Multitasking of Smart Cameras

T. Castanheira, A. Bernardino, J. Gaspar
Instituto Superior Técnico / UTL, Lisbon, Portugal
tfacastanheira@gmail.com, alex@isr.ist.utl.pt, jag@isr.ist.utl.pt

ABSTRACT

In this dissertation we propose an automatic system for the visual-tracking of buses in a parking lot, based in a set of Pan-Tilt-Zoom (PTZ) cameras. The parking lot has specific entry points equipped with bus passing detectors which inform our system of new buses. Bus detection is based in background subtraction. In a first step the result of background subtraction is used to obtain an estimate of the location of the center of the bus, given the intrinsic and extrinsic parameters of the cameras relative to the ground plane. In a second step, the estimated location is refined with an optimization algorithm, more precisely a box fitting methodology, which allow in addition to estimate the bus orientation. The refined estimation of the bus pose is finally filtered by an Extended Kalman Filter (EKF). The EKF contains a kinematic model of a bus. Given the multiple Extended Kalman Filters, one for each of the buses, we design a resource (PTZ camera) allocation algorithm in a multitasking (multiple buses) environment, giving priorities to the tasks based in the estimated uncertainties of the poses of the buses. Our system was developed and tested in simulation. The simulator encompasses a scenario and objects (buses and cameras) described in Virtual Reality Modeling Language (VRML), which are controlled using the Virtual Reality toolbox of Matlab.

Keywords: Multitasking, car kinematic model, PTZ cameras, detection, tracking, Extended Kalman Filter.

I. INTRODUCTION

Task automation has been increasing in a variety of industries and services, due to its capability to improve efficiency without compromising the quality of the work. In particular this work is focused on the automation of a visual tracking system composed only by calibrated PTZ (pan, tilt and zoom) cameras that process video to locate and track buses in a parking lot.

Multitasking is considered in our work as a way to allow the tracking of multiple buses with a number of cameras that is lesser than the number of buses. To make it work we propose associating to every bus an Extended Kalman Filter (EKF) that is constantly predicting the poses of the buses, even when the cameras are not feeding the EKF with observations. These predictions are based on the motion model of the buses. With the estimations of the poses of the buses we have associated covariance matrices. The poses with higher uncertainties are expected to have cameras assigned as soon as possible in order to keep the summation of uncertainties as small as possible.

To test this system we made a simulator which is a virtual world in 3D with virtual buses that respect their real kinematic motion model and virtual PTZ cameras.

A. Related Work

In [6], the authors list various geometric methodologies for camera calibration. Comparing with the use of a planar chess pattern, as proposed by J. Y. Bouguet [1], in [6] are explored alternatives based on using just image features or using image features combined with the camera odometry. In our work the emphasis is placed in controlling the cameras, and therefore we use a simpler calibration methodology based in knowing 3D scene points and their images.

In [9] a number of motion modalities for one pan-tilt camera are assessed with respect to omni-awareness or, more precisely, maximizing the percentage of events found. Starzyk and Qureshi [7] consider multiple PTZ cameras to track pedestrians (moving events). They design a behavior based architecture which handover tracking inter-cameras and maximize the zoom of each camera while not loosing tracking of all pedestrians. In our work the events to find and track are moving buses. Contrarily to the motion of people, the motion of buses can be predicted. In this work we take the predictions of the motion into account in the design of PTZ control.

B. Problem Formulation

Our motivation for this work is having a cost effective automated system to track and detect buses, in a large bus parking lot.

The proposed solution for this problem is based in PTZ cameras that communicate with each other to detect and track all the buses that enter the park and find out their parking location. The goal was to minimize the number of cameras for cost reasons. To do that we need to have a multitasking algorithm so that the cameras used can divide tasks in order not to loose any target and have a filter to predict their location when needed. This filter will be an EKF with the car motion model, since the buses are longer cars.

In this work we take the predictions of the motion into account in the design of the PTZ control.

C. Paper Structure

Section 1 introduces the problem to approach in the thesis, in particular presents a short discussion on the state of the art on the subject. Section 2 presents the PTZ camera model and its direct and inverse kinematics. Section 3 describes the bus motion model, detection and tracking algorithms.

Section 4 details our complete autonomous system. Section 5 presents the simulator results. Section 6 summarizes the work performed and proposes further work to extend the activities described in this document.

II. Camera Model and Calibration

In this chapter we will describe the camera model which is the pin-hole model and explain the back projection. Our work is based on calibrated cameras and thus we need to introduce calibration methods such as the DLT (Direct Linear Transformation) to be used in a real world. We will finish by detailing the direct and inverse kinematics of the pan-tilt-zoom camera.

A. Camera Model

Since one of our goals is to locate the buses and we only have pixels as input, we need to find a model to translate those pixels into world coordinates so that for every $[u\ v]$, with u representing the pixel column and v its line, we have an $[X\ Y\ Z]$ world position. The model we chose is the pin-hole camera model, described in the next section.

1) Pin-hole Camera Model: The pin-hole camera model, which is represented in Fig.1, is the model used in this work to represent the relashionship between world and image coordinates [4]. In red we represent the image plane which is in 2D and in blue the world which is in 3D. In capital letters are represented 3D world variables while the lower-case letters represent 2D image variables.

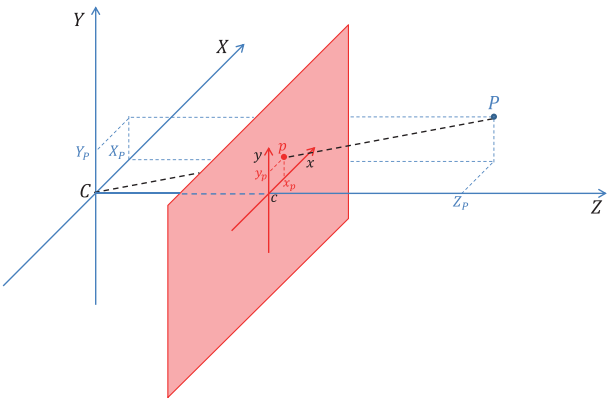


Fig. 1. Pin-hole model in 3D

The variable p is a point in the image plane and its coordinates are $p = [x_p\ y_p]$ while P is a point in space whose coordinates are $P = [X_P\ Y_P\ Z_P]$. C represents the optical center, which is the origin of the world referential. This coordinate is very important because the mapping of P on p is made by a line that passes through both C and P and is represented by the dashed black line in Fig.1. p is the point resulting in the intersection of that line and the plane image. c is called the principal point and is the central coordinate of the image. The variable Z is the principal axis because it represents the line that intersects the optical center and the principal point.

We also must add f which is called the focal length and is the distance between the optical center and the principal point. Since the relation between 2D and 3D points is made with triangle similarities we can conclude that:

$$\begin{cases} x = f\frac{X}{Z} \\ y = f\frac{Y}{Z} \end{cases} \tag{1}$$

To go from meter to pixels coordinates we will multiply them by k_x and k_y, which represent the number of pixels per meter in both directions and add an offset that we will call u_0 and v_0 and are the pixel coordinates of the principal point c. These changes will result in $u = x \cdot k_x + u_0$ and $v = y \cdot k_y + v_0$. Now if we make $s_u = f \cdot k_x$ and $s_v = f \cdot k_y$ and change to matrix notation we end up with:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_u & 0 & u_0 \\ 0 & s_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

where λ is a constant related with depth that must be add to compensate the difference in the number of variables in both vectors and the matrix is called K and it is the intrinsic parameters matrix. To transform world coordinates into camera coordinates we need to add a rotation matrix R and a translation vector t and we end up with:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3}$$

In conclusion, the relation between a coordinate in the world M and the pixel m in the image plane of a camera is given by Π, a 3×4 matrix called projection matrix, up to a scale factor λ:

$$\lambda m = \Pi M \tag{4}$$

2) Back Projection: With Eq.4, given the projection matrix we can obtain the image coordinates in pixels that correspond to any point in the world. The inverse is not possible with this equation because it would need a Π^{-1} which does not exist since Π is not a square matrix. It is also impossible to obtain three unknowns ($[X\ Y\ Z]$) with only two inputs ($[u\ v]$).

To solve this problem we need to define all 3D points as points belonging to a line that goes through the optical center and has the optical ray direction. That is given by $M = C + \alpha \cdot D$ where D is a point in infinity and α is a 1×1 scaling factor. Since our parking lot is in the plane $Z = 0$ we can get α and solve the X and Y coordinates that we wanted to get.

B. Calibration

The main objective of calibrating a camera is calculating the projection matrix Π since it holds all the information about the relation between the m and the M vectors. This is important so that when the image signal is generated by the camera, the world coordinate of each pixel can be easily computed. This is called back projection and was written with detail in Sec.II-A2

The DLT (Direct Linear Transformation) is the method used. Only the geometric calibration will be considered.

The Non-linear Least Squares Method is the form of least squares analysis which is used to fit a set of m observations with a model that is non-linear in n unknown parameters, with $m > n$. In this case, the function to be minimized is:

$$E = \sum_{i=1}^{N} \|m_i - f(M_i, \Pi)\|^2 \tag{5}$$

with Π as the unknown parameter. m_i and M_i are corresponding coordinates and N must be bigger than 5 although the bigger the better because the measurements usually have noise and with more data the accuracy will also be better.

In order to minimize the error energy we need to calculate it first. With the Eq.4 and by making Π_n, the nth line of matrix Π we reach the following equation:

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} \Pi_1 M \\ \Pi_2 M \\ \Pi_3 M \end{bmatrix} \Leftrightarrow \begin{cases} \lambda = \Pi_3 \cdot M \\ u \cdot \Pi_3 \cdot M = \Pi_1 \cdot M \\ v \cdot \Pi_3 \cdot M = \Pi_2 \cdot M \end{cases} \tag{6}$$

The error energy e that is defined as the distance between the observed image coordinates and the image coordinates computed from the world coordinates with matrix Π, which we want it to be zero, is given by:

$$e = \underbrace{\begin{bmatrix} M_1^T & 0^T & -u_1 \cdot M_1^T \\ 0^T & M_1^T & -v_1 \cdot M_1^T \\ \vdots & \vdots & \vdots \\ M_n^T & 0^T & -u_n \cdot M_n^T \\ 0^T & M_n^T & -v_n \cdot M_n^T \end{bmatrix}}_{2N \times 12} \cdot \underbrace{\begin{bmatrix} \Pi_1^T \\ \Pi_2^T \\ \Pi_3^T \end{bmatrix}}_{12 \times 1} \tag{7}$$

with 0^T being a 1×4 line with zeros and N the number of points. We can see that the system has eleven unknowns (nine Π coordinates plus two for vector e) which will require at least six pairs of m/M points, as stated above. This equation can be simplified as $e = A \cdot p$. Since the error energy is calculated as $E = \|e\|^2 = e^T e$ we will have the error energy function as:

$$E = p^T \cdot A^T \cdot A \cdot p \tag{8}$$

The Lagrangian function will then be calculated knowing the constraint $p^T \cdot p = 1$, which means:

$$L = p^T \cdot A^T \cdot A \cdot p - \lambda(p^T \cdot p - 1) \tag{9}$$

and by deriving in respect to p and making it equal to zero we reach the final equation system:

$$\begin{cases} A^T \cdot A \cdot p = \lambda \cdot p \\ p^T \cdot p = 1 \end{cases} \tag{10}$$

The solution for p will be the eigenvector associated with the smallest eigenvalue of $A^T \cdot A$ since when replaced in Eq.8 returns the minimum error energy.

C. Pan-Tilt-Zoom Camera

In our work we have multiple cameras fixed on the top of a post which can pan, tilt and zoom.

For tracking purposes it is important to study the relation between the pan and tilt angles and the world coordinates so that, given a world coordinate, we can pan and tilt the camera in order make its back projection into the principal point [2].

The reference frame chosen was the one in Fig.3.

1) Direct Kinematics: With the direct kinematics we will be able to tell which is the world coordinate of the point in the center of the camera, given by the pan angle α and the tilt angle β. In order to do that we need to calculate a matrix that will give a transformation from the camera to the world coordinates. That matrix function is ${}^{W}T_{C}(\alpha, \beta)$.

Instead of computing ${}^{W}T_{C}$ we will compute ${}^{C}T_{W}$ which is its inverse (${}^{W}T_{C} = {}^{C}T_{W}^{-1}$). ${}^{C}T_{W}$ is defined as the transformation from world to camera coordinates. To compute this transformation we need to make a translation and a rotation. If the South camera (Fig.12) is at position $[X_B \ Y_B \ Z_B]$ we will have:

$${}^{C}T_{W} = {}^{C}T_{B} \cdot {}^{B}T_{W} = \begin{bmatrix} 1 & 0 & 0 & X_B \\ 0 & 0 & 1 & Z_B \\ 0 & -1 & 0 & -Y_B \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

These transformations are static and are represented in Fig.2

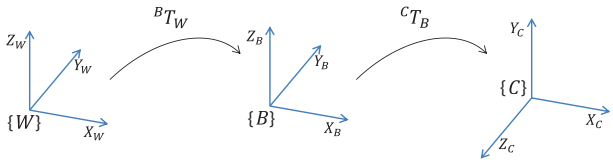


Fig. 2. Transformation from world to base and then to camera coordinates

Now, to calculate the transformation from the static camera to the camera after pan and tilt reference, ${}^{C}T_{Cpt}$, we need to take into account that the reference frames are mobile, they depend α and β shown in Fig.3.

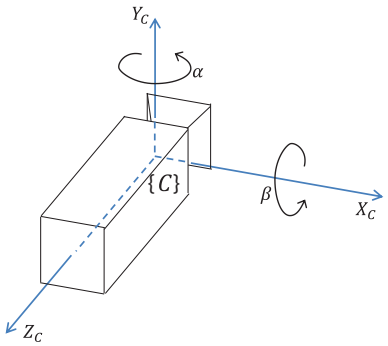


Fig. 3. Representation of pan (α) and tilt (β) angles

There are three transformations that take place. The first transformation is ${}^{pan}T_{C}$ which represents the pan rotation and is given by the equation:

$$^{pan}T_C = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12)$$

The second transformation is $^{tilt}T_{pan}$ which represents the tilt rotation and is given by:

$$^{tilt}T_{pan} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) & 0 \\ 0 & \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

The third and last transformation is $^{Cpt}T_{tilt}$ which is an identity matrix. With these three transformations we can now calculate $^{Cpt}T_C = {}^{Cpt}T_{tilt} \cdot {}^{tilt}T_{pan} \cdot {}^{pan}T_C$ which is represented in Fig.4.

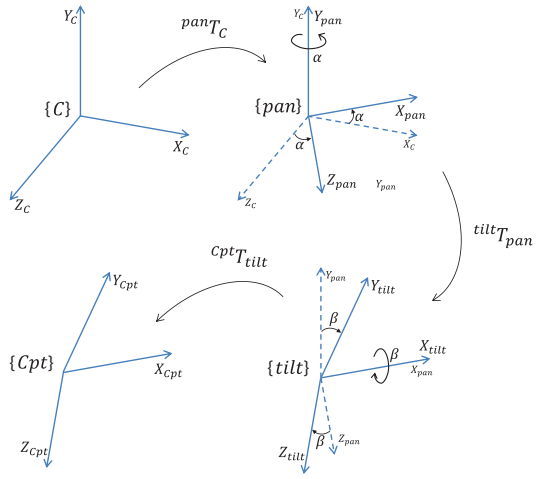


Fig. 4. Transformation from camera to camera after pan and tilt rotation

In the end, we have the transformation from the world to the camera coordinates after pan and tilt given by $^{Cpt}T_W = {}^{Cpt}T_C(\alpha, \beta) \cdot {}^{C}T_W$

2) Inverse Kinematics: The inverse kinematics give us which pan and tilt angles should we have, in order to point the center of the camera to the desired world coordinate.

Starting with camera coordinates before pan and tilt ($^{C}M = {}^{C}T_W \cdot {}^{W}M$) the pan and tilt angles are given by a trigonometric formula that has the following solution:

$$\begin{cases} \alpha = \arctan(\frac{^{C}M_X}{^{C}M_Z}) \\ \beta = \arctan(\frac{^{C}M_Y}{\sqrt{^{C}M_X^2 + {}^{C}M_Z^2}}) \end{cases} \quad (14)$$

3) Zoom: Zoom is a mechanical assembly of lens for which the angle of view (or field of view) can be varied. The field of view, fov, is the opening angle of the camera lens and is represented in Fig.5 where rd is the circle radius.

We can see that the angle between $|^{C}M|$ and rd is right, which means the fov is given by:

$$fov = 2\arctan(\frac{rd}{|^{C}M|}) \quad (15)$$

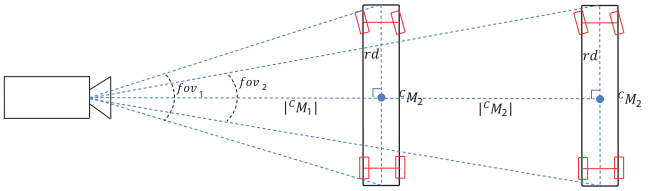


Fig. 5. Representation of the field of view

With Eq.15 instead of manipulating the field of view directly we manipulate the radius. If we want to track a bus and have it with a fixed size in our image, since we know its position we can manipulate the field of view by keeping the radius constant. As we can see in Fig.5, fov_2 is smaller than fov_1 because the distance from the camera to the bus $|^{C}M_2|$ is larger than $|^{C}M_1|$.

III. Target Detection and Tracking

In this chapter we will talk about the bus motion model. This motion model is crucial for tracking algorithms like the Extended Kalman Filter. The EKF, since is the one used in the simulator, is detailed in three subsections: EKF Predict, EKF Update, and Error Ellipse. After the tracking we will explain our detection algorithm that consists in a pose estimation based in background subtraction for multiple targets.

A. Target Motion Model

The main objective of the thesis is tracking buses, which have a predictable motion model since they cannot move sideways and its acceleration is low. By knowing the buses motion model we can assume its position even if it is not detected for some time [2]. A bus has the same motion model as a car with the slight difference that the bus is longer.

1) Continuous Model: The model chosen to characterize the buses is the bicycle model, which is represented in Fig.6 because a bus, in its motion, behaves like a bicycle with its wheels centered in the center of the buses axle shafts.

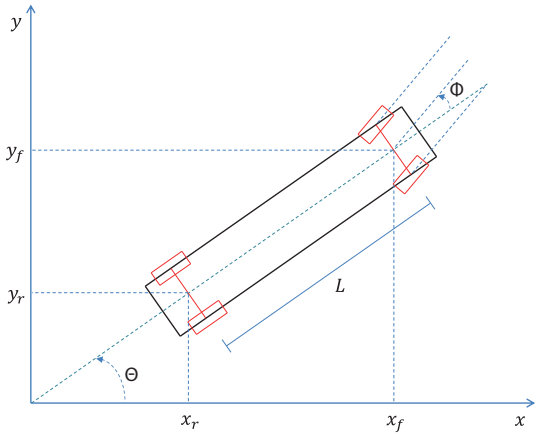


Fig. 6. Bus model

This model is composed by three observed variables: x_f and y_f, which correspond to the position of the center of the front axle shaft along the correspondent axis, and θ, the angle of the bus orientation in relation to the axis x.

The non-observed variables are the linear velocity v and the steering angle ϕ, which represent the orientation of the wheels with respect to θ.

The turning radius will also depend on the constant L which represents the distance between the centers of both axle shafts.

To understand the continuous motion model we will represent the coordinates of the centers of the front and rear axle shafts as (x_f, y_f) and (x_r, y_r).

The bus constraints are the non-holonomic restriction of the rear wheel axle shaft center and the fixed distance between axle shafts. They can be represented through the following equations:

$$\begin{cases} \dot{y}_r \cos(\theta) - \dot{x}_r \sin(\theta) = 0 \\ x_r = x_f - L\cos(\theta) \\ y_r = y_f - L\sin(\theta) \end{cases} \tag{16}$$

and by observing the relation between the observed and the non-observed variables we can add:

$$\begin{cases} \dot{x}_f = v\cos(\theta + \phi) \\ \dot{y}_f = v\sin(\theta + \phi) \end{cases} \tag{17}$$

By combination and manipulation of Eq.16 and Eq.17 we can conclude that the continuous motion model is given by:

$$\begin{cases} \dot{x}_f = v\cos(\theta + \phi) \\ \dot{y}_f = v\sin(\theta + \phi) \\ \dot{\theta} = \frac{v}{L}\sin\phi \end{cases} \tag{18}$$

2) Discrete Model: In order to use the Extended Kalman Filter we need to discretize de motion model differential functions. We define the system state as $X_{k+1} = (x_{k+1}, y_{k+1}, \theta_{k+1}, v_{k+1}, \phi_{k+1})$ and must compute it in respect to X_k.

Using the leapfrog integration method we conclude that the discrete motion model is given by:

$$\begin{cases} x_{k+1} = x_k + T \cdot v_k \cos(\theta_k + \phi_k + \frac{T}{2} \cdot \frac{v_k}{L}\sin\phi_k) \\ y_{k+1} = y_k + T \cdot v_k \sin(\theta_k + \phi_k + \frac{T}{2} \cdot \frac{v_k}{L}\sin\phi_k) \\ \theta_{k+1} = \theta_k + T \cdot \frac{v_k}{L}\sin\phi_k \\ v_{k+1} = v_k \\ \phi_{k+1} = \phi_k \end{cases} \tag{19}$$

B. Bus Tracking Algorithm - Extended Kalman Filter

The algorithm we chose to track the buses was the Extended Kalman Filter [2].

In this model the estimation of the state in the next time step $(k+1)$ is given by:

$$\begin{aligned} X_{k+1} &= f(X_k, \xi_k) \\ z_{k+1} &= h(X_{k+1}, \eta_k) \end{aligned} \tag{20}$$

The f function is used to estimate the next state based on the motion model while the function h estimates the sensor readings from the information of the state estimation.The ξ_k represents system noise while η_k represents measurement noise and they are both independent variables with normal distribution and zero mean.

The EKF has two modes, predict and update. These modes are described in the next sections along with the error ellipse, which is the geometric quantizer of the EKF uncertainty.

1) EKF Predict: The EKF predict mode takes into consideration the previous system state and covariance matrix and predicts the next states. It does it by maintaining v and ϕ constant and computing all the observed variables in the next desired time step.

The equations used to predict the state and system variance used are the following:

$$\begin{cases} \hat{X}_{k+1|k} = f(\hat{X}_{k|k}) \\ \hat{P}_{k+1|k} = F_k \cdot \hat{P}_{k|k} \cdot F_k^T + N_k \cdot \hat{Q}_k \cdot N_k^T \end{cases} \tag{21}$$

where $\hat{P}_{k+1|k}$ and $\hat{P}_{k|k}$ are the predicted covariance matrices of the current and the last state, respectively and F_k is the Jacobian matrix of f and N_k represents the partial derivatives of f with respect to ξ. Since we assume that we have additive process noise N_k is simply an identity matrix and Q_k is a square diagonal matrix representing the system noise covariance which is assumed to be Gaussian with zero mean.

2) EKF Update: The EKF update mode balances the predicted results in EKF predict with the observed ones to give the best estimation of system state and covariance matrix. In this case our bus detection algorithm, detailed in Sec.III-C, only observes position and orientation, which means x, y and θ.

The equations used to update the state and system variance is computed from the prediction data and are the following:

$$\begin{cases} \tilde{y}_{k+1} = z_{k+1} - h(\hat{X}_{k+1|k}) \\ S_{k+1} = H_{k+1} \cdot \hat{P}_{k+1|k} \cdot H_{k+1}^T + M_k \cdot \hat{R}_k \cdot M_k^T \\ K_{k+1} = \hat{P}_{k+1|k} \cdot H_{k+1}^T \cdot S_{k+1}^{-1} \\ \hat{X}_{k+1|k+1} = \hat{X}_{k+1|k} + K_{k+1} \cdot \tilde{y}_{k+1} \\ \hat{P}_{k+1|k+1} = (I - K_{k+1} \cdot H_{k+1}) \cdot \hat{P}_{k+1|k} \end{cases} \tag{22}$$

$\tilde{y}_{k+1}$ is the measurement residual with S_{k+1} being the residual covariance. K_{k+1} represents the matrix of near-optimal Kalman gains and H_{k+1} the Jacobian matrix of h. The matrix of partial derivatives in respect to the observed noise, M_k, is an identity matrix since the observed noise is also addictive. R_k is the matrix containing the observation noise covariance which is assumed to be Gaussian with zero mean.

If the system is linear, this filter becomes the Kalman Filter instead of Extended Kalman Filter and the estimation can be optimal. On the other hand, if the motion model is not correct or the initial estimate is wrong, the EKF may diverge due to the linearization. See [8].

3) Error Ellipse: Since the position of the bus is considered as a 2 dimensional Gaussian probability density function we can describe the uncertainty in its position by an error ellipse [3].

The error ellipse is a 2D representation of a confidence bound on a given position. These ellipses only represent the uncertainties in x and y coordinates, which not only depend on themselves but also on θ, v and ϕ. These uncertainties are described by the top-left 2×2 covariance sub-matrix of P. The error ellipse equation is given by:

$$X = P^{\frac{1}{2}} Y + X_C \tag{23}$$

with X as the ellipse points, X_C the coordinate of the center of the ellipse and Y a point in the unit circle.

The error ellipse is a good visual representation of uncertainty but to compare ellipses we need its area which is given by $A = \pi r_{max} r_{min}$ where r_{max} and r_{min} are the biggest and smallest radius of the ellipse.

C. Bus Detection Algorithm

This section explains how the observed state used in the EKF is calculated from an image. Our detection algorithm only detects pose: position and orientation.

First we locate the bus in the image using background subtraction, which is explained in Sec.III-C1. Afterwards we estimate the bus front axle shaft position with a back projection algorithm. With this estimation we use a cost function minimization algorithm to improve the position estimate and to also estimate the bus orientation, that is detailed in Sec.III-C2.

1) Background Subtraction: In the simulation environment there are two worlds running in parallel. The first of them has the buses inserted and we call it the original world. The second world, which we call background world, simulates a database with pictures of the background taken with all possible pan, tilt and zoom angles which in a real camera have a finite number of possibilities.

For the background subtraction algorithm [5] we will subtract the original image im_{org} with the background image, im_{bck}, and compute a binary image im_{bgs} that will have pixels with the value 0 when the subtraction is zero and 1 otherwise. Since the background model is almost exact we do not need to add a threshold bigger than zero.

2) Pose Estimation: To estimate the pose we use a cost function minimization algorithm that finds the minimum of a non linear scalar function of several variables, starting from an initial estimate. The cost function used is the following one:

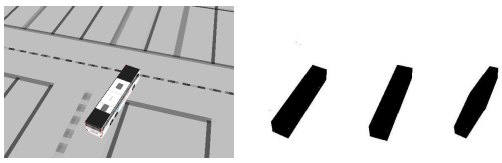
$$F(X, Y, \theta) = 1 - \frac{\#(A \cap B)}{\#B} \tag{24}$$

where A and B are both binary masks. A is obtained from the background subtraction algorithm of section III-C1. B is a syntethic mask, generated by placing a 3D model of the bus in an image of the background and then using it in the same background subtraction algorithm.

With each iteration of the algorithm the X, Y and θ values are updated so that the value of the cost function $F(X, Y, \theta)$ approaches zero. When the search ends, the masks A and B will ideally be overlapping perfectly. In practice this overlap may not be perfect ($F \neq 0$ in Eq.24). $A \cap B$ correspond to a binary multiplication or an AND logical operation between the pixels of mask A and B. An example of this operation is shown in Fig.7. Note that the pixel values are inverted in pictures (b), (c) and (d).

The initial 3D model orientation used is the θ predicted with the EKF predict.

For the estimation of the pose starting position we used the back projection algorithm described in Sec.II-A2. From the mask A we compute the center of mass of the bus and with it we compute the bus the coordinates X and Y by making Z equal to half the bus height and shift the coordinate to match the front axle shaft.



(a) (b) (c) (d)

Fig. 7. (a) Camera view (b) Mask A (c) Mask B (d) Intersection of A and B

3) Multitasking: When we have more buses than cameras inside the park we need to have a multitasking algorithm to manage which camera tracks which bus and which bus will not be tracked. Our multitasking algorithm is simply a scheduling method. Our chosen scheduling discipline was the fixed-priority pre-emptive scheduling. It consists in executing always the highest priority task of all tasks that are ready to be executed. In the simulator case we will always assign to the cameras the buses with highest associated uncertainty, which means ellipse area.

After having decided which buses will be tracked we must assign a camera to them. This assignment will be made with a FIFO algorithm (First In First Out). The bus with the highest uncertainty will have the closest camera to it in pan, $tilt$ and $zoom$ units and so on.

IV. Complete System

In this section we describe the complete system. This system has three main components: the 3D world, the buses and the cameras. The 3D world, which is the parking lot floor, is described as a plane; the buses have autonomous motion (are driven by on board drivers); and the cameras are PTZ and are mounted at known positions and orientations in the 3D world.

Every second we run five main modules: EKF Predict, Multitasking, Move Cameras, Process Images and EKF Update. We also have a sixth module which only runs once per bus, which is the EKF Initiate.

The system is described in the diagram of Fig.8. Blue rectangles denote the modules and red rectangles denote their inputs and outputs. A red rectangle near the arrow tip denotes an input, while near the base, it denotes an output. The blue circle represents a time iteration, k is the time variable in seconds, and the diamonds represent an if function with the filled red rectangles representing their possible answers. Finally we have the start and end blocks.

The starting block initiates the structures with the information about the buses and available cameras. These structures are constantly being updated and all blocks have access to them. The buses structure start empty, since the parking lot is empty in the beginning, and the cameras structure starts with their initial orientation in pan, $tilt$ and $zoom$ units. The ending block corresponds to stopping the system.

A. EKF Predict

The EKF Predict module will predict the next system state based on previous observations and on the buses motion

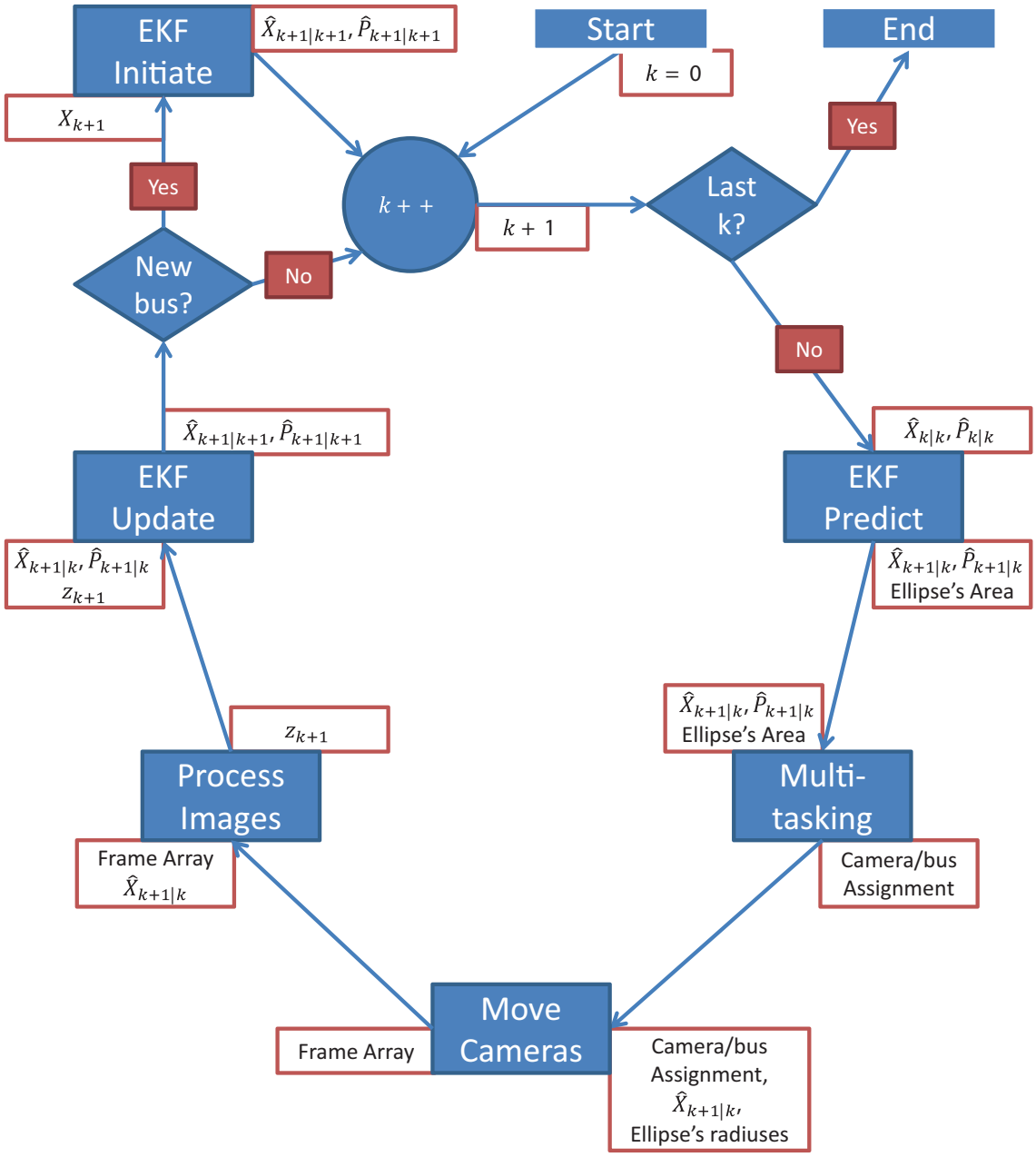


Fig. 8. Schematic of the Complete System

model. This module receives as input the previous system state estimation and the covariance matrix for each bus that is inside the parking lot. As output it will return the prediction of the current system state, the correspondent covariance matrix and error ellipse area for every bus.

The prediction of the current system state will be computed from Eq.21 as explained in Sec.III-B1. Q_k, the system noise covariance, is constant and is explained in Sec.IV-F

The error ellipse will be composed by fifty points and the area of the ellipse is calculated with $r_{max} = max(X_i)$ and $r_{min} = min(X_i)$.

In the end, our whole prediction will be represented as shown in Fig.9 were we have a bus which has the front wheels red and the rear wheels violet. As we can see in the image the front wheels have approximately the same direction of the maximum radius of the ellipse. Since given the EKF gives more importance to the velocity than to the steering. Next step: Multitasking.

B. *Multitasking*

The Multitasking module is what allows the tracking of the buses when there are more buses than cameras. It applies an algorithm that assigns the available cameras to the buses. It receives as input the predicted states and correspondent covariance matrices and ellipse areas for all the buses. As output it returns a camera/bus assignment vector. Our multitasking algorithm, that is detailed in Sec.III-C3. After the assignment, the next step is the camera movement.

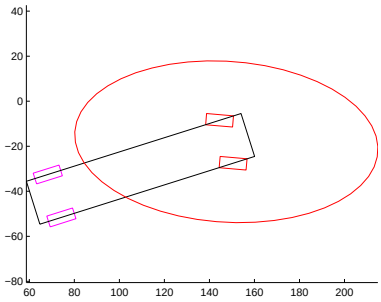


Fig. 9. State prediction and error ellipse

C. Move Cameras

The Move Cameras module is the module were the cameras move to point to the predicted position of the buses that they are supposed to track. As input it receives the camera/bus assignments and the predicted position and ellipse radius. As output it returns a frame array with the captured frames at the designated position.

The first step of this module is computing the necessary pan and tilt angles so that the predicted position is centered in the image frame with Eq.14. To compute the necessary zoom we use Eq.15 were the radius rd is equal to the biggest ellipse radius.

After doing all the calculations the pan, tilt and zoom angles are applied to the cameras which can only make one movement per second. The pictures are taken and are sent to the next module for processing. Fig.10 is an example of a sent picture.

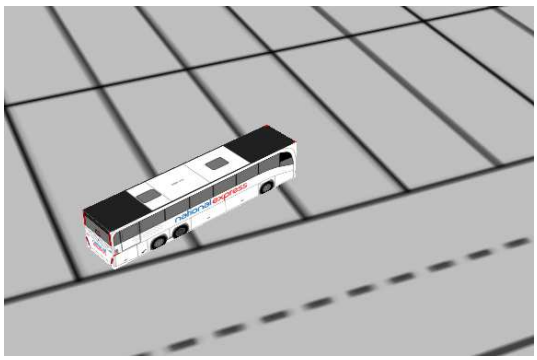
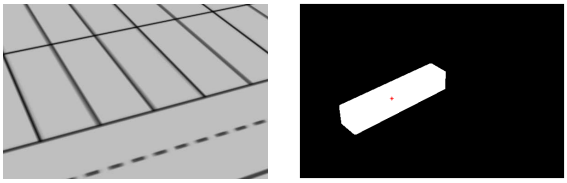


Fig. 10. Example of a picture taken by the camera

D. Process Images

In the Process Images module we run the bus detection algorithm described in Sec.III-C. This module receives image frames and pose estimations and returns the observations of bus poses.

We start by doing a background subtraction with the help of an auxiliary background model. After the background subtraction we will end up with a silhouette, as shown in Fig.11. The original picture is the one from Fig.10. Then we calculate the center of mass of the bus in pixels and with it we can estimate the starting point for the minimization algorithm. Then, we compute the center of mass of the bus in pixels that will simply be the mean position of the silhuette pixels, and is represented with a red star in the same figure.



(a) (b)

Fig. 11. (a) Background, (b) Background Subtraction Mask

Since we know that the center of mass is located in the plane $Z = 1$ in meters we apply the back-projection algorithm that is described in section II-A2. To solve the back projection equation we have the projection matrix of the cameras to compute C, the center of mass pixel coordinate to compute D and the Z coordinate to finally compute X and Y. Now if we shift the coordinates to match the front axle shaft we have a good estimation of the starting point for the minimization algorithm.

The minimization algorithm will then minimize the cost function Eq.24 by trying to approximate a synthetic bus pose to the original one by comparing their silhouettes. When the function is minimized we use the synthetic pose as our observation value.

If there is more than one bus in the image it will not affect the detection of the desired bus since the algorithm computes the mean for all silhouettes, applies the back projection algorithm for all means and compares them to the predicted state. The closest point will be our choice.

E. EKF Update

The EKF Update module takes as input the predictions made by the EKF and the observations made by our bus detection algorithm and compute the current system state and covariance for all observed buses. The buses that had no camera assigned to them were not observed and do not enter this module.

The procedure used is described in Sec.III-B2 which is based in Eq.22 with R_k, the observation noise covariance matrix described and explained in the next section.

After this module we have two options: if there was a new bus entering we go to the EKF Initiate, if there is not any we simply iterate the time k and start the process again.

F. EKF Initiate

The EKF Initiate module will initiate all variables needed for the EKF predict to work in the next time step. As input it will receive X_{k+1}, which is the real state of the bus and return as output the estimation of the real state which will be the same of the input but with an initial covariance matrix.

This real pose assignment, without the camera having to actually look at the bus simulates the entrance of the bus in the park.

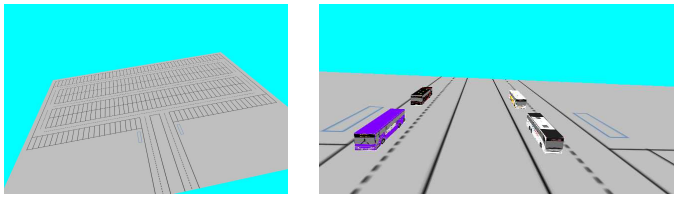
This module initializes the initial covariance matrix which will result in an ellipse error very small when in comparison

with the ellipses returned by the EKF Predict module since there is almost no uncertainty.

This module also initializes the system and measurement noise covariance matrices which will be constant and whose values must be chosen by trial and error and will depend on the constraints of the hardware.

V. Simulator Results

In order to test the complete system a 3D simulator was developed in MATLAB using VRML. The 3D world is composed by a plane at $Z = 0$ which is the parking lot floor, Fig.13(a). This world has buses which are represented as parallelepipeds with 2m height and width and with 10m of length, Fig.13(b), and cameras that are VRML viewpoints. The buses have predefined trajectories. Since we do not have a bus controller we decided to make a joystick to control linear velocity and steering wheel and apply the motion model.



(a) (b)

Fig. 13. (a) Parking lot floor, (b) The four used buses

The simulated system encompasses two cameras (C_1 and C_2) and four buses (B_1, B_2, B_3 and B_4). Figure 12(a) shows an aerial view of the parking lot at a simulation iteration where the four buses were all in the scene. The center of the world referential frame is at the center of the floor plane of figure 12(a). The cameras were positioned at the red dots location and oriented in the direction of the ”red triangles” base associated to each red dot. Typical images acquired by the camera are shown in figures 12(b) and 12(c).

The chosen initial covariance matrix was $diag(0.1^2, 0.1^2, 1^2, 0.1^2, 1^2)$ which results in a error ellipse much smaller than the bus itself which means we have a high degree of confidence on the bus initial pose. The covariance diagonal values, before the square operation, are in m, m, deg, m/s and deg. The real pose will have v and ϕ equal to zero, meaning the bus stops before entering the park.

The system and measurement noise covariance matrices were $Q_k = diag(1^2, 1^2, 5^2, 2^2, 7.2^2)$ and $R_k = diag(1^2, 1^2, 15^2)$, respectively. All these values were chosen by trial and error.

At the end of the experiment the plots presented in figure 14 were obtained.

In these plots the rectangles position and orientation represent the cumulative estimates obtained from the EKF predict step for each bus, at the end of the simulation. The red wheels are the front wheels while the purple are the rear wheels. The position of the red wheels is also calculated with the EKF predict. The elliptical lines are the limits of the error area associated with the EKF estimation. As can be noted in these graphs the program was able to keep track of the positions of the buses during all the simulation. There were however certain positions where the estimation was not accurate. However the system was always able to recover from the errors and continue tracking the buses.

In Fig.15 we can see the same graphs from Fig.14 but zoomed in specific locations to show more detail.

In Fig.15(a) we see an example of what happens when one camera is attached to one bus for a certain period of time. This may happen if there are less or the same number of buses than cameras in the park. We can see that the error ellipses do not evolve through time, they just move in position and orientation.

In Fig.15(b) there is an example of a bad observation that evolves into a bad prediction that is constantly being corrected. Since the error ellipses manage the zoom, the camera never looses track of the bus. In this particular example what happened was that the bus was moving with high velocity and when it braked and turned there was no camera observing it and the EKF predicted that the bus was still moving forward. When the bus was observed again it was not where it was supposed to be, even though it was still inside the error ellipse and the sudden change in pose gave bad EKF update results since it still balanced the observation with the prediction. Even with these mistakes the system was able to stay stable and recover from the bad readings.

In Fig.15(c) we see the error ellipses evolution when the bus is going in a straight line with constant velocity and is being observed and not observed sequentially. When it was observed the error ellipse is smaller, when it was not it is bigger.

Finally, in Fig.15(d) we show a correct observation and prediction during a turn. We can see that the buses front wheels are always parallel to the desired trajectory.

VI. Conclusion and Future Work

The work described in this project consisted in three main themes: camera model and target tracking and detection and multitasking. By combining these themes we built a system that tracks and detects buses in a parking lot only using cameras which are in less number than the buses. To test this system we built a simulator in Matlab VRML.

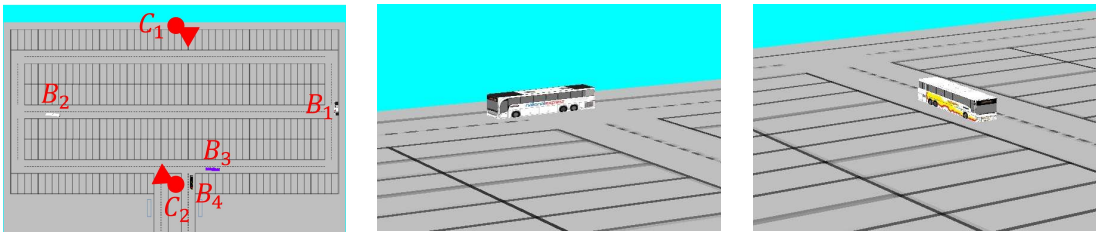
In the camera model we studied the PTZ model, their direct and inverse kinematics and the PTU calibration. With this we could move freely a camera in a scene by giving pan, tilt and zoom inputs or world coordinates.

In the target tracking we studied the EKF. We concluded that the EKFs are good at estimating the position even with not so good observation results. In the target detection we studied a pose estimation algorithm that is good enough for the whole system to work properly and recover from estimation errors.

The system tested in Matlab took into consideration that it could be adapted to receive real camera inputs, even though the detection algorithm would need some adaptations. We did not take into consideration the camera velocity which is not infinite in reality.

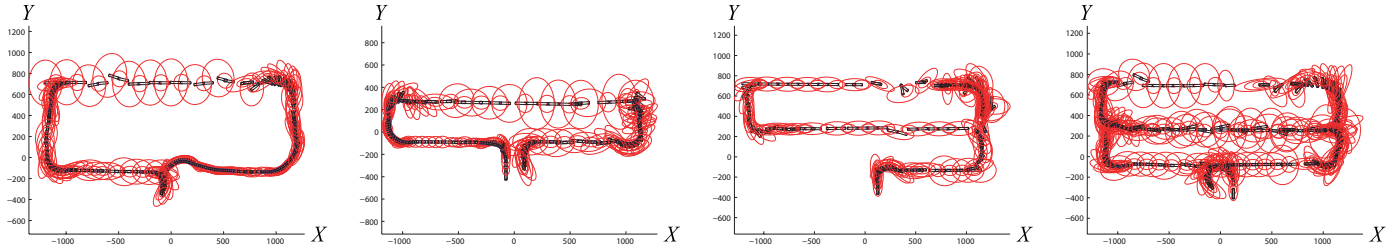
In the future we need to take into consideration automatic calibration methods for PTZ cameras and modify the system to receive cameras with finite velocity.

To adapt tracking to the real world we need to study the trucks plus trailer motion model so that the estimations made



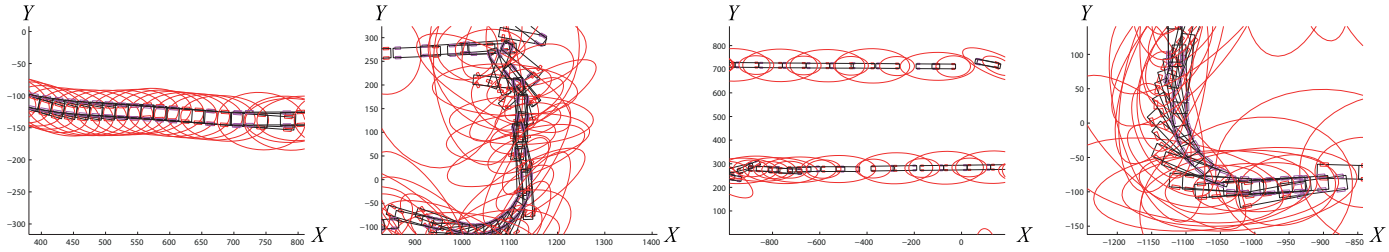
(a) (b) (c)

Fig. 12. (a) Global view (for buses). (b) North camera view(C_1). (c) South camera view(C_2).



(a) (b) (c) (d)

Fig. 14. Trajectories associated to each bus and their correspondent error ellipses in dm. (a) B_1. (b) B_2. (c) B_3. (d) B_4



(a) (b) (c) (d)

Fig. 15. Detail in the trajectories associated to each bus and their correspondent error ellipses in dm. (a) B_1. (b) B_2. (c) B_3. (d) B_4

with the Extended Kalman Filter can work. In the detection algorithm we should improve the algorithm so that we could identify different types of vehicles with different sizes and adapt the EKF accordingly. For the observation we could add an algorithm to calculate velocity and steering from the images received.

On the Multitasking subject future work will focus on algorithms that take into account the cameras pan, tilt and zoom velocities which would make more efficient tracking.

REFERENCES

[1] Jean-Yves Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj.

[2] R. J. B. Carona. Controlo visual de robots tipo uniciclo. *Dissertação para a obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores*, 2009.

[3] P. Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, 2011.

[4] L. de Agapito, E. Hayman, and I. Reid. Self-calibration of a rotating camera with varying intrinsic parameters. *Proc 9th British Machine Vision Conference, Southampton*, pages 883–893, 1998.

[5] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2011.

[6] R. M. F. Galego. Geometric and radiometric calibration for pan-tilt surveillance cameras. *Dissertação para a obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores*, 2011.

[7] W. Starzyk and F. Z. Qureshi. Multi-tasking smart cameras for intelligent video surveillance systems. *8th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 2011.

[8] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[9] D. A. P. Vicente. Event detection with pan-tilt cameras. *Dissertação para a obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores*, 2009.