Dynamic Bayesian Networks for Gesture Recognition

Rafael Cabañas de Paz, M. Julia Flores, Jesús Martínez-Gómez

***Abstract*—In this paper a gesture recognition system using visual information is described. The system relies on probabilistic graphical models and recognizes up to six gestures performed by a human who interacts with a computer. All these gestures can be used to manage a picture viewer. The main novelty of the system is the use of dynamic Bayesian networks and the use of both hands to perform the gestures. The proposed system was evaluated using real video sequences and the results obtained proved the goodness of the proposal.**

***Index Terms*—Computer vision, robotics, probabilistic graphical models, human robot interaction.**

I. Introduction

GESTURE recognition has become one of the most important elements in human-computer interaction. For example, it can be used to create new types of interfaces based on vision [1][2], where the gesture is used to send a command. In fact, the interest on gesture recognition has increased due to the release of new game consoles devices (Sony Eye Toy[1] and Microsoft Kinect[2]) that use the human body as the main interface. Another interesting application is the interaction with robots when vision is the only available channel of communication, for example, in noisy environment [3][4][5].

Computer vision techniques for gesture recognition have to extract punctual information from the separate images acquired by the robot. The relationship between the information extracted from a sequence of consecutive images is used to identify the gesture. For this purpose, several techniques can be use, such as Neural Networks (NNs) [6], Dynamic Time Wrapping (DTW)[7] or Hidden Markov Model (HMM) [8][9]. Our hypothesis is that gesture recognition can be performed using dynamic Bayesian networks. The input for this classifier is the position of the hands that will be tracked for consecutive images. The position of the hands in this proposal is obtained thanks to the use of colour filtering techniques, but other approaches could be applied.

We evaluated the current proposal by developing a gesture recognizer for a picture viewer where six gestures are recognized. Thanks to this system, any user could control the viewer just by using his own hands. Several experiments on real scenarios proved the goodness of the proposal.

The article is organized as follows: Bayesian networks are outlined in Section II. We describe the image processing and tracking in Section III. In Sections IV and V we explain the use of Bayesian networks for tracking and gesture recognition. The experiments and results are shown in Section VI and finally, conclusions and future work are discussed in Section VII.

University of Castilla-La Mancha, Albacete, Spain

[1]http://www.eyetoy.com/

[2]http://www.xbox.com/Kinect/

II. Dynamic and static bayesian networks

Bayesian networks (BNs) are an increasingly popular paradigm for representing problems. A Bayesian network [10][11] is a directed acyclic graph (DAG) whose nodes represent the random variables in the problem. A set of directed edges connect pairs of vertices, representing the direct dependencies (which are often causal connections) between variables. The set of nodes pointing to X are called its parents, and is denoted $pa(X)$. The relationship between variables is quantified by conditional probabilities, usually tables (CPTs), which are associated with each node, namely $P(X|pa(X))$. The CPTs together compactly represent the full joint distribution.

Figure 1 gives an example BN for a small robotics problem [12]. It shows how to model a robotic-based problem, we have chosen a simple application in order to illustrate better the BN elements. The kinematic model describes the relationship between the configuration of the robot, i.e., the joint angles, and the body posture, i.e., the positions of the body parts in space. Fig. 1.(a) shows an example of a simple 2-DOF (Degrees Of Freedom) robotic manipulator. The robot consists of two rotary joints a_1 and a_2, and five body parts $X_1, \ldots, X_5$. The first two body parts are connected rigidly. The shoulder X_2 and the upper arm X_3 are connected by the shoulder joint a_1, and so forth. This is formalised in a BN, Fig. 1.(b).

The graph does not only offer a visualisation of the system, but also, and even more important, attempts to capture (in)dependences in the model. In the simple case (two nodes) an edge indicates a probabilistic dependency, while the absence of an edge indicates probabilistic independency. However, these dependences can be found and express for any pair of non-adjacent nodes, for example by the d-separation concept [11].

The capability of a BN to *express* relationships, dependencies and independencies by its associated graph lies on its qualitative side. But these relationships are also modelled with a second element, quantitative, that forms a BN: probability distributions, as already introduced. Notice that in order to complete the BN definition for 1.(b) we will

need to define those $P(X|pa(X))$, in this particular case: $P(X_1)$, $P(X_i|X_{i-1})\forall i \in [2,5]$, $P(a_1)$, $P(a_2)$, $P(X_3|a_1)$ and $P(X_4|a_2)$

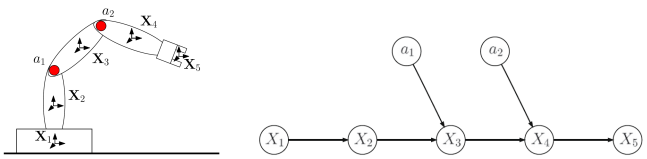

Fig. 1. BN example: (a) The kinematic function of a robotic manipulator constructed by concatenating the individual geometric transformations of each of the joints and links; (b) Its representation as a Bayesian network.

This probabilistic information will be processed by Bayes' Theorem but taking advantage of the inherent partition of the variables' *families* that BNs provide. BNs are primarily used for inference: evidence e in the form of statements on the state of some variables is used to infer the posterior probabilities $P(X|e)$ for all remaining variables. Users can set the values of any combination of nodes in the network that they have observed. This evidence, e, propagates through the network, producing a new posterior probability distribution $P(X|e)$ for each variable in the network. There are a number of efficient exact and approximate inference algorithms for performing this probabilistic updating, providing a powerful combination of predictive, diagnostic and explanatory reasoning. Going back to Fig. 1, one could have observations on all the X_i and would like to infere about a_1 and a_2, since for many robotic applications, it is necessary to compute the required joint angles a_1, a_2 given a target position in the workspace.

BNs framework present the feature of capturing static behaviours. But it is also possible to include Bayesian networks where the environment is inherently dynamic, as a video sequence, which will be our domain. There exist a well-known variant of Bayesian networks called **dynamic Bayesian networks** (DBNs) which can be used to model dynamics in this sequential scenario.

DBNs are a long-established extension [10] to ordinary BNs that allow explicit modelling of changes over time (e.g. [13], [14], [15]). They have been used in a range of applications such as robot navigation and map learning [16], monitoring robot vehicles [17] and traffic monitoring in both [18] and the BATmobile project for monitoring an automated vehicle travelling on a freeway [19].

The general structure of a DBN is shown in Figure 2 (taken from [10]). In a DBN, for each domain variable X_i, there is one node for each time step of interest – X_i^T, X_i^{T+1}, X_i^{T+2}, etc. Each time step is called a time-slice. The relationships between variables at successive time steps are represented by so-called temporal arcs, including relationships between (i) the same variable over time, $X_i^T \rightarrow X_i^{T+1}$, and (ii) different variables over time, $X_i^T \rightarrow X_j^{T+1}$.

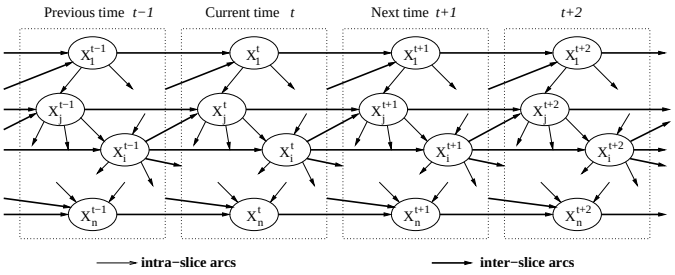

Fig. 2. General structure of a DBN (Figure 4.10 from [20])

Note that in the generic structure there are no arcs that span more than a single time step; this reflects the so-called Markov assumption that the state of the world at a particular time depends only on the previous state and any action taken in it. Given the typical restriction that both the structure and the CPTs are unchanging, a DBN can be specified very compactly.[3]

Hidden Markov Models (HMMs) constitute a simplified case of dynamic Bayesian networks, which are widely used for modeling temporal structures. They have been applied to speech recognition [21], learning and more recently to gesture recognition [22].

In a HMM, the graph has variables $S_1, S_2, \ldots, S_n$ wich represent the state of a dynamic system at time points $1, 2, \ldots, n$ respectively [23]. Moreover, the variables $O_1, O_2, \ldots, O_n$ represent sensors that measure the system state at the corresponding time points. Usually, one has some information about the sensor readings and is interested in computing beliefs in the system state at different time points (See Fig. 3). The independence statement declared by this DAG for state variables is that, once we know the state of the system at the previous time point, $t-1$, our belief in the present system state, at time t, is no longer influenced by any other information about the past. The reason for the term *hidden* is that under the surface (observations made by tests) there is a hidden activity that cannot be observed [11].

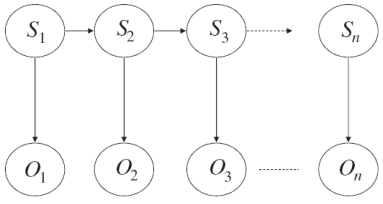

Fig. 3. A directed acyclic graph known as a hidden Markov model (Figure 4.3 from [23])

III. Image Processing and Tracking

For gesture recognition, it would be necessary (at each time-step) to obtain the $< x, y >$ position of the hands. Even

[3]Some BN software packages (e.g. Hugin, GeNIe, Netica) provide a facility to specify a DBN compactly.

though that problem has already been solved with improved sensors such as the Kinect device, we decided to use the most common sensor: a visual camera. The position of the hands is obtained by using two key-coloured balls (shown in Fig. 5) that the human will move. Instead of using these balls, different colours gloves could also have been considered. However, our preliminary experiments obtained better results with the balls.

The use of key-coloured elements was proposed to reduce the complexity of the image processing. For future works, we aim to perform the hand tracking without the use of key-coloured elements. The rest of the section explains our proposal for a robust hand tracking system based on the use of the colour information.

A. Colour segmentation

We used the colour segmentation to identify the pixels that belong to the balls and therefore, to hands. Thanks to this step, we can discard all the other pixels of the image, reducing the computational time. Additionally to this, and taking into account that we have used different colours for each ball, we could also differentiate the two hands.

Instead os using classical RGB colour space, we considered the use of YUV, where each colour has 3 components: a single brightness or luminance component (Y) and two chrominance components (U-blue and V-red). Since brightness information is separated from colour information, this colour space is more robust to changes in the illumination.

In order to check if a pixel belongs to a specific colour, the set of comparisons shown in equation 1 are performed:

$$\begin{array}{lcl} y < Ymax & \wedge & y > Ymin \\ u < Umax & \wedge & u > Umin \\ v < Vmax & \wedge & v > Vmin \end{array} \quad (1)$$

Within Eq. 1 Y, U and V are the components for the colour. $Ymax$, $Umax$ and $Vmax$ are the upper limits and $Ymin$, $Umin$ and $Vmin$ the lower limits. However, colours cannot always be distinguished properly in achromatic areas. Therefore, the method of *Line-based Region Growing* [24] was used: if a pixel does not pass the comparisons but the pixel on the left does, it will be considered as a pixel belonging to the specific colour. The advantage of this method is that is efficient in computational terms. The result is a binary image $C(x, y)$ defined for each pixel $< x, y >$ as follows:

$$C(x,y) = \begin{cases} 1 & \text{if } x \in \text{colour} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The main problem of performing all the comparisons shown in (1) for all the pixels is the computational time efficiency. We decided to use a LUT (Look-up Table) [25] where the result of (2) is stored for each possible pixel in the space colour. As the table is initialized just once (at the beginning) applying the colour detection for a pixel can be performed just by reading the table. In our tracking system, we need to perform a green and orange colour filtering as such shown in Fig. 4 top. The values of limits used for the filtering were determined experimentally.

Fig. 4. Colour filtering: Original image (left), green (center) and orange filter (right)

B. Motion detection

Any gesture performed with the hands involves a variation on the position of the hands. Therefore, we can apply a motion filter to separate the hands from the irrelevant background. In order to achieve this goal, we propose to apply a method named *Joint Difference* [26]. It combines the method image difference with a background substraction in order to avoid the ghosting problem. The result is a binary image $M(x, y)$ defined for each pixel $< x, y >$. In order to remove noise, we can apply a smoothing filter before the motion filter. As done for the colour detection, we can consider the idea of using a LUT. Now, the table is indexed by the value of the pixel in the frame i^{th} and in the frame $i - 1^{th}$.

C. Combination of colour and motion information

As it was mentioned before, the objective of this tracking system is to know the position of the balls (hands). We can obtain such position by combining the colour and the motion information. The result of combining these two sources of information will be a new image:

$$R(x,y) = C(x,y) \wedge M(x,y) \quad (3)$$

Thanks to this reasoning, the objects of the background with a colour similar to the balls will not pass the motion filter. This can be observed in Fig. 5, where some noisy pixels that pass the colour filtering are removed from the final pixel distribution by means of colour filtering.

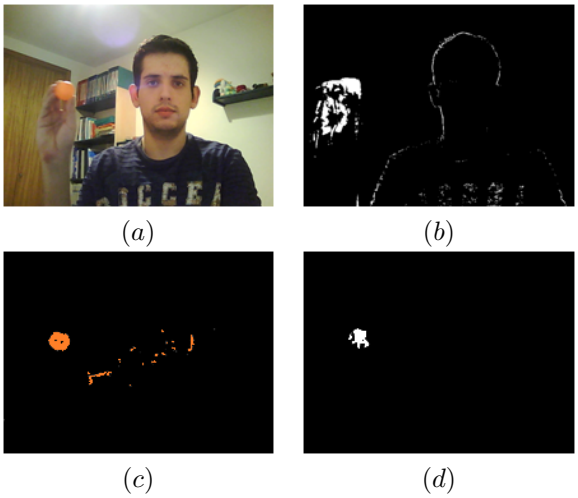

Fig. 5. Combination of colour and motion filters: (a) Input image. (b) Image obtained with the motion filter. (c) Image obtained by the colour filter. (d) Result image obtained after combining both filters.

D. Region of Interest

It can be assumed that no big variations for the position of the balls in consecutive frames are expected. Thus we could apply the filters only for pixels that are near the ball position detected for the previous/last frame. That set of pixels establishes the ROI (Region of Interest). Using the ROI, the tracking system can be faster and more robust. When the position of the ball is not detected, the filters will be applied to the whole image.

E. Outliers

The outliers or false positives can be discarded by setting a minimum number of pixels that must be detected to consider a prediction as reliable. This value must be selected considering the distance between the user and the camera together with the size of the balls (hands).

IV. Bayesian Networks for tracking

Hidden Markov Models (HMM) can be used in computer vision to estimate the position of an object at time t. This estimation will be based on the previous position at time $t-1$ and a set of observations. An example is the particle filter or the condensation algorithm [27][28]. However, this method requires using sampling methods that are not efficient at all. In this section we propose a method that uses independent HMMs and Gaussians to estimate the position for each ball.

A. Two different HMMs for each ball

The aim of HMMs is to avoid abrupt changes in the estimation of the ball position. In order to simplify the problem, two independent HMMs are used: one for each coordinate (Fig. 6). Since two balls have to be tracked, four independent HMMs are needed.

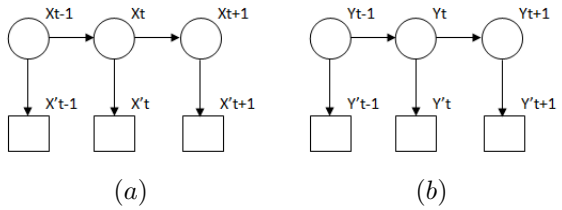

Fig. 6. (a) HMM for coordinate X. (b) HMM for coordinate Y

X_t denotes the position in the x-axis of the object at the time step t. X'_t denotes the observation at time step t used to estimate X_t. In this problem, the observations used as input for the model are the $< x, y >$ positions of the hands presented in the previous section.

B. Inference

Instead of using probability tables, we are going to use a single Gaussian to store the probability of a random variable. This Gaussian will be described by its mean and by its variance. Let be $X_{t-1} \sim N(\mu_{t-1}, \Sigma_{t-1})$ a random variable and $X'_t \sim N(\mu'_t, \Sigma'_t)$ an observation, the value of a node X_t is calculated as follows:

1) Calculate D, the weighted mean of k last transitions.
2) Add D to the mean of previous node: $X_{t-1} \sim N(\mu_{t-1} + D, \Sigma_{t-1})$
3) Compute the value of current node: $X_t \sim X_{t-1} \cdot X'_t$
4) Multiply by a constant α the variance of X_t

The prediction of the coordinate will be the average of the resulting Gaussian distribution. The multiplication by a constant α was carried out to avoid getting a value of the variance close to 0 when $t \to \infty$. We used $\alpha = 2$ in all our experiments. We remark that this is a specific method of performing probabilistic inference as explained in Section II. The value of μ'_t and Σ'_t was obtained from the centroid and variance of all the pixels that passed the motion and color filters.

C. Robustness

The use of HMMs will provide robustness to the tracking procedure: the accuracy of the system will not decrease too extremely when facing noisy images. Since the position of the hands is estimated using two sources of information (observation and last estimated positions), the dependency of the system on the visual information is reduced. Despite the robustness of the tracking system is improved, its accuracy will decrease if a large sequence of low quality or noisy frames is acquired. The goodness of this system has been tested in different scenarios, comparing the result of the tracking with and without the HMMs (see section VI-A).

V. Bayesian Networks for gesture recognition

This section proposes the use of dynamic Bayesian networks for gesture recognition. The input of the model will be the information generated by the tracking system. This information is the average variation obtained for the position of the left and right hands during the last frames.

A. Set of gestures

There are 6 gestures (see Fig. 7) that are considered to be recognized. These gestures were proposed to use the human body as the main interface for a picture viewer program. Just by using his/her hands and our gesture recognizer, a human could manage a picture viewer and perform the most common actions such as showing the next picture or rotating the current picture. All the gestures can be split into 3 groups: gestures for changing the picture that is being shown (*next* and *back*), gestures for rotating the current picture (*rotate ACW* and *rotate CW*) and those for changing the size of the current picture (*zoom in* and *zoom out*).

B. Structure of DBNs

In order to identify the gesture that is being performed, 3 independent DBNs were used (*change*, *rotate* and *zoom*). These DBNs are based in the well-known clasiffier Naive Bayes [29] where each class node is connected to the class node of the following time-slice.

Fig. 7. Set of gestures to be recognized

Each class variable has 3 possible values: two gestures and the value *none*, which means that none of the gestures of the group is made. Therefore, the change-DBN will classify into *next*, *back* and *none*; the rotate DBN into *rotate ACW*, *rotate CW* and *none*; the zoom DBN into *zoom in*, *zoom out* and *none*. These DBNs are shown in Fig. 8.

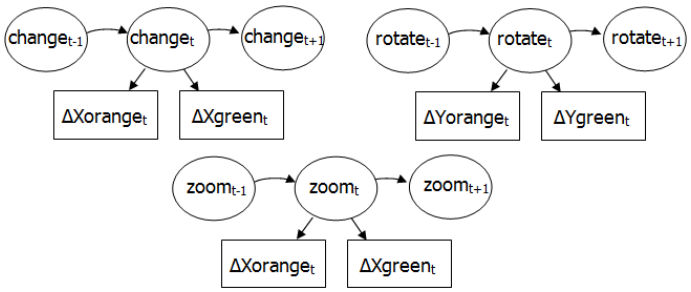

Fig. 8. DBNs used for gesture recognition: (top) Change. (middle) Rotate. (bottom) Zoom.

The inputs (known nodes) of the DBNs are the variation of the position of each ball. In the case of rotate-DBN, only Y component is considered. Meanwhile, for the zoom and change DBNs only the X component is considered. In order to improve the recognition, the variation of the latest k frames can be calculated. Notice that our set of gestures (commands) do not imply movements in Z component.

Since the variation of the position is a continuous variable, it must be discretized. In our system, these variables can take 3 values: *negative*, *positive* and *null* variation. The intervals were chosen experimentally.

C. Learning

In order to determine the conditional probability tables (CPTs) for the variables in the DBN, we have performed automatic learning. To do so, 18 training videos were generated (3 for each gesture). At each video, only a single gesture was made. Using the tracking procedure described in section III we obtained the variation of the position of each ball at every frame. We discretized all data and used the *Weka 3.6* tool[4] to get the conditional probabilities. Since this software does not work with DBNs, the conditional probabilities have been obtained for the corresponding static Naive Bayes. Finally the conditional probability between two class nodes were determined experimentally so that higher probabilities were given to the case where the prediction maintains the same, instead of changing with respect to the previous one.

D. Post-processing

Inference process will provide us a probability for every interest node in the DBNs. That is, after making inference, nine probabilities values will be at our disposal, corresponding to the three children nodes for each DBN. However, the result of the gesture recognizer must be a single prediction: the gesture the human is performing, which can be *none*.

We assume that the human is doing the gesture that obtained the probability p_i if the two following conditions are satisfied:

- $p_i > T1$
- $p_i > T2 * \sum_{j=1}^{3} p_j$

The first condition avoids recognizing gestures with low probability. The second one follows a conservative approach and is used to abstain from a decision when two or three gestures obtain similar probabilities. T1 and T2 should be selected experimentally.

VI. Experiments

All the experiments were carried out processing video sequences acquired with a standard webcam. These video sequences display a user performing the gestures already presented and using an orange and green ball for that purpose. The recognizer was able to process up to 15 frames per second and the size of such frames was 640x480. The sequences were acquired under optimal and unchanging lighting conditions.

A. HMMs for tracking

Our first experiment was designed to prove whether the use of dynamic Bayesian networks improves the result of the tracking procedure. We used a video sequence with 20 frames while the gesture *rotate CW* was made. While the gesture was being done, we stored the real position of the orange ball and the position estimated with two methods: Vision and DBN. The Vision method obtained the most probable ball position at frame t (f_t) using only the visual information from that frame. This position was the centroid of the orange pixel distribution

[4]http://www.cs.waikato.ac.nz/ml/weka/

obtained by the colour filtering. DBN method used our proposal for tracking that combines the observation obtained at frame t with the position of the ball estimated at frame $t-1$.

The comparison of both methods for the X and Y components can be seen in Figures 9 and 10. The difference on the position estimation of the ball for both methods is not significant. However, thanks to the use of the Bayesian networks, the DBN tracking method avoided to obtain abrupt changes in the prediction. The result of this improvement was a more stable tracking method that generated the input for our gesture recognizer.

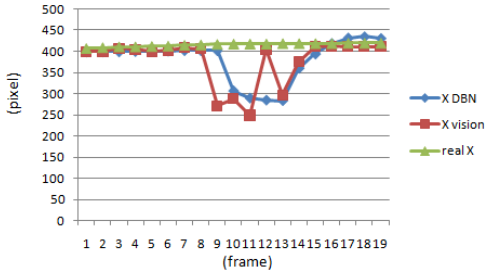

Fig. 9. Results of tracking the coordinate X of the orange ball.

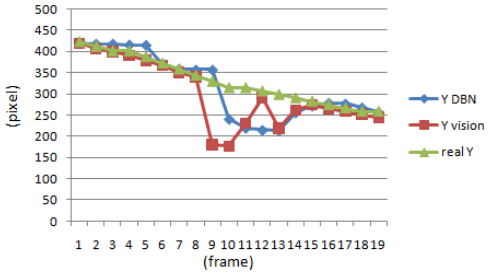

Fig. 10. Results of tracking the coordinate Y of the orange ball.

B. DBNs for gesture recognition

In a second set of experiments, we studied the result of using dynamic Bayesian networks for gesture recognition. Our hypotheses were that they could be used to generate an accurate and robust gesture recognizer and that a post-processing step could also improve the result of the prior prediction. The test sequence for this set of experiments was a complete video sequence with 424 frames acquired while a user performed all the gestures. The user performed the next sequence of gestures: *next*, *back*, *rotate ACW*, *rotate CW*, *zoom in*, *zoom out*. After doing each gesture, the user stopped his hands without moving them from the final position of the last gesture. However, after doing the gesture *back*, both hands returned fast to the central position.

The input for the Bayesian networks was the variation in pixels of the position of the hands during the 3 latest frames. This input was discretized using the following intervals: $(-\infty, -10)$ for *negative*, $[-10, 10]$ for *null* and $(10, +\infty)$ for *positive*.

The probability tables of the static BN were calculated using *Weka 3.6*. With these tables we could obtain the probability of being performing a gesture after integrating some evidences or observations (average variation for the position of the balls in last frames). The next step consisted on generating the tables of probability for the transition between two consecutive frames ($P(gesture_t|gesture_{t-1})$). These tables were defined manually and we proposed two alternatives that are shown in tables I and II.

$P(gesture_t\|gesture_{t-1})$	none	subgesture1	subgesture2
none	0.8	0.1	0.1
subgesture1	0.1	0.8	0.1
subgesture2	0.1	0.1	0.8

TABLE I
TRANSITION PROBABILITY A

$P(gesture_t\|gesture_{t-1})$	none	subgesture1	subgesture2
none	0.95	0.25	0.25
subgesture1	0.025	0.5	0.25
subgesture2	0.025	0.25	0.5

TABLE II
TRANSITION PROBABILITY B

The first table (*Transition Probability A*) was used to represent situations where there are enormous probabilities of doing the same gesture in the next frame, independently of the gesture that is being performed. The second table (*Transition Probability B*) represents scenarios where repeating the prediction *none* will be more probable than repeating any other gesture.

The experiment consisted on evaluating the accuracy of our proposal using three different configurations:

- Static Bayesian Network.
- Dynamic BNs with the transition probability A.
- Dynamic BNs with the transition probability B.

The results of the three systems are graphically presented in Fig. 11. These graphs show the gesture that the human is performing (top row) and the value for the nodes that represent gestures. Only the probabilities for the two nominal values (by node) that represent a real gesture were shown. Therefore, the probability for the *none* value is not presented.

It can be observed in Fig. 11 that after doing the gesture *back*, the gesture *next* is detected. This behaviour was expected and it corresponds to a small sequence of frames when the user moves both hands to the central position. This movements are expected to appear after the *next* (and *back*) gesture. This problematic situation can be solved by disabling the detection for a few following frames after recognizing the *next* (and *back*) gesture.

After comparing the output probabilities obtained with three systems proposed (static BNs and two alternatives for DBNs) we can extract some useful information. The static configuration obtained the lowest probabilities for the gesture that is really being performed. This would make difficult to select the optimal value for the post-processing thresholds

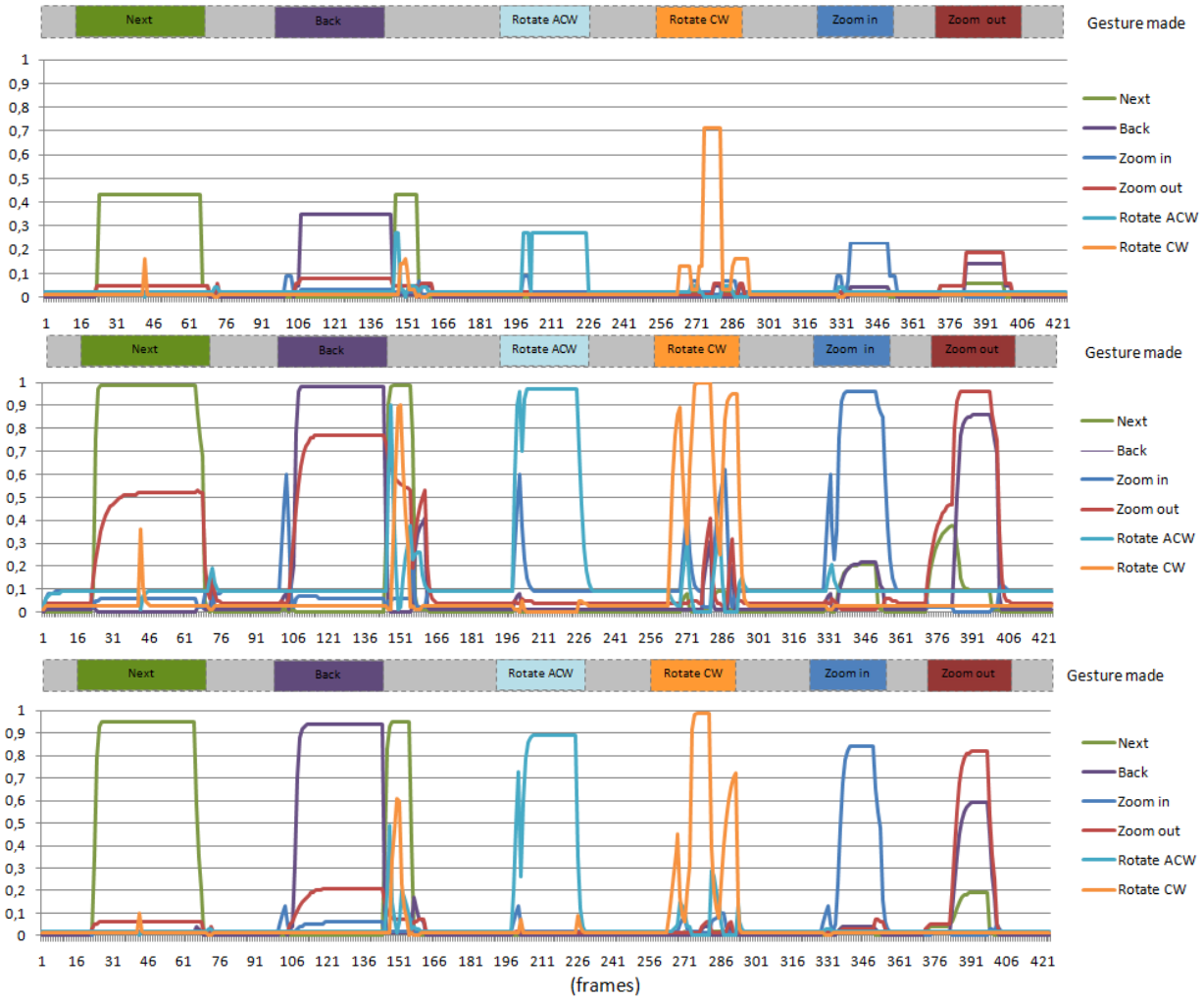

Fig. 11. Output probabilities using static (top) BNs and dynamic BNs with the transition probability A (middle) and B (bottom).

that will determine when the gesture is being performed. The results obtained with the second configuration (DBN with the transition probability A) present an important drawback: the probabilities for the gestures that are not being performed (false positives) are too high. As happened with the static configuration, the threshold selection for the post-processing would be a difficult task.

The configuration that obtained the most promising probabilities was the third configuration (DBN with the transition probability B). The appearance of false positives is reduced (if we compare it to the other proposals) in spite of needing more frames to start detecting a gesture. This conservative approach is more appropriated for any interface where the false positives have a negative impact.

C. Post-processing

Since our objective is to generate a gesture recognizer for a picture viewer, the output probabilities obtained with the system should be translated into real actions. Therefore, the system should provide a specific answer for each frame: the gesture that the human is performing. This gesture can be the special value *none*.

The post-processing uses all the output probabilities as input to obtain (for each test frame) the most probable gesture. Two thresholds presented in Section V-D (T1 and T2) should be selected and we carried out different experiments using several threshold values. We post-processed the output probabilities obtained in the second experiment (only for the DBNs) using three non-overfitted combinations of T1 and T2 values. After performing the gesture next and back, the detection was disabled for a second (15 frames) and the final test sequence consisted on 394 frames.

For each threshold combination we stored the number of true positives (gestures correctly detected), false positives (gestures detected incorrectly), true negatives (*none* gesture correctly detected) and false negatives (gestures not detected). All these results are presented in Table III.

Any gesture recognizer should avoid the appearance of false positives, due to its negative impact for the user acceptance. Considering this information and the results shown in Table

T1=0.8 and T2=0.75					
Estimated\Real	Gesture	None	Estimated\Real	Gesture	None
Gesture	48	2	Gesture	117	0
None	178	166	None	109	168
Transition Probability A			Transition Probability B		
T1=0.75 and T2=0.55					
Estimated\Real	Gesture	None	Estimated\Real	Gesture	None
Gesture	109	3	Gesture	161	0
None	117	165	None	95	168
Transition Probability A			Transition Probability B		
T1=0.1 and T2=0.5					
Estimated\Real	Gesture	None	Estimated\Real	Gesture	None
Gesture	176	19	Gesture	173	7
None	50	149	None	53	161
Transition Probability A			Transition Probability B		

TABLE III
RESULTS OBTAINED WITH THE POST-PROCESSING AND SEVERAL THRESHOLD VALUES

III, the best configuration is a dynamic Bayesian network using the transition Probability B (the most conservative). The best results were obtained with T1=0.75 and T2=0.55. Using the optimal threshold selection and the best combination, 100% of the frames (168) where the user was not performing any gesture the system was able to recognize it. The accuracy of the system for recognizing gestures was 71.23% (161 out of 226 frames). The global accuracy of the system was 83.5% (329 out of 394 frames).

It is difficult to compare our system with others since different set of gestures are recognized. For example, in [3], the average recognition for 5 gestures is 87.75% with a 12.50% of false positives. The main improvement of our system is that we do not get false positives.

VII. CONCLUSIONS AND FUTURE WORK

According to the results obtained from the experiments, visual gesture recognition can be carried out by using dynamic Bayesian networks. We have presented a robust vision system than recognizes several gestures performed by a user using only his/her hands. For future work, we aim to integrate the system with a basic picture viewer tool, obtaining a real and useful application. Moreover, the development of hand tracking system is also considered in order to avoid the use of colour balls.

ACKNOWLEDGEMENTS

This work has been partially supported by the European Social Fund, FEDER, Spanish Ministerio de Ciencia e Innovación (MICINN) and Junta de Comunidades de Castilla-La Mancha regional government under TIN2010-20900-C04-03, PBI08-0210-7127 and PPII11-0309-6935 projects.

REFERENCES

[1] Kisačanin, B., Pavlović, V., Huang, T.: Real-time vision for human-computer interaction. Springer (2005)
[2] Stenger, B., Woodley, T., Cipolla, R.: A vision-based remote control. Computer Vision (2010) 233–262
[3] Aviles-Arriaga, H., Sucar, L., Mendoza, C., Vargas, B.: Visual recognition of gestures using dynamic naive Bayesian classifiers. The 12th IEEE International Workshop on Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. 133–138
[4] Waldherr, S., Romero, R., Thrun, S.: A gesture based interface for human-robot interaction. Autonomous Robots **9**(2) (2000) 151–173
[5] Sigalas, M., Baltzakis, H., Trahanias, P.: Temporal gesture recognition for human-robot interaction. MONTH (2010)
[6] Lee, Y., Tsai, C.: Taiwan sign language (tsl) recognition based on 3d data and neural networks. Expert Systems with Applications **36**(2) (2009) 1123–1128
[7] Lichtenauer, J., Hendriks, E., Reinders, M.: Sign language recognition by combining statistical dtw and independent classification. IEEE Transactions on Pattern Analysis and Machine Intelligence (2008) 2040–2046
[8] Tanguay, D.O.: Hidden Markov Models for Gesture Recognition. Electrical Engineering (1995)
[9] Zafrulla, Z., Brashear, H., Yin, P., Presti, P., Starner, T., Hamilton, H.: American sign language phrase verification in an educational game for deaf children. In: 2010 International Conference on Pattern Recognition, IEEE (2010) 3846–3849
[10] Korb, K.B., Nicholson, A.E.: Bayesian artificial intelligence. Chapman & Hall/CRC (2004)
[11] Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs. 2nd edn. Springer Verlag, New York (2007)
[12] Sturm, J., Plagemann, C., Burgard, W.: Body schema learning for robotic manipulators from visual self-perception. Journal of Physiology-Paris **103**(3-5) (2009) 220–231
[13] Nicholson, A., Flores, M.: Combining state and transition models with dynamic bayesian networks. Ecological Modelling (2010)
[14] Kjærulff, U.: A computational scheme for reasoning in dynamic probabilistic networks, San Mateo, CA (1992) 121–129
[15] Nicholson, A.E.: Monitoring Discrete Environments using Dynamic Belief Networks. PhD thesis, Department of Engineering Sciences, Oxford (1992)
[16] Dean, T., Wellman, M.P.: Planning and Control. Morgan Kaufman Publishers, San Mateo, CA. (1991)
[17] Nicholson, A.E., Brady, J.M.: The data association problem when monitoring robot vehicles using dynamic belief networks. (1992) 689–693
[18] Pynadeth, D., Wellman, M.P.: Accounting for context in plan recognition, with application to traffic monitoring, San Francisco, CA (1995) 472–481
[19] Forbes, J., Huang, T., Kanazawa, K., Russell, S.: The BATmobile: Towards a Bayesian automated taxi. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95). (1995) 1878–1885
[20] Korb, K.B., Nicholson, A.E.: Bayesian artificial intelligence. 2nd edn. Chapman & Hall/CRC (2010)
[21] Rabiner, L.R.: Readings in speech recognition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1990) 267–296
[22] Moore, D.J., Essa, I.A., Iii, M.H.H.: Exploiting human actions and object context for recognition tasks. Computer Vision, IEEE International Conference on **1** (1999) 80
[23] Darwiche, A.: Modeling and Reasoning with Bayesian Networks. Cambridge University Press (2009)
[24] Stichling, D., Kleinjohann, B.: Low Latency Color Segmentation on Embedded Real Time Systems. In: Design and analysis of distributed embedded systems: IFIP 17th World Computer Congress: TC10 stream on distributed and parllel embedded systems (DIPES 2002), August 25-29, 2002, Montréal, Québec, Canada, Kluwer Academic Pub (2002) 247
[25] Bruce, J., Balch, T., Veloso, M.: Fast and inexpensive color image segmentation for interactive robots. In: Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on. Volume 3., IEEE (2002) 2061–2066
[26] Migliore, D.a., Matteucci, M., Naccari, M.: A revaluation of frame difference in fast and robust motion detection. Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks - VSSN '06 (2006) 215
[27] Isard, M., Blake, A.: Contour tracking by stochastic propagation of conditional density. Computer Vision ECCV 96 (1996) 343–356
[28] Khan, Z., Balch, T., Dellaert, F.: An MCMC-based particle filter for tracking multiple interacting targets. Computer Vision-ECCV 2004 (2004) 279–290
[29] Ripley, B.D.: Pattern recognition and neural networks. Cambridge University Press (1996)