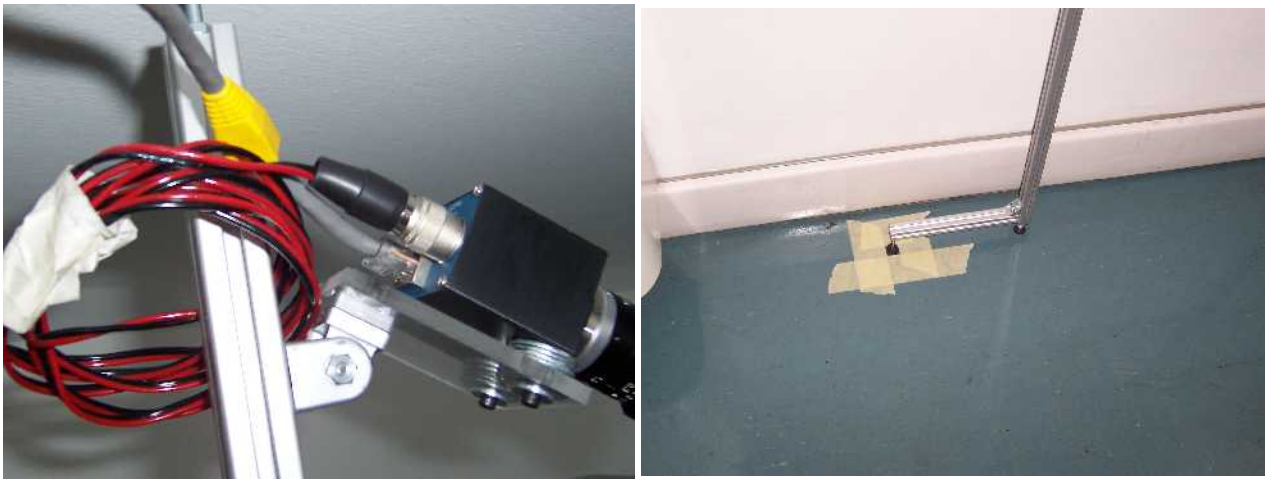


Connecting the power of one of the four cameras and fixing the camera on a pole.

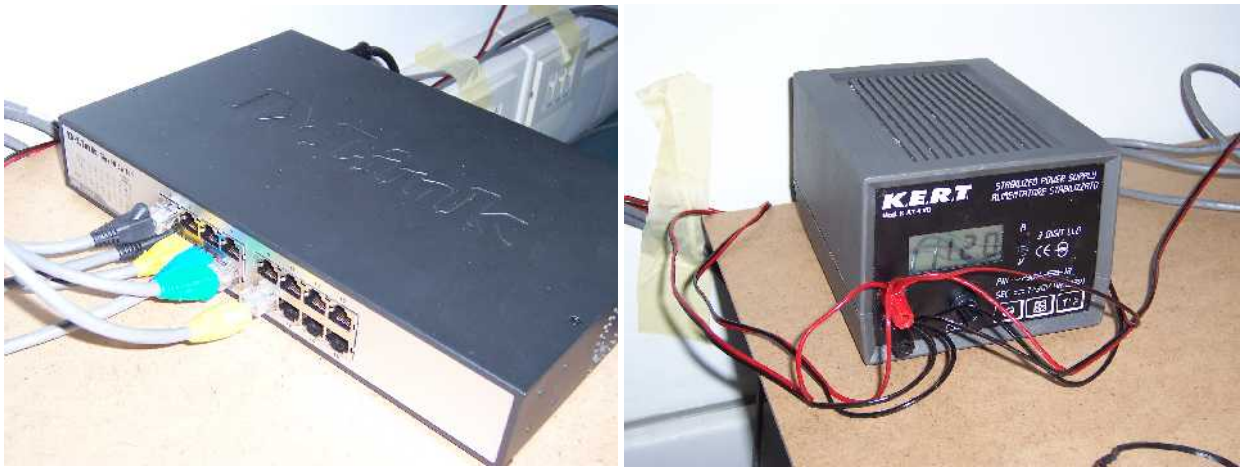
Positioning a pole taking care to correct every possible skew with the spirit level.

Fixing a pole to the floor, both to avoid rotation (left) and translation (right).

Overview of a mounted pole (left) and particular of a camera on it (right).



A camera connected from behind (left) and pieces of paper-tape around the pole to detect undesired rotations (right).

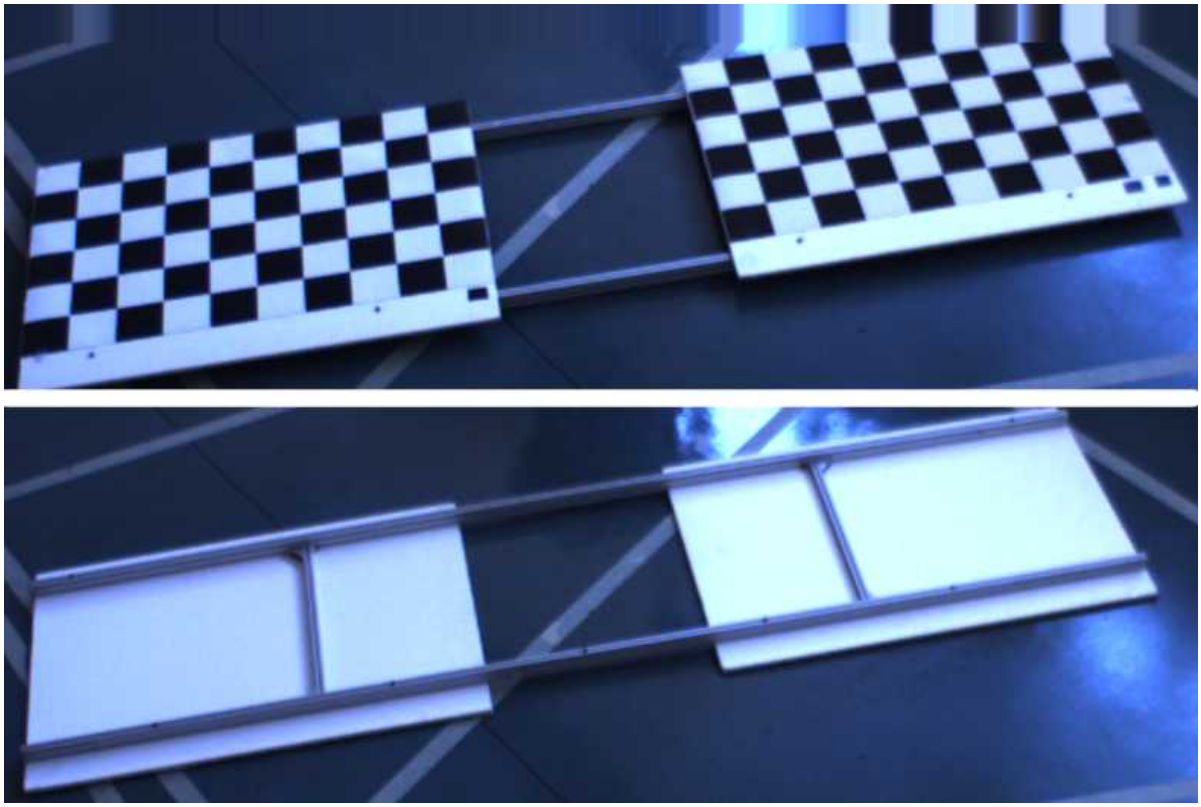


The GigaBit switch used to connect the 100/1000 ethernet cameras (left) and the power unit to which all cameras were connected.

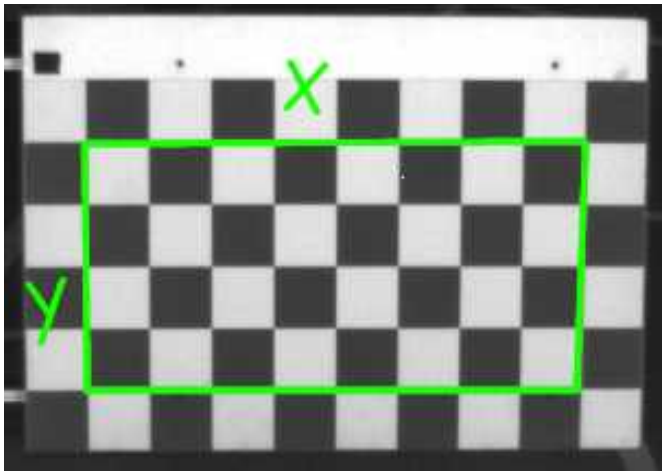
Overview of the fields of view of all the cameras together (left) and a picture of the mounted poles (right).

Single camera calibration

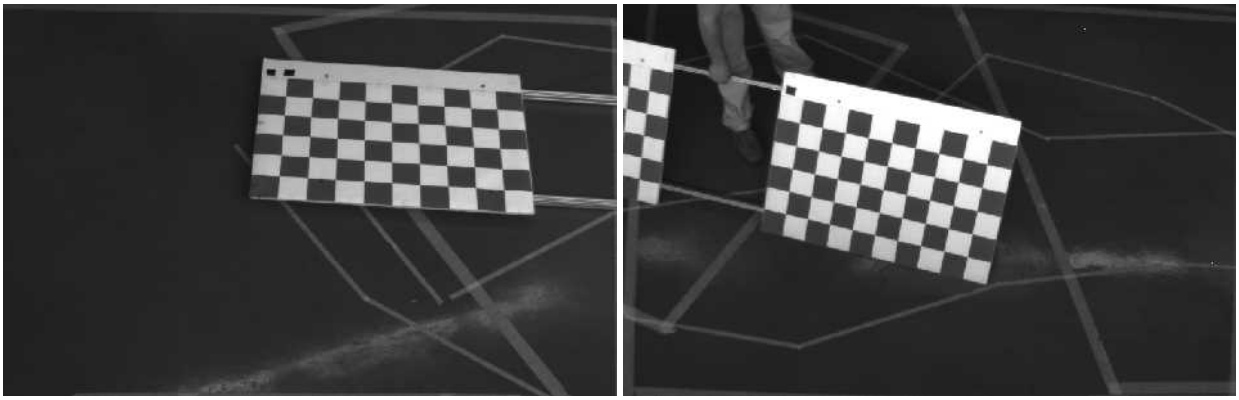
In order to compute the projection parameters of each single camera, we used the software tool for camera calibration developed by Jean-Yves Bouguet's, the "Camera Calibration Toolbox for MATLAB", available at `http://www.vision.caltech.edu/bouguetj/calib_doc/`, which can be considered as the state-of-the-art. This calibration toolbox requires an appropriately sized chessboard that we defined as in the pictures below, so to be easily perceivable in the whole working range. The calibration tool takes in input images of the calibration pattern, and it is important to have images in this set covering poses of the pattern in all parts of the image, and also both close and far in the depth, and also featuring different pattern orientations. From the images of the calibration pattern, the calibration toolbox computes the projection parameters, it also gives out the roto-translation between the camera and each pose of the calibration pattern.



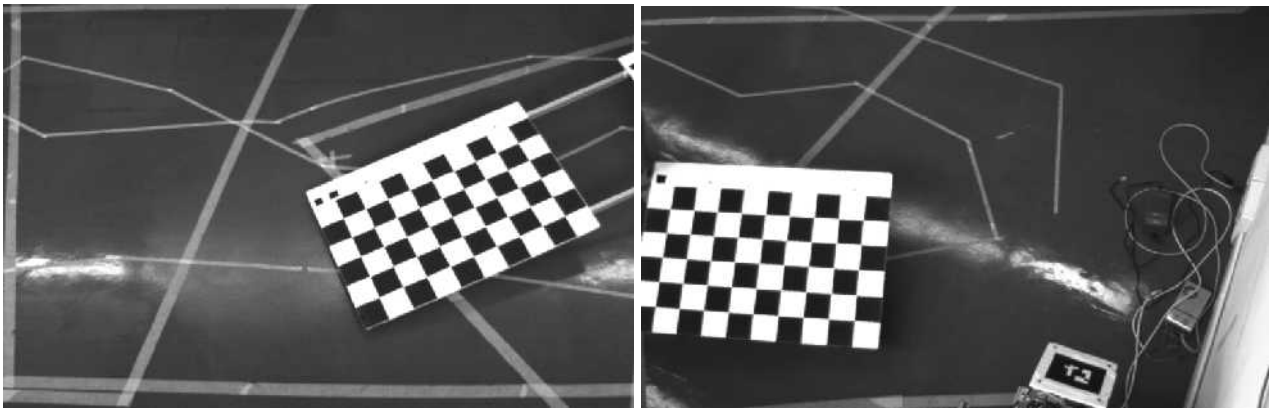
The calibration pattern is a chessboards made up of 6 raws and 10 columns of squares, 100mm by 100mm in side. There are two patterns, for reasons that will be clarified below, and they are fixed together with two aluminum bars and eight black screws, placed in the black squares. The distance between the two pattern is 0.5m.



The reference frame on the pattern, the localization of the pattern with respect to the camera represents the pose of this reference frame.



Example of images used to calibrate the camera 2 (left) and the camera 1 (right).

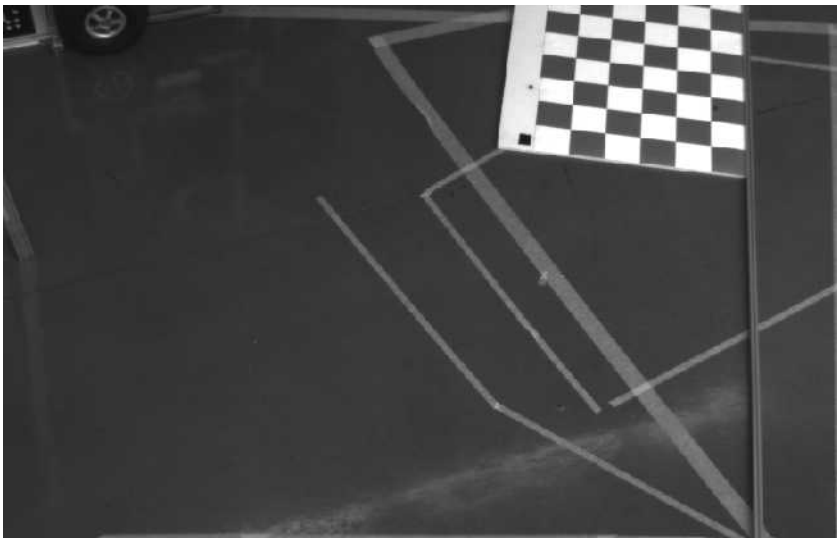


Example of images used to calibrate the camera 4 (left) and the camera 3 (right).

By selecting one of the poses of the pattern, and thus the corresponding camera-to-pattern roto-translation, one can define a frame that will be used for the localization of the objects observed by the camera; in other words, the pose of the objects will be expressed in this frame. Not making this selection would mean to have all poses represented in the intrinsic camera frame. This is relevant for us as it concerns the robot poses output by the GT system.

In order to validate the GT system, we need to have the GTframe as well as the Vframe in a known relative position, so that it would be possible to express both outputs in the very same frame. Our choice has been to have the GTframe and the Vframe in full coincidence. In order to reach this objective we physically attached the two. This attachment has been obtained by putting the calibration pattern on the floor in the "exact" (the meaning of this word here is: by manual alignment) superimposition with the Vframe, see pictures below (the same pictures presented at the end of the section on collection of validation data). This pattern was then selected as the reference one. In particular, given the specific pose of the Vframe with respect to the FOV of first camera in the chain, we had to put the frame shifted by a known translation (2 checkers) with respect to the origin of the Vframe. Still we could easily compute the roto-translation between the pattern and the camera. This is then composed with the GT output, which will be natively expressed in the camera frame. In conclusion, we have a chessboard, and consequently we can compute the output of the first GT camera (which is camera 2) in the chain, referred to the GTVframe.

Calibration pattern put in a known position with respect to the GTVframe, so to compute the camera-to-GTVframe roto-translation.



Calibration pattern put in a known position with respect to the GTVframe, as seen from camera 2

To check that each camera was correctly calibrated we compared the distance of two known points, as obtained by the vision system, with their real distance. The calibration chessboard was positioned in two positions at a known distance each other and then, using the images and the intrinsic parameters within the Matlab Toolbox, the two poses were computed, and their relative distance was checked.

Calibration of the camera network

As mentioned above, the overlapping field of view for two adjacent cameras has to be quite narrow, since we

have to replicate the real setup. Past experience actually shows that there was not enough space to put a chessboard in the overlapping region. Of course this will not be the case in the GTroom, but we will behave like it were the case.

Given that publicly available software for calibration of camera network works in the overlapping region of the cameras, we had to invent some trick to connect the two adjacent FOVs. We built a "double" calibration pattern, where we put two patterns on a mechanical frame and their relative position is considered known. During the usage we have to put each of the two patterns in the field of view of each of the two adjacent cameras, see the pictures above related to the single camera calibration: they are, in pairs, the very same images grabbed for the joint camera calibration, notice the metal bars keeping the two patterns in the given relative position.

It's important to have a naming convention for the acquired images because the image acquisition should be done in pairs with the cameras: every snapshot from the first camera of the pair should have a corresponding snapshot from the second camera. We used the number of the cameras and the iteration of the snapshot as filename, e.g.:

4th snapshot from camera n.2 (when paired with camera n. 1) : 2-1-4.bmp

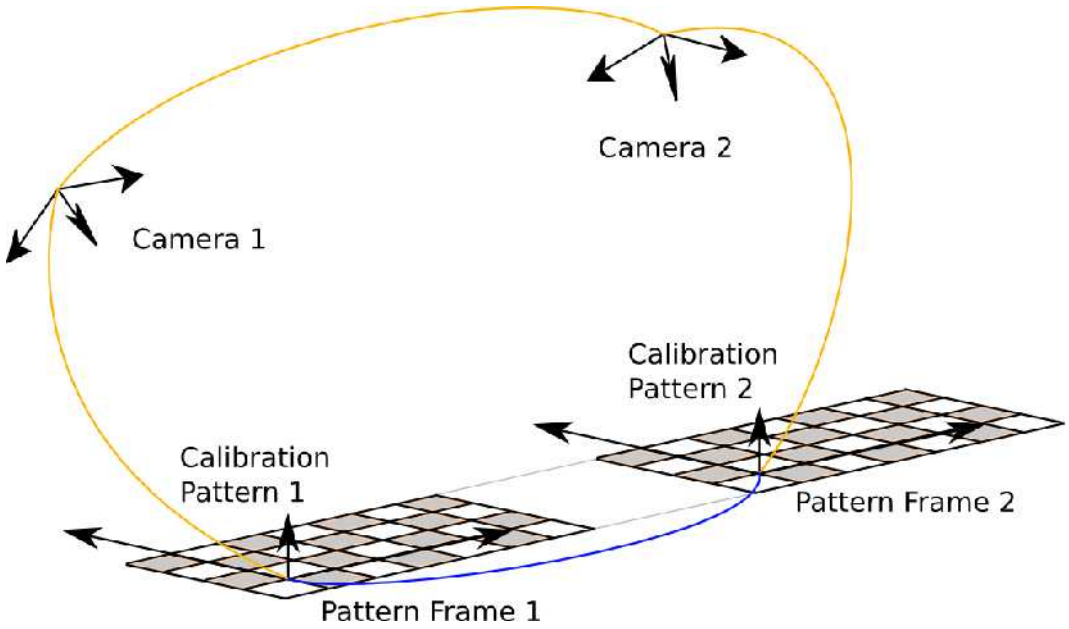
4th snapshot from camera n.1 (when paired with camera n. 2) : 1-2-4.bmp

3th snapshot from camera n.1 (when paired with camera n.4) : 1-4-3.bmp

3th snapshot from camera n.4 (when paired with camera n.1) : 4-1-3.bmp

To avoid other time-consuming activities related to the file input-output, images are grabbed in a format compatible with the "Camera Calibration Toolbox for MATLAB". The procedure for calibrating a pair of consecutive cameras is as follows:

1. grab the pair of images from the adjacent cameras with the ad-hoc developed "doubleclick" software, previous painful experience showed that it takes too long to put the double-pattern in a given pose, fix it with boxes, seats, tables, etc., and then repeat this many times so to get a good coverage of the neighboring parts of the two FOVs; with doubleclick we grab two images at the same time, so we can just keep the double-pattern with the hands, really speeding up the process;
2. compute, using some ad-hoc developed Matlab code (this is based on the above mentioned calibration toolbox, plus composition of its results), the roto-translation between each two cameras: camera1 - pattern1 - mechanical structure of the double-pattern - pattern2 - camera2, see drawing below;
3. compose the roto-translations between cameras so that the vision-based GT system can output the robot pose in the GTV frame, independently on the camera that is observing the robot.



Drawing representing the roto-translations between the different frames involved in the joint camera calibration, in blue the one known after building the double pattern, the ones in yellow from each camera to each pattern are determined by the single camera calibration, the output of the joint camera calibration is the camera_i to camera_j roto-translation.

Notice that each double pattern would allow to estimate the external roto-translation between adjacent cameras, but we will pay attention to have more than one image of the double pattern, i.e., more than one estimate of such external roto-translation, in order to reduce the noise effects. The current implementation applies averaging to the elements of the roto-translation matrices, instead of averaging the 6 degrees of freedom representing each relative pose. This has been considered more robust to noise.

Accuracy of the roto-translation matrix of the camera with respect to the GTVframe was checked in a simple trial by using a small chessboard placed on the corner of a table. The picture below shows an example of how we positioned the pattern and the table both for the position and for the rotation, by using two plum-lines.

The chessboard used to check the calibration of the camera chain.

The table was positioned on some validated pose, i.e., known with respect to the GTVframe, and the intrinsic parameters of the camera were loaded into the Calibration Toolbox. The extrinsic parameters (roto-translation) between the reference frame located on the chessboard and the camera reference frame was then computed. Combining this roto-translation with the one from the camera to the GTVframe, as computed by the joint camera calibration procedure, it was possible to estimate the pose of the pattern in the GTVframe and compare it with the validation. This procedure was repeated at least once for each camera. The table height is 0.720m.

The results are presented hereafter. We are quite happy because the results show that one potentially critical issue is not a problem. We were fearing a sort of "cumulative errors" distortion in chaining the FOV of the cameras, which could have tuned into estimates increasingly bad with the increasing position of the camera in the chain, i.e., the farther the camera the worse the estimates. This is still likely the case, but at a grain not perceivable with the accuracy of our validation, i.e., not perceivable at all.

pose 2

	validation robot-frame	chess board on table	valid – chess board
x	1.8838	1.8819	0.0019
y	-0.6341	-0.6364	0.0023
z	0.0000	0.7162	0.0038

pose 6

	validation robot-frame	chess board on table	valid – chess board
x	0.8781	0.8905	-0.0124
y	1.9556	1.9576	-0.0020
z	0.0000	0.7388	-0.0188

pose 5

	validation robot-frame	chess board on table	valid – chess board
x	1.3285	1.3401	-0.0116
y	1.3213	1.3249	-0.0036
z	0.0000	0.7270	-0.0070

pose 7

	validation robot-frame	chess board on table	valid – chess board
x	0.5397	0.5396	0.0001
y	2.6828	2.6848	-0.0020
z	0.0000	0.7226	-0.0026

pose 23

	validation robot-frame	chess board on table	valid – chess board
x	-0.0333	-0.0613	0.0280
y	2.1745	2.1598	0.0147
z	0.0000	0.7288	-0.0088

pose 11

	validation robot-frame	chess board on table	valid – chess board
x	-0.7092	-0.7207	0.0115
y	4.7668	4.7767	-0.0099
z	0.0000	0.7530	-0.0330

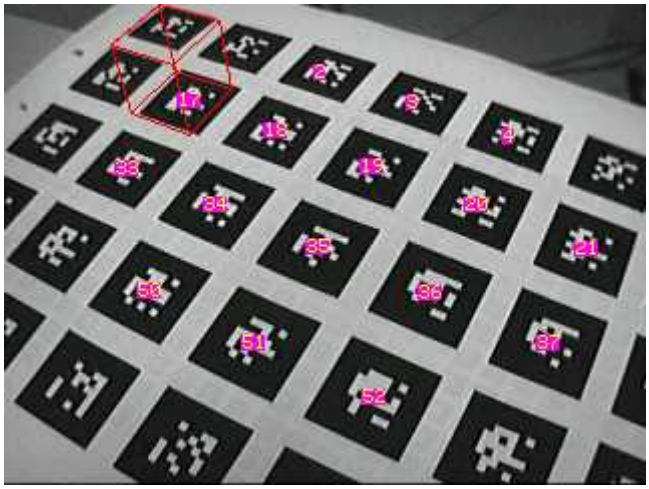
	chessboard estimation stats		
	average Err	standard deviation Err	max of abs values Err
x	-0.0049	0.0095	0.0116
y	-0.0006	0.0042	0.0036
z	0.0083	0.0179	0.0280

Table about the accuracy evaluation of the vision-based GT system with the calibration pattern on the table. Units are meters.

ARToolkit

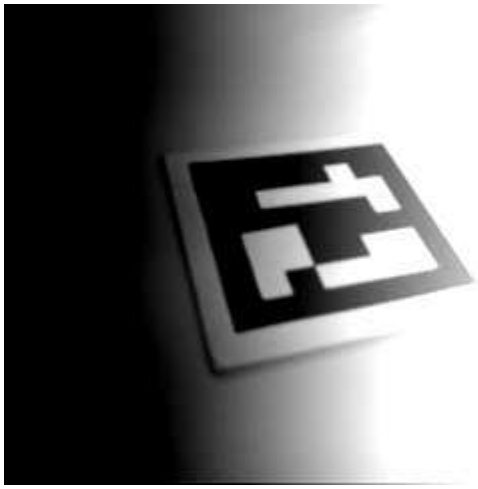
The vision-based GT system that we developed is based on a publicly available software, capable to recognize and localize one out of a large set of markers; it is known as Artoolkit Plus. ARToolKit [Wood et al. 2003] is a software library that can be used to calculate camera position and orientation relative to physical markers, in real time. This enables the easy development of a wide range of Augmented Reality applications. ARToolKit Plus wraps ARToolKit into a C++ class (called Tracker) and adds some new (derived) classes for single (class TrackerSingleMarker) and multi-marker tracking (class TrackerMultiMarker). The features that made us selecting the ARToolKit Plus for the identification and detection of robot poses are mainly (reporting from the ARToolkit Plus description):

- ***Simple Id-encoded markers:*** ARToolKit Plus adds the possibility to switch to id-encoded marker detection instead of the built-in template matching. This allows using up to 512 different markers without training and without speed penalty and gives a speedup even if just one marker is used.
- ***Automatic thresholding:*** ARToolKit Plus can do dynamic thresholding by looking at the marker content (pattern) and taking the average between the darkest and brightest pixels. If no marker is found the threshold value is randomized. This feature requires almost no additional processing power.



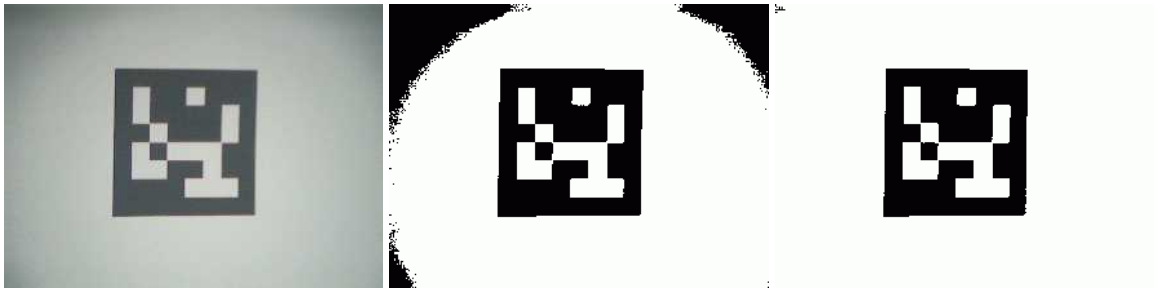
Encoding of the marker ID.

Id-based markers encode a 9-bit number into a 6x6 pattern (see image below). The 9 bits of the number are repeated four times to fill up the 36 bits that can be stored in this pixel array. To improve robustness (and allow marker numbers such as 0 and 511) all pixels are scrambled with an XOR mask. In order to use id-encoded markers the pattern size has to be set to 6x6, 12x12 or 18x18.



Automatic thresholding allows ARToolKit Plus to work under several illumination conditions.

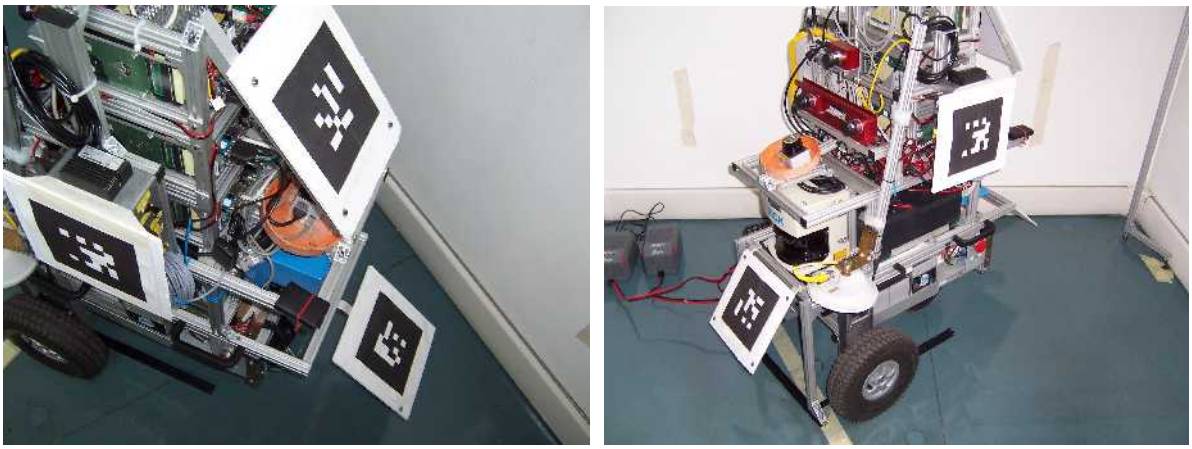
- ***Vignetting (radial luminance falloff) compensation:*** Some cameras have a radial falloff in luminance in the image, as can be seen in the first image below which was taken with a Spectec SD camera. When applying a threshold value of 150 (thresholding is the first thing ARToolKit does...) the result looks like in the second image below. In this case markers near the corners will no longer be detected. After activating the radial luminance falloff compensation, the thresholded image looks like in the third image which allows tracking marker in the whole image.



Vignetting Compensation

- ***Improved camera calibration model (MATLAB camera calibration toolbox support):*** As of version 2.0, ARToolKitPlus is compatible with the camera calibration model used by Jean-Yves Bouguet's Camera Calibration Toolbox for MATLAB. An improved version of the toolbox (highly recommended) that includes automated corner/calibration object detection is available from the Graphics Media Lab at Moscow State University.
- ***Implementation of the "Robust Planar Pose" (RPP) algorithm:*** The robust pose estimator algorithm has been provided by G. Schweighofer and A. Pinz (Inst.of Measurement and Measurement Signal Processing, Graz University of Technology). Details about the algorithm are given in the Technical Report: TR-EMT-2005-01, available at the developers website.

We have put ARToolKit Plus markers on the robot with a configuration that allows at least one marker to be visible by one camera at any time. The ARToolKit Plus returns the 6 degree of freedom pose of each detected marker with respect to the intrinsic camera frame; it is therefore required, on one side, to change this output into the GTV frame by using the rigid transformations between the camera and the GTVframe whose calibration has been previously described. On the other side, we also need to take into consideration the rigid transformations from each marker to the robot-frame.

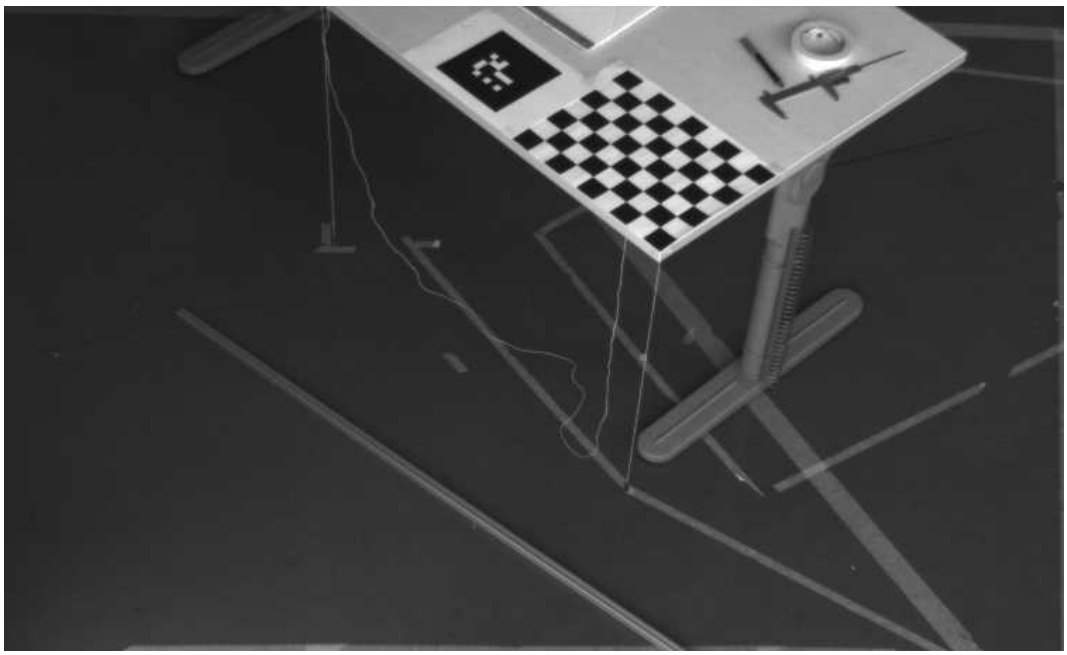


A first configuration of markers on the robot. Afterwards we choose to put more markers on it.

While the side "camera-to-GTVframe" has already been dealt with, we need to introduce some more detail on the side of the "marker-to-robot-frame".

Before delving into this aspect, which might be the source of some error, we present the activity performed

for analyzing the pure ARToolKit performance and its impact on the overall vision-based GT performance. This activity has been performed again basing on the validated poses. We prepared a table, with a single marker and a chessboard (calibration pattern) on it, the two are near each other, in order to compare the results of the pose estimates obtained by using the ARToolkit (on the marker) and by using the Calibration Toolbox (on the calibration pattern). The table was then moved to each validated pose, again exploiting plum-lines, see picture below.



A picture on the comparative analysis of the accuracy with a calibration pattern and with ARToolkit marker, both on a table.

This work was done in six different poses, at least one for each camera. The results are presented hereafter.

pose 2

	validation robot-frame	tag15 on table	chess board on table	valid-tag15	valid – chess board
x	1.8838	1.7816	1.8819	0.1022	0.0019
y	-0.6341	-0.6457	-0.6364	0.0116	0.0023

pose 7

	validation robot-frame	tag15 on table	chess board on table	valid-tag15	valid – chess board
x	0.5397	0.5047	0.5396	0.0350	0.0001
y	2.6828	2.6093	2.6848	0.0735	-0.0020

pose 5

	validation robot-frame	tag15 on table	chess board on table	valid-tag15	valid – chess board
x	1.3285	1.4056	1.3401	-0.0771	-0.0116
y	1.3213	1.2493	1.3249	0.0720	-0.0036

pose 11

	validation robot-frame	tag15 on table	chess board on table	valid-tag15	valid – chess board
x	-0.7092	-0.6621	-0.7207	-0.0471	0.0115
y	4.7668	4.6785	4.7767	0.0883	-0.0099

pose 6

	validation robot-frame	tag15 on table	chess board on table	valid-tag15	valid – chess board
x	0.8781	0.8304	0.8905	0.0477	-0.0124
y	1.9556	1.9530	1.9576	0.0026	-0.0020

pose 23

	validation robot-frame	tag15 on table	chess board on table	valid-tag15	valid – chess board
x	-0.0333	-0.1014	-0.0613	0.0681	0.0280
y	2.1745	2.0934	2.1598	0.0811	0.0147

estimation of marker #15 stats				chessboard estimation stats		
average Err	standard deviation Err	max of abs values Err		average Err	standard deviation Err	max of abs values Err
0.0215	0.0693	0.1022	x	0.0029	0.0152	0.0280
0.0549	0.0376	0.0883	y	-0.0001	0.0083	0.0147

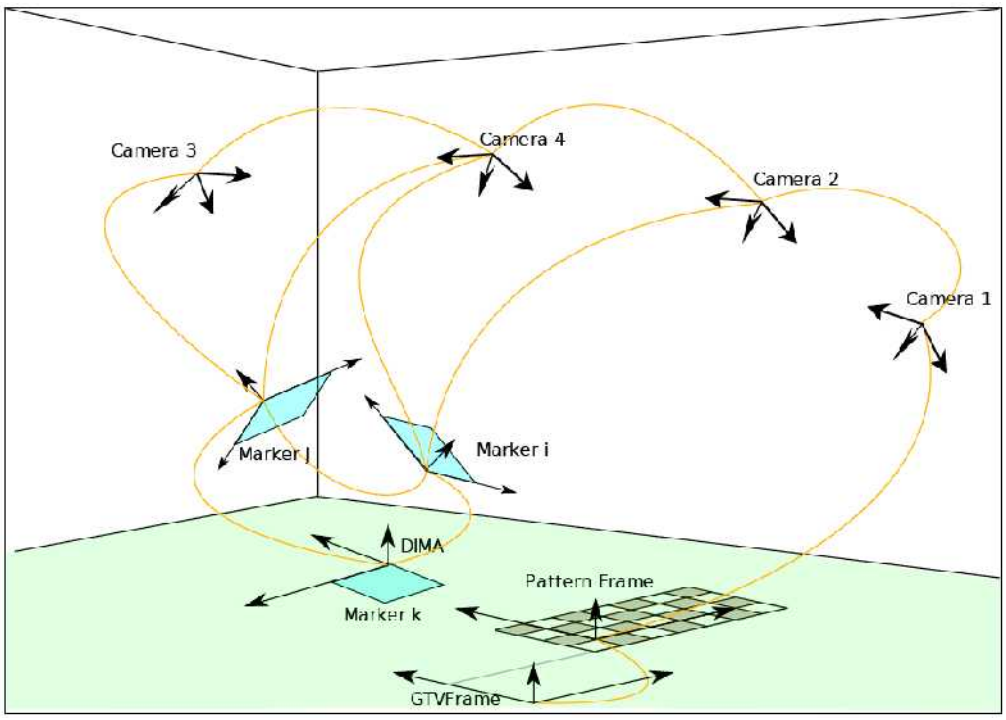
Tables of results. Units are in meters.

We are not happy of the accuracy demonstrated by the ARToolkit, although sufficient for the purpose of the vision-based GT, it clearly shows the potential of the camera sensor for a much higher accuracy, as demonstrated by the localization accuracy attained by means of the calibration pattern. Both the single poses and the average figures show that the accuracy of the marker localization is quite worse than that performed by means of the calibration pattern, although in the limits of the accuracy required for the GT system.

We devised a few alternatives for improving the accuracy, in the (unlikely) case of having some extra workforce available. One option is to look at the sources of the ARToolkit in order to check the actual implementation of the localization part, as we believe the (not impressing) quality to be not appropriate with respect to the potential accuracy attainable with such markers, even considering the slightly larger size of the calibration pattern that we used. A second alternative, for the vision-based GT system, is to compute the robot pose using the calibration toolbox and manual inspection of all poses; this can be done basing on chessboards affixed to the robot, in place of current ARToolkit markers; this solution requires a manual intervention because the detection and identification of multiple calibration patterns in the same image is a feature currently not available in public domain computer vision software. A last option is to automate this missing functionality, so to avoid any manual intervention, given the large number of images produced by the vision-based GT system.

Robot-frame

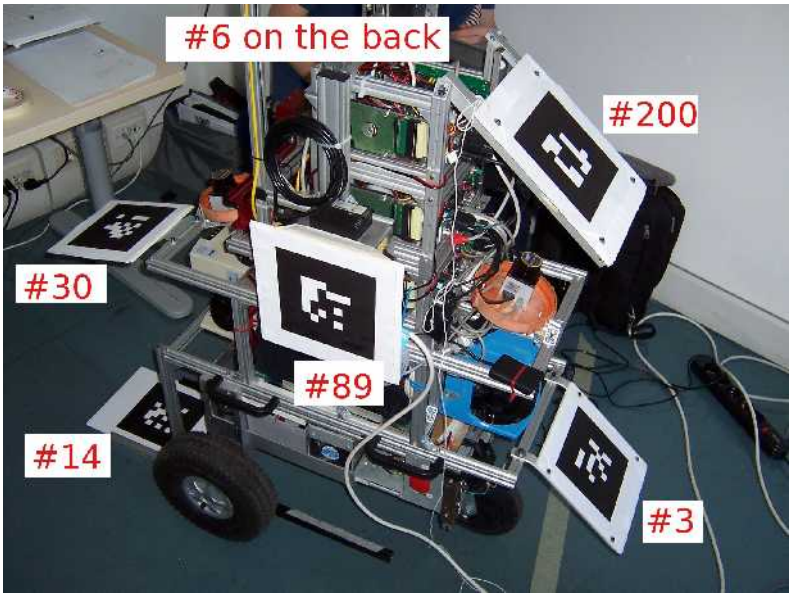
For an overall view at the frames involved in this vision-based GT system see the figure below. The robot-frame, i.e., the frame whose pose will be given in output, has already been defined as the mechanical frame called "dima". This frame has been introduced for easing the validation, i.e., for writing down on the floor, by moving a felt-tip pen along its orthogonal axes, its relevant poses. Its poses are given with respect to the GTVframe. We now need to define a simple way to relate this frame to the frames of each marker, so that the GT system can output the robot pose instead of the marker pose. This means to determine the roto-translation matrices between each marker and the dima.



Sketch representing the roto-translations between the markers and the marker-dima (called "Marker k" in the figure), the camera and the markers, and the cameras and the GTVframe. The sketch does not include the roto-translation between the dima and marker-dima, which is quite easy to determine by hand measurement.

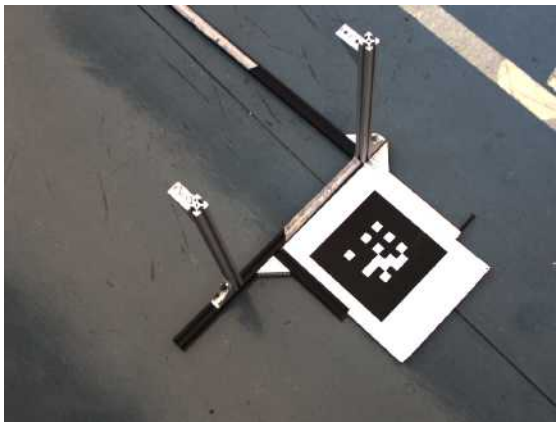
We proceeded as follows:

1. we put different markers on the robot, see photos below;



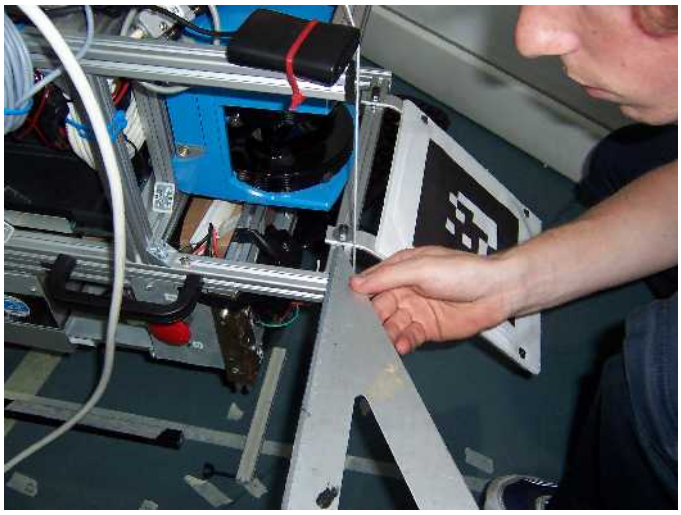
The six markers on the robot and their position, the front side of the robot is on the left.

2. notice the "dima-marker", which is a marker in a simple-to-measure pose with respect to the robot-frame, i.e., the dima, see photo below;



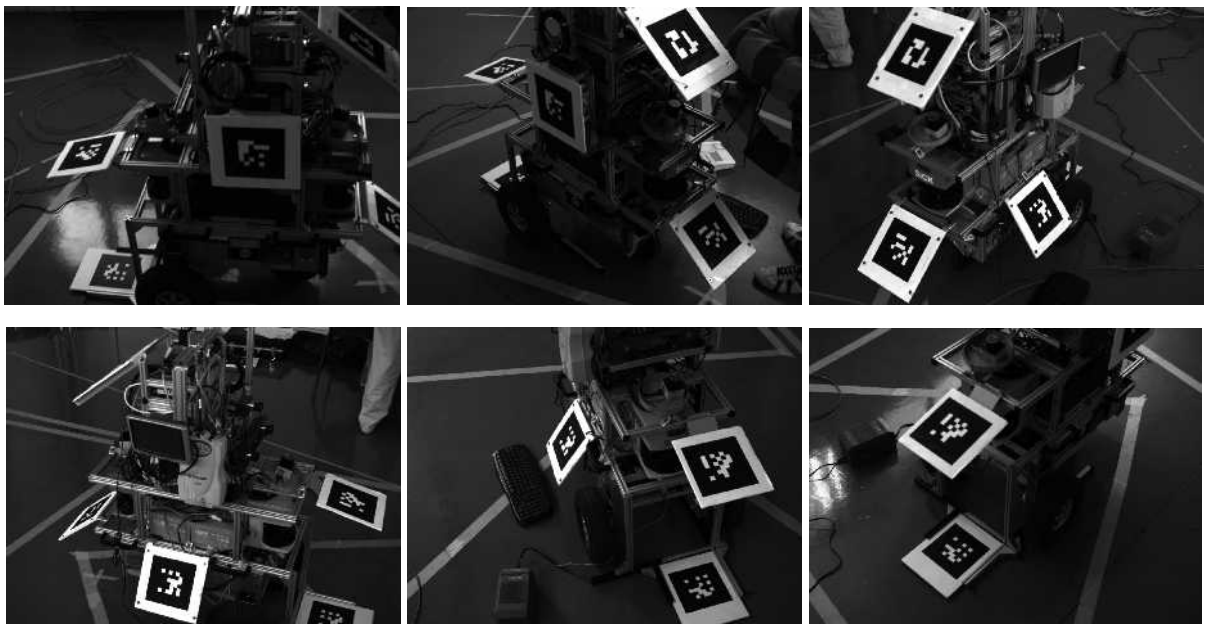
The dima-marker (marker 14), that is the marker to which all the other markers are referred to.

3. for each marker, we determined, with a very complex, cumbersome, and error-prone set of hand-measurements, the roto-translations with respect to the dima, see photos below;



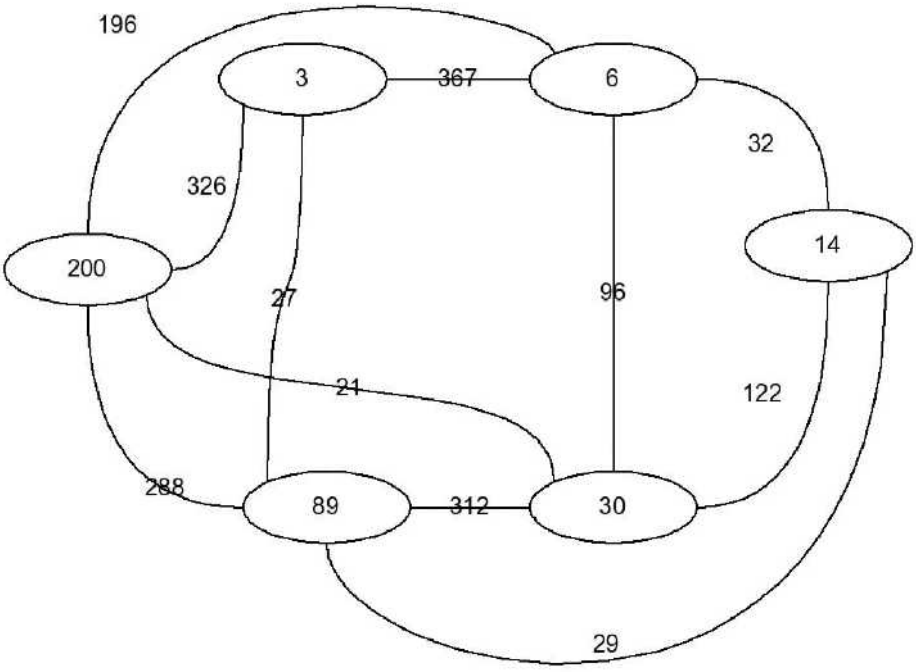
Manual measurements of the roto-translations between the makers.

4. we took a large set of images, i.e., a movie, see below, from a good and well calibrated camera, and used the images including two markers, so to be able to compute the relative poses of the markers, by using again the ARToolkit;



A few pictures from the movie around the robot with a calibrated camera.

5. we averaged all roto-translations found between the same pairs of markers, see picture of neighboring poses and the number of estimates;



The graph contains the relationships found between the markers; the weight on each edge represents the number of images where a correspondence was detected.

Relative position between the markers

In the final setup there are six square markers mounted on the robot (with identification numbers 3, 6, 14, 30, 89, 200). The length of the side, for markers n. 3, 6, 30 and 200 is 156mm; for marker 14 the length of the side is 160mm.

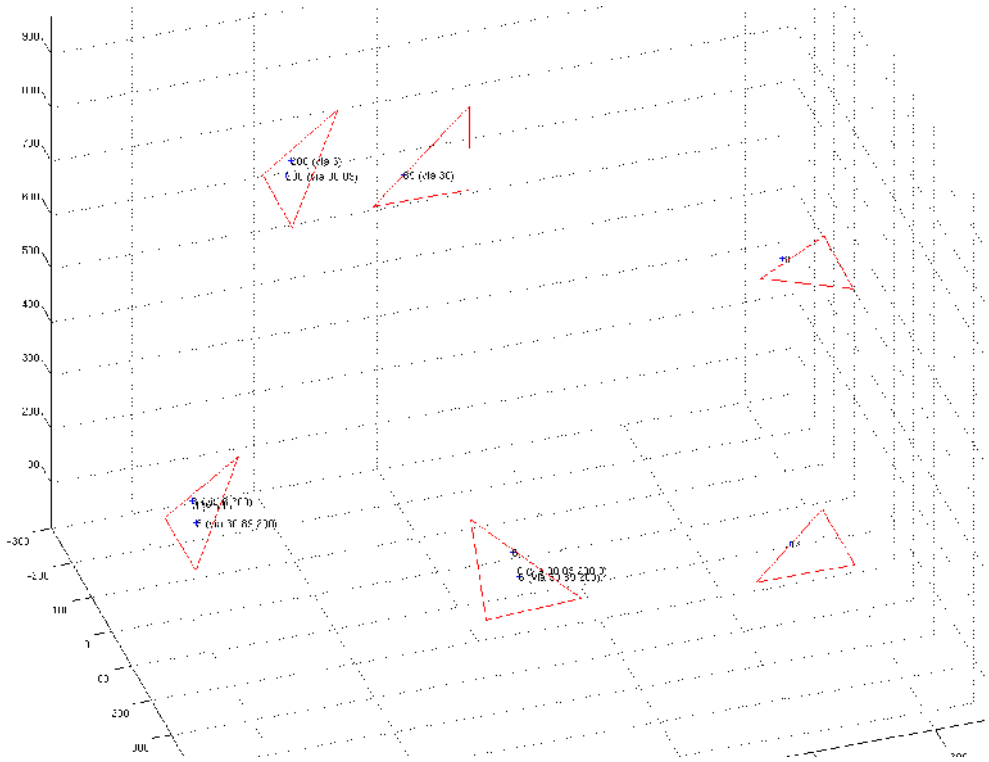
Marker 14 is the the dima-marker, it would be very simple to refer the pose of all other markers to the dima, provided we can represent them with respect to the dima-marker. The roto-translation matrix of the dima-marker with respect to the robot-frame is:

0	-1	0	-143mm
1	0	0	98mm
0	0	1	18mm
0	0	0	1

The positions of the markers were both manually measured (involving plumb-lines, ruler, caliper and some tricks), and also estimated by ARtoolkit with the image sequence mentioned above.

For what concerns the ARToolkit-based approach we averaged all the roto-translations matrices between the same markers; we then concatenated the marker-to-marker roto-translations matrices in order to obtain all the roto-translations matrices between each marker and the dima-marker; in doing so we selected the chains with the highest number of views. The last roto-translation (dima-marker to dima), a pure translation, was then combined.

Whenever more than one marker is in view from a camera, we can combine the two (independent) estimates into a single estimate of the robot pose. On the other hand, when more than one camera is observing the robot, independently on the identity of the markers observed by each camera, we do not proceed to any noise-reduction, as this is considered unfair with respect to the real data-collection conditions, where we expect narrow superimposition of adjacent FOVs, in the GT room such condition was more frequent instead. The current implementation of this combination, i.e., the combination of the two robot poses coming from having observed two markers from the same camera, is by means of averaging of the elements of the roto-translation matrices, instead of averaging the 6 degrees of freedom representing each relative pose. This has been considered more robust to noise. One comment on this combination: we fear that the simple averaging might decrease the accuracy, as the accuracy of each single estimate is heavily dependent on the foreshortening on the marker view. Unfortunately, there is currently no uncertainty estimate in the ARToolkit output, so, beside delving into that software, which is beyond our reach in this very short period, we had to simplify the process with a trivial averaging.



Centers of mass of each marker (w.r.t. the dima reference system)

In the picture above are showed the centers of mass of each marker (w.r.t. the dima reference system). The red triangles are the hand-made measurements and the blue crosses are the estimates obtained by concatenating the ARToolkit estimates.

The comparison of the hand-measured 6DoF pose of each marker, with respect to the dima, with the ones determined automatically with ARToolkit, showed that the performance of the automatic method is comparable to the manual one. As it is impossible to establish which one is more accurate, because the manual approach is also imprecise and error prone, we just know that in future we can avoid manual measurement. This has the relevant practical implication that whenever we fear the markers might have been moved we can simply collect a movie about the robot, and from it we will be able to re-calibrate all the relative poses of the markers; this is an important relief, given the burden involved in a real data-collection.

Collection of the GT data for static poses

As already mentioned we defined a trajectory, see picture in the previous section and also below, involving about 3 runs forward and backward along the longest side of the room, so to allow a complete and repeated observation of the room, which will be relevant mainly for the LRF-based GT system. Along this trajectory we selected some poses, so to split the trajectory itself and the observations from the different cameras. Each such pose was felt-tip pen marked on the floor, the image of the vision-based GT system were then grabbed (and also the scans of the laser-based GT system), the robot moved to the next pose, and the validation data, i.e., the distances of the 3 robot-frame points with respect to the world-points for the previous pose were collected. Then we run the GT system on one hand, and the validation system on the other, therefore determining the values to build a comparison table.

For every pose in which the robot stops, a pair of images from the adjacent cameras closest to the robot was grabbed. This procedure requires to consider granted the fact that, during the GT collection, the images will

be not affected by motion blur and/or other artifacts that might reduce the ARToolkit performance. This is obtained, on the side of image artifacts, by using a room with windows, and not paying attention to the particular conditions of the work, so replicating even the worst cases; on the motion blur side, this is obtained by setting the exposure time short enough to avoid blur, which is possible if the scene is decently lightened, as it will be the case.

The same image grabbing software as in the calibration process was used, but with some changes in the file and folder naming convention. Now we should have four folders, one for each camera, and only one iterator that represent the number of the pose. The two snapshots from the adjacent cameras were saved in the respective folder with the name "poseX.ext".

Pictures showing the path of the robot inside the FOV of the cameras.

Validation results

We present hereafter the results of the vision-based GT systems, for the 26 validated poses. Whenever there is not an entry for that pose, it means that no markers have been detected by the ARToolkit-based approach and therefore in that pose the GT system output is not available.

pose 1

	validation	GTvis cam2	valid – GTvis
x	1.2653	1.1901	0.0752
y	-1.1740	-1.1817	0.0077
th	3.4456	3.4440	0.0016

pose 2

	validation	GTvis cam2	valid – GTvis
x	1.8838	1.8271	0.0567
y	-0.6341	-0.6309	-0.0032
th	4.0013	3.9980	0.0033

pose 4

	validation	GTvis cam1	valid – GTvis
x	1.9801	1.9670	0.0131
y	0.7345	0.7440	-0.0095
th	5.1600	5.2140	-0.0540

pose 5

	validation	GTvis cam1	valid – GTvis
x	1.3285	1.2656	0.0629
y	1.3213	1.3404	-0.0191
th	5.5105	5.5173	-0.0068

pose 6

	validation	GTvis cam1	valid – GTvis
x	0.8781	0.7987	0.0794
y	1.9556	1.9908	-0.0352
th	5.1869	5.1800	0.0069

pose 7

	validation	GTvis cam4 + cam1	valid – GTvis average
x	0.5397	0.4641	0.0757
y	2.6828	2.6861	-0.0033
th	5.1143	5.0985	0.0159

pose 8

	validation	GTvis cam4	valid – GTvis
x	0.2290	0.2035	0.0255
y	3.3570	3.3317	0.0253
th	5.1503	5.1709	-0.0206

pose 10

	validation	GTvis cam3	valid – GTvis
x	-0.2102	-0.2700	0.0598
y	4.4968	4.4440	0.0528
th	5.4880	5.4853	0.0027

pose 11

	validation	GTvis cam3	valid – GTvis
x	-0.7092	-0.7530	0.0438
y	4.7668	4.8110	-0.0442
th	6.0680	6.0657	0.0023

pose 12

	validation	GTvis cam3	valid – GTvis
x	-0.9756	-1.0400	0.0644
y	4.2708	4.1500	0.1208
th	1.7449	1.6469	0.0980

pose 13

	validation	GTvis cam3	valid – GTvis
x	-0.7689	-0.9170	0.1481
y	3.5655	3.3690	0.1965
th	1.9824	1.9869	-0.0045

pose 21

	validation	GTvis cam2	valid – GTvis
x	0.4930	0.3610	0.1320
y	-0.1814	-0.1260	-0.0554
th	5.1983	5.1809	0.0174

pose 16

	validation	GTvis cam4	valid – GTvis
x	-0.0228	-0.1273	0.1045
y	1.5651	1.4206	0.1445
th	1.5150	1.5333	-0.0183

pose 22

	validation	GTvis cam1	valid – GTvis
x	0.5174	0.4740	0.0434
y	1.1805	1.1830	-0.0025
th	4.4479	4.4263	0.0216

pose 18

	validation	GTvis cam1	valid – GTvis
x	1.1491	1.0245	0.1246
y	0.4747	0.3892	0.0855
th	2.6219	2.6050	0.0169

pose 23

	validation	GTvis cam4	valid – GTvis
x	-0.0333	-0.1330	0.0997
y	2.1745	2.0730	0.1015
th	5.5533	5.3897	0.1636

pose 19

	validation	GTvis cam1 + cam2	valid – GTvis average
x	1.8410	1.7694	0.0716
y	-0.3749	-0.4210	0.0461
th	1.9983	1.9843	0.0140

pose 24

	validation	GTvis cam4	valid – GTvis
x	-0.9847	-1.1170	0.1323
y	3.0299	3.0139	0.0160
th	5.3534	5.3101	0.0433

pose 20

	validation	GTvis cam2	valid – GTvis
x	1.2070	1.1040	0.1030
y	-0.8692	-0.8730	0.0038
th	6.0600	6.0626	-0.0026

pose 26

	validation	GTvis cam3	valid – GTvis
x	-0.9872	-1.0790	0.0918
y	4.8136	4.7180	0.0956
th	3.5275	3.5540	-0.0265

Tables of the results, in the columns: the validation value, the vision-based GT estimate computed with ARtoolkit, and the error of the estimate with respect to the validation value.

	GT Vision Stats				GT Vision	
	average Err	standard deviation Err	max of abs values Err		confidence interval for the mean (with 95% confidence level)	
x	0.0804	0.03657	0.1481	x	0.0651	0.0957
y	0.0362	0.06777	0.1965	y	0.0079	0.0645
th	0.0137	0.04618	0.1636	th	-0.0056	0.0330

Statistics of the Vision-based Ground-Truth system. On the left table: the first column is the average error, the second is the standard deviation of the error and the third is the maximum error committed. On the right table: the lowest and the highest values of the confidence interval for the error.

Overall evaluation

The accuracy issue was tackled by basing on well-validated and trust-able poses. The outcome of the validation, in our view, allows us to claim that the devised vision-based GT system is capable to fulfill its accuracy requirements.

The validation has been performed in conditions similar to the ones of the real data-collections, therefore confirming its validity, under the point of view of avoiding motion blur, replicating missed detection due to uncontrollable lighting, detecting motion in the cameras of the network and in case discarding and rerunning the data-collection, etc.

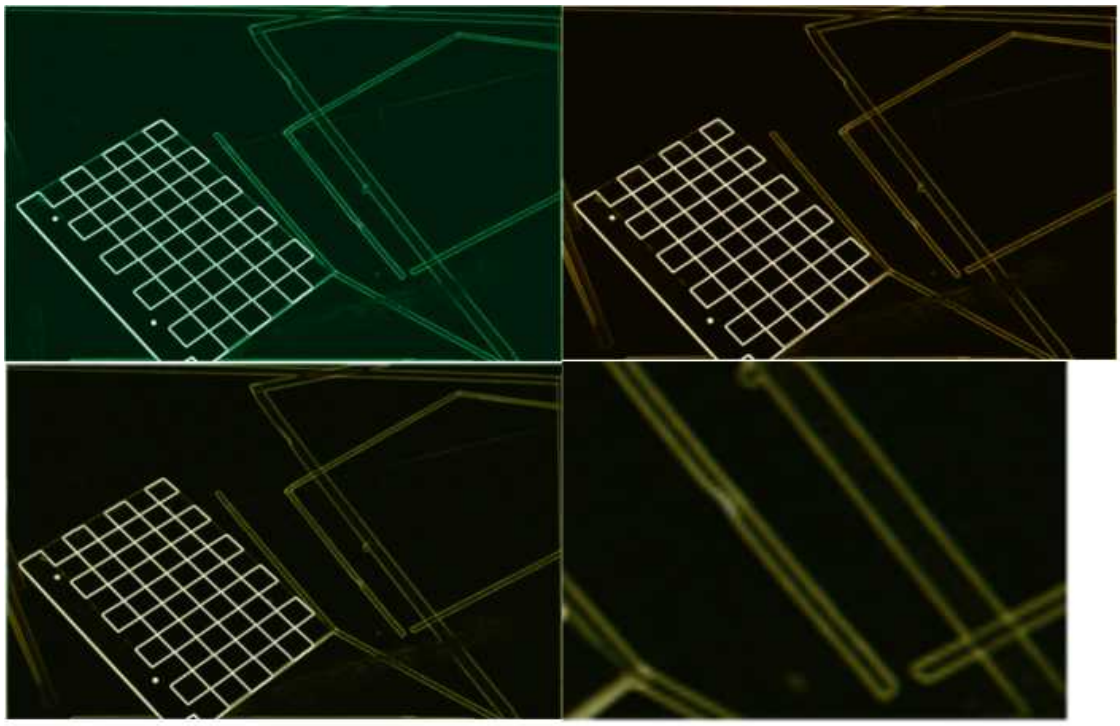
Verification of the steadiness of the camera network

Acquiring images from the cameras, before and after the data-collection, is required, in order to guarantee the steadiness of the camera network. We used two techniques to check the stability of the rig.

The first one is the evaluation of the error between the estimates, for the same scene point, from pictures taken at two different times, i.e., before and after the data-collection; see pictures below. The scene point was the usual calibration pattern re-positioned in the same pose, which was first marked with felt-tip pen.

Images used to evaluate the error between the estimates of a chessboard in the same pose, before (left) and after (right) the data collection.

The second one is based on a direct image comparison, see pictures below.



Example of direct image comparison for camera n. 2. A processed snapshot before the data-collection (top-left), a processed snapshot after the data-collection (top-right), overlap of the two images (bottom-left) and a zoom of the overlap.

Marker detection

In the following we present the success rate of ARToolkit in detecting the markers. As it can be seen the success rate is not as high as one might expect, though usable for GT collection. In a sense this is confirming we did stress-test the ARToolkit software, and the results are realistic.

camera 1							
# pose	number of viewed marker	marker recognized (by marker ID)					
		3	6	14	30	89	200
3	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
4	2	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
5	3	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE
6	1	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
7	1	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
16	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
17	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
18	1	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
19	1	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
20	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
21	1	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
22	1	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
23	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

camera 2							
# pose	number of viewed marker	marker recognized (by marker ID)					
		3	6	14	30	89	200
1	2	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
2	3	FALSE	TRUE	TRUE	TRUE	FALSE	FALSE
3	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
19	1	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
20	2	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE
21	1	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE

camera 3							
# pose	number of viewed marker	marker recognized (by marker ID)					
		3	6	14	30	89	200
8	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
9	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
10	2	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE
11	1	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
12	1	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
13	1	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
14	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
24	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
25	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
26	1	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE

camera 4							
# pose	number of viewed marker	marker recognized (by marker ID)					
		3	6	14	30	89	200
6	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7	2	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
8	1	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
9	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
13	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
14	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
15	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
16	1	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
22	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
23	2	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
24	1	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE

This table resume how many markers are recognized by ARToolkit in all the static poses.

Collection of GT data in continuous acquisition

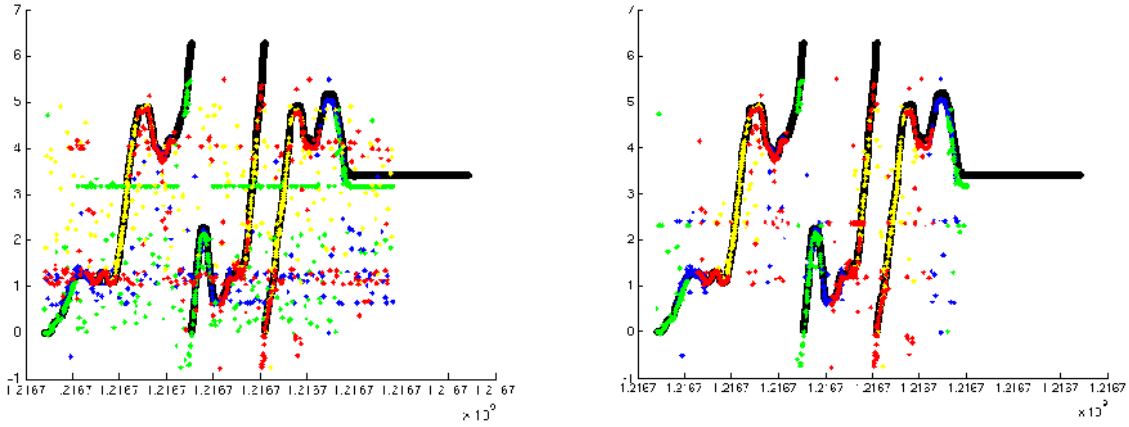
When working on-line, using continuous acquisition, we should take into account the synchronization between the external GT measurement system and the data collected by the robot. In the document "***Project status and plan for reaching project objectives"*** edited in February 2008 UNIZAR performed a preliminary evaluation of the data collected by the robot and spotted a delay between the odometry and the lasers. Such

delay could be introduced in the GT acquisition as well, voiding its effectiveness.

To have synchronization, we timestamp each frame collected by the GT Visual system and we use PTP to synchronize the PC collecting the GT and the robot; nevertheless, cameras are not triggered so we might expect some delay between the image acquisition and the saving of such image on the hard drive. With synchronization validation, we mean the procedure we use to check whether such delay is significantly smaller than the cameras frame-rate so we can consider the timestamping, and thus the vision-based GT, accurate enough.

To perform this validation, we collected a new dataset in the GTroom with the robot moving and the camera network acquiring the GT images; the robot and the camera network were synchronized with the PTP protocol. To check the synchronization between the logs on the robot and the vision-based GT system we compared the estimate of the robot orientation as computed by the odometry and the orientation, in the GTVframe as computed by the GT cameras. We are not interested in the exact value here, but in the correlation between the two signals; in fact, by computing signal cross-correlation we are able to spot delays between the data logged on the two machines, and thus between the GT and the onboard sensing.

Before comparing the signals we need to align them since the odometry estimate is in the robot-frame while vision-based GT estimate is obviously in the GTVframe. The result of the alignment is reported in the picture below.



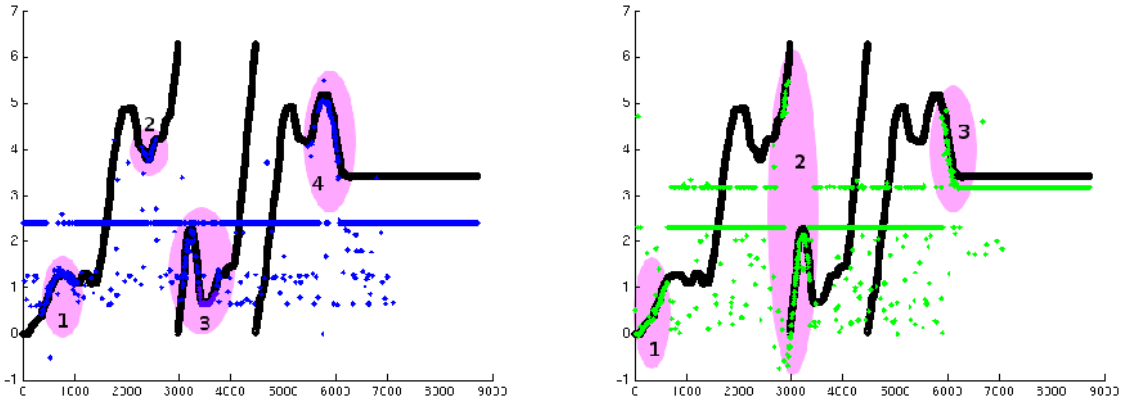
Signal alignment (on the left), and a after preliminary spurious removal (on the right). In black we have the odometry, in blue camera 1, in green camera 2, in yellow camera 3, and in red camera 4.

Note that the odometry accumulates errors in the orientation so we expect the two signals to be different at some point. Moreover, the signals from the GT system that we are using are quite noisy since we did not check for the robot being in view or for the partial presence of markers; when collecting the GT in the real set-up we will have to carefully check the result and remove spurious detections as it has preliminary done in the picture on the right; moreover, we expect to improve the quality of the GT signal with a proper filter, but this is out of the scope of this report.

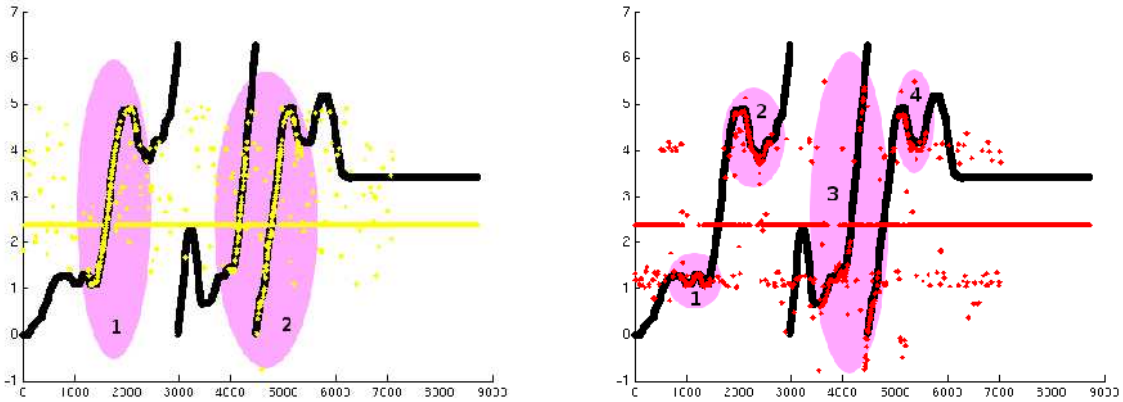
We used cross-correlation to find the delay between GT and odometry in the whole signal, for each camera, by seeking the maximum of the correlation in a time window of 2 seconds. The maximum delay found is 20ms for camera1, 2s for camera 2, 20ms for camera 3, and 2s for camera 4.

The delays for camera 2 and 4 are quite strange, so we decided to check, i.e., to look "with human eyes" at the signals, on selected parts of it, i.e., in those parts where the robot was perceived. Those area are depicted

in the pictures below for the 4 cameras.



Selected part of camera 1 (left) and camera 2 (right) signals (note: on the x axis we have samples, each of them corresponds to 20 ms)



Selected part of camera 3 (left) and camera 4 (right) signals (note: on the x axis we have samples, each of them corresponds to 20 ms)

The numbers given to the interesting parts of each signal are relative to each signal; of course, the reader can base on the odometry, which is common to all pictures, to put all signals in the same timeline, though this is not interesting here.

For camera 1 the delays, in the interesting parts, are: -20ms in part 1, 0ms in part 2, 140ms in part 3 and 0ms in part 4.

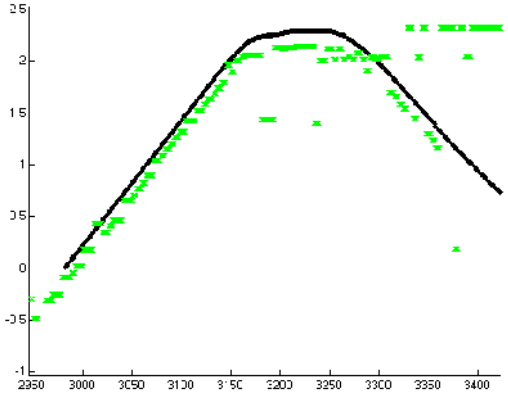
For camera 2 the delays, in the interesting parts, are: 0ms in part 1, 340ms in part 2, and 0ms in part 3.

For camera 3 the delays, in the interesting parts, are: 0ms in part 1 and 20ms in part 2.

For camera 4 the delays, in the interesting parts, are: 0ms in part 1, 0ms in part 2, 80ms in part 3, and 0ms in part 4.

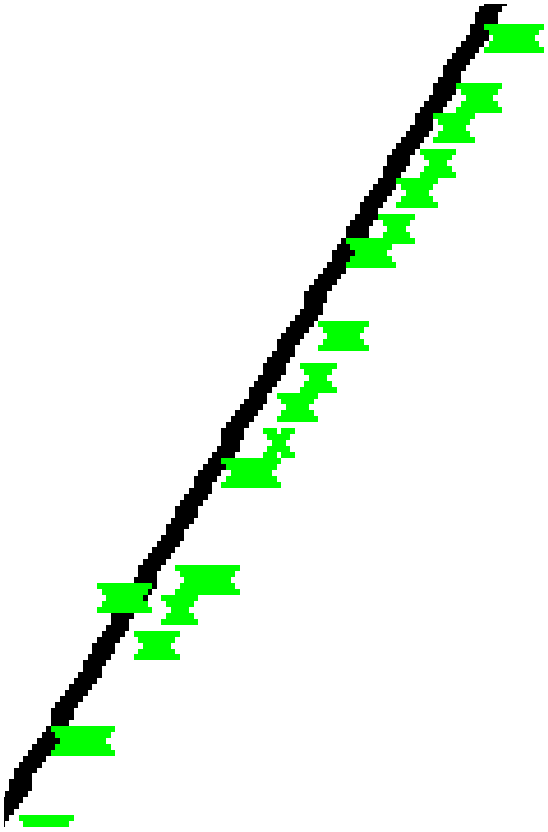
These "human-based" analysis of the signals confirms the hypothesis of spurious correlations in the previous results, and validate the synchronization between the robot and the vision-based GT system. Considering that the frame-rate of the cameras was 10Hz, each picture is taken every 100ms. According to the analysis above the delay is always below this limit, thus we can accept the synchronization between vision-based GT and the timestamped streams taken by the onboard sensors, i.e., the datasets.

The only unclear point in the signals above, is in the second part of camera 2, where the delay is significantly higher, but we assume that this single case is mainly due to spurious correlation with the noise in the signal: refer to the plot in the pictures below.



A closer view of the camera 2 signal with the highest delay (was called "part 2" above).

Looking at the plot, we can see that the mis-alignment does not matches the 340ms computed by using the whole part 2 sequence; the enlarged view below shows that the delay is much less than the 340 / 20 = 17 samples; remind that on the *x* axis we have samples, each of them corresponds to 20ms, and images are replicated 5 time, as its acquisition is 5 times slower than odometry.



An even closer view to a piece of part 2 signal from camera 2.

Validation of the laser-based GT system

Description of the procedure followed

We developed a second method for providing the GT on the robot pose. The reasoning behind such second development is that we expect a significant amount of Benchmarking Solutions, i.e., algorithms, exploiting our datasets for benchmarking, which are not using all the sensor streams onboard the robot. It is actually one of the added values of the RAWSEEDS datasets, to include many sensor streams from the same data-collections, so to allow comparison not only between algorithms (quite of interest for the academia), but also between sensing suites (of more interest for companies). In particular,we expect a significant number of solutions based on sensors other than Laser Range Finders (LRFs), therefore freeing, for such solutions, the option to provide pose validation by means of today state-of-the-art localization methods based on them. Of course comparison between sensing suites including LRFs need to base on the fully independent vision-based GT, like purely LRFs approaches, but other solutions might benefit of the higher accuracy provided by such sensors.

A method for obtaining highly accurate, relative pose estimates between two nearby robot locations uses the laser range observations of the robot and aligns the scans by means of a scan alignment procedure which is described below. This technique or a similar scan matching method is used in most graph-based SLAM methods that operate on 2D laser data for identifying constraints. The high precision of the laser range finder allows small errors in this alignment and provide and efficient way to measure robot displacement.

Given one knows the relative displacement between locations from which the robot obtained laser range observation, one can evaluate the quality of maps learned with different SLAM approaches by comparing the relative distances of the corresponding locations in the map. In using the LRF alignment for GT evaluation we defined a specific robot pose in the GTVframe as the LRFframe and aligned all scans with respect to this reference frame. By combining the roto-translation provided by the scan alignment, with the roto-translation between the laser and the robot dima, and then the roto-translation between the LRFframe and the GTVframe, we can refer any robot pose w.r.t. the GTVframe.

However, an automatic procedure for aligning laser range observations recorded at different locations is not free of errors. Errors can result from the fact that scans cover a too small overlapping area, the data association between the measured obstacle locations is not known, and that the optimization procedures used to find the alignment are local procedures. Thus, it is important to manually inspect the matchings provided by an automatic procedure to eliminate inaccurately aligned scans. Laser range scans recorded with accurate sensors (such as SICK LMS 291 scanners) provide a dense set of proximity readings with small measurement errors. Therefore, the automatic procedure, in combination with manual inspection, allows for providing the relative displacements between pairs of locations from which scans are recorded with a high accuracy, and we take it for ground truth. The next three sections describe the individual steps for the matching in more detail.

Finding Pairs of Corresponding Scans

Before the system is able to align two scans, it needs to identify which scans cover approximatively the same area. Given an initial guess about the poses of the robot, which can be obtained from odometry or using an existing SLAM algorithm, our approach searches for laser range observations that have been recorded from nearby locations given this initial guess. This procedure can be efficiently carried out by means of a kd-tree data-structure. Obviously, this procedure requires a reasonable initial guess. In most practical scenarios, a calibrated odometry or incremental scan matching provides such an appropriate initial guess. To validate if an appropriate initial guess is available, one can simply validate the map built from the initial guess for topological correctness by visual inspection.

Automatic Matching Procedure

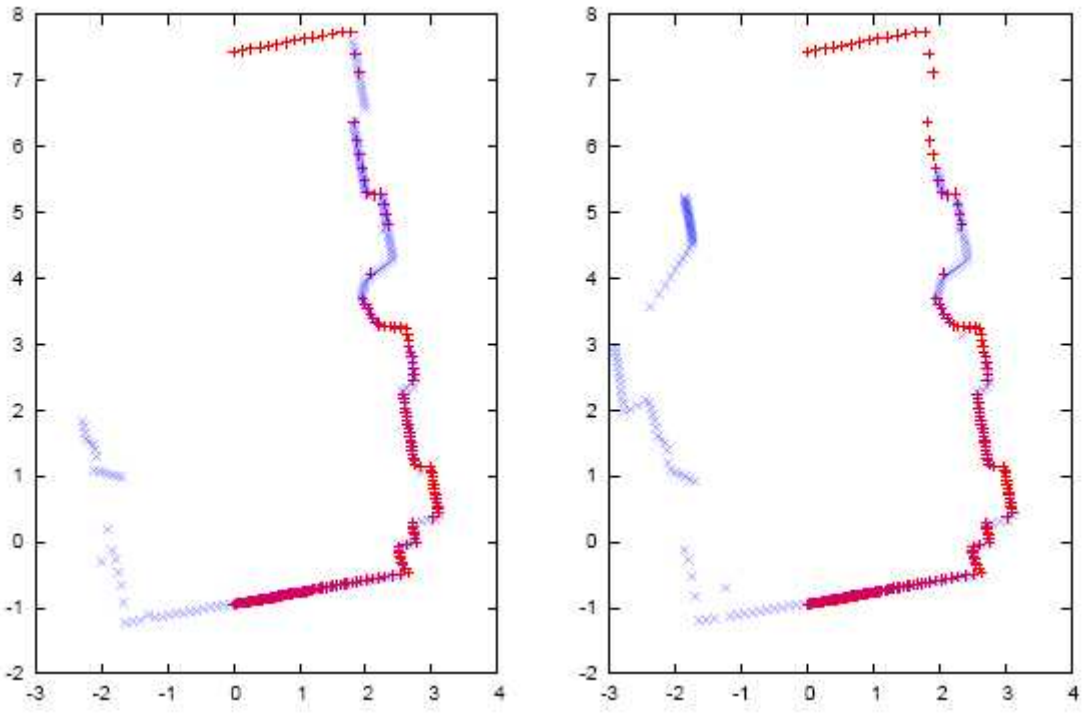
To finally align two scans recorded with a laser range finder, we apply a method proposed by Censi [1]. This algorithm extracts local normals from the two scans to match and utilizes this information in the alignment procedure. To determine the translation, which moves the laser end points of the second scan on the one generated by the first scan, it is sufficient to know the associations between the points in the first scan and the points in the second scan. Given this association, the relative rotation is the angular difference between the normals of the corresponding points. The translation is the difference between the x and y coordinates of the matched end points.

Since in practice, the data association is now known, the algorithm iterates over all potential assignment and computes for each association the relative transformation to align the scans. It then returns the transformation as the most likely one that generates the smallest matching error.

Since in practice, the data association is now known, the algorithm iterates over all potential assignment and computes for each association the relative transformation to align the scans. It then returns the transformation as the most likely one that generates the smallest matching error.

Manual Inspection

Each matching generated between two scans is manually inspected by a human to account for the fact that only accurate matches are considered to generate accurate relative displacements between scans. To do so, we use a plotting program (such as gnuplot) and plot the matched scans in different colors, as in the Figure bellow red and blue. In this way, a human operator can quite efficiently validate a matching given that the human knows the structure of the mapped environment.



Manual inspection of two pairs of correctly aligned scans (scan 1 in plotted in red, scan 2 in blue).

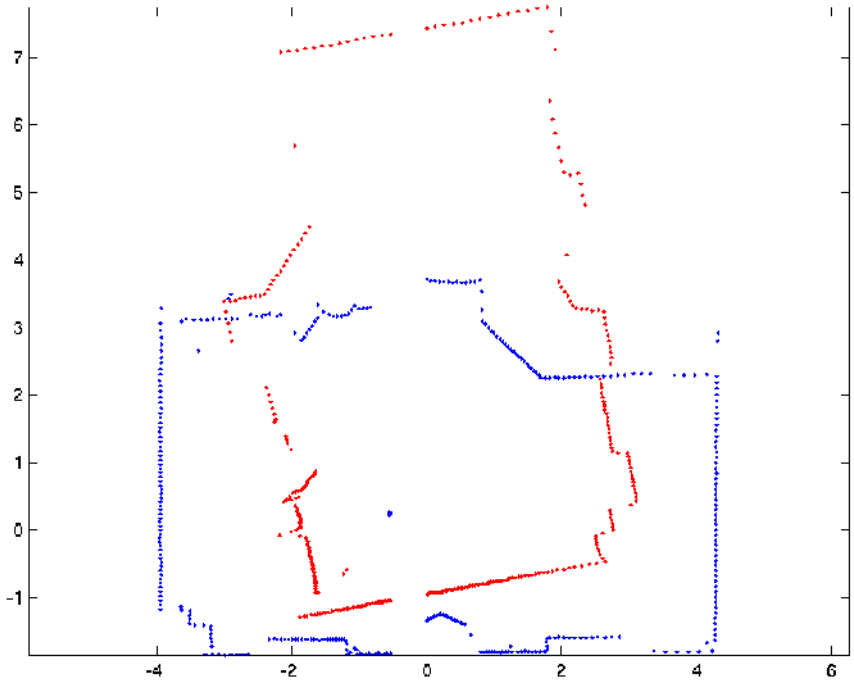
Using two laser range finder with a genetic algorithm

By using the previous method we were able to match 50% of the scans obtained. To improve the number of pose for which we can give a GT evaluation we went a little bit further in scan matching. Since the robot is provided of 2 LRF we applied a second method and simulated the presence of a 360 degrees LRF for scan matching. We were also interested in finding if it was possible to perform accurate scan matching without introducing manual inspection and proper initialization; to do this we developed a genetic algorithm for scan matching.

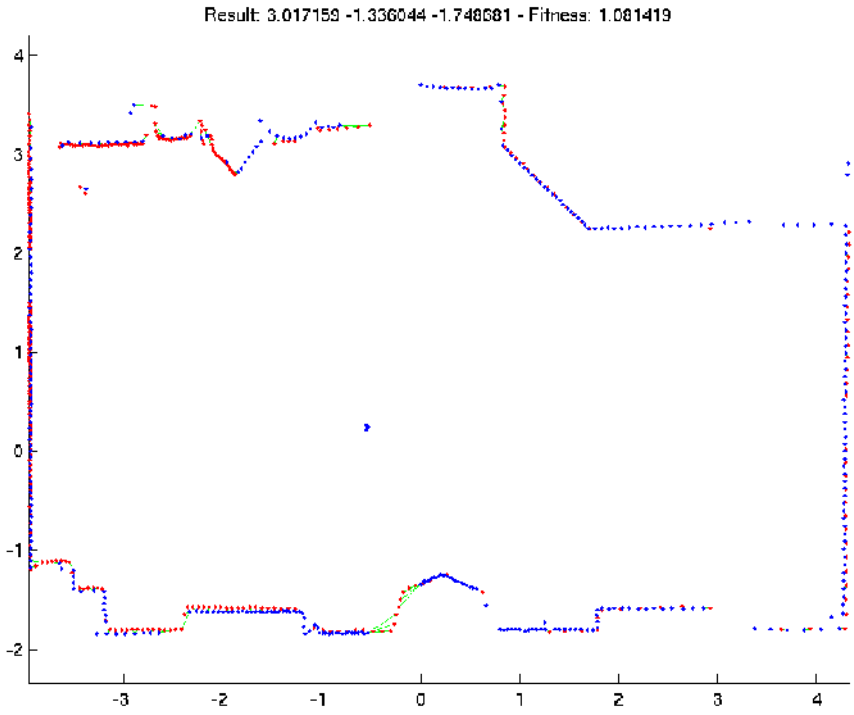
The genetic algorithm is a simple real coded genetic algorithm using 3 genes to represent the 3 degrees of freedom of the robot. Data association is quite naïve but it turned out to be effective: a point is associated to the closest one. Although this data association does not check if multiple points get associated to a single one or vice versa, it turns out that world regularity (i.e., we do not have randomly scattered points, but mostly points clustered in lines or curves) always allows proper matches. A different issue is given by out-layers that, by definition, get associated always to a wrong point and this has been taken into account in the evaluation of the genes. The fitness of the match is computed by summing the squared distances of associated points; to take into account out-layers we sorted all the matches and discarded the first 5 (this is

~1% so we are quite sure it does not affect the proper evaluation of the solution).

An example of the result obtained by the algorithm with a population of 100 individuals after a maximum number of 1000 generation is given in the pictures below. As you can notice the matching is quite good and the few out-layers on the bottom of the map do not affect the final solution.

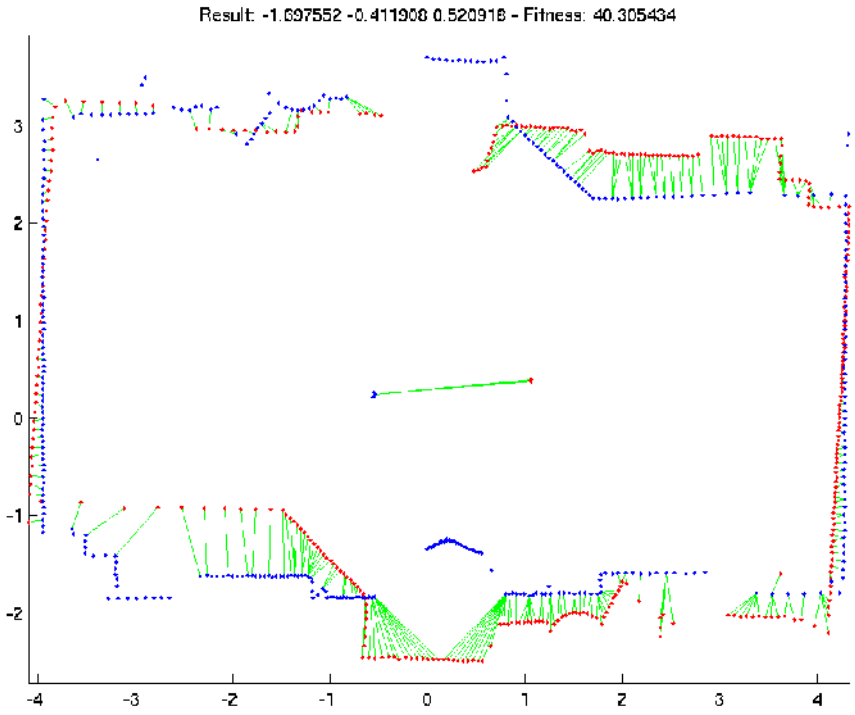


Genetic match between pose 6 and pose 1. The two scans before the alignment.

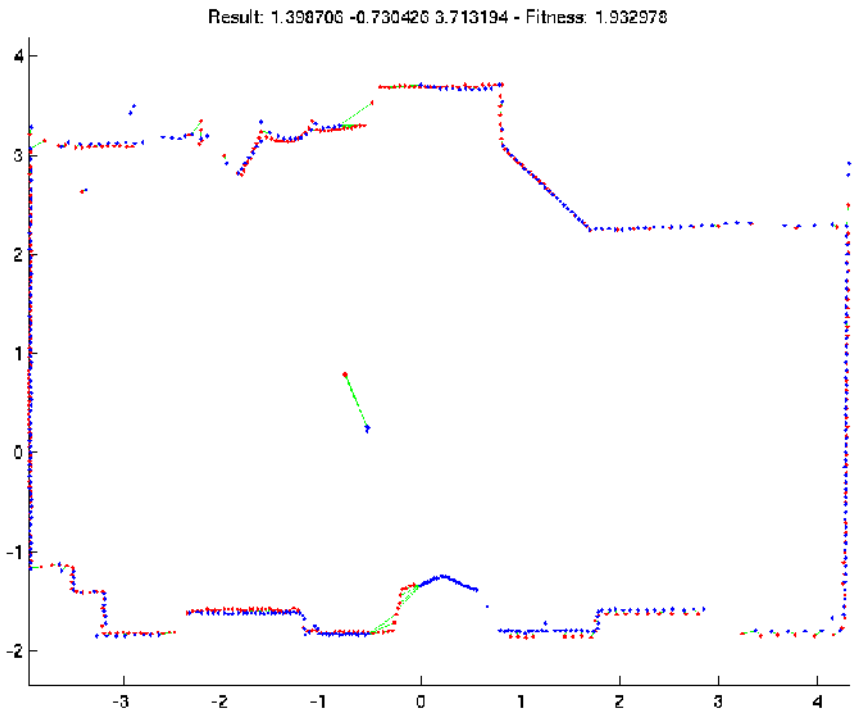


Genetic match between pose 6 and pose 1. The two scans after the alignment.

To speed up the performance of the algorithm we seeded the initial population with one individual equal to the odometry estimate and reduced the number of generation to 300. Although the genetic algorithm is a global optimization method, using a fixed number of generations turns into the need for manual inspection of the final solution in order to check if the alignment was correct or not. Once an error is detected, as in the pictures below, it is possible to add new initial guesses in the initial population to ease the matching procedure. For the 3 (out of 25, since one is the LRFframe) poses where we were not happy, we manually aligned these scans and put these solutions in the initial population obtaining the perfect match at the end.

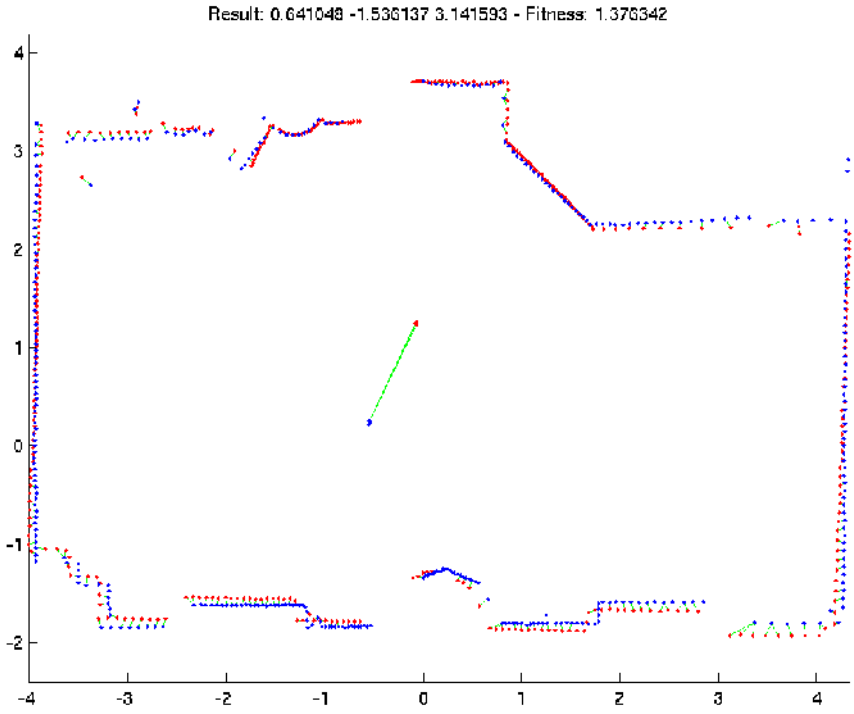


A wrong match

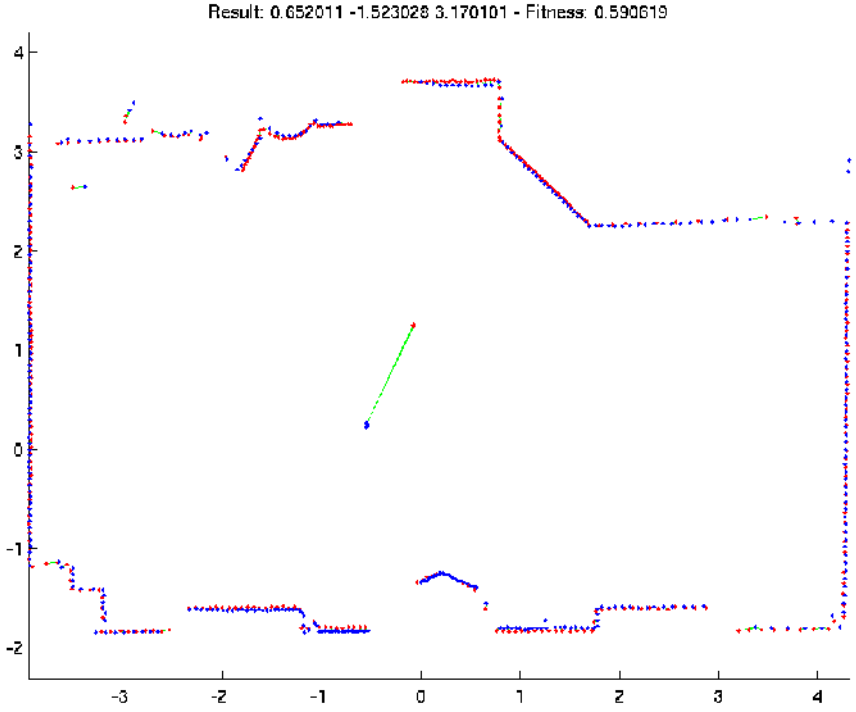


The correct match, including the manual match in the initial population.

On a first look, it might seem enough to manual check only those final matches with a fitness value too high; in fact, in the case above the fitness is ~40 times higher than the usual fitness when a match is correct. This is true in general, but it might happen that also matches with low fitness are not good enough and we might be interested in refining them. This is the case of the one in the picture below and, after inserting the manual alignment in the first population, a second run of genetic optimization obtained the desired result.



A match that could be improved.



The improved match, again by adding the manual match to the initial population.

Validation results

As we described previously, in LRF alignment for GT evaluation we defined a specific robot pose in the GTVframe as the LRFframe and aligned all scans with respect to this reference frame. A generic robot pose, to be compared with the validation, can be obtained combining first the roto-translation provided by the scan alignment with the roto-translation between the laser and the robot dima: the scan alignment gives out the current pose with respect to the reference one, but we know the reference one at the level of the dima, not at the level of the scan, i.e., we felt-tip pen marked the dima when the robot was in the reference position, not the scan; we therefore need to represent the current pose with respect to the dima. Then we have to combine this result with the roto-translation between the LRFframe and the GTVframe. in order to allow the comparison with the validation. This procedure turned out to be less straightforward then expected and spotted out some limitation of the LRF approach to GT gathering.

After composing the described roto-translations we obtained the following results, which constitute the validation of the laser-based GT system.

pose 1

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	1.2653	1.4030	1.2779	-0.1377	-0.0126
y	-1.1740	-1.1612	-1.1763	-0.0128	0.0023
th	3.4456	3.4382	3.4518	0.0074	-0.0062

pose 2

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	1.8838	1.9895		-0.1057	
y	-0.6341	-0.5965		-0.0376	
th	4.0013	3.9938		0.0075	

pose 3

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	2.2171	2.2794	2.1014	-0.0623	0.1157
y	0.0456	0.0667	0.0731	-0.0211	-0.0275
th	4.4645	4.4598	4.4593	0.0047	0.0052

pose 4

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	1.9801	1.9923		-0.0122	
y	0.7345	0.7182		0.0163	
th	5.1600	5.1512		0.0088	

pose 5

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	1.3285	1.3146	1.1441	0.0139	0.1844
y	1.3213	1.2932	1.2854	0.0281	0.0359
th	5.5105	5.5271	5.524	-0.0166	-0.0135

pose 6

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	0.8781	0.878		0.0001	
y	1.9556	1.9556		0	
th	5.1869	5.1869		0	

pose 7

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	0.5397	0.5509		-0.0112	
y	2.6828	2.6919		-0.0091	
th	5.1143	5.1093		0.0050	

pose 8

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	0.2290	0.2452	0.0784	-0.0162	0.1506
y	3.3570	3.3741	3.3479	-0.0171	0.0091
th	5.1503	5.1446	5.1766	0.0057	-0.0263

pose 9

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	-0.0079	0.0198	-0.1681	-0.0277	0.1602
y	4.1598	4.1953	4.1730	-0.0355	-0.0132
th	4.9086	4.9145	4.9311	-0.0059	-0.0225

pose 10

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	-0.2102	-0.2156		0.0054	
y	4.4968	4.4706		0.0262	
th	5.4880	5.4887		-0.0007	

pose 11

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	-0.7092	-0.7683		0.0591	
y	4.7668	4.8182		-0.0514	
th	6.068	5.4771		0.5909	

pose 12

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	-0.9756	-0.8205	-0.9415	-0.1551	-0.0341
y	4.2708	4.131	4.1329	0.13980	0.1379
th	1.7449	1.6365	1.6159	0.1084	0.129

pose 13

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	-0.7689	-0.5913	-0.747	-0.1776	-0.0219
y	3.5655	3.4626	3.4581	0.1029	0.1074
th	1.9824	1.9845	1.9759	-0.0021	0.0065

pose 14

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	-0.1228	-0.0798	-0.2444	-0.043	0.1216
y	3.1824	3.2223	3.1927	-0.0399	-0.0103
th	2.7651	2.8258	2.828	-0.0607	-0.0629

pose 15

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	0.039	0.167	0.0075	-0.128	0.0315
y	2.4743	2.3301	2.312	0.1442	0.1623
th	1.4604	1.4579	1.4676	0.0025	-0.0072

pose 16

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	-0.0228	0.1142	-0.0176	-0.137	-0.0052
y	1.5651	1.4267	1.4489	0.1384	0.1162
th	1.515	1.5142	1.4865	0.0008	0.0285

pose 17

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	0.1349	0.3058		-0.1709	
y	0.694	0.5841		0.1099	
z	0	0		0	
th	2.074	2.0738		0.0002	

pose 18

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	1.1491	1.34	1.224	-0.1909	-0.0749
y	0.47468	0.414	0.4313	0.06068	0.04338
z	0	0	0	0	0
th	2.6219	2.6169	2.611	0.005	0.0109

pose 19

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	1.841	2.005	1.8632	-0.164	-0.0222
y	-0.3749	-0.5015	-0.5286	0.1266	0.1537
th	1.9983	1.9909	1.9752	0.0074	0.0231

pose 20

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	1.207	1.201		0.006	
y	-0.8692	-0.9689		0.0997	
th	6.06	6.0561		0.0039	

pose 21

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	0.49296	0.4885		0.00446	
y	-0.18139	-0.1967		0.01531	
th	5.1983	5.1878		0.0105	

pose 22

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	0.5174	0.5887		-0.0713	
y	1.1805	1.2259		-0.0454	
th	4.4479	4.4449		0.003	

pose 23

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	-0.0333	-0.0708		0.0375	
y	2.1745	2.136		0.0385	
th	5.5533	5.3718		0.1815	

pose 24

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	-0.9847	-0.9824		-0.0023	
y	3.0299	3.0275		0.0024	
th	5.3534	5.3501		0.00330	

pose 25

	validation robot-frame	SICK genetic		valid - SICK genetic	
x	-1.5081	-1.4684		-0.0397	
y	4.2249	4.2727		1	
th	4.9689	4.9716		-0.00270	

pose 26

	validation robot-frame	SICK genetic	SICK scanmatch	valid - SICK genetic	valid - SICK scanmatch
x	-0.9872	-0.8123	-0.9437	-0.1749	-0.0435
y	4.8136	4.8442	4.8611	-0.0306	-0.0475
th	3.5275	3.5406	3.5344	-0.0131	-0.0069

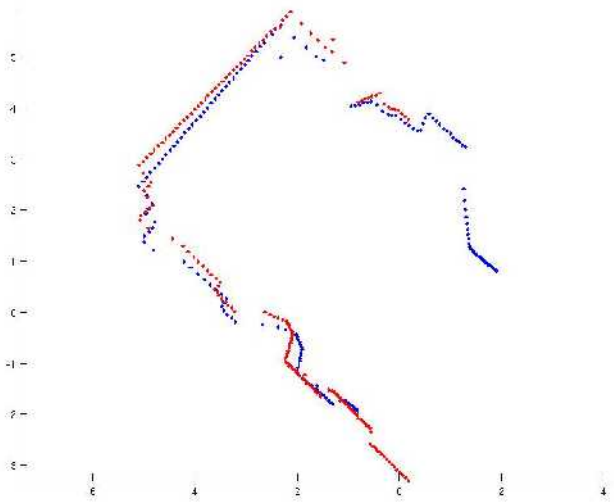
Table of results for the laser-based GT system, in the columns: the validation value, the LRF estimates computed with the genetic algorithm, the LRF estimates by the scan-matching algorithm and the errors of the two estimates with respect to the validation value.

	GT LRF Scanmatching				GT LRF Genetic				GT LRF Scanmatching			GT LRF Genetic		
	average Err	standard deviation Err	max of abs values Err		average Err	standard deviation Err	max of abs values Err		confidence interval for the mean (with 95% confidence level)			confidence interval for the mean (with 95% confidence level)		
x	0.04228	0.09042	0.1844	x	-0.06543	0.07787	0.1909	x	0.0045	0.0801	x	-0.0954	-0.0355	
y	0.05151	0.07420	0.1623	y	0.02695	0.06620	0.1442	y	0.0205	0.0825	y	0.0015	0.0524	
th	0.00444	0.04415	0.129	th	0.03287	0.12173	0.5909	th	-0.0140	0.0229	th	-0.0139	0.0797	

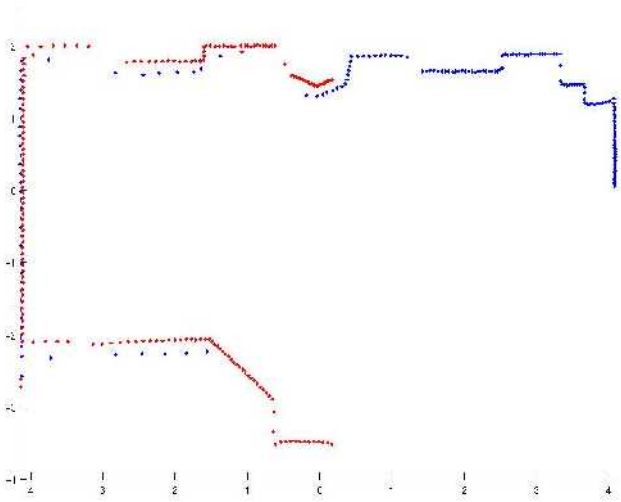
Tables with statistics of the two GT RLF methods. On the left two tables: the first column is the average error, the second the standard deviation of the error and the third is the maximum error committed. On the right table: the lowest and the highest values of the confidence interval for the errors.

These errors are comparable with the ones obtained by the vision-based GT system, while we were expecting a significantly higher accuracy. We were somehow puzzled by this result and decided to investigate a little bit this issue and, assuming the scan matches to be quite accurate, we focused on the sequence of roto-translations. Being the roto-translation between the LRFframe and the GTVFrame one of the validation poses, we were quite confident that the laser-to-dima roto-translation was the weak ring of the chain, although it was quite hard to make an error in such simple measurement.

To double check the quality of the roto-translation between the laser and the dima we manually inspected the result of plotting the second scan over the first one by using a sequence of validated roto-translations plus the laser-dima one. We took the first scan, and roto-translated it into the dima reference frame, we then roto-translated the result into the GTVframe (by means of a validated pose measure), then we roto-translated this result into the dima reference frame of the second scan (again by means of a validated pose measure), and finally, by using again the laser-dima roto-translation we arrived to the second scan. As it can be noticed in the following images the scans do not align even if we are using the validated roto-translations.



Scans from poses 13-14, not aligned although we are using validated poses



Scans from poses 6-2, not aligned although we are using validated poses

Apparently the error was in the roto-translation between the laser and the dima. We played a little bit with the dima values and two scan close each other: pose1 and pose2. Using the laser - dima manual measurements (X = 0.1780, Y = 0.1915, Theta = $\pi/2$) we had no proper alignment, as presented in the pictures above. On the other hand, using X = 0.090, Y = 0.150, Theta = $\pi/2$ we were able to align the scans. However, when we used this new dima-values for different poses we got back to the same issue. This analysis is obviously not conclusive, however a consideration can be drawn out of it: the manual tweaking of the laser-dima roto-translation might give correct scan matches, but at the cost of unreasonable physical displacements (we do not believe we can mistake the manual measure by 4 - 10cm), and these values do not give proper results for all scans.

A complete investigation of the limits of laser-based GT is out of the scope of this report, as we were only interested in determining its accuracy. Therefore we did not proceed in the analysis and decided the best we could do is to use the real roto-translation, as measured by hand using also the laser measurement device. Beside this, we conjectured two possible reasons for such an inaccurate solution:

- The first and simpler one, we are simply reaching the intrinsic limits of the laser scanner, as its measure is known to be depending on the surface reflectivity, and on incidence of the beam. Whenever the laser generates relevant errors, the scan matching procedure can generate errors instead of correcting them. On one hand, this is not convincing as it implies a non-white noise, but it is also true that the matter constituting the walls does not change its properties every few millimeters, which favors non-white effects.
- Secondly, the laser-based GT, and in general laser-scanning approaches, assume a complete 2D flat world; this is not true in general and this is the case of the GT room in particular; moreover perfect parallelism between the laser scanning plane and the floor is assumed, which is again to rely on a flat floor assumption. These generally agreed assumptions, when voided, might turn out in scan matching errors like the one we experienced since tilting (or even turning) the laser introduces distortions, i.e., non-white noise, in the measurements that, partially "compensated" by the automatic scan matching procedure, turn into pose errors.

Overall conclusions for indoor validation

The work performed, on both vision-based and laser-based GT systems, allows us to evaluate the performance of the two systems. Both evaluations are limited to to the accuracy aspects of the performance, and are based on the validation activity presented first in this document.

	GT Vision Stats			GT LRF Scanmatch			GT LRF Genetic	
	average Err	standard deviation Err		average Err	standard deviation Err		average Err	standard deviation Err
x	0.0804	0.03657	x	0.04228	0.09042	x	-0.06543	0.07787
y	0.0362	0.06777	y	0.05151	0.07420	y	0.02695	0.06620
th	0.0137	0.04618	th	0.00444	0.04415	th	0.03287	0.12173

The table summarizes the results of the GT systems, with respect to the validation data.

We can claim, as expected, that both approaches are good enough for being used for GT collection. The laser-based GT system is confirmed to be more accurate than the vision-based, although this larger accuracy is not as large as we were expecting. The laser-based system, instead, appears less prone to improvements in the accuracy, at least as long as it is not clarified whether the current performance is limited by the physical limits of the sensor or by the approach itself, i.e., the un-avoidable un-flatness of the floor where the robots will operate.

Also noticeable is the fact that the vision-based system is based on a component, i.e., the ARToolkit, that turned out less accurate than what the physical sensor would allow, so leaving space to a large increase the accuracy of vision-based GT systems. To evaluate this potential, we performed a comparison between the ARToolKit outcome and the outcome of the localization of the calibration pattern, a part of the camera calibration process, and it turned out that the errors in the marker localization are significantly higher then the localization errors of the chessboards used for calibration.

It also remains confirmed that we can have a double GT stream, one available for all users of the dataset, i.e., allowing comparisons among all onboard sensors, including LRFs; the other, more accurate, but generated on the basis of the onboard LRFs, and therefore useful only for evaluating the performance of the approaches not making use of that stream.

Outdoor GT Validation

The outdoor GT system of RAWSEEDS is a Real Time Kinematic (RTK) GPS system. As from Wikipedia:

> *"Real Time Kinematic (RTK) satellite navigation is a technique used in land survey based on the use of carrier phase measurements of the GPS, GLONASS and/or Galileo signals where a single reference station provides the real-time corrections of even to a centimeter level of accuracy. When referring to GPS in particular, the system is also commonly referred to as Carrier-Phase Enhancement, CPGPS. Standard satellite navigation receivers compare a pseudo-random signal being sent from the satellite with an internally generated copy of the same signal. Since the signal from the satellite takes time to reach the receiver, the two signals do not "line up" properly, the satellite's copy is delayed in relation to the local copy. By progressively delaying the local copy more and more, the two signals will eventually line up properly. That delay is the time needed for the signal to reach the receiver, and from this the distance from the satellite can be calculated.*
>
> *The accuracy of the resulting range measurement is generally a function of the ability of the receiver's electronics to accurately compare the two signals. In general receivers are able to align the signals to about 1% of one bit-width. For instance, the C/A signal sent on the GPS system sends a bit every 0.1 microsecond, so a receiver is accurate to 0.01 microsecond, or about 3 meters in terms of distance. The military-only P(Y) signal sent by the same satellites is clocked ten times as fast, so with similar techniques the receiver will be accurate to about 30 cm. It is important to note that other effects introduce errors much greater than this, and accuracy based on an uncorrected C/A signal is generally about 15 m. RTK follows the same general concept, but uses the satellite's carrier as its signal, not the messages contained within. The improvement possible using this signal is potentially very high if one continues to assume a 1% accuracy in locking. For instance, the GPS C/A signal broadcast in the L1 signal changes phase at 1.023 MHz, but the L1 carrier itself is 1575.42 MHz, over a thousand times faster. This corresponds to a 1% accuracy of 19 cm using the L1 signal, and 24 cm using the lower frequency L2 signal.*
>
> *The difficulty in making an RTK system is properly aligning the signals. The navigation signals are deliberately encoded in order to allow them to be aligned easily, whereas every cycle of the carrier is similar to every other. This makes it extremely difficult to know if you have properly aligned the signals, or are "off by one" and thus introducing an error of 20 cm or a larger multiple of 20 cm. This integer ambiguity problem can be addressed to some degree with sophisticated statistical methods that compare the measurements from the C/A signals and by comparing the resulting ranges between multiple satellites. However, none of these methods can reduce this error to zero.*
>
> *In practice, RTK systems use a single base station receiver and a number of mobile units. The base station re-broadcasts the phase of the carrier that it measured, and the mobile units compare their own phase measurements with the ones received from the base station. This allows the units to calculate their relative position to millimeters, although their absolute position is accurate only to the same accuracy as the position of the base station. The typical nominal accuracy for these dual-frequency systems is 1 centimeter ± 2 parts-per-million (ppm) horizontally and 2 centimeters ± 2 ppm vertically. Although this limits the usefulness of the RTK technique in terms of general navigation, it is perfectly suited to roles like surveying. In this case, the base station is located at a known surveyed location, often a benchmark, and the mobile units can then produce a highly accurate map by taking fixes relative to that point. RTK has also found uses in autodrive/autopilot systems, precision farming and similar roles."*

The error on the estimation of the relative position of the mobile unit (with reference to the base station)

given by an RTK GPS system is good when compared to typical precision standards for mobile robotics. However, as the description above clearly states, this level of precision applies also to the estimation of the absolute position of the mobile unit only if the fixed offset due to the error in the localization of the base station has been removed. To do so, the actual position of the base station on the Earth's surface must be known. In the case of RAWSEEDS, the base station's assumed position is calculated by the base station itself using the available (non-RTK) GPS signals, and then held valid for the entire duration of the current data-gathering or GT-validation session. This introduces an unknown offset in the absolute position of the robotic mobile unit (from now on called also "**rover**") but - crucially - this offset is constant over the course of a single acquisition session (i.e., until the base station is turned off or its position is re-evaluated).

The above geometric offset is absolutely not an issue, because within RAWSEEDS the RTK GPS positioning data are not used to determine the trajectory of the rover with reference to an absolute, geo-referenced frame. They are used, instead, to specify the actual trajectory of the robot with reference to the environment that it explores during a data-gathering or GT-validation session, and it is expected to be confronted with the data obtained from every other source of robot positioning information that can be implemented by using the data of the robot's onboard sensors: be such source presently available or not yet devised. In the starting instant of each session, the position of the robot within the environment must, by definition, be the same when evaluated by the RTK GPS system and by any localization algorithm applied to the robot's data: this, in turn, brings to zero the geometric offset of the position estimated by the RTK GPS system with reference to the data produced by any localization systems that uses the robot's own sensor data: which is exactly what is needed by RAWSEEDS. The fact that the RTK GPS ground truth data associated to RAWSEEDS' outdoor datasets include (when considered as an absolute measure of the robot's position on the surface of the Earth) a constant offset is, in the context of RAWSEEDS, unimportant.

Limits of classical GPS localization systems

GPS localization systems are heavily dependent on the actual satellite *constellation* visible from the point of measurement in the specific instant considered. GPS satellites are not geostationary, and their geometric configuration is subject to large variations within the day, and from a day to the other; moreover, this configuration depends heavily by the location where the receiver is.

Geometrically, a GPS receiver needs to receive signals from at least 3 satellites to calculate its position (to be precise, the position of the GPS antenna attached to it) on the surface of Earth. Such a calculated position is called a "GPS fix" or simply a "fix": this term will also be used in this document. In real-world conditions, due to limitations in the precision of time synchronization between receiver and satellites, a stand-alone GPS receiver is able to reconstruct its position if at least 4 satellites are visible. Satellites which are too close to the horizon have to be discarded because they lead to unreliable estimates: satellites which are low above the horizon must be ignored because the thick layer of atmosphere that the signal has to cross distorts the signal itself, moreover, the peculiar geometric configuration of such satellites greatly reduces estimation precision. Concluding, the above figure (minimum 3 satellites) actually applies to "well-positioned" satellites only.

For RAWSEEDS, the threshold over which the position of a satellite was considered acceptable was set to 10° over the horizon. With this elevation threshold, at the time of our acquisition sessions (spring and summer of 2008) a GPS antenna positioned in an optimal location usually received signals from 6 to 9 satellites, with a typical peak value of 8. This number is rapidly variable during the day, so planning on acquisition sessions based on satellite visibility was, for RAWSEEDS, essentially impossible. In the section "Setup of the GT system", below, an example graph of satellite visibility over the course of a typical day will be given.

An *optimal location* is, in this context of GPS reception (and considering the urban setting of RAWSEEDS' operations), the top of a building tall enough that any obstruction to satellite visibility falls below the

elevation threshold. Such obstructions can be other buildings, large objects or even thick foliage on trees: we verified that even the hand of a man, if put between the antenna and a satellite near the antenna, can disrupt GPS reception. The base station of the RTK GPS system used for the gathering of outdoor GT data within RAWSEEDS was set in an optimal location. Of course the same kind of positioning was not possible for the rover that, especially when its path ran close to buildings, frequently lost contact with satellites. Outdoor situations where the rover received less than 4 satellites were frequent.

All in all, the usability of GPS systems for robot localization in urban environments is barely acceptable: something that we already knew from the start, but that we considered acceptable for the specific needs of a once-upon-a-time collection of pose GT. It must be underlined that - exactly for the above reason - RAWSEEDS did not plan to use such a system for odometry: instead, it was planned to be used as a source for pose GT data on a subset of the robot's complete trajectory. In this role the RTK GPS system proved to be a very effective way to get precise trajectory data.

RTK GPS systems are even more critical than single-receiver GPS systems in their requirements on satellite visibility. For RTK operation, a minimum of 5 satellites must be available above the elevation threshold. Moreover, the *same* five (or more) satellites must be visible both by the base station and by the rover: so, if the base station is not set in an optimal position (i.e., a position where its antenna "sees" *all* the satellites that are above the horizon), it is frequent the case when both the rover and the base station receive 5 or more satellites, but RTK operation is impossible nonetheless because the sets of received satellite do not overlap fully. This is the reason why RAWSEEDS ensured that its base station was always set in an optimal position.

When RTK GPS is operational, two levels of localization performance can be defined, in correspondence to the so-called "RTK Float Integers" and "RTK Fixed Integers" conditions. These conditions, where the second one gives the best precision, are characterized by the fact that the RTK GPS positioning algorithm has reached or not an optimal estimate of its parameters, which can require time (typically a few tens of seconds) from the instant when RTK operation becomes available. More details about this will be given in the following sections.

Best positioning accuracy in RTK GPS Fixed Integers condition is 2cm + 2ppm of baseline length, where the latter is the distance between base station and receiver. (Source: specifications of the Trimble 5700 GPS receiver used by RAWSEEDS. System used in "Low Latency" mode, where accuracy is slightly reduced in exchange for reduced latency between reception of satellite data and calculation of position. Disabling the Low Latency mode, accuracy grows to 1cm + 1ppm of baseline length). In the case of the GT validation tests performed by RAWSEEDS, baseline length was 250m maximum, so the second term of the above sum is negligible with respect to the first. A characterization of the real-world positioning performance of RAWSEED's RTK GPS setup will be given in the following sections.

The validation procedure chosen by RAWSEEDS has been focused on the verification of all the conditions needed to reach the performances described above in the areas devoted to data gathering, i.e., in typical urban outdoor and mixed (indoor/outdoor) environments. In other words, given the incredible amount of effort the GPS community put in the evaluation of the accuracy of their products, we believe we can trust the specifications, but we need to check that the hypotheses are true.

In particular, we focused the analysis on the worst case scenario of "mixed" data acquisition, where indoor and outdoor tracts are interleaved and both buildings and vegetation are close to the trajectory of the robot.

Setup of the GT system

There are several different manufacturers of RTK GPS positioning system. While performing preliminary market research in the field, which included talking with professionals working in the field of land survey, it emerged that the performance of such products is mainly defined by the physical limitations inherent in the

type of signals and algorithms used, due to the maturity of the associated technology. For this reason, we consider the actual performance obtained by our setup as representative of the state of the art of practical implementations of the whole RTK GPS concept, much more than of the characteristics of one specific maker's products.

Trimble devices were chosen for the RTK GPS system to be used by RAWSEEDS: these are well-established commercial products, and their data sheets can be downloaded from http://www.trimble.com. Specifically, the system was composed of:

- a MS750 GPS receiver, used as base station in conjunction with a Zephyr GPS antenna and a PDL450 2Watt, 450MHz radio-modem with whip antenna;
- a 5700 GPS receiver, used as rover, mounted on the robot and equipped with a Zephyr GPS antenna and an internal 450MHz radio receiver (with rubber antenna) for connection to the base station.

The base station was powered by a 12V battery, while the rover was connected to the robot's 24V batteries. The GPS antenna of the base station was set on a tripod, while the one on the robot required a short pole to be distanced from the onboard PCs, which generated radio interference signals significantly lowering the reception capabilities of the antenna. The rubber antenna of the rover's radio-modem benefited slightly from being mounted on the above pole too. Below you can see images of the base station in its working location, and of the GPS-equipped Robocom robot.

Base station in its working location (left), and GPS-equipped Robocom robot (right).

A satellite elevation threshold of 10° was configured both for the base station and for the rover, to avoid corrupting the position data by incorporating information from ill-positioned satellites. No threshold was specified for positioning data quality (PDOP: Position Dilution of Precision), to maximize the time intervals when a position estimate was available. This choice was made after verification of the difficult GPS satellite reception in the test environments, and of the good precision of the calculated positions (when available). As the actual error on position estimate is available as part of the receiver's output, the above choice introduced no undocumented errors and widened the time intervals when robot position was known.

In preliminary tests the base station was set on the roof of a building which had a distance of 500m from the farthest test areas. While in theory the radio-modem used should have had no problems with such a baseline length, in practice reception of the correction signal from the base station proved to be erratic. The cause of this phenomenon was perhaps a combination of heavy obstructions (buildings) and strong radio interference, as the tests were performed in an area (Politecnico di Milano Campus) where many radio sources were active for transmission and research purposes.

We decided, then, to mount the base station in a new location: a 5-storey building set within the campus area where the experiments had to be conducted. This new location proved to be optimal, and the rover did not lose the correction signal from the base station any more, even within buildings (where, of course, it was of no use lacking the GPS signal). Satellite reception from the base station was optimal, so positioning quality was only limited by the number and position of the satellites visible from the rover. With the new setting of the base station, maximum rover distance between it and the rover was 250m: more information about the positioning of the base station with reference to the data gathering areas will be given in the following sections.

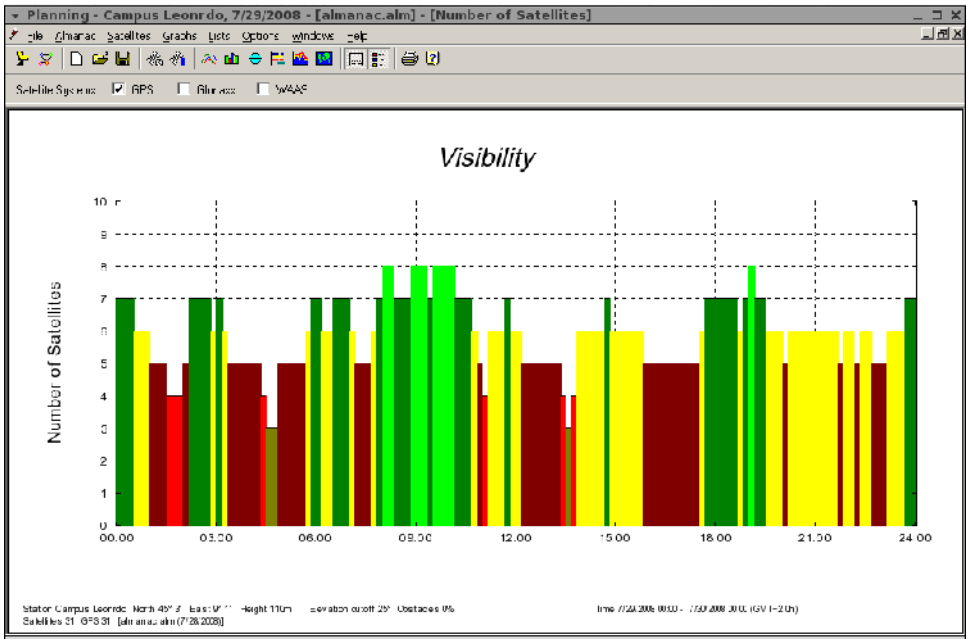
Collection of the validation data

The GPS receiver autonomously estimates the quality of the data it produces, and outputs the position error statistics for every measure, so it's possible to associate an uncertainty ellipse to every GPS measurement (RTK or not). The 1-sigma error ellipse is trustworthy, as the algorithms used to compute the position and the uncertainty are published and validated by several studies carried on since the introduction of the GPS.

As noted before, to reach a RTK GPS fix both the BS (Base Station) and the R (Rover) must use the same constellation of 5 satellites, that is, must see and use the same set of 5 satellites. This poses a serious constraint on the RTK GPS covered areas due to the masking effects of the buildings that surround the rover.

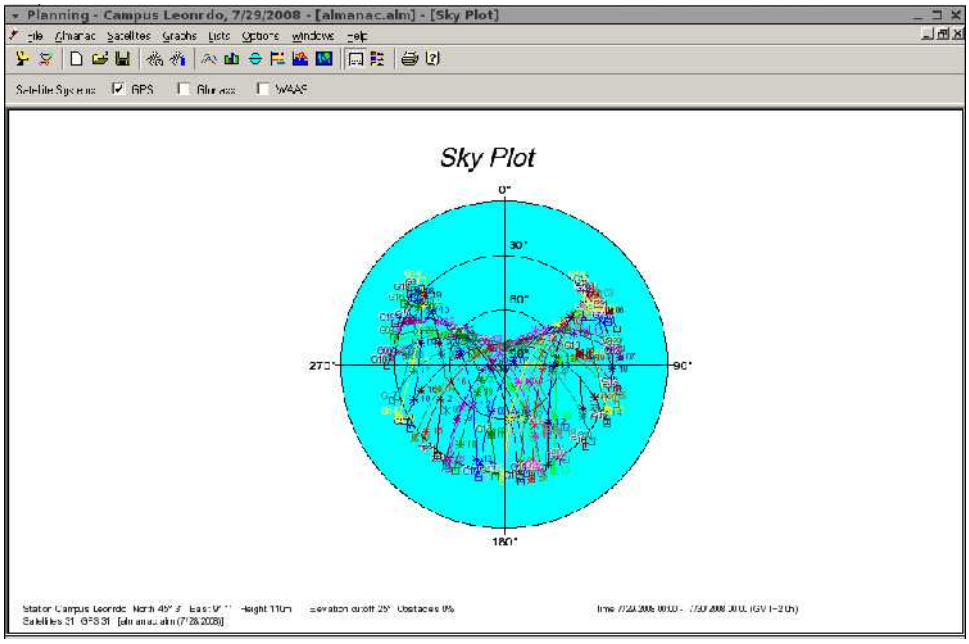
There are 2 different kinds of RTK GPS fix obtainable by the receiver: a float and a fixed differential carrier phase solution. The difference between the 2 lies in the *integer ambiguity problem* solution: in the former the receiver has found a solution, but doesn't know yet the exact integer number of wavelengths that constitutes the offset, while in the latter knows this datum exactly. For our purposes the only kind of fix of any usefulness is the fixed one, as it's the only kind of fix that guarantees a sufficient precision (< 10 cm) for the GT system. That's why it's imperative to plan the acquisition in such a way to maximize the area covered by an integer RTK GPS fix.

The acquisitions are planned using Trimble's Planning software, with a "maximum number of satellites visible" criterion see picture below.



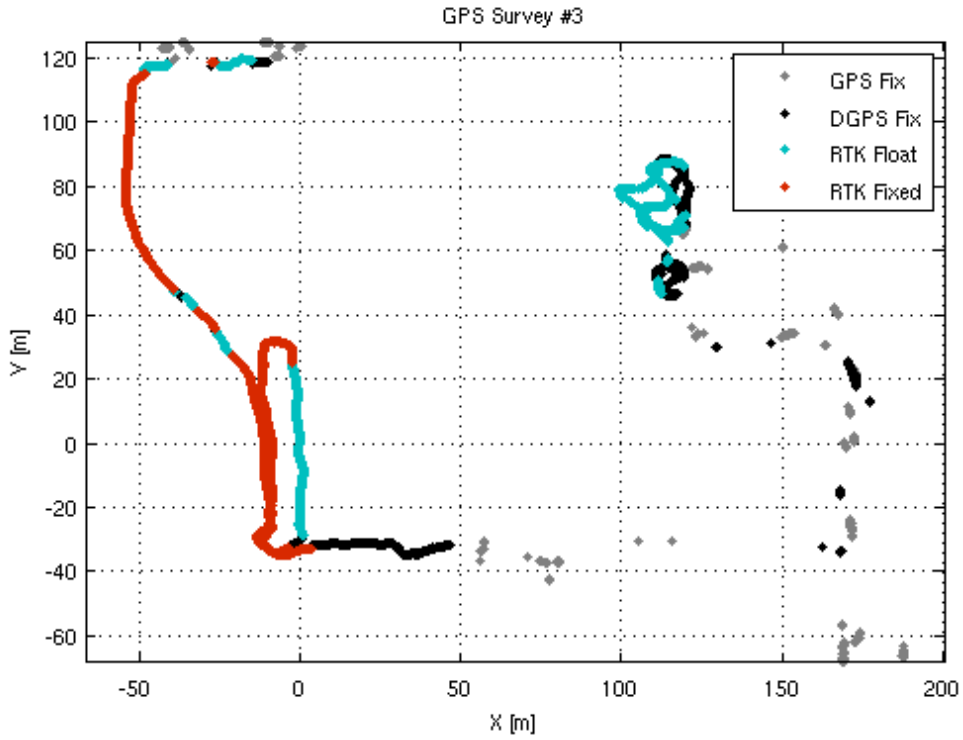
The Trimble Acquisition Planning software

In the Sky Plot, see picture below, is possible to see that the South direction is a preferred one, as there is a hole in the coverage facing the north direction. This polar chart represents the orbits of the different satellites (each colored arc) during 24 hours. The angle represents the azimuth, the radius the elevation. So (0,0) is the horizon looking at north, (90,0) is the horizon looking at east, (0,90) is the zenith. The optimal instants for the acquisition changes from time to time because the orbits of the GPS satellites are not geostationary.



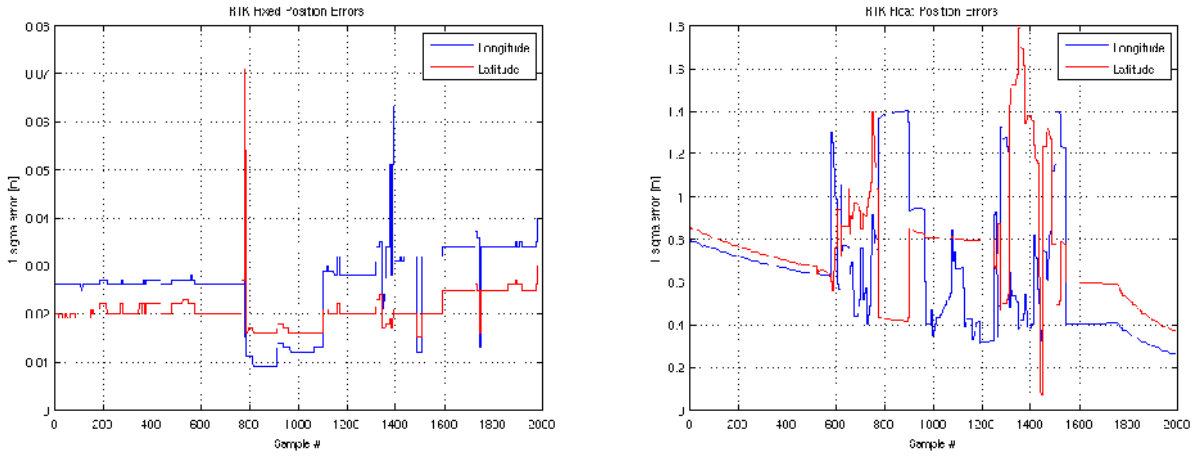
The Sky Plot given by the Trimble Acquisition Planning software

As an example of the performance achievable with the RTK GPS system, one of our preliminary GPS surveys (Survey #3) is reported in the picture below.



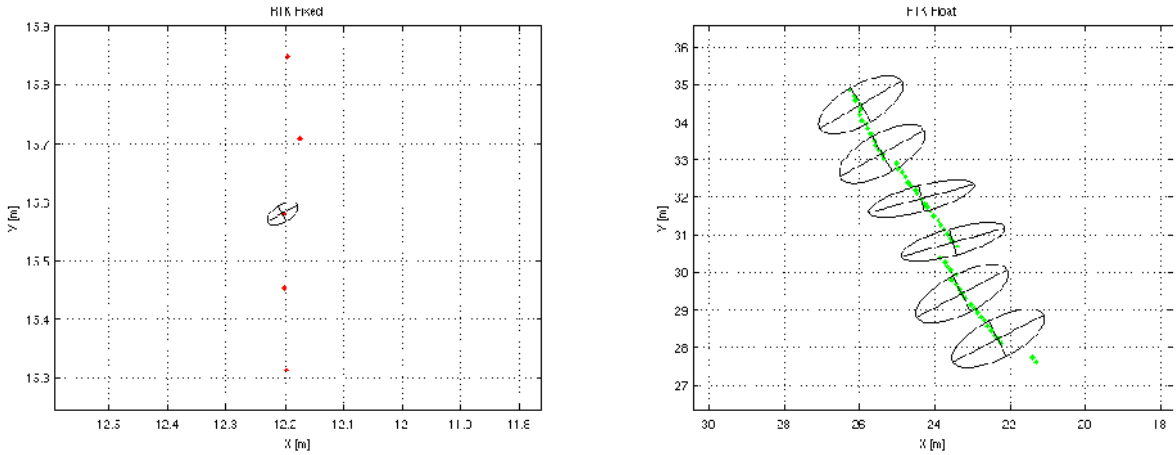
As an example of the performance achievable with the RTK GPS system, one of the preliminary GPS surveys (Survey #3) is reported.

The areas of interest are the cyan and red ones, that are the RTK covered areas. In the following graphs the Position Errors in longitude/latitude are plotted for the RTK fixed and RTK float fixes. Please note the difference in the magnitude of the errors in the 2 different fix states.



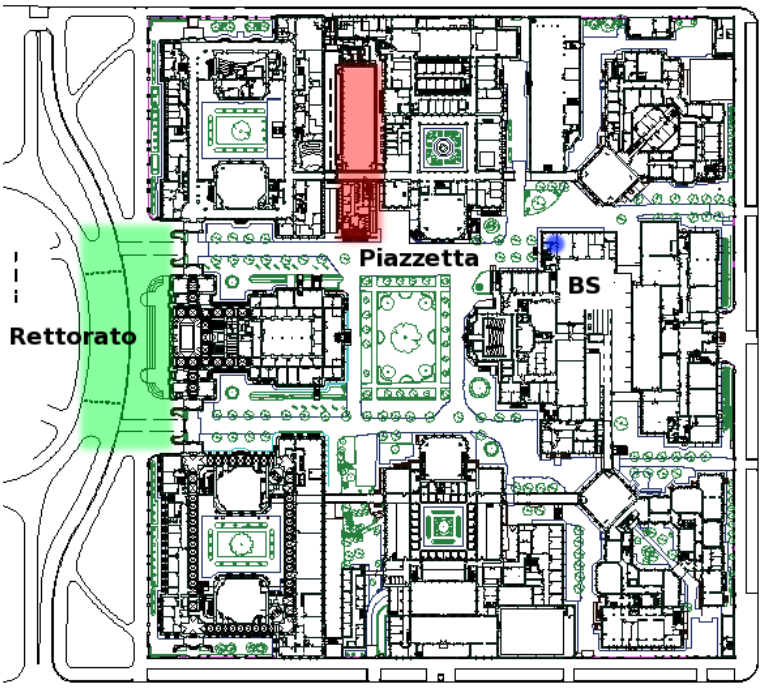
RTK-GPS errors in the Leonardo Campus, with (left) and without (right) integer fix

A sample of the error ellipses are displayed for each state in the following pictures, just to show at a glance the entity of the error.

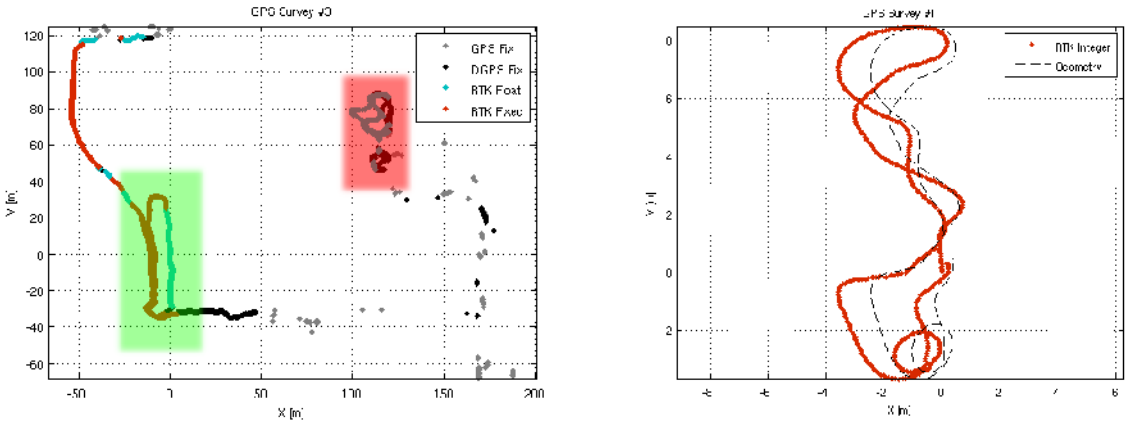


RTK-GPS errors ellipses in the Leonardo Campus, with (left) and without (right) integer fix

From a survey taken roaming around the Campus Leonardo with the robot, 2 different areas resulted reliable for obtaining an integer RTK GPS fix, probably because of the absence of obstacles in the south direction. In fact, due to the inclinations of the orbits of the satellites, in Italy the South is a preferred direction in terms of number of satellites visible. These areas are reported in the picture below. The first area (green shaded), called "Rettorato" from now on, is a wide open area within a square, placed just out of the east side of Campus Leonardo. Nearby obstacles are limited to a short building in the west direction and far away (> 100m) buildings in other directions. The second area (red shaded), called "Piazzetta" from now on, is a rectangular square placed on an elevated position along the north-south direction inside the Campus Leonardo.



Areas covered by the RTK-GPS integer fix signal, in the mixed dataset location.



Left: the quality of the RTK-GPS in the two areas (Rettorato on the left in green, and Piazzetta on the right in red). Right: Odometry and GPS superimposition.

Conclusions for outdoor validation

From the preparatory and preliminary data acquisition sessions, aimed at the validation of the RTK-GPS-based GT system, it emerged that the experimental data collected perfectly fits with the literature and published specifications. The RTK-GPS-based GT system was tested and installed, and it satisfies the required accuracy for the RAWSEEDS outdoor data sets; we also verified the coverage of satellites in the acquisition areas so to provide the prescribed accuracy.

Initially we were intended to validate the RTK-GPS accuracy by using laser scan-matching, but the considerations done in the indoor section regarding the errors induced by improper surface reflections and robot and/or wall inclinations were going to be even more relevant for the outdoor scenario.

During the data collection sessions the robot logs all data necessary to evaluate the quality of the GT. Besides the coordinates (and their associated uncertainty) it logs several other parameters including:

- # of satellites used to calculate the fix
- Quality Index (that is the kind of fix)
- HDOP (Horizontal Dilution Of Precision, a figure of merit of the geometric strength of satellite configuration)
- height (and its associated uncertainty) above the reference WGS84 ellipsoid that models the Earth.

These information will be provided together with the GT data in the RAWSEEDS datasets, in order to give a proper estimate of the accuracy of the Ground Truth at the time of data acquisition.

References

- L. Matthies and S. Shafer, "Error Modeling in Stereo Navigation", IEEE Journal of Robotics and Automation, vol. RA-3, n. 3, June 1987, pp. 239 - 250
- A. Censi, "Scan matching in a probabilistic framework", in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 2291 - 2296, Orlando, Florida, 2006
- E. Woods, P. Mason, M. Billinghurst, "MagicMouse: an Inexpensive 6-Degree-of-Freedom Mouse", in Proceedings of Graphite 2003, 2003, Melbourne
- G. Schweighofer, A. Pinz, "Robust Pose Estimation from a Planar Target", IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 28, Issue 12, Dec. 2006, pp. 2024 - 2030