RESUMEN

Este proyecto trata de aplicar diferentes conceptos de la visión por computador a dos casos prácticos, incluidos en el campo de "interacción persona computador" (HCI). En el primero de los casos la meta es la implementación de una interfaz gestual, en el que con ayuda de una web-cam y los movimientos de nuestra mano logremos sustituir al ratón. Y el segundo caso consiste en crear una base de conocimiento sobre el alfabeto dactilológico del lenguaje de los sordos consistente en una serie de filmaciones y clasificación de estas. Además se realizará un estudio sobre la viabilidad de implementación de un sistema de reconocimiento automático de los gestos contenidos en estos videos, realizando para ello una serie de análisis sobre fotogramas de los videos capturados. Para la consecución de estas actividades es necesario manejar conceptos como espacios de color, segmentación, reconocimiento de formas y otros que se presentarán a lo largo del trabajo.

Palabras clave:

Visión por computador, interfaz gestual, reconocimiento de gestos.

ABSTRACT

In this Project we are going to aply diferents computer vision techniques in two practise cases, both of them inside de human computer interaction (HCI) area. In the first part the goal is to implement a gestual interface, where with a web-cam and our hand movements we are going to replace the mouse as a pointing device. And the second parts consits of create a knowledge base about the deaf lenguage alphabet, which is made up of some video sequences and the clasification of them. We are going to do a study also about the posibility of the implementation of a system which recognize automaticaly the signs contained on the video sequences, doing for it an analysis workin with photograms which has been took of that video sequences. To carry out these activities is necessary to know some concepts as color spaces, segmentation, gesture recognition, and another more that we are going to show along the work.

Keywords:

Computer vision, interface, gesture recognition.

1. INTRODUCCIÓN

Todos conocemos el gran desarrollo de la informática en los últimos años y hemos visto cómo el ordenador ha pasado de ser una compleja máquina manejada únicamente por expertos a ser un electrodoméstico más en el hogar, teniendo como usuarios a personas sin grandes conocimientos de informática, lo cual ha sido propiciado en gran medida por la aparición de las interfaces gráficas (GUI).

Ahora bien, lo que ya es más difícil es saber que evolución se dará en los próximos años en los ordenadores y en la forma de interactuar con ellos. Según artículos como el publicado por Julio Abascal en la revista Novatica [1], parece que nos dirigimos hacia los ordenadores ubicuos, concepto que viene a decir que los ordenadores estarán tan integrados en otros dispositivos que no seremos conscientes de ellos salvo por las funciones que ofrecen.

Viendo este cambio en el concepto de ordenadores personales a ordenadores ubicuos, el avance que se esta dando en la domótica, y el creciente uso de entornos de realidad virtual hace también que se pueda predecir un gran cambio en los interfaces, de hecho en el articulo nombrado anteriormente se habla de dos tendencias, una evolucionaria, consistente en la mejora de los sistemas de interacción actuales y otra revolucionaria, que trata de crear nuevos interfaces, cambiando incluso los dispositivos de entrada/salida. Es en este punto donde encaja este proyecto, ya que se encargará principalmente del estudio de interfaces gestuales, de su concepción y de su implementación.

Hasta este momento se ha trabajado mucho en interfaces de lenguaje natural, de hecho los sistemas actuales logran reconocer el habla con un gran porcentaje de éxito, pero el problema es la dificultad de manejar la interfaz por la voz, ya que originalmente estaba pensada para ser utilizada con otro dispositivo, clásicamente un ratón. Nuestra idea es, en vez de utilizar la voz como canal de comunicación, utilizar los gestos, ya que este además de ser un lenguaje tan o más potente que el habla, nos permite imitar

acciones que realizamos con los dispositivos de interacción actuales, léase: ratón o guante de datos. Concretamente la primera parte del proyecto consistirá en la implementación de un interfaz gestual, en el que con la ayuda de una web-cam y los movimientos de nuestra mano lograremos sustituir al ratón para el manejo del cursor de una interfaz gráfica, en este mismo sentido hay otros estudios como los que se presentan en [2,3], donde se utilizan los gestos para control de robots o diferentes aplicaciones. Para poder realizar las mismas acciones que estamos acostumbrados a realizar con el ratón como posicionarlo o hacer un clic, además de aplicar técnicas de visión para el tratamiento de las imágenes, será necesario definir una gramática de un lenguaje muy simple, su semántica, es decir, dar un significado a unos gestos, y su sintaxis, o dicho de otra forma, decir la forma en la que se utilizaran los gestos definidos para hacer uso del interfaz.

La segunda parte del proyecto consiste en la filmación de un serie de personas con conocimiento del lenguaje de los sordos signando el alfabeto dactilológico, después una serie de palabras que contienen todas las letras del alfabeto y además un serie de frases formadas por las palabras mencionadas. Los videos serán digitalizados y clasificados. Además, a partir de fotogramas de los videos, haremos un estudio sobre métodos de clasificación de los gestos contenidos en las imágenes para en un futuro poder implementar un sistema reconocedor de gestos automático, que a partir de imágenes de video donde aparece gente signando diferentes palabras y/o frases utilizando el alfabeto dactilológico del lenguaje de los sordos, generará una salida escrita donde aparecerá lo que la persona estaba signando. Esta segunda parte puede parecer poco relacionada con los interfaces, pero no es así. Lo que intentamos en este momento es dar una utilidad inmediata al reconocimiento de gestos que puede servir para la integración de un colectivo desfavorecido en la sociedad, como es la comunidad sorda. Pero realmente estos datos que se generarían como una salida escrita pueden servir en un futuro como entrada a un sistema informático y que cada signo o sucesión de estos se corresponda con una acción determinada.

2. ANÁLISIS DE ANTECEDENTES

Como ya se decía en la introducción, los sistemas informáticos están siendo testigos una gran evolución en los últimos tiempos, y cada vez las formas de interactuar con estos se han ido adecuando a las necesidades, a continuación, en vez del clásico ratón y teclado que todos conocemos, vamos a enumerar una serie de dispositivos disponibles en la actualidad con los que nos comunicamos con el ordenador. Además de los sistemas como los guantes de datos, hemos encontrado un par de sistemas novedosos, de los que uno intenta ser simplemente un sustituto del ratón, el Shotone, y otro según palabras del presidente de la empresa creadora, intenta ser el inicio de un lenguaje de gestos para la comunicación con la computadora, el iGesture.

2.1. SHOTONE

Podemos decir que este es un dispositivo inalámbrico con forma de anillo con un componente en su interior basado en tecnología CMOS, que emite señales de alta frecuencia que el ordenador captará mediante una antena conectada al puerto correspondiente al ratón y realizará las funciones de este. Mas información en http://www.noticiasdot.com/publicaciones/2003/0503/1603/noticias160503/noticias160503-19.htm.

2.2. iGESTURE

Realmente la compañía FingerWorks esta desarrollando una serie de dispositivos táctiles que reconocen el movimiento de los dedos y los transforma en órdenes para el ordenador.

El producto estrella es una especie de alfombra táctil, en la que se representa un teclado y que además hace la función de ratón, más reconocedor de una gran gama de gestos.

Figura 1. imagen del TouchStream LP

Para la escritura de texto basta con presionar sobre la posición que ocupa la letra en la alfombrilla, eso si, haciéndolo cada vez con un solo dedo. Si por ejemplo son dos los dedos que apoyamos, siendo estos concretamente el índice y corazón, pasamos a utilizar el ratón, para lo cual hay una serie de gestos definidos:

Figura 2. gestos correspondientes a acciones de ratón

Además de estos, hay combinaciones de dedos y movimientos que generan eventos para aplicaciones concretas, como photoshop, o para espacio de trabajo

concretos, como el escritorio de Windows. Veamos algunos ejemplos del lenguaje definido para este último caso:

Figura 3. gestos correspondiente a acciones propias de un entorno ventanas

En definitiva, estamos ante un periférico con gran potencia comunicativa, ya que con los diez dedos de nuestras manos, combinados además con distintos movimientos el número de comandos que podemos generar es enorme.

Personalmente me parece un sistema con un gran potencial, incluso demasiado, ya que mi duda es si un usuario puede familiarizarse con la gran cantidad de símbolos posibles, más si tenemos en cuenta que muchos de estos no son intuitivos. Otro de los problemas que encuentro es la necesidad de tener los dedos continuamente "despegados" del teclado, dudo que esta postura sea cómoda, aunque todo es cuestión de probarlo. Al margen de esto me parece un sistema muy interesante, y supongo yo, que con especial aceptación para ordenadores portátiles.

Más información: http://www.fingerworks.com/index.html

2.3. INTERFACES MULTIMODALES

Aunque cada vez los dispositivos de interacción van mejorando de forma independiente, se ve que se está avanzando hacia los llamados "interfaces multimodales", que se caracterizan por la utilización de más de un canal para la comunicación, es decir, en vez de usar simplemente un teclado este puede combinarse con el uso de la voz y de gestos, haciéndose así mas rica la comunicación entre la persona y el ordenador. Un ejemplo de sistema de este tipo es la aplicación QuickSet, que es una aplicación militar que combina el reconocimiento de voz con una entrada gestual mediante una pluma. Mediante la voz lo que se busca es dar órdenes de forma rápida y mediante los gestos se tiene capacidad de señalar objetos que estén apareciendo en la pantalla, o para diseñar rutas sobre mapas, etc.[4]

3. ANÁLISIS DE FACTIBILIDAD

Para la consecución de este proyecto son necesarias una serie de herramientas software, un hardware específico y el trato con una determinada gente. Lo ambicioso de este proyecto hacía también que tuviera un alto factor de riesgo, es por eso que también dedicaremos un punto al análisis de estos.

3.1. COMPONENETES HARDWARE

Estando este un proyecto enmarcado en el área de la visión por computador, necesitamos un ordenador con ciertas peculiaridades. Estas son que disponga de una webcam y de una tarjeta capturadora de video.

En nuestro caso disponíamos de:

- Una estación Silycon Graphics 320 (capturadora de video integrada), que fue donde se realizo la implementación del sistema, conectada
- A una cámara Sony EVI D30
- Un PC Athlon 1600, con una tarjeta capturadora Avermedia y con la misma cámara dedicado a las pruebas del sistema.

Eso por parte del hardware disponible gracias al departamento de CCIA de la universidad. Además de esto, se cuentó con:

- Un PC Athlon 1800 (256MB) con una capturadote miro PCTV propiedad del alumno
- Y una cámara de video convencional Sony Handycam (8mm) cedida por Manuel Graña, tutor del proyecto, que fueron utilizados para la grabación y digitalización de video de las personas que colaboren signando el alfabeto dactilológico.

3.1.1. SONY EVI D30

En este apartado más que las características técnicas de la cámara, vamos a hablar de las opciones de ajuste que nos da, ya que para la puesta en marcha del sistema es muy importante cuidar la calidad de las imágenes que estamos captando.

El primer valor parametrizable a destacar es la resolución. Por motivos de rendimiento se ha tomado una resolución de 320*240 píxeles.

Otro parámetro importante es la exposición. Es conveniente que esta se ajuste manualmente seleccionando los valores de apertura del iris, ganancia y shuter adecuados para que la representación del color que queremos captar sea lo más cercana posible a la realidad.

Por último y también relacionado con el color están los parámetros de brillo, contraste y saturación.

Todos estos parámetros nombrados son ajustables a partir del software que proporciona la cámara, llamado (nombre ProgramA)

Además hay que señalar que esta cámara tiene capacidad PTZ, es decir Pant (movimiento horizontal), Tilt (movimiento vertical) y Zoom. Esta característica hace que podamos ajustar el área de captación de la imagen a las necesidades del usuario sin que este tenga que moverse.

3.2. COMPONENTES SOFTWARE

Para el desarrollo del proyecto se han utilizado las siguientes herramientas software (todas ellas para Windows 2000):

- Microsoft Visual Studio 6.0: el desarrollo del proyecto se ha realizado íntegramente en C++, habiéndose elegido como entorno de programación el Visual C++.
- Librería de visión "vision software development kit": es una libraría de bajo nivel de tratamiento de imágenes de Microsoft. Proporciona una serie de rutinas y estructuras muy útiles para la adquisición de imágenes desde la cámara, para su procesamiento y para la presentación de estas al usuario. [5] bibliografía con manual de librería
- Matlab 6.0: este software ha sido ampliamente empleado para la implementación de prototipos de funciones de tratamiento de la imagen que posteriormente se han incluido en el proyecto, además de para la creación y entrenamiento de las redes neuronales utilizadas.
- Ulead Video Studio 7: utilizado para la captura y edición de video.
- Otras: para la documentación y presentación se han utilizado dos programas de Microsoft Office, Word y Power Point. Además, para entender mejor el funcionamiento de los mensajes del entorno Windows, se ha utilizado el Spy++. Y otros también obvios para la adquisición de información como Explorer o acrobat reader.

3.3. EQUIPO HUMANO

Para el desarrollo de este proyecto es necesario crear una base de imágenes y videos a partir de la grabación de personas signando el alfabeto dactilológico.

3.4. ANALISIS DE RIESGOS

Hay dos grandes fuentes de riesgo:

- La mayor es la dependencia de otra gente para lograr las imágenes de video para su posterior estudio. Esto se dice, porque ya antes del proyecto se habló con la Unión de Sordos de Guipúzcoa, y aunque parecían interesados en el proyecto no han puesto nada de su parte por colaborar en la creación de la base de datos de videos e imágenes. Esto ha llevado a tener que depender de conocidos con conocimiento del lenguaje de los sordos y estar sujetos a su disponibilidad.
- Otro riesgo es que el alumno al comienzo del proyecto no estaba en absoluto familiarizado con C++ ni con su entorno, lo que supuso un esfuerzo añadido.
- Como riesgo menor tenemos que al ir ejecutándose el proyecto surjan tareas no planificadas y este crezca desmesuradamente.

4. OBJETIVOS DEL PROYECTO

En este punto vamos a hablar del alcance real del proyecto, y este lo vamos a definir en función de los entregables que se requerirán al concluirlo. Estos son:

- Implementación y puesta en funcionamiento de un prototipo de interfaz gestual.
- Captura en video de los signos pertenecientes al alfabeto dactilológico del lenguaje de los sordos.
- Digitalización, edición y clasificación de estos videos. (se entregará CD)
- Extracción de fotogramas de videos donde aparezcan todas las letras del alfabeto.
- Estudio sobre clasificación de las imágenes del alfabeto.
- Memoria que recoja el desarrollo del proyecto
- CD con código fuente de las aplicaciones

5. DESARROLLO DEL PROYECTO

Habiendo visto ya cual es la finalidad del proyecto, sus objetivos, sus límites, etc, es hora de entrar en detalle en cada una de sus partes. Este capítulo está dividido en dos grandes apartados, correspondiendo cada uno de ellos a una de las principales actividades del proyecto, la primera, el desarrollo de un interfaz gestual y la segunda correspondiente al estudio del del alfabeto dactilológico a partir de una secuencia de video.

5.1. INTERFAZ GESTUAL

Hasta el momento, a modo de introducción, se ha hablado de un interfaz gestual en el que con ayuda de una web-cam y nuestra mano seriamos capaces de sustituir al ratón com periférico para el movimiento del cursor y generación de eventos de clic. Como elementos del sistema identificamos el ordenador, la cámara conectada a este, y el usuario. Lo que hará básicamente el sistema es estar continuamente capturando imágenes y en función de hacia donde mueva el usuario la mano en el espacio, así se posicionara o actuará el ratón en la pantalla. Viendo el siguiente gráfico correspondiente al comportamiento funcional del sistema podremos hacernos una idea más aproximada de lo que se explicaba.

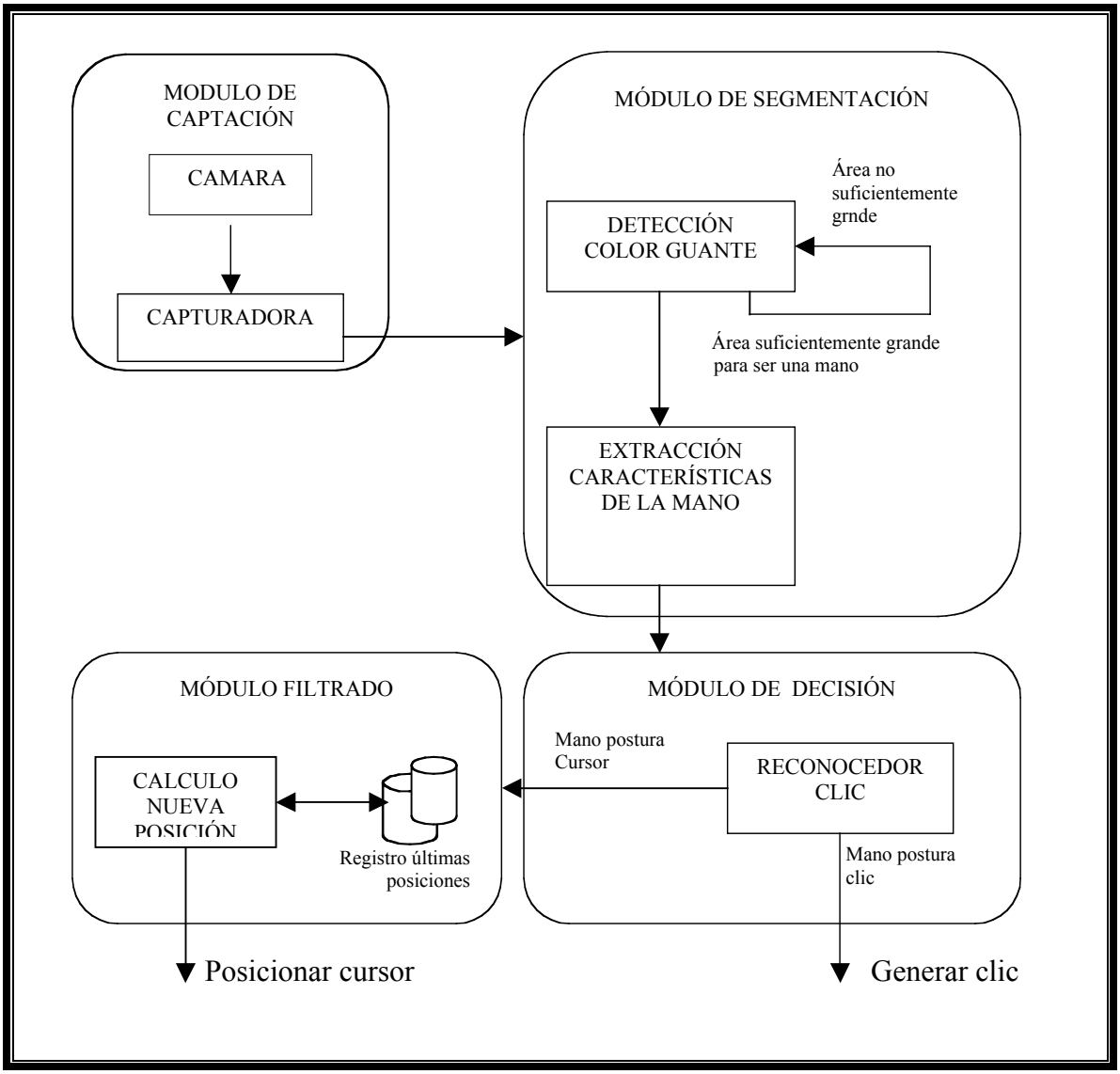

Figura 4. esquema funcional del interfaz gestual

Más adelante describiremos cada módulo del sistema, pero cualquiera de las tareas de los módulos requieren que se defina previamente una gramática del lenguaje con el que vamos a comunicarnos.

5.1.1. GRAMÁTICA DEL LENGUAJE

Para comunicarnos con el ordenador, como es lógico, debemos definir un lenguaje, su semántica y su sintaxis. Esta no es una fase trivial del proyecto, ya que dependiendo las decisiones aquí tomadas, la utilización de nuestro interfaz será más o menos cómoda.

En cuanto a la semántica, decir que nuestro lenguaje esta formado por dos símbolos, correspondientes a dos posturas diferentes de la mano. La postura indicada en la imagen izquierda de la figura 5 significa *posicionar*, mientras que la postura indicada en la otra imagen de la misma figura significa *clicar*.

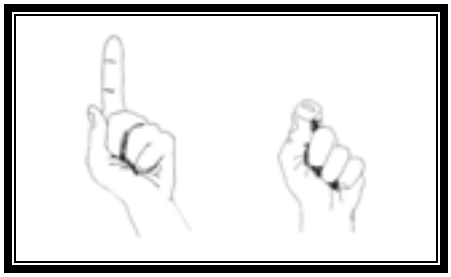

Figura 5. semántica del lenguaje

La sintaxis también será muy simple, definiendo que mientras el usuario mueva la mano manteniéndola en la postura de la imagen izquierda de la figura 5 el cursor se moverá por la pantalla en función de en que posición se detecte la mano en la imagen capturada (veremos más adelante como se calculan estas posiciones), y cuando el usuario posicione la mano como se indica en la imagen derecha de la figura 5 se generará un evento de clic de ratón en el punto en el que se encuentra el cursor en ese momento (para saber más sobre eventos de ratón y como las ventanas de la interfaz capturan y tratan estos eventos recomiendo el libro [6])

Este lenguaje se considera suficientemente amplio para el manejo de cualquier programa con una interfaz gráfica sencilla, ya que podemos posicionar el cursor del ratón sobre cualquier punto de la pantalla y clicar sobre él, pudiendo corresponder este

punto a un botón, menú desplegable, o cualquier otro elemento con los que estamos acostumbrados a convivir en las interfaces. Lo que por ejemplo no podemos hacer con la sintaxis que hemos definido es el clásico "pinchar y arrastrar", ya que no se contempla el mover el cursor del ratón mientras se genera un clic, y la semántica nos limita a la hora de generar un clic correspondiente a la oreja derecha del ratón (antes no se nombraba, paro los clics generados pertenecen a los de la oreja izquierda). Estas ideas quedan propuestas como futuras mejoras del trabajo, pero no se han realizado en este momento porque no aportan nada nuevo en cuanto a conceptos de visión por computador y reconocimiento de imágenes.

La gramática del lenguaje que acabamos de exponer es la que se a utilizado en el desarrollo del resto del proyecto, no obstante también se han estudiado otras posibilidades. Antes de documentarlas falta por explicar qué punto de la mano corresponderá realmente a el hot spot del ratón. Llamamos hot spot al punto activo del ratón en el mapa de bits de la pantalla. El punto que elijamos debe ser invariante entre las dos posibles posiciones de la mano, ya que de otro modo en la transición de un signo a otro variaría la posición del hot spot, en consecuencia la posición del ratón en la pantalla también lo haría y acabaríamos generando los clic en puntos no deseados, para verlo gráficamente observar la figura 6, en ellas se supone que tomamos como hot spot el extremo superior del dedo índice:

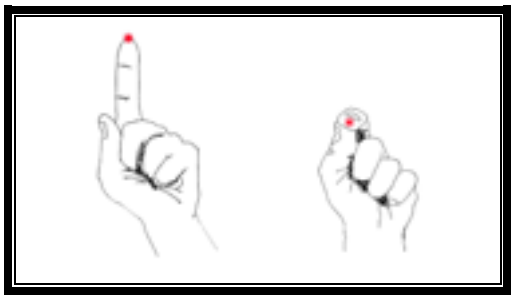

Figura 6. correspondencia entre dedo índice y hot-spot del ratón

En nuestro caso hemos tomado el centro de masa de la palma de la mano como punto característico. Otra prueba que también se realizó fue tomar como hot spot el punto situado más al extremo de la mano sobre el dedo meñique (ver figura 7) , ya que este en la gramática que hemos definimos este siempre está encogido y es también un

punto válido. Entre ambas opciones nos quedamos con la primera porque en procesos posteriores, tras la segmentación y reconocimiento de la imagen, se observaba que aunque ambos puntos estaban fluctuando constantemente, era más acusada utilizando la segunda opción.

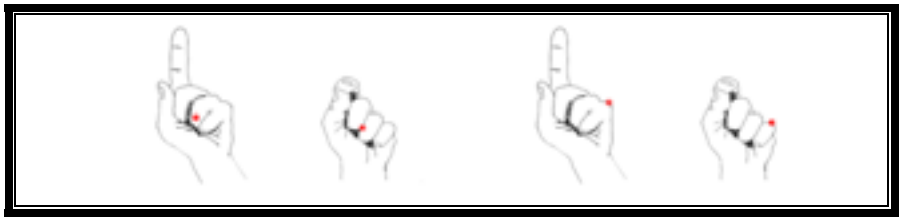

Figura 7. correspondencias entre punto de la mano y hot-spot

Retomando el tema del resto de semánticas en las que se trabajo tenemos las siguientes opciones:

a) Puño cerrado con dedo índice extendido = posicionar
   Puño cerrado con dedo índice encogido = clic
b) Puño cerrado con dedo índice extendido y pulgar encogido = posicionar
   Puño cerrado con dedo índice extendido y pulgar extendido = clic
c) Mano con la palma extendida mirando al frente = posicionar
   Mano con la palma extendida girada respecto al eje y = clic

Visualmente:

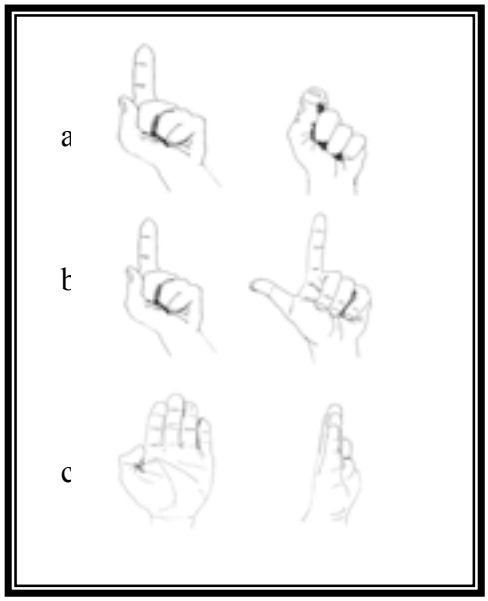

Figura 8. semántica de lenguajes explorados

La opción c fue la primera en descartarse, ya que, tomando como hot spot el centro de la mano, este método sólo funcionaría si el eje de giro de la mano coincidiera exactamente con el centro de la mano, cosa que en la práctica no ocurría. Siempre que intentábamos clicar el cursor se desplazaba a derecha o izquierda en la pantalla.

Entre la opción a y b, además de por comodidad para el usuario, nos decantamos por la opción a por características morfológicas propias de la mano que harían la fase de reconocimiento más sencilla. Concretamente se valoró que la anchura del dedo índice extendido respecto a la anchura da la mano era muchísimo menor que la proporción entre la altura del dedo pulgar extendido respecto a la altura de la mano. Cuando veamos como se localiza la palma de la mano en la imagen nos daremos cuenta de la bondad de esa decisión.

5.1.2. SEGMENTACIÓN EN BASE AL COLOR

Mediante el proceso de segmentación lo que se busca es dividir la imagen en un conjunto de regiones disjuntas atendiendo a características similares, en nuestro caso la búsqueda de la mano en la imagen la hacemos atendiendo a sus propiedades de color. Esta decisión se toma partiendo de que es una aplicación que debe funcionar en tiempo real, por lo que la segmentación debe cumplir una doble finalidad, ser suficientemente robusta para detectar continuamente la posición de la mano en la imagen, y hacerlo de forma rápida, utilizando algoritmos simples.

5.1.2.1. CONCEPTOS PREVIOS

5.1.2.1.1. LUZ Y MODELO DE IMAGEN SIMPLE

La luz es una radiación electromagnética, cuyo espectro visible se sitúa entre los 380 y 790 nanómetros, correspondientes al color violeta y rojo respectivamente.

Los objetos tienen básicamente dos propiedades, su forma, y los efectos que produce la luz al incidir sobre él. Por lo tanto en el nuestro proceso de captación de imágenes habrá una fuente de luz que emitirá una energía que será reflejada por los objetos y a su vez parte de esta energía será recibida por los sensores de la cámara. En un modelo de la imagen simple esto sería decir que la radianza de la escena, l(x,y), se caracteriza por dos componentes, la cantidad de luz que incide en la escena que se esta captando, y la cantidad de luz que reflejan los objetos. Estos dos últimos conceptos se conocen con el nombre de irradiación y reflectancia, representados mediante e(x,y) y r(x,y). Su producto define a l(x,y), es decir

$$l(x,y) = e(x,y)r(x,y)$$

donde

$$0 < e(x,y) < \infty \text{ y } 0 < r(x,y) < 1$$

Ahora lo que necesitamos es saber como se relaciona la radianza de la escena con los puntos correspondientes en la imagen, para ello utilizamos

$$E = L\frac{\Pi}{4}\frac{d^2}{f^2}\cos^4(\alpha)$$

Siendo

L=radiancia reflejada

d=diámetro de la lente

f=distancia focal (distancia al plano imagen)

α =ángulo entre el eje óptico y el rayo

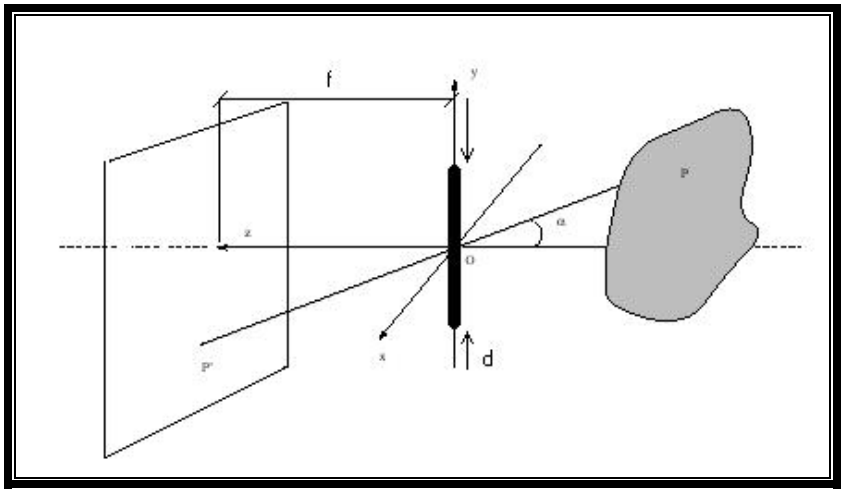

Figura 9. modelo simple de formación de imágenes

5.1.2.1.2. ESPACIOS DE COLOR

Los colores puros se definen mediante un único parámetro λ ,y los valores compuestos por radiaciones de λ diferentes deben definirse de otra forma. Se sabe que el ojo es capaz de distinguir unos 500.000 matices diferentes, entonces, ¿cómo clasificar todos esos colores perceptibles?. Atendiendo a la percepción humana, las características de un color son:

- Brillo: relacionado con la noción de intensidad.
- Matiz: longitud de onda dominante en el color.
- Saturación: Pureza relativa o cantidad de luz blanca mezclada con el matiz.

Siendo la suma del matiz y la saturación igual a la cromacidad.

Como ejemplos de espacios de color tenemos el RGB, el RGB normalizado, YUV o HSI cuyas especificaciones podemos encontrarlas en [7] .

5.1.2.2. MÉTODO DE SEGMENTACIÓN

Tomaremos la decisión del método de segmentación en base al color a partir del estudio de dos proyectos desarrollados en la UPV/EHU.

En el proyecto de Iñigo Barandiaran [8], se trabaja con dos muestras del color de la piel, una con iluminación constante y otra con cambios de iluminación. Para cada espacio de color estudia las distribuciones de distintos pares de sus componentes, calculando la media y la desviación de cada una. Con este enfoque se llegó a la conclusión de que los espacios HSV y YCrCb eran los que mejores características presentaba, seleccionando en ambos casos como variables discriminatorias las relativas a la crominancia, HS y CrCb en cada caso.

También se llegó a una importante conclusión aplicable a todos los clasificadores basados en color, y es que el sistema, para su correcto funcionamiento, no debe verse sometido a cambios bruscos de iluminación.

Para la creación del clasificador tomó como base unos estudios que mostraban como la distribución del color de la piel sigue una distribución de probabilidad normal bidimensional y como variables de esta función se tomaron aquellas relacionadas directamente con la crominancia del espacio de color seleccionado, esto es, CRCb en caso de utilizar el espacio YCrCb o HS si se utiliza el espacio HSV.

Pablo por su parte Pablo presenta una alternativa más robusta para el caso que estamos tratando que el clasificador estadístico, la Segmentación por Suma de Cubos de Color (SCC) [9].

El algoritmo SCC trabaja tomando muestras de los colores a segmentar con un tamaño de 80x60 píxeles. Una vez recogidas estas se analizan los histogramas RGB. Concretamente se ve como están distribuidos los píxeles en cada uno de los histogramas y se toman los límites de la zona donde hay una mayor acumulación de puntos

correspondientes al color. Esto se hace fijando un umbral, y tomando como puntos característicos aquellos en los que corta la línea umbral al histograma. Pongamos un ejemplo de cómo se hace el estudio para un color con una componente azul acentuada:

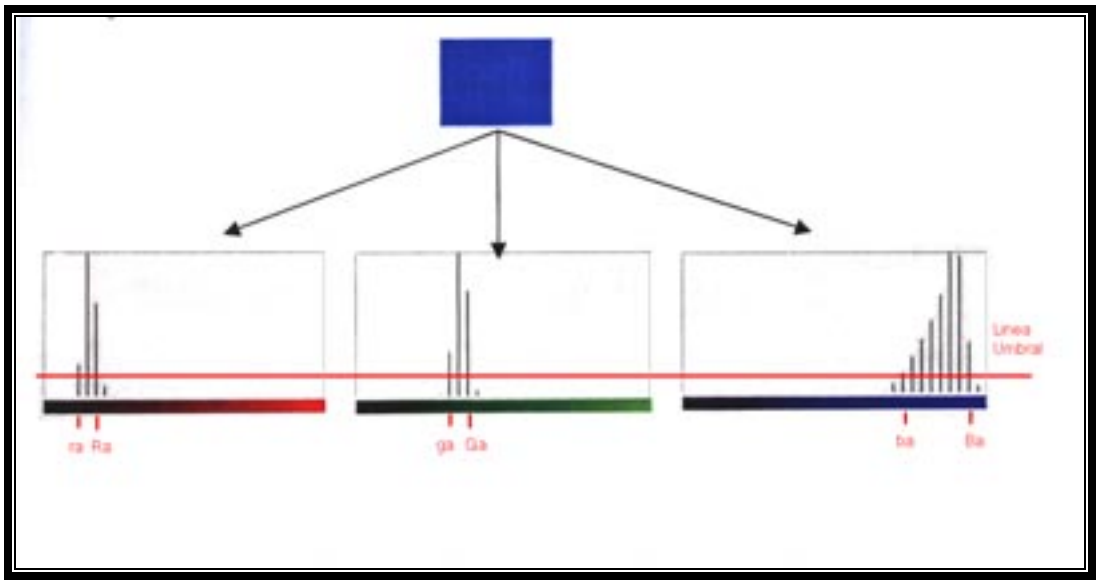

Figura 10. segmentación color azul usando SCC

Una vez tenemos los intervalos para cada componente de color, utilizaremos estos como umbrales para clasificar píxeles, clasificándolo como miembro de una clase si sus valores RGB están dentro de los intervalos de una muestra. Pongamos un ejemplo en el que queremos ver si un píxel corresponde a un color azul o a uno verde.

Sean R_p, G_p, y B_p los valores RGB del píxel.

Sean r_a, R_a, g_a, G_a, b_a y B_a los valores obtenidos para una muestra azul.

Sean r_v, R_v, g_v, G_v, b_v y B_v los valores obtenidos para una muestra verde.

Para clasificar cada píxel:

Si ($R_p \in \{ r_a, R_a \}$ y $G_p \in \{ g_a, G_a \}$ y $B_p \in \{ b_a , B_a \}$) entonces marcar pixel azul

Si ($R_p \in \{ r_v, R_v \}$ y $G_p \in \{ g_v, G_v \}$ y $B_p \in \{ b_v , B_v \}$) entonces marcar pixel verde

En otro caso el píxel no forma parte de ninguno de los dos colores.

Con los colores que realizó Pablo las pruebas se observó que con una sola muestra de cada color era pequeño el número de píxeles reconocido, esto se debía a que la luz modificaba el color de las cartulinas usadas como muestras en función de su ángulo de incidencia. Este fenómeno unido a la ausencia de falsos positivos le llevó a ampliar el rango de colores que pudieran ser reconocidos como azul o verde en este caso.

Para aumentar dicho rango se capturan varias muestras de cada color colocando el la muestra en distintos ángulos respecto a la cámara.

Con esta mejora el resultado fue muy satisfactorio, ya que con tres muestras o cinco en función de si la iluminación era artificial o natural respectivamente, se lograron clasificar los colores de una forma robusta.

En el caso de utilizar tres marcadores de color (rojo, azul y verde) esta es la representación espacial de los umbrales obtenidos:

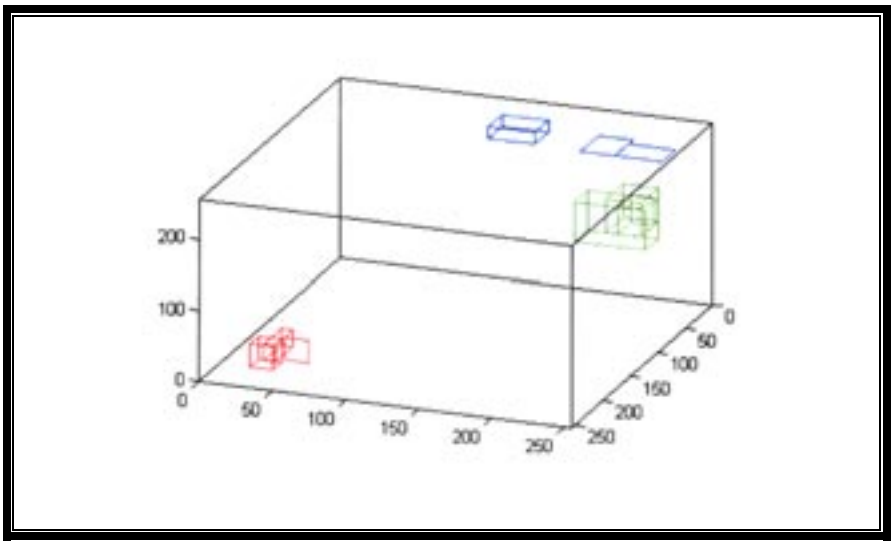

Figura 11. Representación gráfica SCC

Vemos se forman cubos o planos en el espacio RGB en función del número de componentes que tengan las muestras. En este caso concreto podemos observar como el color azul utilizado como muestra no es puro, ya que su componente G es elevada.

Concluyendo con ese estudio se tiene que dependiendo del número de muestras de cada color el rendimiento se ve afectado de la siguiente forma:

Número de muestras	1-3	4-7	8-13	14-21	22-25
Imágenes por segundo	23	22	21	20	19

Figura 12. Tabla. Rendimiento segmentación SCC

Como se puede observar nos hemos extendido más en la explicación del algoritmo utilizado por Pablo para la segmentación, y esto es debido a que es este el que vamos a utilizar en nuestro proyecto. También se probaron otras opciones como la segmentación por umbral absoluto para comparar resultados, y en cualquier caso estos fueron muy pobres. Un factor determinante para la elección, además del mejor rendimiento del algoritmo presentado por Pablo, es que si utilizáramos el primero, tendríamos más problemas para localizar la mano en imágenes en las que también aparezca el rostros del usuario, brazo, o se encuentre alguien tras nosotros, mientras que con el segundo método utilizando un guante de color solucionaríamos todos los males, eso si, teniendo en cuenta que este color no se encuentre en el fondo o en al ropa del usuario. Conclusión, introducimos una restricción al sistema, consistente en la utilización de un guante de color (negro en nuestro caso), y la ausencia de este en el fondo para el correcto funcionamiento del sistema.

La elección de este color negro en nuestro caso además ayuda a que el algoritmo trabaje bien con una sola muestra del color, ya que absorbe toda la radiación de luz y no produce reflejos.

5.1.3. RECONOCIMIENTO DE LA MANO

Sabiendo que aplicando a las imágenes el algoritmo de segmentación explicado en el apartado anterior de ya somos capaces de clasificar los píxeles como pertenecientes o no al color de nuestro guante, ahora es necesario reconocer exactamente la posición que ocupa nuestra mano en la imagen.

Teniendo en cuenta que el color del guante que estamos utilizando es único en la escena, una forma sencilla de dar con la posición de la mano es aplicando las proyecciones por filas y por columnas de los píxeles correspondientes a su color, teniendo así la distribución por filas y columnas de estos píxeles. Las siguientes expresiones corresponden a las proyecciones.

$$P\_Filas(i) = \sum_{x=1}^{n} imagen(x,i)$$

$$P\_Columnas(i) = \sum_{y=1}^{m} imagen(i,y)$$

Figura 13. fórmula de cálculo de proyecciones

Recordemos que nuestro lenguaje definía el centro de masa de la palma de la mano cerrada como hot spot del ratón, para encontrar este lo que hacemos es, una vez calculadas las proyecciones, recorrer ambos vectores, el correspondiente a las filas y el correspondiente a las columnas, buscando el primer valor que supere un cierto umbral (parametrizable en nuestra aplicación) y tomando el índice del lugar que ocupaba ese valor en el vector como comienzo del área de interés. Tras esto seguimos avanzando en el recorrido de los vectores hasta encontrar otro valor inferior al mismo umbral, teniendo así ya el final del área de interés (en caso de no encontrarse un valor inferior al del umbral se toma como índice el correspondiente a la última fila o columna de la imagen). Con estos índices logramos calcular la altura y la anchura del área de interés, y hay que decir que ambos deben superar también unos ciertos umbrales, ya que un área

muy pequeña puede hacer complicado un posterior reconocimiento de la postura de la mano, y aún más importante, como nos daremos cuenta más adelante, que la mano ocupe una parte tan pequeña de la imagen se debe a que el usuario estará sentado muy lejos de la cámara o no habrá hecho un adecuado uso el zoom y no será capaz de posicionar la mano de forma cómoda en todos los puntos de la imagen que tengan correspondencia con la pantalla.

A continuación se muestra una representación gráfica de este proceso de localización del área de interés.

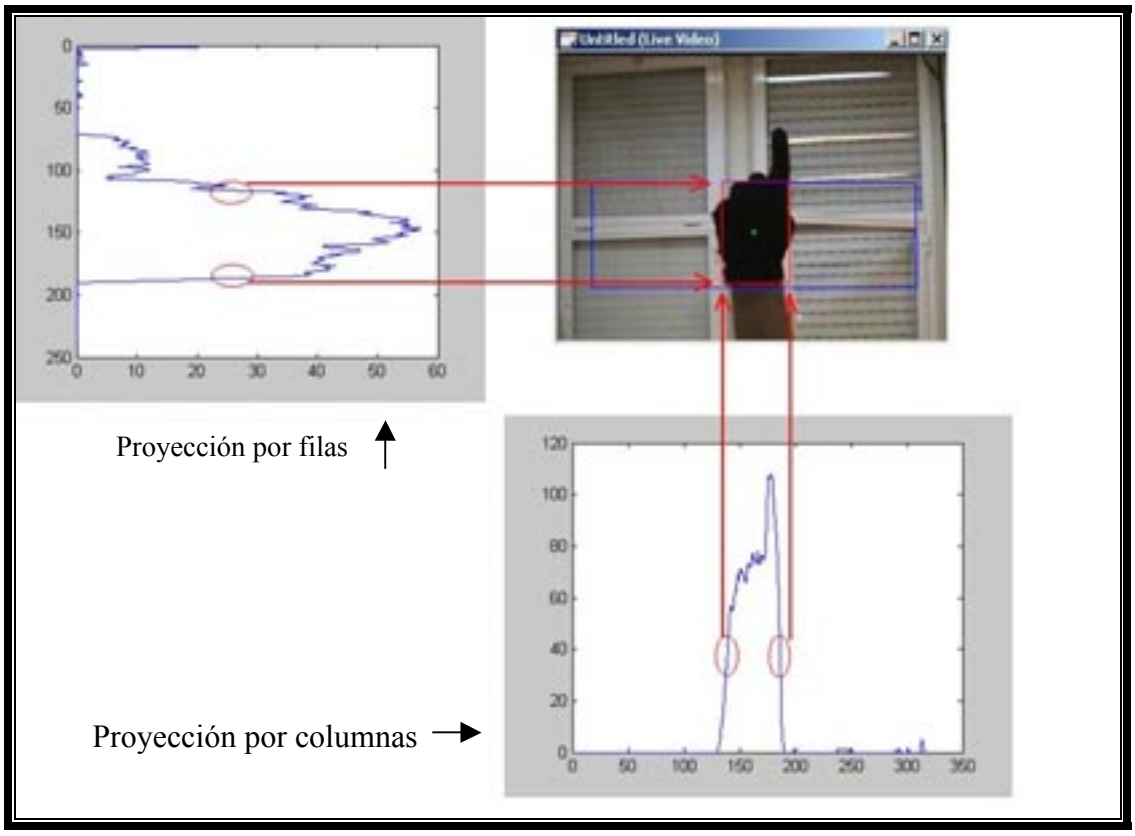

Figura 14. mano con sus proyecciones, e índices

Las pruebas que se realizaron con el algoritmo de las proyecciones presentó algunas debilidades. Básicamente el ruido recogido por la cámara o cualquier otro factor que hiciera fallar mínimamente a la segmentación, hacía que la localización de la mano en el eje *y* no fuera buena en ocasiones. Viendo las proyecciones en la figura 14 podemos observar que en la proyección por columnas, bajo los picos de las

proyecciones hay un área más o menos grande, mientras que en la proyección por filas esta es menor, y si en el recorrido del vector de proyecciones caemos en algún mínimo local antes del final real de la mano, producido por esos errores en la segmentación que comentábamos, ya estamos tomando un índice erróneo como referencia.

Para solucionar esto se optó por aplicar un filtro de suavizado al vector de la proyección por filas. Su funcionamiento es muy simple, lo que hacemos es recorrer el vector de la proyección por filas con una ventana de tamaño 19 (impuesto experimentalmente para un tamaño de imagen de 320*240, que es con el que se ha trabajado) y sustituimos el valor del centro de la ventana por la media aritmética de todos los elementos de la ventana. Este es el código del filtro implementado en matlab que va dejando en el vector suave el resultado del suavizado.

```
function suave=suavizado(proyeccion)
long=size(proyeccion,2);
%tratamiento de las primeras posiciones
for i=1:9
   suma=0;
   for k=1:i
      suma=suma+proyeccion(k);
   end
   for l=i+1:i+9
      suma=suma+proyeccion(l);
   end
   suave(i)=suma/(i+9);
end
%tratamiento del centro
suma=0;
for w=1:19 //suma de las primeras 19 posiciones
   suma=suma+proyeccion(w);
end
```

```
suave(10)=suma/19; //media de las 19 primeras posiciones
for e=20:long //desplazamiento de la ventana mientras hay elementos por tratar
   suma=suma-proyeccion(e-19)+proyeccion(e); //calculo de nuevo valor
   suave(e-9)=suma/19; //asignación de la media
end
%fin centro
%tratamiento del final
for b=(long-8):long
   suma=0;
   for n=(b-9):long
      suma=suma+proyeccion(n);
   end
   suave(b)=suma/((long-b)+10)
end
```

(codigo filtro de suavizado)

Y estos son sus resultados.

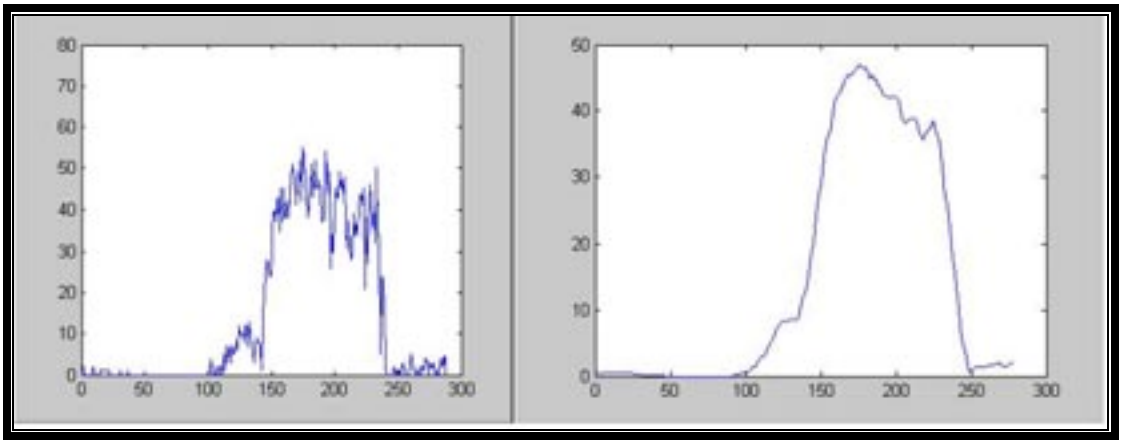

Figura 15. imágenes de proyección por filas y misma proyección suavizada

El filtro dio muy buen resultado, y aunque no recarga el sistema en exceso se puede prescindir de este ajustando bien los umbrales de las proyecciones, cosa que podemos hacer ya que todos estos valores son parametrizables desde la aplicación en

tiempo de ejecución, cosa lógica si pensamos que el interfaz debe funcionar en escenarios con diferentes iluminaciones, diferentes cámaras, diferentes usuarios...

5.1.4. CORRESPONDENCIA IMAGEN-PANTALLA

Un tema que a priori puede pasar por alto es como haremos corresponder los puntos de la imagen que recogemos con los puntos de la pantalla del ordenador, más concretamente el punto que nos interesa es el correspondiente al hot spot el ratón. El usuario, al mover la mano libremente en el espacio puede acercar y alejar la mano de la cámara, y nuestro interfaz debe adaptarse a estos cambios. Además hay que tener en cuenta que tenemos que ser capaces de reconocer la mano tanto con el dedo extendido como encogido en todo momento, lo cual hará también que perdamos una parte de la imagen como zona útil.

Imaginémonos las siguientes situaciones:

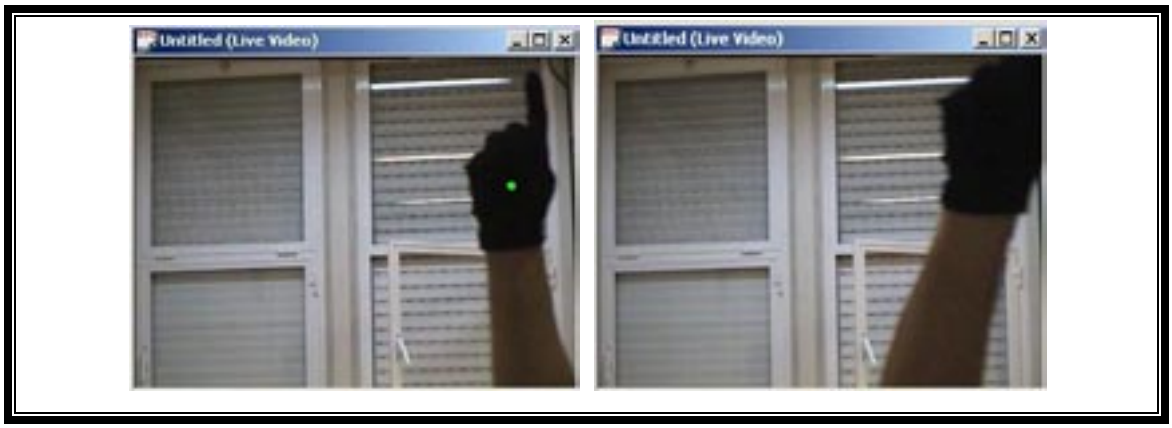

Figura 16. mano completa en la parte superior derecha de la foto y mano "cortada" en la imagen

En ambas imágenes aparece una mano con el guante de color, el problema es que en la imagen derecha de la figura 16 no aparece la mano completa, con la consecuente imposibilidad de ser reconocida. Es este suceso precisamente el que nos lleva a pensar en que sólo habrá una parte de la pantalla en el que podamos posicionar la mano y reconocerla como tal. Además, aún estando completamente dentro de la imagen, fijándonos en la imagen izquierda de la figura 16 vemos como al tomar el centro de la

mano como hot spot del ratón, nunca podremos posicionar este punto más arriba y a la derecha que el punto verde que se muestra en la imagen como centro de la mano. En el ejemplo hemos tomado como muestra los límites superior y derecho, pero lo mismo ocurre para la zona inferior e izquierda de la imagen, en todas ellas habrá una serie de puntos en las que nunca podremos posicionar el centro de la mano mientras esta es reconocida. Por todo esto, lo que hacemos es tomar un área de la imagen como parte con correspondencia con puntos de la pantalla. Este área es calculado de acuerdo a la morfología de nuestras manos, y lo que hacemos es poner unos márgenes a derecha e izquierda de la imagen restándole una longitud igual a la mitad de la anchura de la palma de la mano, restamos también la mitad de la altura de la palma de la mano a la parte inferior de la imagen y a la parte superior le restamos 2.5 veces la mitad de la altura. Todas estas explicaciones se ven mejor gráficamente en la figura 17.

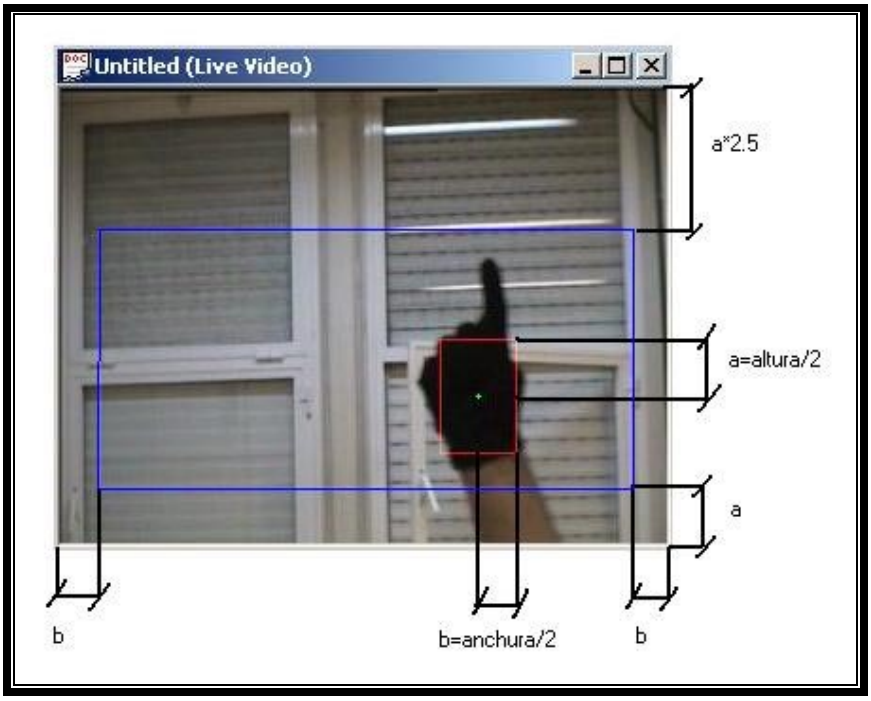

Figura 17. área enmarcada en cuadro azul = área de la imagen con puntos con correspondencia en pantalla

Ya que el usuario no moverá la mano siempre en un mismo plano imaginario del espacio, hará que estas variaciones de distancia entre la mano y la cámara se reflejen en la imagen como cambios en el tamaño de la mano. Estos cambios de tamaño provocarían alguna de las siguientes situaciones:

a) Con el nuevo tamaño de mano, en caso de que este sea menor, estamos posicionando el cursor en puntos que antes no podíamos hacerlo. (desaprovechamos una zona que este momento sería útil)
b) En caso de que el nuevo tamaño fuera mayor, habría puntos en los extremos en los que ya no podríamos posicionarnos y antes sí. (Dejando de poder mover el cursor del ratón por toda la pantalla)

Nuestro interfaz se adaptará a estos cambios dinámicamente, por ello recalcularemos periódicamente el área de imagen con correspondencia con puntos del monitor, teniendo así en todo momento la resolución máxima que podemos tener en la zona de imagen con correspondencia. Esta resolución por otro lado conviene que siempre sea la mayor que podamos, ya que cuanto más grande, más suaves serán los movimientos que generemos con el ratón. Una vez más buscaremos un equilibrio entre la distancia a la que situaremos la mano para que el movimiento sea fluido, y a la vez no sea incomodo abarcar toda la imagen con los movimientos de nuestra mano. Este equilibrio es el que se buscaba cuando en el apartado anterior hablábamos del umbral del tamaño de la mano en la imagen. Las tres fotos de la figura 18 reflejan la mano a una distancia normal a la cámara (medio), una distancia muy lejana con la que es incómodo trabajar por los amplios movimientos que debemos realizar para mover el ratón (derecha), y una distancia excesivamente cercana, que deja un área de correspondencia tan pequeña que no tenemos ninguna precisión en los movimientos (izquierda).

Figura 18. imagen con mano a distancia normal, a distancia muy lejana y a distancia muy cercana

Cundo ya tenemos el área de correspondencia calculada, y en la captura de una imagen localizamos el centro de la mano en una posición determinada, lo que tenemos que realizar es una conversión de coordenadas. Es debido a que el área de imagen con puntos con correspondencia nunca tiene la misma resolución que la pantalla. Esto se da porque la resolución de captura de las imágenes es menor que la de cualquier monitor de un PC, además sabemos que la parte con puntos con correspondencia es aún menor, y para colmo esta área de la imagen de puntos con correspondencia se recalcula constantemente mientras que la resolución del monitor no suele variar.

Antes de explicar como se realiza la conversión es preciso matizar ciertas peculiaridades de Windows.

a) La primera es que el origen del sistema de coordenadas de la pantalla se encuentra en la esquina superior izquierda.
b) La segunda es que sea cual sea la resolución que estemos utilizando para el monitor, Windows mapea internamente la pantalla como una matriz de 65535 x 65535 elementos.

La conversión se realiza mediante los siguientes cálculos.

```
mediox = (65535-((medioc/2)+inicioc-desplazamientoc)*multiplicac);
medioy = (finf-(mediof/2)-(desplazamientof*2.5))*multiplicaf;
```

(código comentado sobre cálculo de área con correspondencia y posición de centro de la mano y correspondencia en pantalla)

Para entender el código hay que señalar que la variable multiplicac es el factor de escala del eje x, este valor se calcula cuando decíamos que cada 20 imágenes tratadas se recalculaba el área de la imagen con puntos de correspondencia en la pantalla. Así mediox se calcula como 65535, que es el número correspondiente a la última columna de la pantalla, menos el punto correspondiente al centro de la mano en el eje x (inicioc es el índice de inicio de la mano, a lo que se le suma la mitad de su anchura) menos el

desplazamientoc, que es otro valor calculado durante el proceso de calcular el área de puntos con correspondencia, y que es equivalente a la mitad de la anchura de la mano en el momento que se realizó el ajuste, y todo esto por el factor de escala del eje X para transformar este punto de la imagen en un punto de la pantalla. La resta de 65535, menos el valor calculado es así porque al captar las imágenes, esta se ve como reflejada en un espejo, de forma que cuando el usuario desplaza la mano hacia la derecha porque quiere mover el ratón en esa dirección, en la imagen realmente se esta viendo la mano desplazarse a la izquierda y la columna punto más a la izquierda de la imagen será la que corresponda con el extremo derecho de la pantalla. El cálculo de medioy es muy similar, salvo que en vez de restar, estaríamos sumando a la fila inicial de la pantalla (fila 0) el valor que calculemos, y desplazamientof esta multiplicado por 2.5 porque a la parte superior de la pantalla se le ha restado esta área como puntos sin correspondencia en la pantalla en el momento de calcular el área con puntos con correspondencia.

5.1.5. FILTROS DE MOVIMIENTO.

Aunque a estas alturas ya somos capaces de mover el cursor por la pantalla con relativa fluidez, nos damos cuenta que el problema es la precisión en el posicionamiento. Si utilizamos para posicionar el ratón en la pantalla los datos obtenidos directamente del apartado anterior, vemos como por fluctuaciones en la detección de los extremos de la mano y los cambios de resolución al recalcular el área con puntos de la imagen con correspondencia en la pantalla, aunque nosotros dejemos la mano lo más quieta posible, el cursor está continuamente oscilando. Este tembleque hacia que posicionarnos sobre menús con precisión o pulsar botones de pequeño tamaño fuera una misión prácticamente imposible. Por esto se desarrollaron una serie de filtros.

5.1.5.1. FILTRO EN FUNCION DE LA DIRECCIÓN

Ya que la naturaleza del ruido indicaba que el ratón oscilaba continuamente en cualquier dirección, se pensó en un filtro muy simple, que trataba de generar un evento de posicionamiento del ratón sólo si la dirección del movimiento del ratón era similar en momentos sucesivos. Esto es, si el ángulo formado por la dirección que se llevaba en el momento t-1 y la dirección hasta el momento actual t es menor a un cierto umbral, entonces se toma el movimiento como válido y se posiciona el cursor en la nueva coordenada.

Para realizar este filtro es necesario conocer algunos conceptos básicos sobre geometría analítica plana, para ello esta muy bien la consulta de la página web bleblebla incluida en la bibliografía.

Retomando la cuestión, a medida que vamos tratando imágenes podemos aproximar la trayectoria del ratón a una sucesión de rectas. Ahora es cuando introducimos la condición para considerar que el movimiento es producto de un movimiento voluntario del usuario y no del ruido, esto se da si el ángulo formado por las dos o tres (parametrizable en la aplicación) últimas rectas que modelan el movimiento del ratón no superan en ningún caso un cierto umbral (umbral también parametrizable desde el interfaz de la aplicación). Para hallar el ángulo que forma cada par de rectas hacemos lo siguiente. Colocamos la posición del cursor en el instante t-2 en el origen de coordenadas de una base (normal?ortonormal?ortogonal?), y sobre esta base colocamos también los puntos correspondientes al instante t-1 y t (puntos rojos).

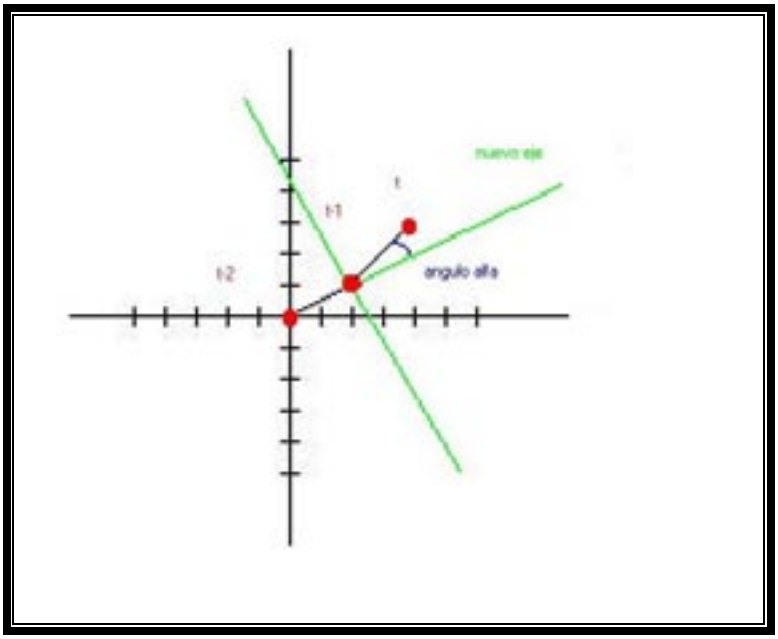

Figura 19. grafico de ángulos y bases

Una vez situados estos puntos, calculamos una nueva base formada por el primer vector (t-2,t-1) y un vector normal de este, ya que así tendremos los vectores linealmente independientes necesarios, (en verde el nuevo eje de coordenadas) calculado como:

> Estando la recta r definida por los puntos (b,d), el vector (d,-b) es normal a r ya que es perpendicular a su vector de dirección:
>
> (d,-b).(b,d)=db-bd=0

Figura 20. fórmula calculo vector normal para nueva base

y calculamos el ángulo que forma el vector (t-1,t) respecto al eje x de la nueva base sabiendo que $tg\theta = {}^{y}\!/_{x}$ (ángulo dibujado en azul).

Para considerar que la trayectoria esta siendo medianamente uniforme, el vector (t-1, t) en la nueva base debe estar situado en el primer o cuarto cuadrante, esto es, el ángulo formado con el eje x nunca deberá ser mayor de 90º, aunque podremos restringir más este parámetro, ya que el interfaz nos da la opción de hacerlo.

El código en matlab que sirvió como prototipo para esta función es:

```
function nuevaCoordenada=cambioBase(Coordenada,vectorBase)
//coordenada es la correspondiente al instante t, y vectorBase en el instante t-1
baseAlfaX=vectorBase(1);
baseAlfaY=vectorBase(2); //separamos cada elemento de los parámetros
baseBetaX=baseAlfaY*(-1);//calculamos vector normal
baseBetaY=baseAlfaX;
//con estos base alfa y beta ya tenemos losvectores linealmente independientes
//que definen la nueva base
X=Coordenada(1);
Y=Coordenada(2);
nuevoX=X-baseAlfaX;//x posicionado en el origen de coordenadas
nuevoY=Y-baseAlfaY;//y posicionada en el origen de coordenadas
//hallamos coordenadas correspondientes a nuevos y Y en la nueva base. alfa=x y
//beta=y. Esta cuenta es equivalente a resolver el sistema de dos ecuaciones condos
//incógnitas resultantes de:
//(coordenadaNueva)=alfa(vectorBase1)+beta(vectorBase2);
 beta=(nuevoY-(baseAlfaY*(nuevoX/baseAlfaX)))/((baseAlfaY*(((-
1)*baseBetaX)/baseAlfaX))+baseBetaY);
 alfa=(nuevoX-(baseBetaX*beta))/baseAlfaX
(coordenadaNueva)=alfa(vectorBase1)+beta(vectorBase2);
//calculamos el angulo que forma la coordenada con el eje x de la base
if(alfa > 0) //valor de primera coordenada positiva, inidca que estamos en el
        //primer o cuarto cuadrante
   angulo=abs(((atan(beta/alfa))*(180/pi)));
```

```
else
   angulo=180 //devolvemos un valor superior al permitido por cualquier umbral
end
nuevaCoordenada=[alfa beta];
```

En la siguiente imagen reflejaremos ejemplos de cada una de las posibilidades del filtro, considerando dos o tres últimas trayectorias

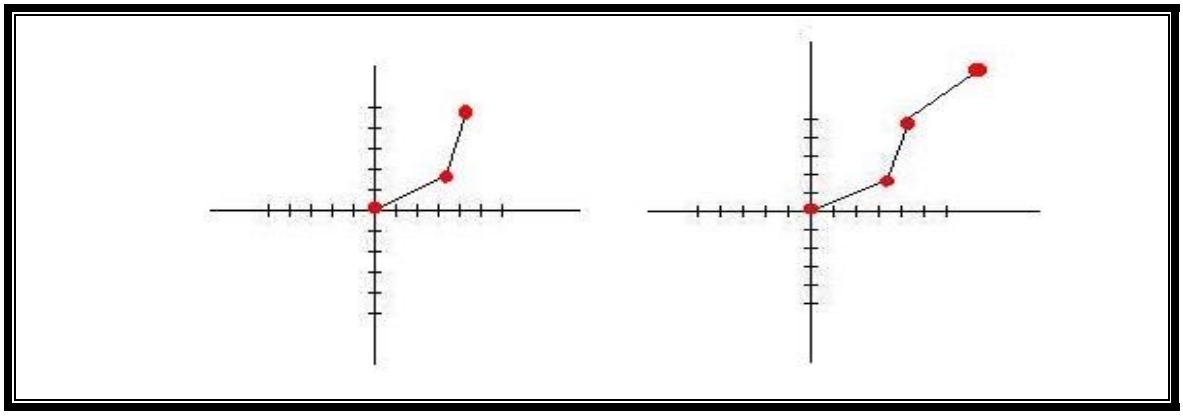

Figura 21. imagen de dos rectas formando un ángulo bueno y trayectoria de dos rectas

Con este filtro se intenta que en situaciones como la que se recoge en la figura 22 se identifiquen como producto del ruido y el cursor en vez de estar continuamente moviéndose se quede fijo en el último punto en el que el usuario lo ha posicionado intencionadamente.

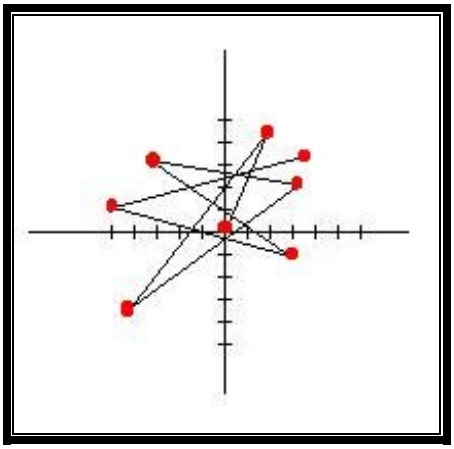

Figura 22. movimiento producido por el ruido

El funcionamiento de este filtro no fue nada bueno, pues hacia que el ratón se desplazara a saltos. Esto es debido a que al tratarse cerca de 20 imágenes por segundo, el desplazamiento de la mano entre una y otra es muy pequeño, por ello el ruido hace que en estos pequeños desplazamientos sea difícil de mantener una trayectoria más o menos uniforme.

5.1.5.2. FILTRO EN FUNCIÓN DE LA DISTANCIA

Otra forma de filtrar el ruido introducido inherentemente por el proceso de segmentación es aplicando las siguientes fórmulas de suavizado:

$$P_t = (x, y)$$
$$\Delta P_t = (\Delta x, \Delta y)_t$$
$$P_t = P_{t-1} + \alpha \Delta P_{t-1}$$

Figura 23. formula filtro

Modelando cada estado como la posición (x,y) que ocupa el hot spot del ratón en un momento determinado y definiendo la variación entre estados como el cambio en sus coordenadas x e y, tenemos que podemos filtrar el ruido posicionando el ratón, no exactamente en el punto de la pantalla correspondiente al centro de la mano, sino en el punto correspondiente al estado anterior más una variación de este en función a la distancia a la que nos hemos movido realmente. Aquí juega un papel fundamental el parámetro alfa, α, con el indicamos en que porcentaje afecta el nuevo estado al anterior. Poniendo un α elevado, como pudiera ser 0.9, estamos haciendo que la propiedad de filtrado prácticamente desaparezca, ya que estamos tomando como punto actual el anterior más casi todo el desplazamiento que se ha dado en el movimiento, mientras que si tomamos un α muy pequeño, con un valor cercano al cero, como 0.1, estamos haciendo que los incrementos de movimiento sean tan pequeños que el ratón tardará muchísimo en llegar al lugar deseado, haciendo así inútil el interfaz.

Primeramente se implementó una versión del filtro en la que tomábamos un único alfa, y esto hacia que si tomábamos como valor de alfa un número muy pequeño,

aunque dejando la mano quieta el cursor no sufría prácticamente ninguna oscilación, que era justamente lo que queríamos lograr, al intentar desplazarnos a puntos lejanos al actual el cursor se movía tan lentamente por la pantalla que hacia difícil calcular donde acabaría parándose y la velocidad era tan escasa que llevaba al usuario a la desesperación. También se podía tomar como alfa un valor mayor, pero entonces los efectos serían que se perdería la propiedad de filtrado.

Para solucionar esto se optó por implementar una segunda versión en la que se diferenciaran los movimientos con pequeñas variaciones de posición, que son los que una persona utiliza cuando esta buscando una cierta precisión y los que también se producen por el ruido, y los movimientos con gran variación en la posición, que son utilizados por el usuario cuando quiere ir de un lado de la pantalla al otro rápidamente. El siguiente código es el correspondiente a esta segunda versión del filtro.

```
if ((deltay+deltax)<(umbralDistancia))
    alfaMov=alfa1;
 else
 alfaMov=alfa2;
 posicionX=medioxAnterior1+(alfaMov*deltax);
 posicionY=medioyAnterior1+(alfaMov*deltay);
```

(codigo de filtro)

Como vemos, estamos diferenciando el movimiento en función de la distancia a la que queremos movernos, así si buscamos la precisión en el movimiento entramos en la primera sección del código donde tendremos parametrizado el (alfa) con un valor pequeño para que ejerza bien como filtro, mientras que si queremos desplazarnos entre puntos distantes entraremos en el segundo bloque de código para tener una rápida respuesta en el movimiento del ratón. Este segundo filtro dio buenos resultados, aunque cuando nos movemos notamos como se da una cierta inercia en el movimiento, producida porque según se acerca el cursor al punto en el que queremos posicionarnos, llega un momento que la distancia es tan pequeña que entramos en la primera parte del código, con el consecuente cambio de velocidad en el posicionamiento del cursor

motivado por el parámetro alfa más pequeño. Esto es, realmente lo que se da no es una inercia en el movimiento, sino una deceleración en el movimiento a medida que nos acercamos al punto en el que queremos posicionarnos. Los resultados de las pruebas indicaron que esta deceleración, aunque en un principio resultaba un poco incomoda para el usuario, este enseguida se acostumbraba y controlaba el cursor perfectamente

5.1.6. RECONOCIMIENTO DEL CLIC

Como ya se definió en la gramática, nuestro interfaz podrá generar eventos correspondientes al clic del ratón. Recordemos que estos se generarían encogiendo el dedo índice partiendo de la postura general con la mano cerrada y únicamente con el dedo índice extendido. Como se ve el parámetro que varía en la generación de un clic es la altura de nuestro dedo índice y las técnicas de reconocimiento del clic se basan en detectar este decrecimiento de su altura, aunque cada una lo hace de un modo.

5.1.6.1. RECONOCIMIENTO DEL CLIC MEDIANTE DECREMENTOS SUCESIVOS DE LA ALTURA

Este método de reconocimiento lo que hacia era considerar que el usuario estaba generando un clic cuando la longitud de la altura de su dedo había ido decreciendo en cinco capturas de imágenes consecutivas y su longitud final llegaba a ser 2/3 de la inicial.

Para calcular la altura del dedo en cada imagen, lo que hacemos es, basándonos en la definición de píxel vecino, calcular el píxel de mayor altura del color del guante y conectado a este.

En el párrafo anterior hemos introducido dos conceptos, píxel vecino y componente conectado. Los vecinos de un píxel están en filas y columnas adyacentes y podemos definir dos tipos de vecindario, 4-vecindario, que es aquel formado por los

píxeles que se encuentran a la derecha izquierda, arriba y debajo de uno, y 8-vecindario, donde lo que hacemos es añadir al 4-vecindario los píxeles que están inmediatamente en la diagonal superior derecha e izquierda y en las diagonales inferiores.

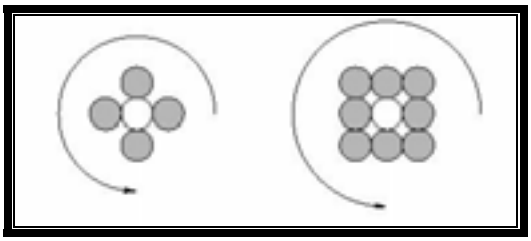

Figura 24. figura de vecindarios

En cuanto a la conectividad, también tenemos dos definiciones. Llamamos región 4-conexa a aquel conjunto de píxeles en el que cualquier par de ellos pueden ser unidos por un camino de 4-vecinos pertenecientes a la región. Análogamente tenemos la definición de región 8-conexa.

Tomando como referencia un proyecto del grupo de procesado de señal y simulación de la Universidad Politécnica de Madrid [10] y tras revisar los algoritmos de llenado y separación de regiones, útiles para calcular la altura del dedo, se vio la complejidad de estas tareas y su costo computacional. Finalmente se optó por aportar una solución propia, que aunque no es tan robusta como los métodos detallados en la memoria, si es más rápida y da buenos resultados aplicada a nuestro contexto de trabajo. El algoritmo lo que hace es, partiendo de la fila más alta reconocida como mano, para cada uno de sus píxeles va mirando si el superior derecho, el superior, o el superior izquierdo es del color del guante, y de ser así lo marca como perteneciente a este. Este proceso se repite mientras haya filas con elementos conectados, y finalmente se da como altura esa línea en la que se ha parado el algoritmo menos la línea perteneciente a la base del dedo.

Esta una representación del resultado obtenido a partir de una imagen cualquiera de ejemplo, donde se ha simbolizado con unos los píxeles reconocidos como conectados al guante y cero los que no.

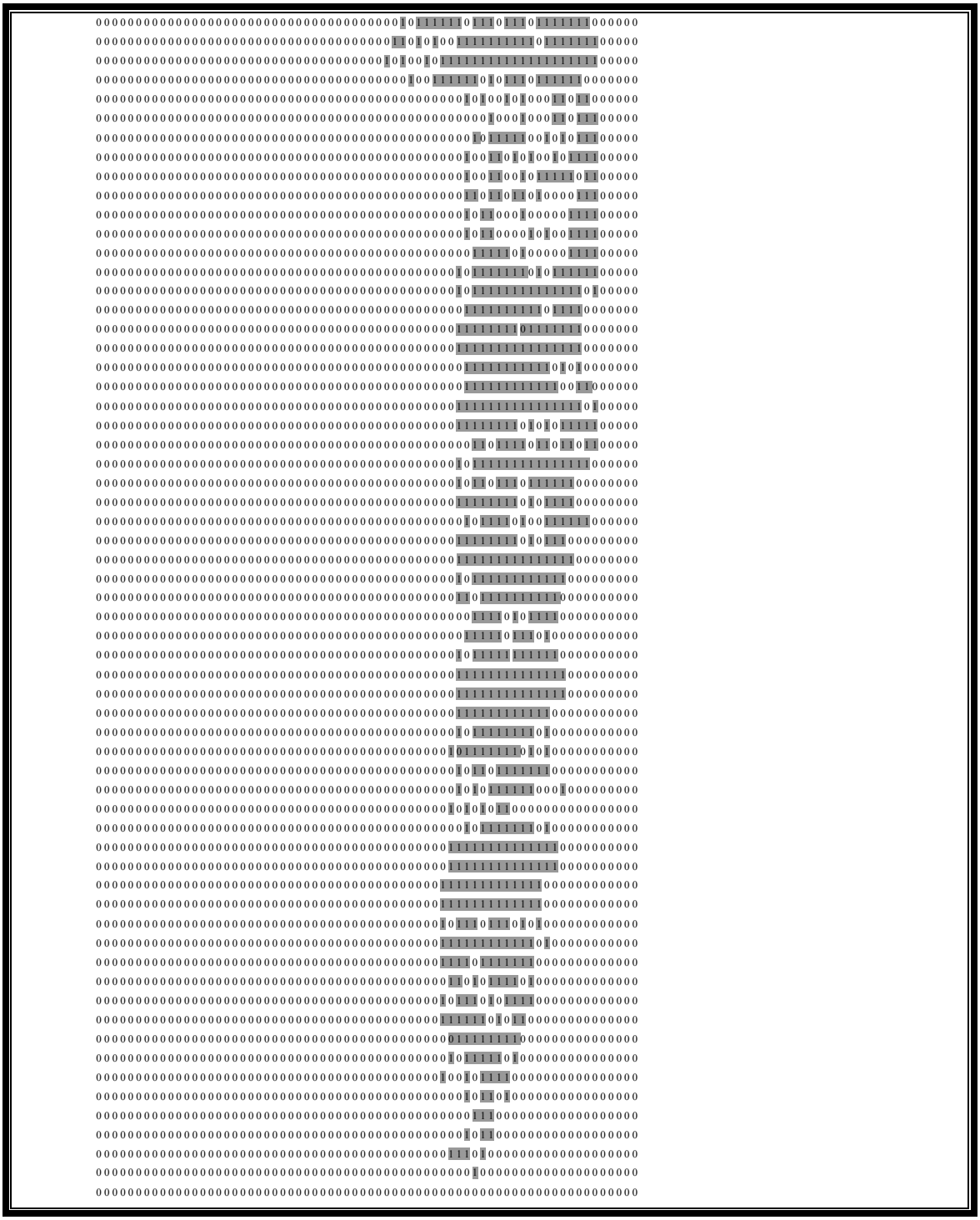

Figura 25. matriz de unos representativo de dedo

Podemos ver como resaltado en fondo gris se adivina la silueta de un dedo (invertido, es así por como hemos capturado los datos, nada más).

Para zanjar el tema haremos una comparativa entre los algoritmos expuestos en la memoria consultada y el nuestro.

Con una segmentación ideal de la imagen en base al color, se pueden suponer los siguientes resultados como píxeles pertenecientes al dedo índice extendido:

0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0

Supongamos también que por problemas de la segmentación, realmente tenemos esto:

0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0

Pues bien, el algoritmo utilizado por el grupo de Madrid, hubiera obtenido como resultado la localización de todos los puntos conectados de la región y hubiera dado como altura 9 (numero de filas a partir de la parte más alta de la mano con componentes conectados al guante de color), mientras que con nuestro algoritmo no hubiéramos

localizado todos los puntos conectados al guante, pero la altura del dedo hubiera sido también 9, es decir, el resultado es correcto.

0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0

La diferencia radica en que su algoritmo hace ocho comparaciones por cada píxel que trata, mientras que nosotros hacemos tres y además ellos se tienen que encargar de la gestión de una pila que utilizan como marcadora de píxeles. Con lo que tal y como funciona nuestra segmentación y la naturaleza de la postura del dedo, los resultados obtenidos con nuestra técnica son satisfactorios para trabajar en tiempo real.

5.1.6.2. RECONOCIMIENTO DEL CLIC UTILIZANDO RED NEURONAL

Este algoritmo de reconocimiento tiene en común con el anterior que también utiliza la altura del dedo índice como dato característico, sólo que en vez de generar el clic en función de cómo esta decrece, lo hace en función de la relación entre la altura de la mano y la altura del dedo. Con el método anterior, podíamos generar un clic simplemente alejando rápido la mano de la cámara, ya que en las imágenes recogidas y tratadas observamos que la altura del dedo decrecía, pero no teníamos en cuenta que en este caso se debía a un cambio de escala producido por el alejamiento de la mano y no por el encogimiento del dedo índice.

5.1.6.2.1. INTRODUCCION A LAS REDES NEURONALES

Antes de explicar más vamos a repasar muy someramente la base de las redes neuronales, así quien no este familiarizado con ellas podrá entender mejor los siguientes apartados.

Una red neuronal es un sistema para el tratamiento de información que intenta simular la estructura biológica del sistema nervioso humano, utilizando estructuras de neuronas. Y fundamenta su funcionamiento en el aprendizaje a partir de la experiencia.

Lo que básicamente ocurre en una neurona biológica es que esta es estimulada o excitada a través de sus entradas (inputs) y cuando pasa de un cierto umbral, la neurona se activa, pasando una señal hacia el axón (salida de la neurona conectada a otras).

Si llevamos los conceptos expuestos a la práctica tenemos que una red neuronal tiene el siguiente aspecto:

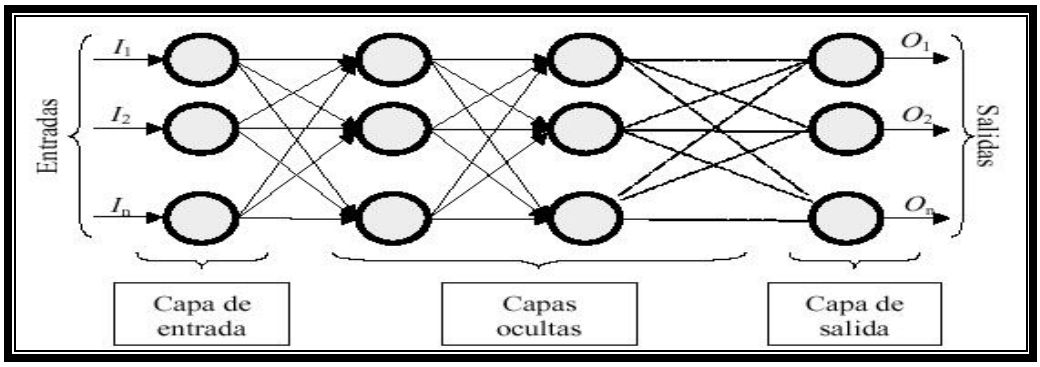

Figura 26. figura red neuronal

La red esta constituida por neuronas interconectadas organizadas en varias capas. Los datos son adquiridos por medio de la capa de entrada, pasan a través de la capa oculta (formada por una o varias capas) y salen por la capa de salida.

Tomando como unidad de procesamiento de la red el esquema presentado a continuación

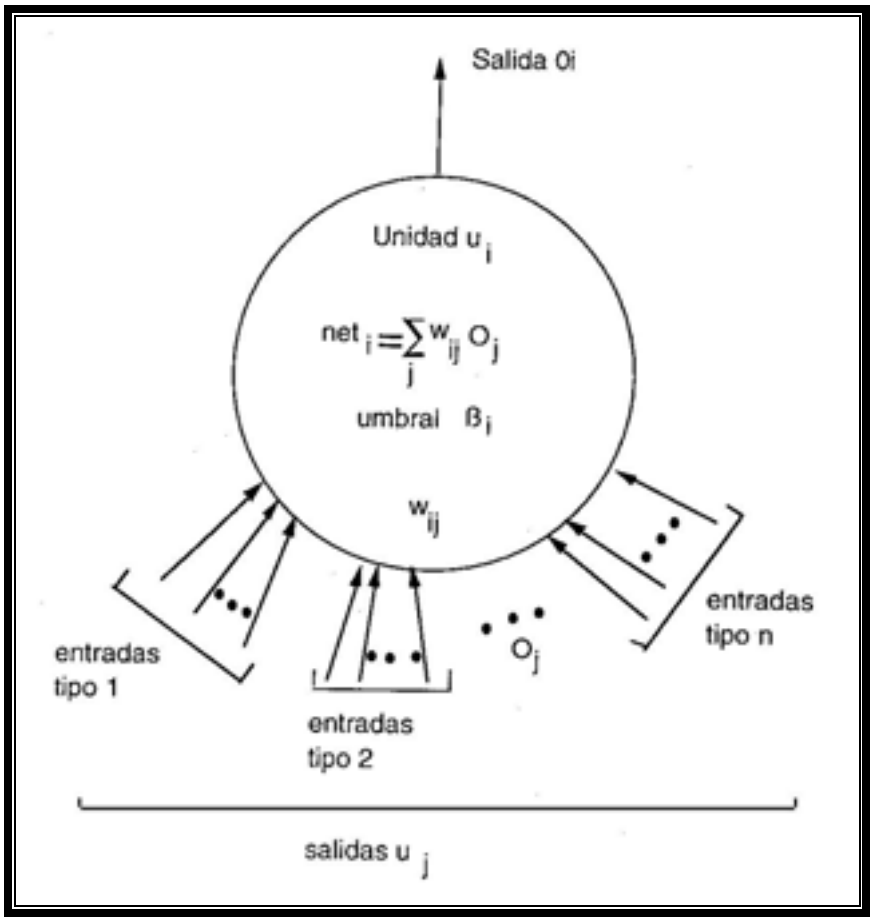

Figura 27. esquema unidad de procesamiento de redes neuronales

donde:

- la unidad u_i recibe entradas de las unidades u_j
- la salida de la unidad u_i es única, aunque luego se desdobla atacando a un conjunto de unidades
- cada conexión lleva asociado un valor real (peso) w_{ij}

Modelamos su comportamiento a partir de la función de activación y su función de salida. La función de activación tiene la siguiente forma:

$$a_i(t+1) = F_i(net_i(t) - \beta_i)$$

(fórmula función activación)

siendo

$net_i = \sum_j w_{ij} O_j$ (tambien pudiera ser otra operación como $\prod$ o Max)

β_i = umbral de activacion, bias

(fórmula net y explicación de bias)

y la de salida:

$$O_i = f_i(a_i)$$

(fórmula función salida)

Pudiendo ser esta función:

- una función lineal: $F(x) = \alpha x$
- una función escalón: $F(x) = \begin{cases} +\gamma & \text{si } x > 0 \\ -\gamma & \text{en otro caso} \end{cases}$
- una función sigmoidea: $F(x) = \frac{1}{1+e^{-x}}$
- etc

De esta forma van pasando los datos de capa en capa, logrando que la entrada sea procesada por la red neuronal con el propósito de lograr una salida. Para conseguir esto falta decir que la red debe aprender a calcular la salida correcta adaptando sus pesos a partir de un conjunto de datos, conocido como conjunto de entrenamiento, del cual conocemos las salidas que se deben producir. Para conocer con mas detalle los procesos de entrenamiento, estructuras de red, etc ver [11].

5.1.6.2.2. NUESTRA RED NEURONAL

La red neuronal utilizada en este proyecto tiene una topología 2-3-3, es decir, dos unidades de entrada, tres neuronas en la capa hiden y otras tres en la capa de salida. Hay que decir que también se probaron otras configuraciones, pero el rendimiento resulto ser similar en todas.

Para obtener los datos de entrenamiento de la red neuronal se hizo que tres usuarios manejaran el interfaz gestual y mientras se iban capturando dos parámetros, la altura de la mano y la altura del dedo. Con estos datos la red debería ser capaz de clasificar el gesto como clic o no en función de cómo estén relacionadas estas medidas. La idea de tomar estos como datos característicos era para solventar el problema que presentaba la utilización de reconocer los clics basándonos en los decrementos continuos de la altura del dedo.

La red que usamos está formada por unidades tangente sigmoideas, las cuales trabajan en el rango (-1,1), por esto fue preciso normalizar la entrada de los datos a la red, primeramente normalizándolos y después encajando estos datos en el intervalo mencionado.

Para el entrenamiento tomamos el modelo “lm”, que aunque en teoría es mejor para la aproximación de funciones que para ser utilizado como clasificador, en la practica se comporto ten bien o mejor que el “rp”.

Los resultados de los entrenamientos parecían esperanzadores ya que obteníamos valores bajos del error cuadrático medio, pero en la práctica, aunque el resultado no era del todo malo, presentaba un problema difícil de solucionar, que eran los falsos positivos, es decir, veces en las que el usuario simplemente quería posicionar el ratón y se generaba un clic indeseado, lo cual dependiendo de su posición puede generar efectos muy negativos, imaginémonos un diálogo en el que se nos pregunta si estamos seguros de eliminar un documento y en vez de pinchar en el deseado ‘no’, según vamos camino a él generamos el clic en el ‘si’.

Para solucionar el este problema se podía hacer un postproceso a los datos de salida de la red más riguroso, el problema es que entonces costaría generar un clic cuando se quisiera y tampoco se considera esta una buena solución.

Finalmente indicar que todo el proceso de creación entrenamiento y pruebas de la red se realizo en Matlab y después se paso el código del funcionamiento de la red y los datos obtenidos del entrenamiento a código C, para más información sobre este proceso consultar [12].

5.1.6.3. RECONOCIMIENTO DEL CLIC BASADO EN PROPIEDADES DE LAS PROYECCIONES

Esta última versión del reconocimiento del clic es la que mejores resultados ha dado. Hasta ahora tomábamos la altura del dedo índice como dato característico para el reconocimiento de un clic, calculándolo con el algoritmo ya presentado, pero ahora en vez de así, lo que haremos es decir si el dedo se encuentra o no estirado en función de ciertas propiedades de la proyección por filas.

Veamos primeramente un gráfico con la proyección por filas de los píxeles correspondientes al color del guante de una imagen en la que esta presente una mano con el dedo índice extendido y otra en la que está encogido.

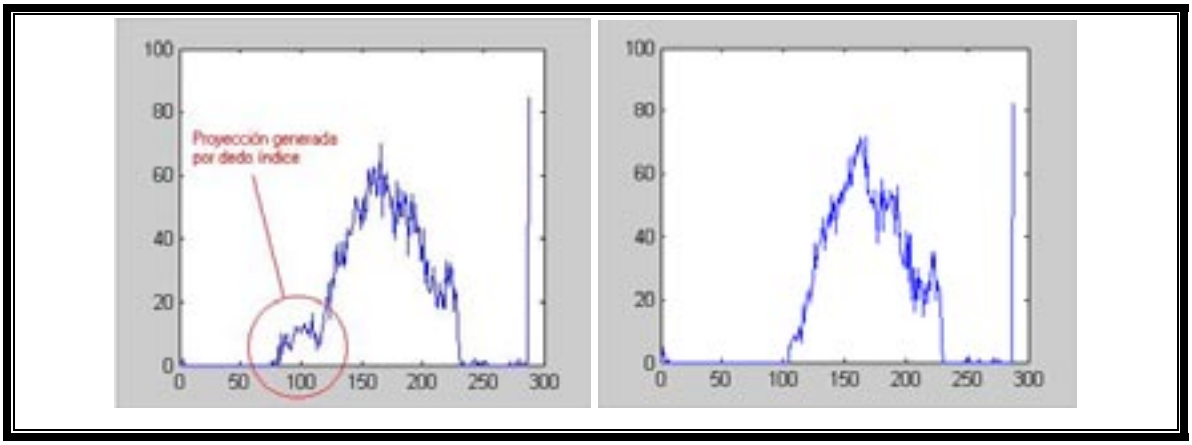

Figura 28. figuras con proyección clic y no clic

Observando ambas proyecciones podemos ver como cambia la primera parte de la gráfica. En la primera imagen vemos como hay, antes de una subida pronunciada en la proyección, una serie de filas con un número considerable de píxeles del color de guante, esta parte corresponde a los píxeles del dedo índice. En la segunda imagen

observamos como la subida es pronunciada desde el principio, lo que indica que directamente estamos localizando el área correspondiente a la palma de la mano, sin dedo índice extendido, o lo que es lo mismo, se esta generando un clic.

Para diferenciar esto lo que se hace es utilizar la función de acumulación de los valores de la proyección. El código (matlab) para hacer este cálculo es el siguiente

```
function sumaP=sumaProyeccion(array)
long=size(array,2); //obtenemos longitud del vector de proyecciones a  tratar
indiceInicio=-1; //inicializamos variables
%indiceAuxiliar=-1;
sumaMayor=0;
IndiceAreaMayorInicio=0;
IndiceAreaMayorFin=0;
for i=1:long  //tratamiento de cada elemento del vector
   if (array(i)>0) //si tiene un valor positivo el elemento de esa posición del vector
        if (indiceInicio == -1) //pasamos de sección con valores nulos a valores positivos
                indiceInicio=i; //registramos este indice como comienzo de area con
                                //valores positivos
                suma=array(i); //comenzamos a acumular en la variable suma, el área
                               // contiua que hay bajo la línea de la proyección a partir
                               //del índice i
        else //si ya estábamos en una sección de la proyección con valores positivos
                suma=suma+array(i); //Añadimos a la variable suma el valor
                                    //correspondiente a la actual posición del vector
        end
        if (suma>sumaMayor) //si el valor acumulado en la variable suma es el mayor
                                //hasta el momento
                indiceAuxiliar=indiceInicio //registramos el indice de inicio del mayor
                                            //area bajo la proyección
                sumaMayor=suma // actualizamos el valor del mayor área contiuo bajo la
```

```
                                        //línea de la proyección
        end
    else //si el valor de la posición actual de la proyección es igual a cero
        if (indiceInicio ~= -1) //este es el primer cero tras un area de valores positivos
            if (indiceAuxiliar==indiceInicio) //acabamos de pasar un área mayor que
                                              //los anteriores
                IndiceAreaMayorInicio=indiceInicio; //registramos el indice de
                                                    //inicio del mayor área continuo bajo
                                                    //la proyección
                IndiceAreaMayorFin=i-1; //y registramos también el indice del
                                        //final de area
            end
            indiceInicio=-1;
        end
        suma=0; //actualizamos la variable suma a cero, ya que el valor de la posición
                // actual de la proyección es nulo
    end
    sumaP(i)=suma; //una vez tratado el elemento actualizamos nuestro vector de
acumulación de los valores de la proyección.
end
```

(código para función de acumulaciones)

Una vez calculadas las funciones de acumulación de las proyecciones de la figura 28 obtenemos las siguientes gráficas:

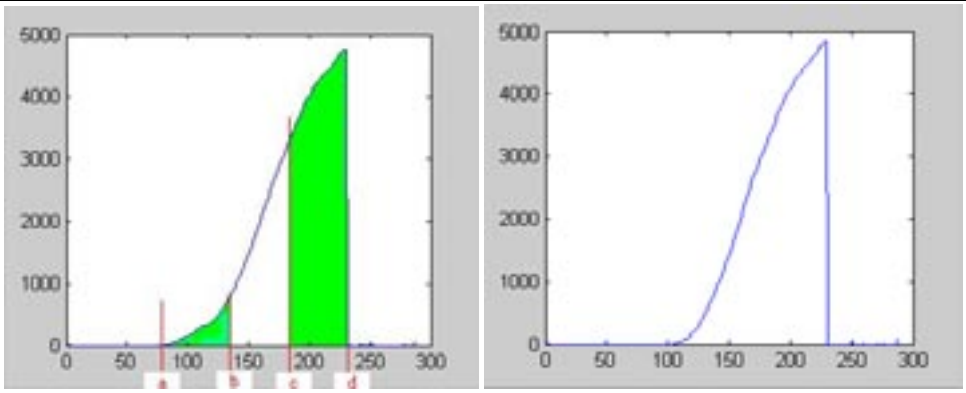

Figura 29. figura acumulaciones

Ahora calculamos el área que hay bajo la curva entre el índice correspondiente al comienzo del mayor intervalo de valores positivos continuos (punto a), y este índice mas 1/3 de la anchura de la mano (punto b) y el área que hay bajo la curva entre el índice correspondiente al final del mayor intervalo de valores positivos continuos (punto d), y este índice menos 1/3 de la anchura de la mano (punto c). Con estos datos lo que hacemos es modelar el estado de la mano de acuerdo al cociente entre ambas áreas, ya que la subida cambia en función da la posición de la mano. Si el valor es inferior a un cierto umbral, que ajustaremos de acuerdo a las características de la mano de cada usuario, querrá decir que tenemos el dedo encogido, y por tanto generaremos el evento de clic del ratón. Bueno, realmente no producimos un clic en cuanto se detecta el estado anterior, sino que se debe dar estas situaciones en dos estados de tiempo consecutivos para que se genere el clic (esto mismo se da en el método del apartado anterior). Resultados experimentales hacen suponer que un buen valor para este umbral del que hablábamos sea 13, ya que el resultado del cociente da valores cercanos a 9 en caso de tener la mano en la postura correspondiente a la generación de un clic, y cercanos a 18 en otro caso.

Como notas finales, explicar que un evento de clic ratón se genera utilizando las siguientes funciones del API de Windows

GetCursorPos(&mouseXY);

```
mouse_event(MOUSEEVENTF_ABSOLUTE|MOUSEEVENTF_LEFTDOWN,
mouseXY.x,mouseXY.y,0,0);

mouse_event(MOUSEEVENTF_ABSOLUTE|MOUSEEVENTF_LEFTUP,mou
seXY.x,mouseXY.y,0,0);
```

(código posiconamiento del ratón)

A la función mouse_event se le pasa como parámetros la acción que queremos generar (presionar botón derecho, liberar botón,) y el punto en el que se quiere generar este evento, indicando también si estas coordenadas están en función de la ventana cliente o son coordenadas absolutas. Se destaca esto, no por adentrarnos en las funciones que ofrece el API de Windows para el manejo del ratón (mas datos en [6]), sino para señalar que en nuestra aplicación este evento se genera realmente en el punto en el que el usuario esta viendo el ratón posicionado (de ahí la captura de la posición del ratón mediante la utilización de GetCursorPos), y no en el punto al que se esta moviendo. Esta diferencia es fundamental, ya que con la inercia introducida por los filtros, había veces que el ratón aun no se había detenido y el evento del ratón que generaba el usuario se estaba enviando a un punto distinto al deseado. Esta apreciación se hizo tras probar la aplicación inicial y ver como muchas veces fallábamos al intentar pinchar en un elemento del menú y pinchábamos uno cercano a él.

5.1.7. LA APLICACIÓN

Ya que en esta misma memoria se incluye un manual de usuario, no vamos a explicar en detalle cada ventana de la interfaz, ni el significado exacto de cada parámetro ajustable.

De todos modos mostraremos el aspecto general del interfaz, que es este:

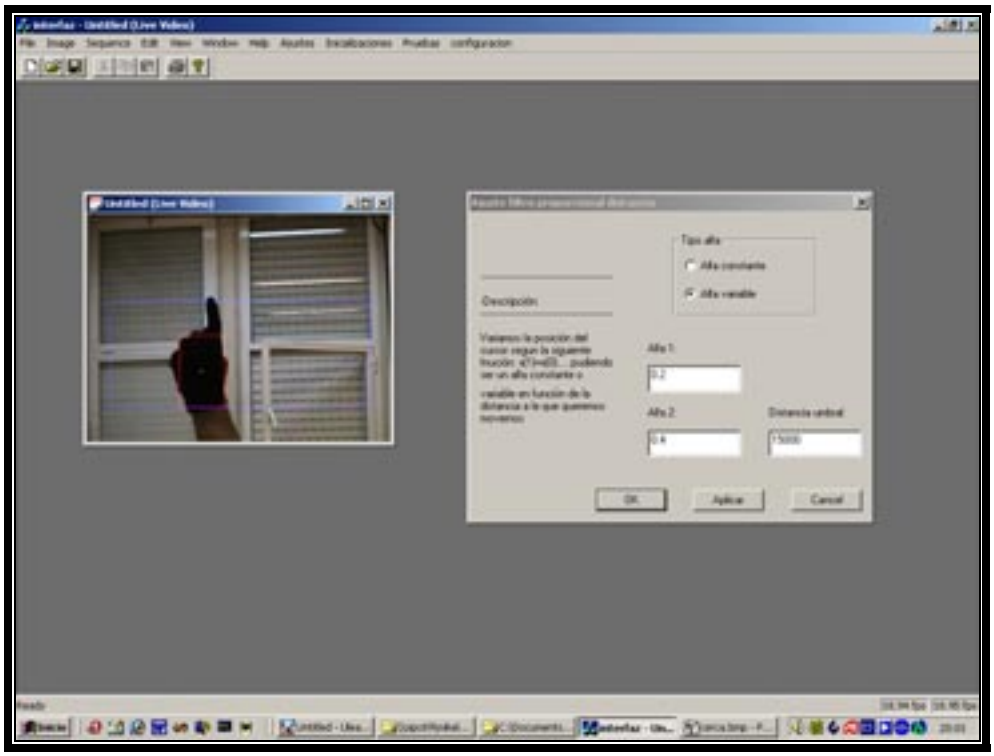

Figura 30. aspecto de la ventana del interfaz gestual

El interfaz es muy simple, en la ventana en la que se muestran las imágenes de video, se marca también la zona de la imagen que tiene correspondencia con la pantalla real (cuadro Azul), además, según procesamos las imágenes también mostramos el lugar en la imagen donde se esta detectando la mano (cuadro Rojo) y por último marcamos el punto que correspondería al hot spot del ratón (punto verde).

Además mediante el menú superior podemos acceder a las distintas ventanas de configuración, donde podemos seleccionar los filtros que queramos o el método de detección del clic y ajustar sus parámetros.

Lo interesante llegado el punto en el que ya tenemos el interfaz configurado y esta funcionando correctamente sería minimizar la aplicación y dejar que trabaje en segundo plano mientras nosotros utilizamos el resto de programas normalmente. Pues bien, una vez teníamos implementado todo el sistema, se vio que al minimizar la ventana de la captura de video, el sistema dejaba de funcionar, el fallo se achaca a que las librerías utilizadas por lo visto dejan de capturar imágenes cuando estas no están siendo mostradas. Lo ideal sería solucionar este problema, pero al tratarse de un

prototipo y ya que no afectar a su funcionamiento se optó por dejarlo así, es decir, podemos abrir cualquier programa, movernos por cualquier menú de la pantalla, etc, pero siempre teniendo de fondo la ventana de nuestra aplicación maximizada. Si no se consiguiera solucionar el problema en un futuro mediante la edición del código, otra posible solución seria que el código de esta aplicación, de nuestra interfaz, estuviera embebido en el código de las aplicaciones que quisieran funcionar con este tipo de sistema de interacción.

Como último punto de este apartado señalar que la aplicación trabaja procesando unas 20 imágenes por segundo, en la version debug, y en tiempo real en la version release.

5.1.8. RESULTADOS DE USABILIDAD

Para comprobar la usabildad de este tipo de "dispositivo" se puso a prueba con diferentes usuarios. A cada uno de ellos se le sometió a la misma tarea, que consistía en presentarle la siguiente ventana

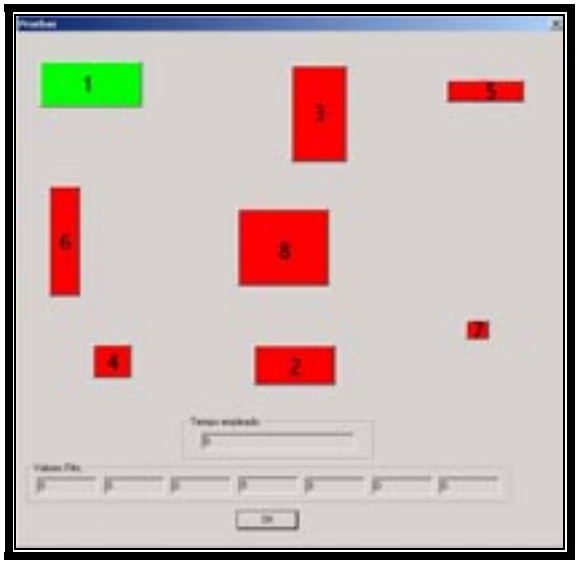

Figura 31. imagen capturada de la pantalla con ventana de prueba abierta

y cronometrar el tiempo que tardaba en posicionar y pinchar en cada botón en un orden establecido, concretamente el correspondiente a la numeración estos.

Para comparar la usabilidad del sistema y dar unos datos más objetivos, se sometió a 100 personas a la prueba presentada, solo que 50 utilizaron el interfaz gestual para el desarrollo de la tarea, y las otras 50 utilizaron el touchpad de los ordenadores portátiles para realizarla.

Los botones presentan diferentes propiedades, diferentes alturas y anchuras, pudiendose utilizar así la ley de Fitts para modelar las dificultades asociadas a cada transición. Para guiar al usuario, y no perder tiempo en localizar el siguiente botón a pinchar, si no solamente en el posicionamiento y generación del clic, se hacía que el siguiente botón sobre el que había que clicar fuera de color verde, pasando a ser de rojo tras pinchar en él.

A cada usuario se le dieron siete intentos, cinco seguidos y otros dos tras tres minutos de descanso, y estos fueron los tiempos medios obtenidos por los usuarios tras los siete intentos dependiendo el tipo de dispositivo utilizado:

Figura 32. tiempos medios de realización de la prueba del interfaz gestual

Las curvas de aprendizaje denotan una mejora de los tiempos según las personas se van adaptando al tipo de interfaz y a la prueba, pero en cualquier caso las personas que han realizado la prueba usando el interfaz gestual han obtenido mejores tiempos medios en todos los intentos. Además, aunque no se muestra en la gráfica, los tiempos de los últimos intentos se van acercando bastante a los tiempos que obtenía el desarrollador de la aplicación para esta misma prueba (6 segundos aproximadamente), con lo que queda patente lo intuitivo del sistema.

Para ver unos resultados más riguroso puede leerse [13], que es un artículo que trata sobre este tema publicado en …. ¿??revisar parrafo, si no lo publican, ponemos referencia tb o ampliamos información en eeste punto?

5.1.9. CONCLUSIONES SOBRE EL INTERFAZ GESTUAL

Las sensaciones han sido muy buenas. Se ha trabajado tanto en el área de la visión por computador, como en el de las redes neuronales. Se han explorado diferentes gramáticas para la comunicación gestual. Se ha aprendido el funcionamiento de la interfaz gráfica de Windows. Se ha utilizado un lenguaje de programación hasta ahora desconocido para el alumno, el C++, otro conocido, matlab, pero haciendo una especial utilización de las toolbox de redes neuronales, y afrontando la tarea de transformación de código de matlab a código C.

Además se han alcanzado los objetivos marcados, teniendo operativo un prototipo funcionando en tiempo real y todos los usuarios del sistema se mostraban contentos con su funcionamiento y destacaban lo novedoso de este.

También es verdad que queda mucho campo por explorar, y como posibles mejoras mencionar la exploración de un mayor número de gramáticas, probar otros métodos de segmentación y filtros y por último intentar eliminar restricciones del

sistema, como la de no tener que aparecer en el fondo nada del color del guante, mediante la aplicación de algoritmos como el que se presenta en el articulo [14], que básicamente intenta modelar el fondo suponiendo una distribución gausiana del color de cada píxel de la imagen a lo largo del tiempo, pudiendo así suponer que parete de la escena corresponde al fondo.

5.2. RECONOCIMIENTO ALFABETO DACTILOLÓGICO

En este apartado veremos un trabajo totalmente diferente al de la primera parte. Ahora no se desarrollara un sistema completo, sino que trabajaremos en la creación de una base de conocimiento sobre el alfabeto dactilológico del lenguaje de los sordos, que consiste en la filmación y catalogación de una serie de videos donde aparece gente con conocimiento del lenguaje de los sordos signando diferentes palabras y frases. De estos videos además se extraeran fotogramas donde aparecen los gestos correspondientes a cada letra del alfabeto.Y se realizará un estudio de clasificación de estos para la futura implementación de un sistema de reconocimiento automático del lenguaje de los sordos a partir de una secuencia de video.

Esta sección está dividida en dos grandes subsecciones, la primera donde se detalla el proceso de captación de imágenes, digitalización y organización de estas y una segunda sección donde hablaremos de la clasificación de las imágenes.

5.2.1. CAPTURA DE DATOS

Ante la imposibilidad de encontrar en Internet una base de datos con videos de personas sordas signando el alfabeto dactilológico en un entorno controlado, de donde poder extraer fotogramas con los gestos correspondientes a cada letra y poder extender el estudio a la traducción del lenguaje de los sordos a partir de un video, se tomó la determinación de grabarlos de primera mano. Lo que si se encontro en Internet fue una página que daba la posibilidad de conseguir videos donde se trataban diferentes temas utilizando el lenguaje de los sordos completo, y otra donde hay una gran cantidad de información de lenguaje, material pedagógico, videos de poesias acompañadas del texto original, etc. Estas direcciones son:

http://www.cnse.es/Titulos.htm

http://www.cervantesvirtual.com/portal/signos/

El proceso de grabación no hubiera sido posible sin la colaboración de una amiga, Ana Belen, la cual se prestó para la primera prueba de grabación de los videos y mediante la cual se consiguió contactar con ARANSGI (Asociación para la rehabilitación auditiva de los niños sordos en Gipuzkoa), donde gracias al trato con Maria Espín, se logró la colaboración para la filmación de varios profesores con conocimiento del lenguaje de los signos y alumnos sordos. En total fueron 4 alumnos, 5 profesores y Ana Belén.

- **Entorno y Condiciones de Grabación**

La primera taréa para realizar las grabaciones era definir las condiciones en las que se harían. Se impusieron los siguientes requisitos:

- La grabación se realizó con luz artificial, dentro de aulas con luz fluorescente.
- Había una tela negra como fondo
- La persona que aparezcía signando el alfabeto iba vestida con una camisa negra.
- La persona estarba enfocada de cintura para arriba, teniendo así flexibilidad para signar cada gesto de forma cómoda.

En la siguiente imagen se aprecia la disposición de la camara y resto de elementos en el entorno de grabación.

Figura 33. Entorno de grabación

- **Captación, Digitalización Y Almacenamiento De Datos**

Sabiendo las condiciones de la grabación falta por definir los datos que se recogieron en ellas.

- Primeramente cada persona signaó el alfabeto dactilológico completo cinco veces
- A continuación signaron una lista de palabras en las que se encuentran recogidas todas las letras del alfabeto. Las palabras que forman esta lista son: becerro, corredor, futurista, galápago, higo, hijo, kilogramo, llave, murciélago, niño, payaso, quesito, water, xilófono, zarzamora.
- Y por último signaron una serie de frases dos veces. Estas frases estan formadas por palabras que aparecen en la lista anterior, y además se utilizó un símbolo especial para separación de palabaras. Las frases son:
    - El murciélago veloz pesa un kilogramo
    - Me gusta el quesito con zarzamora
    - Mete el galápago en el water
    - Mi hijo es un becerro

La idea signar primero el alfabeto completo y después palabras y frases se hizo pensando en utilizar los símbolos que parecen en el deletreo como modelo y después validarlo pasando al sistema las palabras y las frases.

Una vez se grabó a los voluntarios utilizando la camara Sony Handycap 8mm, se pasó al proceso de digitalización. Para ello se conectó la cámara con la tarjeta capturadora de video y se utilizó como software el Ulead Video Studio 7.0.

Primeramente se digitalizó la secuencia completa de las grabaciones, pasando posteriormente a dividir este video (utilizando también otras funciones del Ulead Video Studio) en fragmentos correspondientes a cada persona y cada una de estas secciones a su vez fue dividida en otras tres, correspondiendo una al deletreo del alfabeto, otra al deletreo de las palabras y la última correspondiente al deletreo de las frase.

Así en el CD entregado encontramos dentro de la carpeta “videos” una serie de carpetas etiquetadas con letras de la ‘a’ a la ‘j’, conteniendo cada una los videos correspondientes a cada persona. Cada una de estas carpetas contiene tres videos, una con el deletreo del alafabeto (cuyo nombre esta formado por la letra correspondiente a la persona + deletreo), otra con el deletreo de las palabras (nombre=letra correspondiente a la persona + palabras) y el tercero con el deletreo de las frases (nombre=letra correspondiente a la persona + frases).

Figura 34. captura de organización de directorios y nombres

Además en el directorio correspondiente a cada persona encontramos otro subdirectorio llamado ‘fotos’, donde se recogen las imágenes correspondientes a los frames del video donde se ve a la persona con la mano en la postura propia de cada letra. Se han capturado cinco imágenes fijas de cada gesto, pero unicamente de aquellos que representan a una letra de forma estática (ver anexo b), y se han denominado de la siguiente manera:

Nombre archivo = letra correspondiente a persona que signa (nombre de la marpeta) + numero de letra (tal y como aparecen ordenadas en el anexo b) + numero de captura de esa letra (de 1 a 5)

Por ejemplo la primera captura de la letra ‘k’ signada por el sujeto ‘a’ tendrá por nombre: a121.jpg y este es su contenido:

Figura 35. imagen a121.jpg

5.2.2. CLASIFICACIÓN DE GESTOS

Las imágenes fijas nombradas en el aprtado anterior se han utilizado para hacer un estudio de diferentes métodos de clasificacion de gestos con el fin de ser utilizados en un futuro para la procesar los videos grabados y saber que es lo que se está signando en ellos.

- **Preprocesamiento de Imágenes**

Un tabajo previo a la clasificación fue extraer de cada imagen el area donde aparece únicamente la mano, tarea que se automatizó en gran medida por el gran número de imágenes a tratar.

Este preprocesamiento de las imágenes tenía que conseguir que solamente apareciera la mano sobre fondo negro en la nueva imagen, y que el área de la mano ocupara siempre una proporción similar del area total de la imagen, y así ha sido.

Para lograr este cometido se realizó una aplicación en matlab que pasandole como parámetro el nombre del directorio donde tenía guardadas las fotos actuba de la siguiente maneraa:

- La aplicación recorre las fotos una a una y las va mostrando al usuario en una ventana.

- El usuario pincha con el ratón en la parte de la imagen donde aparece la mano

- Y automáticamente se crea una nueva imagen de dimension 100x100 con la imagen de la mano centrada. El nombre de esta nueva imagen esta formado por 'seg'+'nombre de archivo de origen de la foto'

El archivo de matlab que realiza este proceso es el "procesado_directorio", cuyo código puede verse en el cd (en cualquiera de las carpetas dentro de 'videos').

(1)   (2)

(3)   (4)

Figura 36. Figura procesamiento de la imagen

(5)

Viendo la figura 36 comprenderemos mejor el proceso que hemos descrito. Primeramente se pasa la imagen a escala de girises y se binariza, quedando resaltadas el área correspondiente a la mano y a la cara principalmente. El usuario señala con el ratón cual es el area de color blanco que representa a la mano, y el programa guarda automaticamente los pixeles de la imagen (1) correspondientes a ese area de pixeles blancos en la imagen (4) en una nueva imagen (5) de tamaño 100x100 con la imagen de la mano en su centro.

- <u>PCA</u>

Para la clasificación de las imágenes anteriores se ha tomado como método de clasificación, más concretamente como método de extracción de características, el PCA, método que explicaremos brevemente.

Planteamiento del problema: Sea un vector aleatorio de observaciones de dimension **n**: $\mathbf{x}=(x_1 \quad x_2 \ \ldots \ x_n)^t$ con media nula $E[x]=0$ y matriz de autocorrelación $R_x=E[xx^t]$

El objetivo del análisis en componentes principales es encontrar una transformación lineal **W** a un espacio de dimensión menor **m**<**n**, tal que la reconstrucción de la señal tenga un error cuadrático medio mínimo, es decir:

Proyección en un espacio de dimension menor: $\mathbf{y}_{mx1}=\mathbf{W}_{mxn}\mathbf{x}_{nx1}$

Reconstrucción del espacio original: $\mathbf{x`}_{nx1}=\mathbf{F}_{nxm}\mathbf{y}_{mx1}$

Objetivo: error de reconstrucción $Je=E[\_x-x`\_^2]$ mínimo

El vector **y** se denomina "vector de características": es el vector de dimensión **m** que mejor "explica" (de manera lineal) al vector original **x**.

La matriz de autocorrelación puede descomponerse en autovalores y autovectores ($\mathbf{R}_x\mathbf{u}_i=\__i\mathbf{u}_i$) como:

$$\mathbf{R}_x=\sum_{i=1}^{m}\__i\mathbf{u}_i\mathbf{u}_i^T=\mathbf{U}\_\mathbf{U}^T$$

Siendo $\mathbf{U}=[\mathbf{u}_1,\mathbf{u}_2,\ \ldots,\mathbf{u}_m]$ y _ una matriz diagonal con los autovalores $\__1 \geq \__2 \geq \cdots \geq \__n$.

La matriz óptima de proyección la forman los m autovectores correspondientes a los mayores autovalores

$\mathbf{W}_{mxn}=[\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_m]^T$

y la matriz de reconstrucción es

$\mathbf{F}_{nxm}=[\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_m]$

En definitiva la transformación PCA se reduce a encontrar aquellos vectores en los que las proyecciones den máxima varianza.

Como muestra de que este es un método muy conocido y utilizado en problemas de clasificación de imágenes se recomienda ver [15], donde se utiliza para clasificar

imágenes de leones marinos y además explica como afecta algún tipo de transformación simple de la imagen, como la rotación, al funcionamiento del método.

- **Clasificadores**

En el apartado anterior faltaría decir que para clasificar las imágenes una vez proyectadas en el nuevo espacio, se tomará como referencia la distancia euclidea al cuadrado componente a componente entre los elementos, calculada como:

$$d(x_i, x_j) = \sum_k [x_i(k) - x_j(k)]^2$$

Figura 37. fórmula distancia euclidea

También tenemos que señalar cuantas son las imágenes capturadas de cada signo para su posterior clasificación, estando repartidas así:

a	37
b	45
c	46
ch	21
d	27
e	34
f	40
g	40
h	6
i	41
k	31
l	45
m	24
n	33
o abierta	28
o cerrada	18
p	35
q	34
r	28
s	29
t	36
u	45
w	27

Figura 38. Tabla con número de imágenes de cada clase

Podemos observar como el número de representantes de la ‘ch’ y la ‘h’ es inferior al resto, y esto se debe a que los gestos correspondientes a ambas letras son de naturleza dinámica (ver anexo b), y se intentaron capturar frames similares con escaso éxito. También se separó la ‘o’ en dos grandes grupos, ya que aunque cada uno signaba cada letra con ciertas peculiaridades, se podía distiguir claramente como había dos formas de signar la ‘o’.

Ya centrandonos en la clasificación, diseñamos hemos utilizado tres tipos de clasificador:

- tipo I:
    - 1NN: Una vez proyectados todos los elementos en la nueva base, sacamos uno a uno de ella, y lo clasificamos/etiquetamos con la clase de aquel elemento que se encuentre a menor distancia de él.
    - Los resultados se reflejan en la siguiente matriz de confusion:

	a	b	c	ch	d	e	f	g	h	i	k	l	m	n	o abierta	o cerrada	p	q	r	s	t	u	w	total	mal clasificados
a	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	37	1
b	0	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	0
c	0	0	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	46	1
ch	0	0	0	17	0	1	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	21	4
d	0	0	0	0	24	0	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	27	3
e	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0
f	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	1	7	0	0	40	8
g	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0
h	0	0	0	0	0	0	0	1	3	0	0	0	0	0	0	1	1	0	0	0	0	0	0	6	3
i	0	0	0	0	0	0	0	0	0	41	0	0	0	0	0	0	0	0	0	0	0	0	0	41	0
k	0	0	0	0	0	0	1	0	0	0	29	0	0	0	0	0	0	0	1	0	0	0	0	31	2
l	0	0	0	0	0	0	0	0	0	0	0	45	0	0	0	0	0	0	0	0	0	0	0	45	0
m	0	0	0	0	0	0	0	0	0	0	0	0	21	3	0	0	0	0	0	0	0	0	0	24	3
n	0	0	0	0	0	0	0	0	0	0	0	0	2	31	0	0	0	0	0	0	0	0	0	33	2
o abierta	0	0	1	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	2	1	0	0	28	4
o cerrada	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	18	1
p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	1	0	0	0	0	35	1
q	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0	34	1
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	26	0	0	0	0	28	2
s	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	25	0	0	0	29	4
t	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	1	31	0	0	36	5
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	0	45	0
w	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	4	21	27	6

    - Valoración: de la 750 imágenes que se quieren clasificar, se han clasificado mal 51, es decir, sólo el 6.8% del total.
- tipo II:
    - 3NN: Al igual que en la primera forma de clasificar, una vez proyectados todos los elementos en la nueva base, vamos sacando de

ella los elementos uno a uno, y lo etiquetamos en función de la mayoria de las clase de los tres vecinos más proximos, y en caso de que no resulte una clase ganadora, se etiqueta como la clase de aquel elemento más cercano.

- Tambien veremos los resultados reflejados en una matriz de confusión:

	a	b	c	ch	d	e	f	g	h	i	k	l	m	n	o abierta	o cerrada	p	q	r	s	t	u	w	total	mal clasificados
a	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37	0
b	0	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	0
c	0	0	46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	0
ch	0	0	1	16	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	21	5
d	0	0	0	0	24	0	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	27	3
e	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0
f	0	0	0	0	0	0	31	0	0	2	0	0	0	0	0	0	0	0	0	1	6	0	0	40	9
g	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0
h	0	0	0	0	0	0	0	1	3	0	0	0	0	0	0	1	1	0	0	0	0	0	0	6	3
i	0	0	0	0	0	0	0	0	0	41	0	0	0	0	0	0	0	0	0	0	0	0	0	41	0
k	0	0	0	0	0	0	1	0	0	0	29	0	0	0	0	0	0	0	1	0	0	0	0	31	2
l	0	0	0	0	0	0	0	0	0	0	0	45	0	0	0	0	0	0	0	0	0	0	0	45	0
m	0	0	0	0	0	0	0	0	0	0	0	0	14	9	0	0	0	1	0	0	0	0	0	24	10
n	0	0	0	0	0	0	0	0	0	0	0	0	3	30	0	0	0	0	0	0	0	0	0	33	3
o abierta	0	0	1	0	0	0	0	0	0	0	0	0	0	0	23	0	0	1	0	2	1	0	0	28	5
o cerrada	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	1	0	0	0	18	2
p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	3	0	0	0	0	35	3
q	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	0	0	0	0	34	1
r	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	23	0	0	0	0	28	5
s	0	0	0	0	0	2	0	0	0	1	0	0	0	0	0	0	0	1	0	25	0	0	0	29	4
t	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	1	30	0	0	36	6
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	43	1	45	2
w	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	5	20	27	7

- Valoración: de la 750 imágenes que se quieren clasificar, se han clasificado mal 70, es decir, el 9.33% del total.

- tipo III:
  - Aproximación gausiana: Aquí lo que hacemos es suponer una distribución gausiana de los datos, y tomamos como representate de cada clase la media de la mitad de sus elementos. Y clasificamos la otra mitad de elementos en función a la menor distancia a los representantes anteriormente mencionados.
  - Veamos la matriz de confusión con los resultados:

	a	b	c	ch	d	e	f	g	h	i	k	l	m	n	o abierta	o cerrada	p	q	r	s	t	u	w	total	mal clasificados
a	13	0	0	0	0	0	0	3	0	2	0	0	0	0	0	0	0	1	0	0	0	0	0	19	6
b	0	21	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	2
c	0	0	16	0	0	1	0	0	0	0	0	1	0	0	0	0	0	3	0	0	0	2	0	23	7
ch	0	0	0	5	0	0	0	1	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	12	7
d	0	0	0	0	5	0	0	0	0	1	2	2	0	0	0	0	0	0	3	0	0	0	0	13	8
e	0	0	0	0	0	12	0	0	0	0	0	2	0	0	3	0	0	0	0	0	0	0	0	17	5
f	0	0	0	0	0	0	6	0	0	2	1	1	0	0	10	0	0	0	0	0	0	0	0	20	14
g	0	1	0	1	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	2
h	0	0	0	2	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	4	3
i	0	0	0	0	0	0	1	0	0	15	0	0	0	0	5	0	0	0	0	0	0	0	0	21	6
k	0	0	4	0	1	0	1	0	0	0	7	3	0	0	0	0	0	0	1	0	0	0	0	17	10
l	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	23	0
m	0	0	0	0	0	0	0	0	0	0	0	0	4	8	0	0	0	0	0	0	0	0	0	12	8
n	0	0	0	0	0	0	0	0	0	0	0	0	6	10	0	0	1	0	0	0	0	0	0	17	7
o abierta	0	0	0	0	0	0	1	0	0	0	0	1	0	0	10	0	0	1	0	0	1	0	0	14	4
o cerrada	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	9	3
p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	2	0	0	0	0	18	2
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	17	0
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	6	0	0	3	0	14	8
s	0	0	0	1	0	3	2	0	0	1	0	1	0	0	4	0	0	1	0	1	1	0	0	15	14
t	0	0	0	0	0	0	5	0	0	1	0	0	0	0	9	0	0	0	0	0	3	0	0	18	15
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	18	0	23	5
w	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	11	14	3

- Valoración: de la 383 imágenes por clasificar, se han clasificado mal 159, o lo que es lo mismo, el 36.3% veces del total.

NOTA: En el cd, en las carpetas “estudio1”, “estudio 2” y “estudio 3” (dentro de la carpeta “imágenes”), aparecen los archivos de matlab utilizados para la creación de la nueva base y para la realización de las clasificaciones, con un archivo leeme.txt donde aparecen una serie de instrucciones.

5.2.3. RESULTADOS

Antes de hablar de resultados definitivos, se realizaron una serie de pruebas complementarias.

En el apartado anterior las pruebas no tienen en cuenta todos los factores experimentales presentes para la clasificación, a saber:

- Tipo de algoritmo para la clasificación
- Tamaño de las particiones
- Nº de componentes a utilizar
  - Mayores
  - Menores
- Nº de repeticiones del muestreo para particionar los datos

Lo que hemos hecho realmente hasta ahora ha sido tener en cuenta sólo el tipo de clasificación, sin realizar particiones para el proceso de validación, y sin variar el numero de componentes principales utilizados para la clasificación (no se cayó en la cuenta que puede que no sea óptima la utilización de todos ellos en la clasificación).

Por lo tanto se pasó a definir unas pruebas más rigurosas:

Para cada tipo de algoritmo, se dividen los datos en dos particiones del 50% de elementos cada una, utilizando una parte como representantes de la clase, y la otra parte como conjunto de test. Se clasificará cada elemento de la partición de test utilizando primeramente los 10 mayores componentes, después con los 20 mayores…y así sucesivamente hasta llegar a las 750 componentes. Y para que los resultados no dependan del azar se realizará 30 veces el mismo proceso tomando cada vez particiones aleatorias. También se realizará el mismo estudio pero tomando en vez de los mayores componentes los menores.

Estos son los resultados dependiendo del tipo de algoritmo de clasificación:

- Tipo I: 1NN

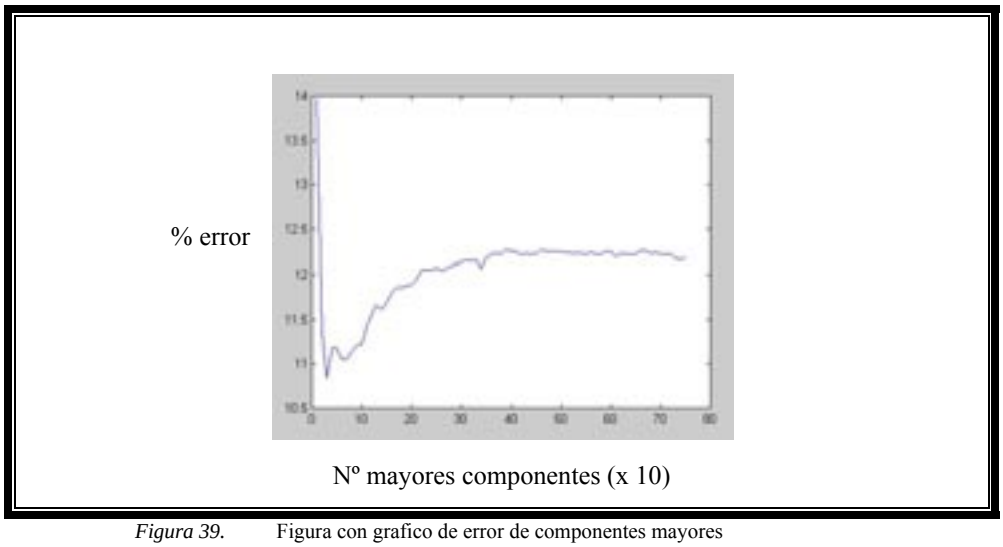

Figura 39. Figura con grafico de error de componentes mayores

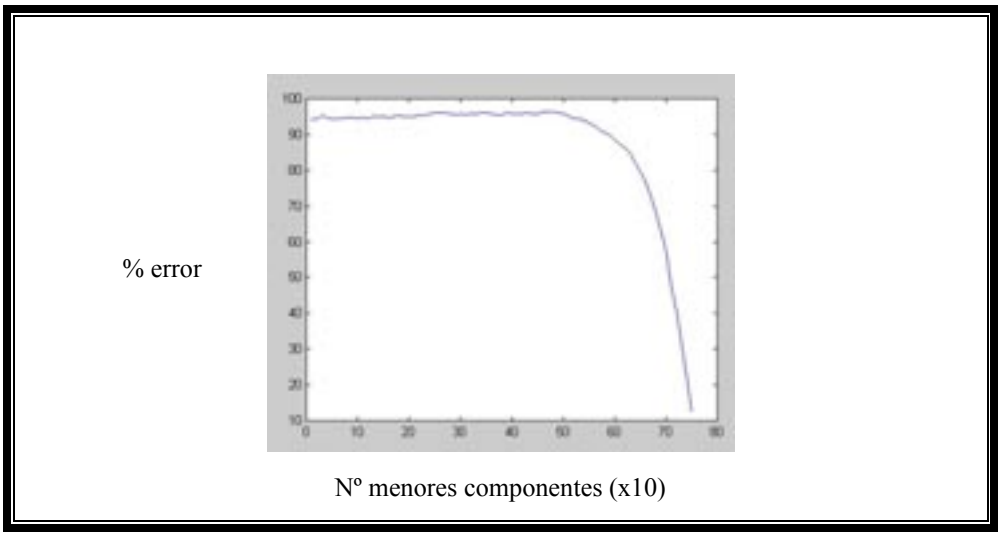

Figura 40. Figura con grafico de error de componentes menores

Ahora en vez de presentar los resultado de la clasificación en forma de matriz de confusión, lo que hacemos es ver la evolución del error total de la clasificación (calculado como porcentaje de elementos mal clasificados del total del conjunto de test) en función del numero de componentes, mayores en la figura 39 y menores en la figura 40, utilizados para realizar la clasificación.

En al figura 39 vemos como, aumentando inicialmente el número de componentes para la clasificación, el error decrece, pero sólo hasta que usamos 30 de ellos, ya que a partir de este punto la curva de errores comienza a crecer. Si miramos la grafica de la figura 40 observamos el efecto contrario, el error es bastante grande y sostenido hasta que llegamos a utilizar la mayoria de componentes (concretamente esos últimos 30, que son los 30 mayores componentes), que hacen que el error decaiga drásticamente.

Con el conjunto de resultados podemos asegurar que son los mayores componentes los que mejor caracterizan a nuestrao conjunto de imágenes, siendo un número óptimo para la clasificación los 30 mayores.

- Tipo II: 3NN

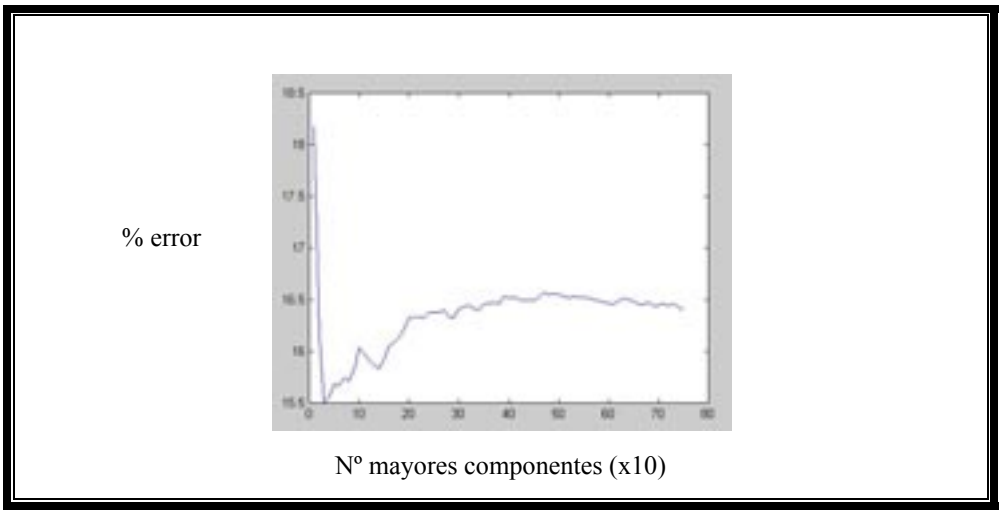

Figura 41. Figura con grafico de error de componentes mayores

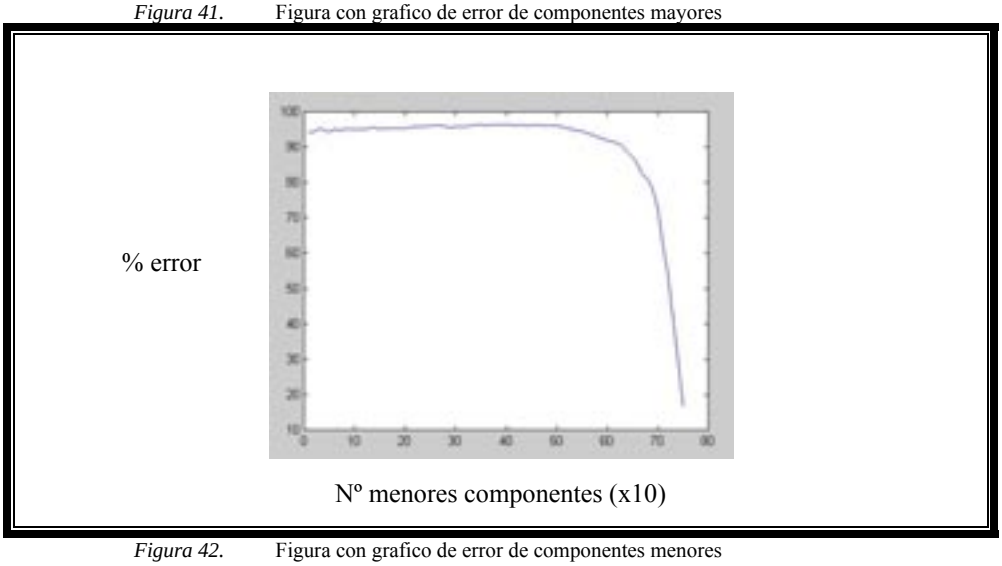

Figura 42. Figura con grafico de error de componentes menores

En esta segunda prueba se repite el comportamiento de la primera. Utilizando los mayores componentes para realizar la clasificación, volvemos a observar un acusado descenso del error utilizando tan solo los 30 primeros componentes, volviendo a aumentar (aunque se estabiliza) a medida que vamos utilizando un mayor número de ellos.

La gráfica de la figura 42, donde se utilizan los menores compoenntes para la clasificación, también repite el comportamiento del la prueba anterior, donde vemos que el error no decrece hasta que no utilizamos los mayores componentes en la clasificación.

También hay que decir que el error en cualquier caso es mayor utilizando este tipo de clasificación, que si utilizamos el tipo del apartado anterior.

- Tipo III: Aproximación gausiana

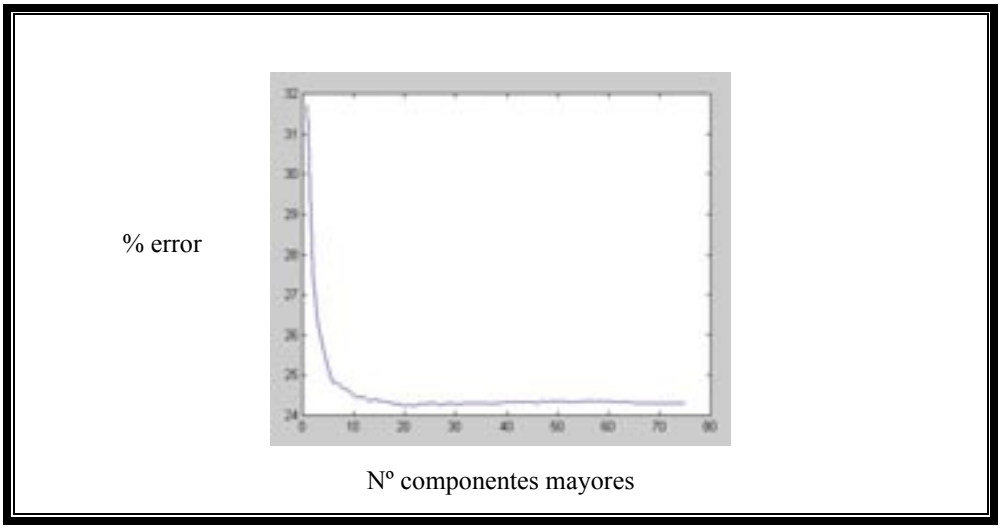

Figura 43. Figura con grafico de error de componentes mayores

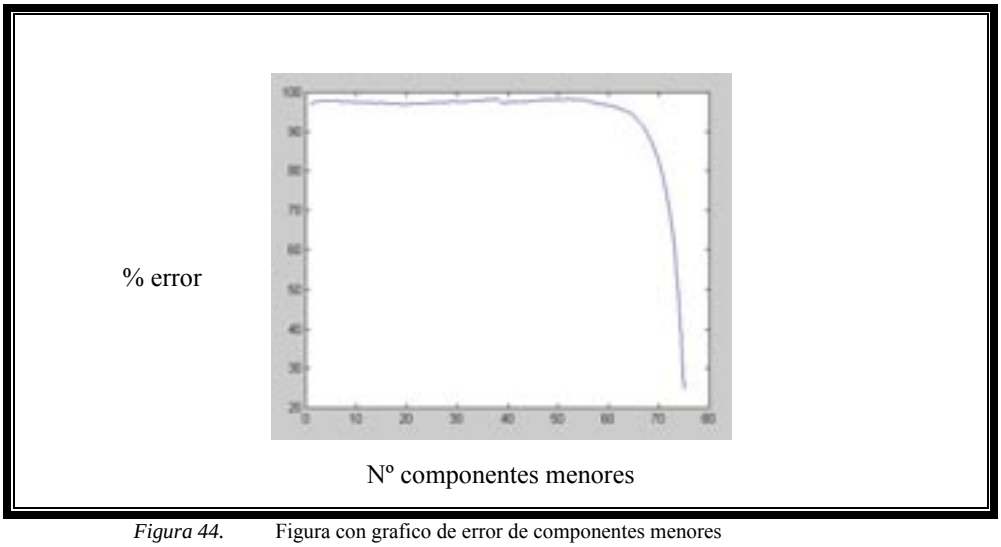

Figura 44. Figura con grafico de error de componentes menores

La figura 43, donde se muestra el error la clasificación en función de la clasifiación utilizando los mayores componentes, cambia un poco respecto a lo que hemos visto en los dos tipos de clasificación anteriores. En este caso vemos que se necesita usar un mayor número de compoenentes para reducir el error, y tras lograr este mínimo prácticamente se mantine a lo largo de toda la gráfica. En cambio la figura 44 sigue la tónica de las anteriores en las que se utilizaban los menores componentes para la clasificación.

En cualquier caso hay que señalar también, que utilizando este tipo de clasificación es con el que peores resultados se obtienen.

5.2.4. CONCLUSIONES RECONOCIMIENTO ALFABETO DACTILOLÓGICO

Con los datos obtenidos, la recomendación para la clasificación de estas imágenes es utilizar el método de clasificación 1NN, utilizando en cualquier caso los 30 mayores componentes.

Volviendo a realizar la clasificación de los datos utilizando el primer método presentando,es decir, una vez proyectados todos los datos en el nuevo espacio, sacar uno a uno de el y clasificarlo como su vecino más próximo, pero utilizando únicamente los 30 mayores componentes obtenemos el siguiente resultado:

	a	b	c	ch	d	e	f	g	h	i	k	l	m	n	o abierta	o cerrada	p	q	r	s	t	u	w	total	mal clasificados
a	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	37	2
b	0	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	0
c	0	0	46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	0
ch	0	1	0	17	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	21	4
d	0	0	0	0	23	0	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	26	3
e	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0
f	0	0	0	0	0	0	33	0	0	0	0	0	0	0	0	0	0	0	0	1	6	0	0	40	7
g	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0
h	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0
i	0	0	0	0	1	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	41	1
k	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	1	0	0	0	0	31	1
l	0	0	0	0	0	0	0	0	0	0	0	45	0	0	0	0	0	0	0	0	0	0	0	45	0
m	0	0	0	0	0	0	0	0	0	0	0	0	21	3	0	0	0	0	0	0	0	0	0	24	3
n	0	0	0	0	0	0	0	0	0	0	0	0	3	30	0	0	0	0	0	0	0	0	0	33	3
o abierta	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	1	0	0	0	2	0	0	0	28	3
o cerrada	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	1	0	0	0	18	2
p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	1	0	1	0	0	35	2
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	1	0	0	0	0	34	1
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	27	0	0	0	0	28	1
s	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	26	0	0	0	29	3
t	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	1	31	0	0	36	5
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	0	45	0
w	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	4	22	27	5

Tenemos un 6.14% de mal clasificados, porcentaje que juntando las clases f y t baja a un 4.8% y juntando a su vez la u y la w y la m y la n llega a ser de tan solo 4.

El tomar dos clases y etiquetarlo como una sola se hace por la gran similitud entre ambas letras. En el caso de la f y la t, como se puede ver en el anexo b, la única diferencia es que en una el dedo gordo pasa por delante del índice, y en la otra por detrás, y estando las imágenes capturadas con la resolución y condiciones en las que se ha hecho es prácticamente imposible distinguirlas incluso a simple vista. Capturando las imágenes con una mayor resolución, o con una iluminación diferente puede que el problema mejorase.

El problema entre la confusión de la m y la n y la u y la w es diferente. En este caso el hecho de que la gente no separe bien los dedos a la hora de signar, hace que los símbolos se parezcan, pero en este caso la solución pasaría por hacer que la gente signe con más cuidado.

6. CONCLUSIONES

Como conclusiones finales destacar mi satisfacción por la realización de este proyecto, tanto por el desarrollo del interfaz gestual, su muestra a un gran número de personas y la satisfacción de estas al usarlo, como por el estudio realizado sobre las imágenes del lengaje de los signos y su potencial para futuras aplicaciones de interes social.

Aunque el número de horas empleado ha sido muy elevado, se considera que ha sido un tiempo muy bien aprovechado. Los conocimientos tecnicos adquiridos han sido muy interesantes, pero el haber tratado con gente discapacitada y haber tomado conciencia de sus problemas ha supuesto también un gran enriquecimiento.

Centrandonos más en el partado técnico, recordar los buenos resultados obtenidos por el interfaz gestual, eso si, con unas cuantas restricciones como el uso de un guante de color y la necesidad de un fondo sin el color de este. Estas restricciones hacen pensar en un trabajo futuro para eliminarlas, utilizando por ejemplo modelos del color de la piel a la hora de localizar la mano en la imagen y utilizando también algoritmos que logren modelar el fondo pudiendo tener objetos del mismo color sin interferir.

Respecto a la segunda parte, la del reconocimiento del lenguaje de los signos, también hay que decir que tiene puntos donde poder seguir trabajando. Se puede, desde eliminar restricciones de color de fondo y ropa del usuario, hasta pasar a estudiar los signos dentro de una secuencia de video.

7. BIBLIOGRAFIA

[1] Julio Abascal.(mayo-juni 2000). La interacción persona computador en los próximos 25 años, NOVATICA,especial 25 aniversario.

[2] Chao HU, Max Qinghu,Peter Xiaoping, Xiang Wang (octubre 2003).Visual gesture recognition for human-machine interface robot teleoperation. Conference on Intelligent Robots and Systems. Las Vegas, Nevada

[3] Masaki Omata, Kentaro Go, Atsumi Imamiya.A gesture-based interface for seamless communication between real and virtual worlds. 6th ERCIM Workshop "user interfaces for all"

[4] http://www.cse.ogi.edu/CHCC/QuickSet/mainProj.html

[5] The Microsoft Vision SDK (2000), Vision Technology Group, Microsoft Research.

[6] Paul Yao, Richard C. Leinecker (1998) Todo Visual C++ IDG Bible (pp. 491 – 581). IDG Books woldwide

[7] Apuntes de visión por computador. Universidad Miguel Hernández. "Color en imágenes digitales"

[8] Iñigo Barandiaran (2002) PFC.

[9] Pablo Ayala (2003) PFC.

[10] Jaime Silvela Maestre (2000) PFC. Sistema eficiente de reconocimiento de gestos de la mano.

[11] Apuntes Redes Neuronales (2003). UPV/EHU

[12] Matlab C++ book (2003) . LePhan Publishing

[13] articulo nuestro

[14] Alexandre R. J., François and Gérard G. Medioni. Adaptative color background modelling for Real-Time segmentation of video streams.(1999) Integrated Media Systems Center. Institute for Robotics and Intelligent Systems - PHE 204.University of Southern California, Los Angeles, CA, 90089-0273, USA (http://iris.usc.edu/~afrancoi/pdf/CISST1999-paper.pdf)

[15] Cesar Caiafa, A.N. Proto (2003) Desarrollo de un software para la identificación de elefantes marinos por eigenfaces. Reportes técnicos en Ingeniería del Software. Vol. 5(2) pp. 27-40
(http://www.itba.edu.ar/capis/rtis)
[16] Jon Bates, Tim Tomkins (1999) Descubre Visual C++ 6. Prentice Hall Iberia, S.R.L.
[17] Manuel Graña. (2003) Apuntes de vision por computador

Anexo A

Para poder hacer un uso casi inmediato de la aplicación desarrollada en la primera parte del proyecto haremos una pequeña guia de usuario.

- Inicilalización

Una vez lanzada la aplicación es necesario tomar muestras del color del marcador/guante que estemos utilizando en la mano.

Ir al menú "inicializaciones" y pulsar en "tomar muestras"

Una vez nos salga el nuevo dialogo, si pulsamos en "area para muestra" aparecera un recuadro rojo en la imagen que la camara esta capturando, cuyo area deberemos llenar del color del guante y pulsar la tecla "tomar muestra" para recoger una muestra de esta.

Pulsando en "ver blobs" hará que al cerrar este dialogo se destaquen en color azul en la imagen que se está captando los píxeles del color correspondiente a la muestra.

Por fin, pulsando el botón "ok" hacemos que se cierre el dialogo y tengan efectos los cambios.

NOTA: Si queremos cambiar el color de muestra, pulsaremos al botón "reset muestra" y volveremos a proceder como se especifica en este apartado para tomar otro color.

- Ajustes

Este es el menú donde encontramos la opción de elegir el tipo de reconocimiento del clic que queremos utilizar, el tipo de filtro de movimiento que queremos usar, así como la posibilidad de ajustar ciertos parámetros de la aplicación.

  - El menú "ajustes gráficos" nos da la opción de dibujar en la imagen que estamos capturando el area donde se está localizando la mano, el area de la imagen con puntos de correspondencia en la pantalla, y el punto de la mano correspondiente al hot-spot que se esta locaclizando en cada momento. Para ello basta marcar o desmarcar una serie de casillas.
  - En "umbrales" vemos una serie de campos de texto donde podemos cambiar los umbrales por defecto utilizados para la localización de la mano.

Umbral1 y 2 se corresponden con el número de filas y columnas que debemos tener como minimo con píxeles del color del guate para suponer que ese area se trata de una mano.

Umbral3 y 4 corresponden al número de píxeles del color del guante que debe haber en las filas y columnas para localizar la palma de la mano.

Umbral5 se especifica como umbral para determinar si se esta produciendo un clic o no si se esta utilizando como método para su detección el de "división de areas de acumulación"

- El menú de selección de filtro nos da la opción de seleccionar el método de filtrado de posicionamiento que queremos utilizar y la parametrización de este

- Por último encontramos el menú donde seleccionar el tipo de reconocimiento del clic que queremos utilizar, el método de "deteción por decrementos de la altura sucesivos", utilizando una red neuronal, o mediante el algoritmo de "división de areas de acumulación"

- Salvar / Cargar configuración

Una vez realizados todos los ajustes necesarios para el correcto funcionamiento del sistema en un entorno concreto el programa nos da la opción de guardar la configuración (dentro del menú "configuración"), así al volver a ejecutar la aplicación en las mismas condiciones bastará pulsar al botón cargar configuración en vez de tener que volver a a ajustar cada parámetro, volver a capturar muestras de color, etc.

- Pruebas

Si desplegamos este menú y pinchamos en "probar interfaz" se abrirá una nmueva ventana con la prueba diseñada para medir la eficiencia de nuestro sistema como ya se ha explicado en el punto 5.1.8 de la memoria.

Anexo B

«Deletreo manual», «dactilología» o «alfabeto manual» son algunos de los términos empleados para referirnos a ciertas representaciones manuales del alfabeto. Existe una correspondencia entre una forma concreta de la mano y una letra del alfabeto escrito. Podríamos decir que la dactilología es una forma de escritura «en el aire».

Los alfabetos manuales son, en la actualidad, principalmente empleados por las personas sordas en su comunicación para, por ejemplo, deletrear nombres propios, términos orales que no tienen correspondencia exacta con un signo concreto, etc. Son asimismo empleados en contextos educativos con la finalidad de servir de facilitador para el aprendizaje de la lengua oral.

La siguiente imagen nos muestra el alfaeto dactilológico español:

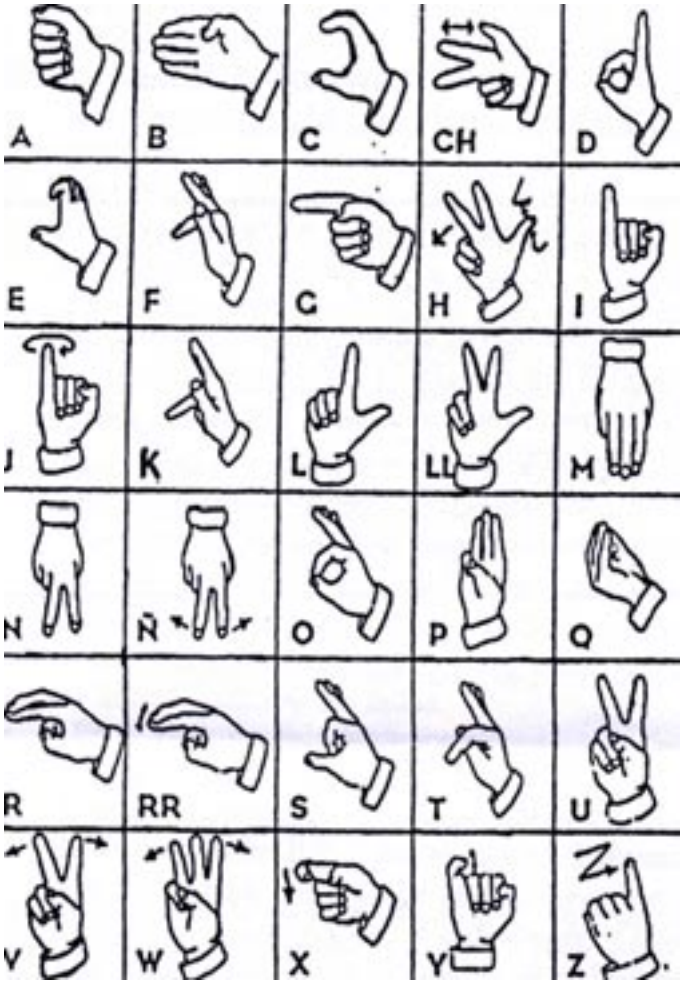

Figura 45. Alfabeto dactilológico español

Anexo a con funcionamiento de menus del interfaz, especie de manual de usuario
Hacer anexo b con lenguaje de los sordos
Anexo c sobre paso codigo matlab-c++??

Hacer archivo leeme en carpetas donde hay .m para explicar como se hace una ejecución.

Leido hasta pag 30