Independently Moving Objects Detection based on Stereo Visual Odometry

Abaabu Amina
Instituto Superior Técnico, Lisbon, Portugal
ist179454@mail.ist.utl.pt

***Abstract*—In this project we propose an approach for detecting objects with independent movement using solely the images of a stereo rig assembled on a mobile robot. Our solution is based on a three-dimensional and sparse reconstruction of the environment, followed by the analysis of the velocities of the obseved points. By clustering proximal points with simlar velocities, we decide on the possibility of existence of independently moving objects in that instant. The algorithm works in dynamical settings, even if the system has the same velocity than some moving object. We leverage the visual odometry library LIBVISO with an additional analysis of the points velocities and robot egomotion. Experiments show that our approach has good results in the detection of independelty moving objects in a significant range of distances. Furthermore, we show applications of the proposed method to the case of tracking and following objects.**

Keywords: Stereo Vision, Features Matching, Visual Odometry, Bayes Classifier, Kalman Filter

I. Introduction

The detection of independently moving objects is an important issue in robotics, and it is relevant for performing certain actions, such as target tracking, definition of trajectories, and avoidance of obstacles which may interfere with the robot movement.
The large technological growth, in particular at computer's processing power, enabled the real-time analysis of high resolution images. Also, the use of camera systems has been increasing due to the various advantages they offer, such as their reduced cost and size. Recent works have demonstrated the ability to use computer vision in real time for visual odometry and egomotion estimation [1]. According to that work "visual odometry is the high accuracy compared to wheel speed sensors. Especially in slippery terrain". Building on this new technology, we propose the use of visual odometry as a means for the detection of independently moving object.
The method presented here is based on an analysis of the interest points of the image, thus texture of surrounding environment is critical for the algorithm's performance. We also consider the camera system to have its own motion. Thus, all the available information, including that related to background has also its own movement relative to the robot. This fact makes more difficult to detect and distinguish all information whose movement is integral with the robot motion, or in other words the static background.

A. Related Work

There are several approaches for moving objects detection that were developed in recent years. In some of them, the respective authors developed an object segmentation feature (e.g. [3] and [4]), while in the others the exact shape of the object is not so relevant (e.g. [5] and [6]). Even so, the most important difference between the algorithms is whether they allow to obtain the motion data of the moving object (e.g. [3]) or not.
Most of the approaches consider a static background and then they just proceed to separate it from the moving areas. An example of this kind of methodology is that presented by Zheng et al. at [5]. They obtain the absolute difference between two consecutive images, and then consider the existence of movement in a certain region if that difference is greater than some threshold. However this approach does not deal well with occlusion situations. Liu et al. [6] introduced a method for solving the occlusion problem.
In [3], Bramhe and Kulkarni present an approach that is very similar to the previous ones, when referring to the way that moving objects are detected. However they introduced new issues: vehicle counting and vehicle tracking, for Traffic Surveillance System, thus obtaining the speed and flow of each vehicle.
Also a very interesting method for simultaneous turbulence mitigation and moving object detection is addressed by Oreifej et al. in [7].
Contrary to all these methods, there is DECOLOR presented by Zhou et al. [4], that is a method which allows the cameras to move, when detecting moving objects. Despite coping with the moving cameras case, notice that this method does not estimate the egomotion of the cameras. This is a very interesting approach, although it cannot be applied for real-time cases.
Lastly, there is another approach that performs visual odometry based on stereo images, using RANSAC for outliers rejection [1]. It can work both monocular cameras and stereo-camera rigs. Due to this approach efficiency, this algorithm is used in the work presented here.

B. System Overview

We propose to solve the problem using LIBVISO, a Visual Odometry library. This library is used to estimate the movement of a stereo-camera rig. It expects to receive the calibration for both cameras, specifically the intrinsic as well as the extrinsic parameters. It also expects the supply of good quality images, in other words enough textured images. A strong point of this approach is that cameras calibration can vary over the time.

Firstly features of two consecutive stereo image pair are extracted and matched. This library doesn't require the tracking of image features over multiple frames; Features between consecutive stereo images are enough. After that, a technique named Bucketing is used. Then, in order to get rid of image feature outliers that derive from independently moving objects and false matches, the approach uses the RANSAC algorithm; Thus it allows working in dynamic environments. The resulting matches are the inliers whose movement is integral with the system motion.

Then the ego-motion of the system is computed using the trifocal tensor and the Kalman Filter. In fact, there is used an Iterated Sigma Point Kalman Filter which allows to deal with the non-linearities in the measurement equation.

While the previous step requires only the inliers of the process, the following one requires just the outliers, that result from points located at independently moving objects. So, at this point, we obtain the three-dimensional points that resulted in the outlier matched points, for both previous and current stereo image sequences. Here, the novelty is solving this problem using the visual odometry library. As this library outputs the motion between two consecutive instants of time, $T-1$ and T, we can estimate the position of a point P at time $T-1$ as whether it would move like the stereo camera rig. Then the velocity of this point is computed using its predicted and observed positions. We compare this velocity with a threshold which determines whether the point is really an outlier. We only consider a moving object, if there is a group of outliers close enough between them. Then a search algorithm seeks for all possible objects, for each of the detected velocities bins. This step guarantees that several moving objects can be detected, even if some of them have the same velocity.

Due to LIBVISO, this algorithm does not require the tracking of specific image features over multiple frames. We just require the feature matches between two consecutive stereo image pairs.

II. Geometry of the Cameras

A. Cameras Model

In this section we introduce the camera model that we use in this work. It is obtained by a combination of two methods: one to get camera intrinsic parameters and another to get the extrinsic ones.

1) Intrinsic Parameters: It is not practical to have the units of image plane frame in meters and its origin at the center of the image plane. So, a scaling and a translation are performed in order to get a more practical configuration, (x', y'):

$$\begin{cases} x' = S_x x + C_x \\ y' = S_y y + C_y \end{cases} \tag{1}$$

Factors S_x and S_y are used to convert units to pixels, instead of meters. Finally, constants C_x and C_y are used to take image frame origin to the desired position; Always to an image corner, normally to the upper-left one. All these operations are shown in Figure 1.

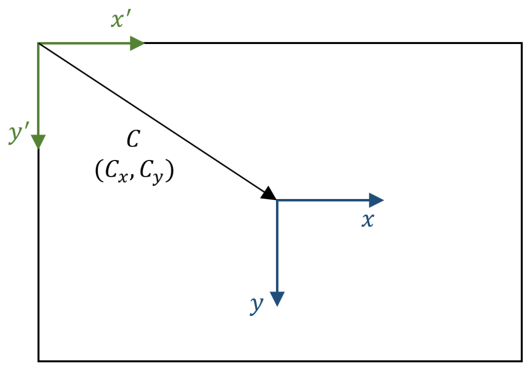


Fig. 1. Intrinsic Parameters Scheme

Combining the perspective projection equations and equations obtained in 1 we get,

$$\begin{cases} x' = S_x f \frac{p_x}{p_z} + C_x \\ y' = S_y f \frac{p_y}{p_z} + C_y \end{cases} \Leftrightarrow \tag{2}$$

$$\Leftrightarrow \begin{cases} p_z x' = f S_x p_x + C_x p_z \\ p_z y' = f S_y p_y + C_y p_z \end{cases} \tag{3}$$

In a matrix form,

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} fS_x & 0 & C_x \\ 0 & fS_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. \tag{4}$$

This 3x3 matrix is known as Intrinsic Parameters Matrix K; It is an upper triangular matrix. Equation 4 can be written in a condensed form, $\lambda\tilde{x} = K\tilde{p}$.

2) Extrinsic Parameters: The extrinsic parameters refer to the transformation from a world reference frame to the camera's frame. It is given by

$$p^c = Rp + t \tag{5}$$

$$p^c = [R|t]\tilde{p} \tag{6}$$

p^c is a point in camera frame
$\tilde{p}$ is the same point in world frame and in homogeneous coordinates.

R is a rotation matrix and t is a translation vector. Both are called extrinsic parameters or pose of the camera, and they define the transformation from world to camera's frame.

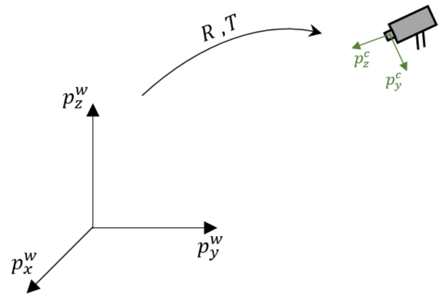


Fig. 2. Extrinsic Parameters

Now that it's possible to express the relationship between a point in the world reference frame and a point in the image camera plane, in pixels (equations 6 and 4), we can derive,

$$\lambda \tilde{x} = K p^c = K[R|t]\tilde{p} \tag{7}$$

Equivalently,

$$\lambda \tilde{x} = P\tilde{p} \quad , \; P = K[R|t] \tag{8}$$

Matrix P is known as Camera Matrix and its dimension is 3x4.

B. Relationship between the Stereo Image Sequences

Trifocal tensor is a function which relates features between three images of the same static scene. It has dimension 3x3x3. Considering three cameras, A, B, and C, whose camera matrices are given by

$$P_A = K_A[R_A|t_A]$$
$$P_B = K_B[R_B|t_B]$$
$$P_B = K_B[R_B|t_B],$$

the entries of the trifocal tensor are given by

$$T_i^{qr} = (-1)^{i+1} \cdot \det \begin{pmatrix} \sim a^i \\ b^q \\ c^r \end{pmatrix} \tag{9}$$

being $\sim a^i$ equal to matrix P_A without row i. b^q represents the q-th row of P_B and c^r is the r-th row of P_C.

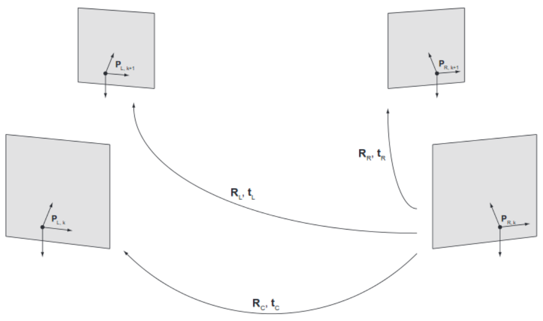


Fig. 3. Configuration of the stereo cameras at consecutive instants [1].

Figure 3 represents the use of the trifocal tensor and the relation between each of the reference frames of two pairs of stereo sequences of the same scene. In this case, the reference frame of the previous right camera at the instant of time k is the world frame. So the respective camera matrices are given by,

$$P_{R,k} = K_R[I|0] \tag{10}$$

$$P_{L,k} = K_L[R_C|T_C] \tag{11}$$

$$P_{R,k+1} = K_R[R_R|T_R] \tag{12}$$

$$P_{L,k+1} = K_L[R_L|T_L] \tag{13}$$

The trifocal tensors which relate previous stereo image pair to both current left (T_L) and right (T_R) cameras are respectively

$$T_L = T(K_R, K_L, R_C, t_C, R_L, t_L, \Delta t) \tag{14}$$

$$T_R = T(K_R, K_L, R_C, t_C, R_R, t_R, \Delta t) \tag{15}$$

And finally the non-linear mappings of the points in respective current image frames are,

$$x_{R,k+1} = h_R(T_R, x_{R,k}, x_{L,k}) \tag{16}$$

$$x_{L,k+1} = h_L(T_L, x_{R,k}, x_{L,k}) \tag{17}$$

III. Visual Odometry

In this section we explain better the concepts implemented in this library, and that were referred in I-B. It only remains to explain about Bucketing, RANSAC and Kalman Filter.

A. Bucketing

Bucketing consists in to divide the image into several non-overlapping rectangles, and in each one there is allowed a maximal number of features. Bucketing is represented in Figure 4.

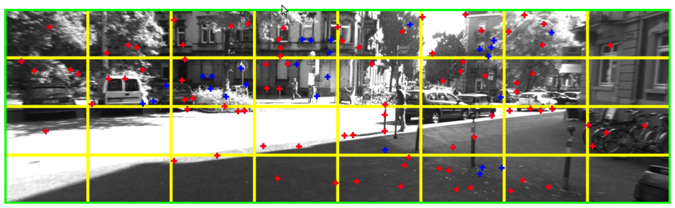


Fig. 4. Bucketing results. The yellow lines define each bucket. The red crosses in the image represent the selected stereo matches, and the blue ones represent the rejected features by means of bucketing [1].

This technique allows to reduce the computational time of the algorithm, and guarantees an uniform distribution of the features over the entire image, even along the depth axis. This last detail ensures that far as well as near features are influential to the result. Further, [1] refers that they observed that the drift rates of the approach are reduced when using bucketing.

B. RANSAC

Random Sample Consensus (RANSAC) is an iterative algorithm that is used for estimating parameters of a mathematical model. In the same way it can also be used for outliers detection. Here this algorithm is mainly used for rejecting feature matches present on moving objects. Firstly, it chooses subsets of matches and uses it to estimate the egomotion, through a kalman filter computation. Once that the egomotion is available, we can determine the inliers by comparing the Euclidean reprojection error of the matches with a certain threshold. These inliers form the consensus set of the sample. This process is repeated for a defined number of times, and then it is computed a final egomotion with all the inliers of the greater consensus set.

C. Kalman Filter

The Kalman Filter is used to estimate the current state of a dynamic system, which is assumed to be disturbed by zero-mean white noise. It is defined by two steps: a prediction and an update step. The measurements are assumed also to be disturbed by white noise with zero-mean [1].
Here the state y represents the ego-motion and is defined by

$$y = (V_X, V_Y, V_Z, \omega_X, \omega_Y, \omega_Z)^T. \tag{18}$$

The measurements are the feature coordinates in current images, and its relation to the state of the system are given by equations 16 and 17. Finally, the discrete-time space filter equations, which allow to estimate the current state y_{k+1} based on the previous one y_k are

$$y_{k+1} = f(y_k) + \omega_k \tag{19}$$

$$z_{k+1} = h(y_{k+1}) + v_{k+1} \tag{20}$$

where $h(.)$ refers to the same non-linear measurement equation presented by equations 16 and 17. $f(.)$ is the non-linear system equation and z_{k+1} is the measurement vector. The system noise is represented by $\omega_k \sim \mathcal{N}(0, Q_k)$ and the measurement noise is $v_{k+1} \sim \mathcal{N}(0, R_{k+1})$. $Q_k \in \mathbb{R}^{6\times6}$ and $R_{k+1} \in \mathbb{R}^{4N\times4N}$ are the respective covariance matrices. Finally, it is worth mentioning that R_{k+1} is a diagonal matrix and N is the number of features.
As previously referred, here it is used an Iterated Sigma Point Kalman Filter (ISPKF) in order to cope with the errors caused by the linearization around the current state. This operation is performed using a first order Taylor approximation. Furthermore the prediction step (Equation 19) is also linearized to $y_{k+1} = y_k + \omega_k$, under a constant velocity assumption.

IV. Moving Objects Detection

Once we have the feature matches and the egomotion, we proceed to detect the moving points that are close between them and have the same velocity, in order to be able to decide about the presence of moving objects.

A. Velocity Computation

As LIBVISO is only focused on inliers, we recover the outlier matches because of the moving objects detection purpose. Then considering (u_{left}, v_{left}) a point on image of left camera, and (u_{right}, v_{right}) its match on the image of the right camera, both at an instant of time T, and considering also the camera models for both cameras, we get that,

$$\lambda_{left} \begin{bmatrix} u_{left} \\ v_{left} \\ 1 \end{bmatrix} = P_{left} \begin{bmatrix} P_{T_x} \\ P_{T_y} \\ P_{T_z} \\ 1 \end{bmatrix} \tag{21}$$

$$\lambda_{right} \begin{bmatrix} u_{right} \\ v_{right} \\ 1 \end{bmatrix} = P_{right} \begin{bmatrix} P_{T_x} \\ P_{T_y} \\ P_{T_z} \\ 1 \end{bmatrix}. \tag{22}$$

$P_T(P_{T_x}, P_{T_y}, P_{T_z})$ is the 3D point which resulted in the matched points.

With manipulation of these equations (21 and 22) we can obtain P_T. The matches of the same point the in the previous stereo image pair are also available. So, similar computations can be done for these points, at instant of time $T-1$, thus obtaining P_{T-1}. Physically these two 3D points are the same, although they can have different coordinates depending on the reference frames involved. Taking advantage of this, we are interested on expressing these points in a reference frame on the robot. At this point takes place the novelty of this approach, that consists on solving the problem using this visual odometry library. As this library outputs the motion T_r between instants of time $T-1$ and T, we can estimate the position of P_{T-1} as whether it would move like the stereo camera rig. That estimation $\hat{P_T}$ is given by,

$$\hat{P_T} = T_r P_{T-1} \tag{23}$$

Now there are achieved all conditions to compute velocity V of the point P. So,

$$V = \frac{\|P_T - \hat{P_T}\|}{\Delta t} \tag{24}$$

$$\Delta t = \tau_T - \tau_{T-1} \tag{25}$$

τ_N represents the timestamp of image acquisition in instant N, and Δt is the interval of time between the picking of both stereo image pairs.

B. Bayes Classification

Just notice that not all the features with velocity greater than zero, should be considered moving outliers. So, we must draw some type of decision rule for classifying whether a point is an inlier or outlier, based on its velocity. A threshold must be imposed and only features with velocity greater than it should be considered. Among the several techniques for this purpose, there are the Bayes classifiers which are based on the Bayes Theorem: Considering two random variables X and Y,

Auxiliary Concepts : Conditional Probability
their joint probability is given by

$$P(X \cap Y) = P(X|Y) \cdot P(Y) \tag{26}$$

or equivalently

$$P(X \cap Y) = P(Y|X) \cdot P(X) \tag{27}$$

and manipulating equations 26 and 27, we can get

$$P(X|Y) \cdot P(Y) = P(Y|X) \cdot P(X) \tag{28}$$

and therefore,

$$P(X|Y) = \frac{P(Y|X) \cdot P(X)}{P(Y)}. \tag{29}$$

Equation 29 represents what is known as Bayes Theorem. It can be used for classification of data, given some evidence: Considering a variable C that contains two classes C_1 and C_2, and also considering X as a(n) feature/evidence of C, by application of the Bayes Theorem we can get that

$$P(C_1|X) = \frac{P(X|C_1) \cdot P(C_1)}{P(X)}. \tag{30}$$

$P(C_1|X)$ is known as the posterior probability, while $P(C_1)$ is the prior one. $P(X|C_1)$ is the likelihood of the evidence for C_1 class and $P(X)$ refers to the evidence's probability.
A similar equation can be got for class C_2. For this kind of applications, the likelihood of the evidence refers always to the approximation of the data to a probability density function, such as the Gaussian or the Rayleigh distributions. Then the prior probability is known as the probability of a point to belong to that class. Lastly, the evidence probability is got after marginalization of the class variable, thus it can be also written as

$$P(X) = \sum_{C=\{C_1,C_2\}} P(X|C) \cdot P(C). \tag{31}$$

At this time, we can apply the Decision rule for classification of the entity. This rule says that if $P(C_1|X) > P(C_2|X)$ then the entity must be assigned to class C_1. Otherwise, the entity is assigned to C_2. Notice that the point where these two probabilities are the same, represent the threshold for X which separates the two classes. The experimental results for obtaining this parameter are presented in Section V-A.

C. Grouping and Rectangular Fitting

Due to the errors involved in the process, the point velocities won't be exact; So we preceded it and we did a needful discretization of the velocities. Once again the matching errors and noise take a role, and some undesired outliers will appear, hence only a group of outliers should be considered a moving object, reducing the probability of appearing false estimates. The threshold for the minimum number of outliers for a group was empirically determined. Then there must have a maximum distance between the points, because it is quite important to avoid errors and determine more precisely the object location. Again we found out a threshold for this distance empirically. As the used approach takes into account the possibility of existing several moving objects, even some of them with the same velocity, a search algorithm for the points with the same velocity was developed.
Now there are all the necessary conditions to group the outliers if they have the same velocity and if they are close enough between them. So, they are identified in the image with one rectangle around them. The four vertices of this boundary are defined by the minimum and maximum values of both x and y coordinates of the group points. The position of the object is defined by the centroid of these points and its motion is the mean motion of the same points.

V. Experimental Results

This section presents some results that we obtained when testing this work's algorithm. For these experiments we used as the stereo vehicle two Microsoft Kinect cameras placed above a Pioneer P3-DX robot. For representing the independently moving object we used another Pioneer robot with a textured box placed above it, in order to get more features. Notice that these results refer to Simulation data. Figure 12, at last page, shows this environment. The same tests were done in real environment, but with no data analysis due to lack of equipment. So to know about them, consult the thesis document, Chapter Results.

A. Statistical Modelling

Firstly we present the results for obtaining the threshold for outliers classification based on their velocities, as specified in Section IV-B. So, we started by extracting 450 samples for random inliers velocities and the same number for outliers velocities. Then we needed to fit these data to Rayleigh distributions, since we are using vector magnitudes.

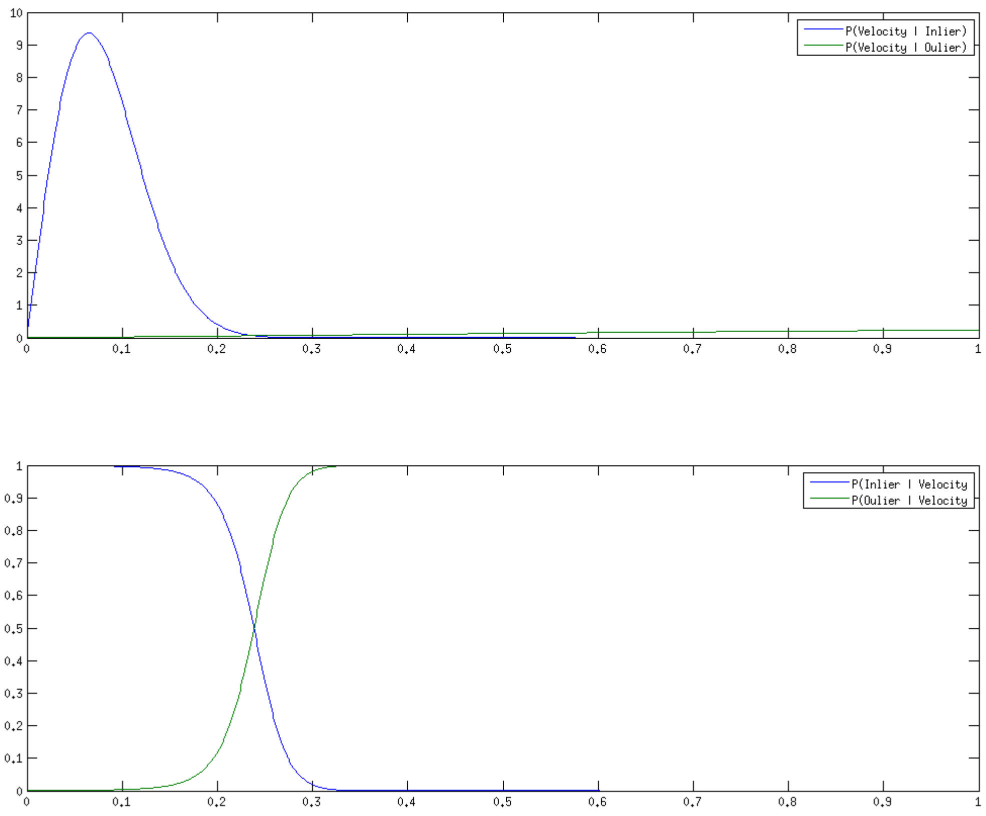


Fig. 5. Probability density functions.

The Rayleigh probability density function is given by

$$R(x;\sigma_R) = \frac{x}{\sigma_R^2} e^{\frac{-x^2}{2\sigma_R^2}} \tag{32}$$

$$\sigma_R = \sqrt{\frac{1}{2n}\sum_{i=1}^{n} x_i^2}, \tag{33}$$

where x corresponds to the data vector and n to its size.σ_R is the scale parameter of the distribution.

Using our collected data, we get $\sigma_R = 0.0648$ for the inliers and $\sigma_R = 1.9003$ for the outliers. The resulting distributions are the Velocity Probabilities given its class, $P(\text{Velocity}|\text{inlier})$ and $P(\text{Velocity}|\text{outlier})$. They are presented on the upper image of Figure 5.

At this point we have met all the conditions for applying the Bayes Rule, to get the probability functions which allow to decide on the separating data threshold. So we show the respective results on the image at Bottom of Figure 5. Analysing these results, we know with a confidence of $50,18$ that a point with velocity greater than $0,2385$ is an outlier. And with a slight further analysis we can increase more the confidence of classification to $99,0$; And for that the threshold becomes equal to $\underline{0,3090}$.

B. *Moving Object Detection*

We tested the algorithm in the detection of a moving target with several kind of movements, but here we are just presenting two cases; We show these results for stopped cameras rig and then for it moving, trying to show that the detection process is independent of the cameras velocity.

1) Stopped Cameras and Target with a forward motion and constant acceleration: We introduced in the target's motion a constant acceleration equal to $0,015\text{m/s}^2$ and an initial velocity equal to $0,4$. Figures 6, 7 and 8 demonstrate how the system behaved.

As this method depends on the detected features, the system can detect any region of the target's surface. So we plotted the magenta lines in the $x(t)$ graph of Figure 6 in order to have an idea about the target limits. So observing the same graph, we can see that the estimation of x occurred successfully with no errors. However for $y(t)$ there are more errors as the depth increases, what was expected due to the increasing of the camera errors.

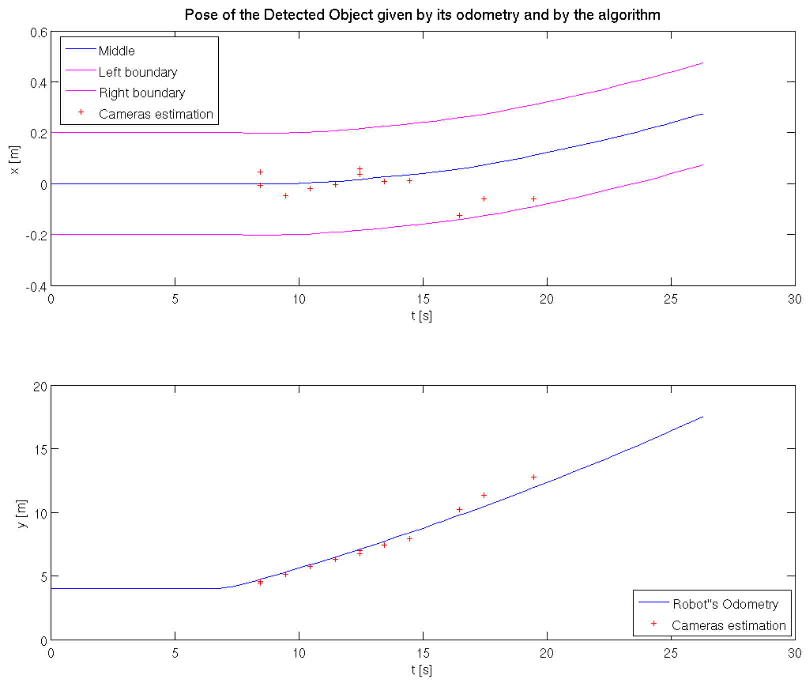


Fig. 6. Pose of the target in an external fixed reference frame. The x axis represent the horizontal coordinate while the y refers to the depth axis. The two magenta lines represent the vertical boundaries of the target.

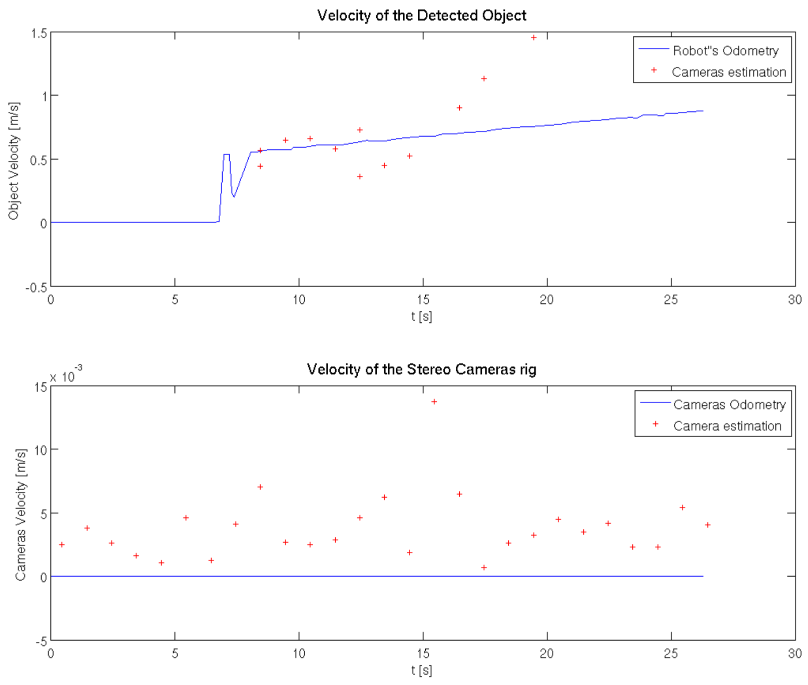


Fig. 7. Estimation of the velocity of the target on top image, and estimation of the stereo rig's velocity on bottom.

By Figure 7 we can say that the cameras velocity was relatively well estimated, what was expected because the cameras are stopped; The major errors start to occur when the target moves, because it is still close to the cameras and some points located on it can introduce errors in the egomotion estimation. And finally, the object velocity's graph represents the previously referred effect of the errors increasing with the depth. We can also observe that the major errors on the target pose estimation start to appear after the 9, approximately. That's an acceptable distance, and considering also the errors presented on Figure 8, we can consider that the results are great.

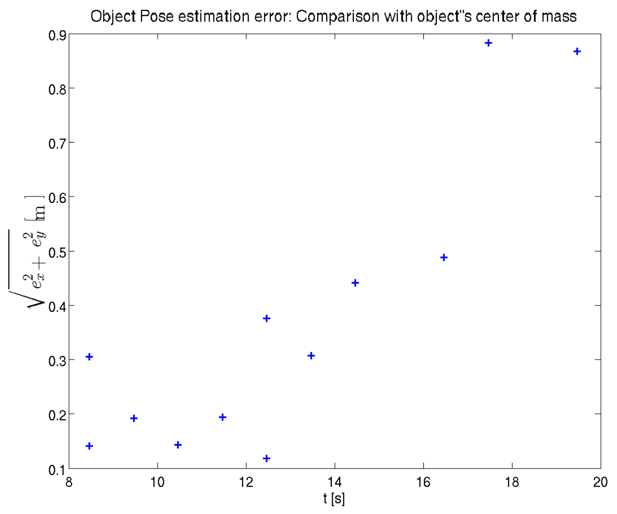


Fig. 8. Error for estimation of the target's position.

2) Moving Cameras and Target with a forward motion and constant acceleration: In this case we gave the target the same motion as the previous case. But here the difference is that the cameras rig is moving with $0,4$ in forward motion. Obviously, we want also to prove that even if the system is moving, it can detect well the target. We performed the test and we depict the results at the three next figures.

Through these figures, we can confirm that the depth error decreased, since the depth of the target is just increasing due to the acceleration; The cameras are moving. We can also see by $y(t)$ (Figure 9) that the camera estimation points are well fitted to the odometry curve. By Figure 10 we observe that even both the velocities are accurately estimated. This experiment shows that the method presented here can produce very accurate results.

However, Figure 11 is denying what we said about the accuracy of the results and shows that for this case the error is still considerable. But, let us to reinforce that these error plots are comparing to the centroid of the target, what eases these differences.

VI. Conclusions

We observed that the texture of the surrounding environment is the critical point of this approach; There should have a well-textured environment with features well-distributed. This occurs by the fact that this method is based on image features. Then, the object's texture is also important because it controls the range of depth in which the target can be detected. The system behaved as expected on detecting properly the object in any kind of motion. Nevertheless, we observed that the higher the object's velocity, the harder is the detection. The baseline of the stereo rig determine how much of the images are overlapped, and thus it is also a very important parameter. In this work, it is equal to $0,375$. We observed that the higher the baseline, the higher is the the minimum depth that the system can detect the object. For this simulation environment and this target object, the minimum distance observed is $0,866$, while the maximum is $13,537$.

Then, it is important to state that the mean system processing time is approximately $0,41$ for 800x600 images, allowing a good processing rate for real time applications. However, in these experiments we used cameras publishing the images at 1, as a way of reducing the effect of the noise.

For concluding, we think that the system can be applied to many applications, due to the results it can provide.

A. Future Work

We observed that the farther points have more reconstruction error, as expected, but we thought that this can be prevented. Thus a characterization of the camera error, in terms of depth, would improve significantly this work, allowing even the object detection for farther distances.

Lastly, for taking this work to an higher level, some type of object recognition can be implemented. In this way, the system would only track some specific object or some kind of object. We were thinking about applying the Bag of Features algorithm. In that way, for example the system could be applied for checking cars velocity, like a radar.

References

[1] B. Kitt and A. Geiger and H. Lategahn, *Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme*. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 486–492. IEEE, 2010.

[2] Y. Feng and R. Liu and B. Jia and M. Zhu, *An efficient pixel-wise method for moving object detection in complex scenes*. In *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*, pages 389–394. IEEE, 2013.

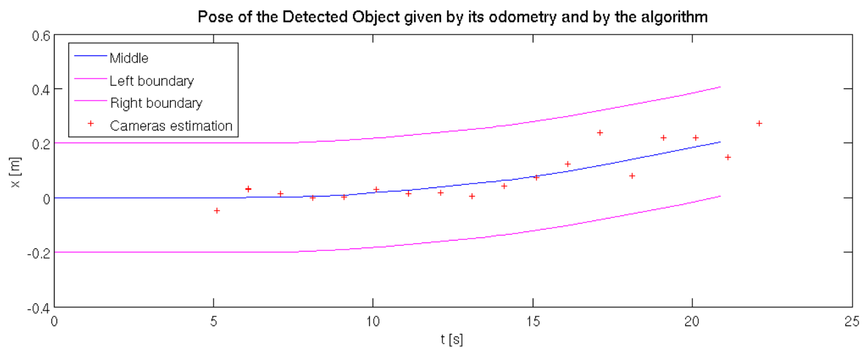
[3] Y.B. Brahme and P.S. Kulkarni, *An implementation of moving object detection, tracking and counting objects for traffic surveillance system*. In *omputational Intelligence and Communication Networks (CICN), 2011 International Conference on*, pages 143–148. IEEE, 2011.

[4] X. Zhou and C. Yang and W. Yu, *Moving object detection by detecting contiguous outliers in the low-rank representation*. In *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(3): pages 597–610, 2013.

[5] X. Zheng and Y. Zhao and N. Li and H. Wu, *An automatic moving object detection algorithm for video surveillance applications*. In *Embedded Software and Systems, 2009. ICESS'09. International Conference on*, pages 541–543. IEEE, 2009.

[6] D. Liu and M.-L. Shyu and Q. Zhu and S.-C. Chen, *Moving object detection under object occlusion situations in video sequences*. In *Multimedia (ISM), 2011 IEEE International Symposium on*, pages 271–278. IEEE, 2011.

[7] O. Oreifej and X. Li and M. Shah, *Simultaneous video stabilization and moving object detection in turbulence*. In *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(2): pages 450–462, 2013.



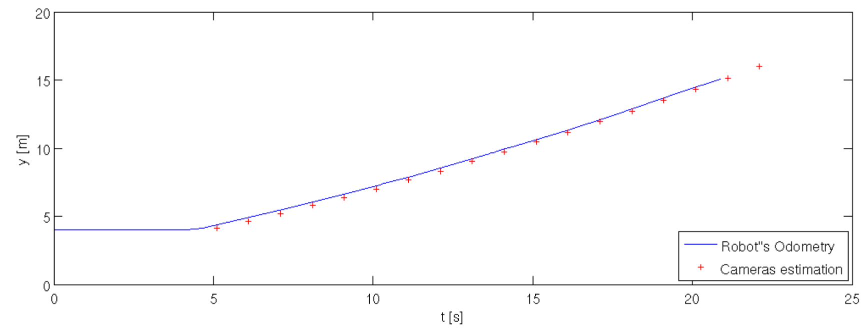
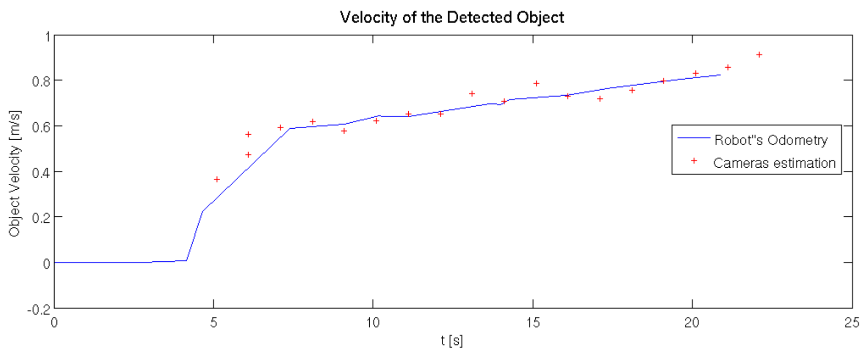


Fig. 9. Pose of the target in an external fixed reference frame.



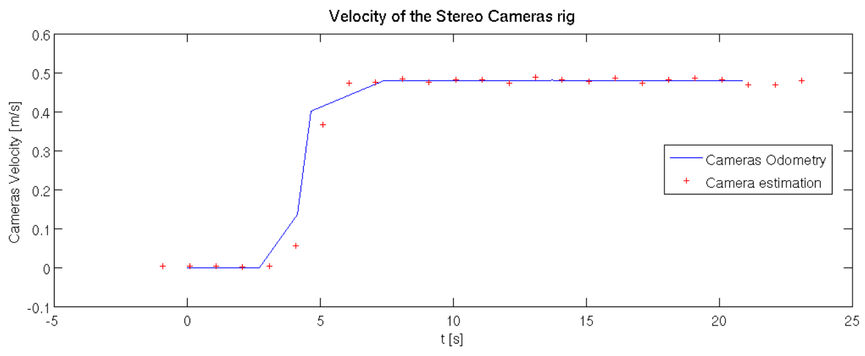


Fig. 10. Estimation of the velocity of the target on top image, and estimation of the stereo rig's velocity on bottom.

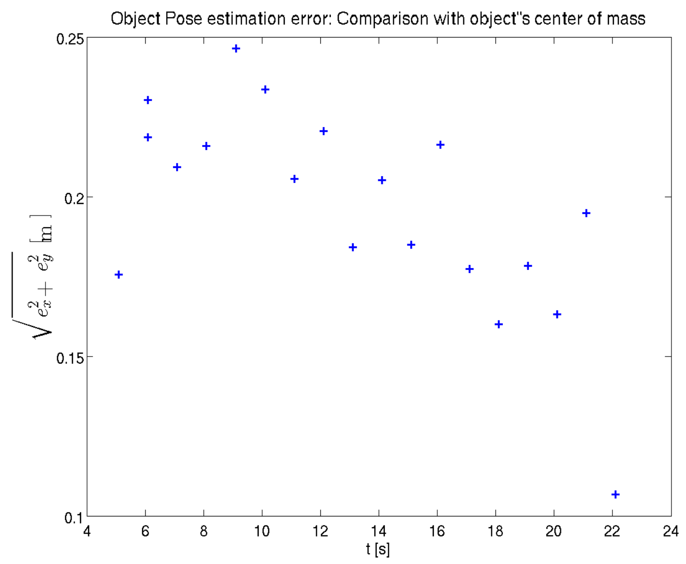


Fig. 11. Error for estimation of the target's position.

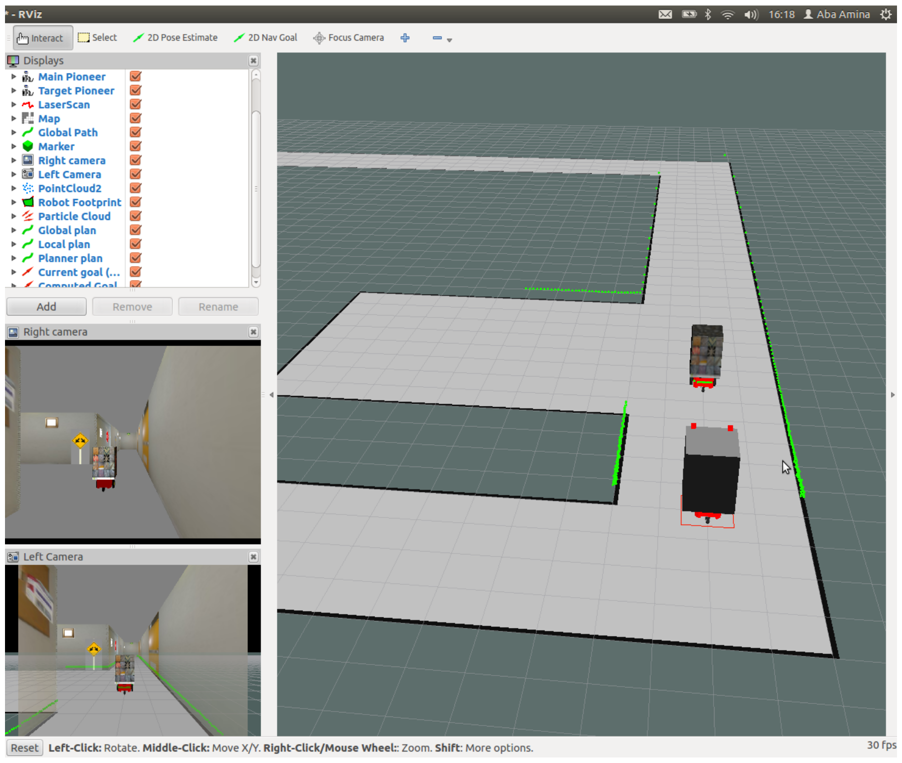


Fig. 12. Simulation environment. This image refers to RViz environment which allows visualization of the simulation world. Those red boxes on top of the first robot are the cameras, and the other robot is the moving target. At bottom left corner, there are the resulting images from the cameras. Notice that we placed in the walls some objects, in order to increase the surrounding texture.