Evaluation of Real Time Tracking Methods for an Open Source Motion Capture System

Daniel Pacheco de Queiroz*, João Victor Boechat Gomide† and Arnaldo de Albuquerque Araujo‡
*Universidade Federal de Minas Gerais - DCC - UFMG
Belo Horizonte - MG - Brazil
Email: danielpqp@gmail.com
†Universidade FUMEC
Belo Horizonte - MG - Brazil
Email: jvictor@fumec.br
‡Universidade Federal de Minas Gerais - DCC - UFMG
Belo Horizonte - MG - Brazil
Email: arnaldo@dcc.ufmg.br

***Abstract*—This paper presents new results for the open source optical motion capture (mocap) system that is being constructed and is in its final stage of development. The open mocap system at the present moment realizes the entire pipeline for the real time data acquisition of the three dimension positions of markers that are moving in time. These data are organized with a semantics that obeys the way virtual skeletons are constructed, with the markers corresponding to specific articulate joints in the skeleton. In this work, three different methods to track the marker positions are tested and compared in the software. The scope of this comparison is to determine the method that better combines a low computational cost and a high precision in the real time tracking of the markers. These methods predict the position of the markers in the next frame and find them there, maintaining the semantics and reducing the search area. The results are valid for any motion capture system and are very important for the software that is being developed, OpenMoCap.**

***Keywords*-mocap;tracking;**

[1]

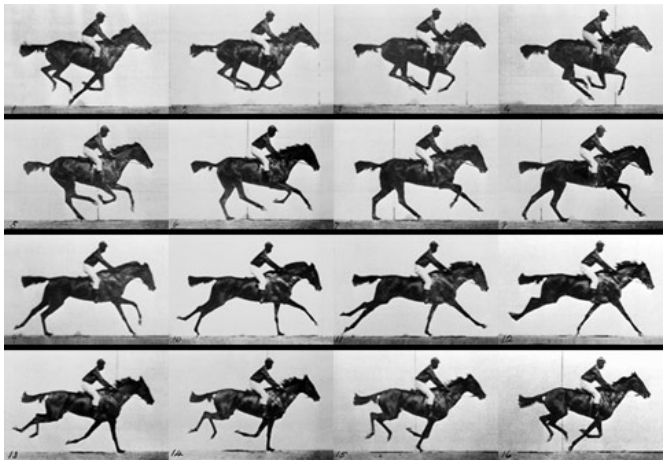

Fig. 1. Eadward Muybridge's 1878 experiment to investigate if all four of a horse's hooves left the ground at the same time during the gallop

I. Introduction

Digital motion capture of living beings and objects has important applications in character animation, biomechanics research, athletics performance improvement, augmented reality, among many others. Unfortunately, the available robust and efficient equipments are very expensive and the source codes of the applications are proprietary and not easy to be modified or adapted for specific tasks.

In face of this reality, the open source real time optical motion capture software was developed in the last two years and is now working and being improved and tested [1] [2]. The software, *OpenMoCap*, performs all the workflow to output motion data in the appropriate format for the existing 3D modeling software.

The whole process of capturing motion can be divided in basically four steps [3]. The first one, *initialization*, is done only in the beginning of the process and relates special given points from a scene with points from a previous defined structure. These points are called points of interest (POIs). The second task is called *tracking*, that is, it monitors the POIs over a period of time. The third one is *reconstruction* or *pose estimation*. And the last one is *output* and consists in outputting data in some special format. To perform these steps, *OpenMoCap* was developed in a modular and multithread structure. The separation of modules by threads was done to take advantage of the tendency of modern processors to have more cores. Further, tasks executed in real time such as POI detection, tracking, triangulation and visualization could be made parallel.

The concept of abstract modules is very important to ensure the software extensibility, one of the goals of this work. If a new camera is to be used and it is not compatible with the current implementation, just some specific methods need to be implemented, following the pattern of the abstract camera module. Another beneficial example is the possibility to substitute the POI detection algorithm with one based on body parts recognition instead of regions intensity. In other

[1]This text is based in a Master Thesis

words, transform the system into one without markers.

Given the nature of the built application, an efficient and mature computer vision library was specially usefull for its construction, OpenCV. [4] made it to demonstrate its processors performance. Unfortunately, the existing modules that support cameras in OpenCV are not robust for more than two of those devices and were not developed with object-oriented concepts in mind. Since the software was designed to grow and solidly support multiple cameras, videoInput library [5] was integrated. Theoretically, it supports up to 20 cameras and is compatible with every video input device that provides a DirectShow interface [6].

In this work, the performance of three well know algorithms for tracking was evaluated. The scope of this comparison is to determine the method that better combines a low computational cost and a high precision in the real time tracking of the markers. In order to do this, each tracker was executed in failure situations, as occlusion and semantics marker swapping.

With *OpenMoCap*, markers are used as the points of interest (POIs). The markers can be active, emitting light, or passive, reflecting light. To capture the marker positions, visible or infrared cameras are used. Knowing the positions of the cameras and the markers in each camera, with the appropriate semantics, then the triangulation is possible and the POIs 3D coordinates are obtained.

The most important tracker function is to maintain the marker semantics frame to frame. This semantics consists of a label associated to the marker in the initialization phase and must be maintained along all the capture process. This information is used in the pose estimation step to do the triangulation. *OpenMoCap* was constructed with the Alpha-Beta filter at the beginning. But as it is modular and versatile, another choice of filter can be easily implemented in the software.

This paper is organized as follows. In the next section related works are introduced, including approaches for optical motion capture and trackers. The applied methods are described in Section III. The experiments for trackers evaluation and their results are presented in Section IV. Finally, the conclusions are discussed in Section V.

II. Related Work

In the broad survey of computer vision-based human motion capture made by [7] [3], it is observed that the Kalman filter and the CONDENSATION (**Con**ditional **Dens**ity Propag**ation**) algorithm are the most used methods to track markers. Both are stochastic and the Kalman filter is based on a Markov chain built on linear operators perturbed by Gaussian noise [8]. The CONDENSATION algorithm uses nonlinear movement models, allowing its application in more complex cases [9].

Other Kalman filter variations are present in the literature. The Extended Kalman Filter [10] [11] works with nonlinear equations, being necessary a linearization of the currently estimation using Taylor series. Due to the high computational cost of this variation, another form (also nonlinear) was proposed and is called Unscented Kalman Filter. With this method, the linearization is not necessary, this is possible with the use of a deterministic sampling technique known as unscented transform [12] [13] [14].

OpenMoCap was first implemented using the Alpha-Beta filter [1]. This algorithm prizes for its simplicity, low computational cost and efficiency. In this filter the next position is predicted by the velocity calculated in the prior state, considering the movement as rectilinear and uniform in each state. This previewed velocity suffers a smoothing defined by the alpha and beta parameters, which were obtained empirically. The other two methods are implemented in *OpenMoCap* and evaluated.

In [15] is presented an approach for tracking markers to human movement analysis. In this work two tracking methods were evaluated, the Kalman filter aforementioned and the extrapolation. The latter is based in an extrapolation function for each coordinate, whose parameters were experimentally defined. Using as input the values obtained in the prior frames (in this case, three frames), it is possible to calculate a predicted position of the marker in the fourth frame. Nevertheless, the experiments showed that the Kalman filter was superior to the extrapolation.

In [16] a system for tracking and counting fishes is presented. Again, the Kalman filter and CONDENSATION are cited, but the author used another tracking method called BraMBLe (A Bayesian Multiple-Blob Tracker), considering this method more befitting with the problem. This method is based on CONDENSATION, in conjunction with a probabilistic observation model capable to separate the background and the interest objects. However, the *OpenMoCap* done the segmentation between the background and the interest objects in another step of the process. This way, it is not necessary to use the BraMBLe's training phase to do this segmentation. Hence, only the tracking task accomplished by the CONDENSATION method is necessary to this work.

III. Evaluated Methods

A. Alpha-Beta

The Alpha-Beta tracking filter is a single target tracking widely used in radar systems. The most important propriety of this tracker is its simplicity, which provides a low computational cost allowing real time running [17]. The recursive filter is compound by two states, with one of them being obtained by the integration of the other over time. The first state variable is the position x_p and the second is the velocity v_p. In this way, the position can be obtained by the integration of the velocity over time. To do this, the velocity is taken as constant in a small time interval Δt.

Due to noise in the acquisition of the images and in the motion model, almost never the predicted position x_p is the same of the measured x_m. To represent this difference, it is used the residual error r, which is calculated by the difference between x_m and x_p in a given time k (Equation 1). This error describes the impact of the perturbations occurred in the scene and the trajectory of the observed object. This value, together

with the constants α and β, is used as show the equations (2) and (3) to control the intensity of smoothing, obtaining the smoothed position x_s and de smoothed velocity v_s.

$$r(k) = x_m(k) - x_p(k) \tag{1}$$

$$x_s(k) = x_p(k) + \alpha r(k) \tag{2}$$

$$v_s(k) = v_p(k) + \frac{\beta}{\Delta t} r(k) \tag{3}$$

The constants α and β are used to control the intensity of smoothing. As bigger as their values are, faster is the response of the tracker to abrupt changes in the trajectory. Contrariwise, as small as their values, less the noise will prejudice the prediction. Hence, the definition of the value of these constants is related with the compromise between the smooth of the noise and fast responses to changes in the trajectory. Then, this values need to be experimental defined and varies for each application.

B. Kalman filter

The Kalman filter is a recursive technique that uses a set of mathematical equations to predict the process state. Setting some initial values, the process state in each step can be predicted. The algorithm can automatically adjust the parameters of the model using the measurements of each step, obtaining an error estimation in each updating. The possibility to incorporate error effects and its computational structure gave to the Kalman filter a wide field of applications in different areas, such as robotics or computer vision.

The discrete Kalman filter algorithm can be separated in two parts. In the first one the algorithm predicts the new process state. After that, in the second part, it corrects that prediction using the feedback of the prediction. In this way, the equations for the Kalman filter can be separated into two groups: time update (or predictor) equations and measurement update (corrector) equations. The first group is responsible for projecting forward, in time, the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback, i.e., for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

The time update equations are present below.

$$\hat{x}_{k+1}^{-} = A_k \hat{x}_k \tag{4}$$

$$P_{k+1}^{-} = A_k P_k A_k' + Q_k \tag{5}$$

In the equation (4) the variable $\hat{x}^{-}$ is the *a priori* estimated process state vector. A example of state vector is $\hat{x}' = \{x, v_x, y, v_y\}$, where x and y are the coordinates of a marker and v_x and v_y the velocities in each axes. The transition matrix A is defined by the model of motion, for example, a rectilinear movement where $x = \Delta t v_x$.

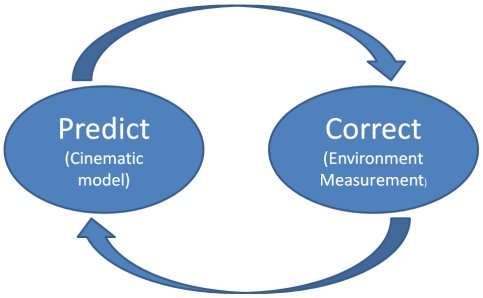

Fig. 2. The Kalman filter cycle. The *time update* project the state ahead in time, then the *measurement update* corrects the last prediction by the actual measurement

In the equation (5) P_{k+1}^{-} is *a priori* estimate error covariance matrix and P_k is *a posteriori* estimate error covariance matrix. The covariance error matrix Q_k represents the noise (or the error) of the system that allows to adjust the values of P_k. To obtain a variable matrix P_k, used to generate a variable area of search, we need to vary the matrix Q_k, for example, basing in the velocity of the markers.

The measurement update equations are present below.

$$K_k = P_k^{-} H_k' (H_k P_k^{-} H_k' + R_k)^{-1} \tag{6}$$

$$\hat{x}_k = \hat{x}_k^{-} + K_k (z_k - H_k \hat{x}_k^{-}) \tag{7}$$

$$P_k = (I - K_k H_k) P_k^{-} \tag{8}$$

The first task during the measurement update is to compute the Kalman gain K (6). In this equation the matrix H_k relates the state to the measurement and R_k is the measurement error covariance matrix. The next step is do the measures to obtain z_k (the measurement vector), and then generate an *a posteriori* state estimate by incorporating the measurement as in (7). The last step is to obtain an *a posteriori* error covariance estimate through (8).

C. CONDENSATION

Like the Kalman filter, the CONDENSATION algorithm propagates a probability density function (*pdf*) in time [9]. Nevertheless, this algorithm does not restrict the shape of this distribution. Thus, the function curve can have multiples local maxima, being multimodal and nonlinear, which increases the cost of their propagation in time. In order to accomplish this operation in a feasible time, it is necessary to do an approximation. This is the purpose of the particles in this kind of filters. In the CONDENSATION algorithm, this technique is called factored sampling and is described as following.

In the factored sampling technique, the *pdf* is represented by a set of N particles or samples, represented by $\{s_1, \ldots, s_N\}$. Therefore, this approximation of the *pdf* increases the accuracy as N increases. Each particle stores a possibility of the system state. To measure this, it is assigned to each particle a probability of the state stored in the particle being the true

state. This value is represented by π_n, with $n \in \{1, \ldots, N\}$ and measures the confidence of the particle.

The CONDENSATION algorithm is based on factored sampling extended to work iteratively to successive images in a sequence. In the execution beginning, all the particles are randomly spread over all the search area. As we has no information about each particle to discriminate them, all particles receive the same confidence value, given by $\frac{1}{N}$.

After the definition of initial step, the others can be explained. In a brief definition, the next steps consist in sample the particles in each time instant. In spite of the re-sampling, the particle-set still have the same size N in all steps, so that the algorithm can be guaranteed to run within a given computational resource.

The particle sampling could be separated in three phases: **selection**, when the particles that will be sample are chosen based in the confidence value, in the way that the particles with greater value have more chances to be choose; **prediction**, the particles undergoes a drift and, since this is deterministic, identical elements in the new set undergo the same drift, after that, the particles undergoes a diffusion by adding a noise in their position; the last phase, **observation**, calculates the particles' confidence, based in the real marker position, in the way that closer particles obtain greater confidence values.

IV. Experiments

Initially, the values of the parameters of the methods are determined. These values and settings were chosen because they showed better results, or because they have a good relationship between execution time and tracking quality. For the Alpha-Beta filter, the parameter alpha was set with the value 1.0 and beta received 0.8.

For the Kalman filter, it is important to report that the matrix Q (which contains the errors of speed) is updated as the errors are larger as the speed is greater. Thus, the search area increases with the rise of markers' speed. The state vector stores the position variables x and y, and their respective speeds.

The CONDENSATION algorithm was executed with 100 particles. The number of particles is directly proportional to the execution time of the method. This value causes the method to have an execution time consistent with the other methods, without affecting much the accuracy of the predictions. The state vector stores only the position variables x and y, making the method completely probabilistic and nonlinear.

Experiments were performed to measure the robustness of the tracking and the execution time for each method. To measure the robustness, videos at 30fps with a resolution of 640 by 480 pixels were generated, which simulate real situations prone to failures. These situations were divided into three categories: nonlinear motion, occlusion and semantics markers swap (change of the names of two markers). The loss of the markers was registered during the tracking and in the tests of nonlinear motion. In cases which no loss has been registered, the distance between the predicted point and measured point was calculated.

A. Tracking quality

In the tests of nonlinear motion, a marker performed zigzags in various directions and motions with acceleration. These motions were separated into slow, fast and with acceleration. In the experiments of occlusion, a barrier of variable size was positioned on the trajectory of the marker, which could be linear or not. Twenty videos were generated for each test situation, and the number of losses of the marker was counted for each method. The Figures 3, 4 and 5 show some videos's frames examples and the Table I shows the results.

TABLE I
NUMBER OF LOSSES.

	Alpha-Beta	*Kalman*	*CONDEN.*
Slow	3	0	0
Rapid	9	0	4
Accleration	4	1	2
Occlusion	7	6	8
Marker Swapping	9	11	16

As each method made a correct tracking in at least 11 videos, and each of them have about 80 frames. A number close to or exceeding 1000 error values was obtained, by method. This information is important, because applying the Shapiro-Wilk test, the result was negative, i.e., the error values do not follow a normal distribution. Despite this, by the Central Limit Theorem, due to the large number of values, statistical moment analysis can be applied. Results are presented separately for the videos with slow, fast, and acceleration.

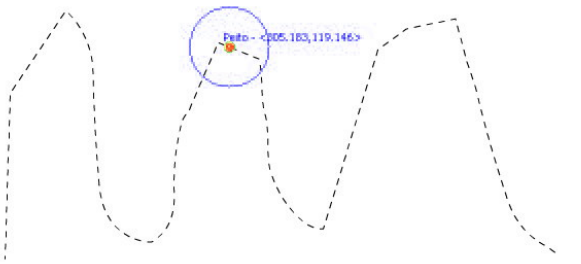

Fig. 3. A frame of a video used in the robustness tests, a zigzag rapid movement. The dashed line represents the trajectory of the bullet, the red dot represents the estimated position of the marker (over the marker) and the blue circle represents the search area.

Tables II, III and IV, in a confidence interval of 99%, show for each method the number of obtained samples, the lower limit of errors, the mean and the upper limit of the values.

B. Computational Cost

The test to measure the computational cost of each algorithm was done by measuring the time required to process

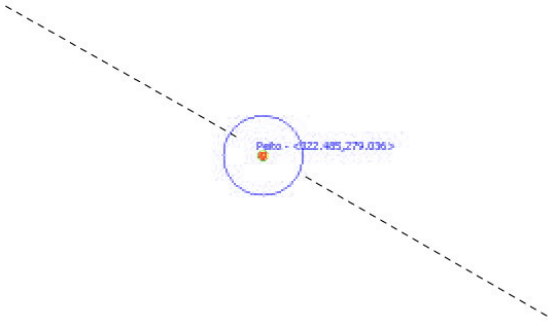

Fig. 4. A frame of a video used in the occlusion tests. In this frame the marker is passing through the area of occlusion, then only the predicted position is displayed.

Fig. 5. A frame of a video used in the Marker Swapping tests. In the bottom is the tracking marker with the label "Peito", covered by predicted red point. In the top is a yellow point with the label NULL.

each video frame. This measurement considers only the task of tracking the marker position, excluding tasks as the image processing, detection of the POI, etc. Although the execution time is extremely dependent on the hardware employed, the goal was to make a comparison between the costs of each method. Therefore, these comparisons are valid, provided that the same conditions are maintained for all algorithms, as was done in this work.

In this experiment the markers follow simple trajectories, because the goal was not to test the robustness of the tracking, previously measured. To measure the performance of algorithms with different numbers of markers, videos with 1, 5, 10, 15, 20, 25 and 30 markers were generated. The time measurement was done for 200 frames and the results are showed below.

TABLE II
SLOW MOVIMENTS.

Method	*Samples*	*L. Lim.*	*Mean*	*U. Lim.*
Alpha-Beta	1445	6,67	7,21	7,75
Kalman	1700	9,71	10,31	10,91
CONDEN.	1700	38,51	39,18	39,83

TABLE III
RAPID MOVIMENTS.

Method	*Samples*	*L. Lim.*	*Mean*	*U. Lim.*
Alpha-Beta	594	9,77	10,81	11,84
Kalman	1080	14,03	14,98	15,93
CONDEN.	864	38,77	40,02	41,27

V. CONCLUSIONS

Analyzing the tables with the values of prediction error of the methods, a considerable difference between values of each method is observed. The Alfa-Beta and Kalman filters results are more similar, showing the lowest error values. Otherwise their mean values do not fall within the range determined by lower and upper limits of each other. The CONDENSATION algorithm presents the highest error values in all the cases.

An important remark about these results is that smaller prediction errors are not consistent with a greater number of hits in the videos. The Alpha-Beta filter, which showed the lowest prediction error, was always the method that missed at most the markers in the videos. With this information, it can be concluded that a very accurate prediction, or with a high confidence in the previous system state, is not the best approach. It seems that a portion of diffidence in the prediction is necessary, using predictions with a defined amount of uncertainty. This is calculated by the error matrices in the Kalman filter and the confidence of the particles in the CONDENSATION method. They provided better total results, but the Kalman filter gives the best overall results.

Another factor that improves the results of the Kalman filter and the CONDENSATION algorithm is the search area of variable size. In situations that are more error-prone, if the search area in these methods increases with the growth of the uncertainty, or the speed, the results are more precise.

Analyzing the tables with the computational costs, the longer processing time was 2.812 ms, obtained by the algorithm CONDENSATION. The difference between the values obtained using the other methods looks great, but this value corresponds to only 8.44% of the 33 ms time window available for process each frame in a real time application (30 fps). Although the ratio is small, other stages of the motion capture should also be performed in this available time. Based on work [18], it is known that the average processing time of the other steps of the capture process is no more than 20 ms, which allows to spend up to 13 ms with the tracking. Therefore, even considering the higher peak time obtained, it would still be possible a real-time tracking at a rate of 30 frames per second.

From these observations, a good solution for a tracker is to combine the simplicity of the Alpha-Beta filter with a variable

TABLE IV
ACCELERATION MOVIMENTS.

Method	*Samples*	*L. Lim.*	*Mean*	*U. Lim.*
Alpha-Beta	1360	2,86	3,17	3,48
Kalman	1615	5,50	5,98	6,47
CONDEN.	1530	21,82	22,67	23,51

TABLE V
ALPHA-BETA. FOR EACH NUMBER OF MARKERS, THE TABLE SHOWS THE 5^o PERCENTILE, THE MEDIAN, THE 95^o PERCENTILE AND THE MAXIMUM VALUE AMONG THE PROCESSING TIMES OBTAINED (IN MS).

Markers	*5º Percentile*	*Median*	*95º Percentile*	*Maximum*
1	0,043	0,046	0,053	0,077
5	0,064	0,067	0,084	0,104
10	0,086	0,091	0,110	0,116
15	0,109	0,113	0,126	0,218
20	0,130	0,140	0,155	0,221
25	0,136	0,164	0,184	0,211
30	0,168	0,194	0,214	0,274

search area, based on speed at the beginning. At the present moment, the Kalman filter is used to implement the tracker module of *OpenMoCap*.

ACKNOWLEDGEMENTS

The authors are grateful to CNPq, CAPES and FAPEMIG, Brazilian research funding agencies, for the financial support to this work.

TABLE VI
KALMAN FILTER.

Markers	*5º Percentile*	*Median*	*95º Percentile*	*Maximum*
1	0,067	0,077	0,100	0,126
5	0,120	0,130	0,148	0,174
10	0,172	0,175	0,195	0,268
15	0,226	0,234	0,256	0,331
20	0,293	0,296	0,314	0,386
25	0,338	0,356	0,376	0,453
30	0,409	0,418	0,445	0,503

TABLE VII
CONDENSATION.

Markers	*5º Percentile*	*Median*	*95º Percentile*	*Maximum*
1	0,134	0,146	0,162	0,180
5	0,387	0,499	0,529	0,801
10	0,824	0,950	1,017	1,095
15	0,958	1,384	1,423	1,604
20	1,285	1,842	1,920	2,116
25	1,560	1,934	2,333	2,427
30	1,886	2,105	2,791	2,812

REFERENCES

[1] D. L. Flam, D. P. de Queiroz, T. L. A. de Souza Ramos, A. de Albuquerque Araujo, and J. V. B. Gomide, "Openmocap: An open source software for optical motion capture," *Games and Digital Entertainment, Brazilian Symposium on*, vol. 0, pp. 151–161, 2009.

[2] J. V. B. Gomide, D. L. Flam, D. P. de Queiroz, and A. de Albuquerque Araujo, "An open source motion capture system and its applications in arts and communication," *World Congress on Communication and Arts - WCCA*, 2010.

[3] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Comput. Vis. Image Underst.*, vol. 104, no. 2, pp. 90–126, 2006.

[4] Intel Corporation, "Laptop, notebook, desktop, server and embedded processor technology - intel," Dec. 2010, this is an eletronic document available at: http://www.intel.com. Last visited on: December 9, 2010.

[5] T. Watson, "videoInput Library," Dec. 2010, this is an eletronic document available at: http://muonics.net/school/spring05/videoInput. Last visited on: December 1, 2010.

[6] Microsoft Corporation, "Microsoft corporation," Dec. 2010, this is an eletronic document available at: http://www.microsoft.com. Last visited on: December 9, 2010.

[7] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Comput. Vis. Image Underst.*, vol. 81, no. 3, pp. 231–268, 2001.

[8] Kalman, Rudolph, and Emil, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[9] M. Isard and A. Blake, "Condensation-conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, August 1998. [Online]. Available: http://dx.doi.org/10.1023/A:1008078328650

[10] G. Welch and G. Bishop, "An introduction to the kalman filter," Department of Computer Science - University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, Tech. Rep., 1995.

[11] S. Julier and J. K. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," Robotics Research Group, Department of Engineering Science - University of Oxford, Tech. Rep., 1996.

[12] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *Proc. SPIE Vol. 3068, Signal Processing, Sensor Fusion, and Target Recognition VI, Ivan Kadar; Ed.*, 1997, pp. 182–193.

[13] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *American Control Conference, 1995. Proceedings of the*, vol. 3, jun. 1995, pp. 1628 –1632 vol.3.

[14] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *Automatic Control, IEEE Transactions on*, vol. 45, no. 3, pp. 477–482, August 2002. [Online]. Available: http://dx.doi.org/10.1109/9.847726

[15] P. J. Figueroa, N. J. Leite, and R. M. L. Barros, "A flexible software for tracking of markers used in human motion analysis," *Computer Methods and Programs in Biomedicine*, vol. 72, no. 2, pp. 155 – 165, 2003, http://dx.doi.org/10.1016/S0169-2607(02)00122-0. [Online]. Available: http://www.sciencedirect.com/science/article/B6T5J-47K39GW-1/2/f6bf2ed1112d9d4d915c1650ca420cb8

[16] E. F. Morais, M. F. M. Campos, F. L. C. Padua, and R. L. Carceroni, "Particle filter-based predictive tracking for robust fish counting," in *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 367–. [Online]. Available: http://portal.acm.org/citation.cfm?id=1114697.1115376

[17] J.-C. Yoo and Y.-S. Kim, "Alpha-beta-tracking index (alpha-beta-lambda) tracking filter," *Signal Processing*, vol. 83, no. 1, pp. 169–180, 2003.

[18] D. L. Flam, "Openmocap: Uma aplicao de cdigo livre para a captura ptica de movimento," Master's thesis, UFMG, 2009.