Non-constant Stereo Matching Approach for Feature Initialization in Inverse-Depth Monocular SLAM

Eric Guerrero Font

1. Framework

SLAM (Simultaneous localization and mapping) is a sequential probabilistic filtering approach of the SFM (Structure from motion) problem. It sequentially updates the current location of the camera and the position of the features on the map.

Since the localization is an important problem in autonomous mobile robotics, it is a focus of actual research and several approaches have been developed in order to solve the problem of localization of a robot in a not previously known map.

The probabilistic filter is usually based in EKF (Extended Kalman Filter), the nonlinear version of the Kalman filter, which linearizes about an estimate of the mean and covariance. EKF with Euclidean representation is very efficient to estimate positions for near features, which show high parallax, but is not for low parallax features. The problem is that it assume a Gaussian PDF, and the real uncertainty for low parallax features has not a Gaussian distribution, it rise sharply at a minimum depth and goes slowly to infinite. The key concept for coping with this problem is direct parameterization of the inverse depth relative to the camera position.

There are two main approaches for the feature initialization for monocular SLAM, the delayed and the undelayed initialization. The delayed approach treat new features separately from the main map to reduce the depth uncertainty before the insertion on the main map, the new features don't contribute on camera position until they are included, and the low parallax features over many frames are usually rejected. The undelayed approach use the low parallax features as references for camera orientation. The approach of this project is based on the former.

This project is based on the previous work of E. Guerra, R. Munguia and A.Grau [1], where they used a second monocular camera worn by a human to get a stereo delayed inverse-depth feature initialization and produce estimation for depth when the cameras share common features and they have sufficient parallax.

2. Objective

The objective of this project is to provide a tool to improve the global performance of the Monocular SLAM only computing the stereo matching when the fields of view of the two cameras intersect.

This tool has been designed and tested with offline sequences on MATLAB, for online testing it should be implemented in ROS.

The main topics of the project are the following:

1) Representation of the fields of view of the cameras, the pyramids, and the intersection of them.
2) Implementation of a stereo matching technique to update the position of listed features.

The stereo matching is computed only when the fields of view coincide and also there is a previously computed SLAM feature located in the common region.

3. Methodology

The algorithm has two conditions to start computing the stereo matching and returning the feature depth update. It is computed only if there is an intersection between the fields of view and also if there are listed features inside of it. It is assumed that the monocular SLAM algorithm doesn't need a continuous update, therefore the method can have some rigid conditions to avoid outliers, not all the features inside the intersection have to be estimated.

The algorithm starts computing the shape and the pose of the two fields of view. Then, through some geometric manipulations the polytope that represents the common field of vision is obtained.

Since the implementation has not been done already with the Monocular SLAM algorithm, the list of features is simulated as a matching of points of interest of the two images using SURF descriptors and the projection of them onto world coordinates. Some noise with different variance for each feature has been added.

Having the list of features and the intersection polytope, the algorithm searches for an update of the world position of this features only if they are located in the common region.

After passing this condition the stereo matching algorithm starts. First read the images, and then each feature is represented as a pixel in the monocular SLAM camera image, I1, and as a search space in the auxiliary camera image, I2. The BRIEF descriptor has been developed to match the pixel in I1 with another pixel of the search space in I2.

Once the matching is done the 3D pose can be obtained projecting the points and this estimation of the pose can be returned to the monocular SLAM algorithm.

In the next pages you can find a more detailed explanation of each stage.

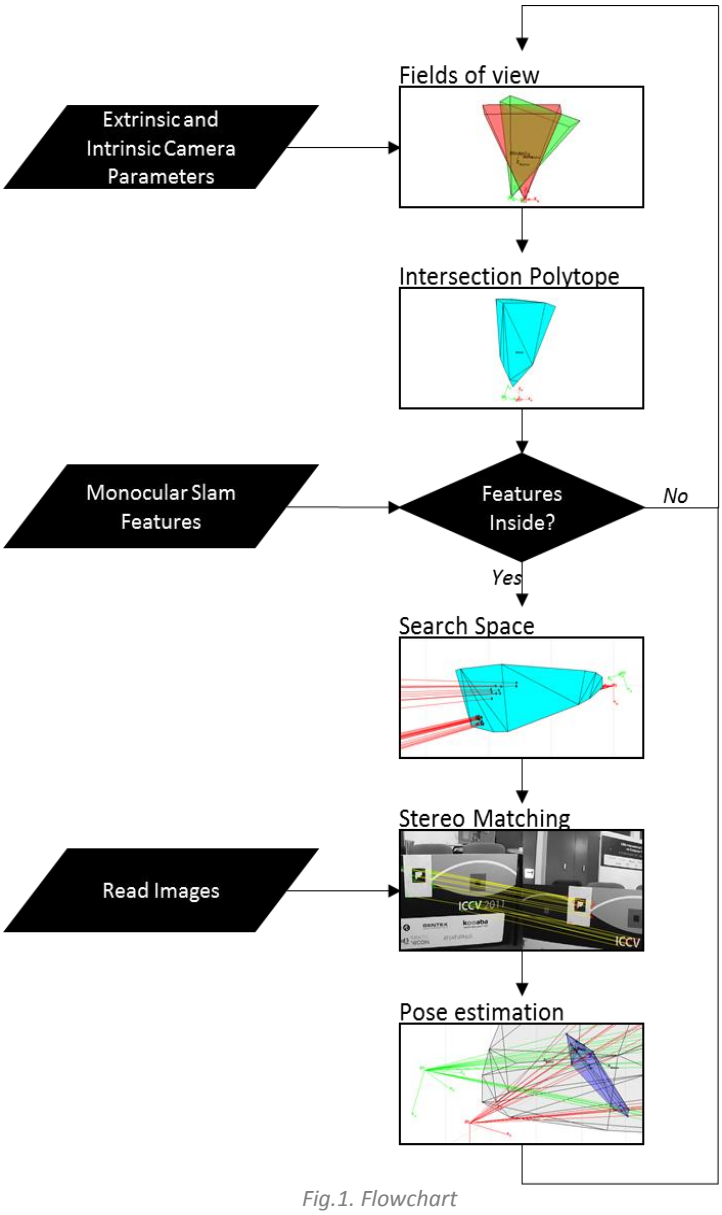

Fig.1. Flowchart

3.1. Fields of view

First of all the shape of the field of view of each camera has to be computed. This shape is a pyramid that depends on the intrinsic parameters of the camera, and the maximum depth. The position and orientation of the camera centers are provided by sensors, an IMU and a laser range finder in the real application, an AR marker in this implementation, world coordinates are taken respect to the robot.
The maximum depth on the field of view has been approximated with the relation between the positions of the two centers of the cameras, and the minimum parallax that is required.

$$d_{max} = \frac{\|p_{c1} - p_{c2}\|_2}{2\ tg(\frac{\varphi_{min}}{2})}$$

Having the top vertex of the pyramid in the position of the center of the camera, the other vertices are found applying the projection of the corners of the image to the maximum depth considered. A camera pinhole model is assumed.

$$[X\ \ Y\ \ Z\ \ 1]^T = P^{-1}\ [u\ \ v\ \ d_{max}\ \ ]^T$$

3.2. Intersection

Once the shape, position and orientation of each field of view has been computed it's time to compute the polytope that represents the intersection that could exist between them.

Considering that each pyramid is composed by triangular planes and line segments, the vertices of the intersection polytope are the points of intersection of all the segments of one pyramid with all the planes of the other pyramid and vice versa, plus the vertices of one pyramid inside of the other. The Line Segment–Triangle Intersection [4] approach has been implemented.

This approach is based in that any point of the triangle can be defined in terms of its position relative to each one of the three vertices in barycentric coordinates (w,u,v) . After solving the intersection of the line segment with the plane, the barycentric coordinates of the intersection respect to the triangle vertices gives a measure to know if the intersection is inside or not.

The polytope is represented with all the previously computed vertices.

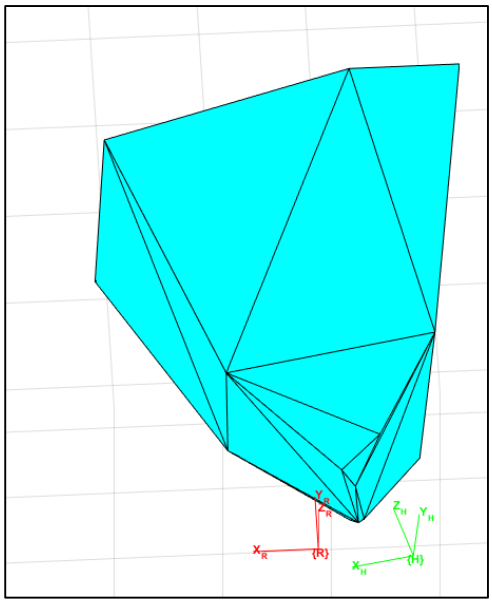

Fig 2. Intersection Polytope

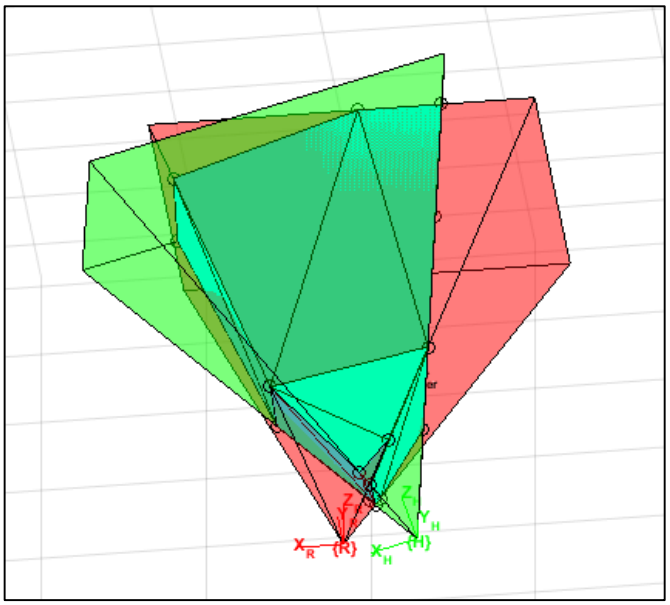

Fig 3. Pyramids and Intersection Polytope

3.3. Stereo Matching

A matching technique between images has to be performed to give an update of the features listed by the monocular SLAM that are viewed by the two cameras. Since we have an estimation of the feature localization respect to the first camera where we only consider uncertainty about the depth of this feature we can get the projection in image 1 and also a descriptor of this position. In order to reduce the search space for the matching of the feature in image 2 the following steps have been followed.

- Get estimated 3D pose of all the listed features that are inside of the intersection.
- Compute ray from the center of camera 1 to feature.
- Get intersection ray-polytope.
- Project intersection points in image 2 (Search region)
- Project features in image 1 and get descriptors. (BRIEF)
- Match descriptors in search regions.

3.3.1. Search Space

The search space represents the image area in which a match of a descriptor from other image can be found, considering that there is uncertainty in the depth computed by the monocular SLAM.
Two approaches have been developed.

3.3.1.1. Intersection with polytope

In order to find the search region for each feature, a ray from the center of the camera 1 direction to each feature is computed, the points where the ray intersects the polytope represent the margins where the real feature could be located. Projecting the intersection points onto image 2 a line that joins the two points is defined. It defines the search space.

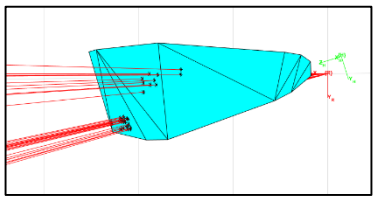

Fig 4. Polytope/Ray intersections per feature

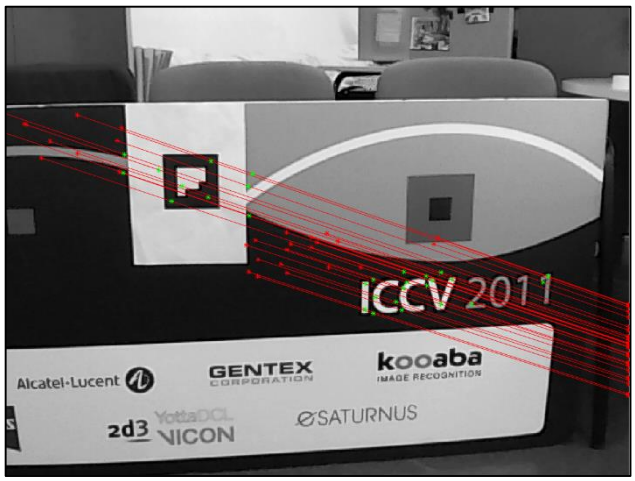

Fig 5. Search Space approach 1.

3.3.1.2. Depth with uncertainty

In this case, the method is based in that the variance of the depth of each feature is already computed. So providing a confidence margin for this variance, the search space is given by the points that are located in the extremes of the margin. If we project them onto the image 2 we have the following.

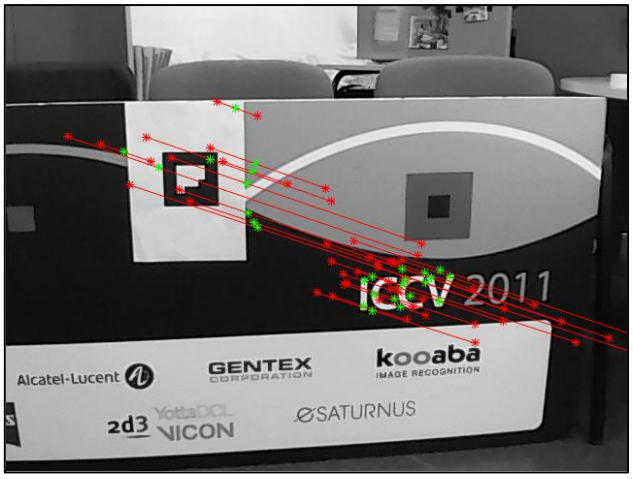

Fig 6. Search Space approach 2.

3.3.1.3. Search space method applied

The method applied has been the combination of both. The confidence intervals are computed and projected to the image if all the extremes of this intervals are located inside of the intersection polytope, in the case that an extreme is located outside, the first method is applied and the intersection of the correspondent ray with the polytope is used.

The search region that now is represented as a continuous line has to be discretized to pick only the pixels that are near to the line. To do that, the parameters of the line are computed and the pixels that are selected as candidates for matching are the ones that are inside the rectangle determined by the two extremes and also satisfy the following condition:

$$|Mx + N - y| < \varepsilon \quad where \begin{cases} x, y & Pixel\ Index \\ M, N & Line\ Parameters \\ \varepsilon & Threshold\ (1\ pixel) \end{cases}$$

A precision factor can be tuned to vary the number of points representing the search space.

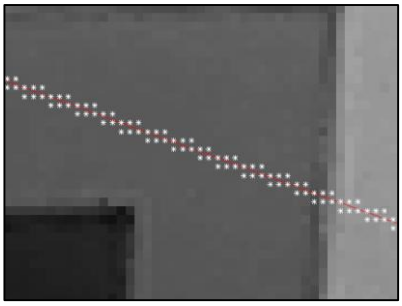

Fig 7. Search Space discretization.

3.3.2. BRIEF descriptor

In order to match the listed points in the first image with some point of the search regions of the second image an image descriptor is needed. A Binary Robust Independent Elementary Features (BRIEF) descriptor [4] has been developed in order to have it integrated with the search space algorithm.

The method is based on comparing intensities of different points of a patch. Steps followed:

- Smooth the image with a Gaussian filter.
- Predefine a random list of 128 pairs of points, each one located inside a square of the same size of the patch in this case 32 pixels. The pairs follow a Gaussian distribution $\left(0, \frac{1}{10}S^2\right)$ were S is the size of the patch.
- Build descriptor for each patch **p** as:

$$\tau(\mathrm{p}; \mathrm{x}, \mathrm{y}) = \begin{cases} 1 & if\ \ p(x) < p(y) \\ 0 & otherwise \end{cases}$$

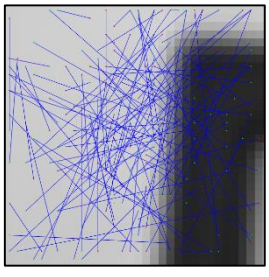

Fig 8. BRIEF descriptor Patch.

3.3.3. Matching

At this point we have a feature point in I1, a search space in I2, and a defined descriptor. Once the descriptor of the pixel in I1, and the descriptors of all the pixels in the search space of I2 have been computed, a cost function has to evaluate which is the best match of all the descriptors in the search space with the descriptor of the pixel in I1. Since the BRIEF descriptor is not rotational invariant, the patch used for the I1 is rotated for the I2 with the known rotation of the two cameras respect to the Z axis. The cost has been computed as 1-norm between vectors, and a threshold has been imposed in order to avoid outliers.

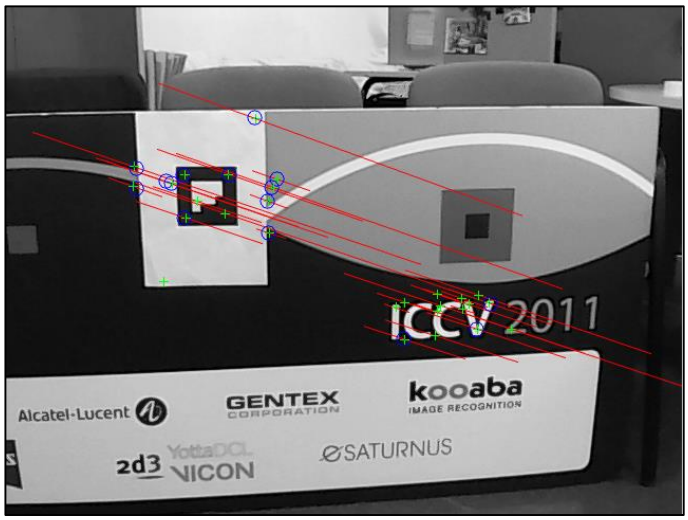

Fig 9. Matching.

In the figure one might see where the match should be, green cross, and where it is, blue circle. There are some features without match since they didn't pass the cost threshold.

3.4. Feature position

Having the matched features between images we are able to get the 3D position of them. To do that a ray for each feature and for both images is computed. This two lines per matched feature, one per image, have initial point at the center of projection of the respective cameras, and the direction of the located feature.
Since the rays are not perfectly crossing with each other, the estimated 3D point per feature has been considered as the point with minimum distance respect to the rays. This is the midpoint between the projections over each line of the nearest point between them.

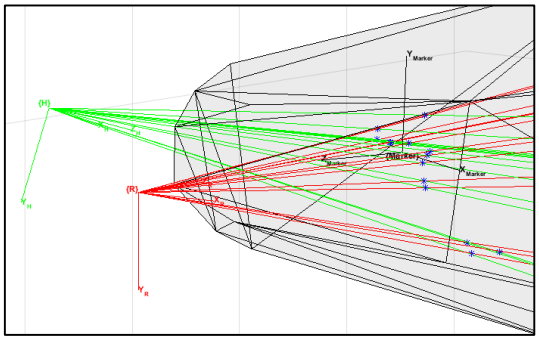

Fig 10. Feature position.

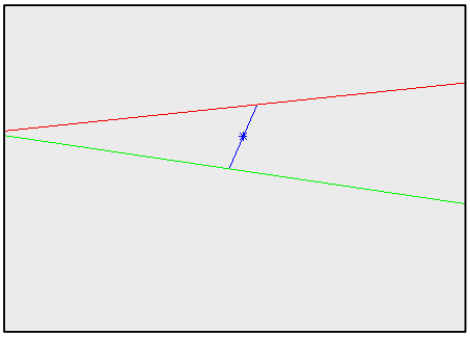

Fig 11. Rays and feature position.

4. Results

In the following figures we might see some of the results. For this first frame of a sequence, the algorithm returns the 3D position of 18 matches of a total of 26 that were located inside of the common region of the field of view of both cameras. This reduction is due to the threshold imposed in the cost index of the matching function.

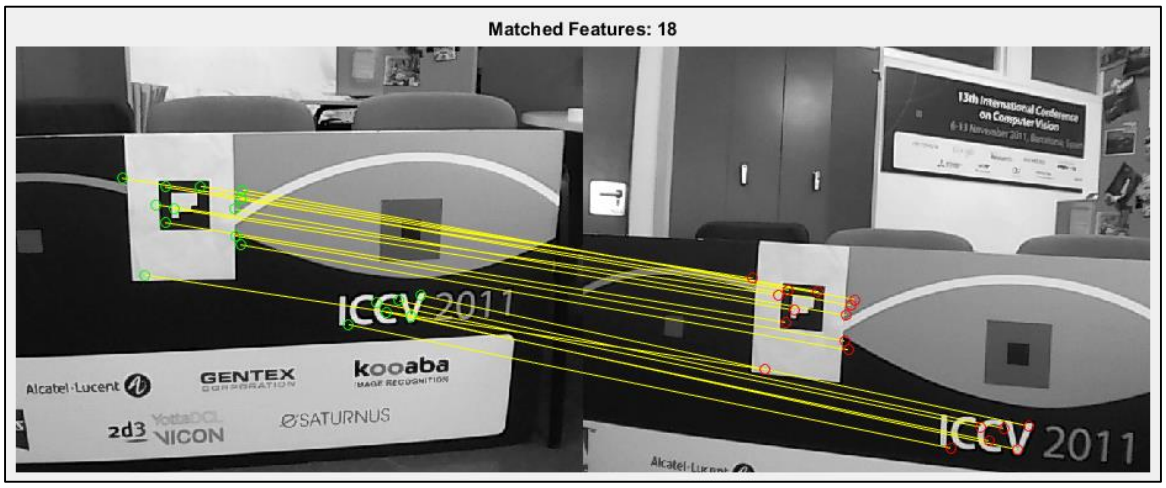

Fig 12. Matching.

Picking as a reference that all the matches should be in the same plane, the plane given by the X and Y axes of the AR marker, it can be seen in the 3D representation that the estimated feature position have a plane distribution in the supposed direction, they are inside the blue convex hull. The dark one represents the convex hull of the positions given by the monocular SLAM simulated list. The algorithm is able to match the features between the images that were estimated with a wrong depth by the SLAM algorithm, and recover the real position with a minimum error.

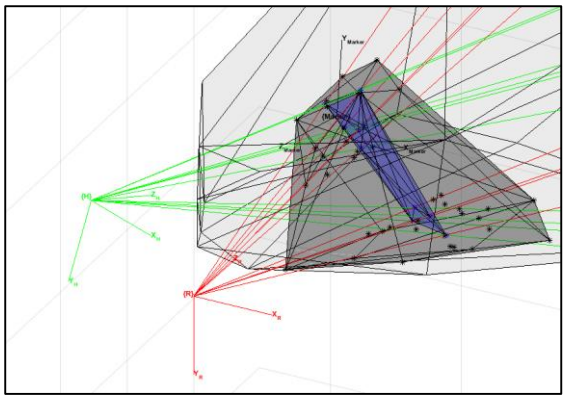

Fig 13. Representation of SLAM position estimation and corrected position estimation (View I).

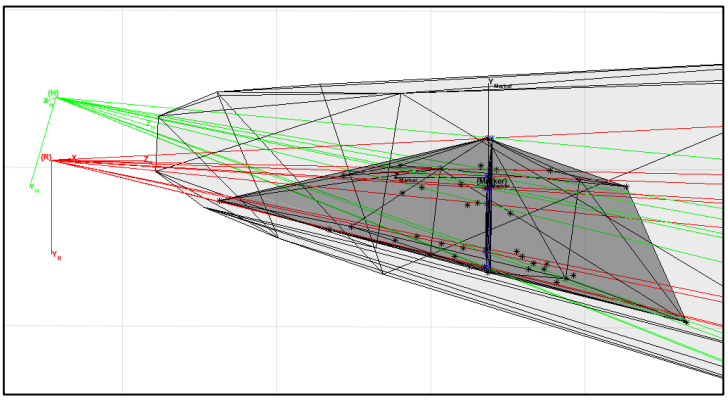

Fig 14. Representation of SLAM position estimation and corrected position estimation (View II).

In order to evaluate the performance along all the frames of a sequence the mean square error is computed from the obtained positions respect to their projection in the X and Y plane of the marker. This give us a measure of how precise is the method, and is useful in order to tune the parameters. The computing time per frame it's about 300ms.

There are two key points were we should focus in order to increase the number of matches, the first one is to have a longer list of features coming from the monocular SLAM algorithm, in this case were only 35 features. The second is the matching technique, there are several parameters to tune, size of patch, number of BRIEF pairs, distribution, Gaussian filter, precision for the search space discretization, and the cost threshold.

5. Conclusion

The implementation of the presented approach provides an intuitive way of representation and understanding of what is happening behind the feature matching process of two moving cameras. Which shape has the intersection of the fields of view and how the matching technique could take advantage of it to have a better performance.

The knowledge of the shape and pose of the intersection polytope give us a tool to compute the stereo matching only if the intersection exists, and only for the features that are located inside of it. The result of this conditions bring us a decrease in the computing effort and the crucial point of avoiding outliers, due to the fact that each feature has a particular search region bounded by its covariance and polytope.

Although the presented approach could be useful for a real Monocular SLAM application, there still a lot of work to do in order to increase the performance and efficiency of the implementation. The BRIEF method has shown that is a very good alternative to compute fast descriptors in order to match similar images, but may be not the best dealing with homography transformations since it is not rotational and scale invariant. Also, in order to improve performance, the approach should consider uncertainty not only in the depth but also in the azimuth an elevation and should return to the SLAM algorithm the uncertainties of the pose estimation.

6. References

[1] - G.Guerra, R.Munguia, A.Grau: Monocular SLAM for Autonomous Robots with Enhanced Features Initialization. Sensors (2014). 6317-6337.

[2] – P.Corke: Robotics, Vision & Control (2011).

[3] – M.Calonder, V.Lepetit, C.Strecha, P.Fua: BRIEF: Binary Robust Independent Elementary Features.

[4] – P.J.Schneider, D.H.Eberly: Geometric tools for computer graphics. 481-492.