Motion Detection and Tracking for an AIBO Robot Using Camera Motion Compensation and Kalman Filtering[1]

Javier Ruiz-del-Solar and Paul A. Vallejos

Department of Electrical Engineering, Universidad de Chile
{jruizd, pavallej}@ing.uchile.cl

Abstract. Motion detection and tracking while moving is a desired ability for any soccer player. For instance, this ability allows the determination of the ball trajectory when the player is moving himself or when he is moving his head, for making or planning a soccer-play. If a robot soccer player should have a similar functionality, then it requires an algorithm for real-time movement analysis and tracking that performs well when the camera is moving. The aim of this paper is to propose such an algorithm for an AIBO robot. The proposed algorithm uses motion compensation for having a stabilized background, where the movement is detected, and Kalman Filtering for a robust tracking of the moving objects. The algorithm can be adapted for almost any kind of mobile robot. Results of the motion detection and tracking algorithm, working in real-world video sequences, are shown.

1 Introduction

Movement analysis is a fundamental ability for any kind of robot. It is especially important for determining and understanding the dynamics of the robot's surrounding environment. In the case of robot soccer players, movement analysis is employed for determining the trajectory of relevant objects (ball, team mates, etc.).

However, most of the existing movement analysis methods require the use of a fixed camera (no movement of the camera while analyzing the movement of objects). As an example, the popular background subtraction movement detection algorithm employs a fixed background for determining the foreground pixels by subtracting the current frame with the background model. The requirement of a fixed camera restricts the real-time analysis that a soccer player can carry out. For instance, a human soccer player very often requires the determination of the ball trajectory when he is moving himself, or when he is moving his head, for making or planning a soccer-play. If a robot soccer player should have a similar functionality, then it requires an algorithm for real-time movement analysis that can perform well when the camera is moving. The aim of this paper is to propose such an algorithm for an AIBO robot. This algorithm can be adapted for almost any kind of mobile robot.

The rationale behind our algorithm is to compensate in software the camera movement using the information about the robot body and robot head movements.

[1] This project was partially funded by the FONDECYT (Chile) project 1030500.

D. Nardi et al. (Eds.): RoboCup 2004, LNAI 3276, pp. 619–627, 2005.

This information is used to correctly align the current frame and the background. In this way a stabilized background is obtained, although the camera is always moving. Afterward, different traditional movement analysis algorithms can be applied over the stabilized background. Another feature of our algorithm is the use of a Kalman Filter for the robust tracking of the moving objects. This allows to have reliable detections and to deal with common situations such as double detections or no detection in some frames because of lighting conditions.

2 Related Work

A large literature exists concerning movement analysis in video streams using fixed cameras. As an example, every year is held the PETS event, in which several state-of-the-art tracking and surveillance systems are presented and tested (see for example [4] and [5]). Different approaches have been proposed for moving object segmentation; including frame difference, double frame difference, and background suppression or subtraction. In the absence of any a priori knowledge about target and environment, the most widely adopted approach is background subtraction [3]. *Motion History* is another simple and fast motion detection algorithm. According to [8], the *Motion History* and *Background Subtraction* algorithms have complementary properties, and when possible it is useful their join use.

Image alignment using gradient descending is one of the most used alignment algorithms. It can be divided into two formulations: the additive approach, which consists on start from an initial estimation of the parameters, and iteratively find appropriates parameters increments until the estimated parameters converge [7]; and the compositional approach, which estimates the parameters using an incremental warp. This last approach iteratively solves the estimation problem using an incremental warp of the images to be aligned with respect to a template. This allows pre-computing the Jacobean more efficiently [1]. But the key for obtaining an efficient algorithm is switching the role of the image and the template. This leads to the formulation of the inverse compositional algorithm [2], where the most computationally expensive operations are pre-calculated, allowing a faster convergence. In [6] it was proposed the robust inverse compositional algorithm as an extension to the inverse compositional algorithm, allowing the existence of outliers into the alignment with almost the same efficiency.

Regarding moving objects tracking, Kalman Filtering, Extended Kalman Filtering and Particle Filtering (also known as Condensation and Monte Carlo algorithms) are some of the most common used algorithms. Due to its simplicity, the Kalman filter is still been used in most of the general-purpose applications.

The here-proposed motion detection and tracking system is based in the described algorithms: background difference and motion history for motion detection, robust inverse compositional algorithm for the image and background alignment, and Kalman filtering for the tracking of moving objects.

3 Proposed Motion Detection System

3.1 System Overview

In figure 1 is shown a block diagram of the proposed system. The system is composed by four main subsystems: *Image Alignment*, *Motion Detection*, *Detection Estimation, and Background Update*. In the *Image Alignment* module, the last updated background image (B_{k-1}) and the last frame image (I_{k-1}) are aligned with respect to the current frame image (I_k). The camera motion angles (α_k) are employed in this alignment operation. Both aligned images, B_k^* and I_{k-1}^*, respectively, are then compared with I_k in the *Motion Detection* module for determining the current moving pixels. As a result of these comparisons the *Motion History* and *Background Subtraction* algorithms generate preliminary detections (a set of moving pixels), D_{H_K} and D_{B_K}, respectively. These detections are joined in the *Rejection Filter* module, and a single set of candidate blobs (in this case moving objects), built using adjacent moving pixels, Det_k, is obtained. The motion detections are analyzed in the *Detection Estimation* module using a Kalman Filter, and the final detections Det_k* are obtained. Finally, the background is updated using B_k^*, I_k and Det_k* (which defines the new foreground pixels) by the *Background Update* module.

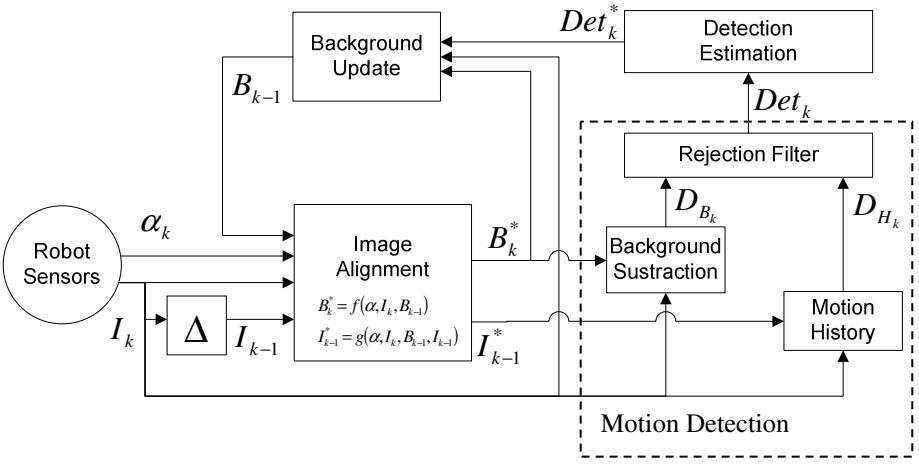

Fig. 1. Block diagram of the proposed system. Parameters are described in the main text

3.2 Image Alignment

The alignment of the last updated background image (B_{k-1}) and the last frame image (I_{k-1}) is implemented using the robust inverse compositional algorithm [6]. The alignment operation is implemented as a sequence of incremental warps (see section 2). The initial estimation of the warp is calculated based on the camera motion angles (stored in the α_k vector; they correspond to the tilt, pan and roll camera rotation

angles). The initial estimated warp is a composition of a rotation followed by a displacement. The angle of rotation is estimated as the variation of the roll angle of the camera, while the displacement *Dx*/*Dy* in the X/Y axis corresponds to the pan/tilt angle:

$$\begin{aligned} \alpha_R &= \alpha_{pan2} \cdot \sin(BA - \alpha_{tilt2}) - \alpha_{pan1} \cdot \sin(BA - \alpha_{tilt1}) + \alpha_{roll2} - \alpha_{roll1} \\ Dx &= (\alpha_{pan2} - \alpha_{pan1}) \cdot 180 \\ Dy &= ((BA - \alpha_{tilt2}) \cdot \cos(\alpha_{pan2}) - (BA - \alpha_{tilt1}) \cdot \cos(\alpha_{pan1})) \cdot 180 \end{aligned} \quad (1)$$

where α_R represent the rotation in radians, *Dx* and *Dy* represent the displacement in pixels in their respective axis, BA is the angle of the body, and the angles α_{tilt1}, α_{pan1}, α_{roll1}, α_{tilt2}, α_{pan2}, α_{roll2} are the tilt, pan and roll angles of the robot's head in the last and in the current image, respectively, measured in radians.

Then the warp is defined by a set of six parameters as:

$$\begin{aligned} Wx &= (1 + P1) \cdot x + P3 \cdot y + P5 \\ Wy &= P2 \cdot x + (1 + P4) \cdot y + P6 \end{aligned} \quad (2)$$

where (*Wx*,*Wy*) define the new pixel coordinates which initial coordinates were (*x*,*y*). A pure displacement warp has the parameters P1 to P4 equals to zero, and the parameters P5 and P6 equals to the displacement in pixels in the x and y axis respectively. A pure rotation warp has the parameters P1 to P4 equals to the rotation matrix, and the parameters P5 and P6 equals to zero. Finally, a compound warp of a rotation followed by a translation have the parameters: $P1 = \cos(\alpha_R) - 1$, $P2 = \sin(-\alpha_R)$, $P3 = \sin(\alpha_R)$, $P4 = \cos(\alpha_R) - 1$, $P5 = Dx$, $P6 = Dy$.

For aligning B_{k-1}, the area of the current image (I_k), which has being estimated to overlap the background, is chosen as a template for the algorithm. This preliminary template is divided into nine blocks (sub-images). In each block is calculated the normalized variance of its pixels (intra-block variance), and the normalized variance of the error with respect to the correspondent block in the background (inter-block variance). A variability factor is computed as the quotient of the intra-block variance and the inter-block variance. The six blocks with the largest variability factor are selected as the final templates for the background alignment. Taking into account the normal camera motion, B_{k-1} and I_k should have different spatial sizes for a correct alignment. In our implementation B_{k-1} has the same height than I_k, but the double of its width. We will denote the set of parameters defining the warping of the background $\mathbf{P_B}$. The algorithm for obtaining $\mathbf{P_B}$ is detailed described in [6].

For aligning I_{k-1} the calculated warp of B_{k-1} is employed as a first approximation. However, given that I_{k-1} and I_k have the same size, the calculated warp has to be actualized with a composition with a prior displacement to achieve the same spatial configuration of the background (I_{k-1} should be translated into background spatial coordinates), and then with a composition with a post displacement to reach the spatial configuration of the current image (the warped image should be taken back to its original coordinates). Thus, defining $\mathbf{P_1}$ as the set of parameters needed to produce a displacement equal to the last position of I_{k-1} inside the background, and defining $\mathbf{P_2}$ as the set of parameters needed to produce a displacement equal to the inverse of the

estimated final position of I_k inside the background, the warp needed to align I_{k-1} is (the set of parameters defining this warping are $\mathbf{P_I}$):

$$W(\mathbf{x},\mathbf{P_I}) = W(\mathbf{x},\mathbf{P_2}) \circ (W(\mathbf{x},\mathbf{P_B}) \circ W(\mathbf{x},\mathbf{P_1})) \tag{3}$$

For simplicity on the notation, **x** denotes both spatial image coordinates. The function W(**x**, **P***) corresponds to a warping operation over **x** using the set of parameters **P***.

3.3 Motion Detection

The *motion detection* module is composed by three algorithms, *Motion History* and *Background Subtraction* for movement detection, and *Rejection Filter* for filtering wrong detections and forming the movement blobs.

3.3.1 Motion History

The difference image DM_k is defined as:

$$DM_k(\mathbf{x}) = \begin{cases} mIncrement & \text{if } \left|I_k(\mathbf{x}) - I^*_{k-1}(\mathbf{x})\right| > T_m \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where *mIncrement* corresponds to a factor of increment in the motion and T_m corresponds to a motion threshold. DM_k contains the initial set of points that are candidate to belong to the MVOs (Moving Visual Object). In order to consolidate the blobs to be detected, a 3x3 morphological closing [10] is applied to DM_k. Isolated detected moving pixels are discarded applying a 3x3 morphological opening [10]. The motion history image MH_k, calculated from DM_k, is then updated as:

$$MH_k = MH_{k-1} * DecayFactor + DM_k \tag{5}$$

Finally, all pixels of MH_k whose luminance is larger than a motion detection threshold (T_h) are considered as pixels in motion. These pixels generate the detection image D_{H_K}. 3x3 morphological closing and opening are applied to D_{H_K}.

3.3.2 Background Subtraction

Foreground pixels are selected at each time k by computing the distance between the current image I_k and the current aligned background B^*_k, obtaining D_{B_K} as:

$$D_{B_k}(\mathbf{x}) = \begin{cases} 1 \ if \ \left|I_k(\mathbf{x}) - B^*_k(\mathbf{x})\right| > T_p \\ 0 \ \ otherwise \end{cases} \tag{6}$$

In order to consolidate the blobs to be detected, a 3x3 morphological closing is applied followed by a 3x3 morphological opening.

3.3.3 Rejection Filter

By means of 8-connectivity movement blobs, composed by connected candidate moving pixels, are built using D_{H_K} and D_{B_K}. For each blob *b* is defined a movement density MD_b as:

$$MD_b = \frac{\sum_{x \in b} |I_k(x) - I_{k-1}(x)|}{Area(b)} \tag{7}$$

MD_b measures the average change in the last frame for the blob *b*. Ghosts (groups of pixel that are not moving, detected like movement because they were part of a MVO in the past) should have a low MD_b, while the MVOs should have a large MD_b. Then, blobs with a small area (area $\leq T_b$), with a large area (area $\geq T_s$) and blobs with a small movement density ($MD_b \leq T_d$) are considered miss detections and discarded.

3.4 Detection Estimation

Targets (the MVOs) are tracked by keeping a list with the state of each of them. The state of a given target *u* includes: the position of the center of mass (x_u, y_u), the speed (Vx_u, Vy_u), the area (a_u) and the growing speed (Va_u). For each received movement blob is calculated the area and the center of mass. These variables are used as sensor measurements, and integrated across the different motion detections (the ones coming from D_{H_K} and D_{B_K}) and over time using a first order Kalman Filter [10]. This process includes 6 stages: *prediction*, *measure-target matching*, *update*, *detection of new targets*, *deprecated targets elimination* and *targets merge*. After those 6 stages the target state list, whose values are estimated by the Kalman Filter, corresponds to the final motion detections (Det_k*).

Prediction. Using a first order cinematic model it predicts the state vector for each target, based on the last estimated state, and projects the error covariance ahead.

Measure-Target matching. In order to update the targets is imperative identifying which (blob) measure affects each target. For each measure-target combination is calculated a confidence value as the probability function given by the Kalman filter for the target evaluated on the measure. For each target, all measures with a confidence value over a threshold T_{t1} are associated with the target. If a measure does not have any associated target, then it is consider as a new target candidate and passed to the *Detection of new targets* stage. For each measure associated with a target, speeds (spatial speed: *Vx* and *Vy*, and growing speed *Va*) are to be estimated. This estimation is performed using the difference between the target state before the prediction and the measured state, divided by the elapsed time since last prediction.

Update. It computes the Kalman gain, updates the state vector for each target using their associated measures, and updates the error covariance. The targets without associated measures are not updated.

Detection of new targets. All measures without an associated target are considered as new target candidates. Their spatial speed is calculated as the distance from the image

border, in the opposite direction of the image center, divided by the elapsed time since the last frame, and their growing speed is set to zero.

Deprecated targets elimination. Targets without associated detections in the last 2 frames are considered as disappeared MVO and eliminated from the target list.

Targets merge. For each target-target combination two confidence value are calculated as the probability function given by the Kalman filter for one target evaluated on the other target state. If any of this confidence values is over a threshold T_j, then the two targets are considered equivalents, and the target with the largest covariance (measured as the Euclidian norm of the covariance matrix) is eliminated from the target list.

3.5 Background Update

The background model is computed as the weighted average of a sequence of previous frames and the previously computed background:

$$B_k(\mathbf{x}) = \begin{cases} B_k^*(\mathbf{x}) & if\ (\mathbf{x}) \in DET_k^* \\ \\ \alpha I_k(\mathbf{x}) + (1-\alpha) B_k^*(\mathbf{x}) & otherwise \end{cases} \tag{8}$$

4 Experiments

For the experiments, an AIBO robot using the motion software of the UChile1 AIBO soccer team [11], configured for allowing just head movements was used. The algorithm runs in the robot in real time. For analysis purposes, two video sequences were employed. In both, the robot moves its head in an ellipsoidal way, keeping the roll angle of the camera approximately aligned with the horizon. While the robot is moving its head, a ball is moving, once in the same direction of the camera movement, and once in the opposite direction. In figure 2 are shown the different stages of the algorithm while processing the frame 28 of the first video sequence.

The here-proposed motion detection algorithm can be enhanced using additional object information, such as color when detecting moving balls. Thus, in the *Rejection Filter* module was implemented a ball color filter applied to the blobs. This color filter uses the average U-V values (YUV color space) from each blob for filtering. If the Euclidean distance between the U-V average value of a blob and the ball U-V value (model) is larger than a threshold *Tc*, then the blob is discard. This filter decreases significantly the number of false positives errors. It should be stressed that this filter can be applied only after blobs have been already detected.

The first/second video sequence was 33/37 frames long. 11/14 frames contain a moving ball, but the first appearance of the ball cannot be detected because there is no way to know if the ball is moving or if it is stopped. Thus the relevant information are only 9/12 frames with moving ball, 6/9 of them were successfully detected, which correspond to a successful detections rate of 67%/75%. In table 1 are shown some statistics of the analysis of these video sequences, using and not using the color filter.

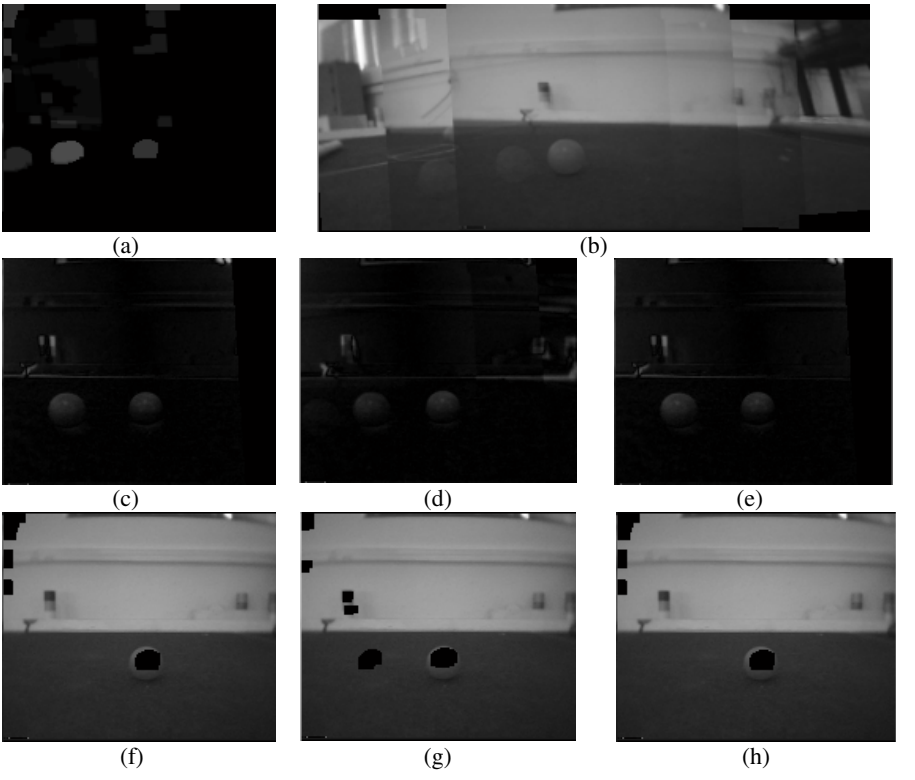

Fig. 2. Process stages at frame 28 in video sequence 1. (a) Motion history representation M_{Hk}. (b) Background model B_k^*. (c) Motion history error image. (d) Background subtraction error. (e) Current Image I_k. (f) Motion history detection D_{Hk} in black overlapped to current image. (g) Background subtraction detection D_{Bk} in black overlapped to current image. (h) Final detections DET_k*, generated by the Kalman Filter

Table 1. Analysis of detections in the video sequence 1 and 2

Sequence number	1		2	
Number of frames	33		37	
Frames with a moving ball present	11		14	
Ball color filter	Off	On	Off	On
Frames with successful moving ball detection	6 (55%)	6 (55%)	9 (64%)	6 (43%)
Frames with a moving ball present but not detected	5 (45%)	5 (45%)	5 (36%)	8 (57%)
Detections corresponding to moving balls	7	6	9	6
Detections corresponding to ghosts	9	0	7	0
Detections corresponding to other moving objects	10	0	8	0
Fake detections (excluding ghosts)	296	2	265	5
Total number of detections	322	8	289	11
False detections average by frame, excluding ghosts	8,97	0,24	7.16	0,14

5 Conclusions

Results of the motion detection and tracking of objects in real-world video sequences using the proposed approach were shown. The system operates in real-time and the relevant moving objects, the ball in this case, are detected and tracked.

In a future work we will extend our system by using also body displacement. This extension would consider additional image displacements and rotations based on the robot joint angles. We will also share the ball tracking information between different robots by implementation a cooperative tracking algorithm. Another feature of the system to be improved is the high amount of false detections. We are working on a heuristic for reducing this kind of detections, beyond the use of a simple color filter.

References

1. S. Baker and I. Matthews, Equivalence and Efficiency of Image Alignment Algorithms, *Proc. of the 2001 IEEE Conference on Computer Vision and Pattern Recognition*, 2001.
2. S. Baker and I. Matthews, *Lucas-Kanade 20 years on: A unifying framework: Part* 1, Technical Report CMU-RI-TR-02-16, Carnegie Mellon University, Robotics Institute, 2002.
3. R. Cucchiara, C. Grana and A. Prati, Detecting Moving Objects and their Shadows - An evaluation with the PETS2002 Dataset, *Proc. of the 3rd IEEE Int. Workshop on PETS*, 18-25, Copenhagen, June 2002.
4. J. Ferryman (Ed.), *Proc. of the 3rd IEEE Int. Workshop on PETS*, Copenhagen, Denmark, June 2002.
5. J. Ferryman (Ed.), *Proc. of the Joint IEEE Int. Workshop on VS-PETS*, Nice, France, October 2003.
6. T. Ishikawa, I. Matthews and S. Baker, *Efficient Image Alignment with Outlier Rejection*, Technical Report CMU-RI-TR-02-27, Carnegie Mellon University, Robotics Institute, October 2002.
7. B. Lucas and T. Kanade, An Iterative image registration technique with an application to stereo vision, *Proc. of the Int. Joint Conf. on Artificial Intelligence*, 1981.
8. J. Piater and J. Crowley, Multi-Modal Tracking of Interacting Targets using Gaussian Approximations, *Proc. of the 2nd IEEE Int. Workshop on PETS*, Hawaii, USA, Dec. 2001.
9. S. Gong, S. McKenna, A. Psarrou, *Dynamic Vision From Images to Face Recognition*, 2000.
10. J. Russ, The Image Processing Handbook, Second Edition, IEEE Press, 1995.
11. J. Ruiz-del-Solar, P. Vallejos, J. Zagal, R. Lastra, G. Castro, C. Gortaris, I. Sarmiento, UChile1 2004 Team Description Paper, *Proc. of the 2004 RoboCup Symposium*.