Reconocimiento de expresiones faciales con Kinect

Memoria final de “Practicas Externas en CITSEM”

10 de Julio de 2015

Estefanía Ferreira Cano

Tutor(a): Martina Eckert

Grupo de trabajo: Realidad Aumentada

Resumen

En la investigación desarrollada en el CITSEM, se ha marcado el objetivo de reconocer emociones a través de las expresiones faciales de una persona. Para su detección, se emplea el sistema Kinect, así como su entorno de desarrollo SDK y las librerías para reconocimiento facial correspondientes.

Para llegar a este objetivo, se ha empleado trabajos anteriores del CITSEM, así como una investigación propia, para asentar los conocimientos necesarios sobre reconocimiento facial y el funcionamiento de Kinect.

Los resultados obtenidos, han alcanzado el objetivo principal, llegando a reconocer emociones mediante diferentes técnicas y patrones. Las principales diferencias entre las técnicas de reconocimiento implementadas han sido: detectar el movimiento espacial de puntos estratégicos en la cara o detectar el movimiento de un músculo facial.

Abstract

In research performed in the CITSEM , the main purpose has been recognizing emotions through facial expressions of a person. For detection, the Kinect has been used to reach this goal, in addition to its development environment and the SDK libraries which correspond with facial recognition.

To reach this goal, it has been used CITSEM previous work and an own research, to learn the necessary knowledge of facial recognition and Kinect operation.

The results obtained have achieved the main objective, recognize emotions using different techniques and patterns. The main differences between recognition techniques that have been implemented are : detect spatial movement of strategic points on the face or detect the movement of a facial muscle.

---

1. Introducción y objetivos de la práctica

Las presentes prácticas en investigación, han tenido dos objetivos: el esclarecimiento de cómo a través de Kinect se pueden reconocer las expresiones faciales de un usuario, y mediante este método averiguado, la implementación y análisis de nuevos programas para dicho fin.

Este trabajo se ha apoyado en el entorno de desarrollo SDK de Microsoft para Kinect [1], tanto en sus librerías como en ejemplos ya implementados, como se comenta más adelante.

Para conseguir el objetivo principal, se han presentado una serie de objetivos a realizar:

- Estudio de reconocimiento de expresiones faciales.
- Estudio del funcionamiento de Kinect.
- Estudio del entorno de desarrollo SDK.
- Estudio de las implementaciones realizadas para reconocimiento de expresiones faciales.
- Análisis del reconocimiento de expresiones faciales de Kinect.
- Estudio de lenguaje C#.
- Implementación de nuevas posibilidades para el reconocimiento facial y de expresiones.

Las implementaciones que se han llevado a cabo, fueron realizadas basándose en el trabajo anterior realizado en el CITSEM por el grupo de trabajo Face Group, en concreto, por los últimos integrantes Almudena Gil y Diego Zapatero. De su trabajo, se obtuvo una base sobre reconocimiento facial, así como, la forma de relacionar los movimientos de la cara con la emoción que sentía el usuario.

2. Estado del reconocimiento de emociones y aplicaciones con Kinect

Para comenzar con el objetivo de estas prácticas, se realizó un estudio del reconocimiento de emociones. Para ello, se estudió trabajos anteriores realizados sobre este tema en el CITSEM [2]. En la actualidad, las emociones más estudiadas, a la hora de reconocimiento de emociones, son: felicidad, tristeza, miedo, disgusto, enfado y sorpresa. Estas emociones, serán sobre las que se investigará para poder detectarlas en tiempo real con Kinect. Para comenzar con esta investigación, hay que definir dos términos esenciales en este campo:

- Método de las FACS (*Facial Action Coding System*): que permite identificar los músculos faciales que causan cambios en la cara. Al movimiento de estos músculos se le denomina *Action Units* (AUs) y existen 46.Es el método en el cual se basa esta investigación. Actualmente ha evolucionado a FACES (*Facial Action Coding Expression System*), que permite disminuir el tiempo de detección de las emociones.
- Se define como **Action Units AUs** a los movimientos fundamentales de un músculo o grupo de músculos.
- Se define como **Shape Unit (SU)** a la forma de un músculo facial.

Tras esta primera toma de contacto, se realizó un estudio de diferentes trabajos, ya existentes, para el reconocimiento de expresiones faciales con Kinect [3]

Se clasifican los trabajos sobre este campo en función de la información entrante, los pasos de pre-procesado, las características de los tipos de extracciones de información, los métodos de clasificación y los resultados de post-procesamiento. La solución que se presenta para el reconocimiento de expresiones faciales y construcción de un modelo virtual 3D usando información de profundidad y RGB obtenido con Kinect, consiste en primero detectar la cara y segmentarla en regiones. Después se identifica la expresión mediante EigenFaces en las imágenes RGB y se reconstruye la cara mediante la información de profundidad. Todo ello sin supervisión. Se analizan varios parámetros de Kinect para llegar a esta solución:

- Primero se analiza la extracción de la profundidad y RGB de la cara. Primero se corrige la desviación producida por el ligero desplazamiento entre los dos sensores mediante la biblioteca OpenCV.

- El siguiente punto a analizar es la detección de la cara y el seguimiento. Se emplea un detector de cara basado en el aprendizaje en cascada Adaboost.

- A continuación, se analiza la extracción de cada parte de la cara. Para detectar las emociones, primero localiza los ojos y la boca y luego proporciona una expresión. Debido a la disposición geométrica de la cara, se emplean las coordenadas x e y para localizar las posiciones en el espacio, y las coordenadas W y H para la profundidad.

- Otro punto a analizar es el procesamiento de la expresión. Cualquier cara x puede ser reconstruida por una imagen promedio m añadiendo un número de detalles u. Se obtiene el vector propio usando la covarianza de la matriz y se aplica el análisis PCA para obtener la imagen promedio querida. El sensor de profundidad genera una cuadrícula de 640x480 que describe la distancia entre cada píxel hacia el proyector IR.

- El último factor que se ha estudiado es la reconstrucción facial. Tras la generación del archivo pcd que contiene la nube de puntos de toda la escena, se filtra dos veces para obtener los puntos relativos a la cara. Para la animación se realiza un sumatorio de imágenes de la cara.

En [4] se estudió la librería SHORE (*Sophisticated High-speed Object Recognition Engine*), así como una utilidad del reconocimiento facial con Kinect.
La librería SHORE es empleada para detectar y clasificar las valoraciones emocionales de las expresiones detectadas de un vídeo en vivo. El método fue escogido porque es capaz de entender estados afectivos del usuario de una manera no intrusiva, comparado con la alternativa de la electromiografía facial, donde la actividad del musculo en la cara es medido mediante electrodos que tienden a influir en la evaluación.

Con el método SHORE, primero se detectan las caras, luego se asigna un número a cada cara (habiendo 4 números). Cada número está asociado a su vez con una expresión facial: felicidad, enfado, tristeza o sorpresa.

Por otra parte, se realizó un estudio de Vikrti [5]. Vikrti es una aplicación de escritorio para Windows que usa el sensor Kinect para reconocer los gestos faciales que hace el usuario y con ellos poder interpretar las emociones, figura 1.

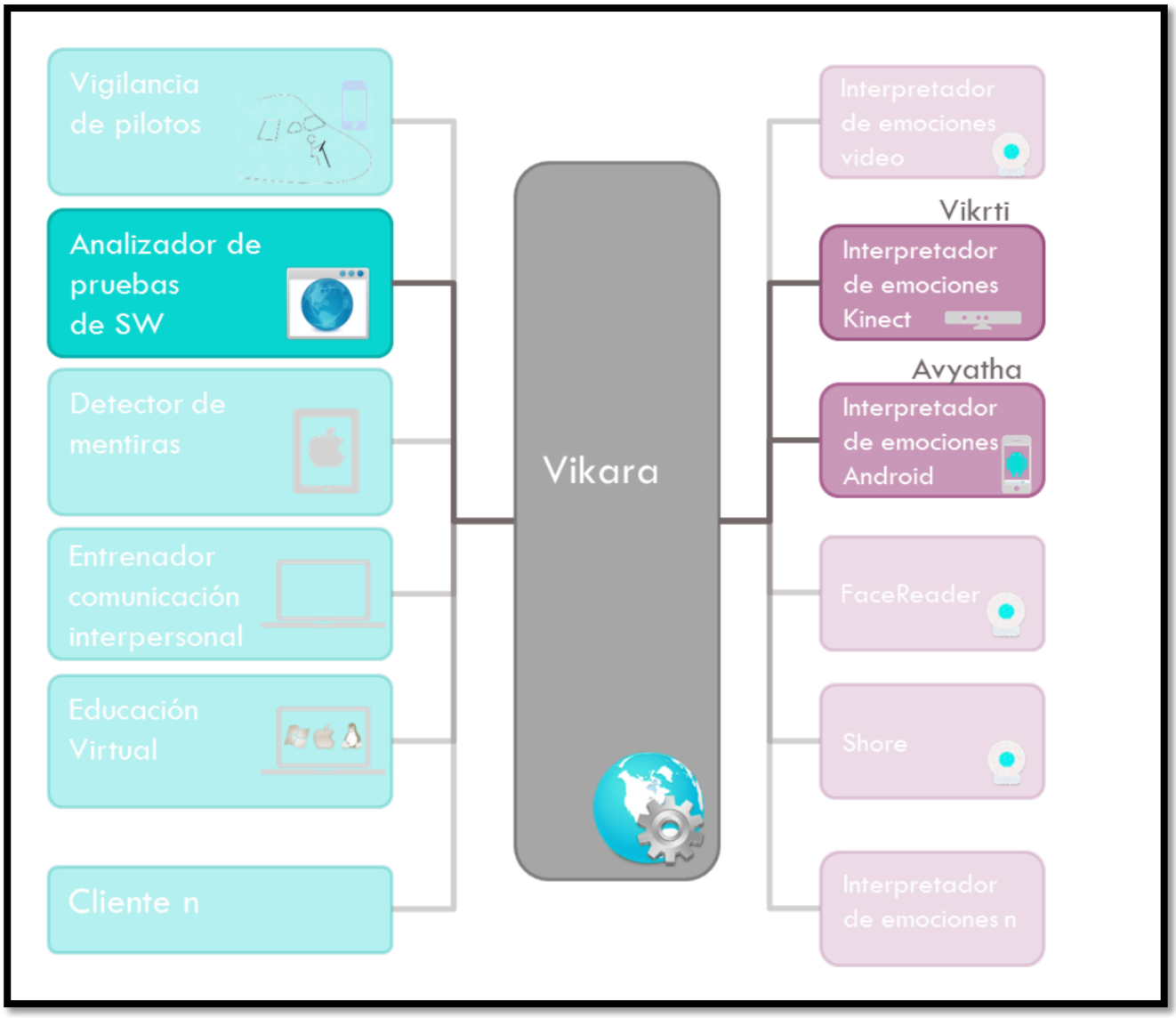


Figura 1. Arquitectura de Vikrti [5].

Todo ello, está soportado por Vikara, como se mostraba en la figura. Vikara es una plataforma que facilita el desarrollo, la gestión e integración de aplicaciones cuya finalidad es interpretar emociones. Vikara, a través de servicios web, se encarga de guardar las emociones detectadas por diversos interpretadores automáticos de emociones y de facilitar el acceso a dicha información para que pueda ser usada por medio de otras aplicaciones. En la tabla 1, se compara esta plataforma de desarrollo, con otras, entre ellas, la mencionada anteriormente SHORE, se aprecia que la más versátil es Vikara, debido a que es la única que el número de expresiones es ampliable, y el único parámetro que no aporta es la clasificación del estado de ánimo, siendo la que más características cubre.

Tabla 1. Comparación entre sistemas.

Sistema	Emociones reconocidas	Clasifica estado de ánimo (+/-)	Reporte de emociones (historial)	Código abierto	Múltiples usuarios en un video	API	Calibración	Ojos abiertos/cerrados	Boca abierta/cerrada	Soporta cara parcialmente cubierta	Detección de mirada	Método de clasificación
eMotion	Neutral + 6 básicas	Sí	No	No	No	No	No	No	No	Sí	No	Facial: Unidades de Movimiento y diversos clasificadores de aprendizaje automático
FaceReader	Neutral + 6 básicas	Sí	Sí	No	No	Sí	Sí	Sí	Sí	No	Sí	Facial: FACS
Shore	Neutral + 4 básicas (alegría, ira, tristeza, sorpresa)	No	No	No	Sí	Sí	No	Sí	Sí	Sí	No	Facial: Aprendizaje automático (AdaBoost) que usa tablas de búsqueda en el proceso de clasificación
RealEyes	Aburrimiento, ira, ansiedad, tensión , sorpresa, euforia, y goce	No	No	Sí	No	No	No	NA	NA	NA	NA	Aprendizaje automático
Vikara	***Extensible, inicialmente Neutral + 6 básicas***	***No***	***Sí***	***Sí***	***Sí***	***Sí***	***Extensible, con la implementación inicial facial de FACS***					

Por último, en la tabla 2, se muestran otros entornos de desarrollo para Kinect. Se puede observar que el único que soporta librerías ya implementadas para reconocimiento facial es SDK.

Tabla 2. Entornos de desarrollo [6].

	Kinect SDK	OpenNI	LibFreeNect	Skeltrack
Reconocimiento de Esqueleto	Si	Si	No	Si
Usuarios simultáneos	2	2	-	1
Calibración necesaria	No	Si	-	No
Joints	20	20	-	7
Resolución máxima	640x480	320x240	320x240	-
Grabación de sonido	Si	No	Si	No
Dependencia de la plataforma	Si - Windows	No (Windows, Linux, Mac OS X, etc)	No (Windows, Linux, Mac OS X, etc)	No (Windows, Linux, Mac OS X, etc)
Reconocimiento de manos	Si	Si	No	No
Rotación de *Joints*	Si	Si	No	No
Reconocimiento facial	Si	No	No	No

Finalmente se escoge el entorno de desarrollo Kinect SDK, para realizar la implementación para el reconocimiento de emociones.

2.1. Kinect SDK

En el presente apartado, se explica cómo el entorno de desarrollo SDK emplea Kinect e implementaciones en C# propias para conseguir el reconocimiento facial de usuarios.

Antes de comenzar a explicar el reconocimiento facial realizado por Kinect, es preciso explicar dos definiciones: Candide y AAM.

2.1.1. Candide

Candide es una malla parametrizada para la cara humana. Esta malla está controlada por las unidades de acción (AUs) locales y globales. Las locales rotan alrededor de tres ejes. Los globales controla los gestos de la cara que producen las distintas expresiones humanas.

Candide fue creado por Mikael Rydfalk en el Image Coding Group de la Universidad de Linköping, Suecia, en 1987. [7]

La primera versión del modelo Candide, estaba formado por 100 triángulos y 75 vertices. Actualmente, existen tres versiones, tales versiones se muestran en la figura 3.

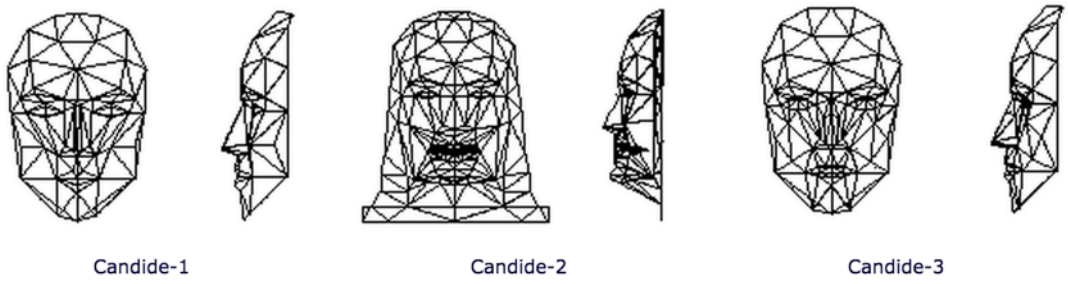


Figura 2. Versiones de Candide [8].

2.1.2. Active Apperance Model

AAM es un algoritmo matemático basado en estadística, que mediante un modelo de un objeto, en este caso de la cara humana, moldea para adaptarse a la imagen real. Se adapta a través de la posición de imágenes juntas con coordenadas que detecta y sitúa en puntos estratégicos de la imagen. En la figura 4 se muestra el proceso de reconocimiento y en la figura 5 los puntos finales detectados.

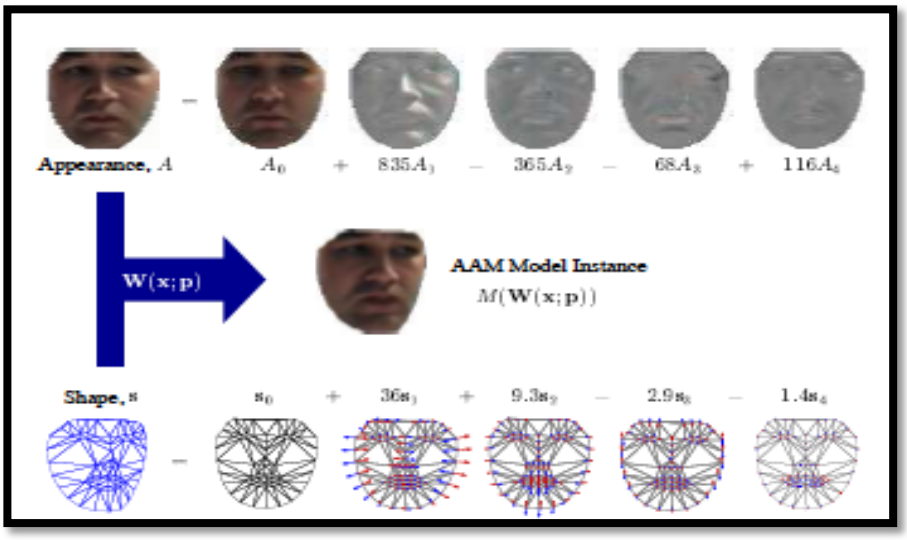


Figura 3. Obtención de puntos mediante AAM [9].

Figura 4. Puntos obtenidos por AAM [9].

Tras explicar estos dos términos, se procede a explicar cómo SDK realiza el reconocimiento facial. SDK emplea un sistema de coordinadas para localizar los resultados de reconocimiento facial 3D. En la figura 6, se puede observar la dirección del sistema de coordenadas empleado y la disposición de la malla Candide que emplea.

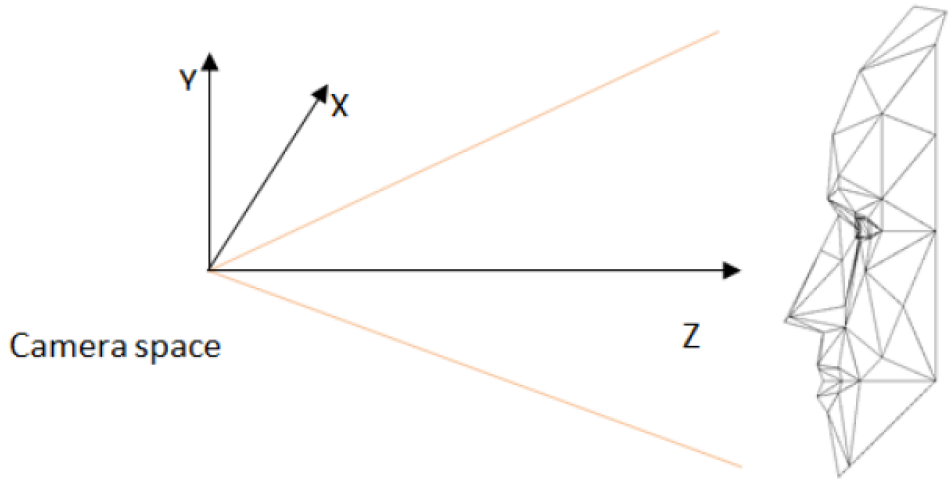


Figura 5. Sistema de coordenadas SDK [10].

Los datos que le llegan a SDK desde Kinect son las imágenes de color y profundidad, la claridad de éstas (disposición de la cara, forma, calidad de la imagen, …) influirá en el reconocimiento facial.

- Puntos y malla 2D

SDK asigna 87 puntos en zonas clave de la cara para obtener la expresión del usuario como se muestra en la figura 7, la estimación para localizar estos puntos se realiza mediante la aplicación del algoritmo AAM.

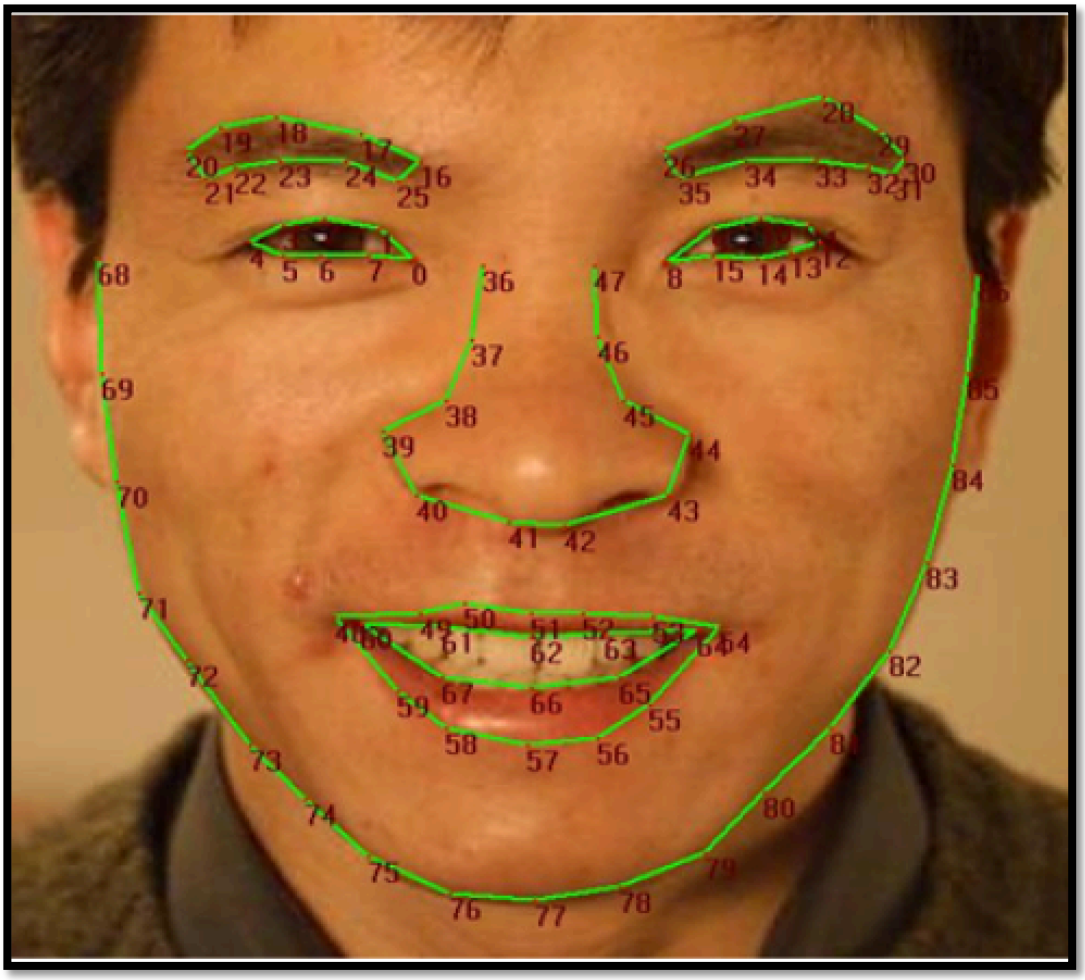


Figura 6. Puntos de reconocimiento de expresiones [10].

Adicionalmente, asigna 13 distribuidos en los centros de los ojos, en las comisuras de los labios, en la punta de la nariz y en un cubo que rodea la cara.

- Posición 3D de la cabeza

Como se ha comentado en el comienzo del documento, se localiza la posición de la cabeza con unas coordenadas con respecto de la cámara. Además, se localiza al usuario según 3 ángulos: pitch, roll y yaw, como se muestra en la figura 8. Estos ángulos oscilan entre -180º y 180º.

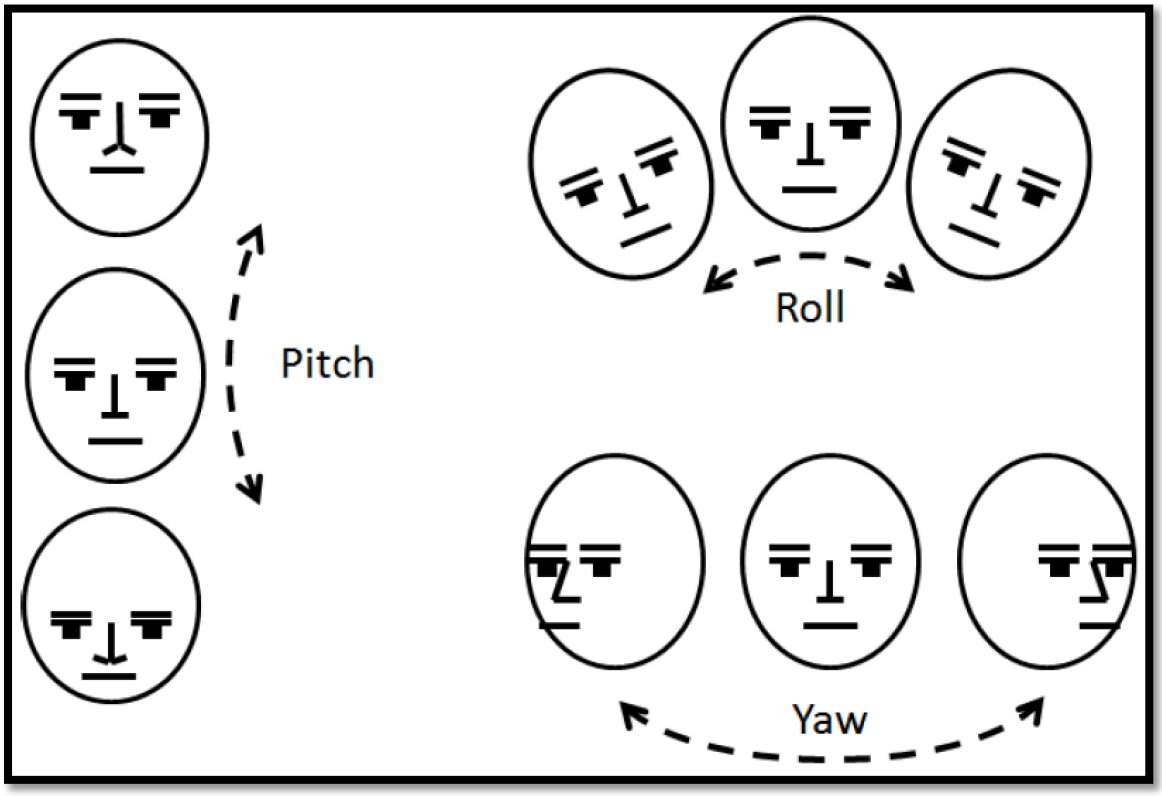


Figura 7. Ángulos de captación [10].

En la tabla 3 se muestra la interpretación que se realiza de los ángulos.

Tabla 3. Significado ángulos de captación [10].

Angle	Value
Pitch angle 0=neutral	-90 = looking down towards the floor +90 = looking up towards the ceiling Face Tracking tracks when the user's head pitch is less than 20 degrees, but works best when less than 10 degrees.
Roll angle 0 = neutral	-90 = horizontal parallel with right shoulder of subject +90 = horizontal parallel with left shoulder of the subject Face Tracking tracks when the user's head roll is less than 90 degrees, but works best when less than 45 degrees.
Yaw angle 0 = neutral	-90 = turned towards the right shoulder of the subject +90 = turned towards the left shoulder of the subject Face Tracking tracks when the user's head yaw is less than 45 degrees, but works best when less than 30 degrees

SDK, sitúa dos mallas sobre el usuario, una malla 2D, que es la mencionada anteriormente, y otra

malla 3D.

La malla 3D, consiste en 121 puntos distribuidos en distintas zonas de la cara. En estos puntos, se sitúa los vértices de la malla Candide-3. Esta malla, se adapta en tamaño y posición a la cara, gracias a la malla de puntos, formando así la malla 3D. La malla de puntos se muestra en la figura 9.

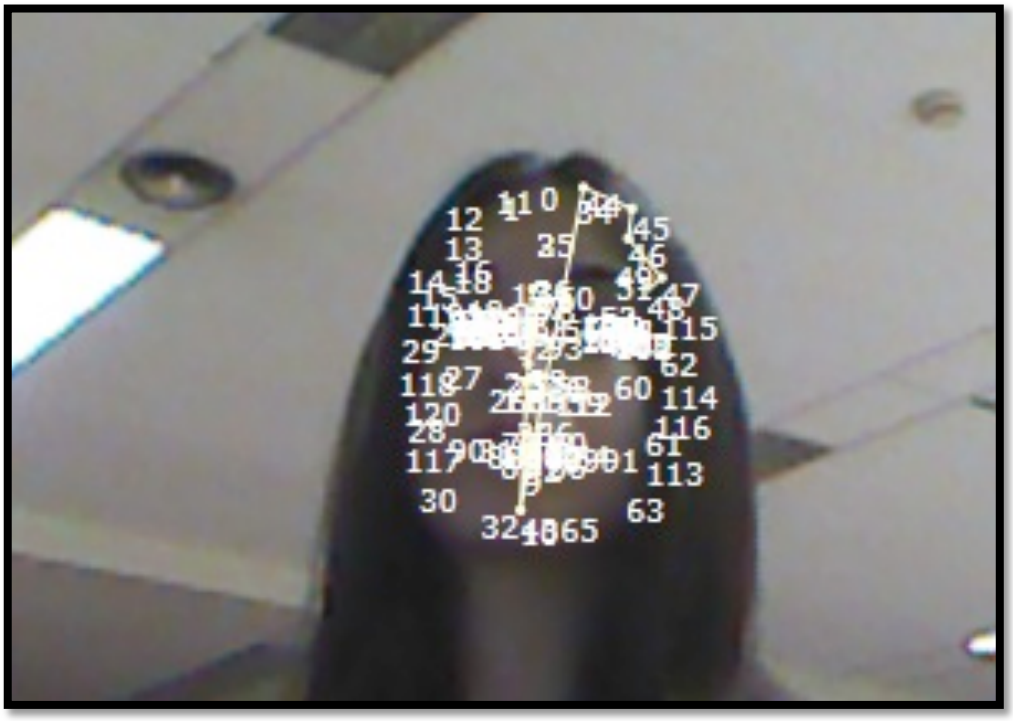


Figura 8. Malla 2D.

3. Entorno de programación

A continuación, se explica cómo realizar una implementación en el entorno de desarrollo SDK. Se presentan las distintas interfaces que soporta este entorno, así como, las funciones para el reconocimiento facial en Kinect.

En las prácticas realizadas, se ha implementado en lenguaje C#, el entorno de desarrollo SDK, se puede desarrollar también en C++.

3.1. Librerías necesarias

Para poder emplear los métodos anteriormente explicados, son necesarias dos librerías de la SDK. Estas librerías son:

- Microsoft.Kinect: esta librería es más global.
- Microsoft.Kinect.Toolkit.FaceTracking: esta librería, aporta todos los métodos propios del reconocimiento facial.

Para incluir las librerías, ha de disponerse de los archivos ".dll" correspondientes a las mismas, y siguiendo los pasos explicados en el tutorial [11] para realizar un proyecto en Microsoft Visual Studio. Una vez añadidas en referencias del proyecto, se debe añadir al comienzo del archivo ".xaml.cs" donde se implemente, las siguientes líneas:

```
using Microsoft.Kinect;

using Microsoft.Kinect.Toolkit.FaceTracking
```

3.2. Descripción de la API de SDK para *Face Tracking*

En la tabla 4, se presentan las cuatro principales interfaces de SDK para reconocimiento facial.

Tabla 4. Interfaces para reconocimiento facial.

Interfaz:	Representa:	Método para crear un objeto de este tipo:
IFTFaceTracker	Interfaz principal	FTCreateFaceTracker()
IFTResult	Resultado de las operaciones de reconocimiento facial	
IFTModel	Modelo de la cara 3D a medida	
IFTImage	Interfaz de ayuda que empaqueta varios *buffers* de imágenes	FTCreateImage()

Los métodos mostrados, son de la API para C#. Más adelante, se explicarán los problemas sucedidos con ellos.

Las estructuras de datos empleadas se recogen en la tabla 5.

Tabla 5. Estructuras de datos.

Estructura	Descripción
FT_SENSOR_DATA	Contiene toda la información de entrada requerida para detectar el usuario proveniente del sensor de Kinect.
FT_CAMERA_CONFIG	Contiene la configuración de vídeo o del sensor de profundidad cuyas imágenes son reconocidas.
FT_VECTOR2D	Contiene los puntos del vector 2D
FT_VECTOR3D	Contiene los puntos del vector 3D.
FT_TRIANGLE	Contiene el modelo 3D de la cara en triángulos.
FT_WEIGHTED_RECT	Contiene el rectángulo ponderado enviado por el SDK *Face Tracking* que encuadra la cara

3.2.1. IFTFaceTracker

El proceso de reconocimiento facial comienza llamando a la función StartTracking, la cual proporciona una imagen para la cara, determinando la orientación de la misma y comenzando su reconocimiento. El desarrollador puede dar un indicio de dónde se encuentra la cara mediante el parámetro pROI (ROI=*Region of Interest*) o pasar un valor NULL a la función para que busque la cara por toda la imagen.

Por otro lado, la orientación de la cara también puede proporcionarse al programa mediante el parámetro headPoints o derivarse de la orientación del esqueleto que ha sido analizado.

Cuando StartTracking reconoce con éxito la cara, o lo que es lo mismo, que se retorne un resultado con pFTResult, la aplicación continúa con el proceso llamando al método ContinueTracking, sino el programa seguirá buscando la cara hasta que el usuario pare el programa. Este método se ejecutará hasta que se pare la aplicación o, bien, ocurra un error en el proceso de reconocimiento facial, que será indicado por pFTResult.

Si existen diferentes personas frente Kinect, el reconocimiento múltiple facial se realiza mediante la invocación del método DetectFaces de IFTFaceTracker, que crea distintas instancias de IFTFaceTracker para cada cara detectada.

Se puede recuperar o ajustar SUs (Shape Units) con GetShapeUnits y SetShapeUnits, respectivamente.

En el anexo 1, se muestran las líneas de código para comenzar con el reconocimiento.

3.2.2. IFTImage

A través de FT_SENSOR_DATA, se proporciona los datos de la imagen, esta estructura de datos alberga características de la imagen como su formato, alto y ancho, tamaño o bytes por píxel.

Proporciona un buffer para almacenar la imagen y tener control sobre ella, teniendo la posibilidad de borrarla.

3.2.3. IFTResult

Se crea invocando IFTFaceTracker.CreateFTResult. Los métodos que proporciona son:

- GetStatus: indica si el reconocimiento se ha realizado con éxito.
- GetFaceRect: coordenadas del rectángulo que rodea la cara.
- Get2DShapePoints: coordenadas (x, y) de los 87 puntos que se distribuyen por la cara para el reconocimiento de expresiones que se presenta en la figura 7.

Esta interfaz, no se pudo crear en las prácticas, debido a la falta de conocimiento si se da implementada para C# o la invocación de la misma. En la API de SDK para C++, lo explicado anteriormente, sí se puede invocar de esa forma. Debido a este problema que se presentó en el desarrollo de la investigación, la implementación realizada empleando puntos en coordenadas 2D en la cara, se hizo con la malla de puntos creada para la adaptación de Candide-3.

3.2.4. IFTModel

Se crea con la llamada al método IFTFaceTracker::GetFaceModel(). Los métodos que proporciona son:

- GetSUCount, GetAUCount: devuelve el número de SU y AU en el modelo 3D.
- GetTriangles: devuelve la malla de triángulos del modelo 3D.
- GetVertexCount: devuelve el número de vértices en la malla 3D.
- Get3DShape: devuelve el modelo 3D adaptado a la cara.
- GetProjectedShape: devuelve el modelo 3D adaptado a la cara y proyectado en la imagen.

- Animation Units (GetAnimationUnitsCoefficients()) y Shape Units

Con el método GetProjectedShape, se obtienen el modelo 3D, de puntos donde luego se adaptarán los vértices que forman Candide-3. Con este método, se implementarán todos las aplicaciones. Este método, tiene dos formas de llamar a los puntos localizados: mediante nombres significativos y recorriendo el array que los almacena, en el anexo 2 se muestra cómo se llama por nombres. En el apartado "Implementaciones y pruebas", se explicará la forma que se trabaja con los datos que devuelve este método.
Los resultados son expresados por 6 AUs y 11 SUs siguiendo el modelo de la versión 3 de Candide, Candide-3.

En la tabla 6 se muestra qué valores se toman para las diferentes Animation Units y en la tabla 7 los números asignados a cada Shape Unit.

En el caso de las presentes prácticas, se han empleado las Animation Units o Action Units, para ello, se invoca el método GetAnimationUnitsCoefficients, éste devolverá, en función de expresión del usuario, un valor entre 1 y -1, según corresponda con la AU que tenga en ese momento.

Tabla 6. Significado cada AU [10].

AU Name and Value	Avatar Illustration	AU Value Interpretation
Neutral Face (all AUs 0)		
AU0 – Upper Lip Raiser (In Candid3 this is AU10)		0=neutral, covering teeth 1=showing teeth fully -1=maximal possible pushed down lip

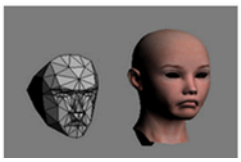
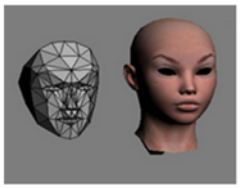
AU1 – Jaw Lowerer (In Candid3 this is AU26/27)		0=closed 1=fully open -1= closed, like 0
AU2 – Lip Stretcher (In Candid3 this is AU20)		0=neutral 1=fully stretched (joker's smile) -0.5=rounded (pout) -1=fully rounded (kissing mouth)
AU3 – Brow Lowerer (In Candid3 this is AU4)		0=neutral -1=raised almost all the way +1=fully lowered (to the limit of the eyes)
AU4 – Lip Corner Depressor (In Candid3 this is AU13/15)		0=neutral -1=very happy smile +1=very sad frown
AU5 – Outer Brow Raiser (In Candid3 this is AU2)		0=neutral -1=fully lowered as a very sad face +1=raised as in an expression of deep surprise

Tabla 7. Significado cada SU [10].

SU Name	SU number in Candide-3
Head height	0
Eyebrows vertical position	1
Eyes vertical position	2
Eyes, width	3
Eyes, height	4
Eye separation distance	5
Nose vertical position	8
Mouth vertical position	10
Mouth width	11
Eyes vertical difference	n/a
Chin width	n/a

4. Implementaciones y pruebas

Tras asentar todos los conocimientos sobre cómo funciona Kinect, se procede a comenzar a implementar nuevas posibilidades de reconocimiento. Para ello, se emplea como plantilla la aplicación FaceTrackingBasics-WPF proporcionada por el Toolkit de Kinect [12]. Esta aplicación, implementada en C#, será la base de todos los próximos trabajos desarrollados durante esta investigación.

La aplicación mencionada consiste en reconocer la cara del usuario y dibujar la malla 3D Candide ajustada al usuario. Se muestra, así, como esta malla se adapta a los movimientos que se realicen.

Se han realizado tres implementaciones: una basada en cinco expresiones detectadas por puntos, una que detecta las seis emociones explicadas en el apartado 2 mediante AUs y, por último, una que con los 19 puntos seleccionados por Face Group, detectar las mismas seis emociones.

A continuación, se explica cada implementación, así como los resultados obtenidos.

4.1. Primera implementación basada en puntos faciales

Una vez analizado el código de ejemplo, se procede a implementar la primera aplicación, que consiste en conseguir trabajar con los puntos de la malla de puntos, donde se sitúan los vértices de Candide. Como se ha explicado en el apartado anterior, estos puntos son obtenidos mediante la función GetProjected3DShape().

Empleando como ayuda el código de El Bruno [13], se realiza la modificación en el código para representar la malla de puntos, así como la numeración que se le proporciona a cada punto en el array que los almacena. Con la ejecución de este programa se obtiene lo mostrado en la figura 10.

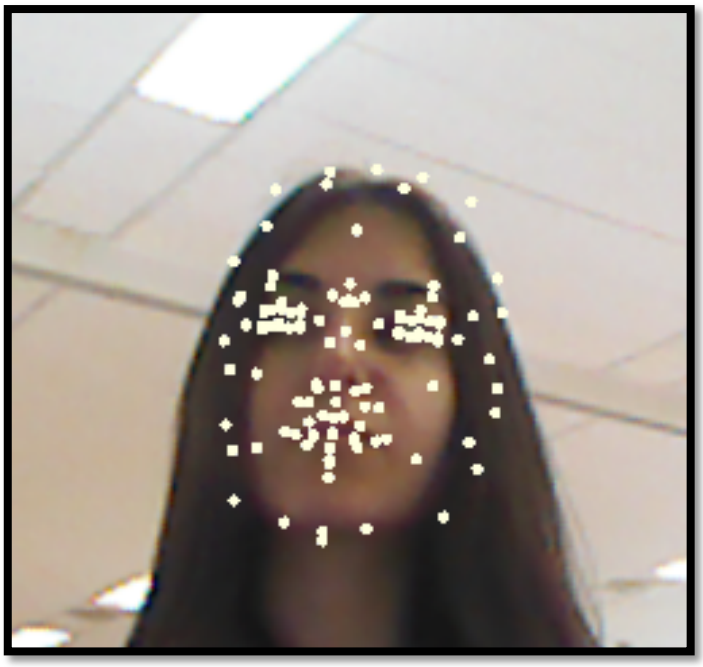


Figura 9. Malla 2D.

Como un sencillo programa de reconocimiento de emociones, se implementa un código de reconocimiento de movimientos [14]. Estos movimientos se basan en cinco puntos: uno en el labio superior (pto. 7), otro en el inferior (pto. 8), un punto en cada comisura (ptos. 31 y 64) y uno en la barbilla (pto. 10).

Mediante las coordenadas de los puntos mencionados, se comparan los cambios de posiciones y se detectan movimientos del usuario. Para tener una referencia de las coordenadas normales del usuario y detectar el movimiento, se almacena las primeras coordenadas de los puntos estudiados, así como las distancias de los puntos correspondientes a la boca.

La comparación que se ha realizado para la detección de los movimientos se han basado en el siguiente razonamiento:

- Detección de apertura de boca: para este movimiento se comparan los puntos superior e inferior del centro de la boca. En la primera detección de la cara del usuario se guarda la distancia entre ambos puntos, esto servirá de referencia para saber cuánto es la distancia cuando el usuario no abre la boca. Cuando la distancia sobrepasa la distancia de referencia, saldrá un mensaje anunciando el movimiento mencionado. Figura 11.

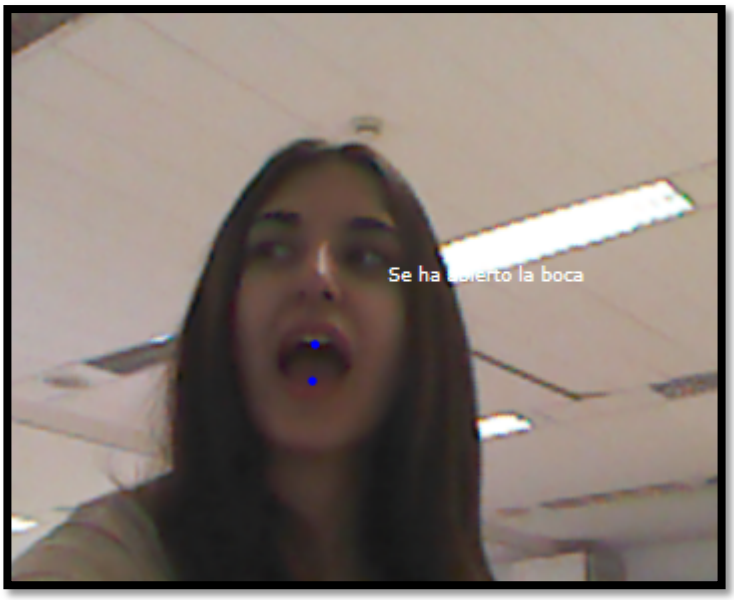


Figura 10. Detección de apertura de boca.

- Detección de la sonrisa: al igual que en el caso anterior, se guarda la distancia inicial entre los puntos de la comisura del usuario. Cuando la distancia detectada entre dichos puntos sobrepasa la distancia inicial, se avisará del movimiento del usuario. Figura 12.

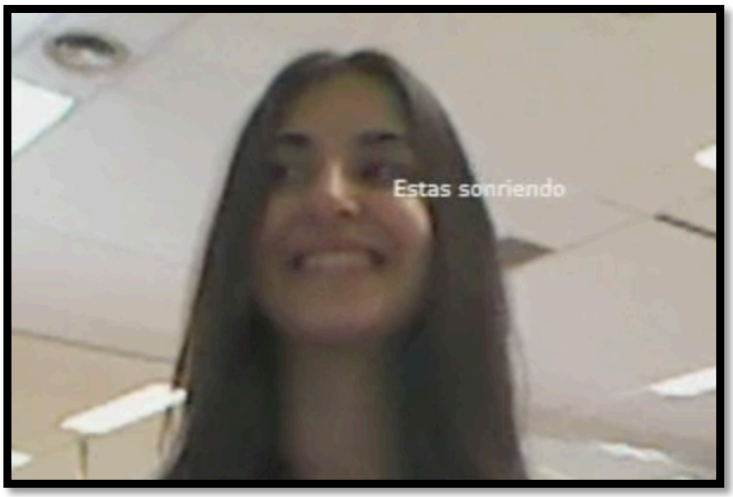


Figura 11. Detección de sonrisa.

- Detección subir cabeza: en este caso, el movimiento es detectado comparando la posición de la barbilla en el inicio de la detección y en el presente momento, si la posición es mayor en el eje Y que en la posición inicio, e detectará como que la cabeza se ha subido. Figura 13.

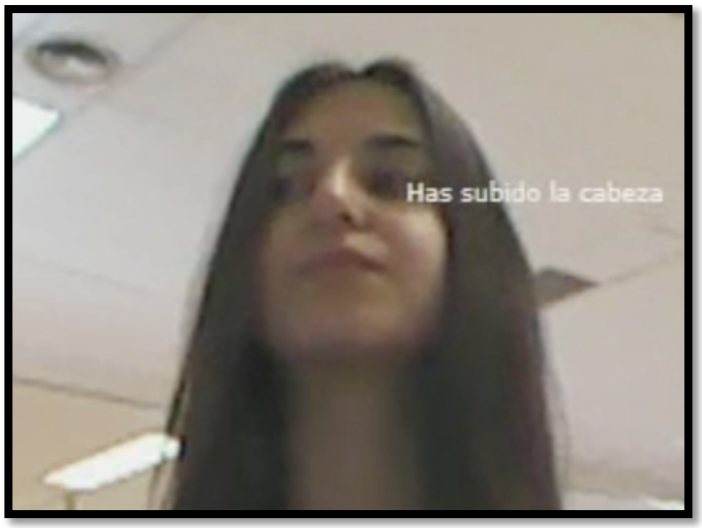


Figura 12. Detección subir cabeza.

- Detección bajar cabeza: al igual que en el caso anterior se comparan la posición del inicio comparado en el eje Y, con la posición en el presente, si en el momento presente la posición es menor, se interpreta como que la cabeza se ha bajado. Figura 14.

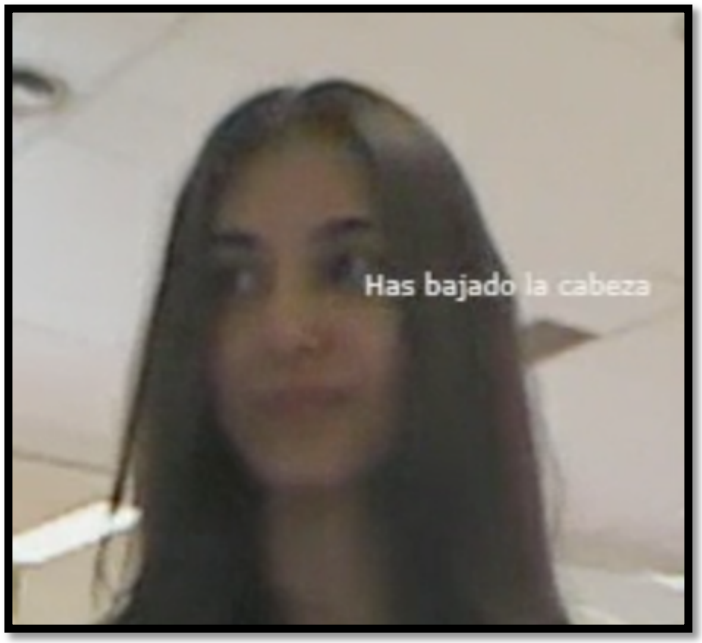


Figura 13. Detección bajar cabeza.

4.2. Reconocimiento de emociones con AUs

Por otra parte, se emplea la malla 3D para la finalidad buscada: reconocimiento de expresiones faciales [15]. Como se ha explicado en el apartado 3.1, las librerías propias de Kinect proporcionan métodos que ya detectan ciertos Action Units. El entorno de desarrollo, proporciona las

equivalencias entre los AUs detectados y el valor numérico con el que se representa, vista en la tabla 6.

Con esta información, se implementa un método de detección de emociones. Para establecer la relación entre la emoción detectada y las AUs detectadas, se ha empleado el trabajo realizado anteriormente por Almudena Gil y Diego Zapatero [2] donde según las tablas 8 y 9 se relacionan.

Tabla 8. Emoción y AUs [2].

Emoción	AUs (Programa)
Fear	1,8,13,15,20
Anger	3,4,13,14,15,17,18
Sadness	1,13
Disgust	2,3,4,11,14,17,18
Happy	3,4,12,15,20
Surprise	8,21

Tabla 9. AUs y su significado [2].

AUs (Programa)	AUs (FACS)	Denominación AUs	Puntos (de 19)
1	1	Interior de las cejas elevado	3,4
2	2	Exterior de las cejas elevado	1,6
3	-	Interior de las cejas bajado (INVENTADO)	3,4
4	4	Cejas bajadas	2,5
5	5	Párpado superior elevado	8,12
6	6	Mejillas elevadas =ojos entrecerrados	8,10,12,14
7	7	Párpados tensos	10,14
8	-	Cejas subidas (INVENTADO): AU1+AU2	2,5
9	9	Nariz arrugada	19
10	10	Labio superior elevado	16
11	11	Nasolabial: Labio superior elevado> 10 pixeles	16
12	12	Comisuras de la boca estiradas y elevadas	15,17
13	15	Comisuras de los labios hacia abajo	15,17
14	16	Labio inferior hacia abajo (0-5 pixeles)	18
15	20	Labios estrechados y estirados en horizontal (comisuras hacia los lados <- ->)	15,17
16	22	Labios crateriformes (comisuras hacia dentro, labios abiertos)	15,16,17,18
17	23	Labios apretados y encogidos (comisuras hacia dentro -> <-)	15,17
18	24	Labios comprimidos (superior e inferior)	16,18
19	25	Labio inferior hacia abajo (5-10 pixeles)	18
20	26	Labio inferior hacia abajo (10-30 pixeles)	18
21	27	Labio inferior hacia abajo (>30 pixeles)	18

Basándose en estas relaciones realizadas por Face Group, se traducen estos valores a los AUs de Kinect, mostrados en la figura 15. En la tabla 10, se muestran tales resultados.

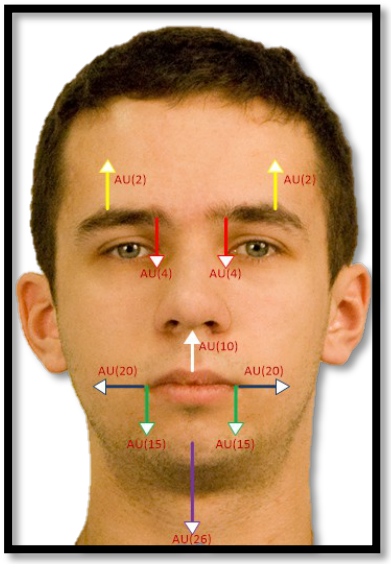


Figura 14. AUs de Kinect [16].

Tabla 10. Relación entre AUs de Kinect y emociones, basándose en el trabajo del Face Group.

Emoción	AUs (Imagen)
Fear	4,2,15,20
Anger	4,15, 20
Sadness	4,15
Disgust	2,4,10,26,20
Happy	2, 4, 15,20, 26
Surprise	2,4,26

Al igual que en el caso del programa anterior, se almacena los primeros AUs detectados para comparar el estado normal del usuario y el estado presente, y reconocer las emociones que sufre. En la figura 16, se muestra un ejemplo del funcionamiento de este programa.

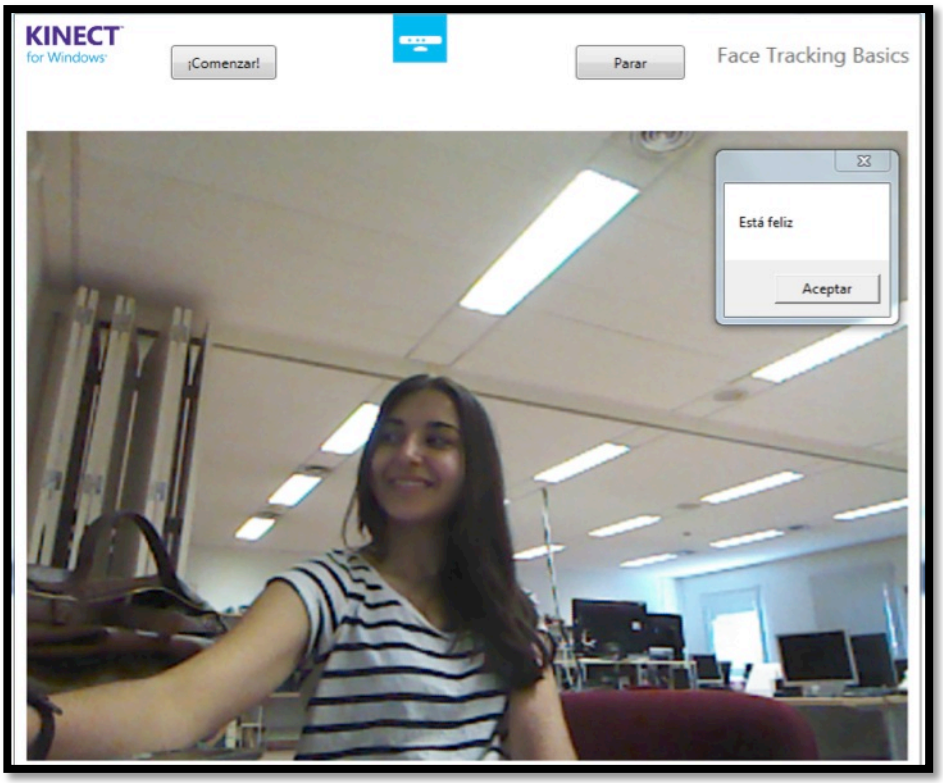


Figura 15. Aplicación de reconocimiento de emociones basada en AUs.

La interfaz de la aplicación, se basa en un botón que se pulsa cuando el usuario esté preparado con una expresión neutral, para comenzar el reconocimiento de expresiones. Otro botón para parar el reconocimiento cuando el usuario desee. Además de ello, al comienzo de la aplicación se mostrará un mensaje informativo, para dar las indicaciones de uso, y al detectar una emoción, saldrá una ventana emergente informando de la misma.

Como se ha explicado en el apartado anterior, los valores que deben tomar los parámetros devueltos por el método al cambiar de expresión serían los de la tabla 6, pero los valores que se alcanzan no corresponden con los que indican en el entorno SDK, debido a que tras una serie de pruebas realizadas que se presentarán a continuación, la detección de AUs no alcanza los valores indicados. En posiciones extremas, los valores máximos alcanzados en la mayoría de AUs no llegan a 1 o -1.

En el programa implementado de detección de emociones con la malla 3D se analizan los resultados con 74 muestras. Se obtienen los valores de las AUs reconocidas por Kinect y se comparan entre sí, mediante la desviación estándar de los valores y con el valor que debería aparecer al reconocer las AUs, según lo indicado por el entorno de desarrollo.

En la tabla 11, se muestran los resultados, apreciando que hay valores muy distintos a los que se quieren alcanzar.

Tabla 11. Pruebas del programa implementado para reconocimiento de expresiones faciales con malla 3D.

	Labio superior			Mandíbula		Longitud boca				Interior cejas			Comisuras			Exterior cejas		
	arriba	neutral	abajo	cerrada	abierta	neutral	joker	redondeada	beso	neutral	levantadas	bajadas	neutral	feliz	triste	neutral	triste	sorprendido
Desviación	0,10	0,08	0,21	0,06	0,13	0,08	0,09	0,19	0,10	0,06	0,12	0,03	0,07	0,06	0,09	0,08	0,06	0,04
Max	1,00	0,75	0,90	-0,01	0,61	-0,28	0,02	-0,30	-0,41	0,40	0,37	0,36	-0,04	-0,47	-0,37	0,25	0,23	0,41
Min	0,66	0,41	0,08	-0,26	-0,11	-0,63	-0,38	-1,00	-1,00	0,15	-0,06	0,19	-0,42	-0,69	-0,79	-0,03	-0,07	0,21
Valor indicado	1,00	0,00	-1,00	>0	1,00	0,00	1,00	-0,50	-1,00	0,00	-1,00	1,00	0,00	-1,00	1,00	0,00	-1,00	1,00

Como se muestra, la detección de AUs que se proporciona en las librerías de Kinect, no es muy precisa, por ejemplo, se analiza uno de los peores casos: apretar los labios bajando al máximo el superior. Cuando esta posición se produce, el valor que se devuelve para identificarla debería ser -1. Como se muestra en la tabla 11, tras la captación de 74 frames detectando esta Action Unit del usuario, el valor máximo alcanzando es 0.9 y el mínimo 0.08, habiendo una desviación estándar entre todos los valores alcanzados del 0.21. Esta desviación es muy elevada, lo que quiere decir que entre los resultados obtenidos, los valores numéricos varían mucho entre sí. Ninguno de los valores que se han generado con la detección se acerca al valor que indica SDK, acercándose más al AUs de subir el labio o, incluso, labio en posición neutral. En otros casos, se podría tomar que los valores negativos corresponden a la AU que corresponda con el valor -1, pero en este caso, tampoco se puede tomar esa cota.

Por ello, la precisión de los movimientos que se detecten, será muy complejo, ya que no se tienen límites fiables para establecer el cambio de movimientos.

4.2.1. Otra alternativa de reconocimiento por AUs en Kinect

Otra prueba realizada, fue cambiar los Action Units, para cada emoción, basándose en una investigación realizada por la tutora de estas prácticas, donde se escogió los AUs más empleados para cada emoción comparando distintos estudios.

Los resultados de estos estudios, se muestran para los 19 puntos que se trabajaban en Face Group, mostrados en la figura 17.

Figura 16. 19 puntos de Face Group.

A continuación, se relacionan los movimientos fundamentales de cada punto con un Action Unit, tabla 12.

Tabla 12. Relación entre movimientos de los 19 puntos y AUs.

Action Units	Puntos faciales y movimientos
1	3: hacia arriba 4: hacia arriba
2	1: hacia arriba 5: hacia arriba
4	1: hacia abajo 2: hacia abajo 3: hacia abajo 4: hacia abajo 5: hacia abajo 6: hacia abajo
5	8: hacia arriba 12: hacia arriba
6	10: hacia arriba 14: hacia arriba
7	8 y 10 cercanos 12 y 14 cercanos
10	16: hacia arriba
12	15: hacia arriba 16: hacia arriba 17: hacia arriba 18: hacia abajo
15	15: hacia abajo 17: hacia abajo
16	16: hacia abajo

20	15: hacia la izquierda 17: hacia la derecha
23	16 y 18 cercanos
25	16: hacia arriba 18: hacia abajo
26	18: hacia abajo
27	15: hacia la derecha 16: hacia arriba 17: hacia la izquierda 18: hacia abajo

El último paso, fue relacionar estos AUs con las emociones, comparando distintos estudios y escogiendo las AUs más empleadas para cada emoción, tabla 13.

Tabla 13. Relación entre AUs y emociones.

Emoción	Action Units
Anger	4, 5, 7, 23, 24
Disgust	9, 10
Fear	1, 2, 5, 25
Hapy	6, 12
Sadness	1, 4, 15, 17
Surprise	1, 2, 5, 26

Con esta información, se procedió a relacionar los AUs proporcionados por Kinect con las emociones, siguiendo la tabla 13. Primero se tradujo, la numeración de AUs que se emplea en los casos anteriores, a lo proporcionado por Kinect , tabla 14.

Tabla 14. Relación entre AUs investigados y AUs de Kinect.

Action Units	Action Units de Kinect
1	BrowLower = -1
2	BrowRaiser = +1
4	BrowLower = +1 BrowRaiser = -1
5	-
6	-
7	-
9	-
10	LipRaiser = +1
11	-
12	LipStretcher = +1 LipCornerDepressor = -1

15	-
16	-
17	LipStretcher = +1
20	LipStretcher = -0.5
22	-
23	-
24	-
25	-
26	.
27	JawLower<=0

Observando la anterior tabla, se puede intuir que Kinect no proporciona la suficiente información para la detección correcta por AUs.

El siguiente paso, fue realizar una comparativa entre los AUs que se podrían implementar para cada emoción, basándose en la última mejora, y la relación de AUs y emociones extraída del trabajo de Face Group, que se explicó anteriormente. Esta comparativa se muestra en la tabla 15.

Tabla 15. Comparativa entre las diferentes relaciones de AUs y emociones.

Emoción	AUs (mejora)	AUs (Face Group)
Anger	4, 5, 7, 23	4, 15, 20
Disgust	10	2, 4, 10, 26, 20
Fear	1, 2, 5, 25	2, 4, 15, 20
Happy	6, 12	2, 4, 15, 20, 26
Sadness	1, 4, 15	4, 15
Surprise	1, 2, 5, 26	2, 4, 26

Debido a la poca cobertura de AUs que Kinect aporta con sus librerías, la mejora de relaciones entre AUs y emociones, no se podría llevar a cabo, debido a que, hay emociones, como disgustado, que sólo tendría un AU para detectarse, lo que no sería nada preciso.

4.3. Detección de emociones empleando los 19 puntos

Para finalizar con el trabajo llevado a cabo en las prácticas, se realizó una implementación basada en los 19 puntos expuestos en la figura 17 [17], estudiados con anterioridad por Face Group en el CITSEM.

Siguiendo las relaciones presentadas en el apartado anterior, se lleva a cabo la implementación del programa, con la misma filosofía que las implementaciones anteriores: el usuario comienza con

una cara neutral, que se capturará y servirá de patrón de comparación para los diferentes cambios de expresiones que sufra (las que equivalen a las 6 emociones que se detectan) a lo largo del tiempo de ejecución.

Como se ha dicho anteriormente, existen dos formas de llamar a los puntos de la malla con la que se ha trabajado: una mediante el valor numérico de la posición que ocupa en el array q los almacena, y otra mediante nombres descriptivos que Kinect asigna. En el caso de esta implementación, se emplearon estos nombres, para que el código fuese más accesible y compresible. La relación entre los nombre aportados por Kinect y la numeración realizada por Face Group, se muestra en la tabla 16.

Tabla 16. Relación entre puntos Face Group y puntos Kinect.

Puntos (Face Group)	Puntos Kinect
1	LeftOfRightEyebrow
2	MiddleTopOfRightEyebrow
3	RightOfRightEyebrow
4	RightOfLeftEyebrow
5	MiddleTopOfLeftEyebrow
6	LeftOfLeftEyebrow
7	OuterCornerOfRightEye
8	MiddleTopRightEyeLid
9	InnerCornerRightEye
10	UnderMidBottomRightEyeLid
11	InnerCornerLeftEye
12	MiddleTopLeftEyeLid
13	OuterCornerOfLeftEye
14	UnderMidBottomLeftEyeLid
15	LeftCornerMouth
16	MiddleBottomUpperLip
17	RightCornerMouth
18	AboveChin

Para cada movimiento, como en el caso de la primera implementación realizada por puntos, se comparan las posiciones espaciales en X o Y, dependiendo del tipo de movimiento que se quiera analizar, como en el caso del primer programa de detección por puntos, se compara si los puntos se alejan o acercan, dependiendo de la expresión que se quiera detectar, siguiendo las pautas marcadas en la tabla 12. Tras la detección de la expresión, se muestra un mensaje por pantalla, cuando se detecta una emoción en el usuario.

Los resultados obtenidos en este caso, son muy desfavorables, existiendo emociones que no llegan a ser detectadas. Para la realización de estas pruebas, se procede de la misma forma que la anterior, se establece una emoción en la cara del usuario, sin cambiar de expresión, se capturan los resultados durante un tiempo estimado. En la tabla 17, se muestran el promedio de aciertos, así como la cantidad de pruebas realizadas por cada emoción. La cantidad de pruebas varió, debido al bajo número de aciertos en algunas emociones y el alto número de aciertos en otras.

En la tabla 18, se muestra el porcentaje de aciertos en cada prueba, así como las demás emociones detectadas erróneamente en cada prueba.

Tabla 17. Resultados del reconocimiento mediante 19 puntos.

Emoción	Felicidad	Disgusto	Enfado	Tristeza	Sorpresa	Miedo
Nº de pruebas	115	99	612	958	735	414
Probabilidad de acierto	33,9	100,0	0,0	0,0	0,0	0,0

Tabla 18. Emociones detectadas para cada expresión.

Emoción detectada/ Emoción probada	Felicidad	Disgusto	Enfado	Tristeza	Sorpresa	Mied
Felicidad	33,9	66,1	0,0	0,0	0,0	
Disgusto	0,0	100,0	0,0	0,0	0,0	
Enfado	33,9	33,0	0,0	0,0	0,0	
Tristeza	48,7	49,6	0,0	0,0	0,0	
Sorpresa	33,9	33,0	0,0	0,0	0,0	
Miedo	34,3	34,8	0,0	0,0	0,5	

5. Conclusiones y trabajo futuro

A lo largo de estas prácticas, se ha conseguido asentar una base sobre el reconocimiento facial, cómo desde los primeros trabajos hasta la actualidad ha mejorado a través de distintas técnicas. Se ha profundizado en especial, en el trabajo ya realizado sobre este tema en CITSEM [2], que ha servido de base para comenzar esta investigación.

Una vez que se han tenido claras ciertas definiciones fundamentales, se procedió a investigar este tema pero desde el punto de vista de Kinect, aprendiendo el funcionamiento de ésta y las librerías empleadas para dicho fin. Por último, para abordar el objetivo principal de las prácticas, el

reconocimiento de expresiones faciales mediante Kinect, se aprendió el lenguaje C# y se desarrollaron nuevas aplicaciones de reconocimiento, con la ayuda de las librerías y de las investigaciones realizadas sobre reconocimiento de emociones.

Con los resultados obtenidos, se presenta aun trabajo futuro, ya que habría que mejorar tanto los resultados en reconocimiento por AUs como por puntos. Las mejoras se centrarían en encontrar las mejores relaciones entre AUs y emociones, relaciones que restrinjan más las posibilidades, para no obtener tantas variaciones.

Otra mejora se presenta con la nueva Kinect, la cual aporta mucha más resolución, 1080p, frente a la resolución VGA con la que se ha trabajado. Mejora el campo de visión, además de reducir la distancia mínima a la que se tiene que situar el usuario de la cámara. También mejora su procesador, procesando 2 Gbps de datos. Puede detectar seis usuarios a la vez, así como, el pulso de un usuario.

Con todas las mejoras que presenta la nueva Kinect, se podría mejorar la precisión de los movimientos faciales capturados, así como incluir mejoras a las aplicaciones, como detectar las expresiones de varios usuarios a la vez.

6. Referencias

[1] Microsoft Developer Network, “Kinect for Windows SDK”, https://msdn.microsoft.com/en-us/library/hh855347.aspx

[2] Almudena Gil y Diego Zapatero, “Reconocimiento facial de emociones”, CITSEM, enero 2015.

[3] Bassem Seddik, Houda Maàmatou, Sami Gazzah, Thierry Chateau, Najoua Essoukri, Ben Amara, “Unsupervised Facial Expressions Recognition and Avatar Reconstructuib from Kinect”, Hammamet, Tunisia, March 2013.

[4] Rune A. Andersen, Kamal Nasrollahi, Thomas B. Moeslund, “Interfacing Assessment Using Facial Expression Recognition”, Aalborg University, Denmark.

[5] Ximena Cortés Silva, “Vikara: Plataforma de desarrollo y gestión de aplicaciones para detección de emociones”, Universidad de las Américas Puebla, 2013

[6] Daniel Ramos Gutierrez, “ESTUDIO CINEMÁTICO DEL CUERPO HUMANO MEDIANTE KINECT”,

EUITT, Madrid, Septiembre 2013.

[7] Mikael Rydfalk, "*Candide, a parameterised face*", Linköping University, Sweden, 1987.

[8] Jörgen Ahlberg, *CANDIDE, a parameterized face*, http://www.icg.isy.liu.se/candide/

[9] Iain Matthews and Simon Baker, "Active Appearance Models Revisited", The Robotics Institute Carnegie Mellon University.

[10] Microsoft Developer Network, *Face Tracking*, https://msdn.microsoft.com/en-us/library/jj130970

[11] Carlos Lázaro y Marcos López García, "Creación de un nuevo proyecto en Visual Studio 2013 compatible con Kinect en C#", CITSEM, 2014.

[12] Microsoft Developer Network, "Face Tracking Basics-WPF C# Sample", https://msdn.microsoft.com/en-us/library/jj131044.aspx

[13] El Bruno, "[#KINECT] HowTo: Utilizar Face Recognition con #KinectSdk (III)", http://elbruno.com/2012/08/09/kinect-howto-utilizar-face-recognition-con-kinectsdk-iii/

[14] Estefanía Ferreira, "PrimerProgramaPtos", CITSEM, 2015.

[15]Estefanía Ferreira, "AUs", CITSEM, 2015.

[16] Adam Wyrembelski, "Detection of the selected, basic emotions based on face expression using Kinect".

[17] Estefanía Ferreira, "19 puntos", CITSEM, 2015.

Anexos

1. ANEXO 1: Código para comenzar con el reconocimiento

En el main del programa, se ejecutan las siguientes líneas:

```
//Se inicia el sensor de Kinect
sensorChooser.KinectChanged += SensorChooserOnKinectChanged;
sensorChooser.Start();
//Se inicia la clase faceTrackingViewer, donde se realiza el reconocimiento facial
```

var faceTrackingViewerBinding = new Binding("Kinect") { Source = sensorChooser };

faceTrackingViewer.SetBinding(FaceTrackingViewer.KinectProperty, faceTrackingViewerBinding);

En la clase faceTrackingViewer, el reconocimiento facial se inicia de la siguiente forma:

this.faceTracker = new FaceTracker(kinectSensor);

FaceTrackFrame frame = this.faceTracker.Track(

colorImageFormat, colorImage, depthImageFormat, depthImage, skeletonOfInterest);

En estas líneas se crea un objeto del interfaz FACETRACKER. Se puede ver estas líneas de código en [14], [15] y [17].

2. ANEXO 2: Puntos de la malla para Candide-3 en Kinect

a. Lista de puntos

```
public enum FeaturePoint
  {
      TopSkull                    = 0,
      TopRightForehead            = 1,
      MiddleTopDipUpperLip        = 7,
      AboveChin                   = 9,
      BottomOfChin                = 10,
      RightOfRightEyebrow         = 15,
      MiddleTopOfRightEyebrow     = 16,
      LeftOfRightEyebrow          = 17,
      MiddleBottomOfRightEyebrow  = 18,
      AboveMidUpperRightEyelid    = 19,
      OuterCornerOfRightEye       = 20,
      MiddleTopRightEyelid        = 21,
      MiddleBottomRightEyelid     = 22,
      InnerCornerRightEye         = 23,
      UnderMidBottomRightEyelid   = 24,
      RightSideOfChin             = 30,
      OutsideRightCornerMouth     = 31,
      RightOfChin                 = 32,
      RightTopDipUpperLip         = 33,
      TopLeftForehead             = 34,
      MiddleTopLowerLip           = 40,
      MiddleBottomLowerLip        = 41,
      LeftOfLeftEyebrow           = 48,
      MiddleTopOfLeftEyebrow      = 49,
      RightOfLeftEyebrow          = 50,
      MiddleBottomOfLeftEyebrow   = 51,
      AboveMidUpperLeftEyelid     = 52,
      OuterCornerOfLeftEye        = 53,
      MiddleTopLeftEyelid         = 54,
      MiddleBottomLeftEyelid      = 55,
      InnerCornerLeftEye          = 56,
```

```
    UnderMidBottomLeftEyelid        = 57,
    LeftSideOfCheek                 = 63,
    OutsideLeftCornerMouth          = 64,
    LeftOfChin                      = 65,
    LeftTopDipUpperLip              = 66,
    OuterTopRightPupil              = 67,
    OuterBottomRightPupil           = 68,
    OuterTopLeftPupil               = 69,
    OuterBottomLeftPupil            = 70,
    InnerTopRightPupil              = 71,
    InnerBottomRightPupil           = 72,
    InnerTopLeftPupil               = 73,
    InnerBottomLeftPupil            = 74,
    RightTopUpperLip                = 79,
    LeftTopUpperLip                 = 80,
    RightBottomUpperLip             = 81,
    LeftBottomUpperLip              = 82,
    RightTopLowerLip                = 83,
    LeftTopLowerLip                 = 84,
    RightBottomLowerLip             = 85,
    LeftBottomLowerLip              = 86,
    MiddleBottomUpperLip            = 87,
    LeftCornerMouth                 = 88,
    RightCornerMouth                = 89,
    BottomOfRightCheek              = 90,
    BottomOfLeftCheek               = 91,
    AboveThreeFourthRightEyelid     = 95,
    AboveThreeFourthLeftEyelid      = 96,
    ThreeFourthTopRightEyelid       = 97,
    ThreeFourthTopLeftEyelid        = 98,
    ThreeFourthBottomRightEyelid    = 99,
    ThreeFourthBottomLeftEyelid     = 100,
    BelowThreeFourthRightEyelid     = 101,
    BelowThreeFourthLeftEyelid      = 102,
    AboveOneFourthRightEyelid       = 103,
    AboveOneFourthLeftEyelid        = 104,
    OneFourthTopRightEyelid         = 105,
    OneFourthTopLeftEyelid          = 106,
    OneFourthBottomRightEyelid      = 107,
    OneFourthBottomLeftEyelid       = 108
}
```

b. Cómo llamarlos desde el programa

Un ejemplo para emplear los puntos en el programa es el siguiente:

```
Point ojoIabajo = new Point(facePoints[FeaturePoint.UnderMidBottomRightEyelid].X,
facePoints[FeaturePoint.UnderMidBottomRightEyelid].Y);
```

Siendo facePoints el resultado de:

```
this.facePoints = frame.GetProjected3DShape();
```