Real Time Trajectory Based Hand Gesture Recognition

DANIEL POPA, GEORGIANA SIMION, VASILE GUI, MARIUS OTESTEANU
Faculty of Electronics and Telecommunications
"Politehnica" University of Timisoara
Bd. V. Parvan, Nr. 2, 300223 Timisoara
ROMANIA
gheorghe.popa@etc.upt.ro, georgiana.simion@etc.upt.ro, vasile.gui@etc.upt.ro,
marius.otesteanu@etc.upt.ro

Abstract: The recognition of hand gestures from image sequences is an important and challenging problem. This paper presents a robust solution to track and recognize a list of hand gestures from their trajectory. The main tools of the proposed solution are robust kernel density estimation and the related mean shift algorithm, used in both video tracking and trajectory segmentation. The gesture definition is based on strokes in order to allow the use of a low complexity gesture recognition method. The gesture recognition process is trivial, being reduced to a syntactic analysis of the feature vector avoiding the necessity of complex classification methods based on curve matching. Despite the restrictions derived from the stroke based definition of gestures, the low computational complexity of the algorithm allows its implementation on low-cost processing systems.

Key-Words: gesture recognition, mean shift, robust methods, video tracking, human machine interface.

1 Introduction

Human gestures [1] are expressive human body motions, which generally contain spatial and temporal variation. To handle these variations, an appropriate representation must be chosen.

A vast amount of work in gesture recognition has been performed in the area of computer vision, and is reviewed in [2]. These works can be divided into two categories: trajectory based and dynamics model-based analysis. The trajectory based approach matches curves in configuration space to recognize gestures [3]. The dynamics model-based approach learns a parametric model of gestures.

Gesture recognition systems in general are composed of three main [4] components: image preprocessing, tracking, and gesture recognition. In individual systems some of these components may be merged or missing, but their basic functionality [5] will normally be present.

Image preprocessing is the task of preparing the video frames for further analysis by suppressing noise, extracting important clues about the position of the object of interest (for example hands) and bringing these on symbolic form. This step is often referred to as feature extraction.

Tracking – based on the preprocessing, the position, and possibly other attributes of the object, (hands) must be tracked from frame to frame. This is done to distinguish a moving interest object from the background and from other moving objects, and to extract motion information for recognition of dynamic gestures.

Gesture recognition decides if the user is performing a meaningful gesture based on the collected position, motion and pose clues.

The classical algorithms from the field of pattern recognition are Hidden Markov Models (HMM), correlation, and Neural Networks. Especially the first two have been successfully used in gesture recognition while the Neural Networks often have the problem of modeling non-gestural patterns [6]. HMM is a typical dynamics model and was proven to be robust in its recognition of gestures [7]. The HMM model has been extended to a more general model named Dynamic Bayesian Networks [8].

Recognition of human gestures is important for human-computer interfaces, automated visual surveillance, video library indexing [1], remote control of home appliances, such as TV sets and DVD players [9], which in the future could be extended to the more general scenario of ubiquitous computing in everyday situations, control of a videogame etc.

In this paper, hand gestures and a trajectory based approach are used. We propose a unitary solution centered on the mean shift algorithm, which is used in the 2 major components of the gesture recognition system: video tracking and gesture recognition.

Our goal is to develop a low computational cost real-time gesture recognition algorithm for a human-machine interface (HMI), able to run on low-complexity hardware systems. The system must be able to learn from the user a small number of gestures

in order to recognize them later. The trajectory based approach was chosen, as skin detection is quite well developed and robust [10], and relatively robust low computational cost tracking algorithms are also available [11], [12].

2 Related work

Traditionally the trajectory based hand gesture recognition systems match curves in a configuration space to recognize gestures [3]. Solutions with different levels of complexity and performance were developed for various applications like sign language recognition [13], [14], handwriting recognition [15], [7], user interfaces [16] etc.

In [17] Yang et al. studied the extraction of 2D motion trajectories and its applications in hand gesture recognition.

Black and Jepson extended the CONDENSATION algorithm [18], to recognize gestures and facial expressions, in which, human motions were modeled [17], [16], [19] as temporal trajectories of some estimated parameters (which describe the states of a gesture or an expression) over time. A large variety of particle filters is used for object tracking in video sequences. Most recent approaches include mean shift based particle filters [20], [21], cascade particle filters [22], boosted particle filters and adaptive particle filters [23].

Many gesture recognition methods used colored gloves or markers to track hand movements. Fels and Hinton used data gloves and Polhemus sensors to extract 3D hand location, velocity, and orientation [24].

Bobick and Wilson [25] extract 3-D location of hands using stereo cameras and skin color to recognize a set of 32 gestures using parameterized Hidden Markov Model and data glove.

In [26] video object planes together with a Hausdorff tracker and dynamic time warping are used for hand trajectory estimation, and then the trajectory is classified based on features like trajectory length, location, orientation and hand velocity from the estimated trajectory.

The current state of the art algorithms in gesture recognition, HMM, have an increased computational complexity requiring complex training and inference algorithms [27].

3 Theoretical background

The proposed solution uses a consistent approach based on kernel density estimation and the related mean shift algorithm, which is used in both hand tracking and trajectory segmentation. Mean shift is a non-parametric technique used in feature space analysis [28]. The mean shift algorithm was introduced by Fukunaga and Hostetler in 1975 [29] and it is a procedure for locating stationary points of a density function given discrete data sampled from that function [28]. In the last decade the mean shift algorithm has been used in a large variety of computer vision applications, from image segmentation to background subtraction and video tracking [30]. Evolutionary algorithms have also been successfully used for local maxima detection [31].

3.1 Kernel based probability density estimation

Kernel estimators were introduced by Rosenblatt (1956) and Parzen (1962). In the last decade these estimators received more and more attention, becoming the most popular estimators of the probability density function (pdf) [28]. A d-dimensional kernel based estimator is defined as:

$$\hat{f}_H(x) = \frac{1}{n}\sum_{i=1}^{n} K_H(x - x_i), \qquad K_H(t) = |H|^{-1/2} K\left(H^{-1/2} t\right). \tag{1}$$

A multivariate kernel may be obtained either as product of 1D kernels or by rotating a 1D kernel in the d-dimensional space. The transform matrix, H, can be theoretically completely parameterized, but for practical applications – especially when real-time processing is required – a simplified version is used, in the form of a diagonal matrix, containing on the main diagonal the squares of the kernel's bandwidths in the d dimensions:

$$H = \begin{pmatrix} h_1^2 & 0 & \dots & 0 \\ 0 & h_2^2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & h_d^2 \end{pmatrix}. \tag{2}$$

Computational complexity may be further reduced at the cost of loosing optimality [32], using the same bandwidth, h, for all the dimensions. Thus, the density estimator becomes:

$$\hat{f}_h(x) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right). \tag{3}$$

The kernels used generally in computer vision applications are radial symmetric and can be expressed as:

$$K(x) = c_{k,d} k\left(\|x\|^2\right). \quad (4)$$

In equation (4) $c_{k,d}$ is a normalization constant, which ensures that the kernel integrates to 1 over the definition domain, while *k* defines the so called profile of the kernel (defined only in the positive domain to allow symmetrical construction of the kernel). The probability density estimator becomes:

$$\hat{f}_h(x) = \frac{c_{k,d}}{nh^d} \sum_{i=1}^{n} k\left(\left\|\frac{x - x_i}{h}\right\|^2\right). \quad (5)$$

The most widely used kernels are the Epanechnikov and normal kernels, but the first is not continuously differentiable, while the second has infinite support [32], [33]. Because both these kernels are symmetric, each of them can be defined using a profile function:

$$k_E(x) = \begin{cases} 1 - x, & 0 \le x \le 1 \\ 0, & x > 1 \end{cases} \Rightarrow$$

$$\Rightarrow K_E(x) = \begin{cases} \frac{1}{2c_d}(d+2)\left(1 - \|x\|^2\right), & \|x\| \le 1 \\ 0, \text{ otherwise} \end{cases} \quad (6)$$

$$k_N(x) = e^{-\frac{1}{2}x}, \; x \ge 0 \Rightarrow$$

$$\Rightarrow K_N(x) = (2\pi)^{-d/2} e^{-\frac{1}{2}\|x\|^2} \quad (7)$$

Analyzing the feature space characterized by the probability density function requires first of all finding the maximums of this function. This problem can be reduced at finding the zeros of the derivative (1D case) respectively of the gradient (multivariate case) of the pdf. The mean shift algorithm can be used to directly identify the zeros of the derivative/gradient of the pdf, without having to estimate the probability density first.

3.2 Estimating the probability density gradient

In the case of multidimensional kernels with a single bandwidth, h, for all the dimensions, noting $g(x) = -k'(x)$, the gradient of the pdf can be written as in equation (8). The profile *g* can be used to build the kernel $G(x) = c_{g,d} g\left(\|x\|^2\right)$. The profile *g* must fulfill the condition $\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|\right) > 0$, which is accomplished by all the kernel profiles used in practical applications. The kernel K is also named the shadow of the kernel G [28]. The Epanechnikov kernel is the shadow of the uniform kernel, while the normal kernel has the same shape as its shadow.

$$\hat{\nabla} f_{h,K}(x) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (x_i - x) g\left(\left\|\frac{x - x_i}{h}\right\|^2\right) =$$

$$= \underbrace{\frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)\right]}_{F1 \sim \hat{f}_{h,G}(x)} \times$$

$$\times \underbrace{\left[\frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x\right]}_{F2 = ms_{h,G}(x)}. \quad (8)$$

The F1 factor in the expression of the estimated gradient is proportional to the probability density at x estimated with the kernel G, while the F2 factor represents the mean shift (i.e. the difference between the weighted mean computed using the kernel G and the current central point of the kernel window). The mean shift can be expressed as [28]:

$$ms_{h,G}(x) = \frac{1}{2} h^2 c \frac{\hat{\nabla} f_{h,K}(x)}{\hat{f}_{h,G}(x)}. \quad (9)$$

3.3. The principle of the mean shift algorithm

The local mean weighted by the kernel G is shifted toward the area which contains the most points (samples). From equation (9) it is clear that the mean shift vector – having the same direction as the estimated gradient – always points to the direction where the probability density increases the most, being able to define a trajectory that leads to a stationary point of the estimated density. The estimated density's local maximums are such stationary points and the mean shift algorithm allows finding them by successively repeating the following steps [28] until convergence:

- computation of the mean shift at the current point, $ms_{h,G}(x)$,
- translation of the kernel G with $ms_{h,G}(x)$.

In [28] was proved that in order to insure the convergence of the algorithm, it is enough for the kernel K to have a convex and monotonically

decreasing profile. If these conditions are fulfilled, the convergences of both the trajectory and the pdf are guaranteed, the last one being also, monotonically increasing.

For other gradient based algorithms, shifting toward the local gradient direction does not guarantee the convergence, unless infinitesimal steps are used. The choice of step size is problematic for those algorithms. The mean shift algorithm is able to guarantee the convergence under the above mentioned conditions due to the adaptive value of the magnitude of the mean shift vector (it decreases as the algorithm gets closer to the local maximum point).

The number of steps required for convergence depends on the kernel G. The uniform kernel allows the algorithm to converge after a finite number of steps, while the kernels which weight the points according to the distance from the window's center lead to convergence only after infinite number of steps. Therefore, when a kernel of the second category is used in practical applications it is required to define a low limit for the mean shift magnitude, in order to avoid infinite looping of the algorithm.

The use of the normal kernel guarantees a smooth trajectory toward the stationary point [28], and the results are superior to those obtained with the uniform kernel. The practical usage of the normal kernel is limited by the large number of steps required for convergence, the uniform kernel being preferred instead.

The mean shift algorithm is attracted by a local maximum point if this maximum is unique in a limited neighborhood (inside a small hyper-sphere). Once the algorithm reaches a point close enough to a local maximum, it will converge to that maximum. The points that converge to the same local maximum form the so called basin of attraction of that local maximum [28].

The stationary points returned by the mean shift algorithm have to be pruned in order to retain only the local maxima. A stationary point can be tested to determine whether it is or not a local maximum, by perturbing it with a low amplitude noise vector. If the algorithm converges again (with some tolerance) to the same stationary point, the point is a local maximum. The use of anisotropic symmetric kernels can improve the robustness of the mean shift algorithm [34]. Asymmetric kernels can further improve the performance, especially for video tracking algorithms [35], but their disadvantage is that in most cases they don't have an analytical form.

3.4 Color spaces

In image and video processing applications, color is an essential characteristic of objects. The frames obtained from the camera are represented based on the Red, Green, Blue (RGB) color space.

RGB is a device specific color space, as it was designed for CRT displays. RGB uses additive color mixing, which describes what kind of light must be emitted in order to produce a given color, starting from complete darkness. CMY (Cyan, Magenta, Yellow) is another device specific color space, designed for printers and uses subtractive color mixing (i.e. it describes what kind of inks must be applied on the paper in order to obtain a given color starting from white).

In real situations lighting is not constant and a multitude of factors can cause variations in lighting (e.g. shadows from other objects, self-shadowing, switching between sunlight and overcast, artificial light etc.). The same object, exposed to different lighting conditions, appears to have different colors. Since the color of the object is used for tracking, the separation of brightness from chrominance is essential in order to have at least one component of the color model invariant to lighting changes. The RGB color space is not well suited for tracking, as it does not isolate the brightness and chrominance information (all the three components vary with illumination changes). A lot of research has been conducted in order to define lighting invariant color functions. The most widely used functions are the chromatic color models: normalized RGB, YUV, YCrCb, HSV and CIELAB [36].

CIE (fr. *Commission Internationale de l'Eclairage*) defined in 1931 the first color space based on measurements of human color perception, CIEXYZ. CIEXYZ is the basis for almost all other color spaces. The same CIE defined the CIELUV color space, as a modification of CIEXYZ to display color differences more conveniently. The necessity of a more perceptual linear color space led in 1976 to the definition of the CIELAB color space. In CIELAB the three coordinates represent the lightness of the color (L*), its position between magenta and green (a*) and its position between yellow and blue (b*).

YUV and YCrCb are other color spaces which use one lightness and two chrominance coordinates. YUV is used in PAL TV systems, while YCrCb is used in the very popular image and video compression standards JPEG and MPEG.

The HSV (Hue, Saturation, Value) color space attempts to describe the perceptual color relationships more accurately than the RGB, while preserving a low

computational complexity. The three coordinates represent the color (hue), color's concentration (saturation) and brightness (value). A conic representation of the HSV space is shown in figure 1. Hue is defined as an angle between 0° and 360°, while saturation and value range each from 0 to 1.

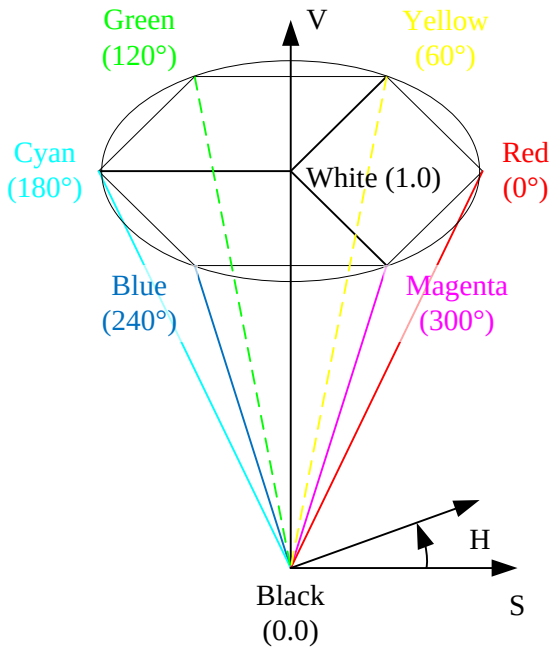


Fig. 1. Conic representation of the HSV color space

The relations (10), (11) and (12) define the transformation from RGB to HSV.

$$H=\begin{cases}0, \text{ if } MAX=MIN\\ \left(0+\dfrac{G-B}{MAX-MIN}\right)\cdot 60°, \text{ if } R=MAX\neq MIN\\ \left(2+\dfrac{B-R}{MAX-MIN}\right)\cdot 60°, \text{ if } G=MAX\neq MIN\\ \left(4+\dfrac{R-G}{MAX-MIN}\right)\cdot 60°, \text{ if } B=MAX\neq MIN\end{cases} \quad (10)$$

$$S=\begin{cases}0, \text{ if } MAX=MIN\\ \dfrac{MAX-MIN}{MAX}, \text{ otherwise}\end{cases} \quad (11)$$

$$V=MAX. \quad (12)$$

The HSV color space is usually preferred in human skin tracking applications [11] because hue is less sensitive to different skin colors and because it is more robust to illumination changes. It was observed [11] that the hue of human skin is the same for all the races, except the albinos. Different races' skins differ only in color saturation (i.e. dark-skinned people have greater saturation, while light-skinned people have lower saturation).

It must be noted that for all the lighting invariant color functions mentioned above, the lighting invariance is guaranteed only under some particular assumptions. Violating these assumptions can severely influence the results of color analysis methods which use these color spaces.

As shown in figure 1, HSV assumes that black is represented as R = G = B = 0, and all colors meet each other at this point at reduced brightness. Another assumption of HSV is correct white balance (i.e. all unsaturated colors – grays – have R = G = B). Violation of these assumptions may be caused by incorrect white balance, non-ideal camera sensitivity and heterogeneous lighting [36].

Theoretically, hue is invariant with illumination changes, but in the case of pixels with low brightness, the R, G and B values obtained from real cameras are low and severely affected by the camera noise. Transforming these values to HSV leads to low values of V and S, and noisy H, as shown in figure 1. The hue may also become noisy when the saturation is low, regardless of V. Depending on the lighting source, bright illuminated objects of light colors (i.e. high V) can get a hue close to that of the human skin.

In order to use hue for tracking, all the pixels, which may produce a confusing hue for the tracker, must be removed from analysis.

4 Outline of the proposed system

Our purpose is to track and discriminate some hand gestures composed of multiple quasi-linear segments (strokes). The algorithm must be able to distinguish the gestures based on the number of segments, the angles between segments and the horizontal axis, the angles between consecutive segments and segments proportionality.

Our approach is based on a restrictive definition of the gesture alphabet, each gesture being composed of a limited number of strokes, in order to maintain a low computational cost for the gesture recognition algorithm.

The feature vector consists of 3 components:

- number of strokes,
- stroke angle sequence and
- stroke lengths (optional).

Using this definition for the feature vector the gesture recognition can be realized at a low computational cost, following a sequential syntactic analysis of the feature vector. The use of these parameters allows the definition of a variety of gestures which can be easily discriminated: square, wide/tall rectangle, triangle, Z, N vertical/horizontal

multiple strokes, cross etc. Each gesture may have a different meaning depending on the movement direction. For example one action can be associated with drawing a square clockwise, and another (possibly opposite) action can be associated with drawing a square counterclockwise.

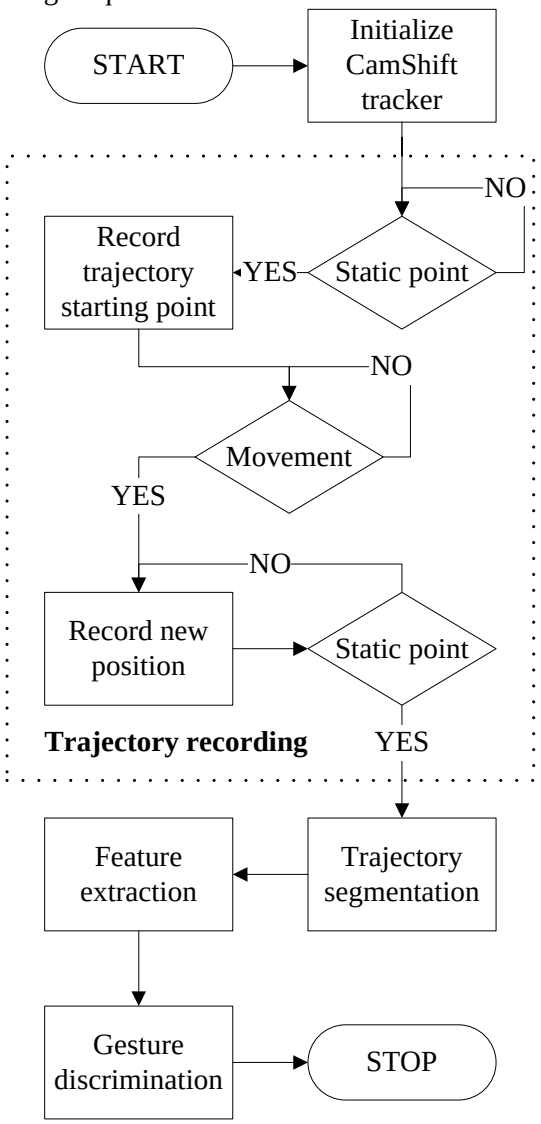


Fig. 2. Flow diagram of the trajectory based recognition algorithm

The total number of possible gestures composed of a certain number, *n*, of strokes using the 8 directions is $8\times 7^{n-1}$. For the first stroke, all the 8 directions are possible. For each subsequent stroke only 7 possible directions are available (the direction of the previous stroke is excluded). Over 3000 gestures are possible even with a maximum of only 4 strokes, and the total number of possible gestures increases exponentially with the number of strokes.

However, not all of the possible gestures can be used in practical applications (e.g. successions containing more than 2 strokes with positive vertical movement, Δy, without at least one stroke with negative vertical movement, $-\Delta y$, are very likely to exceed the frame space captured by the camera).

Robust kernel density estimation methods and the mean shift algorithm represent the backbone of the proposed dynamic gesture recognition solution. They are the core of the system's 2 main components: tracking and trajectory segmentation.
The method we propose, presented in figure 2, uses a CamShift tracker to track the hand of the user and saves a trajectory obtained from the centers of the tracked region. As the orientation of the tracked hand returned by the CamShift algorithm may have different meanings (not only the modification of the hand's orientation) we choose not to use this parameter in the analysis of the gestures. The saved trajectory is then segmented into strokes. Considering all the gestures used are composed of a reduced number of strokes, information like number of strokes, average angles with horizontal axis, angles with neighboring segments and segments proportionality can be easily derived from the segmented trajectory. Finally, based on the extracted features the gesture is uniquely identified.

5 System components description

5.1 Tracking

As the overall computational cost must be low, complex tracking solutions like those based on particle filters are not appropriate, and a tracking algorithm based on target representation and localization must be used. Such a solution is offered by the CamShift (Continuously Adaptive Mean Shift) algorithm introduced by Bradski [11]. An implementation of this algorithm is available in the OpenCV library [37].

The CamShift Algorithm mainly consists of the following steps:

1. Choose the initial location of the search window;

2. Mean Shift (one or many iterations); store the zeroth moment (distribution area or window size);

3. Set the search window size equal to a function of the zeroth moment found in Step 2;

4. Repeat Steps 2 and 3 until convergence.

The algorithm outputs the center, size and orientation of the tracked object. The object trajectory can be obtained from the positions of the center of the object in successive frames.

The region containing the hand to track must be firstly selected. Then, a mask is applied on the HSV

image in order to eliminate the pixels which have too small saturation and also pixels with too small or too big value. Hue is too noisy for these removed pixels.

A model of the desired hue is created for the first frame using a color histogram. The CamShift algorithm is used to track the object in the next frames.

CamShift operates on a color probability distribution image derived from color histograms. CamShift calculates the centroid of the 2D color probability distribution within its 2D window of calculation, re-centers the window, and then calculates the area for the next window size. Thus, in the next frame, it is not necessary to calculate the color probability distribution over the whole image, but only for a smaller image region surrounding the current CamShift window. This saves computational time when flesh color does not dominate the image [11].

CAMSHIFT resolves the bandwidth selection problem by continuously re-scaling itself in a way that naturally fits the structure of the data. An object's potential speed scales with its distance to the camera. But the area occupied by the object in the image also scales with the distance from the camera. Objects that are close to the camera can move rapidly, but they also occupy a large area in the image determining a large window size for CamShift, allowing it to catch the large movement. Objects that are far from the camera, occupy a small area in the image, reducing the CamShift's window size, but, their apparent speed in the scene is proportional to the inverse of their distance from the camera, allowing the tracker to catch the object even at high real speeds. This natural adaptation to distribution scale and translation allows a successful tracking without the need for predictive filters or variables, thus saving important computational time [11].

5.2 Trajectory recording

The consecutive centers of the tracked region define a relatively rough (noisy) trajectory, increasing the difficulty of strokes detection. Therefore the trajectory is smoothed so that each new trajectory point, $t[i]$, is obtained as a weighted average of the new measured point, $m[i]$, and the previous trajectory point, $t[i-1]$:

$$t[i] = \alpha m[i] + (1-\alpha) t[i-1]. \qquad (13)$$

The weighting parameter must be carefully chosen, as a too small value would oversmooth the trajectory, while larger values lead to relatively rough trajectories.

The recording of a new gesture trajectory is triggered by a movement of the user's hand occurring after a short interval (1-2 seconds) of static position. Minimum thresholds are imposed to the amplitude and speed of the movement in order to avoid false triggering due to tracking noise or hand trembling. The gesture trajectory recording ends when the movement speed falls below the imposed threshold for at least 1.5 seconds.

5.3 Trajectory segmentation

A set of angles with the horizontal axis is computed over the recorded trajectory. Computing the angle for each small segment determined by two consecutive points of the trajectory may result in a very noisy angle set, with many false angle discontinuities. This noise is caused by angles between trajectory points that are relatively close to each other, because the image is sampled on a rectangular grid. Selecting a reduced number of trajectory points using a fixed step (e.g. choosing each second or third point of the trajectory) results in a relatively smoothed angle set. To further improve the results, we chose to adaptively select trajectory points based on a threshold distance. An adequately chosen threshold distance significantly reduces the number of outliers in the resulting angle sequence. Even with the fixed step selection, a distance threshold must be imposed in order to avoid computing the angle if the two points have the same position.

In order to split the trajectory into strokes, the ends of these strokes must be detected. The starting point of the trajectory is also the starting point of the first segment, and the end point of the trajectory is the end point of the last segment. All other strokes' end points are detected as angle discontinuity points, the starting point of each segment being the end of the previous segment. A stroke discontinuity is detected as a point where the segment angle with the horizontal axis changes significantly. A threshold must be imposed on the minimum length of a stroke, in order to avoid detection of false strokes.

As the raw angle set computed adaptively over the trajectory may still contain some outliers – especially around the strokes joint points – filtering is necessary in order to simplify the feature extraction process.

In typical situations, median filtering within a 5 angles window reduces the roughness of the angle sequence over a stroke, but used alone it is not very helpful when either very low or very high frequency noise is present.

A second filtering approach, we implemented and tested, uses an intelligent classification filter with a 7 angles window. Each of the 7 angles in the window is assigned to one of 8 angle classes, and the number of angles in each class is counted. Ideally all the angles over a stroke should belong to the same class. Exceptions may occur due to high segment angle noise or due to a neighboring stroke's end/joint point. If at least 4 angles in the window belong to the same class, the processed angle is replaced with that class's typical value. If either the resulting class differs from the current stroke's class or no class has at least 4 representatives in the window, the filter advances and checks the next angles to detect if there is a new stroke, or just a noisy angle group. A new stroke is assumed if all the angles for a group of consecutive segments that cover the minimum stroke length are filtered to the same class.

The third approach, we implemented and tested uses a mean shift based clustering of the angle set and is the most robust and efficient. A 2D anisotropic kernel obtained by multiplication of 2 1D symmetric kernels with different profiles is used. Considering that the data vector, $\mathbf{x}=[\theta, p]$, consists of the segment angle, θ, and position in the angle sequence, p, the kernels for the two dimensions are:

$$G_1(\theta)=c_{g_1}\cdot g_1\left(\|\theta\|^2\right),\qquad G_2(p)=c_{g_2}\cdot g_2\left(\|p\|^2\right) \tag{14}$$

Considering different bandwidths for the 2 kernels, the resulting product kernel is:

$$G(\mathbf{x})=\left[\frac{c_{g_1}}{h_\theta}\cdot G_1\left(\frac{\theta}{h_\theta}\right)\right]\cdot\left[\frac{c_{g_2}}{h_p}\cdot G_2\left(\frac{p}{h_p}\right)\right]=\frac{c_{g_1}c_{g_2}}{h_\theta h_p}\cdot G_1\left(\frac{\theta}{h_\theta}\right)\cdot G_2\left(\frac{p}{h_p}\right). \tag{15}$$

Bandwidth selection is an important problem in nonparametric methods and there is a considerable amount of literature concerning this aspect. A recent review of bandwidth selection techniques is presented in [38].

Product kernels obtained by multiplication of 1D kernels with identical profiles but different bandwidths in different dimensions were used in [39]. Anisotropic kernels were also previously used in computer vision for image and video segmentation [40] and video tracking [34]. In the related approach, bilateral filtering, Tomasi and Manduchi [41] also use separable kernels for the space and range domains. The relationship between the bilateral filters and the mean shift algorithm is emphasized in [42].

In order to guarantee the convergence of the mean shift algorithm, the shadows of both 1D kernels must be convex and monotonically decreasing so that their resulting 2D product kernel is also convex and monotonically decreasing.

In the angle domain, a uniform kernel is used with an initial bandwidth, h, which may be later increased to merge clusters that are very close to each other in this domain. The shadow of the uniform kernel is the Epanechnikov kernel, which has a convex and monotonically decreasing profile, satisfying the convergence condition. If the mean shift algorithm is applied based only on the statistical information, some outliers may survive the filtering process (if they correspond to the cluster of a different stroke present in the trajectory).

Spatial information is taken into account using a spatial isotropic kernel (e.g. triangular, Epanechnikov, normal) that assigns larger weights to the angles of the segments in the middle of the window and smaller weights to those in the extremities. Each of the above mentioned isotropic kernels have shadows that satisfy the convergence condition.

Similar to the bilateral filtering approach, the mean shift is applied only for the angle dimension, while the kernel in the space domain does not change its central position between the mean shift iterations.

Using the uniform kernel for angles and the Epanechnikov kernel for the spatial information (the position in the angle sequence) the mean shift vector becomes:

$$ms_{h_\theta,h_p,G}(\theta)=\frac{\sum_{i=1}^{n}\theta_i G_1\left(\frac{\theta-\theta_i}{h_\theta}\right)G_2\left(\frac{p-p_i}{h_p}\right)}{\sum_{i=1}^{n}G_1\left(\frac{\theta-\theta_i}{h_\theta}\right)G_2\left(\frac{p-p_i}{h_p}\right)}-\theta. \tag{16}$$

Clusters with very few angles (not many enough to define a stroke) are removed from the analysis and the corresponding angles are assigned each to the spatial neighboring cluster that is the nearest in angle domain. Finally only clusters that correspond to plausible strokes are kept and the stroke ends are detected as points where the angle cluster changes in the angle sequence.

In the mean shift clustering process, special care is taken for the angles in the intervals [180° – h, 180°] and (–180°, h – 180°], due to the circular definition of the angle as shown in figure 3.

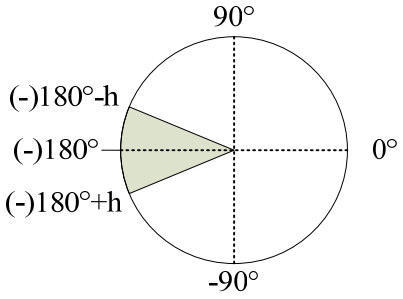


Fig. 3. Dealing with the circular definition of the angle for correct mean-shift clustering

Therefore, if the center of the kernel window falls within the [180° – *h*, 180°], angles from the (–180°, *h* – 180°] interval will also be taken into account by converting them to the (180°, *h* + 180°] domain. Similar, when the angle kernel window is centered on a value within the (–180°, *h* – 180°] interval, values from the [180° – *h*, 180°] interval will be taken into account by converting them to the [–180° – *h*, –180°]. Finally, if the resulting filtered value is outside the interval (–180°, 180°], it will be converted to fit this interval.

The results obtained with the second and third filtering approaches are further improved if the median filtering is used as a preprocessing step.

5.4 Feature extraction

After the stroke ends are detected, the parameters of the trajectory are extracted. The useful parameters which must be extracted are: the number of strokes, the strokes' angle sequence and the strokes' lengths.

A simple analysis of the mean shift filtered angle sequence obtained at the previous step allows the extraction of the required parameters angle and end points for each stroke, and total number of strokes. The strokes' lengths can be easily obtained using the coordinates of the stroke ends.

Each gesture known by the system is represented by a codified angle sequence. The angles of the 8 directions allowed and the associated codes are presented in table 1. Opposite directions have complementary codes.

Table 1

θ [°]	0	45	90	135	–45	–90	–135	180
Code	0	1	2	3	4	5	6	7

The angle of each stroke must be classified and assigned to one of the 8 classes. In order to assign the angle to a class, its value must fit a ±20° window around the standard value. A 5° guard space is left between consecutive windows as shown in figure 4.

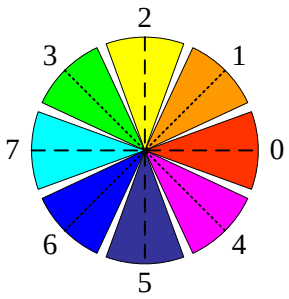


Fig. 4. Angle classes

If a stroke's angle falls within a guard space it is not possible to classify it and the analysis is stopped, invalidating the current gesture.

The first angle of the sequence is the angle between the first stroke and the horizontal axis, while the next angles can be either the angles made by each stroke in the sequence with the horizontal axis or with the previous stroke. The second solution may increase the robustness to small global trajectory rotations, but reduces the possible gestures alphabet's size.

A global rotation correction may be applied to the strokes angle sequence if a large median deviation from the nearest class is detected. Figure 5 shows an example of a 3-stroke gesture that exhibits a global –15° rotation. Direct classification of the strokes angle sequence (left side) produces the erroneous sequence (0°, –90°, 0°), while the global rotation correction allows a correct classification o the sequence to (0°, –135°, 0°).

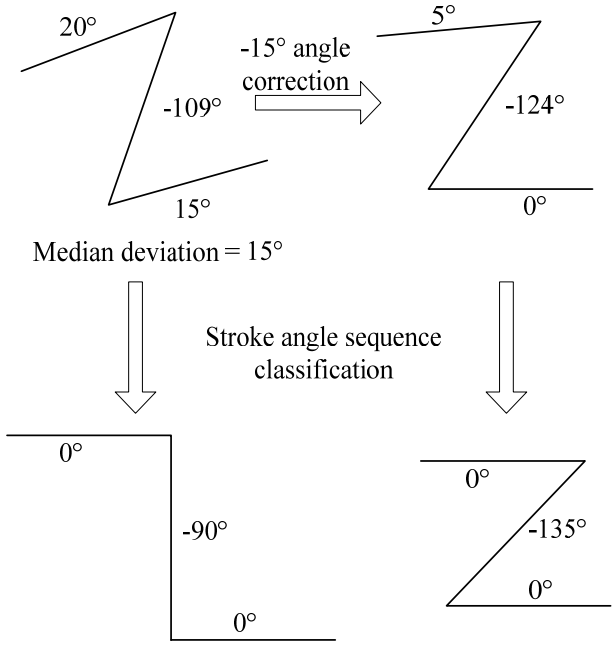


Fig. 5. Effect of global rotation correction

At the end of the trajectory segmentation process, the endpoints of all strokes are known. As

all the valid strokes are assumed to be linear, it is straightforward to compute the approximate length for each stroke, as the Euclidean distance between its endpoints:

$$l_i = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2},$$

$(x_i, y_i), (x_{i+1}, y_{i+1}) - (x, y)$ coordinates of the i-th stroke endpoints

(17)

5.5 Gesture discrimination

The gesture discrimination process is very simple, and does not require complex classification based on curve matching.

A gesture consists of a minimum of 2 strokes. Each gesture is uniquely identified based on:

- number of strokes (>1),
- angle sequence and
- strokes proportionality (optional).

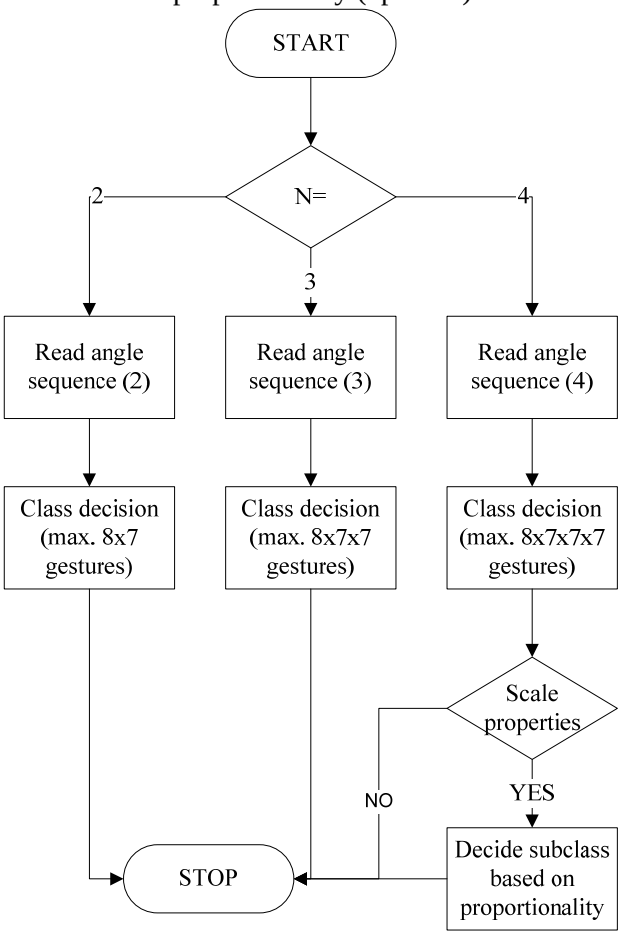


Fig. 6. Sequential gesture discrimination

The discrimination process is done sequentially, based on the 3 parameters, as shown in figure 6. As all the gestures that reach this processing step were already validated from the number of strokes and angle sequence point of view, these parameters are strict, while the stroke proportionality is allowed to vary within an error window.

The first parameter to take into account when discriminating between gestures is the number of strokes. This step removes from further analysis all the gestures with different number of strokes.

In the next step the angle sequence is decoded. In most cases this step is enough to uniquely identify the gesture and terminate the analysis.

Some 4-stroke gesture codes (e.g. square/rectangle codes) need further classification based on strokes proportionality. The strokes proportions are allowed to vary within an error window. The error window must be chosen large enough to accommodate the imperfections of the hand drawn trajectory, but small enough to allow correct discrimination between different gestures.

6 Results

The hardware processing system we used to implement and test the solution was a PC with a 1.6 GHz AMD Athlon XP processor and 768 MB of RAM. Video acquisition was realized using a commercial USB webcam with 352×288 video resolution. The software application was implemented in Visual C++ and some functions from the OpenCV library were used.

Fig. 7. The CamShift tracker.

Our application uses a tracker based on the CamShift function of the OpenCV library. This algorithm is able to track well a hand based on the hue, assuming saturation and value thresholds are relatively well calibrated and no occlusions with objects of similar color occur. Figure 7 presents the tracker focused on a hand.

Our practical tests revealed that a value of 0.4 for the weighting parameter, α, in equation (13) produces a relatively smooth trajectory. A threshold distance of 3 pixels was used in the computation of the segments angle sequence, for the 352×288 frame size. A threshold was also imposed on the minimum length of a stroke, in order to avoid detection of false strokes. A reasonable value for this threshold is 1/10 of the image height. The initial bandwidth of the uniform kernel used in the angle domain is 20°, but may be later increased to merge clusters that are very close to each other in this domain. The bandwidth used for the Epanechnikov kernel in the position domain is 20.

The use of Epanechnikov kernel in the position domain, even with such a large bandwidth, avoids taking into account angle values from a far stroke which could be similar to some noisy angles around the current processed position. Even if such angles would fall within the kernel window, their influence would be diminished by the small weights near the kernel window's ends. Without using the kernel in the spatial domain, noisy angles may survive in the filtered angle sequence, if their values are similar to those of another stroke.

The recorded trajectory for a 3-stroke gesture is presented in figure 8.

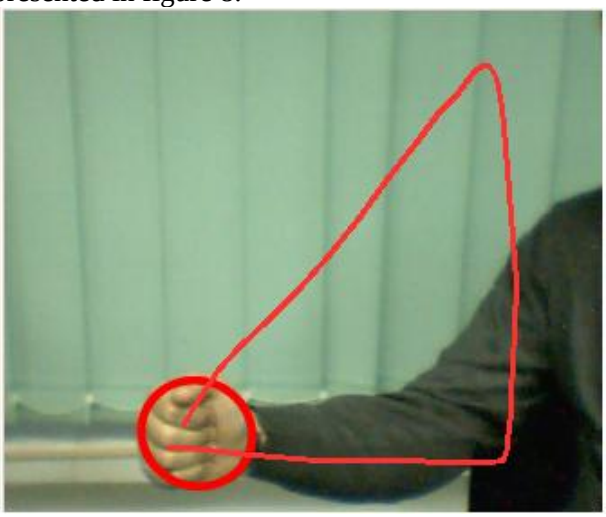


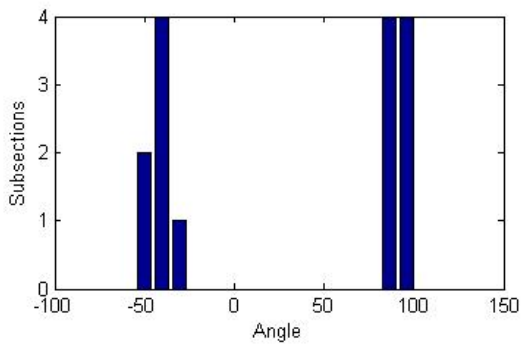
Fig. 8. Trajectory of a 3-strokes gesture.

Histograms for 2, 3 and 4-strokes gestures are presented in figure 9. The 3 histograms of subsection angles indicate that the subsection angles are clustered, allowing easy separation of the strokes.

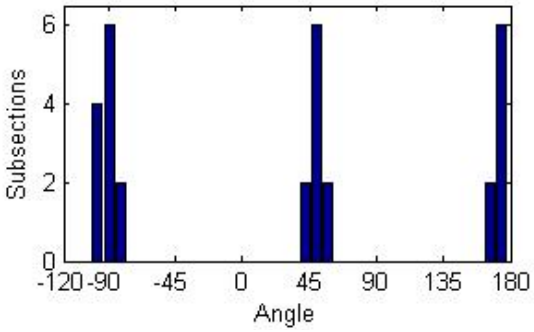
The resulting stroke angle sequence for the gesture in figure 8 is: 45°, -90°, 180° and the resulting coded sequence for this example is "1, 5, 7".

Tests were performed on a total number of 60 gestures executed by 3 subjects in both daylight and artificial light. The tested gesture sequences were composed of 2, 3 and 4 variable length strokes, with sharp and/or right angles between consecutive strokes.

a)



b)



c)

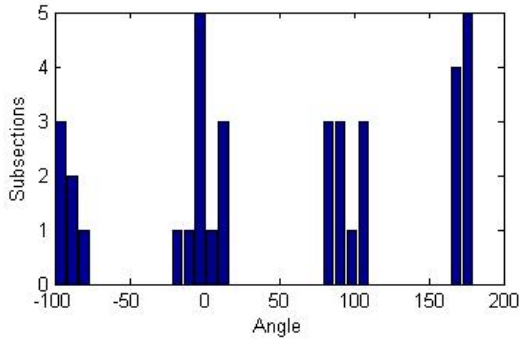


Fig. 9. Angle histogram for: a) 2-stroke gesture, b) 3-stroke gesture, c) 4-stroke gesture.

Figure 10 presents a histogram of standard deviations in raw angle sequences realized using 70 strokes from 25 randomly chosen gestures. The standard deviation of the angle over a stroke varied between less than 5° for near linear strokes and over 30° for the rough ones. After combining median filtering and mean shift filtering the standard angle deviation over a stroke is reduced to 0° – 3°.

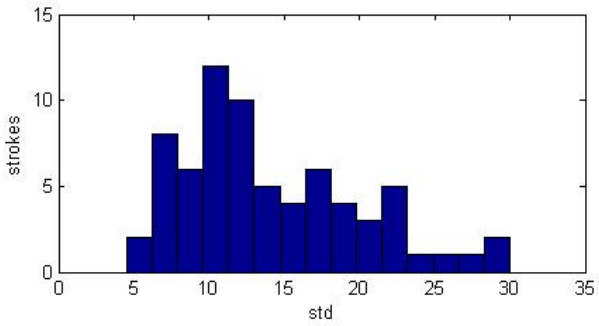


Fig. 10. Histogram of the standard deviations over a set of 70 strokes

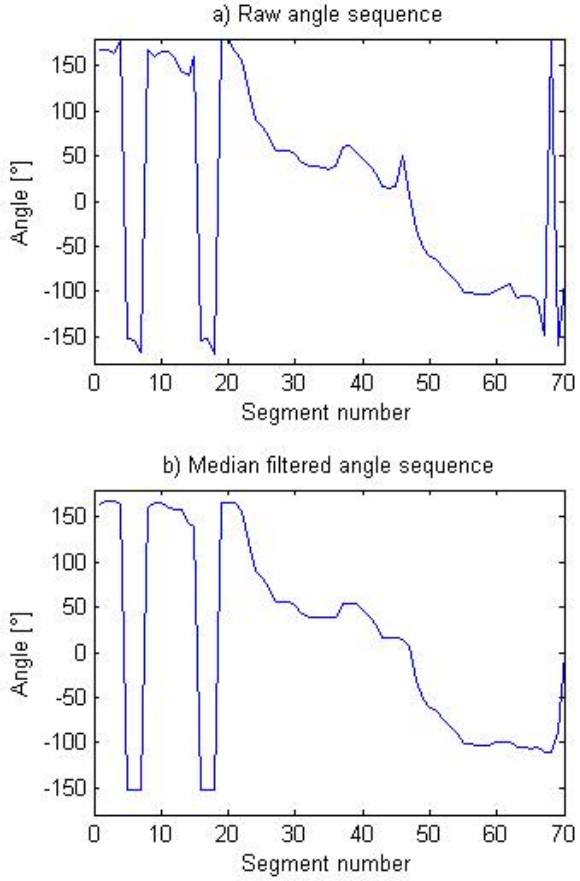


Fig. 11. Original and filtered angle sequences plot

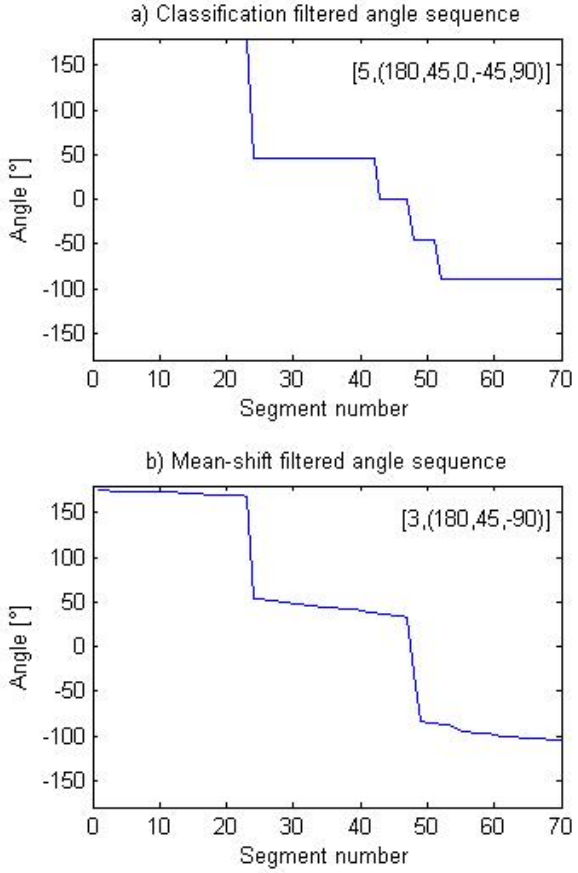


Fig. 12. Segmented trajectory and extracted features

The mean shift based algorithm was able to correctly identify the gestures even under hardly challenging conditions like hand trembling or even short term target loses in a cluttered environment. Our algorithm runs in real time on the specified hardware system, being able to recognize successfully the gestures we tested so far.

Figure 11.a) presents a high noise angle sequence obtained from a 3 stroke gesture, affected by hand trembling. The median filtered sequence is represented in figure 11.b). The classify-and-filter method leads in this case to an erroneous 5 strokes detected gesture (figure 12.a)), while the mean shift based algorithm (figure 12.b)) is still able to correctly interpret the gesture as the 3 strokes 180°, 45°, -90°.

7 Conclusions

Other gesture recognition solutions, [43], [44], rely on processing different successions of shape and positions of the hand and achieve relatively good recognition rates at the price of significantly higher computational complexity.

Our solution has a reduced computational complexity, using a mean-shift based tracker, trajectory segmentation and trivial syntactic, trajectory based gesture recognition. The project is still in an early phase, but the tests we were able to make so far reveal the reliability of the proposed solution.

The low computational cost allows the use of the algorithm to implement HMIs on relatively cheap systems. The user can use the system with a minimal training. He may decide the gesture-action correspondence.

Further developments include definition of some recommended optimized gesture alphabets and studying the possibility to relax the angle restrictions for some classes of gestures (e.g. allow equilateral triangles). An optimized alphabet involves the selection of gestures such that the general user can easily memorize and execute them. It is therefore desirable to select gestures composed of reduced number of strokes, representing some significant geometric shapes and minimizing the

possibility for the user to exceed the visual area of the camera while executing the gesture.

An audio feedback to the user would also be useful, in order to allow it to focus its sight on its main activity (e.g. driving). The audio feedback is straightforward to implement (i.e. play a pre-recorded audio track for each recognized gesture) and allows the user to know whether his gesture was correctly executed and correctly recognized by the system. Definition of a procedure for canceling the previous command is also necessary to allow the user to cancel a command generated by a wrong execution of a gesture.

Some further developments are also possible in the tracking algorithm. The use of a more generally lighting invariant color function [36] and the use of some semantic information about the tracked object (human hand) can lead to the development of an automatic tracker initialization method and can also improve the tracking results under severe conditions like important lighting variance or long term occlusions [45].

References:

[1] T. S. Wang, H. Y. Shum, Y. Q. Xu and N. N. Zheng, Unsupervised Analysis of Human Gestures, *IEEE Pacific Rim Conference on Multimedia*, 2001, pp. 174-181.

[2] Y. Wu, T. Huang, Vision-Based Gesture Recognition: A Review, *Proceedings of the International Gesture Recognition Workshop*, 1999, pp. 103-115.

[3] L. Campbell and A. Bobick.: Recognition of Human Body Motion Using Phase Space Constraints, Proceedings of the Fifth International Conference on Computer Vision, 1995, pp. 624-630.

[4] T. Moeslund and L. Nrgaard, *A Brief Overview of Hand Gestures used in Wearable Human Computer Interfaces*, Technical Report CVMT 03-02, Computer Vision and Media Technology Laboratory, Aalborg University, DK, ISSN: 1601-3646, 2003.

[5] V.I. Pavlovic, R. Sharma, and T.S. Huang, Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, 1997, pp. 677–695.

[6] H.K. Lee and J.H. Kim, An HMM-based Threshold Model Approach for Gesture Recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, No. 10, 1999, pp. 961–972.

[7] T. Starner and A.Pentland.: Visual Recognition of American Language Using Hidden Markov Models, Proceedings of the International Workshop on Automatic Face and Gesture Recognition, Zurich, 1995, pp. 189-194.

[8] V. Pavlovic, J. M. Rehg and J. MacCormick, Impact of Dynamic Model learning on Classification of Human Motion, *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2000, pp. 788-795.

[9] S. Lenman, L. Bretzner and B. Thuresson, Computer Vision Based Recognition of Hand Gestures for Human-Computer Interaction, *Technical report TRITANA -D0209, CID-172*, ISSN 1403-0721, Department of Numerical Analysis and Computer Science, 2002.

[10] M. J. Jones and J. M. Rehg, Statistical Color Models with Application to Skin Detection, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1999, 274-280.

[11] G. R. Bradski, Computer vision face tracking as a component of a perceptual user interface, *Intel Technology Journal Q2 1998*, http://developer.intel.com/technology/itj/archive/1998.htm.

[12] D. Comaniciu, V. Ramesh, P. Meer, Kernel-Based Object Tracking, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 5, 2003.

[13] T. Starner, J. Weaver, A. Pentland, Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 12, 1998 pp. 1371-1375.

[14] N. D. Binh, E. Shuichi, T. Ejima, Real-Time Hand Tracking and Gesture Recognition System, *ICGST International Journal on Graphics, Vision and Image Processing*, Vol. 06, Special Issue on Biometrics, 2006, pp. 31-39.

[15] G. S. Chen, Y. D. Jheng, H. C. Yao, H. C. Liu, Stroke Order Computer-based Assessment with Fuzzy Measure Scoring, *WSEAS Transactions on Information Science and Applications*, Issue 2, Vol. 5, 2008 pp. 62-68.

[16] L. Nørgaard, *Probabilistic Hand Tracking for Wearable Gesture Interfaces*, Master's thesis, Laboratory of Computer Vision and Media Technology, Aalborg University, Denmark, 2003.

[17] M. H. Yang, N. Ahuja and M. Tabb, Extraction of 2D Motion Trajectories and Its Application

to Hand Gesture Recognition, *IEEE Transactions on pattern analysis and machine intelligence*, Vol. 24, No. 8, 2002, pp. 1061-1074.

[18] M.J. Black and A.D. Jepson, A Probabilistic Framework for Matching Temporal Trajectories: CONDENSATION-Based Recognition of Gestures and Expressions, *Proceedings of the Fifth European Conference on Computer Vision*, 1998, pp. 909-924.

[19] J. Lin, Y. Wu, and T.S. Huang. Capturing human hand motion in image sequences. *Proceedings of IEEE Workshop on Motion and Video Computing*, 2002, pp. 99-104.

[20] B. Zhang, T. Weifeng, Z. Jin, Joint tracking algorithm using particle filter and mean shift with target model updating, *Chinese Optics Letters*, vol. 4, Issue 10/2006, pp.569-572.

[21] C. Shan, Y. Wei, T. Tan, F. Ojardias, Real time hand tracking by combining particle filtering and mean shift, *Proceedings of Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, 2004, pp. 669-674.

[22] Y. Li , H. Ai, T. Yamashita, S. Lao, M. Kawade, Tracking in Low Frame Rate Video: A Cascade Particle Filter with Discriminative Observers of Different Lifespans, *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[23] G. Fan; V. Venkataraman; L. Tang; J. Havlicek A Comparative Study of Boosted and Adaptive Particle Filters for Affine-Invariant Target Detection and Tracking *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, 2006, pp. 138.

[24] S.S. Fels and G.E. Hinton, Glove-Talk II: A Neural Network Interface which Maps Gestures to Parallel Format Speech Synthesizer Controls, *IEEE Transactions on Neural Networks*, Vol. 9, No. 1, 1997, pp. 205-212.

[25] A.D. Wilson and A.F. Bobick, Parametric Hidden Markov Models for Gesture Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 9, 1999, pp. 884-900.

[26] M. K. Bhuyan, D. Ghosh, P. K. Bora, Hand motion tracking and trajectory matching for dynamic hand gesture recognition, *Journal of Experimental & Theoretical Artificial Intelligence*, Volume 18, Issue 4, 2006 , pp. 435-447.

[27] S. Rajko, G. Qian, T. Ingalls, J. James, Real-time Gesture Recognition with Minimal Training Requirements and On-line Learning, *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[28] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, 2002, pp. 603–619.

[29] K. Fukunaga, L. D. Hostetler, The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition, *IEEE Transactions on Information Theory*, Vol. 21, No. 1, 1975, pp. 32–40.

[30] C. Ianăşi, V. Gui, F. Alexa, C.I. Toma, Noncausal, Adaptive Mode-Tracking Estimation for Background Subtraction in Video Surveillance, *WSEAS Transactions on Signal Processing*, Issue 1, Volume 2, 2006, pp. 52-59.

[31] Q. Hua, B. Wu, H. Tian, A Detecting Peak's Number Technique for Multimodal Function Optimization, *WSEAS Transactions on Information Science and Applications*, Issue 2, Vol. 5, 2008 pp. 37-43.

[32] D. W. Scott, *Multivariate density estimation*, Wiley Interscience, 1992.

[33] B.A. Turlach, Bandwidth selection in kernel density estimation: A review, *Discussion Paper 9317, C.O.R.E and Institut de Statistique, Universite Catholique de Louvain*, Louvain-la-Neuve, 1993.

[34] R. Collins, Mean-shift blob tracking through scalespace, *IEEE Conference on Computer Vision and Pattern Recognition*, 2003, Volume 2, pp. 234-240.

[35] A. Yilmaz, Object Tracking by Asymmetric Kernel Mean Shift with Automatic Scale and Orientation Selection, *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[36] J.F. Lichtenauer, M.J.T. Reinders, E.A. Hendriks, A self-calibrating chrominance model applied to skin color detection, *Proceedings of the Second International Conference on Computer Vision Theory and Applications, VISAPP 2007*, Vol. 1, 2007, pp. 115-120.

[37] http://opencvlibrary.sourceforge.net.

[38] D. Comaniciu, V. Ramesh, P. Meer, The variable bandwidth mean shift and data-driven scale selection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.25, No.2, 2003, pp.281-288.

[39] A. Elgammal, R. Duraiswami, D. Harwood, L. S. Davis, Background and foreground modeling using nonparametric kernel density estimation

for visual surveillance, *Proceedings of the IEEE*, Volume 90, No. 7, 2002, pp. 1151-1163.

[40] J. Wang, B. Thiesson, Y. Xu, and M. Cohen. Image and video segmentation by anisotropic mean shift, *European Conference on Computer Vision*, 2004.

[41] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, International Conference on Computer Vision, Bombay, India, 1998, pp. 839-846.

[42] D.Barash, A Fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.24, No.6, 2002, pp.844-847.

[43] L. Bretzner, I. Laptev and T. Lindeberg, Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering, *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, 2002, pp 405-410.

[44] J.H. Shin, J.S. Lee, S.K. Kil, D.F. Shen, J.G Ryu, E.H. Lee, H.K. Min and S.H. Hong, Hand Region Extraction and Gesture Recognition using entropy analysis, *IJCSNS International Journal of Computer Science and Network Security*, Vol. 6 No. 2, 2006, pp. 216-222.

[45] R. R. Pinho, J. M. R. S. Tavares, M. V. Correia, An Improved Management Model for Tracking Missing Features in Computer Vision Long Image Sequences, *WSEAS Transactions on Information Science and Applications*, Issue 1, Vol. 4, 2007 pp. 196-202.