



**EDUCACIÓN**

SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO  
NACIONAL DE MÉXICO

# Tecnológico Nacional de México

**Centro Nacional de Investigación  
y Desarrollo Tecnológico**

## Tesis de Maestría

**Generación de código para identificar gestos de la  
mano mediante la cámara Intel RealSense SR 300 y  
Project Gesture**

presentada por

**Ing. Humbertino Avilez Carpintero**

como requisito para la obtención del grado de  
**Maestro en Ciencias de la Computación**

Director de tesis

**Dr. Máximo López Sánchez**

Codirector de tesis

**Dr. Juan Gabriel González Serna**


Cuernavaca, Morelos, México. Abril 2022


Cuernavaca, Morelos, 28/marzo/2022

OFICIO No. DCC/025/2021  
Asunto: Aceptación de documento de tesis  
CENIDET-AC-004-M14-OFICIO


DR. CARLOS MANUEL ASTORGA ZARAGOZA  
SUBDIRECTOR ACADÉMICO  
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del C. **HUMBERTINO AVILEZ CARPINTERO**, con número de control M19CE053, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado "GENERACIÓN DE CÓDIGO PARA IDENTIFICAR GESTOS DE LA MANO MEDIANTE LA CÁMARA INTEL REALSENSE SR 300 Y PROJECT GESTURE", y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

  
DR. MÁXIMO LÓPEZ SÁNCHEZ  
Director de tesis

  
DR. JUAN GABRIEL GONZÁLEZ SERNA  
Codirector de Tesis

  
DR. DANTE MÚJICA VARGAS  
Revisor

  
DR. NOÉ ALEJANDRO CASTRO SÁNCHEZ  
Revisor

C.c.p. Depto. Servicios Escolares  
Expediente / Estudiante  
JCCS/lbm

**cenidet**  
Centro Nacional de Investigación y Desarrollo Tecnológico

Interior: Internado Palmar, C. Cuernavaca, Morelos  
Tel. 01 (777) 3427770, ext. 3201, e-mail: dco@tecnm.mx | tecnm.mx | cenidet.tecnm.mx

SEP TecNM CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO  
**RECIBIDO**  
30 MAR 2022  
SUBDIRECCIÓN ACADÉMICA

EDUCACIÓN  
29 MAR 2022  
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO  
SERVICIO ESCOLARES  
**RECIBIDO**



Cuernavaca, Mor.,  
No. De Oficio:  
Asunto:

30/marzo/2022  
SAC/60/2022  
Autorización de  
Impresión de tesis

**HUMBERTINO AVILEZ CARPINTERO**  
**CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS**  
**DE LA COMPUTACIÓN**  
**PRESENTE**

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado "GENERACIÓN DE CÓDIGO PARA IDENTIFICAR GESTOS DE LA MANO MEDIANTE LA CÁMARA INTEL REALSENSE SR 300 Y PROJECT GESTURE", ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

**ATENTAMENTE**  
Excelencia en Educación Tecnológica®  
"Educación Tecnológica al Servicio de México"



**DR. CARLOS MANUEL ASTORGA ZARAGOZA**  
**SUBDIRECTOR ACADÉMICO**

C. c. p. Departamento de Ciencias Computacionales  
Departamento de Servicios Escolares



CMAZ/CHG

---

## **Dedicatoria**

**Para papá y mamá que siempre estuvieron, están y estarán apoyándome...**

---

---

## Agradecimientos

A mis padres por su apoyo incondicional, por impulsarme a seguir creciendo en el ámbito académico.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico que me fue otorgado para desarrollar este trabajo de investigación.

Al Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) por darme la oportunidad de participar en el programa de maestría de ciencias de la computación.

Al Dr. Máximo López Sánchez por ser mi director de tesis y siempre brindándome su apoyo a lo largo de la maestría.

A mi codirector de tesis, Dr. Juan Gabriel González Serna por su aportación durante el desarrollo de la tesis.

A mis revisores el Dr. Dante Mújica Vargas y el Dr. Noé Alejandro Castro Sánchez por sus observaciones y consejos para mejorar el desarrollo de la tesis.

A mis compañeros del centro de investigación por su apoyo, consejos y amistad a lo largo de la maestría.

---

---

## Resumen

En el desarrollo de software hay tres fases que son especialmente importantes: la elicitación de requerimiento, el diseño del sistema y la codificación del mismo. Estas fases de desarrollo tienen características específicas que las diferencian unas de las otras, y cada una de ellas contiene un cierto nivel de impacto si no se realizan de una manera adecuada.

En la fase de codificación se lleva a cabo la escritura de las sentencias que harán que el sistema muestre su funcionalidad, aquí puede presentarse un problema y que en muchas ocasiones el tener fallos o defectos de escritura puede tener un gran efecto en el tiempo y costo en la reparación o corrección del código (Saini A, Security Consultant y Cigital/Synopsys, 2017).

Se reporta que el costo de arreglar un defecto encontrado durante la etapa de implementación, es aproximadamente seis veces más caro que uno encontrado durante el diseño; el costo de arreglar un error encontrado después del lanzamiento del producto es de cuatro a cinco veces más que descubrir uno es la etapa de diseño y hasta cien veces más de uno encontrado en la etapa de mantenimiento. En otras palabras, el costo de un defecto se incrementa exponencialmente a medida que el software avanza en su ciclo de vida (Saini A, Security Consultant y Cigital/Synopsys, 2017).

Por otra parte, la evolución de la tecnología ha tomado muchos enfoques, uno de ellos es en el mejoramiento de la manipulación de sistemas computacionales, más que nada, ayudar a los usuarios que no cuentan con la experiencia de saber cómo utilizar una computadora y así manipular el software que desean. Por eso se ha tomado como alternativa de solución el desarrollar interfaces naturales de usuario, donde estos sistemas solo requieren que se utilicen los movimientos corporales para su manipulación y facilitarle al usuario hacer uso de la tecnología (Ceruzzi, 2018).

En este trabajo de investigación el objetivo es generar código que reconozca los gestos realizados de la mano frente un sensor de profundidad llamando *Intel RealSense SR300*, y que utilizando el *SDK* de *Project Gesture*, genere código en lenguaje de programación, de los gestos que fueron capturados por el sensor, de esta manera se desea disminuir los defectos de escritura en la fase de programación.

El sistema inicia con la captura de la pose de la mano frente al sensor, el cual manda información que describe el comportamiento de la mano y es recibida por un sistema difuso, el cual mediante un conjunto de reglas ya establecidas devolverá una cantidad de poses primitivas que juntas describirán los elementos de la mano que forman la pose que está realizando la mano. Estas poses primitivas ayudarán en el proceso de la generación de código, puesto que son utilizadas por la definición de una gramática libre de contexto para darle sentido y posteriormente hacer la conversión a sus respectivos fragmentos de código. Estos fragmentos de códigos, podrán ser utilizados en interfaces naturales de usuario que se desarrollen con la librería *Project Gesture*.

---

---

## Abstract

In software development there are three phases that are particularly important: requirement elicitation, system design and system coding. These development phases have specific characteristics that differentiate them from each other, and each of them contains a certain level of impact if not performed properly.

In the coding phase, the writing of the sentences that will make the system show its functionality is carried out, here a problem can arise and that in many occasions having failures or defects in writing can have a great effect on the time and cost in the repair or correction of the code (Saini A, Security Consultant y Cigital/Synopsys, 2017).

It is reported that the cost of fixing a defect found during the implementation stage is approximately six times more expensive than one found during design; the cost of fixing a bug found after product release is four to five times more than discovering one in the design stage and up to one hundred times more than one found in the maintenance stage. In other words, the cost of a defect increases exponentially as the software progresses through its life cycle (Saini A, Security Consultant y Cigital/Synopsys, 2017).

On the other hand, the evolution of technology has taken many approaches, one of them is to improve the manipulation of computer systems, more than anything else, to help users who do not have the experience to know how to use a computer and manipulate the software they want. That is why it has been taken as an alternative solution to develop natural user interfaces, where these systems only require the use of body movements for manipulation and facilitate the user to make use of technology (Ceruzzi, 2018).

In this research work the objective is to generate code that recognizes the gestures made by the hand in front of a depth sensor called Intel RealSense SR300, and using the Project Gesture SDK, generate code in programming language, of the gestures that were captured by the sensor, in this way we want to reduce the defects of writing in the programming phase.

The system consists first of capturing the gestures one at a time, so that an algorithm automatically generates its code fragment in `c#`, which describes its behavior and then assigns a name to identify the gesture, these gestures are stored in a database so that after capturing several gestures, they can be emptied into a file and easily make use of it from some programming code of a natural user interface supported by the Project Gesture library.

---

---

# Índice

<b>1. Introducción .....</b>	<b>1</b>
<b>1.1. Planteamiento del problema.....</b>	<b>1</b>
<b>1.2. Justificación.....</b>	<b>2</b>
<b>1.3. Objetivos.....</b>	<b>3</b>
1.3.1 Objetivo general.....	3
1.3.2 Objetivo específico.....	3
<b>1.4. Alcances y limitaciones .....</b>	<b>4</b>
1.4.1 Alcances .....	4
1.4.2 Limitaciones .....	4
<b>1.5. Antecedentes .....</b>	<b>4</b>
1.5.1 Generación de palabras a partir de la lengua de señas mexicana .....	4
1.5.2 Reconocimiento de movimientos de manos para la manipulación de una interfaz computacional .....	5
<b>2. Marco teórico.....</b>	<b>7</b>
<b>2.1. Lenguaje corporal .....</b>	<b>7</b>
<b>2.2. Posición de la mano .....</b>	<b>7</b>
<b>2.3. SDK .....</b>	<b>7</b>
<b>2.4. NUI (Natural User Interface).....</b>	<b>8</b>
<b>2.5. Generación de código .....</b>	<b>8</b>
<b>2.6. Sensor de profundidad.....</b>	<b>8</b>
<b>2.7. Lógica difusa.....</b>	<b>9</b>
2.7.1 Conjuntos difusos.....	9
2.7.2 Operaciones de conjuntos difusos.....	9
2.7.3 Controlador difuso .....	10
<b>2.8. Gramática.....</b>	<b>14</b>
<b>2.9. Distancia euclídea .....</b>	<b>16</b>
<b>2.10. Matriz de confusión .....</b>	<b>17</b>
<b>3. Estado del arte.....</b>	<b>20</b>
<b>4. Metodología de solución.....</b>	<b>38</b>
<b>4.1. Project Gesture.....</b>	<b>40</b>
4.1.1 Composición de la mano con referencia al SDK Project Gesture.....	41
4.1.2 Sintaxis para definir una posición de mano en Project Gesture .....	42
<b>4.2. Investigación y análisis de métodos para la generación de código .....</b>	<b>44</b>
4.2.1 Comentarios.....	51

---



---

<b>4.3. Análisis y evaluación de un método que genere código a partir de la definición de una posición de mano .....</b>	<b>52</b>
4.3.1 Evaluación .....	52
<b>4.4. Poses primitivas para la formación de un gesto .....</b>	<b>55</b>
<b>4.5. Almacenamiento de gestos mediante poses primitivas en base de datos.....</b>	<b>60</b>
4.5.1 Captura de gestos .....	60
4.5.2 Diseño de base de datos .....	67
<b>4.6. Generación de código .....</b>	<b>68</b>
4.6.1 Definición de gramática .....	68
4.6.2 Generación de código .....	70
<b>4.7. Interfaz.....</b>	<b>71</b>
<b><i>5. Pruebas y resultados .....</i></b>	<b><i>77</i></b>
<b>5.1. Pruebas .....</b>	<b>77</b>
5.1.1 Defectos de escritura .....	77
5.1.2 Funcionalidad de fragmentos de código en <i>Project Gesture</i> .....	79
<b>5.2. Resultados.....</b>	<b>80</b>
5.2.1 Métricas .....	80
5.2.2 Pruebas con otros usuarios.....	82
<b><i>6. Conclusiones.....</i></b>	<b><i>86</i></b>
<b>6.1. Trabajos futuros .....</b>	<b>87</b>
<b><i>7. Bibliografía.....</i></b>	<b><i>89</i></b>
<b><i>Anexos.....</i></b>	<b><i>94</i></b>

---

---

# Índice de figuras

Figura 1: Relación del costo de reparación de defectos basado en el tiempo de la detección (Sanket, 2019). .....	2
Figura 2: Tipo de falla de software por mes en 2017 (Freyja,2017). .....	3
Figura 3: Aplicación SignMx (Romero, 2017). .....	4
Figura 4: Código para detectar gestos (Molina J., 2018). .....	5
Figura 5: Flujo del sistema de reconocimiento de gestos (Molina J., 2018). .....	5
Figura 6: Intel RealSense SR300 (Giang T., 2017). .....	8
Figura 7: Componentes de un sistema difuso. ....	10
Figura 8: Método centroide (Zamora, G., 2015). .....	13
Figura 9: Método máximo central (Zamora, G., 2015). .....	13
Figura 10: Método máximo más pequeño (Zamora, G., 2015). .....	13
Figura 11: Método máximo más grande (Zamora, G., 2015). .....	14
Figura 12: Método bisector de área (Zamora, G., 2015). .....	14
Figura 13: Matriz de confusión (Barrios, J., 2019). .....	17
Figura 14: Marcos coordinados del sistema. Los ejes X, Y, Z son rojo, verde y azul respectivamente (Razjigaev, 2017). .....	21
Figura 15: Interfaz gráfica para el reconocimiento de gestos usando redes neuronales (Mohiminul, 2017). .....	21
Figura 16: Arquitectura del sistema (Jimenez, 2017). .....	22
Figura 17: Esquema del enfoque GDL (Tomasz Hachaj, 2014). .....	23
Figura 18: Partes del sistema generador de código (Amanquah, 2017). .....	24
Figura 19: Vista del sistema (Feng et al, 2014). .....	25
Figura 20: Reconocimiento de gestos (Feng et al, 2014). .....	26
Figura 21: Gestos simbólicos (Hao y li,2014). .....	27
Figura 22: TraceMatch: Sistema de reconocimiento de gestos (Clarke et al,2016). .....	27
Figura 23: Procedimiento de detección (Clarke et al,2016). .....	28
Figura 24: Referencia de áreas entre dispositivos (Bellino et al, 2016). .....	29
Figura 25: Metodología de solución. ....	38
Figura 26: Composición de gestos con Project Gesture (Krupka E. et al., 2017). .....	41
Figura 27: Umple Online (uOttawa, 2018). .....	45
Figura 28: Diagrama UML de un gesto (uOttawa, 2018). .....	45
Figura 29: Diagrama de gesto UML a XML (uOttawa, 2018). .....	46
Figura 30: Generación de código java (uOttawa, 2018). .....	46
Figura 31: Aplicación de recolección de datos. ....	47
Figura 32: Generación de código de la base de datos. ....	48
Figura 33: Ejemplo de máquinas de estado con “Umple Online” (uOttawa, 2018). .....	49
Figura 34: Generación de código de una máquina de estado (uOttawa, 2018). .....	49
Figura 35: Gesto alto. ....	50
Figura 36: Imagen a analizar. ....	50
Figura 37: Detección de gesto con “Template Matching” (Rosebrock A., 2021). .....	51
Figura 38: Código de gesto detectado. ....	51
Figura 39: Ejes descriptores de la palma de la mano. ....	52
Figura 40: Representación gráfica de conjuntos difusos. ....	53

---

---

Figura 41: Orientación de la mano. ....	56
Figura 42: Componentes de un sistema difuso. ....	57
Figura 43: Orientación y Dirección: Adelante. ....	57
Figura 44: Orientación y Dirección: Abajo. ....	57
Figura 45: Orientación y Dirección: Atrás. ....	58
Figura 46 : Orientación y Dirección: Arriba. ....	58
Figura 47: Orientación y Dirección: Izquierda. ....	58
Figura 48: Orientación y Dirección: Derecha. ....	59
Figura 49: Dedo abierto y doblado. ....	59
Figura 50: Dirección de dedos. ....	60
Figura 51: Interacción de dedos. ....	60
Figura 52: Proceso para la generación de código de gestos ....	61
Figura 53: Procedimiento de generación de código de gesto de mano. ....	61
Figura 54: Fragmento de código que describe el comportamiento del gesto de mano. ....	62
Figura 55: Información proporcionada por el sensor. ....	62
Figura 56: Conjuntos difusos de variables de entrada. ....	63
Figura 57: Conjuntos difusos de variables de salida. ....	64
Figura 58: Conjuntos difusos de variables de entrada del sistema difuso de los dedos. ....	65
Figura 59: Conjuntos difusos de la variable de salida del sistema difuso de los dedos. ....	66
Figura 60: Distancia para la interacción entre los dedos. ....	67
Figura 61: Distancia para la pose de “Abierto” o “Doblado”. ....	67
Figura 62: Base de datos para almacenar primitivas de los gestos ....	67
Figura 63: Producciones de la primera gramática. ....	69
Figura 64: Producciones de la segunda gramática. ....	69
Figura 65: Sistema CRUD de gestos de mano. ....	71
Figura 66: Módulo de interfaz para capturar gesto. ....	72
<i>Figura 67: Módulo de interfaz: gestos guardados. ....</i>	<i>72</i>
Figura 68: Archivos de gestos. ....	73
Figura 69: Archivo de texto simple de gestos. ....	73
Figura 70: Archivo XML de gestos. ....	73
Figura 71: Módulo de interfaz: eliminar gesto. ....	74
Figura 72: Módulo de interfaz: modificar gesto. ....	75
Figura 73: Compilador de código de gestos. ....	77
Figura 74: Grupo de señas de números (Esther M. & González R. ,2011). ....	78
Figura 75: Grupo de señas del Alfabeto (Esther M. & González R. ,2011). ....	78
Figura 76: Resultado de generación de código. ....	79
Figura 77: Activación del código del gesto ....	79
Figura 78: Verificación de funcionalidad de código. ....	80
Figura 79: Usuarios de prueba. ....	82

---

---

## Índice de tablas

Tabla 1: Caracteres especiales de expresiones regulares.....	16
Tabla 2: Comparación de artículos 1-1.....	33
Tabla 3: Comparación de artículos 1-2.....	34
Tabla 4: Comparación de artículos 1-3.....	35
Tabla 5: Análisis de métodos de generación de código.....	51
Tabla 6: Conjunto de reglas para el sistema difuso.....	54
Tabla 7: Resultados de sistema difuso.....	55
Tabla 8: Conjunto de reglas para la palma de la mano.....	64
Tabla 9: Conjunto de reglas para la dirección de los dedos.....	66
Tabla 10: Relación de primitivas de Mano.....	70
Tabla 11: Relación de primitivas de Orientación y Dirección.....	70
Tabla 12: Relación de primitivas de Dedo.....	70
Tabla 13: Relación de primitivas Forma.....	70
Tabla 14: Relación de primitivas Interacción.....	71
<i>Tabla 15: Matriz de confusión.....</i>	<i>80</i>
<i>Tabla 16: Métricas: precision, accuracy y recall.....</i>	<i>81</i>
<i>Tabla 17: Resultados de usuario 1 y 2.....</i>	<i>83</i>
<i>Tabla 18: Resultados de usuarios 3 y 4.....</i>	<i>83</i>
<i>Tabla 19: Resultado final.....</i>	<i>84</i>

---

---

# Capítulo I

## Introducción

---

# 1. Introducción

En la actualidad, la generación de código automático durante el desarrollo de software ha sido de gran beneficio para los programadores, debido a que ofrece facilidades que permiten reducir los defectos que se insertan al escribir código. Las primeras aplicaciones de los generadores de código corresponden a los compiladores, los que generan código máquina a partir de un lenguaje de programación de alto nivel (escritura de programación adecuada a la capacidad cognitiva humana) (Tapia, Antonio, Osmolik y Vlasdislay, 2019). Otro ejemplo de la generación de código son las herramientas CASE (Ingeniería de Software Asistida por Computadora), ya que a partir de diagramas o modelos se puede generar código en un lenguaje específico, dando como resultados la optimización y disminución de defectos en la etapa de codificación, esto aun sin ser un profesional en el desarrollo de una aplicación y menos de conocer un lenguaje específico de programación (Battaglia, Neil, Fernández y Milanese, 2019).

Debido al avance de la tecnología, la interacción entre hombre – máquina, ya no depende únicamente de los expertos en el tema, tanto en su programación como en su control, sino, también con los lenguajes de alto nivel; dicha interacción facilita el desarrollo de aplicaciones mediante imágenes. Por esto se ha convertido en una práctica habitual utilizar metáforas visuales utilizando interfaces de usuario. En sus inicios, alrededor de los años 80's, las primeras interfaces (Interfaz de texto) se controlaban a partir de un apuntador y un teclado, posteriormente se desarrollaron interfaces gráficas (*GUI-Graphical User Interface*), interfaces táctiles, interfaces por voz y las interfaces naturales de usuario (*NUI*). Las interfaces naturales de usuario proporcionan a los usuarios un uso intuitivo en el control de los sistemas, un grado de realismo en la intervención de usuarios sobre un entorno virtual, permiten que personas con discapacidades puedan tener un mejor y más fácil acceso al aprendizaje (Ceruzzi, 2018).

En este trabajo de investigación se plantea el desarrollo de un sistema para la generación automática de código de programación (*C#*), con la finalidad de reducir los defectos que se insertan en la escritura de código en la etapa de su programación, por medio del reconocimiento de las posiciones de la mano derecha y su interpretación para la generación de código. Se hará uso de la cámara con sensor de profundidad *SR300* para la detección de las posiciones de la mano derecha, que en combinación con el sistema que se desarrollará, obtendrá la representación de la posición de la mano en forma de código.

## 1.1. Planteamiento del problema

El desarrollo de software para lograr automatizar procesos es una de las partes esenciales en la tecnología actual. Las formas de lograr un buen trabajo computacional requieren de conocimientos suficientes en un ambiente o lenguaje de programación, esto cuando el desarrollo se realiza desde la vía inicial de un diseño arquitectónico, siguiendo con un diseño detallado y terminando con la codificación correspondiente en un lenguaje de computación que sea procesado por la computadora. Se resalta que el desarrollador debe tener los conocimientos desde el modelado de sistemas, diseño detallado y manejo de lenguajes de programación. Para poder contar con una aplicación que un usuario final pueda operar y/o explotar. ¿Pero qué sucede cuando se quiere iniciar desde el punto de

---

operación o explotación?, esto es, no tener el modelo, diseño o código de la aplicación. (Iftekhar N, et al.,2019).

Cuando se tienen aplicaciones tales como las interfaces naturales, donde el proceso consiste en utilizar dispositivos que permitan reconocer gestos realizados por las manos, regularmente se lleva a cabo el proceso mencionado en el párrafo anterior, pero si se desea realizar el proceso inverso, principalmente cuando no se tiene mucho conocimiento de un lenguaje de programación, es importante que se tenga el soporte de un generador de código para obtener los resultados esperados.

Es por esto que el problema a resolver consiste en: **identificar los gestos realizados por las manos en una interfaz para generar código de programación.**

## 1.2. Justificación

Cuando se requiere trasladar una idea o situación real a un medio digital, se hace uso de un lenguaje de programación, el cual requiere que la persona que vaya a realizar esta actividad cuente con experiencia básica o avanzada sobre el lenguaje a codificar. Cuando la experiencia es mínima o media, se insertan muchos defectos en el código programado, lo que hace que el tiempo para su terminación sea más alto (Krasner H,2018).

El lema de “prevenir es mejor que curar” se aplica a los defectos en el ciclo de vida del desarrollo de un software (véase Figura 1). Los defectos, como es definido por los desarrolladores de software, son variaciones de un atributo deseado, por lo tanto, hacen que el software no cumpla con los requisitos y que los usuarios finales no estén satisfechos. De acuerdo con Computer Business Review, el Instituto de Ciencias de Sistemas de IBM, ha reportado que el costo de arreglar un defecto encontrado durante la etapa de implementación, es aproximadamente seis veces más caro que uno encontrado durante el diseño; el costo de arreglar un error encontrado después del lanzamiento del producto es de cuatro a cinco veces más que descubrir uno es la etapa de diseño y hasta cien veces más de uno encontrado en la etapa de mantenimiento. En otras palabras, el costo de un defecto se incrementa exponencialmente a medida que el software avanza en su ciclo de vida. (Saini A, Security Consultant y Cigital/Synopsys, 2017)

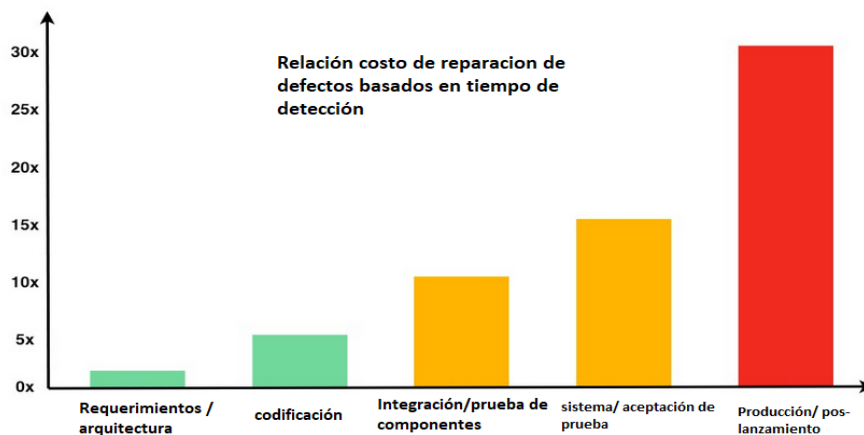


Figura 1: Relación del costo de reparación de defectos basado en el tiempo de la detección (Sanket, 2019).

Los defectos de software exponen a los usuarios finales a un software lento y con defectos. O en el peor de los casos, comprometer la seguridad de sus productos. De acuerdo a Tricentis, la mayoría de las fallas de software en el gobierno, finanzas, comercio, minoristas, servicios y transporte en 2017 se deben a los defectos de software, como se puede apreciar en la siguiente figura: (véase Figura 2)

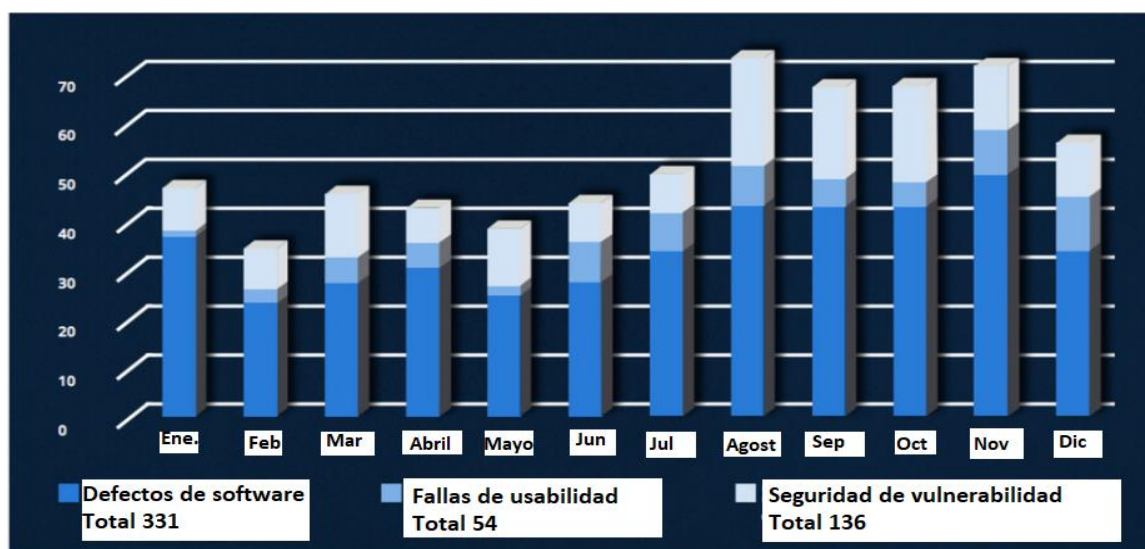


Figura 2: Tipo de falla de software por mes en 2017 (Freyja,2017).

## 1.3. Objetivos

### 1.3.1 Objetivo general

Desarrollar un sistema que detecte los movimientos de la mano derecha y que genere el código fuente correspondiente a las posiciones de esa mano, reduciendo los defectos en la etapa de su programación.

### 1.3.2 Objetivo específico

- Definir un proceso que genere el código automáticamente a partir de cada posición de las manos.
- Realizar un sistema que permita identificar las posiciones de las manos
- Crear un archivo en la cual se almacenará el código que se genere por cada posición.
- Verificar que el código obtenido no presente defectos de escritura de programación.



## 1.4. Alcances y limitaciones

### 1.4.1 Alcances

- El sistema será capaz de detectar las posiciones de la mano.
- El sistema se basará en el apoyo de la herramienta Project Gesture
- El sistema generará y almacenará un solo código en el lenguaje de programación C# correspondiente a cada posición de las manos.
- El sistema podrá asignarle un significado (etiqueta) al código de cada posición de la mano.

### 1.4.2 Limitaciones

- El sistema no será capaz de almacenar una posición inhabitual de la mano.
- El sistema no será capaz de detectar un gesto en exteriores.
- El sistema no creará programas ni aplicaciones mediante la generación de código.
- El sistema no será capaz de darle una función al gesto almacenado.
- El sistema no será capaz de detectar una mano con deficiencias o carencias de dedos.

## 1.5. Antecedentes

### 1.5.1 Generación de palabras a partir de la lengua de señas mexicana

En este trabajo se presentó una aplicación web que lleva por nombre SignMx (véase Figura 3) la cual consiste en reconocer las señas realizadas ante un dispositivo HCI (Human Computer Interfaces) para luego ser escritas y reproducidas a voz (Romero, 2017).

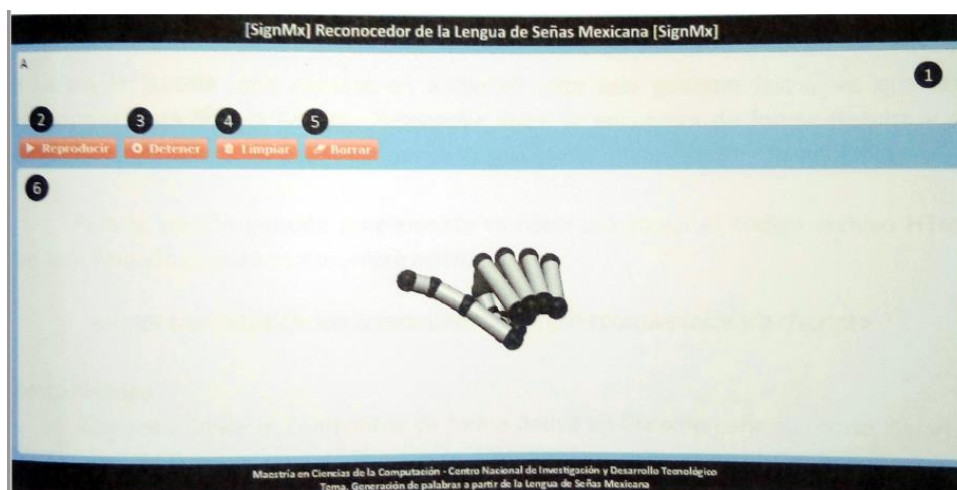


Figura 3: Aplicación SignMx (Romero, 2017).

Esta aplicación consiste en generar palabras a partir de las señas realizadas, haciendo uso de un dispositivo llamado Leap Motion Controller, quien es capaz con sus cámaras y leds infrarrojos junto al Framework LeapTrainer.js capturar y reconocer las señas para así escribirlas y luego reproducir el texto a voz. Respecto a los resultados, es necesario capacitar a las personas que usen la aplicación debido a que las señas deberán de ser realizadas de una manera en específico.

### 1.5.2 Reconocimiento de movimientos de manos para la manipulación de una interfaz computacional

En este trabajo se presentó un método capaz de identificar movimientos de manos basándose en el uso de posiciones definidas, las cuales se establecen a través de la orientación de la palma, los dedos, la distancia entre ellos y su flexión. Para poder ser utilizado en la ejecución de tareas cotidianas a través de movimientos naturales (Molina, 2018). Para detectar cada gesto de la mano es necesario generar el código correspondiente de los gestos (véase Figura 4).

```
var AdelanteA= new HandPose("AdelanteA", new FingerPose(Finger.Thumb, FigerFlexion.Open, PoseDirection.Left), new FingertipDistanceRelation(Finger.Index, RelativeDistance.NotTouching, Finger.Thumb), new FingerPose(new[]{Finger.Index, Finger.Middle, Finger.Ring, Finger.Pinky}, FingerFlexion.Folded));  
  
var AdelanteB = new HandPose("AdelanteB", new FingerPose(Finger.Thumb, FingerFlexion.Open, PoseDirection.Up), new FingertipDistanceRelation(Finger.Index, Relative Distance.NotTouching, Finger.Thumb), new FingerPose(new []{Finger.Index, Finger.Middle, Finger.Ring, Finger.Pinky}, FingerFlexion.Folded));
```

Figura 4: Código para detectar gestos (Molina J., 2018).

El sistema de reconocimiento de gestos sigue el siguiente flujo (véase Figura 5)

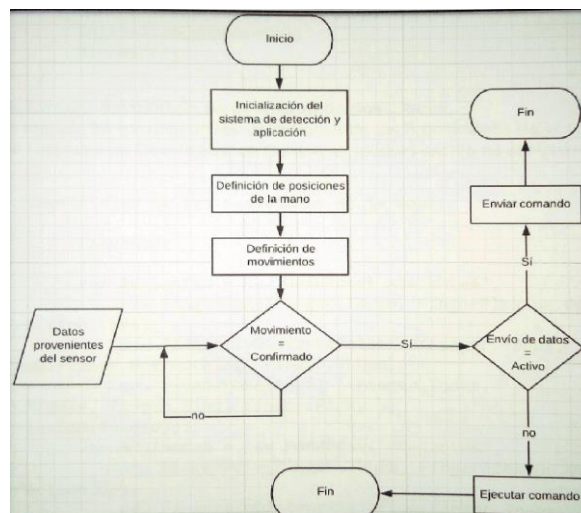


Figura 5: Flujo del sistema de reconocimiento de gestos (Molina J., 2018).

# Capítulo II

## Marco teórico

## 2. Marco teórico

### 2.1. Lenguaje corporal

El lenguaje corporal está relacionado con la comunicación que se transmite mediante un conjunto de señas y gestos, los que sirven para enfatizar y transmitir de manera más vivencial la información que queremos que otro u otros entiendan y comprendan (Marin y Roli, 2019).

### 2.2. Posición de la mano

Una posición de la mano describe una captura momentánea de la mano del usuario, incluidos detalles sobre el estado de la palma y los dedos. A diferencia de un gesto, cuya detección requiere que el usuario ejecute una determinada secuencia de posiciones y movimientos de la mano durante un período de tiempo. La detección de una posición de la mano requiere que el usuario mantenga la postura de la mano durante un breve momento, ésta describe el estado de todas las partes de la mano desde la muñeca hacia arriba (Krupka et al, 2017).

### 2.3. SDK

Un *SDK* (*Software Development Kit*) o kit de desarrollo de software, es un conjunto de herramientas que ayudan a la programación de aplicaciones para un entorno tecnológico particular. Es decir, las aplicaciones desarrolladas sobre el SDK estarán destinadas a algún sistema operativo, plataforma hardware, consola de videojuegos o paquete de software en especial. Son muchos los recursos que puede contener un SDK (Cook, 2013).

Detallamos algunos de ellos:

- Una interfaz de programación de aplicaciones (API). Puede verse como una abstracción del funcionamiento interno del entorno sobre el que vamos a trabajar. Se trata de un conjunto de funciones, rutinas, estructuras de datos, clases y variables que nos permiten manipular el mecanismo de la plataforma sin conocerlo internamente.
- Un entorno de desarrollo integrado (IDE). Un editor que nos ayuda a escribir fácilmente el código fuente del programa. Generalmente brinda una interfaz amigable para cuatro aplicaciones fundamentales:
  - Debugger. Permite testear el programa en cada paso de su ejecución.
  - Compilador. Traduce el código fuente a lenguaje de máquina, obteniendo así un programa ejecutable.
  - Código de ejemplo y otra documentación. Como punto de partida para empezar a desarrollar aplicaciones.
  - Un emulador del entorno. Por ejemplo, si desarrollamos una aplicación para móviles desde una computadora de escritorio, nos permite saber cómo la vería el usuario final.

## 2.4. NUI (Natural User Interface)

Las Interfaces de Usuario Natural (NUI) se constituyen como nuevos métodos para la Interacción Humano Computadora (HCI) y el diseño de aplicaciones informáticas basadas en interfaces con las cuales las interacciones se realizan a partir de las acciones naturales de los seres humanos, tal y como éstos realizan sus actividades en el mundo físico todos los días, sin la necesidad de utilizar periféricos para ingresar los datos, aprovechando de esta forma los conocimientos que sobre este entorno tenemos los seres humanos de manera innata. Para interactuar con sistemas basados en NUI's se han venido utilizando diversas modalidades de entrada, tales como el tacto, reconocimiento de gestos, seguimiento de movimientos, comandos de voz, entre otros (Lozada et al, 2014).

## 2.5. Generación de código

La generación automática de código es el proceso mediante el cual un programa produce, de manera automática, código en un lenguaje, a partir de un esquema expresado en otro lenguaje. Tradicionalmente se usa para traducir esquemas en lenguajes de alto nivel más cercanos a la manera de pensar del humano, hacia lenguajes de más bajo nivel (ensamblador o lenguaje de máquina) orientados a su interpretación por parte de las computadoras. Se denominan generadores de código a las aplicaciones que llevan a cabo dicha tarea. Las aplicaciones de generación de código de uso más extendido son los compiladores (Gregoire y Marc, 2018).

## 2.6. Sensor de profundidad

Para llevar a cabo la detección de las manos, se hace uso de un sensor de profundidad que además es compatible con el SDK de *Project Gesture*. El sensor de profundidad (véase *Figura 6*) *Intel RealSense SR300* es un dispositivo con tecnología de profundidad: luz codificada. Tiene un intervalo de operación (mínimo-máximo) de 0.3m -2m. La distancia máxima alcanza una resolución de VGA 30fps. Se caracteriza por ser un dispositivo pequeño y versátil con dimensiones: 110mm x 12.5mm x 3.75mm (Intel., 2020).



Figura 6: Intel RealSense SR300 (Giang T., 2017).

El uso del sensor de profundidad servirá para detectar las poses de las manos del usuario y así poder generar el código fuente que representará las poses capturadas por el sensor. Se optó por este

sensor ya que de acuerdo a Krupka E. et al., (2017) y las características del sensor, es el mejor candidato para trabajar con el SDK de Project Gesture, teniendo un mejor reconocimiento de las poses de las manos.

## 2.7. Lógica difusa

La lógica difusa proporciona una manera simple y elegante de obtener una conclusión a partir de información de entrada vaga, ambigua, imprecisa, con ruido o incompleta. En general la lógica difusa imita como una persona toma decisiones basada en información con las características mencionadas. Una de las ventajas de la lógica difusa es la posibilidad de implementar sistemas basados en ella tanto en hardware como en software o en combinación de ambos (DNegri C. Luis de Vito E., 2006). La lógica difusa es una técnica de la inteligencia computacional que permite trabajar con información con alto grado de imprecisión, en esto se diferencia de la lógica convencional que trabaja con información bien definida y precisa. Es una lógica multivaluada que permite valores intermedios para poder definir evaluaciones entre sí/no, verdadero/falso, negro/blanco, caliente/frío, pequeño/grande, cerca/lejos, pocos/muchos, entre otros (DNegri C. Luis de Vito E., 2006).

### 2.7.1 Conjuntos difusos

De manera intuitiva se tiene el concepto de conjunto como una colección bien definida de elementos, en la que es posible determinar para un objeto cualquiera, en un universo dado, si acaso este pertenece o no al conjunto. La decisión es "sí pertenece" o bien "no pertenece"

Por otro lado, en un conjunto difuso su frontera no está precisamente definida, y a cada elemento del universo se le asocia un grado de pertenencia, el cual es un valor entre 0 y 1. En tanto al grado de pertenencia sea más cercano a 1 tanto más estará el elemento en el conjunto y en tanto el grado de pertenencia sea más cercano a 0 este elemento menos corresponderá al conjunto.

### 2.7.2 Operaciones de conjuntos difusos

Las operaciones básicas entre conjuntos difusos son las siguientes:

- Unión: Teniendo los conjuntos difusos  $\underline{A}$  y  $\underline{B}$  se define su unión  $\underline{A} \cup \underline{B}$ . Véase la Ecuación 1, 2

$$\underline{A} \cup \underline{B} = \frac{\mu_{\underline{A}}(x_1) \vee \mu_{\underline{B}}(x_1)}{x_1} + \dots + \frac{\mu_{\underline{A}}(x_n) \vee \mu_{\underline{B}}(x_n)}{x_n} \quad (1)$$

O

$$\underline{A} \cup \underline{B} = \max(\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)) \quad (2)$$

- Intersección: Teniendo los conjuntos difusos  $\underline{A}$  y  $\underline{B}$  se define como  $\underline{A} \cap \underline{B}$ . Véase la Ecuación 3, 4.

$$\underline{A} \cap \underline{B} = \frac{\mu_{\underline{A}}(x_1) \wedge \mu_{\underline{B}}(x_1)}{x_1} + \dots + \frac{\mu_{\underline{A}}(x_n) \wedge \mu_{\underline{B}}(x_n)}{x_n} \quad (3)$$

O

$$\underline{A} \cap \underline{B} = \min(\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)) \quad (4)$$

- Complemento: Sea un conjunto difuso  $\underline{A}$ , su complemento se define como  $\underline{\underline{A}}$ . Véase la Ecuación 5.

$$\underline{\underline{A}} = \frac{1 - \mu_{\underline{A}}(x_1)}{x_1} + \dots + \frac{1 - \mu_{\underline{A}}(x_n)}{x_n} \quad (5)$$

### 2.7.3 Controlador difuso

De acuerdo a García, R. et al. (2017) “El principio básico de un modelo basado en lógica difusa es el conjunto de reglas heurísticas, cuyas variables de entrada y salida, ambas lingüísticas, son representadas mediante conjuntos difusos”. Un sistema difuso o controlador difuso (véase Figura 7) basado en reglas, emula el razonamiento de un experto de un área de conocimiento específico, el cual está conformado por cuatro bloques operacionales: Fuzzificación, Conjunto de reglas, Mecanismo de inferencia y Defuzzificación (García, R. et al., 2017).

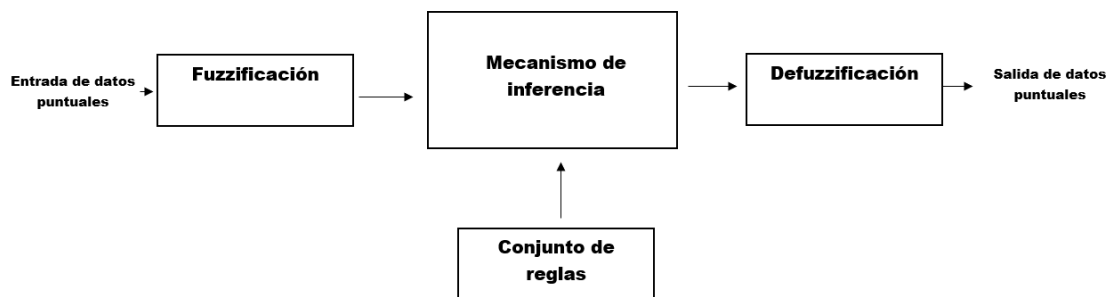


Figura 7: Componentes de un sistema difuso.

#### Fuzzificación

Es un proceso de transformación de valores reales (puntuales, numéricos) en números difusos. Estos valores de pertenencia son representados por un conjunto difuso previamente ya establecidos, dichos conjuntos difusos son representados por alguna de las diferentes funciones de pertenencia ( $\mu_A(x)$ ), por ejemplo:

- Triangular

Está definida por un límite inferior  $a$ , un límite superior  $c$ , y un valor  $b$ . Véase la Ecuación 6.

$$\mu_A(x) = \begin{cases} 0, & \text{si } x \leq a \\ \frac{x-a}{b-a}, & \text{si } a \leq x \leq b \\ \frac{c-x}{c-b}, & \text{si } b \leq x \leq c \\ 0, & \text{si } x \geq c \end{cases} \quad (6)$$

- Trapezoidal

Está definida por los siguientes parámetros  $a, b, c, d$ . Véase la Ecuación 7.

$$\mu_A(x) = \begin{cases} 0, & \text{si } (x < a) \text{ o } (x > d) \\ \frac{x-a}{b-a}, & \text{si } a \leq x \leq b \\ 1, & \text{si } b \leq x \leq c \\ \frac{d-x}{d-c}, & \text{si } c \leq x \leq d \end{cases} \quad (7)$$

- Gaussiana

Está definida por  $\underline{\mu}$  (valor medio) y  $\sigma$  (desviación estándar). Véase la Ecuación 8.

$$\mu_{\underline{A}}(x) = \exp^{-\left(\frac{x-\underline{\mu}}{\sigma}\right)^2} \quad (8)$$

- S

definida por los parámetros: límite inferior  $a$  y límite superior  $b$ . El punto de inflexión:

$m = \frac{a+b}{2}$ . Véase la Ecuación 9.

$$\mu_{\underline{A}}(x) = \begin{cases} 0 & \text{si } x \leq a \\ 2\left(\frac{x-a}{b-a}\right)^2 & \text{si } a < x \leq m \\ 1 - 2\left(\frac{x-b}{b-a}\right) & \text{si } m < x \leq b \\ 1 & \text{si } x \geq b \end{cases} \quad (9)$$



## Conjunto de reglas

De acuerdo a García, R. et al. (2017) el conjunto de reglas “define las reglas lingüísticas del control y la manipulación de la información difusa referente a las funciones de pertenencia de los conjuntos difusos”. Las reglas toman el papel del conocimiento, ya que con ellas se condicionan los datos de entrada para dar un resultado deseado. Véase la Ecuación 10.

$$\begin{array}{c} \text{SI } p \text{ ENTONCES } q \\ \text{o} \\ \text{Si } x \text{ es } A \text{ ENTONCES } y \text{ es } B \end{array} \quad (10)$$

## Mecanismo de inferencia

El mecanismo de inferencia o Sistema de inferencia difuso, se encarga de hacer la combinación/activación tanto de las reglas de entrada como de salida. Para llevar a cabo este procedimiento, existen dos métodos de inferencia difusa:

- Mamdani

Es el método más común y simple, ya que se basa en variables difusas tanto en el antecedente como en el consecuente. Véase la Ecuación 11.

$$\text{IF } x \text{ es } A3 \text{ OR } y \text{ es } B1 \text{ THEN } z \text{ es } C1 \quad (11)$$

Donde el antecedente es  $\{x \text{ A3 OR}, y \text{ es } B1\}$  y el consecuente  $\{z \text{ es } C1\}$ ; los conjuntos de pertenencia son representados por  $\{A3, B1, C1\}$ .

- Takagi-Sugeno

Al solo tener variables difusas en el antecedente y en el consecuente funciones lineales, ya no es necesario realizar una desfuzzificación. Sirve para sistemas más complejos. Véase la Ecuación 12.

$$\text{IF } x \text{ es } A3 \text{ AND } y \text{ es } B1 \text{ THEN } z \text{ es } f(x, y) \quad (12)$$

Donde el antecedente es  $\{x \text{ A3 AND}, y \text{ es } B1\}$  y el consecuente  $\{z \text{ es } f(x,y)\}$ ; los conjuntos de pertenencia son representados por  $\{A3, B1\}$ .

## Defuzzificación

De acuerdo a “la defuzzificación es un proceso matemático usado para convertir un conjunto difuso en un número real. El mecanismo de inferencia difusa obtiene una conclusión a partir de la información de la entrada, pero es en términos difusos es por esto que existen diferentes métodos de defuzzificación y arrojan resultados distintos, el más común y ampliamente usado es el centroide”.

De acuerdo a Lara-Valencia, L. et. al. (2015). “El método del centroide consiste en determinar el centro de gravedad de la distribución obtenida en el eje de las abscisas” véase Figura 8

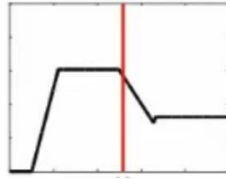


Figura 8: Método centroide (Zamora, G., 2015).

De esta manera se lleva a cabo el cálculo mediante la siguiente ecuación. Véase la Ecuación 13.

$$Z^* = \int \frac{\mu_{\underline{R}}(z)zdz}{\mu_{\underline{R}}(z)dz} \quad (13)$$

Donde  $Z^*$  representa la desfuzzificación por centro de gravedad (valor numérico de la salida) y  $\mu_{\underline{R}}(z)$  la función de pertenencia.

Otros métodos para llevar a cabo la desfuzzificación son los siguientes:

- El Máximo Central (MOM, *middle of maximum*). La salida es el valor medio de los valores cuyas funciones de pertenencia alcanzan el valor máximo (véase Figura 9).

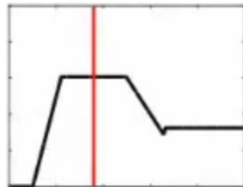


Figura 9: Método máximo central (Zamora, G., 2015).

- El Máximo más pequeño (SOM, *smallest of maximum*). La salida es el mínimo valor de todos aquellos que generan el valor más alto de la función de pertenencia (véase Figura 10).

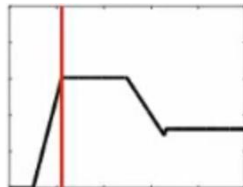


Figura 10: Método máximo más pequeño (Zamora, G., 2015).

- El máximo más grande (LOM, *largest of maximum*). La salida es el máximo valor de todos aquellos que generan el valor más alto de la función de pertenencia (véase Figura 11).

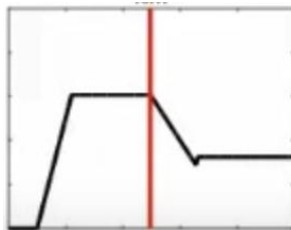


Figura 11: Método máximo más grande (Zamora, G., 2015).

- El bisector de área. La salida es el valor que separa el área bajo la curva en dos subáreas iguales (véase Figura 12).

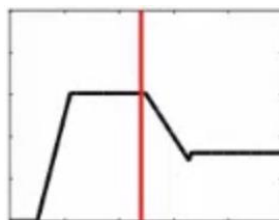


Figura 12: Método bisector de área (Zamora, G., 2015).

## 2.8. Gramática

De acuerdo a la literatura se habla de que el uso de una gramática sirve para poder presentar pensamientos o ideas de una manera clara, de acuerdo a un conjunto de reglas y especificaciones que conlleve dicha gramática que se está abordando (Estela M., 2020). Para esto existen diferentes tipos de gramáticas, las cuales son:

**Gramática prescriptiva o normativa:** Este gramático maneja lo que es un “esquema” o una manera correcta de lo que es el idioma a tratar, para poder guiar al hablante a formular y desarrollar correctamente sus oraciones.

**Gramática descriptiva:** A diferencia de la anterior, no juzga como “correcta” o “incorrecta” la manera en que distintos hablantes hacen uso del idioma, sino que aspira a comprender cómo es el uso real de las normas del idioma dentro de una comunidad o unas comunidades determinadas.

**Gramática tradicional:** Se trata del conjunto histórico de documentos e ideas heredadas de civilizaciones anteriores en torno a lo que la gramática es.

**Gramática funcional:** Aspira a ser una gramática general del lenguaje natural, o sea, un conjunto de normas básicas aplicables a diferentes idiomas dotados de gramáticas distintas.

**Gramáticas formales:** Se llaman así a las gramáticas abstractas, que pueden aplicar su lógica a lenguajes no verbales, como los lenguajes de programación informáticos.

En este punto se puede apreciar que tipo de gramática se abordará, para definir una gramática que ayudará en la generación de código, la cual es la gramática formal. Para esto, se encontró lo que es la

jerarquía de Chomsky, la cual habla que las gramáticas formales se dividen en cuatro tipos, y la diferencia entre cada uno de ellos se basa en el comportamiento que tienen sus respectivas producciones (Gallego A., 2008).

De este modo, ya sea una gramática con la siguiente estructura  $G = (\Sigma_T, \Sigma_N, S, P)$

Donde:

$\Sigma_T$ - Un conjunto de terminales

$\Sigma_N$ - Un conjunto de no terminales

$S$ - Producción inicial

$P$ - Conjunto de producciones

Los tipos de gramáticas formales de acuerdo a la jerarquía de Chomsky (), son los siguientes:

- Tipo 0: Lenguajes recursivos

Conjuntos de objetos formales de cualquier complejidad computacional. Véase la Ecuación 14.

$$\begin{aligned} \mathbf{u} &::= \mathbf{v} \\ \mathbf{u}, \mathbf{v} &\in (\Sigma_T \cup \Sigma_N)^* \end{aligned} \quad (14)$$

- Tipo 1: Lenguajes sensibles al contexto

Conjuntos de conjuntos de secuencias de símbolos (o “cadenas”). Véase la Ecuación 15.

$$\begin{aligned} \mathbf{xAy} &::= \mathbf{xvy} \\ \mathbf{A} &\in \Sigma_N \\ \mathbf{x}, \mathbf{y} &\in (\Sigma_T \cup \Sigma_N)^* \\ \mathbf{v} &\in (\Sigma_T \cup \Sigma_N)^+ \end{aligned} \quad (15)$$

- Tipo 2: Lenguajes libres de contexto

Conjuntos de secuencias de símbolos (o “frases”). Véase la Ecuación 16.

$$\begin{aligned} \mathbf{A} &::= \mathbf{v} \\ \mathbf{v} &\in (\Sigma_T \cup \Sigma_N)^* \\ \mathbf{A} &\in \Sigma_N \end{aligned} \quad (16)$$

- Tipo 3: Lenguajes regulares

Secuencias de símbolos, se dividen en dos grupos:

-Lineales por la izquierda. Véase la Ecuación 17.

$$\begin{aligned} A &::= a \\ A &::= Va \\ S &::= \lambda \end{aligned} \quad (17)$$

Donde  $a \in \Sigma_T$ ,  $A, V, S \in \Sigma_N$ , S es el axioma de la gramática

-Lineales por la derecha. Véase la Ecuación 18.

$$\begin{aligned} A &::= a \\ A &::= aV \\ S &::= \lambda \end{aligned} \quad (18)$$

Donde  $a \in \Sigma_T$ ,  $A, V, S \in \Sigma_N$ , S es el axioma de la gramática

La representación de las producciones de la jerarquía de Chomsky, presentan algunos símbolos que denotan la cantidad de repeticiones que se pueden presentar un símbolo (\*, +), estos símbolos pertenecen a las expresiones regulares. Las expresiones regulares son las unidades de descripción de los lenguajes regulares, que se incluyen en los denominados lenguajes formales (IONOS., 2019).

A continuación, se mostrará el significado de algunos de los caracteres especiales que manejan las expresiones regulares. Véase Tabla 1.

Tabla 1: Caracteres especiales de expresiones regulares.

*	El número del carácter, de la clase o del grupo situado antes del asterisco puede ser aleatorio (cero incluido)
+	El carácter, la clase o el grupo antes de un signo más debe aparecer como mínimo una vez
?	El carácter, la clase o el grupo antes del signo de interrogación es opcional y puede aparecer como máximo una vez.
[]	Los corchetes identifican a una clase de caracteres que siempre representa a un único carácter en un patrón de búsqueda.
()	Los paréntesis identifican a un grupo de caracteres formado por uno o varios caracteres y que pueden operarse unos dentro de los otros
{n}	El carácter, la clase o el grupo anteriores aparecen exactamente n veces.

## 2.9. Distancia euclídea

De acuerdo a EcuRed (2021) se trata de una función no negativa usada en diversos contextos para calcular la distancia entre dos puntos, primero en el plano y luego en el espacio. También sirve para definir la distancia entre dos puntos en otros tipos de espacios de tres o más dimensiones. Y para hallar la longitud de un segmento definido por dos puntos de una recta, del plano o de espacios de mayor dimensión.

En el plano cartesiano (ejes x, y) sean los puntos  $P_1 = (x_{P1}; y_{P1})$   $P_2 = (x_{P2}; y_{P2})$  se define la distancia euclídea mediante la Ecuación 19:

$$d(P_1, P_2) = \sqrt{(x_{P1} - x_{P2})^2 + (y_{P1} - y_{P2})^2} \quad (19)$$

En el espacio (ejes x, y, z) sean los puntos  $P_1 = (x_{P1}; y_{P1}; z_{P1})$   $P_2 = (x_{P2}; y_{P2}; z_{P2})$  se define la distancia euclídea mediante la expresión Ecuación 20:

$$d(P_1, P_2) = \sqrt{(x_{P1} - x_{P2})^2 + (y_{P1} - y_{P2})^2 + (z_{P1} - z_{P2})^2} \quad (20)$$

## 2.10. Matriz de confusión

De acuerdo a Barrios, J. (2019) la matriz de confusión (véase Figura 13) es “Una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado.” Donde dependiendo el autor que sea utilizado como referencia, la matriz contendrá un lado (ya sea filas o columnas) para representar las predicciones de cada clase, mientras que el lado restante representará las instancias de las clases reales. Mostrando el tipo de valores (aciertos y errores) que el modelo utilizado está obteniendo en el proceso de aprendizaje con los datos.



Figura 13: Matriz de confusión (Barrios, J., 2019).

Estas cuatro opciones representan la matriz de confusión y se traducen de la siguiente manera:

- Verdaderos positivos (VP): El valor real es positivo y la prueba predijo también que era positivo.
- Verdaderos negativos (VN): El valor real es negativo y la prueba predijo también que el resultado era negativo.
- Falsos negativos (FN): El valor real es positivo, y la prueba predijo que el resultado es negativo.
- Falsos positivos (FP): El valor real es negativo, y la prueba predijo que el resultado es positivo.

A partir de estas opciones se calculan las métricas: exactitud, precisión y sensibilidad.

### **Exactitud**

De acuerdo a Barrios, J. (2019) “se refiere a lo cerca que está el resultado de una medición del valor verdadero. En términos estadísticos, la exactitud está relacionada con el sesgo de una estimación”. Véase la Ecuación 21.

$$\frac{VP + VN}{VP + FP + FN + VN} \quad (21)$$

### **Precisión**

De acuerdo a Barrios, J. (2019) “Se refiere a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. Cuanto menor es la dispersión mayor la precisión”. Véase la Ecuación 22.

$$\frac{VP}{VP + FP} \quad (22)$$

### **Sensibilidad**

De acuerdo a Barrios, J. (2019) “Es la proporción de casos positivos que fueron correctamente identificadas por el algoritmo”. Véase la Ecuación 23.

$$\frac{VP}{VP + FN} \quad (23)$$

# Capítulo IV

## Estado del arte



### 3. Estado del arte

#### Interfaz de aplicación para el reconocimiento de gestos con el sensor Kinect

En esta investigación se desarrolló una interfaz de aplicación haciendo uso del lenguaje de programación C#, para que reconozca los gestos de los niños mientras bailan, utilizando un dispositivo *Kinect* se pretende que los gestos de los niños sean registrados y reconocidos por la cámara para controlar remotamente un robot Lego y que éste realice las acciones (Dardan, 2016). La comunicación entre el dispositivo Kinect y la aplicación es proporcionada por el Kinect SDK y la biblioteca NUI. Cuando un niño se posiciona frente al dispositivo Kinect a una distancia adecuada, el sensor del dispositivo Kinect rastrea el esqueleto del niño, hecho esto, posteriormente la biblioteca NUI devuelve las posiciones en formato de coordenadas X, Y, Z.

Mediante el uso del software OpenNI se pueden escribir aplicaciones basadas en interacciones naturales gracias a que proporciona interfaces de programación abstracta (API). De esta manera es como se han implementado algoritmos para cada movimiento del proceso de reconocimiento de gestos. El trabajo principal ha sido la implementación de una interfaz fácil de usar y que fuese capaz de reconocer diferentes tipos de gestos realizados por los niños. Utilizando la API definida por OpenNI, el middleware NITE y el dispositivo Kinect se pueden desarrollar aplicaciones que detecten gestos de las personas, definiendo cada gesto por separado. El uso de estas herramientas implica, en ocasiones, la incompatibilidad de las versiones entre ellas, así como también del sistema operativo en el que se ejecuten (Dardan, 2016).

#### Teleoperación de un robot de tubo concéntrico mediante el seguimiento visual de los gestos de la mano

Los robots de tubo concéntrico han sido un estudio popular para procedimientos quirúrgicos mínimamente invasivos, debido a sus diámetros estrechos y curvatura, los que consisten en tubos elásticos precurvados delgados. Los gestos de mano que debe incorporar la interfaz de máquina humana deben ser distintos, intuitivos y reconocibles para el control del robot (Razjigaev, 2017).

En esta investigación se utilizaron los algoritmos para la detección de gestos de la mano con los que cuenta el kit de desarrollo de los estándares de movimiento de Leap Motion, las propiedades que trabajaron fueron: posición de la palma, orientación, radio de la esfera y distancia de pellizco. Este algoritmo de movimiento Leap toma la posición del dedo y la palma para ser tangente a una esfera. Esta medida es bastante estable y es menos propensa al ruido producido por las sacudidas manuales. La orientación de la mano se mide desde la rotación de la palma y se alinea con el siguiente marco de coordenadas (véase Figura 14).

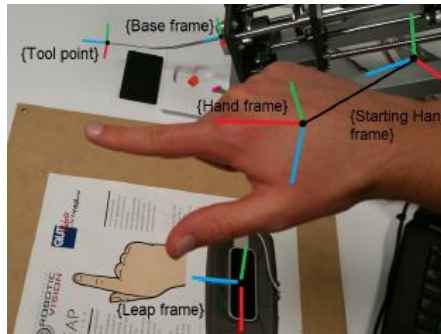


Figura 14: Marcos coordinados del sistema. Los ejes X, Y, Z son rojo, verde y azul respectivamente (Razjigaev, 2017).

El sistema de seguimiento genera datos de traducción del punto de la herramienta (X, Y, Z) y la orientación del balanceo, inclinación y orientación de la mano. Estas mediciones de los seis grados de libertad se utilizan luego para el algoritmo de mapeo que toma estos datos y calcula la cinemática inversa necesaria del robot de tubo concéntrico (Razjigaev, 2017).

## Reconocimiento de gestos de la mano en tiempo real usando diferentes algoritmos basándose en el lenguaje de señas americanas

Un sistema HGR (Reconocimiento de Gestos de Manos) en tiempo real basado en el reconocimiento del lenguaje de señas americano (ASL, por sus siglas en inglés) debe adquirir primeramente imágenes de ASL con fondo negro desde una cámara de video para poder realizar la extracción de cinco de sus características, buscador de punta de dedo, excentricidad, alargamiento, segmentación de píxeles y rotación. Extrayendo un total de 30 vectores de características para cada imagen (Mohiminul, 2017).

La Red Neuronal Artificial (ANN, por sus siglas en inglés) entrenada con un algoritmo de propagación hacia atrás, se utiliza para el reconocimiento en tiempo real de un gesto. Cuando se implementa en una interfaz gráfica se pueden realizar las pruebas realizadas por los usuarios (véase Figura 15).



Figura 15: Interfaz gráfica para el reconocimiento de gestos usando redes neuronales (Mohiminul, 2017).

Este sistema utiliza cinco algoritmos de extracción de características para poder obtener un reconocimiento robusto de gestos de mano. Son cuatro pasos los que lo lleva al reconocimiento de gestos, tales pasos son: la adquisición de imágenes, pre-procesamiento, extracción de características y reconocimiento de características. La ANN está entrenada con 1850 imágenes de muestra de una

base de datos y reconoce los alfabetos y números de ASL con casi 94.32% de precisión en el entorno de tiempo real.

## Reconocimiento de gestos alfanuméricos del lenguaje de señas mexicanas usando características 3d haar-like

“El Lenguaje de Señas Mexicano (LSM) es un lenguaje de la comunidad sorda mexicana, que consiste en una serie de signos gestuales articulados con las manos y acompañados de expresiones faciales” (Jimenez, 2017). El objetivo de esta investigación fue proponer un sistema para el reconocimiento de gestos de la mano correspondientes a la Lengua de Señas Mexicana (LSM). El sistema consta de un dispositivo de visualización (monitor externo), una cámara de profundidad (sensor Kinect) y una estación de trabajo PC (véase Figura 16).

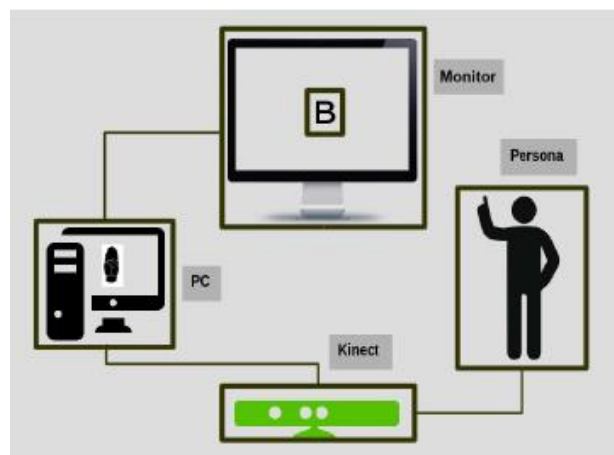


Figura 16: Arquitectura del sistema (Jimenez, 2017).

Este sistema utiliza una base de datos para ser entrenado y probado. Consiste en un conjunto de imágenes generadas a partir de diez señas alfanuméricas diferentes realizadas por 100 personas (hombres y mujeres de complejiones físicas variadas), dando un total de 100 muestras por seña. Además, se hizo un pre-procesamiento de imágenes y se recurrió a la extracción de características haciendo uso de las características tipo Haar, las que consideran regiones rectangulares adyacentes en un lugar específico dentro de una ventana de detección de una imagen, sumando las intensidades de los píxeles en estas regiones y calculando la diferencia entre ellas (Jimenez, 2017). Obteniendo un 95% de eficiencia; para lograr llegar a ello se necesita gran cantidad de muestras y tiempo de procesamiento, puede ser eficiente hasta ese porcentaje, pero la elaboración de este conlleva a la inversión de mucho tiempo (Jimenez, 2017).

## Aprendizaje automático para el reconocimiento de gestos de la mano usando bolsa de palabras

El objetivo de esta investigación fue el desarrollo de un método de aprendizaje para el reconocimiento en tiempo real de 16 gestos de manos utilizando el sensor Kinect (Benmoussa, 2018). Para su desarrollo se hizo uso de los descriptores *Speeded Up Robust Features* (SURF) y *Scale Invariant Features Transform* (SIFT) entrenados con los clasificadores K-means y Support Vector Machine (SVM). Este método consiste en el entrenamiento de un modelo de SVM en datos de

profundidad de mano a partir de los cuales se extrajeron las palabras de los descriptores SIFT y SURF. Se utilizó el sensor Kinect para obtener 8000 imágenes divididas en 16 grupos diferentes que corresponden al número de gestos, donde a cada gesto le pertenecen 500 imágenes. El método logró un 98% de rendimiento para SURF y un 91% para SIFT calculado utilizando el área bajo la medida de la curva ROC (Benmoussa, 2018).

## Enfoque basado en reglas para reconocer gestos y poses del cuerpo humano en tiempo real

El objetivo de este estudio fue la propuesta de un clasificador que fuese capaz de reconocer posiciones del cuerpo y gestos corporales en tiempo real, el nombre del método es lenguaje de descripción de gestos (GDL) (Hachaj,2018).

Se utilizó el sensor de Microsoft Kinect como dispositivo para capturar y rastrear los movimientos del cuerpo humano. La propuesta de una forma intuitiva de escribir los movimientos como un conjunto de reglas de una manera formal, tales como scripts de lenguaje computadora y la generación de un módulo de razonamiento capaz de interpretarlos a la misma velocidad que llegan al sensor multimedia. El siguiente esquema muestra el enfoque GDL (véase Figura 17).

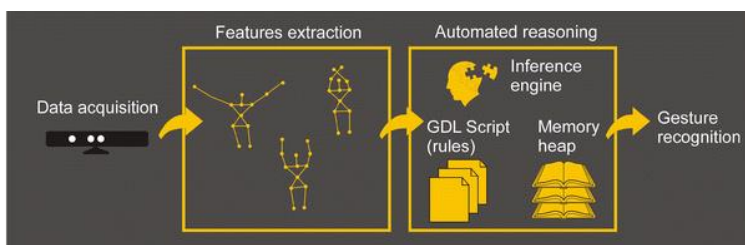


Figura 17: Esquema del enfoque GDL (Tomasz Hachaj, 2014).

Se implementó y probó en un conjunto de 1600 grabaciones de usuarios y los resultados fueron de una tasa de reconocimiento variado del 80.5% al 98.5%. Los errores fueron a causa de la inexactitud del algoritmo de seguimiento y segmentación del usuario y por la baja tasa de cuadros de grabación.

## Generación automática de código fuente procesando imágenes basados en modelos uml en la plataforma hadoop

En este trabajo se propuso un método para crear un modelo utilizando una herramienta de modelado para el desarrollo independiente de la plataforma y generar automáticamente el código fuente basándose en dicho modelo. Este se divide en tres partes: la Guía de diseño HIPI que guía ejemplos para aplicar HIPI MapReduce, el modelo generado por la herramienta de modelado y el generador de código fuente. La guía propuesta se visualiza como un Diagrama de clase y consiste en atributos y métodos básicos para HIPI Map-Reduce. Primero crea un modelo basado en la guía presentada. En función de la información del modelo creado se genera un documento XML y la información XML generada se analiza y se transmite al generador de código fuente, éste último genera el código fuente basado en el resultado analizado (Mi-Eun,2017).

La generación automática de código fuente basada en el modelo reduce el tiempo de desarrollo y reduce los errores debido a la codificación manual. En este documento, definimos un modelo con puntos de vista estructurales y de comportamiento a través de UML y presentamos una guía de diseño para el modelado (Mi-Eun,2017).

## Generación de código en dispositivos móviles para aplicaciones móviles

El objetivo de esta investigación es el desarrollo de una aplicación móvil para la gestión de datos completa que pueda ser ejecutada en dispositivos Android, así como la generación de todo el código para la aplicación nativa, las páginas PHP de soporte para una aplicación web y los scripts SQL (Amanquah,2017).

Se hace uso de una aplicación móvil multiplataforma para recopilar metadatos para la construcción de la aplicación móvil, los metadatos son almacenados en una base de datos para que posteriormente sean leídos por un generador de código y se proceda a generar una aplicación Android utilizable (véase Figura 18).

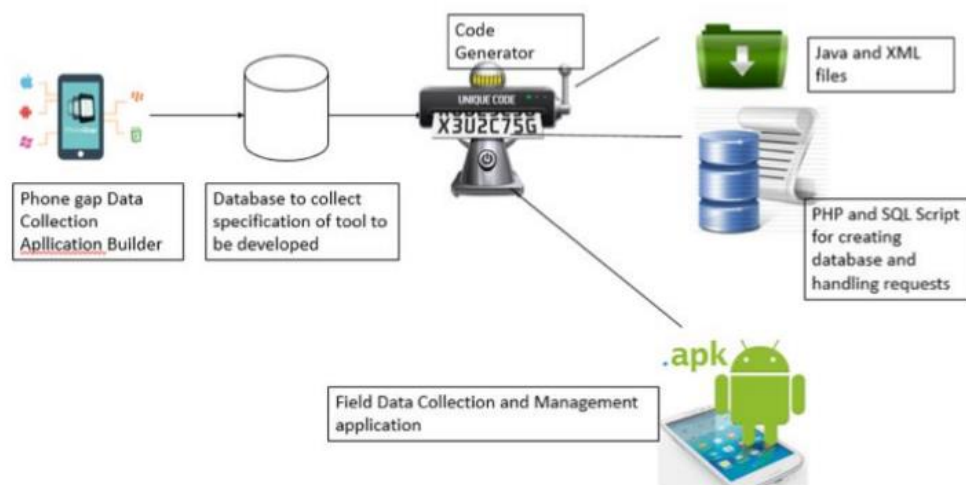


Figura 18: Partes del sistema generador de código (Amanquah, 2017).

El módulo generador de código crea código que los SDK de desarrollo de Android pueden ejecutar. El código incluye una serie de diferentes tipos de archivos. Los archivos XML, los responsables de la gestión de diseños y recursos en Android. Los archivos Java determinan el comportamiento de la interfaz de usuario. Los scripts SQL para la base de datos de la aplicación (Amanquah,2017).

El arrastrar y soltar los componentes no está habilitado en esta aplicación para la especificación de widgets, esto debido al tamaño de la pantalla. El tener una alternativa que tenga habilitado arrastrar y soltar podría ser muy útil, para que pueda ser utilizado en tabletas de pantalla más grande (Amanquah,2017). En conclusión, sería deseable extender el generador para producir código fuente y también crear aplicaciones nativas para otras plataformas, incluidas las plataformas móviles iOS y Windows.

## Aplicación web para un generador de código automático usando una estructura de diagrama de flujo

En este estudio se proporciona una herramienta capaz de generar automáticamente código utilizando un diagrama de flujo estructurado a través de un navegador web. Tomando en cuenta que uno de los principales problemas con los que se encuentra un estudiante que estudia un curso de programación es la comprensión de la sintaxis y la lógica de los lenguajes de programación (Supaartagorn,2017).

Esta herramienta es capaz de convertir el diagrama de flujo en código fuente correctamente escrito en los lenguajes de programación Java y PHP, de igual manera permite corregir errores que llegasen a generarse. Esta herramienta fue desarrollada utilizando el lenguaje JavaScript, la biblioteca Gojs y la biblioteca CodeMirror, y con el marco ExtJs se creó una interfaz de usuarios interactiva. La evaluación de la herramienta se llevó a cabo por dos grupos de participantes, el primero de ellos con 5 expertos en el campo de la programación y 93 estudiantes en el segundo grupo, mostrando un alto índice de satisfacción por parte de los dos grupos.

## Reloj mágico: interactuando y controlando

En este estudio se diseñó el uso de un reloj de muñeca, el cual contiene un conjunto de sensores embebidos capaces de detectar los gestos de la mano e incluso de la muñeca. El reloj puede actuar como un apuntador, un control remoto y un portal de información (Feng et al, 2014). El sistema tiene una construcción de 3 ejes: Acelerómetro, Giroscopio, Magnetómetro y GPS (véase Figura 19).

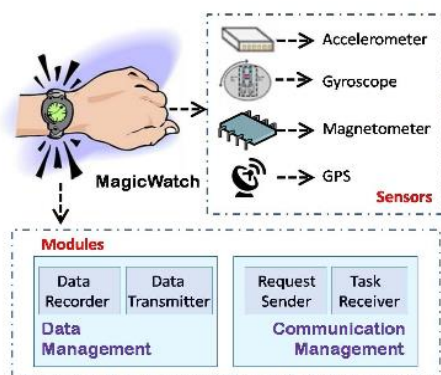


Figura 19: Vista del sistema (Feng et al, 2014)

Para reconocer un gesto el MagicWacht usa una base en aceleración como se muestra en la Figura 20 el reconocimiento de gesto se enfoca en algo llamado FDSVM (Frame-based Descriptor and multi-class SVM). Primero la información referente al acelerómetro es recolectada y representada en un descriptor basado en un Frame, el que permite extraer información discriminativa.

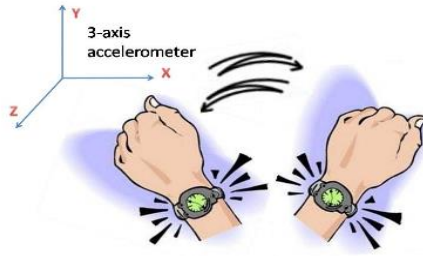


Figura 20: Reconocimiento de gestos (Feng et al, 2014).

En segundo lugar, un clasificador multiclase basado en SVM es construido para el reconocimiento de gestos en el espacio de características de gesto no lineal. Los resultados de reconocimiento alcanzaron más del 95% de exactitud en los experimentos.

## Oscilación de dedo: técnicas sin embrague para la escalación, rotación y traslación de objetos 3d

En este trabajo se utilizó un Leap Motion para localizar el movimiento de los dedos de la mano, desarrollando una aplicación en un ambiente de Unity3D haciendo uso del lenguaje de programación C# (Siju et al, 2014).

Se presentan tres técnicas para la manipulación de objetos en un ambiente 3D con solo el movimiento de dedos.

Las técnicas que se usan son las siguientes:

- FingerShake: para trasladar objetos.
- FingerRotate: para la rotación de objetos.
- FingerSwing: para escalar objetos.

Estas técnicas están inspiradas por CycloStar.

FingerOscillation es un enfoque general de las técnicas basadas en movimientos directamente de los dedos y en periodos circulares, por un modelo oscilatorio elíptico (Siju et al, 2014).

Unas de las propiedades importantes de FingerOscillation son:

- Los gestos oscilatorios permiten mayor control de continuidad de variables para ser manipulados que otros gestos como el arrastrar o aplastar. La acción de recoger puede ser evitada cuando se repite el mismo gesto.
- El usuario no necesita cambiar de postura de mano para indicar el comienzo y el final de una manipulación explícita.

## Gestoremoto: interacción remota con monitores mediante gestos táctiles

En este trabajo se presenta el uso de los gestos que un usuario puede generar mediante un dispositivo con pantalla táctil, como lo son los Smartphone de la actualidad.

*Gestoremoto* hace uso de los gestos que la mano puede generar al tocar una superficie táctil, además de soportar un rango extenso de interacciones con una pantalla remota (Hao y li,2014).

Utiliza gestos simbólicos, que se pueden apreciar cuando se toca la superficie y se arrastre el dedo, para que mediante un toque ejecute una acción determinada bajo el cursor. La forma de la trayectoria del gesto ejecutará determinada función a invocar. Estas funciones están definidas globalmente o se determinan en el contexto actual de la pantalla remota (véase Figura 21).

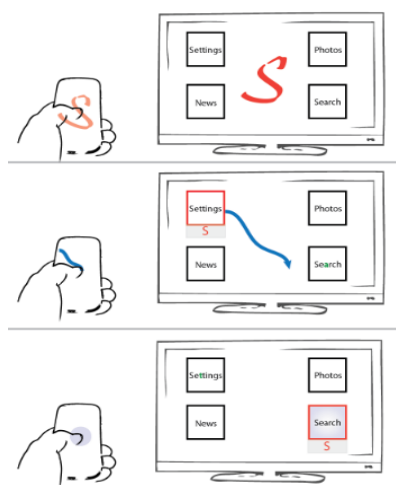


Figura 21: Gestos simbólicos (Hao y li,2014).

## Seguimiento emparejado: Una técnica de visión por computadora para la entrada del usuario mediante el seguimiento de controles animados

En este trabajo se presenta una técnica diseñada para que el usuario disponga de una forma simple de seleccionar funciones con el mínimo esfuerzo, con el uso de una cámara web estándar como dispositivo de entrada, sin especificar alguna parte del cuerpo para ser usada (véase Figura 22).

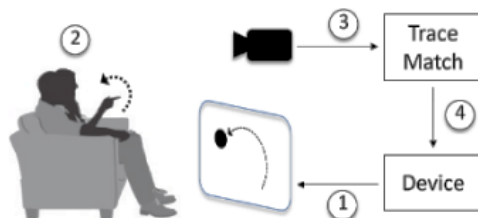


Figura 22: TraceMatch: Sistema de reconocimiento de gestos (Clarke et al,2016).



TraceMatch es una técnica genérica que permite su despliegue en un contexto extenso, con el mínimo esfuerzo, como lo es un control de un televisor inteligente.

No asume una postura específica del usuario, sino que se captura la escena entera del video para analizarla en busca de ocurrencias de algún movimiento que coincida como la visualización de un control (Clarke et al,2016).

Analiza la escena en dos partes:

- La primera parte considera algún movimiento en la escena como una entrada potencial.
- La segunda parte verifica la coincidencia de los movimientos observados contra la visualización de un control, haciendo uso de una combinación de correlación de trayectoria y un ajuste de modelo.

Se hace uso de un movimiento circular, como un movimiento que un usuario pueda realizar con facilidad con diferentes partes del cuerpo, para que así no se produzcan movimientos accidentalmente (véase Figura 23).



Figura 23: Procedimiento de detección (Clarke et al,2016).

Para detectar las características se utilizan los algoritmos de los clasificadores FAST. Se detectan los posibles candidatos y la trayectoria del movimiento de la persona mediante el algoritmo RANSA

## **Táctil y pantalla: colección de artilugios para el control de largas pantallas mediante teléfonos inteligentes**

Este artículo presenta una serie de técnicas para llevar a cabo el control remoto de aplicaciones de pantallas de grandes dimensiones mediante el uso de un Smartphone (Bellino et al, 2016).

El uso de un Smartphone permite la aplicación de los gestos táctiles gracias a la pantalla táctil, principalmente a que en la actualidad es una de las cosas que los caracteriza. Algunas de las técnicas presentadas en este trabajo son novedosas y otras siguen el funcionamiento “estandarizado” de trabajos ya existentes (Bellino et al, 2016).

El diseño de Touch&Screen se basa en asociar ciertas áreas del Smartphone con pequeñas aplicaciones sobre la pantalla grande (véase Figura 24).

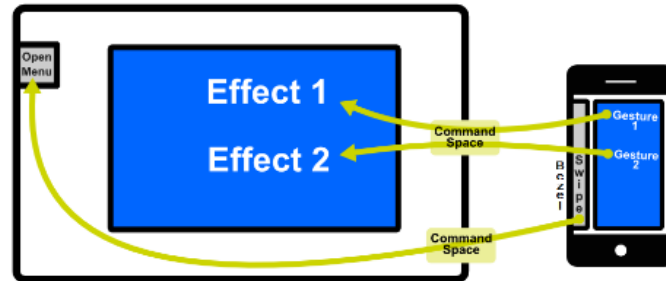


Figura 24: Referencia de áreas entre dispositivos (Bellino et al, 2016).

Para facilitar el uso de las técnicas de gestos se aplicaron animaciones de transición mientras se interactúa sobre la pantalla grande, para evitar que el usuario tenga la necesidad de estar observando el dispositivo remoto al querer ejecutar una acción.

## Entorno de trabajo en la generación de código para la aplicación de una interfaz de programa para un modelo

El *Framework* que se presenta en este trabajo genera código para un modelo de interfaz de programación para aplicaciones (API). Este abarca tres componentes: Un generador de código API, un generador de código de serialización y un generador de código de deserialización (Ivanov et al, 2016).

El *Framework* recibe un modelo como entrada, en algunos casos el modelo es creado de una interfaz JAVA, esquema XML o UML, el cual en algunas realizaciones los modelos son definidos en formato Ecore. Después el generador de código API lo analiza para crear una API, el que es usado para crear una primera versión de instancia del modelo en un primer lenguaje que comprende JavaScript o XSJX. Después el generador de código serializable analiza el modelo para crear una segunda versión de instancia del modelo, de la primera versión instanciada del modelo, en un segundo lenguaje XMI. En seguida el generador de código deserializable analiza el modelo y genera un código deserializado usado para convertir la versión instanciada del modelo que tiene el segundo lenguaje pasándolo al primer lenguaje (Ivanov et al, 2016).

## Generación automatizada de código de interfaz gráfica de usuario multiplataforma para aplicaciones móviles

El entorno de trabajo contiene tres fases: identificación de componentes, conversión del tipo de componente, y la generación de código de la interfaz gráfica de usuario (Lingling et al, 2019).

En la primera fase, se extraen los componentes de las páginas de interfaz de usuario mediante el uso de técnicas para el procesamiento de imágenes, con el propósito de identificar el tipo de componentes (botones, cajas de texto, entre otros), aprovechando los algoritmos de aprendizaje profundo (redes neuronales convolucionales). En la segunda fase, se realiza la conversión de componentes, de acuerdo al tipo de componente que corresponde en cada plataforma objetivo. En la tercera fase, se genera el código de la interfaz gráfica de usuario, basándose en el tipo de

componente junto a sus atributos de alguna de las plataformas (Android o iOS) (Lingling et al, 2019).

Se realizó la comparación de la red neuronal convolucional contra las líneas básicas (regresión logística, máquina de soporte vectorial y k-vecinos más cercanos), logrando en la primera un 85% de exactitud, y en las líneas básicas de un 20%-70% de exactitud (Lingling et al, 2019).

## **Generación de código html automático de imágenes de modelos usando técnicas de aprendizaje profundo**

Modelos dibujados a mano son procesados usando técnicas de visión por computadora y después algunos métodos de aprendizaje profundo son utilizados para implementar el sistema propuesto.

Primero, la detección de objetos fue aplicada como imagen de entrada con las técnicas de procesamiento de imagen como erosión, dilatación y detección de contornos. En el segundo paso la identificación de objetos fue recortada y entonces los objetos obtenidos fueron etiquetados con el modelo entrenado (red neuronal convolucional). Finalmente, la salida de este modelo se ha convertido en código HTML a través del *script* constructor HTML. El sistema logró un 96% de exactitud del método y un 73% de exactitud en la validación (Ensari et al, 2019).

## **Reconocimiento de gestos del dedo usando un generador de línea láser y una cámara**

El trabajo pretende el desarrollo de un sistema que pueda reconocer gestos, los cuales pueden ser usados como entradas de comandos para la interacción con la computadora y puedan ser aplicados a ciertas aplicaciones en el sistema (Deshpande et al, 2019).

El primer paso consiste en capturar imágenes cuadro por cuadro usando una cámara web. El segundo paso consiste en el procesamiento de imágenes, donde involucra el filtrado de color, suavizado y umbralización. En la detección de gestos se usa un método el cual extrae las características de la imagen. Una línea láser es usada para saber qué gesto de dedo fue realizado y las irregularidades en la línea del láser son extraídas. El reconocimiento del gesto se lleva a cabo con ayuda de la extracción de características. Después del reconocimiento, se ejecuta la operación la cual se le fue asignada con el gesto (Deshpande et al, 2019).

## **Generación de código automático a partir de diagramas de gráficos de estado uml**

En este trabajo se presenta un novedoso diseño de patrones para la implementación de diagramas de estados que incluyen jerarquía, concurrencia y estados históricos. Los patrones que se proponen se guardan en una librería de patrones. La librería es usada durante la generación de código. El proceso de generación de código incluye el modelado del sistema en diagramas de estados *UML*, la generación del *XML* para el modelo, analizar el *XML* y luego generar el código (Sunitha y Philip, 2019). En comparación con otras herramientas el método mostrado presenta mejor eficiencia en términos de tiempo necesario para el procesamiento de eventos. Además, el enfoque que se propone, da un código menos complejo y resultados prometedores (Sunitha y Philip, 2019).

---

## **Reconocimiento de gestos de la mano en posición libre usando una red neuronal basada en el detector multicaja de una sola toma**

En este trabajo se emplea una cámara monocular para construir un sistema el cual puede detectar la mano en cualquier posición de la pantalla y reconocer el gesto de la mano. El detector de multicaja de una sola captura localiza la posición de la mano y una red neuronal convolucional realiza la clasificación de los gestos de la mano (Tang et al, 2019).

El detector multicaja de una sola captura es una red neuronal profunda que tiene como propósito el detectar objetos en las imágenes, cuando se detecta la mano se construye una caja con sus diferentes radios y escalados, para poder ser ajustadas y realizar un correcto emparejamiento. Después continúa la segmentación de la mano para aplicar un procesamiento de la imagen. Finalmente, con la red neuronal convolucional se clasifica el gesto detectado, logrando una exactitud del 95% (Tang et al, 2019).

## **Hacia una interfaz realista de gestos con las manos: manteniéndolo simple para desarrolladores y máquinas**

Presenta un lenguaje simple para la descripción de poses y gestos, un conjunto de herramientas de desarrollo para usarlo y una tubería algorítmica que lo reconoce con alta precisión. El lenguaje se basa en un pequeño conjunto de proposiciones básicas obtenidas por la aplicación de cuatro tipos de predicados, enfocados a los dedos y al centro de la palma: dirección, ubicación relativa, tacto de los dedos y estado de plegado de los dedos. El idioma se reconoce desde la entrada de una cámara 3D con un algoritmo de tubería compuesta por múltiples etapas de clasificación/regresión, entrenado en un gran conjunto de datos anotados. Los resultados experimentales indican que la tubería permite el reconocimiento exitoso de gestos con una carga computacional muy baja, lo que ofrece una interfaz basada en gestos en procesadores de gama baja (Krupka , E. et al , 2017).

## **Reconocimiento escrito a mano basado en el reconocimiento de gestos de la mano mediante autómatas finitos deterministas y lógica difusa**

En este documento se propone un nuevo enfoque para el reconocimiento escrito a mano en el contexto del reconocimiento de gestos con las manos. El enfoque se basa en los movimientos de la mano por medio de su información de bordes y las direcciones de los movimientos. Se minimiza la incertidumbre del reconocimiento escrito a mano utilizando lógica difusa y autómatas finitos deterministas (DFA). El resultado experimental en actividades del mundo real muestra el éxito y la utilidad del enfoque propuesto (Zare M., Jampour M., Arezoomand A. y Sabouri M., 2019).

## **Progreso de la inserción de palabras en la generación de código**

A través del análisis del modelo de incrustación de palabras este documento describe su papel vital en la generación de código. Elabora el proceso de usar la incrustación de palabras y la red neuronal

---

para generar automáticamente código de programación, y finalmente espera su tendencia de desarrollo (Li, Z., et al, 2019).

## **Generación automática de api rest a partir de api java, basada en transformación de modelos (mdd).**

La línea de investigación propone un mecanismo de generación de API REST a partir de versiones existentes de API Java en el contexto del desarrollo dirigido por modelos (Model-Driven Development, MDD), para la construcción de WSs. Aplicar esta técnica mediante la transformación de modelos se diferencia de otras formas convencionales, las cuales se basan en generar un AST (Abstract Syntax Tree) mediante algún parser de JAVA. Además, su propuesta permitirá generar código hacia distintas implementaciones de WS REST a partir de un modelo JAVA (Arsaute, A. et al, 2018).

## **Líneas de productos software: generando código a partir de modelos y patrones**

Este artículo es el resultado de una propuesta de diseño de un prototipo para generación de código automatizado a partir de modelos MDA y la implementación de Patrones de diseño como MVC, presentes en la mayoría de los generadores de código de tipo comercial y otras herramientas GNU, las cuales transforman códigos para sistemas Transaccionales CRUD, para entornos Web, en plataformas como JSP, ASP, PHP, Ruby, etc. El uso del prototipo se planteó a partir de un lenguaje común, pero su implementación puede extenderse a otros lenguajes o especificaciones debido a su alto grado de Usabilidad y Fiabilidad (Gitan C., 2017).

### **Comentarios**

Después de la investigación y análisis de los trabajos relacionados al trabajo que se desarrolla en esta tesis se llevaron a cabo la realización de tablas comparativas (véase Tabla 2, Tabla 3, Tabla 4) donde se plasman características que se buscan, haciendo énfasis en dos temas generales: la detección de gestos de la mano y la generación de código. La búsqueda de estas características en trabajos anteriores, es con el fin de asegurar que el trabajo planteado en esta tesis, no fue ya realizado, además de buscar técnicas o métodos para su aplicación en esta tesis.

De acuerdo a las tablas comparativas que se muestran posteriormente, los trabajos encontrados solo se inclinan por un tema, ya sea los gestos de la mano o la generación de código, mas no en los dos, por esta razón se llega a la conclusión que el objetivo general que se plantea en esta tesis no ha sido abordado aun, además de haber proporcionar ideas, técnicas y/o herramientas para haber desarrollado esta tesis.

Tabla 2: Comparación de artículos 1-1.

Artículo	Detección de gestos de la mano		Generación de código					
	Requiere Entrenamiento del sistema	Utiliza Lenguaje de definición de gestos	Cuenta con Interfaz gráfica	Utiliza metadatos	Método o técnica	Utiliza diagramas de flujo	Lenguaje de programación soportado	Genera código
<i>Interfaz de aplicación para el reconocimiento de gestos con el sensor Kinect (Dardan,2016)</i>	SI	NO	SI	NO	API definido por OpenNI	NO	C#	NO GENERA
<i>Teleoperación de un robot de tubo concéntrico mediante el seguimiento visual de los gestos de la mano (Razjigaey, 2017)</i>	NO	SI	NO	NO	Kit de desarrollo de leap motion	NO	No se muestra	NO GENERA
<i>Reconocimiento de gestos de la mano en tiempo real usando diferentes algoritmos basándose en el lenguaje de señas americanas (Islam,2017)</i>	SI	NO	SI	NO	Red neuronal artificial	NO	No se muestra	NO GENERA
<i>Reconocimiento de gestos alfanuméricos del lenguaje de señas mexicanas usando características 3D Haar-like (Jimenez, 2017)</i>	SI	NO	SI	NO	Extracción de características no especificadas	NO	No se muestra	NO GENERA
<i>Aprendizaje automático para el reconocimiento de gestos de la mano usando bolsa de palabras (Benmoussa, 2018)</i>	SI	NO	NO	NO	K-means y máquina de soporte vectorial	NO	No se muestra	NO GENERA
<i>Enfoque basado en reglas para reconocer gestos y poses del cuerpo humano en tiempo real (Hachaj,2014)</i>	SI	SI	SI	NO	Herramientas de Kinect	NO	GDL script	NO GENERA

<i>Generación automática de código fuente procesando imágenes basados en modelos UML en la plataforma Hadoop (Mi-Eun,2017)</i>	NO	NO	NO	SI	Diagrama UML	SI	C++	SI GENERA
<i>Generación de código en dispositivos móviles para aplicaciones móviles (Amanqua,2017)</i>	NO	NO	SI	SI	Diagrama UML	NO	JAVA	SI GENERA

Tabla 3: Comparación de artículos 1-2.

Artículos	Detección de gestos de la mano			Generación de código						
	Requiere Entrenamiento del sistema	Se pueden agregar nuevas señas al sistema	Utiliza Lenguaje de definición de gestos	Cuenta con Interfaz gráfica	Utiliza metadatos	Método o técnica	Utiliza diagramas de flujo	Lenguaje soportado de programación	Genera código	Permite depurar el código generado
<i>Aplicación web para un generador de código automático usando una estructura de diagrama de flujo (Supaartagorn,2017)</i>	NO	NO	NO	NO	NO	Diagrama de flujo	SI	JAVA y PHP	SI	SI
<i>Reloj Mágico: interactuando y controlando (Fenf et al, 2014)</i>	NO	NO	SI	SI	NO	Sensores de movimiento	NO	No se muestra	NO GENERA	NO
<i>Oscilación de dedo: técnicas sin embrague para la escalación, rotación y traslación de objetos 3D (Siju et al, 2014)</i>	NO	NO	SI	SI	NO	Herramientas de Leap Motion	NO	C#	NO GENERA	NO

<i>Gestoremoto: interacción remota con monitores mediante gestos táctiles (Hao y Li ,2014)</i>	SI	NO	SI	SI	NO	Herramientas de pantalla táctil	NO	No se muestra	NO GENERA	NO
<i>Seguimiento emparejado: Una técnica de visión por computadora para la entrada del usuario mediante el seguimiento de controles animados. (Clarke et al,2016)</i>	NO	SI	SI	SI	NO	Clasificadores FAST y RANSAC	NO	No se muestra	NO GENERA	NO
<i>Táctil y pantalla: colección de artilugios para el control de largas pantallas mediante teléfonos inteligentes (Bellino et al,2016).</i>	SI	NO	SI	SI	NO	Herramientas de pantalla táctil	NO	No se muestra	NO GENERA	NO
<i>Entorno de trabajo en la generación de código para la aplicación de una interfaz de programa para un modelo (Ivanov et al, 2016).</i>	NO	NO	NO	NO	NO	Diagrama UML	SI	JavaScript, XSJS, XMI	SI	SI
<i>Reconocimiento de gestos de la mano en posición libre usando una red neuronal basada en el detector multicaja de una sola toma (Tang et al,2019).</i>	SI	NO	NO	No se menciona	NO	Red neuronal convolucional	NO	No se menciona	NO GENERA	NO

Tabla 4: Comparación de artículos 1-3.



Artículos	Detección de gestos de la mano		Generación de código					
	Requiere Entrenamiento del sistema	Utiliza Lenguaje de definición de gestos	Cuenta con Interfaz gráfica	Utiliza metadatos	Método o técnica	Utiliza diagramas de flujo	Lenguaje soportado de programación	Genera código
<i>Reconocimiento de gestos del dedo usando un generador de línea láser y una cámara (Deshpande et al,2019)</i>	SI	SI	NO	NO	Extracción de características	SI	No se muestra	NO GENERA
<i>Generación de código automático a partir de diagramas de gráficos de estado UML (Sunitha &amp; Philip,2019)</i>	NO	NO	No especifica	NO	Máquina de estados finitos	NO	No especifica	SI GENERA
<i>Generación de código HTML automático de imágenes de modelos usando técnicas de aprendizaje profundo (Ensari et al,2019)</i>	SI	NO	SI	SI	Red neuronal convolucional	SI	HTML (etiquetado)	SI GENERA
<i>Generación automatizada de código de interfaz gráfica de usuario multiplataforma para aplicaciones móviles (Lingling et al, 2019).</i>	SI	NO	SI	SI	Red neuronal convolucional	SI	Android y iOS	SI GENERA

# Capítulo IV

## Metodología de solución

## 4. Metodología de solución

En este capítulo se llevará a cabo el desarrollo de la metodología utilizada para dar solución al problema presentado en esta tesis (véase Figura 25), se describe la Fase 1 y Fase 2.

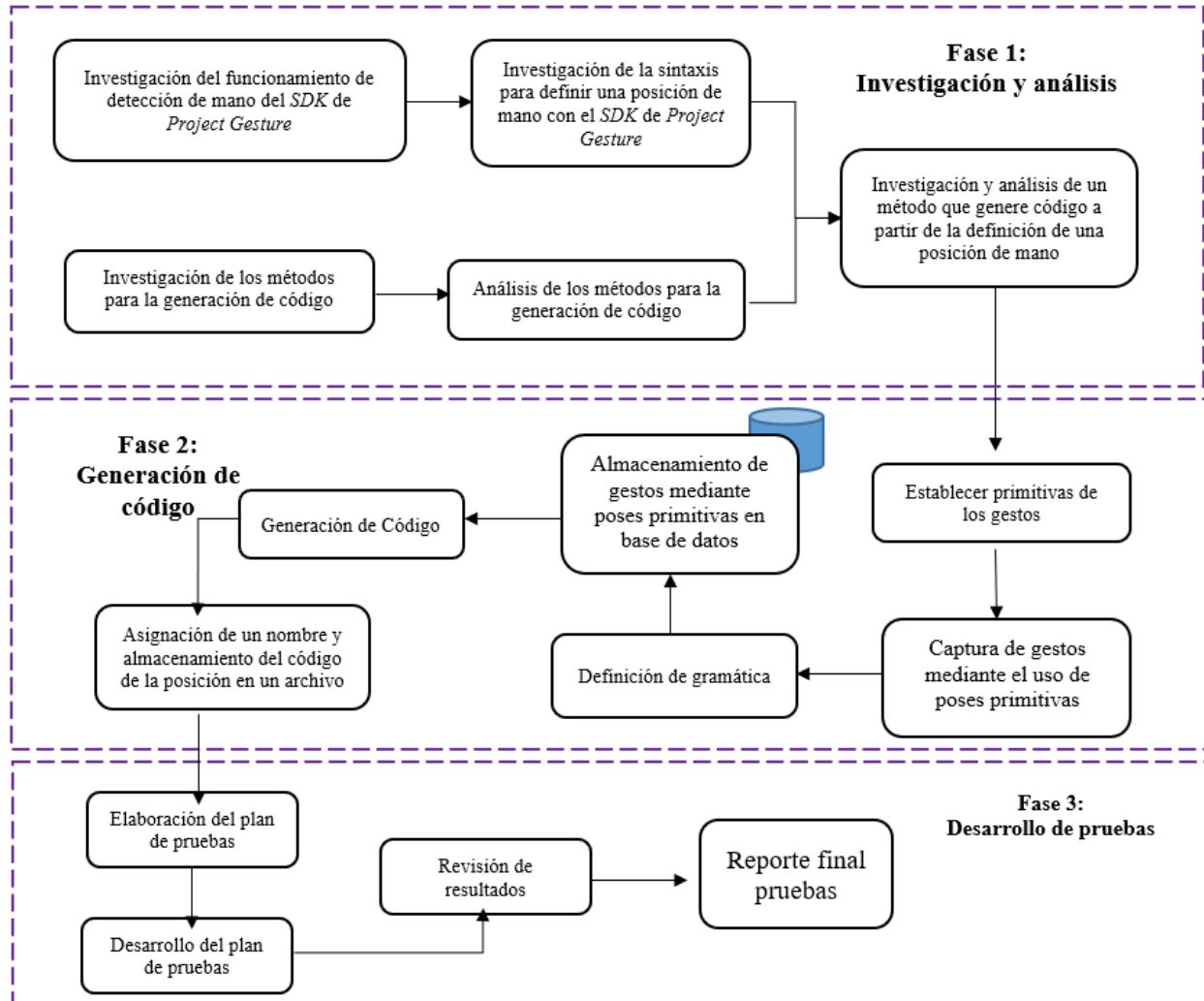
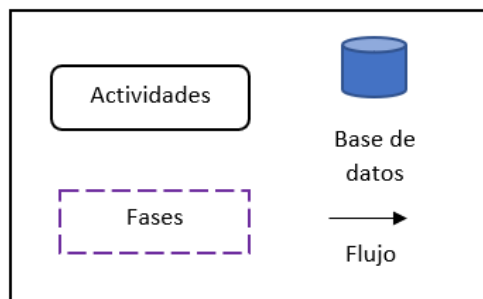


Figura 25: Metodología de solución.



Resumen de cada etapa aplicada en la metodología de solución (véase Figura 25).

#### Fase 1: Investigación y análisis

- Investigación del funcionamiento de detección de mano del SDK de Project Gesture: Como lo indica el nombre, se investigó en qué consiste la detección de manos por parte de Project Gesture, qué información recolecta o toma en cuenta para realizar esta tarea.
- Investigación de la sintaxis para definir una posición de mano con el SDK de Project Gesture: Se investigó la estructura del código de programación que debe existir para Project Gesture pueda identificar cierta pose de mano.
- Investigación de los métodos para la generación de código: Se investigó qué tipo de métodos o técnicas existen para llevar a cabo la tarea de generar código automáticamente y utilizarla para la solución de este trabajo.
- Evaluación de los métodos para la generación de código: Se analizó el desempeño de cada método encontrado en la etapa anterior, para concluir cuál sería la mejor opción para ser utilizado en la siguiente etapa de la metodología.
- Análisis y evaluación de un método que genere código a partir de la definición de una posición de mano: Se desarrolló y evaluó un método en base a lo investigado en las etapas anteriores, que sea capaz de tomar información del sensor de profundidad y transformarla a poses simples para la generación de código.

#### Fase 2: Generación de código

- Establecer primitivas de los gestos: Project Gesture al trabajar con partes específicas de la mano para la detección de una pose de mano, se establecieron las poses simples o primitivas que se usaron para interpretar la información que proporciona el sensor.
  - Captura de gestos mediante el uso de poses primitivas: Se creó un sistema que sea capaz de usar el sensor de profundidad, para poder realizar la detección de las manos con el Project Gesture, aplicando también el método de generación de código para hacer la relación de la información arrojada por el sensor y las primitivas a ser utilizadas para la generación de código.
  - Definición de gramática: Se definió el tipo de gramática que debe de presentar el código generado para que se pueda compilar y así ser reconocido por el Project Gesture.
  - Almacenamiento de gestos mediante poses primitivas en base de datos: Se diseñó la base de datos que servirá para almacenar todas las poses simples o primitivas que son capturadas por el sensor, además de almacenar el código generado de cada pose de mano.
  - Generación de Código: En esta etapa se creó el proceso para generar el código a partir de las poses primitivas o simples, que fueron almacenadas en la base de datos después de ser obtenidas por el sensor de profundidad.
  - Asignación de un nombre y almacenamiento del código de la posición en un archivo: Etapa en donde se llevó a cabo el proceso de solicitar al usuario un nombre o significado, el que será asignado a la pose de mano que fue capturado. Además de elaborar el proceso para generar el archivo que contendrá todos los códigos almacenados en la base de datos.
-

### Fase 3: Desarrollo de pruebas

- Elaboración del plan de pruebas: Se elaboró el plan de pruebas que se aplicó al sistema para ser evaluado.
- Desarrollo del plan de pruebas: Etapa donde se ejecutó el plan de pruebas desarrollado en la etapa anterior.

Iniciando por la descripción de la herramienta Project Gesture.

## 4.1. Project Gesture

Project Gesture es un SDK de vanguardia y fácil de usar, que crea experiencias más intuitivas y naturales al permitir a los usuarios controlar e interactuar con las tecnologías a través de gestos con las manos. Proporciona una API (interfaces de programación de aplicaciones) para C\#, C++ (incluidas las versiones UWP y .NET Core), lo que permite diseñar e implementar fácilmente gestos personalizados e integrarlos en aplicaciones (Krupka E. et al., 2017).

El lenguaje que propone Project Gesture está basado en cuatro predictores básicos, los cuales son naturalmente usados para describir posturas de la mano; estos son aplicados a los seis puntos principales de interés de la mano: las cinco puntas de los dedos y el centro de la palma. Los predictores son:

- Dirección de apuntamiento (“El pulgar apunta hacia arriba”)
- Ubicación relativa (“El dedo índice está sobre el dedo medio”)
- Tocando la punta del dedo (“El dedo anular toca el pulgar”)
- Flexión del dedo (“El meñique está doblado”)

Usando estos predictores son creadas 102 proposiciones básicas, las cuales sirven como los bloques de construcción binarios básicos del cálculo. Una pose de mano (algunas veces llamada “postura”) está definida principalmente como una conjunción de las proposiciones básicas, con disyunciones parcialmente permitidas en ciertos casos. Un gesto está definido como una secuencia de poses de la mano (Krupka E. et al., 2017).

Para resolver la tensión de velocidad y exactitud se hacen uso de clasificadores y regresores de conjunto de tablas convolucionales (CTE: Convolutional Table Ensemble). Estos son predictores extremadamente rápidos, ya que procesan una imagen en menos de un milisegundo. El pipeline incluye varias etapas y en cada una aplica un conjunto de CTEs (Bar-Hillel A., Krupka E., y Bloom N., 2016).

En la primera etapa, la posición del centro de la mano es encontrada y la imagen se centra alrededor de ella. Después se encuentra la orientación global de la mano, enmarcada como un problema de clasificación en 16 grupos de poses discretas, y luego es refinada (Krupka E. et al., 2017).

En el tercer paso la ubicación y dirección de las puntas de los dedos es encontrada, aplicando un regresor específico de grupo. Este regresor a su vez incluye varias etapas de regresión de la ubicación

de la punta del dedo, centrando la imagen alrededor de la punta del dedo y regresando de nuevo para perfeccionarla. Finalmente, el valor verdadero de las 102 preposiciones básicas del lenguaje es inferido de las puntas de los dedos y de la localización del centro de la palma (Krupka E. et al., 2017).

#### 4.1.1 Composición de la mano con referencia al SDK Project Gesture

El lenguaje del SDK Project Gesture consta de un conjunto de proposiciones básicas cualitativas (Figura 26). Las proposiciones son obtenidas por la aplicación de dos predicados de un argumento (dirección, flexión) y dos predicados de dos argumentos (dirección relativa, tangencia) para uno o dos de los seis puntos de interés en la mano: las puntas de los dedos y el centro de la palma. Las relaciones de dirección y orientación se cuantifican en seis valores canónicos: “Left (del sujeto)”, “Right”, “Up”, “Down”, “Forward” y “Backward”. Los otros dos predicados, flexión y tangencia, son naturalmente binarios (Krupka E. et al., 2017). A continuación, se presentan las proposiciones básicas:

- Pose de palma
  - Dirección de la palma: Esta es la dirección normal de la palma, apuntando hacia afuera. Aquí se utilizan los seis valores canónicos.
  - Orientación de la palma: La dirección apuntando desde la muñeca a la base del dedo medio. De nuevo se utilizan los seis valores canónicos.
- Dedos
  - Dirección de dedo: Para cada dedo se definieron las seis proposiciones de apuntamiento de las direcciones canónicas, para un total de treinta proposiciones.
  - Flexión de dedo: Para cada dedo dos estados son definidos, como “open” y “folded”, dando 10 proposiciones.
  - Tangencia de dedo: Para cada una de las 10 posibles combinaciones de pares de dedo, una “a is touching b” proposición es definida, también como una “a is not touching b”, para un total de 20 proposiciones.
  - Posición relativa de dedo: Para cada par de dedos, una proposición se define indicando que “a is in direction c from b” donde C es una de las 6 direcciones canónicas. Desde proposiciones como “middle is above the thumb” y “thumb is below the middle” son equivalentes, esto produce un total de 30 proposiciones.

(Krupka E. et al., 2017).

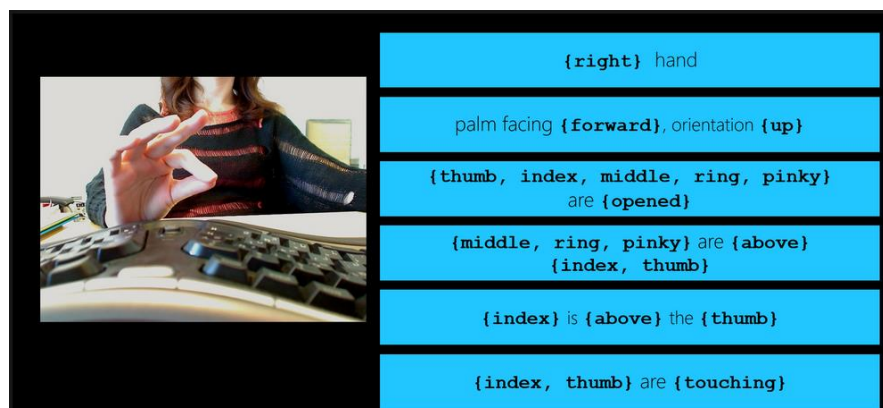


Figura 26: Composición de gestos con Project Gesture (Krupka E. et al., 2017).

## 4.1.2 Sintaxis para definir una posición de mano en Project Gesture

De acuerdo a Krupka E. et al. (2017) la sintaxis que se utiliza para implementar el SDK se presenta a continuación, la cual está incorporada al lenguaje de programación C#, donde se incluirán palabras reservadas propias del lenguaje de programación C# en los ejemplos que se mostrarán.

```
HandPose(nombre, conjunto_de_restricciones[]);
```

**HandPose:** describe el estado de todas las partes de la mano desde la muñeca hacia arriba.

[nombre]: nombre deseado para la nueva instancia de HandPose.

[conjunto\_de\_restricciones[]]: Una lista separada por comas de los objetos PoseConstraint que formarán el nuevo HandPose.

### Objetos PoseConstraint:

- **PalmPose:** Representa una restricción para la pose de la palma, que describe su orientación en el espacio y con qué mano está asociada.

```
PalmPose(HandContext , dirección, orientación);
```

- [HandContext]: Especifica qué HandContext se utilizará para crear el nuevo PalmPose: AnyHandContext o Hand
  - AnyHandContext: Especifica que puede ser cualquier mano, izquierda o derecha:
  - Hand: Contiene dos opciones para especificar qué mano se utilizará, izquierda o derecha: LeftHand o RightHand .
- [dirección]: (**PoseDirection**) La dirección perpendicular al plano de la cara de la palma.
- [orientación]: (**PoseDirection**) La dirección que estaría alineada con el dedo medio si se hubiera extendido.

**PoseDirection:** Especifica una dirección absoluta en el espacio. Se puede utilizar en la definición de restricciones PalmPose y FingerPose. Direcciones: Backward, Down, Forward, Left, Right, Up y Undefined.

Ejemplo:

```
var hi = new HandPose("HI", new PalmPose(new AnyHandContext(), PoseDirection.Forward, PoseDirection.Up));
```

- **FingerPose:** Representa una restricción que describe la flexión y la dirección de apuntado de un subconjunto de dedos.

```
FingerPose(dedos , flexión , dirección );
```

- [dedos]: ([Finger](#)): Enumera los dedos, se puede usar para especificar un `FingersContext` que luego se puede usar para definir las restricciones `FingerPose`, `FingertipDistanceRelation` y `FingertipPlacementRelation`. Dedos: Index, Middle, Pinky, Ring y Thumb
- [flexión]:([FingerFlexion](#)) Especifica el estado de flexión de la clase de un dedo. Se puede utilizar para definir restricciones `FingerPose`. Flexión: `Folded`, `FoldedTucked`, `Open`, `OpenStretched` y `Undefined`.
- [dirección]:([PoseDirection](#)) Especifica una dirección absoluta en el espacio. Se puede utilizar en la definición de restricciones `PalmPose` y `FingerPose`.

Ejemplo:

```
var miPose = new HandPose("MiPose", new FingerPose(new[] {Finger.Index,
    Finger.Thumb}, FingerFlexion.Open, PoseDirection.Forward));
```

- `FingertipDistanceRelation`: Crea una nueva instancia de una restricción

`FingertipDistanceRelation(dedo , relación , dedo);`

`FingertipDistanceRelation`, imponiendo la relación `distanceRelation` entre `finger` y otro `finger`.

- [dedo]: ([Finger](#)): Enumera los dedos. Se puede usar para especificar un `FingersContext` que luego se puede usar para definir las restricciones `FingerPose`, `FingertipDistanceRelation` y `FingertipPlacementRelation`. Dedos: Index, Middle, Pinky, Ring y Thumb.
- [relación]: ([RelativeDistance](#)) Especifica la distancia entre los contextos de dos dedos, se puede utilizar para definir las restricciones `FingertipDistanceRelation`. Distancia: `Touching` y `NotTouching`.
- [dedo] :([Finger](#)): Enumera los dedos, se puede usar para especificar un `FingersContext` que luego se puede usar para definir las restricciones `FingerPose`, `FingertipDistanceRelation` y `FingertipPlacementRelation`. Dedos: Index, Middle, Pinky, Ring y Thumb.

Ejemplo:

```
var PresionarPose = new HandPose("Presionar", new
    FingertipDistanceRelation(Finger.Index,
    RelativeDistance.Touching, Finger.Thumb));
```

- `FingertipPlacementRelation`: Representa una restricción que describe la ubicación relativa de los dedos.

`FingertipPlacementRelation(dedos , colocación ,dedo);`

- [dedo]: ([Finger](#)): Enumera los dedos, se puede usar para especificar un `FingersContext` que luego se puede usar para definir las restricciones `FingerPose`, `FingertipDistanceRelation` y `FingertipPlacementRelation`. Dedos: Index, Middle, Pinky, Ring y Thumb.



- [colocación]: ([RelativePlacement](#)) Especifica la relación de colocación espacial entre contextos de dos dedos. Se puede utilizar para definir restricciones [FingertipPlacementRelation](#). Colocacion: Above, Behind, Below, InFront, Left y Right.
- [dedo] :([Finger](#)): Enumera los dedos, se puede usar para especificar un [FingersContext](#) que luego se puede usar para definir las restricciones [FingerPose](#), [FingertipDistanceRelation](#) y [FingertipPlacementRelation](#). Dedos: Index, Middle, Pinky, Ring y Thumb.

Ejemplo:

```
var indexAboveThumbPose = new HandPose("IndexAboveThumb", new
FingertipPlacementRelation(Finger.Index,
RelativePlacement.Above, Finger.Thumb));
```

## 4.2. Investigación y análisis de métodos para la generación de código

En el análisis de la literatura se encontraron diferentes métodos para llevar a cabo la tarea de generar código. Los métodos encontrados, se llevan a cabo en los siguientes artículos:

- “Automatic Code Generation from UML State Chart Diagrams”
- “Code Generation on Mobile Devices for Mobile Apps”
- “Model Driven Development for Android Apps”
- “Automated Cross-Platform GUI Code Generation for Mobile Apps”

El método presentado en el artículo: “Automatic Code Generation From UML State Chart Diagrams”, consta de cuatro etapas: En primer lugar, el diagrama UML como dato de entrada, después la transformación de dicho modelo UML a XML, a continuación, se aplica un analizador sintáctico del archivo XML y como última etapa el código generado (Sunitha E. y Philip S.,2018).

En la actualidad existen herramientas que nos facilitan experimentar con la generación de código mediante diagramas UML y así poder emular el método presentado en este artículo. Para ello se utilizó la herramienta “Umple Online” (véase Figura 27).

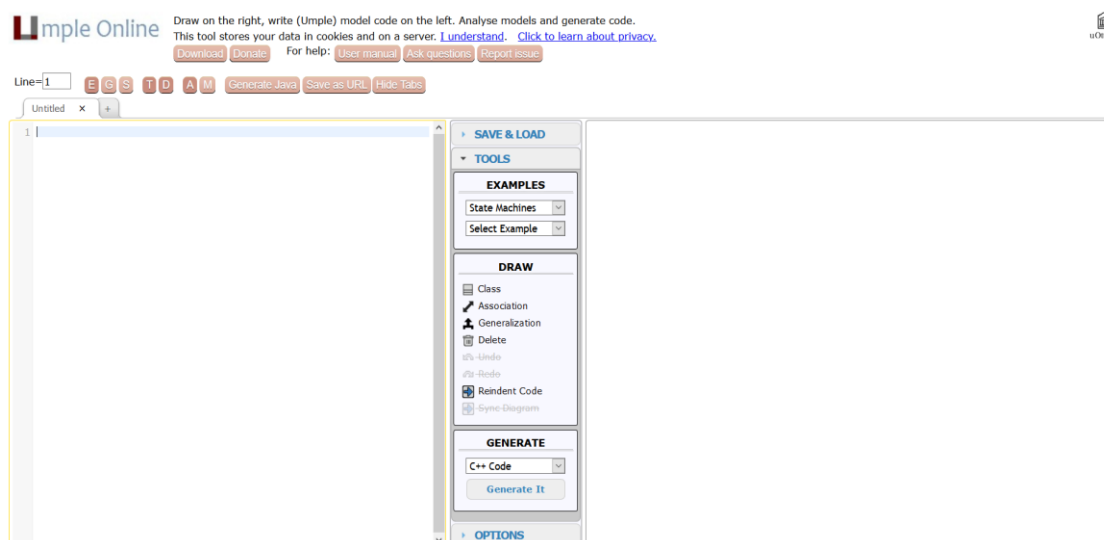


Figura 27: Umples Online (uOttawa, 2018).

Se experimentó con el siguiente diagrama UML como dato de entrada (véase Figura 28). El cuál está formado de la estructura de un gesto de mano, donde se incluye la palma con su respectiva dirección, orientación y la mano que se está usando (izquierda o derecha). También se incluyen los dedos que compondrán el gesto y sus características a tomar en cuenta como: la forma (abierto o doblado), si está tocando a otro dedo o si específicamente no debe estar tocando a otro dedo.

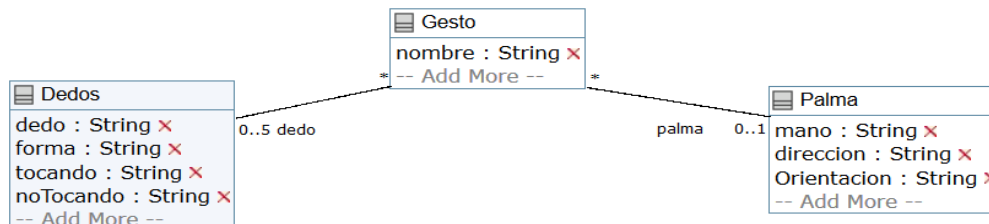


Figura 28: Diagrama UML de un gesto (uOttawa, 2018).

Con la herramienta “Umples Online” se hace la transformación al lenguaje XML (véase Figura 29).

```

01. <?xml version="1.0" encoding="UTF-8"?>
02. <uml:Model xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" xmlns:uml="http://www.eclipse.org/uml2/2.1.0/
03.   name="model">
04.   <packageImport xmi:id="_packageImport_0">
05.     <importPackage xmi:type="uml:Model" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#_0"/>
06.   </packageImport>
07.   <packagedElement xmi:type="uml:Model" xmi:id="dataType" name="dataType">
08.     <packagedElement xmi:type="uml:PrimitiveType" xmi:id="dataType-Time" name="Time"/>
09.   </packagedElement>
10.   <packagedElement xmi:type="uml:Class" xmi:id="_Gesto" name="Gesto">
11.     <ownedAttribute xmi:id="Gesto-nombre" name="nombre" visibility="private">
12.       <type xmi:type="uml:PrimitiveType" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
13.       <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="Gesto-nombre_upperValue" value="1"/>
14.       <lowerValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="Gesto-nombre_lowerValue" value="1"/>
15.     </ownedAttribute>
16.   </packagedElement>
17.   <packagedElement xmi:type="uml:Association" xmi:id="Gesto_Palma:palma" name="Gesto_Palma:palma" memberEnd="Gesto_Palma:palma-gestos_Gesto_Palma:palma"
18.     navigableOwnedEnd="Gesto_Palma:palma-gestos_Gesto_Palma:palma">
19.     <ownedEnd xmi:id="Gesto_Palma:palma-gestos" name="gestos" type="Gesto" association="Gesto_Palma:palma">
20.       <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="Gesto_Palma:palma-gestos_upperValue" value="*" />
21.       <lowerValue xmi:type="uml:LiteralInteger" xmi:id="Gesto_Palma:palma-gestos_lowerValue" value="0"/>
22.     </ownedEnd>
23.     <ownedEnd xmi:id="Gesto_Palma:palma-palma" name="palma" type="Palma" association="Gesto_Palma:palma">
24.       <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="Gesto_Palma:palma-palma_upperValue" value="1"/>
25.       <lowerValue xmi:type="uml:LiteralInteger" xmi:id="Gesto_Palma:palma-palma_lowerValue" value="0"/>
26.     </ownedEnd>
27.   </packagedElement>
28.   <packagedElement xmi:type="uml:Association" xmi:id="Dedos:dedo_Gesto" name="Dedos:dedo_Gesto" memberEnd="Dedos:dedo_Gesto-gestos_Dedos:dedo_Gesto-dedo"
29.     navigableOwnedEnd="Dedos:dedo_Gesto-gestos_Dedos:dedo_Gesto-dedo">
30.     <ownedEnd xmi:id="Dedos:dedo_Gesto-gestos" name="gestos" type="Gesto" association="Dedos:dedo_Gesto">
31.       <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="Dedos:dedo_Gesto-gestos_upperValue" value="*" />
32.       <lowerValue xmi:type="uml:LiteralInteger" xmi:id="Dedos:dedo_Gesto-gestos_lowerValue" value="0"/>
33.     </ownedEnd>
34.     <ownedEnd xmi:id="Dedos:dedo_Gesto-dedo" name="dedo" type="Dedos" association="Dedos:dedo_Gesto">
35.       <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="Dedos:dedo_Gesto-dedo_upperValue" value="5"/>
36.       <lowerValue xmi:type="uml:LiteralInteger" xmi:id="Dedos:dedo_Gesto-dedo_lowerValue" value="0"/>
37.     </ownedEnd>
38.   </packagedElement>
39. </uml:Model>

```

Figura 29: Diagrama de gesto UML a XML (uOttawa, 2018).

Después de generar el archivo XML se aplicaría el analizador sintáctico para generar el código a partir del archivo XML. “Umple Online” puede generar código en diferentes lenguajes como: java, php, c++, entre otros. Se muestra la generación de código en el lenguaje java (véase Figura 30).

```

006. import java.util.*;
007.
008.
009. // line 2 "model.ump"
010. // line 29 "model.ump"
011. public class Gesto
012. {
013.
014. //-----
015. // MEMBER VARIABLES
016. //-----
017.
018. //Gesto Attributes
019. private String nombre;
020.
021. //Gesto Associations
022. private Palma palma;
023. private List<Dedos> dedos;
024.
025. //-----
026. // CONSTRUCTOR
027. //-----
028.
029. public Gesto(String aNombre)
030. {
031.     nombre = aNombre;
032.     dedos = new ArrayList<Dedos>();
033. }
034.
035. //-----
036. // INTERFACE
037. //-----
038.
039. public boolean setNombre(String aNombre)
040. {
041.     boolean wasSet = false;
042.     nombre = aNombre;
043.     wasSet = true;
044.     return wasSet;
045. }
276. import java.util.*;
277.
278. // line 9 "model.ump"
279. // line 36 "model.ump"
280. public class Dedos
281. {
282.
283. //-----
284. // MEMBER VARIABLES
285. //-----
286.
287. //Dedos Attributes
288. private String dedo;
289. private String forma;
290. private String tocando;
291. private String noTocando;
292.
293. //Dedos Associations
294. private List<Gesto> gestos;
295.
296. //-----
297. // CONSTRUCTOR
298. //-----
299.
300. public Dedos(String aDedo, String aForma, String aTocando, String aNoTocando)
301. {
302.     dedo = aDedo;
303.     forma = aForma;
304.     tocando = aTocando;
305.     noTocando = aNoTocando;
306.     gestos = new ArrayList<Gesto>();
307. }
308.
309. //-----
310. // INTERFACE
311. //-----
312.
313. public boolean setDedo(String aDedo)
314. {
315.     boolean wasSet = false;
316.     dedo = aDedo;
317.     wasSet = true;
318.     return wasSet;
319. }
320.

```

Figura 30: Generación de código java (uOttawa, 2018).

En el segundo artículo: “Code Generation on Mobile Devices for Mobile Apps” el método que aplica consta de 5 etapas: en la primera se tiene una aplicación móvil, en la segunda etapa se recolectan los metadatos, en la tercera etapa los metadatos se almacenan en una base de datos, en la cuarta etapa se transforma la información de la base de datos para generar el código en la última etapa (Amanquah N. y Ndede S.,2017).

En esta ocasión no se encontraron herramientas que pudiera facilitar la simulación del método comentado, así que procedió a crear una aplicación que servirá para recolectar los metadatos (véase Figura 31).

Los metadatos que se recolectan con la aplicación, están enfocados al mismo diagrama UML presentado anteriormente (véase Figura 28).



Figura 31: Aplicación de recolección de datos.

Una vez seleccionadas las opciones que el gesto tendrá, los metadatos son enviados a una base de datos mediante un botón colocado al final de la selección de opciones, la base de datos posee la misma forma del diagrama de clase UML. Una vez que el gesto sea almacenado, con el segundo botón “gestos” se llevará al usuario a otra ventana donde podrá imprimir en pantalla el código de los gestos (véase Figura 32).



Figura 32: Generación de código de la base de datos.

El código que se genera toma forma del tema de tesis que se está llevando a cabo, ya que se busca que mediante un sensor de profundidad se puedan detectar gestos y así generar el código del gesto teniendo la misma estructura que este ejemplo.

El tercer artículo: “Model Driven Development for Android Apps” presenta un método bastante similar al primer artículo, ya que se basa en los diagramas de clase UML, pero además utiliza máquinas de estado. El método que se utiliza en este artículo consta de dos partes: la primera parte consiste en unos diagramas de clases UML para obtener la parte estática de la aplicación (cuadros, botones, texto, entre otros), y la segunda parte utilizan las máquinas de estado para obtener la parte dinámica (comportamiento) de la aplicación (Cheon Y. Barua A.,2018).

Para la primera parte se estaría usando la misma herramienta “Umple Online” para hacer la simulación de la generación del código estático del método, y también se estaría usando el mismo ejemplo que se mostró anteriormente. Para la segunda parte del uso de máquinas de estado, para el tema de tesis no se planea que el código generado tenga algún funcionamiento por tal motivo no se puede ejemplificar las máquinas de estado con el tema de la tesis. Pero de igual manera la herramienta “Umple Online” permite hacer uso de máquinas de estado (véase Figura 33) para la generación de código directamente (véase Figura 34, Figura 33).

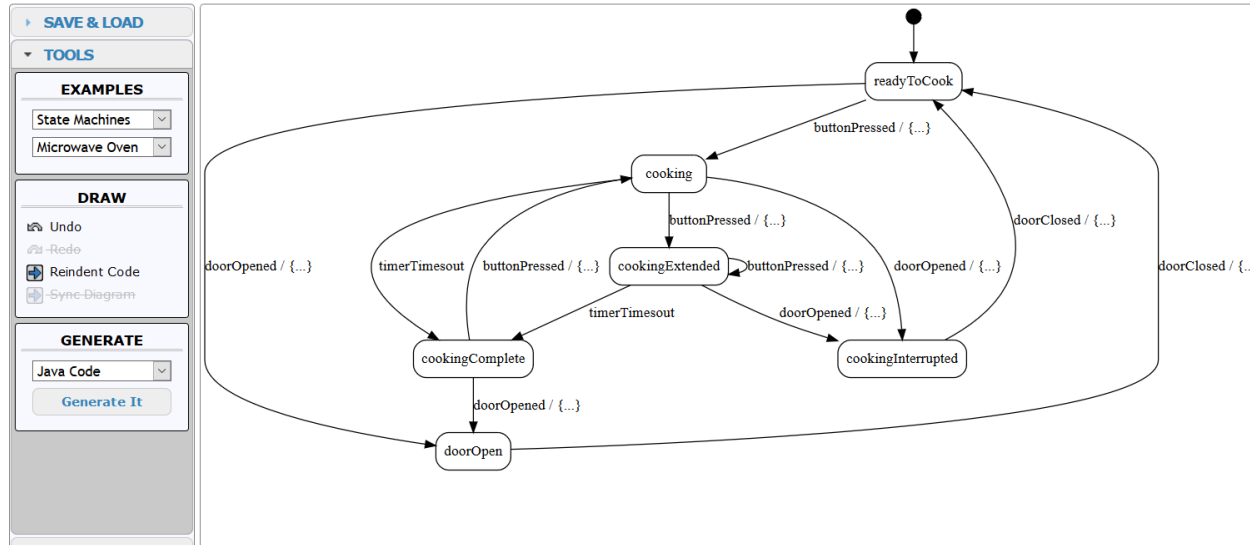


Figura 33: Ejemplo de máquinas de estado con “Uml Online” (uOttawa, 2018).

```

public Microwave()
{
    lightOn = false;
    powerTubeOn = false;
    isDoorOpened = false;
    isButtonPressed = false;
    timer = new Timer();
    setOperatingMicrowaveStateMachine(OperatingMicrowaveStateMachine.readyToCook);
    queue = new MessageQueue();
    removal=new Thread(this);
    //start the thread of Microwave
    removal.start();
}

//-----
// INTERFACE
//-----

public boolean setLightOn(boolean aLightOn)
{
    boolean wasSet = false;
    lightOn = aLightOn;
    wasSet = true;
    return wasSet;
}

public boolean setPowerTubeOn(boolean aPowerTubeOn)
{
    boolean wasSet = false;
    powerTubeOn = aPowerTubeOn;
    wasSet = true;
    return wasSet;
}

public boolean setIsDoorOpened(boolean aIsDoorOpened)
{
    boolean wasSet = false;
    isDoorOpened = aIsDoorOpened;
    wasSet = true;
    return wasSet;
}

public boolean setIsButtonPressed(boolean aIsButtonPressed)

```

Figura 34: Generación de código de una máquina de estado (uOttawa, 2018).

En el cuarto artículo: “Automated Cross-Platform GUI Code Generation for Mobile Apps” el método presentado consta de 4 etapas para la generación de código: la primera etapa consta de la captura de las imágenes, la segunda trata de una red neuronal convolucional entrenada para poder detectar los elementos de interés, en la tercera etapa se hace la relación entre los elementos

detectados en las imágenes y el fragmento de código que corresponderá a dichos elementos, para finalizar en la última etapa se lleva a cabo la generación de código (Sen C. et al., 2019).

En este caso, se hace uso del método “Template Matching” el cual, mediante técnicas de procesamiento de imágenes digitales, permite llevar a cabo la tarea de encontrar partes de una imagen donde coincidan con la imagen plantilla (Rosebrock A., 2021).

Por información de entrada se tendría la imagen a analizar (Figura 36) para detectar el objetivo de interés, en este caso un gesto “alto” (Figura 35).



*Figura 35: Gesto alto.*



*Figura 36: Imagen a analizar.*

Con ayuda del método “Template Matching” se realiza la detección del gesto en la imagen (véase Figura 37).

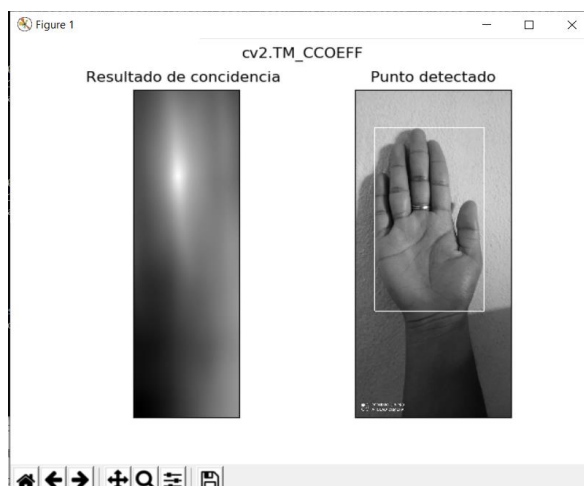


Figura 37: Detección de gesto con “Template Matching” (Rosebrock A., 2021).

El siguiente paso, sería la relación entre el objeto detectado y el código que le corresponde (véase Figura 38).

```

var alto = new HandPose("alto", new FingerPose(new[] {Finger.Thumb, Finger.Index, Finger.Middle, Finger.Ring, Finger.Pink }, FingerFlexion.Open, PoseDirection.Up)
    new FingertipDistanceRelation(Finger.Thumb, RelativeDistance.Touching, Finger.Index),
    new FingertipDistanceRelation(Finger.Index, RelativeDistance.Touching, Finger.Middle),
    new FingertipDistanceRelation(Finger.Middle, RelativeDistance.Touching, Finger.Ring),
    new FingertipDistanceRelation(Finger.Ring, RelativeDistance.Touching, Finger.Pink)
)
    
```

Figura 38: Código de gesto detectado.

### 4.2.1 Comentarios

Hablando acerca de la siguiente tabla (véase Tabla 5) el uso de un método que tenga como entrada un diagrama de clases UML sólo proporcionará código fuente de plantilla, ya que los diagramas de clases al funcionar como un objeto con características, solo serviría para almacenar información en dichas clases y en este caso no serviría para obtener el código fuente que se tiene como objetivo en la tesis. El uso de metadatos da una mejor solución para formar código fuente que tenga ciertas características específicas, como en este caso con formularios en una aplicación móvil. El usar imágenes como método de entrada se presenta como una mejor opción para evitar el proceso tardado que puede llegar a ser al introducir información en una aplicación o elaborar un diagrama UML. Además, usar una etapa intermedia para hacer una relación entre la imagen y el código, también puede ser útil para generar código fuente específico. Pero el hacer uso de “template matching” significaría consumir tiempo y recursos en la recolección de imágenes de todos los gestos posibles de la mano y se estaría dejando de lado el enfoque de utilizar Project Gesture el cual ya cuenta con este preentrenamiento para facilitar la creación de distintos gestos de mano.

Tabla 5: Análisis de métodos de generación de código

Método	Describe pose de mano	Imágenes como dato de entrada	Estructura igual a Project Gesture	No requiere entrenamiento extra
UML	x			x
Formulario App móvil	x		x	x
Máquina de estados				x
“Template Maching”	x	x	x	



### 4.3. Análisis y evaluación de un método que genere código a partir de la definición de una posición de mano

De acuerdo a la etapa anterior, se optó por la selección del método que plantea el artículo “Code Generation on Mobile Devices for Mobile Apps”, solamente que en lugar de hacer uso de una aplicación móvil para obtener los metadatos que describirían el comportamiento del gesto, se tomaron los datos que brindan la unión del sensor de profundidad y Project Gesture, para proporcionar estos metadatos y poder describir un gesto.

Durante el proceso de investigación para la solución del problema, se encontró el trabajo “Reconocimiento de gestos basado en acelerómetros” (Fernández E.,2014) donde se hace uso de la lógica difusa como herramienta de reconocimiento de posiciones de mano, obteniendo valores numéricos como datos de entrada de acelerómetros. Presentando buenos resultados de reconocimiento. Por esta razón y por la similitud que tienen el comportamiento de los acelerómetros y los datos arrojados por el sensor de profundidad junto con Project Gesture, se eligió el uso de la lógica difusa para ser el intermediario entre el usuario y la generación de código.

#### 4.3.1 Evaluación

En esta etapa se puso a prueba el uso de la lógica difusa, junto al método de obtener metadatos para poder generar a partir de estos un código que describa los gestos de mano. Para esto se construyó un sistema difuso enfocado solamente a la palma de la mano, para observar su desempeño y aplicarlo al resto de la mano.

##### Datos de entrada

Con ayuda del sensor de profundidad Intel RealSense SR300 y el SDK de Project Gesture, los datos que se obtienen del sensor son en tiempo real, los cuales pertenecen a tres ejes (x, y, z) que describen el comportamiento de la palma de la mano (véase Figura 39), esta información muestra la rotación presente en cada eje, dicha rotación de los ejes comprende un rango de [-1,1] la cual comprendería una rotación de 360°.



Figura 39: Ejes descriptores de la palma de la mano.

Con el comportamiento que presentaba la información respecto a los tres ejes de la mano, se optó por crear los siguientes conjuntos difusos para cada eje, los cuales fueron las tres variables de entrada al sistema difuso.

### Variables de entrada

Como se reciben información de los tres ejes de rotación de la palma de la mano, se optó por crear tres variables de entrada llevando los siguientes nombres:

- PalmaRotacionX
- PalmaRotacionY
- PalmaRotacionZ

Para cada variable de entrada, se crearon conjuntos difusos, los cuales por el comportamiento de Project Gesture que no existe rangos muy bien pronunciados para saber si la mano se encuentra en cierta dirección como lo haría la forma Trapezoidal, se eligió una forma Triangular, debido a que solo en un punto muy específico se está seguro que la mano está en cierta dirección. Por otra parte, al proporcionar el mismo tipo de información, cada entrada posee los mismos conjuntos difusos, nombres y rangos (véase Figura 40). Fueron nombrados de la siguiente manera (nombre y rango de pertenencia):

- muyNegativo: -1, -1, -0.65
- masNegativo: -0.8, -0.55, -0.25
- Negativo: -0.4, -0.25, -0.05
- pocoNegativo: -0.12, -0.05, 0.02
- pocoPositivo: -0.02, 0.05, 0.12
- Positivo: 0.05, 0.25, 0.4
- masPositivo: 0.25, 0.55, 0.8
- muyPositivo: 0.65, 1, 1

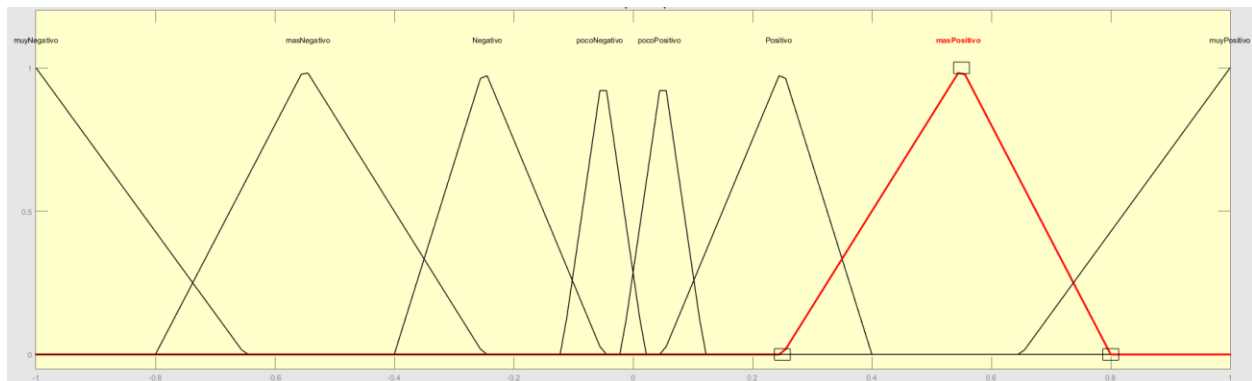


Figura 40: Representación gráfica de conjuntos difusos.

Los nombres de los conjuntos pudieron haber tenido otros nombres, pero por facilidad se eligieron esos nombres.

### Conjunto de reglas

En este caso, se crearon las siguientes reglas de inferencia para el reconocimiento de las direcciones de la mano (véase Tabla 6).

Tabla 6: Conjunto de reglas para el sistema difuso.

No.	<i>PalmaRotacionX</i>	<i>PalmaRotacionY</i>	<i>PalmaRotacionZ</i>	<i>Pose de palma</i>
1	pocoPositivo	muyPositivo	pocoNegativo	AbajoFrente
2	pocoNegativo	muyNegativo	Positivo	ArribFrente
3	Negativo	pocoNegativo	muyPositivo	AtrasArriba
4	masPositivo	masPositivo	muyNegativo	AdelanteArriba
5	muyNegativo	masPositivo	pocoNegativo	DerechaArriba
6	muyPositivo	Positivo	Positivo	IzquierdaArriba
7	Negativo	masPositivo	muyNegativo	AdelanteIzquierda
8	masPositivo	pocoPositivo	muyNegativo	AdelanteDerecha
9	pocoNegativo	masNegativo	muyNegativo	AdelanteAbajo
10	pocoNegativo	Negativo	muyPositivo	AtrasIzquierda
11	pocoPositivo	Positivo	muyPositivo	AtrasAbajo
12	Negativo	Positivo	muyPositivo	AtrasDerecha
13	Negativo	muyNegativo	masPositivo	ArribaIzquierda
14	Positivo	muyNegativo	pocoPositivo	ArribaDerecha
15	pocoPositivo	muyPositivo	pocoPositivo	AbajoDerecha
16	pocoNegativo	muyPositivo	Positivo	AbajoIzquierda
17	muyPositivo	pocoNegativo	Positivo	IzquierdaFrente
18	muyNegativo	Positivo	Negativo	DerechaFrente
19	muyPositivo	Negativo	masPositivo	IzquierdaAbajo
20	muyNegativo	pocoPositivo	pocoNegativo	DerechaAbajo

Haciendo un total de 20 reglas. Como se requiere el uso de las 3 variables, ya que de eso depende en qué direcciones se encuentra la palma de la mano, no se colocaron reglas donde solo se incluyeran dos o menos variables.

### Variables de salida y mecanismo de inferencia

Las variables de salida comprenden conjuntos difusos muy simples, ya que cada salida posee un rango de 1 unidad, teniendo un rango total de 20 unidades, para 20 posibles resultados. Esto se seleccionó de esta forma ya que se pretendió relacionar un número con una dirección específica.

Por otra parte, para el mecanismo de inferencia, se utilizó el método mamdani, ya que se poseen conjuntos difusos tanto en el antecedente como en el consecuente, además de que el sistema difuso no es complejo para el uso del método takagi-sugeno.

## Resultados

Tabla 7: Resultados de sistema difuso.

	N° muestras	Correctas	Incorrectas mismo gesto	Incorrectas gesto erroneo	Reconocimiento
<b>AbaFren</b>	<b>10</b>	<b>6</b>	<b>4</b>	<b>0</b>	<b>60%</b>
<b>ArriFren</b>	<b>10</b>	<b>9</b>	<b>0</b>	<b>1</b>	<b>90%</b>
<b>AtrArri</b>	<b>10</b>	<b>8</b>	<b>2</b>	<b>0</b>	<b>80%</b>
<b>AdeArri</b>	<b>10</b>	<b>10</b>	<b>0</b>	<b>0</b>	<b>100%</b>
<b>DerArri</b>	<b>10</b>	<b>3</b>	<b>4</b>	<b>3</b>	<b>30%</b>
<b>IzqArri</b>	<b>10</b>	<b>10</b>	<b>0</b>	<b>0</b>	<b>100%</b>
<b>AdIzq</b>	<b>10</b>	<b>10</b>	<b>0</b>	<b>0</b>	<b>100%</b>
<b>AdDer</b>	<b>10</b>	<b>7</b>	<b>3</b>	<b>0</b>	<b>70%</b>
<b>AdeAbaj</b>	<b>10</b>	<b>6</b>	<b>4</b>	<b>2</b>	<b>60%</b>
<b>AtrIzq</b>	<b>10</b>	<b>10</b>	<b>0</b>	<b>0</b>	<b>100%</b>
<b>AtrAbajo</b>	<b>10</b>	<b>3</b>	<b>3</b>	<b>4</b>	<b>30%</b>
<b>AtrDer</b>	<b>10</b>	<b>7</b>	<b>3</b>	<b>0</b>	<b>70%</b>
<b>ArrIzq</b>	<b>10</b>	<b>10</b>	<b>0</b>	<b>0</b>	<b>100%</b>
<b>ArriDer</b>	<b>10</b>	<b>7</b>	<b>3</b>	<b>0</b>	<b>70%</b>
<b>AbajDer</b>	<b>10</b>	<b>7</b>	<b>2</b>	<b>1</b>	<b>70%</b>
<b>AbajIzq</b>	<b>10</b>	<b>9</b>	<b>1</b>	<b>0</b>	<b>90%</b>
<b>IzqFren</b>	<b>10</b>	<b>9</b>	<b>0</b>	<b>1</b>	<b>90%</b>
<b>DerFren</b>	<b>10</b>	<b>10</b>	<b>0</b>	<b>0</b>	<b>100%</b>
<b>IzqAbaj</b>	<b>10</b>	<b>10</b>	<b>0</b>	<b>0</b>	<b>100%</b>
<b>DerAbaj</b>	<b>10</b>	<b>0</b>	<b>6</b>	<b>4</b>	<b>0%</b>

### Comentarios

Como se puede observar en la Tabla 7 de resultados, la mayoría de los gestos muestran un porcentaje bastante aceptable, no obstante, en esta prueba solo se utilizó un conjunto de datos de rotación (x, y, z) de los dos que proporciona el sensor de profundidad, a pesar de que no se especifica el significado de cada uno de estos conjuntos de datos, se decidió llamarlos *Orientación* y *Dirección*. Con esto se planeaba que el reconocimiento de los gestos se elevara, ya que el sistema difuso tendría mayor información para poder obtener los resultados esperados. Este conjunto de entradas nuevas al sistema difuso y el replanteamiento de las reglas difusas se presentan más adelante.

Por otro lado, se puede observar que el uso de un sistema difuso para llevar a cabo una interpretación es bastante útil siempre y cuando la información que se establezca sea la más adecuada para lograr dicho fin.

## 4.4. Poses primitivas para la formación de un gesto

Al tratar con gestos de las manos, se espera tener la mayor cantidad de gestos diferentes entre sí, y esto es posible ya que el Project Gesture, aunque no cuenta con muchos elementos, estos pocos

elementos pueden tomar diferentes poses que pueden contribuir a que con un pequeño movimiento sea un gesto totalmente diferente. En esta parte se definen esas poses primitivas o simples que ayudarán a diferenciar y a crear varias poses diferentes.

El SDK toma una estructura específica en el fragmento de código que hace alusión al gesto que se quiere detectar. Este fragmento de código específico consta de pequeñas y simples poses que en su conjunto forman la descripción de un gesto o pose de mano. Por ejemplo, si la mano se encuentra de frente y los dedos están estirados (Krupka E. et al., 2017).

Las poses simples o primitivas que pertenecen en la descripción de una pose de mano o un gesto se conforman en dos grupos.

- Mano
- Dedos

En el primer grupo que es la “Mano” consta de dos categorías: Orientación y Dirección. Para ambas categorías las poses primitivas se definen de la siguiente manera:

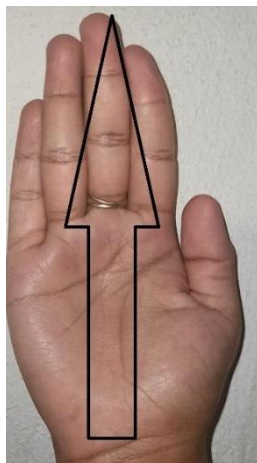
- Adelante (véase Figura 43)
- Abajo (véase Figura 44)
- Atrás (véase Figura 45)
- Arriba (véase Figura 46)
- Izquierda (véase Figura 47)
- Derecha (véase Figura 48)

A pesar que ambas categorías constan de las mismas poses primitivas, no se representan de la misma manera. Para la categoría de “Orientación” se habla de la vista de la palma de la mano (véase Figura 41)



*Figura 41: Orientación de la mano.*

En la categoría “Dirección” se habla del apuntamiento que hace la mano, se trata de una flecha que va desde la muñeca en dirección hacia la punta del dedo medio (véase Figura 42).



*Figura 42: Componentes de un sistema difuso.*

Para una mejor aclaración, a continuación, se hace la comparación de las poses primitivas en ambas categorías.



*Figura 43: Orientación y Dirección: Adelante.*

La orientación adelante, mientras este la palma de frente, tomará ese valor, no importa si la punta de los dedos está en otra dirección, de igual forma la dirección de los dedos y ni la orientación afectará la dirección de la palma de la mano (véase Figura 43), esta condición aplica a las demás poses de la palma de la mano (véase Figura 44, Figura 45, Figura 46, Figura 47, Figura 48).

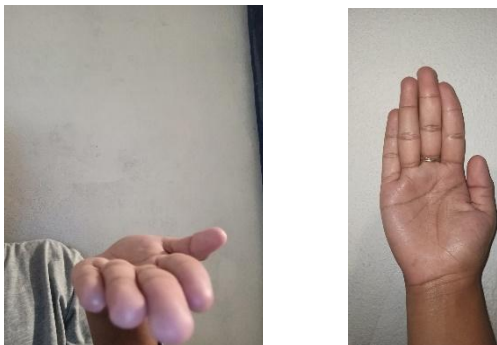


*Figura 44: Orientación y Dirección: Abajo.*



*Figura 45: Orientación y Dirección: Atrás.*

En el caso de la pose primitiva “Atrás” de dirección (véase Figura 45), no es posible que se pueda llegar a detectar, ya que las capacidades del sensor no lo permiten.



*Figura 46 : Orientación y Dirección: Arriba.*



*Figura 47: Orientación y Dirección: Izquierda.*



Figura 48: Orientación y Dirección: Derecha.

Pasando al grupo “Dedos”, este grupo consta de tres categorías las cuales se están manejando de la siguiente manera: forma, dirección e interacción. En la primera categoría que es “forma” las poses primitivas son (véase Figura 49):

- Abierto
- Doblado

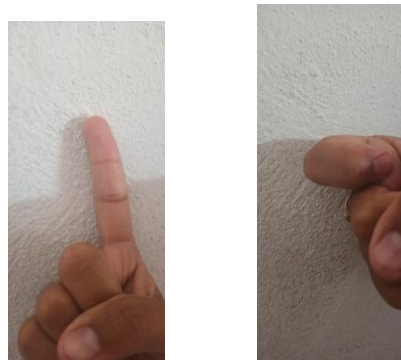


Figura 49: Dedo abierto y doblado.

En la categoría de “dirección” (véase Figura 50) se tienen las mismas poses primitivas que en la mano, las cuales son:

- Arriba
- Abajo
- Derecha
- Izquierda
- Adelante
- Atrás



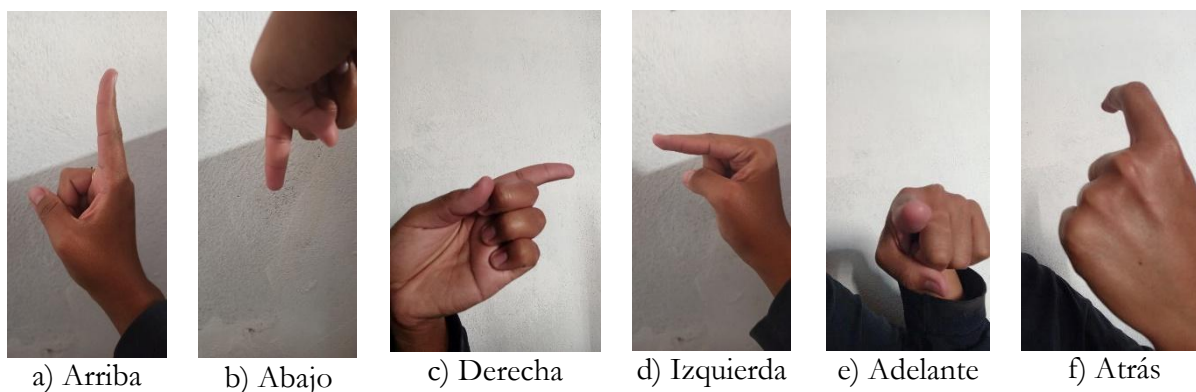


Figura 50: Dirección de dedos.

En la categoría de “interacción” (véase Figura 51) se hace referencia como lo dice su nombre a la interacción que tiene un dedo con otro, en este caso, las poses primitivas son:

- Tocando
- NoTocando

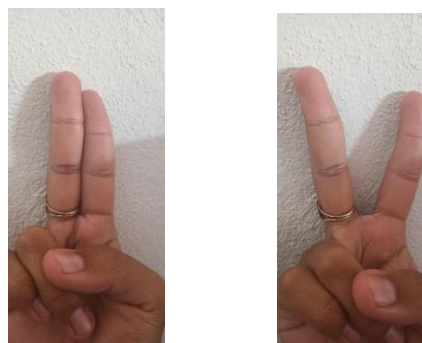


Figura 51: Interacción de dedos.

### Comentarios

A pesar que en una ocasión la pose primitiva “Atrás” de la categoría “dirección” de la palma de la mano no se pueda llegar a identificar, el resto de las poses primitivas indican, por la gran cantidad que existe, que aún se pueden crear e identificar un gran número de poses diferentes y así ser útil para el desarrollo de futuras interfaces naturales de usuario.

## 4.5. Almacenamiento de gestos mediante poses primitivas en base de datos

### 4.5.1 Captura de gestos

En este punto se comienza a explicar cómo se logra cumplir el objetivo específico “Definir un proceso que genere código automáticamente a partir de cada posición de la mano” (véase Figura 53).

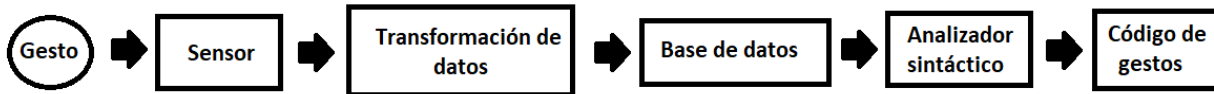


Figura 52: Proceso para la generación de código de gestos

Resumen la Figura 52: En la primera etapa como información de entrada es el gesto, del cual se quiere generar su código de programación de acuerdo a Project Gesture. Este gesto es reconocido por el sensor de profundidad y el SDK de Project Gesture, los cuales proporcionan un conjunto de datos que caracterizan a la mano en cuestión. Estos datos se necesitan transformar para obtener nuestras poses primitivas o simples, así que en esta etapa se hace uso de la lógica difusa para reconocer las poses simples. Una vez obtenidas estas poses simples, son almacenadas en una base de datos de acuerdo a una estructura ya diseñada, para que el analizador sintáctico mediante una gramática formal obtenga las poses simples desde la base de datos y se proceda con la transformación a código de programación.

```
1. Entrada: Gesto de mano
2. Salida: Fragmento de código de programación(c#) que describe el gesto de mano
3. INICIO
4.   datos_normalizados <- Captura de gesto por sensor RealSenseSR300
5.   Sistema_difuso_1 Lee datos_normalizados de la palma de la mano
6.   valores_puntuales1 <- resultado de Sistema_difuso_1
7.   Clasificacion_Direccion_y_Orientacion_palmaMano Lee valores_puntuales1
8.   poses_primitivas_Direccion_y_Orientacion<-
   Clasificacion_Direccion_y_Orientacion_palmaMano
9.   Base_de_datos Guarda poses_primitivas_Direccion_y_Orientacion
10.  REPETIR 5 VECES POR CADA DEDO
11.   Sistema_difuso_2 Lee datos_normalizados de los dedos
12.   Distancias_euclideana_tridimensional_contra_resto_de_dedos Lee datos_normalizados de
13.   Valores_puntuales2 <- resultado de Sistema_difuso_2
14.   Clasificacion_Direccion_dedo Lee valores_puntuales2
15.   pose_primitiva_direccionDedo <- Clasificacion_Direccion_dedo
16.   Base_de_datos Guarda pose_primitiva_direccionDedo
17.   Clasificacion_Interaccion_Dedos<-
   Distancias_euclideana_tridimensional_contra_resto_de_dedos
18.   poses_primitivas_interaccion<- Clasificacion_Interaccion_Dedos
19.   Base_de_datos Guarda poses_primitivas_interaccion
20.  FIN REPETICION
21.  Leer Base_de_datos
22.  poses_primitivas <- Base_de_datos
23.  Gramatica_libre_de_contexto Lee poses_primitivas
24.  Conversion_a_codigo_fuente Lee Gramatica_libre_de_contexto
25. FIN
```

Figura 53: Procedimiento de generación de código de gesto de mano.

### Entrada

Como dato de entrada se utiliza el gesto/pose de mano, el que es realizado, en tiempo real, por el usuario frente al sensor de profundidad *RealSense SR300*.

### Salida

Se obtiene el fragmento de código de programación (C#) que describe el gesto/pose de mano (véase Figura 54).

```
var A= new HandPose("A",new PalmPose(Hand.RightHand,PoseDirection.Forward,PoseDirection.Up)
,new FingerPose(new[] {Finger.Thumb}, FingerFlexion.Open),
new FingerPose(new[] {Finger.Index,Finger.Middle,Finger.Ring,Finger.Pinky}, FingerFlexion.Folded)
,new FingerPose(new[] {Finger.Thumb},PoseDirection.Up)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.Touching,Finger.Middle)
,new FingertipDistanceRelation(Finger.Middle,RelativeDistance.Touching,Finger.Ring)
,new FingertipDistanceRelation(Finger.Ring,RelativeDistance.Touching,Finger.Pinky)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.NotTouching,Finger.Pinky)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.NotTouching,Finger.Ring)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.NotTouching,Finger.Thumb)
```

Figura 54: Fragmento de código que describe el comportamiento del gesto de mano.

### Inicio

Este proceso inicia con la recolección de datos puntuales normalizados, los que son obtenidos por el sensor de profundidad, los cuales representan características del comportamiento que está teniendo la mano al ser detectada. En esta trama de datos se tienen siete grupos de datos de rotación (x, y, z) y seis puntos de ubicación tridimensional (véase Figura 55). De los siete grupos de datos de rotación, dos grupos se utilizan para saber que orientación tiene la palma de la mano (aquí se utilizan los términos: orientación y dirección), el resto de los grupos corresponden a cada uno de los dedos de la mano, los cuales son utilizados para saber la orientación de cada uno de ellos. Los puntos de ubicación tridimensional corresponden a los dedos y al centro de la palma de la mano, estos son utilizados para calcular la distancia que tienen unos con otros y saber el comportamiento que están teniendo los dedos.



Figura 55: Información proporcionada por el sensor.

Los grupos de rotación son tratados mediante dos sistemas difusos independientes, el primer sistema está enfocado para la palma de la mano y el segundo sistema se enfoca en la orientación de los dedos.

Para ambos sistemas difusos las entradas son los grupos de rotación (x, y, z), en el caso del primer sistema éste recolecta seis ejes de rotación, tres que son para determinar a donde “ve” la palma de la mano (nombrándola, así como “Orientación”) y los otros tres son para determinar hacia donde apuntan los dedos (tomando como referencia que hablaríamos de una mano con los dedos extendidos como en la Figura 55 nombrándola, así como “Dirección”).

Para la parte de **fuzzificación** se crearon seis variables de entrada, tres para los tres ejes de rotación de la “Orientación” y otros tres para los ejes de rotación de la “Dirección”. Como los

sistemas difusos no se rigen por un comportamiento específico y estos se diseñan de acuerdo a las necesidades que el autor tenga, después de analizar el comportamiento de los datos y en qué momento los datos tienen una mayor pertenencia respecto a las poses simples: *Adelante, Atrás, Arriba, Abajo, Derecha e Izquierda*; cada una de las variables de entrada cuentan con tres conjuntos difusos. Por el comportamiento que los datos presentan se optó por conjuntos difusos triangulares la cual su función de pertenencia es la siguiente (véase Ecuación 24)

$$\mu_A(x) = \begin{cases} 0, & \text{si } x \leq a \\ \frac{x - a}{b - a}, & \text{si } a \leq x \leq b \\ \frac{c - x}{c - b}, & \text{si } b \leq x \leq c \\ 0, & \text{si } x \geq c \end{cases} \quad (24)$$

Donde  $\mu_A(x)$  representa la función de pertenencia que se define por un límite inferior "a", un límite superior "b", y un valor "c".

Los conjuntos difusos contienen el siguiente comportamiento: [-1 -1 -0.4] [-0.6 0 0.6] [0.4 1 1] teniendo el nombre de: negativo, neutral y positivo; respectivamente (véase Figura 56) esto para las seis variables de entrada.

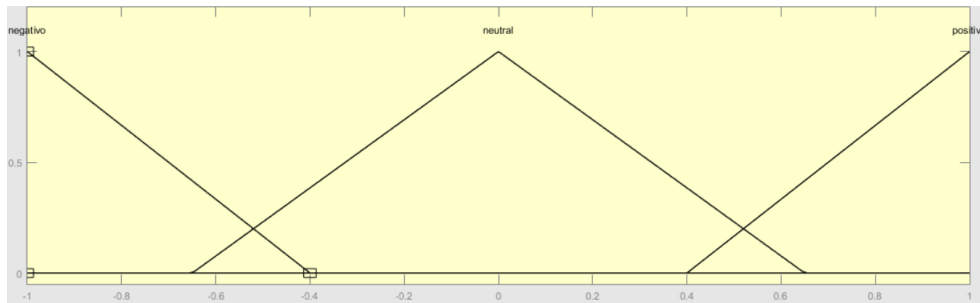


Figura 56: Conjuntos difusos de variables de entrada.

Para las variables difusas de salida en este sistema se cuenta con dos variables de salida, una llamada "Dirección" y otra "Orientación". Cada una de ellas cuentan con seis conjuntos: Arriba, Abajo, Adelante, Atrás, Derecha e Izquierda; cada conjunto difuso contempla la unidad, es decir; el conjunto Arriba va de 0 a 1, el conjunto Abajo va de 1 a 2, y así sucesivamente. También debido al comportamiento se eligieron conjuntos difusos triangulares (véase Figura 57).

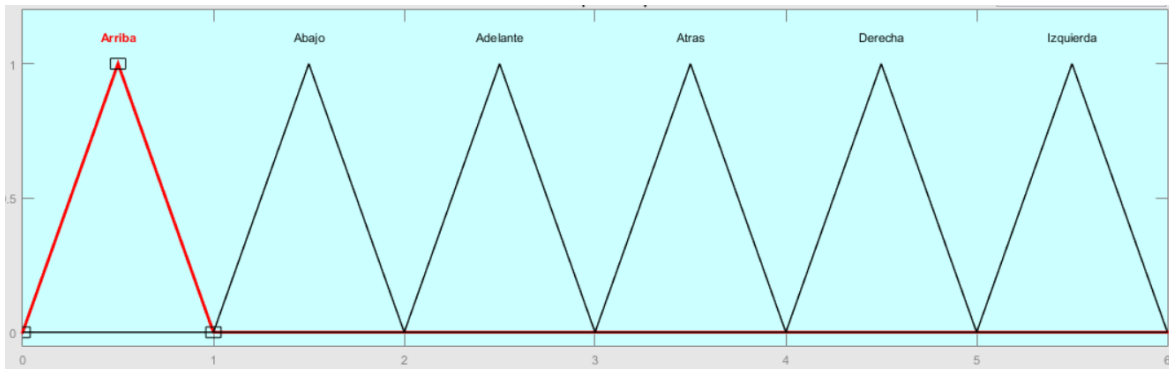


Figura 57: Conjuntos difusos de variables de salida.

Debido a las características presentadas hasta ahora, se hace uso del **método de inferencia mamdani**, ya que es el más común y simple, ya que se basa en variables difusas tanto en el antecedente como en el consecuente (véase Ecuación 25).

$$\text{IF } x \text{ es } A3 \text{ OR } y \text{ es } B1 \text{ THEN } z \text{ es } C1 \quad (25)$$

Donde el antecedente es  $x$  A3 OR,  $y$  es B1 y el consecuente  $z$  es C1, y los conjuntos de pertenencia son representados por A3, B1, C1.

Para el **conjunto de reglas** que toman el papel del conocimiento del sistema difuso, se toman en cuenta un total de 24 reglas (véase Tabla 8), aunque pudiesen crearse más reglas estas restantes no cumplirían con el comportamiento deseado que se esperaría del sistema difuso, originando así resultados erróneos

Tabla 8: Conjunto de reglas para la palma de la mano

No.	Orientación			Dirección		X	Pose simple	Pose simple
	X	Y	Z	X	Y			
1	Neutral	Neutral	Negativo	Neutral	Negativo	Neutral	Adelante	Arriba
2	Neutral	Neutral	Negativo	Negativo	Neutral	Neutral	Adelante	Derecha
3	Neutral	Neutral	Negativo	Positivo	Neutral	Neutral	Adelante	Izquierda
4	Neutral	Neutral	Negativo	Neutral	Positivo	Neutral	Adelante	Abajo
5	Neutral	Neutral	Positivo	Neutral	Negativo	Neutral	Atrás	Arriba
6	Neutral	Neutral	Positivo	Neutral	Positivo	Neutral	Atrás	Abajo
7	Neutral	Neutral	Positivo	Positivo	Neutral	Neutral	Atrás	Izquierda
8	Neutral	Neutral	Positivo	Negativo	Neutral	Neutral	Atrás	Derecha
9	Neutral	Negativo	Neutral	Neutral	Neutral	Negativo	Arriba	Adelante
10	Neutral	Negativo	Neutral	Positivo	Neutral	Neutral	Arriba	Izquierda
11	Neutral	Negativo	Neutral	Negativo	Neutral	Neutral	Arriba	Derecha
12	Neutral	Negativo	Neutral	Neutral	Neutral	Positivo	Arriba	Atrás
13	Neutral	Positivo	Neutral	Neutral	Neutral	Negativo	Abajo	Adelante
14	Neutral	Positivo	Neutral	Positivo	Neutral	Neutral	Abajo	Izquierda

15	Neutral	Positivo	Neutral	Negativo	Neutral	Neutral	Abajo	Derecha
16	Neutral	Positivo	Neutral	Neutral	Neutral	Positivo	Abajo	Atrás
17	Positivo	Neutral	Neutral	Neutral	Negativo	Neutral	Izquierda	Arriba
18	Positivo	Neutral	Neutral	Neutral	Neutral	Negativo	Izquierda	Adelante
19	Positivo	Neutral	Neutral	Neutral	Positivo	Neutral	Izquierda	Abajo
20	Positivo	Neutral	Neutral	Neutral	Neutral	Positivo	Izquierda	Atrás
21	Negativo	Neutral	Neutral	Neutral	Negativo	Neutral	Derecha	Arriba
22	Negativo	Neutral	Neutral	Neutral	Neutral	Negativo	Derecha	Adelante
23	Negativo	Neutral	Neutral	Neutral	Positivo	Neutral	Derecha	Abajo
24	Negativo	Neutral	Neutral	Neutral	Neutral	Positivo	Derecha	Atrás

En conjunto de reglas se utiliza una conexión de unión

Donde la unión de los conjuntos difusos  $\underline{A} \cup \underline{B}$ , es igual al *max* de las funciones de pertenencia de  $\underline{A}$  y  $\underline{B}$ .

Para la parte de **Desfuzzificación** donde se hace la transformación de variables difusas hacia variables puntuales, se hace uso del método más eficiente que es: **Centroide**.

Para el **segundo sistema difuso** el cual se enfoca para calcular la “Dirección” de los dedos, se conserva la mayoría de las características del primer sistema difuso, solo que, en este caso, este sistema solo contiene tres variables de entrada (x, y, z), pero cuenta con el mismo número y comportamiento de conjuntos difusos para cada variable (véase Figura 58).

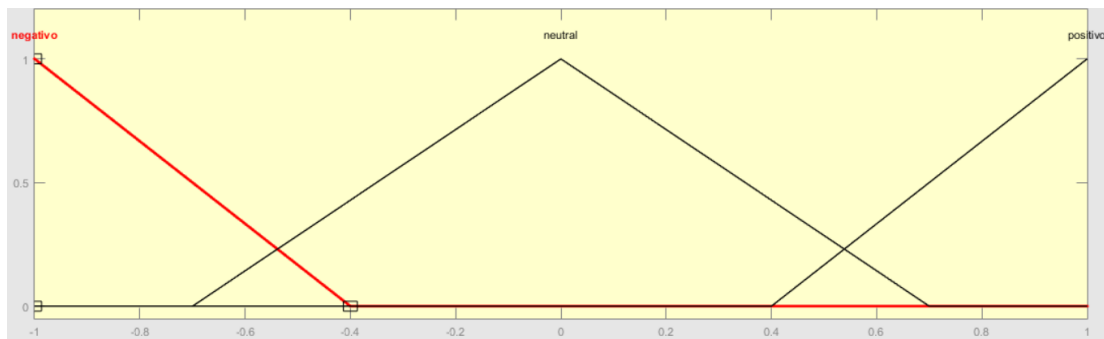


Figura 58: Conjuntos difusos de variables de entrada del sistema difuso de los dedos.

Donde los rangos de los conjuntos son los siguientes: [-1 -1 -0.4] [-0.6 0 0.6] [0.4 1 1] llevando el nombre de “negativo”, “neutral” y “positivo” respectivamente. Y sólo una variable de salida para la desfuzzificación, pero también con los mismo seis conjuntos de salida: Arriba, Abajo, Derecha, Izquierda, Adelante y Atrás (véase Figura 59).

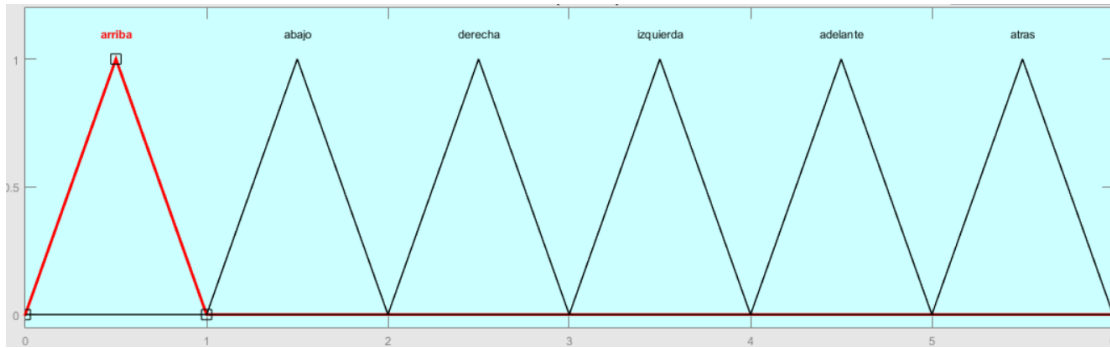


Figura 59: Conjuntos difusos de la variable de salida del sistema difuso de los dedos

El conjunto de reglas (véase Tabla 9) disminuye en este sistema, contemplando un total de seis reglas, de la misma forma que en el caso anterior, solo se escogieron éstas para obtener los resultados deseados.

Tabla 9: Conjunto de reglas para la dirección de los dedos

No	X	Y	Z	Dirección
1	Neutral	Negativo	Neutral	Arriba
2	Neutral	Positivo	Neutral	Abajo
3	Negativo	Neutral	Neutral	Derecha
4	Positivo	Neutral	Neutral	Izquierda
5	Neutral	Neutral	Negativo	Adelante
6	Neutral	Neutral	Positivo	Atrás

Este segundo sistema difuso se aplica a cada uno de los cinco dedos, una vez que se obtienen los resultados se procesa una simple clasificación de los datos puntuales, los que son almacenados como palabras (poses simples) en una base de datos, es decir; si el dato puntual es 1 = Arriba, y así con el resto. Esto también se aplica con los resultados del sistema difuso que está enfocado a la palma de la mano, y de igual forma se almacenan las palabras, poses simples, que hacen referencia a la “Orientación” y “Dirección”.

Dentro de la repetición de los cinco dedos se calculan las **distancias euclidianas tridimensionales** de los seis puntos. De acuerdo al dedo que este en turno, se calcularán las distancias euclídeas respecto a los otros dedos e incluso con el centro de la palma de la mano, esto para obtener la interacción que se está teniendo con los otros dedos (si se encuentra “Tocando” o “NoTocando”, véase Figura 60) y respecto a la palma de la mano es para saber si el dedo en turno se encuentra “Abierto” o “Doblado” (véase Figura 61).

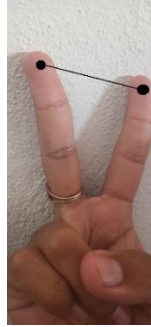


Figura 60: Distancia para la interacción entre los dedos.

Para establecer la distancia necesaria para determinar si los dedos se encuentran “Tocando” o “NoTocando”, se analizó su comportamiento y se estableció que las unidades mínimas para diferenciar ese comportamiento comprenderían a un mínimo  $< 26$  unidades para indicar que los dedos están “Tocando”, de lo contrario estarían “NoTocando”.



Figura 61: Distancia para la pose de “Abierto” o “Doblado”.

De igual forma, para establecer si un dedo está “Abierto” o “Doblado” se analizó el comportamiento y se estableció que las unidades deberían de ser  $< 60$  unidades, para indicar que un dedo esta “Doblado”, en caso contrario sería “Abierto”.

Una vez categorizadas las distancias se almacenan en la base de datos, esto respecto a cada dedo en cuestión.

#### 4.5.2 Diseño de base de datos

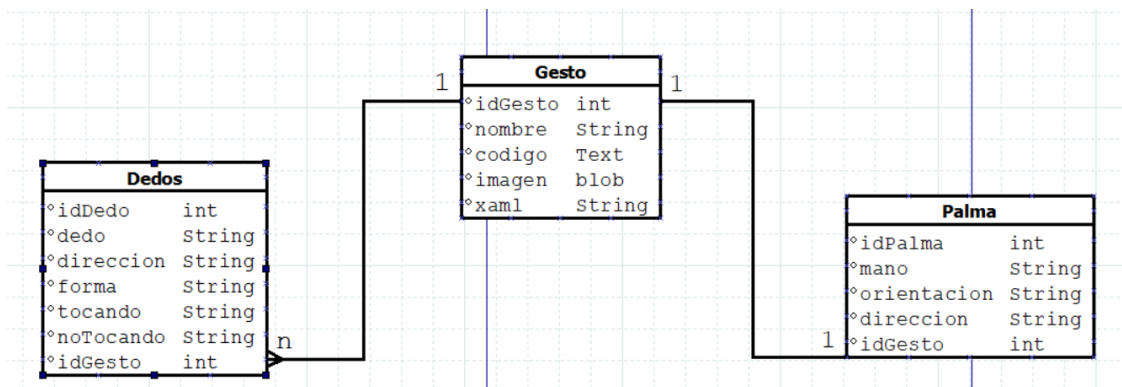


Figura 62: Base de datos para almacenar primitivas de los gestos



Como se puede ver en la Figura 62, la base de datos se conforma de 3 tablas. La tabla **Dedos** como su nombre lo indica, almacenará la información de cada dedo de la mano y el significado de cada campo es el siguiente: idDedo corresponde al identificador del dedo, donde dedo indicará el nombre del dedo a quien corresponda la información, dirección que dirección tiene el dedo, forma si el dedo se encuentra abierto o cerrado, tocando indicará con que dedos tiene contacto, noTocando indicará con qué dedos no está teniendo contacto e idGesto indica a qué gesto corresponde el dedo.

La tabla **Palma** hace referencia la información de la palma de la mano y estos son sus campos: idPalma es el identificador de la palma, mano indica de qué mano (izquierda o derecha) es la palma, orientación indica que orientación tiene la palma, dirección indica que dirección tiene la palma e idGesto indica a qué gesto corresponde la palma.

La tabla **Gesto** representa la unión de las tablas **Dedos** y **Palma**, también contando con algunos campos como: idGesto es el identificador del gesto, nombre indica el nombre o significado que tiene el gesto, código aquí se almacenará el código generado del gesto, imagen este campo contendrá una representación visual del gesto que fue capturado y XAML contendrá el código del gesto en formato XML para poder ser manipulado directamente desde código de programación

De esta forma, es que se llegue a la parte de “base de datos” en el proceso de la generación de código de esta tesis.

## 4.6. Generación de código

El siguiente paso es la *lectura de la base de datos*, la cual contiene las poses primitivas/simples para proceder a la generación de código.

### 4.6.1 Definición de gramática

De acuerdo al comportamiento de las gramáticas de la jerarquía de Chomsky se optó por el tipo 2 que son las de libre de contexto, ya que se trata de un lenguaje de programación y, por otra parte, la estructura de los fragmentos de código que se pretenden generar, sigue este patrón.

Se llevaron a cabo el análisis de dos tipos de gramáticas, una con un mayor número de restricciones, por lo cual significa que tiene un mayor número de producciones. Y otra con un número menor de restricciones. Ambas gramáticas tienen la siguiente forma. Educación 26.

$$G = (NT, T, Gesto, P) \quad (26)$$

La primera gramática tiene lo siguiente:

NT= (mano, orientación, dedos, dedo, direccion1, direccion2, direccion3, direccion4, direccion5, direccion6, reglas1, reglas2, reglas3, reglas4, reglas5, reglas6, reglas7, reglas8, reglas9, reglas10, reglas11, reglas12, condicion1, condicion2, condicion3, condicion4, condicion5, condicion6, pulgar1, pulgar2, pulgar3, flexión, conjuntoDedos, tipoflexión, distancia, relación)

T= (izquierda, derecha, abajo, arriba, adelante, atrás, meñique, anular, medio, índice, pulgar, tocando, notocando, abierto, cerrado)

P = Producciones (véase Figura 63)

```

<Gesto> ::= <mano> <orientacion> <dedos>
<mano> ::= [Izquierda|Derecha]
<orientacion> ::= [adelante <direccion1>|atras<direccion2>|arriba<direccion3>|abajo<direccion4>|derecha<direccion5>|izquierda<direccion6>]
<direccion1> ::= [arriba <reglas1> <pulgar1>|abajo <reglas1><pulgar1>|izquierda <reglas2> <pulgar2>|derecha <reglas2><pulgar2>]
<direccion2> ::= [arriba <reglas11><pulgar1>|abajo <reglas11><pulgar1>|izquierda<reglas12><pulgar2>|derecha<reglas12><pulgar2>]
<direccion3> ::= [izquierda<reglas3><pulgar3>|adelante<reglas4><pulgar1>|derecha<reglas3><pulgar3>]
<direccion4> ::= [izquierda<reglas5><pulgar3>|adelante<reglas6><pulgar1>|derecha<reglas5><pulgar3>]
<direccion5> ::= [arriba<reglas7><pulgar3>|adelante<reglas8> <pulgar2>|abajo<reglas7><pulgar3>]
<direccion6> ::= [arriba<reglas9><pulgar3>|adelante<reglas10><pulgar2>|abajo<reglas9><pulgar3>]
<reglas1> ::= [<condicion1><reglas1>|<condicion3><reglas1>|<condicion5><reglas1>]
<reglas2> ::= [<condicion1><reglas2>|<condicion2><reglas2>|<condicion4><reglas2>]
<reglas3> ::= [<condicion2><reglas3>|<condicion3><reglas3>|<condicion4><reglas3>]
<reglas4> ::= [<condicion1><reglas4>|<condicion3><reglas4>|<condicion6><reglas4>]
<reglas5> ::= [<condicion2><reglas5>|<condicion4><reglas5>|<condicion5><reglas5>]
<reglas6> ::= [<condicion1><reglas6>|<condicion5><reglas6>|<condicion6><reglas6>]
<reglas7> ::= [<condicion3><reglas7>|<condicion4><reglas7>|<condicion5><reglas7>]
<reglas8> ::= [<condicion1><reglas8>|<condicion4><reglas8>|<condicion6><reglas8>]
<reglas9> ::= [<condicion2><reglas9>|<condicion3><reglas9>|<condicion5><reglas9>]
<reglas10> ::= [<condicion1><reglas10>|<condicion2><reglas10>|<condicion6><reglas10>]
<reglas11> ::= [<condicion3><reglas11>|<condicion5><reglas11>|<condicion6><reglas11>]
<reglas12> ::= [<condicion2><reglas12>|<condicion4><reglas12>|<condicion6><reglas12>]
<condicion1> ::= <dedo> adelante <condicion1>
<condicion2> ::= <dedo> izquierda <condicion2>
<condicion3> ::= <dedo> arriba <condicion3>
<condicion4> ::= <dedo> derecha <condicion4>
<condicion5> ::= <dedo> abajo <condicion5>
<condicion6> ::= <dedo> atras <condicion6>
<pulgar1> ::= [pulgar derecha|pulgar izquierda]
<pulgar2> ::= [pulgar arriba| pulgar abajo]
<pulgar3> ::= [pulgar adelante|pulgar atras]
<dedo> ::= [meñique|anular|medio|indice|pulgar]
<dedos> ::= <flexion> <distancia>
<flexion> ::= <conjuntoDedos> <tipoflexion> <flexion>
<conjuntoDedos> ::= <dedo> <conjuntoDedos>
<tipoflexion> ::= [abierto|cerrado ]
<distancia> ::= <dedo><relacion> <dedo><distancia>
<relacion> ::= [tocando|no tocando]

```

Figura 63: Producciones de la primera gramática

La segunda gramática se conforma de la siguiente manera:

NT= (gesto, mano, palma, orientacion, direccion, dedos, posicion, flexion, tipoflexion, distancia, relacion, dedo, dedos)

T= (izquierda, derecha, abajo, arriba, adelante, atrás, meñique, anular, medio, indice, pulgar, tocando, noTocando, abierto, cerrado)

P = Producciones (véase Figura 64)

```

<Gesto> ::= <mano> <palma> <dedos>*
<mano> ::= (izquierda | derecha | alguna)
<palma> ::= <orientacion>? <direccion>?
<orientacion> ::= [adelante | atras | arriba | abajo | izquierda | derecha]
<direccion> ::= [arriba | abajo | izquierda | derecha| adelante]
<dedos> ::= <posicion> <interaccion>*
<posicion> ::= <dedo> <direccion>? <flexion>
<flexion> ::= [abierto | cerrado]
<interaccion> ::= <relacion> <dedo>*
<relacion> ::= [tocando | noTocando]
<dedo> ::= [meñique | anular | medio | indice | pulgar]

```

Figura 64: Producciones de la segunda gramática.

El uso de una gramática es para visualizar cómo está funcionando la estructura que se requiere para los fragmentos de código que se piensan generar para la detección de los gestos o poses de la mano. El usar una gramática muy extensa (véase Figura 63) ayuda a tener un mayor control en las restricciones de lo que se quiere generar de elementos terminales como resultado. Y el usar una gramática muy reducida (véase Figura 64) puede propiciar a valores que semánticamente no estarían correctos, pero a pesar de ello, aún se tiene la parte de programación para arreglar ciertas incongruencias o código extra que se podría llegar a generar

#### 4.6.2 Generación de código

Una vez que se tiene la estructura adecuada respecto a la gramática, se procede a la transformación de las poses simples/primitivas a código fuente (véase Tabla 10, Tabla 11, Tabla 12, Tabla 13, Tabla 14). Además de agregar ciertas palabras reservadas y llaves que requiere el mismo lenguaje de programación para reconocer los fragmentos de código.

Tabla 10: Relación de primitivas de Mano

Mano	
Izquierda	Hand.LeftHand
Derecha	Hand.RightHand

Tabla 11: Relación de primitivas de Orientación y Dirección

Orientación / Dirección	
Arriba	PoseDirection.Up
Abajo	PoseDirection.Down
Derecha	PoseDirection.Right
Izquierda	PoseDirection.Left
Adelante	PoseDirection.Forward
Atrás	PoseDirection.Backward

Tabla 12: Relación de primitivas de Dedo

Dedo	
Meñique	Finger.Pinky
Anular	Finger.Ring
Medio	Finger.Middle
Índice	Finger.Index
Pulgar	Finger.Thumb

Tabla 13: Relación de primitivas Forma

Forma	
Abierto	FingerFlexion.Open
Doblado	FingerFlexion.Folded

Tabla 14: Relación de primitivas Interacción

Interacción	
Tocando	RelativeDistance.Touching
NoTocando	RelativeDistance.NotTouching

Como último paso se genera el fragmento de código de cada gesto, el que es almacenado en la base de datos para cuando sea requerido y se puedan extraer todos los fragmentos de código sin necesidad de estarlos generando todo el tiempo.

## 4.7. Interfaz

Para la siguiente y última actividad de la fase 2, es asignarle un nombre al fragmento de código generado y almacenarlo en un archivo. Para esto se creó un sistema que funciona como *CRUD* (por sus siglas en inglés *Create, Read, Update, Delete*, véase Figura 65), en la cual se lleva a cabo todo el proceso para la generación de código, además de realizar cada una de las tareas de lectura, creación, modificación y eliminación.

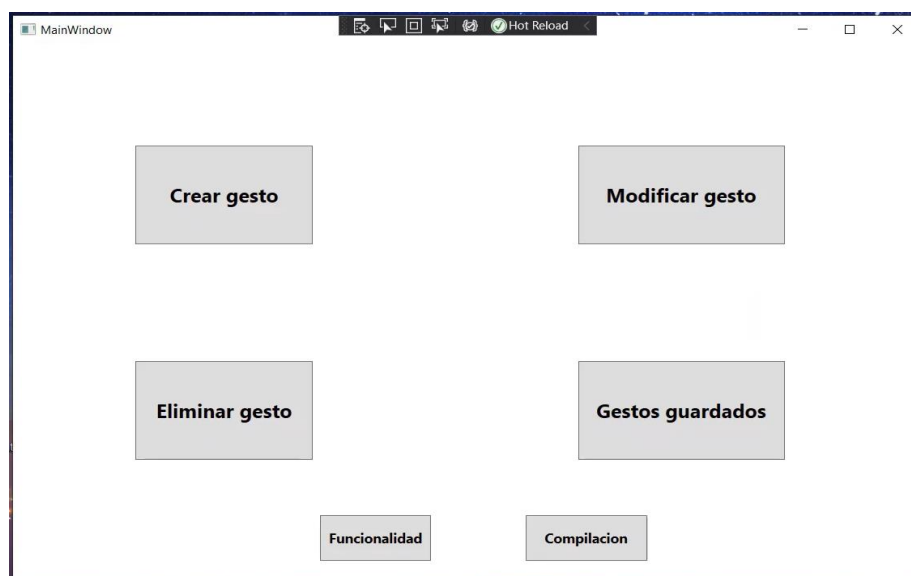


Figura 65: Sistema CRUD de gestos de mano.

En el módulo de *Crear gesto* (véase Figura 66Figura 65) después de que se presenta el gesto frente al sensor, existe un contador de tiempo para capturar el gesto. Una vez que el gesto se haya capturado, aparecerá una ventana, donde se le introducirá el nombre o etiqueta que se le quiere dar a dicho gesto. Después de aceptar, se procederá a almacenar las primitivas a la base de datos junto con su etiqueta o nombre del gesto.

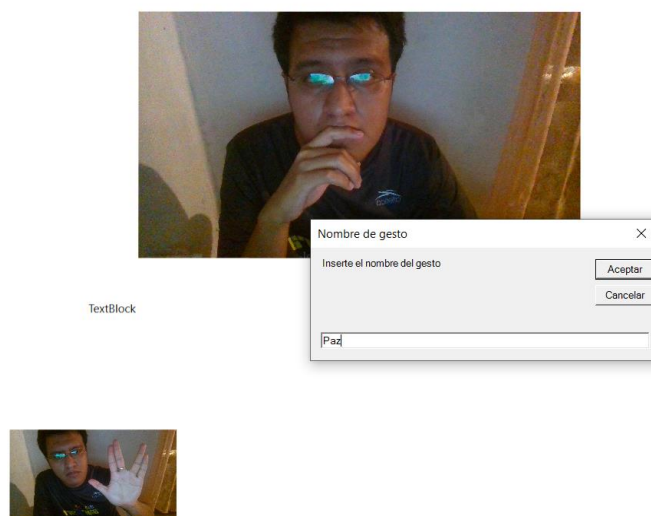


Figura 66: Módulo de interfaz para capturar gesto.

El código fuente se puede visualizar en el módulo de *Gestos guardados* (véase Figura 67) al igual que una captura del gesto.

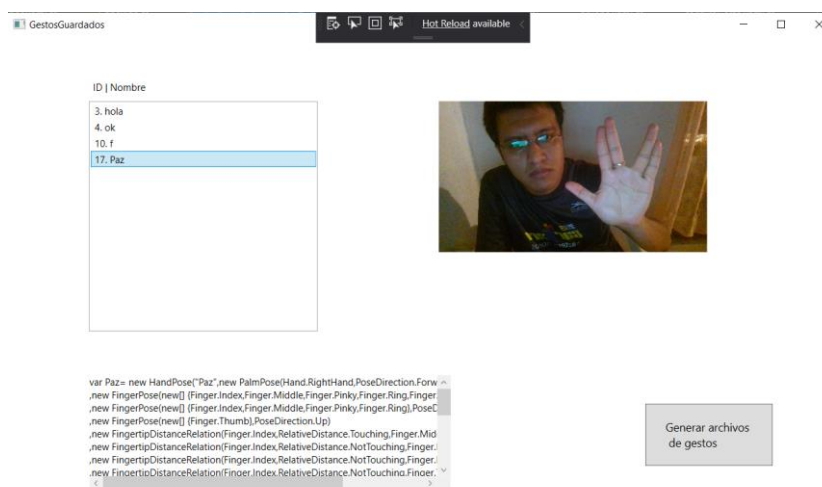


Figura 67: Módulo de interfaz: gestos guardados.

En el mismo módulo también se pueden generar dos archivos (véase Figura 68). El primer archivo corresponde a un archivo de texto plano (véase Figura 69) y el segundo a un archivo de tipo XML (véase Figura 70) para poder trabajar directamente con los gestos desde algún programa.

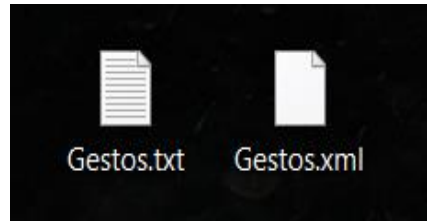


Figura 68: Archivos de gestos.

```
Gestos.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
var hola= new HandPose("hola",new PalmPose(Hand.RightHand,PoseDirection.Forward,PoseDirection.Up)
,new FingerPose(new[] {Finger.Index,Finger.Middle,Finger.Pinky,Finger.Ring,Finger.Thumb}, FingerFlexion.Open)
,new FingerPose(new[] {Finger.Index,Finger.Middle,Finger.Pinky,Finger.Ring,Finger.Thumb},PoseDirection.Up)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.NotTouching,Finger.Middle)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.NotTouching,Finger.Pinky)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.NotTouching,Finger.Ring)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.NotTouching,Finger.Thumb)
,new FingertipDistanceRelation(Finger.Middle,RelativeDistance.NotTouching,Finger.Pinky)
,new FingertipDistanceRelation(Finger.Middle,RelativeDistance.NotTouching,Finger.Ring)
,new FingertipDistanceRelation(Finger.Middle,RelativeDistance.NotTouching,Finger.Thumb)
,new FingertipDistanceRelation(Finger.Pinky,RelativeDistance.NotTouching,Finger.Ring)
,new FingertipDistanceRelation(Finger.Pinky,RelativeDistance.NotTouching,Finger.Thumb)
,new FingertipDistanceRelation(Finger.Ring,RelativeDistance.NotTouching,Finger.Thumb)

var ok= new HandPose("ok",new PalmPose(Hand.RightHand,PoseDirection.Left,PoseDirection.Up)
,new FingerPose(new[] {Finger.Index,Finger.Middle,Finger.Pinky,Finger.Ring,Finger.Thumb}, FingerFlexion.Open)
,new FingerPose(new[] {Finger.Index,Finger.Thumb},PoseDirection.Left)
,new FingerPose(new[] {Finger.Middle,Finger.Pinky,Finger.Ring},PoseDirection.Up)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.Touching,Finger.Thumb)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.NotTouching,Finger.Middle)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.NotTouching,Finger.Pinky)
,new FingertipDistanceRelation(Finger.Index,RelativeDistance.NotTouching,Finger.Ring)
,new FingertipDistanceRelation(Finger.Middle,RelativeDistance.NotTouching,Finger.Pinky)
,new FingertipDistanceRelation(Finger.Middle,RelativeDistance.NotTouching,Finger.Ring)
,new FingertipDistanceRelation(Finger.Middle,RelativeDistance.NotTouching,Finger.Thumb)
,new FingertipDistanceRelation(Finger.Pinky,RelativeDistance.NotTouching,Finger.Ring)
,new FingertipDistanceRelation(Finger.Pinky,RelativeDistance.NotTouching,Finger.Thumb)
,new FingertipDistanceRelation(Finger.Ring,RelativeDistance.NotTouching,Finger.Thumb)
```

Figura 69: Archivo de texto simple de gestos.

```
<?xml version="1.0"?>
- <ArrayList xml:space="preserve" xmlns:ms="clr-namespace:System;assembly=mscorlib" xmlns:cl="clr-namespace:System.Collections;assembly=mscorlib" Capacity="4">
  <s:String><HandPose Name = "hola" xmlns="http://schemas.microsoft.com/gestures/2015/xaml" > <PalmPose Context="RightHand" Direction="Forward" Orientation="Up"/> <FingerPose
Context="Index,Middle,Pinky,Ring,Thumb" Flexion="Open"/> <FingerPose Context="Index,Middle,Pinky,Ring,Thumb" Direction="Up"/> <FingertipDistanceRelation Context="Index
DistanceRelation="NotTouching" OtherContext="Middle"/> <FingertipDistanceRelation Context="Index" DistanceRelation="NotTouching" OtherContext="Pinky"/> <FingertipDistanceRelation
Context="Index" DistanceRelation="NotTouching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Index" DistanceRelation="NotTouching" OtherContext="Thumb"/>
<FingertipDistanceRelation Context="Middle" DistanceRelation="NotTouching" OtherContext="Pinky"/> <FingertipDistanceRelation Context="Middle" DistanceRelation="NotTouching"
OtherContext="Ring"/> <FingertipDistanceRelation Context="Middle" DistanceRelation="NotTouching" OtherContext="Thumb"/> <FingertipDistanceRelation Context="Pinky"
DistanceRelation="NotTouching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Pinky" DistanceRelation="NotTouching" OtherContext="Thumb"/> <FingertipDistanceRelation
Context="Ring" DistanceRelation="NotTouching" OtherContext="Thumb"/> </HandPose></s:String>
<s:String><HandPose Name = "ok" xmlns="http://schemas.microsoft.com/gestures/2015/xaml" > <PalmPose Context="RightHand" Direction="Left" Orientation="Up"/> <FingerPose
Context="Index,Middle,Pinky,Ring,Thumb" Flexion="Open"/> <FingerPose Context="Index,Thumb" Direction="Left"/> <FingerPose Context="Middle,Pinky,Ring" Direction="Up"/>
<FingertipDistanceRelation Context="Index" DistanceRelation="Touching" OtherContext="Thumb"/> <FingertipDistanceRelation Context="Index" DistanceRelation="NotTouching"
OtherContext="Middle"/> <FingertipDistanceRelation Context="Index" DistanceRelation="NotTouching" OtherContext="Pinky"/> <FingertipDistanceRelation Context="Index"
DistanceRelation="NotTouching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Middle" DistanceRelation="NotTouching" OtherContext="Pinky"/> <FingertipDistanceRelation
Context="Middle" DistanceRelation="NotTouching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Middle" DistanceRelation="NotTouching" OtherContext="Thumb"/>
<FingertipDistanceRelation Context="Pinky" DistanceRelation="NotTouching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Pinky" DistanceRelation="NotTouching"
OtherContext="Thumb"/> <FingertipDistanceRelation Context="Ring" DistanceRelation="NotTouching" OtherContext="Thumb"/> </HandPose></s:String>
<s:String><HandPose Name = "f" xmlns="http://schemas.microsoft.com/gestures/2015/xaml" > <PalmPose Context="RightHand" Direction="Left" Orientation="Up"/> <FingerPose
Context="Index,Middle,Pinky,Ring,Thumb" Flexion="Open"/> <FingerPose Context="Index" Direction="Left"/> <FingerPose Context="Middle,Pinky,Ring,Thumb" Direction="Up"/>
<FingertipDistanceRelation Context="Middle" DistanceRelation="Touching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Index" DistanceRelation="NotTouching"
OtherContext="Middle"/> <FingertipDistanceRelation Context="Index" DistanceRelation="NotTouching" OtherContext="Pinky"/> <FingertipDistanceRelation Context="Index"
DistanceRelation="NotTouching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Middle" DistanceRelation="NotTouching" OtherContext="Pinky"/> <FingertipDistanceRelation
Context="Middle" DistanceRelation="NotTouching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Middle" DistanceRelation="NotTouching" OtherContext="Thumb"/>
<FingertipDistanceRelation Context="Pinky" DistanceRelation="NotTouching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Pinky" DistanceRelation="NotTouching"
OtherContext="Thumb"/> <FingertipDistanceRelation Context="Ring" DistanceRelation="NotTouching" OtherContext="Thumb"/> </HandPose></s:String>
<s:String><HandPose Name = "Paz" xmlns="http://schemas.microsoft.com/gestures/2015/xaml" > <PalmPose Context="RightHand" Direction="Forward" Orientation="Up"/> <FingerPose
Context="Index,Middle,Pinky,Ring,Thumb" Flexion="Open"/> <FingerPose Context="Index,Middle,Pinky,Ring" Direction="Up"/> <FingerPose Context="Thumb" Direction="Up"/>
<FingertipDistanceRelation Context="Index" DistanceRelation="Touching" OtherContext="Middle"/> <FingertipDistanceRelation Context="Index" DistanceRelation="NotTouching"
OtherContext="Pinky"/> <FingertipDistanceRelation Context="Index" DistanceRelation="NotTouching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Index"
DistanceRelation="NotTouching" OtherContext="Thumb"/> <FingertipDistanceRelation Context="Middle" DistanceRelation="NotTouching" OtherContext="Pinky"/> <FingertipDistanceRelation
Context="Middle" DistanceRelation="NotTouching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Middle" DistanceRelation="NotTouching" OtherContext="Thumb"/>
<FingertipDistanceRelation Context="Pinky" DistanceRelation="NotTouching" OtherContext="Ring"/> <FingertipDistanceRelation Context="Pinky" DistanceRelation="NotTouching"
OtherContext="Thumb"/> <FingertipDistanceRelation Context="Ring" DistanceRelation="NotTouching" OtherContext="Thumb"/> </HandPose></s:String>
</ArrayList>
```

Figura 70: Archivo XML de gestos.

Los módulos restantes corresponden a las tareas de *Modificar gesto* y *Eliminar gesto* almacenados en la base de datos. En el módulo de *Eliminar gesto* (véase Figura 71) se tienen los gestos enlistados, cada uno cuenta con un *Checkbox* para marcar qué gestos se van a eliminar y se muestra una captura de dicho gesto, para proceder a eliminar, se usa el botón de la parte inferior derecha, el cual desplegará una ventana de emergencia para verificar que se esté seguro de ejecutar la tarea.

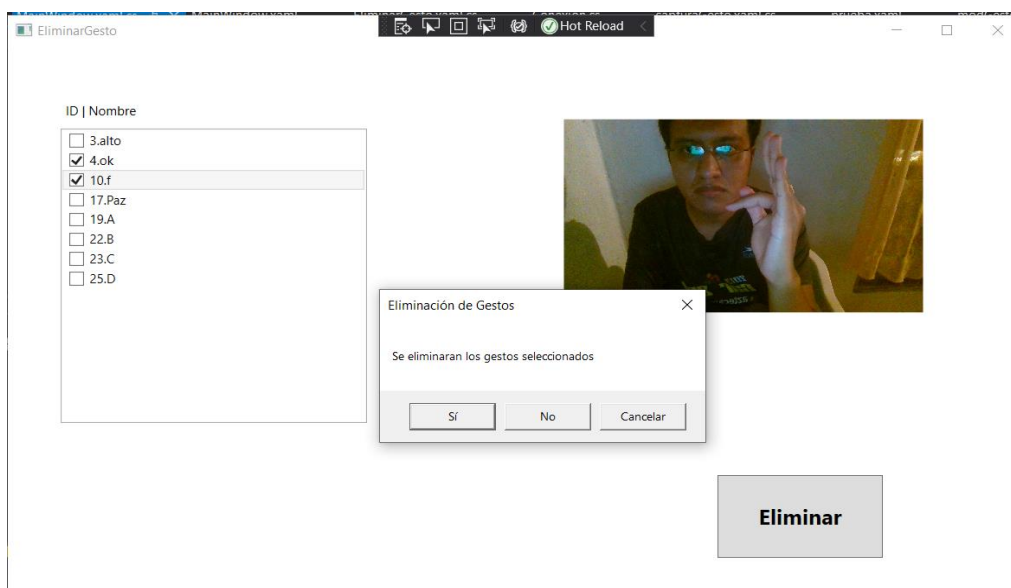


Figura 71: Módulo de interfaz: eliminar gesto.

En el otro módulo que es *Modificar gesto* (véase Figura 72), se llevará a cabo la tarea de modificar tanto las características del gesto (código funcional), tales como el nombre. De igual forma como lo es en el módulo de eliminar gesto; en este módulo se podrá visualizar una lista de los gestos almacenados, así como la imagen del gesto que se seleccione.

En el módulo se puede cambiar de manera rápida el nombre de un gesto, solo basta con escribir el nuevo nombre que se desea en el cuadro de texto, y proceder a presionar el botón “cambiar nombre”, para esto se tendrá un cuadro de advertencia para asegurar que el usuario desea cambiar el nombre además de que se le notificará cual es el nombre actual del gesto y por cual nombre será reemplazado. Una vez confirmada la tarea, se actualizarán los datos y se mostrará el nombre ya actualizado.

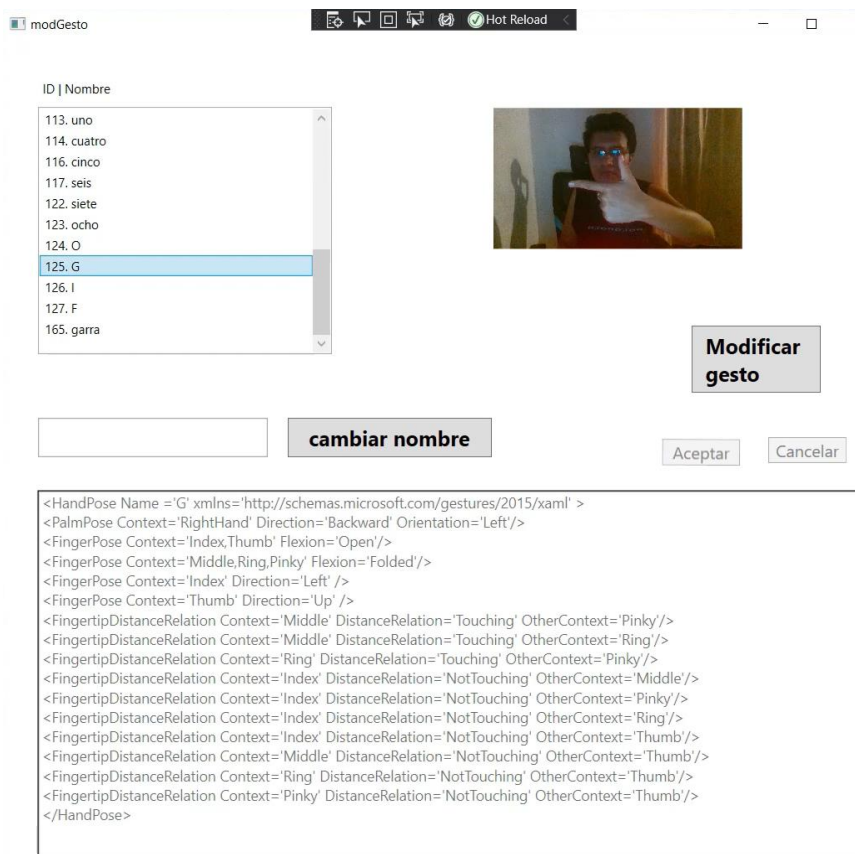


Figura 72: Módulo de interfaz: modificar gesto.

Por otra parte, para la modificación del gesto, se hace uso del segundo botón, el cual activa el panel donde se encuentra el código funcional y los botones de aceptar o cancelar la operación.

Con esto se estaría completando la fase 2, donde se lleva a cabo la parte más importante de la tesis, dando paso a la fase 3 que comprende el área de pruebas y resultados, para medir el porcentaje de precisión de no poseer errores de compilación los fragmentos de código generados por el sistema.



# Capítulo V

## Pruebas y resultados

## 5. Pruebas y resultados

### 5.1. Pruebas

#### 5.1.1 Defectos de escritura

Se desarrolló una interfaz la cual es simple y que consta de dos áreas, una para visualizar el contenido del archivo de texto plano y la segunda área mostrará el resultado de la compilación. Se agrega un botón ubicado en la parte central inferior de la interfaz para ejecutar la tarea de compilación. (véase Figura 73)

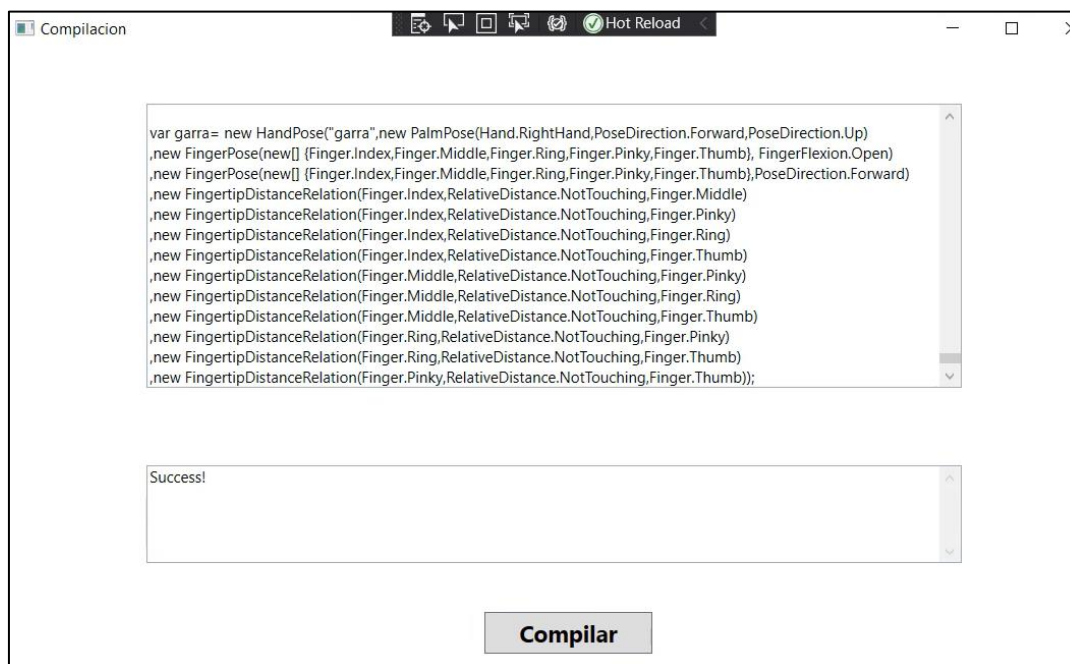


Figura 73: Compilador de código de gestos.

Al momento de abrir el compilador se hizo uso de uno de los archivos generados, específicamente el de texto plano. El compilador de códigos fue colocado en el mismo sistema CRUD del generador de código para facilitar su uso. Además, fueron incluidas líneas de código extra como la misma librería de *Project Gesture*, llaves, palabras reservadas y estructuras de clases propias del lenguaje C# para evitar defectos de escritura ajenos a los fragmentos de código de los gestos.

Para poner a prueba la presencia de errores de escritura de código de gestos se usó el diccionario de lenguaje de señas mexicana manos con voz (Esther M. & González R. 2011), el cual contiene señas para interpretar cierto número de palabras mexicanas. Como el sistema sólo es capaz de capturar poses estáticas, las señas que fueron tomadas en cuenta de este diccionario corresponden al grupo de:

- Números (véase Figura 74)
- Alfabeto (véase Figura 75)

El grupo de *Alfabeto* contiene letras que requieren movimiento, así que esas letras no fueron tomadas en cuenta. A continuación, se muestran algunos ejemplos de los gestos que contienen estos dos grupos.



Figura 74: Grupo de señas de números (Esther M. & González R. ,2011).

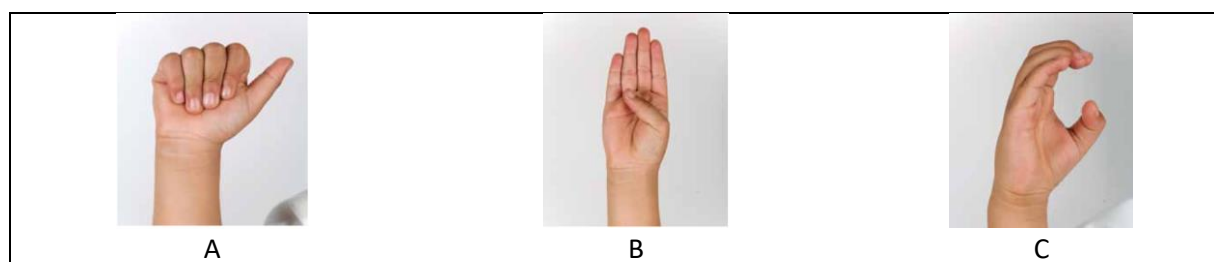


Figura 75: Grupo de señas del Alfabeto (Esther M. & González R. ,2011).

Los resultados de la verificación de errores de escritura de programación se muestran a continuación (véase Figura 76)

Como se puede observar, en todos los casos, la compilación de los fragmentos de código de programación fue correcta, sin ningún tipo de error de escritura, logrando así el objetivo específico 4 en un 100% libre de errores de escritura.

Gesto	Compilación	Errores
A	Correcta	0
B	Correcta	0
C	Correcta	0
D	Correcta	0
E	Correcta	0
F	Correcta	0
G	Correcta	0
H	Correcta	0
I	Correcta	0
L	Correcta	0
M	Correcta	0
N	Correcta	0
O	Correcta	0
P	Correcta	0
S	Correcta	0
T	Correcta	0
U	Correcta	0
V	Correcta	0
W	Correcta	0
Y	Correcta	0

Gesto	Compilación	Errores
1	Correcta	0
2	Correcta	0
3	Correcta	0
4	Correcta	0
5	Correcta	0
6	Correcta	0
7	Correcta	0
8	Correcta	0

Figura 76: Resultado de generación de código.

### 5.1.2 Funcionalidad de fragmentos de código en *Project Gesture*

Se creó una interfaz simple para probar la funcionalidad de los fragmentos generados de código, la cual tiene un área de texto rectangular. Esta interfaz al ser iniciada hace uso del segundo archivo generado, en este caso del archivo tipo XML, para registrar los fragmentos de código generados por los gestos realizados, y estar a la espera de reconocer alguno de esos gestos mediante el sensor de profundidad. Al reconocer algún gesto se ejecutará la tarea de escribir el nombre de dicho gesto en el área rectangular que se encuentra en la interfaz (véase Figura 78).

Para hacer el reconocimiento del gesto, en este caso, se requirió que primero se realice el gesto correspondiente y enseguida se cierre la mano, para así activar la tarea de escribir su nombre en el área de texto (véase Figura 77).

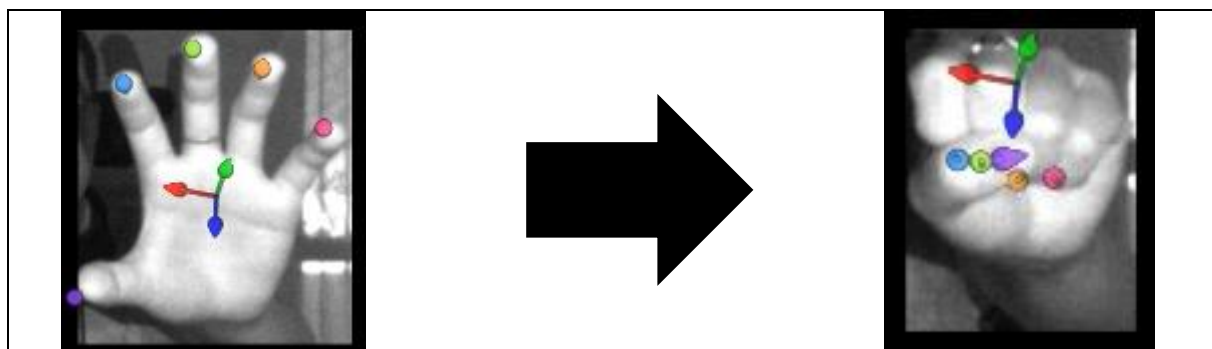


Figura 77: Activación del código del gesto

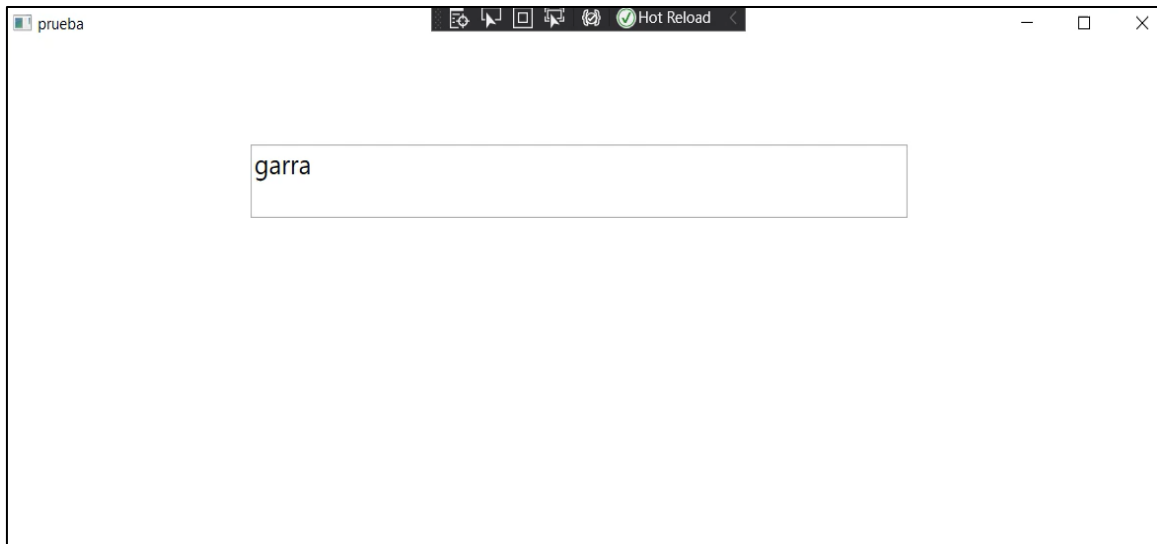


Figura 78: Verificación de funcionalidad de código.

Ya con esto establecido se procedió a realizar la medición del sistema respecto a la detección de los gestos.

## 5.2. Resultados

### 5.2.1 Métricas

Como principal métrica de evaluación para la detección de los gestos, se eligió elaborar una **matriz de confusión** (véase *Tabla 15*) debido a que a partir de ella se pueden fácilmente obtener otras métricas como lo son: **exactitud**, **precisión**, **sensibilidad**, entre otras. Además, la matriz de confusión permite tener una visualización grafica del comportamiento que están presentando los gestos y con cual o cuales el sistema está confundiendo su reconocimiento. Para esto se pusieron a prueba 18 gestos más la opción de los gestos que no fueron capturados para su detección llevando como nombre “No se detecta” y el resto de los gestos son parte del diccionario del lenguaje de señas mexicana “*manos con voz*” (Esther M. & González R., 2011) específicamente el abecedario, donde el gesto fuera estático, es decir, no necesita tener un movimiento para dar interpretación a la letra.

Tabla 15: Matriz de confusión.

Gesto	V A L O R P R E D I C H O																		
	A	B	C	D	E	F	G	H	L	M	N	O	P	R	T	U	V	W	No se detecta
A	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
D	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	3
F	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0

R E A L	L	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
	M	0	0	0	0	0	0	0	0	0	6	4	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0
	O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
	P	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	1	0
	T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	4
	V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0
	No se detecta	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10

De acuerdo a la matriz de confusión se puede observar el comportamiento que está teniendo el sistema al detectar ciertos gestos, como lo es en el caso del gesto “M” con “N” esto se debe a la alta similitud que tienen estos dos gestos y al tratarse de un sistema en tiempo real, al momento de realizar la segunda pose (cerrar la mano) para activar el gesto pasa a detectar ambos gestos al mismo tiempo (M y N). En caso contrario, se puede observar que con los gestos “R” y “O”, estos no son reconocidos una sola vez. Durante el proceso de trabajar con el sensor de profundidad, se notó que el tener los dedos ocultos o muy juntos se le dificultaba al sensor reconocer su ubicación correctamente, dando como resultado valores erróneos en ciertos casos. Para los casos que pertenecen a los “No se detecta” se debe a que los gestos requieren estrictamente cierta colocación de los dedos y palma de la mano, para ser reconocidos exitosamente y esto se dificulta con ciertos gestos que demandan una postura más compleja.

Tabla 16: Métricas: precision, accuracy y recall

Gesto	Precision	Accuracy	Recall
A	1	1	1
B	1	1	1
C	1	0.98	0.8
D	1	1	1
E	1	0.96	0.7
F	1	1	1
G	1	1	1
H	1	1	1
L	1	1	1
M	1	0.97	0.6
N	0.71	0.97	1
O		0.94	
P	1	1	1
R		0.94	
T		0.94	
U	0.4	0.93	0.6
V	1	0.99	1
W	1	1	1
No registrado	0.25	0.84	1
Total:	0.75	0.97	0.77

Para las siguientes métricas (véase *Tabla 16*), comúnmente no se recomienda hacer uso de la métrica *exactitud* si el conjunto de datos no se encuentra balanceados, pero en este caso se decidió que cada gesto fuera ejecutado 10 veces, dándonos así un total del 0.97% de elementos clasificados correctamente. Un total del 0.75% de *precisión* en los elementos identificados correctamente y un 0.77% de *sensibilidad* que nos indica el porcentaje de los elementos identificados correctamente como positivos del total de positivos verdaderos.

### 5.2.2 Pruebas con otros usuarios

Se tenía contemplado llevar a cabo las pruebas con un grupo de personas que trabajan con el uso del lenguaje de señas mexicanas, pero debido a la pandemia por el virus que causa la COVID-19, el grupo de personas decidieron suspender sus labores. Debido a esto, las pruebas se realizaron aun tomando como base de referencia el diccionario de lengua de señas mexicana “*manos con voz*” (Esther M. & González R. ,2011) pero con personas con nulo conocimiento y práctica de esta lengua.

Las pruebas se realizaron con un total de 5 sujetos, tomando como base de referencia a las poses de mano que se usan en el diccionario de lenguaje de señas mexicana *Manos con voz*, específicamente al grupo de abecedario y números. No se incluyó a todas las letras del abecedario y a todos los números, ya que algunos dependen el hacer ciertos movimientos para expresar su significado. Por otra parte, aunque no se tenían en cuenta, se tomaron ciertos gestos de mano que poseen dedos ocultos, con el fin de observar que tantas correcciones deberían aplicarse para que el código generado tuviera un desempeño favorable, haciendo una ejecución de 10 veces por cada gesto. Para esto, todas las pruebas se llevaron a cabo en una habitación con poca iluminación directa del sol, debido a las especificaciones funcionales del sensor de profundidad. (véase *Figura 79*)

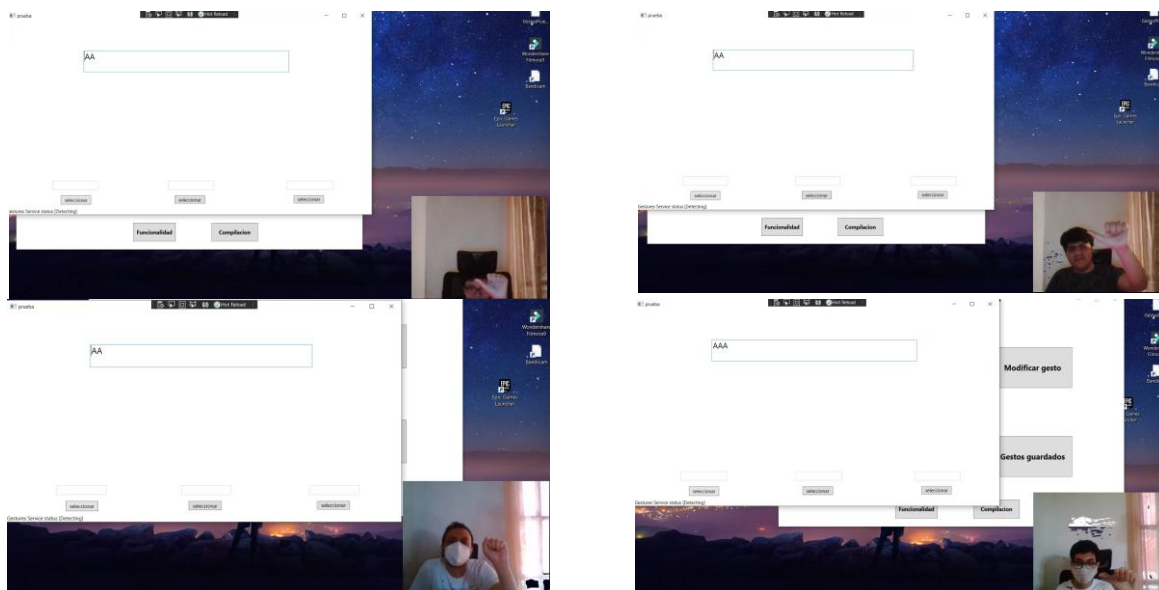


Figura 79: Usuarios de prueba.

El éxito o fracaso de que los fragmentos de código funcionen perfectamente, recae en la facilidad que los usuarios tengan al realizar ciertos gestos/poses de mano, además, de las condiciones del entorno y en el mismo sensor de profundidad, ya que, al no ser un producto perfecto en su

totalidad, puede presentar fallas a la hora de capturar la información, presentando (aunque no muchos) fragmentos de código con poses simples fuera de lo normal (hablando de que los dedos los interpretaría con una posición difíciles de realizar).

Teniendo en cuenta lo anterior mencionado, los resultados de las pruebas con otros usuarios fueron variadas (véase *Tabla 17, Tabla 18, Tabla 19*).

**Tabla 17: Resultados de usuario 1 y 2.**

Gesto	Precision	Accuracy	Recall
A	1	1	1
B	1	0.98	0.7
C	1	0.98	0.8
D	1	0.99	0.9
E		0.94	
F	1	0.98	0.8
G	1	1	1
H	1	0.98	0.8
L	1	1	1
M	1	0.96	0.3
N	0.9	0.98	0.9
O		0.94	
P	0.8	0.96	0.4
R		0.94	
T		0.94	
U	1	0.99	0.9
V	0.9	0.99	1
W	1	1	1
No registrado	0.13	0.66	1
Total:	0.72	0.95	0.65

Gesto	Precision	Accuracy	Recall
A	1	1	1
B	1	0.99	0.9
C	1	0.98	0.7
D	1	0.99	0.9
E	1	0.96	0.4
F	1	0.98	0.8
G	1	1	1
H	1	1	1
L	1	1	1
M	1	0.97	0.6
N	0.90	0.99	1
O			
P	1	0.99	0.9
R		0.94	
T		0.94	
U	1	1	1
V	1	0.98	0.8
W	1	0.98	0.8
No registrado	0.16	0.73	1
Total:	0.79	0.86	0.72

**Tabla 18: Resultados de usuarios 3 y 4.**

Gesto	Precision	Accuracy	Recall
A	1	1	1
B	1	0.99	0.9
C	1	0.98	0.7
D	1	0.97	0.6
E		0.94	
F	1	0.98	0.7
G	1	1	1
H	1	1	1
L	1	0.99	0.9
M	1	0.98	0.8
N	0.77	0.97	0.7
O	1	0.95	0.2
P	1	0.95	0.2
R		0.94	
T		0.94	
U	1	1	1
V	1	1	1
W	1	1	1
No registrado	0.14	0.67	1
Total:	0.78	0.96	0.66

Gesto	Precision	Accuracy	Recall
A	1	1	1
B	1	0.98	0.8
C	1	0.95	0.2
D	1	0.98	0.7
E		0.94	
F	1	0.98	0.7
G	1	0.99	0.9
H	1	0.99	0.9
L	1	1	1
M	1	0.96	0.3
N	0.83	0.98	1
O		0.94	
P	1	0.98	0.7
R		0.94	
T		0.94	
U	1	1	1
V	1	1	1
W	1	1	1
No registrado	0.17	0.75	1
Total:	0.73	0.96	0.64



*Tabla 19: Resultado final.*

Precision	Accuracy	Recall
0.75	0.93	0.66

Los usuarios por falta de práctica con poses/gestos de la mano, se les dificultó realizar algunos gestos, tomándose tedioso colocar los dedos en cierta postura específica para que algunos gestos fueran reconocidos con éxito. Por otra parte, los gestos “E”, “R” y “I” no fueron reconocidos debido a que el sensor se le es difícil capturar gestos con dedos ocultos o demasiado juntos. Aun así, se logró alcanzar un 93% de *accuracy* como resultado final entre los 4 usuarios de prueba. Un 75% de *precisión* de elementos identificados correctamente. Y un 66% de *recall* que en este caso nos indica que los gestos deberán ser ejecutados correctamente por los usuarios para poder ser detectados por el sistema de manera exitosa.

# Capítulo VII

## Conclusiones y trabajos futuros

## 6. Conclusiones

De acuerdo a lo presentado en este trabajo se va observando el cumplimiento de los objetivos específicos mencionados en un inicio los cuales son:

- OE-1: Definir un procedimiento que genere código automáticamente a partir de cada posición de las manos. Ubicado a partir del capítulo 4.5
- OE-2: Realizar un sistema que permita identificar las posiciones de las manos. Ubicado en el capítulo 4.7
- OE-3: Crear un archivo en el cual se almacenará el código que se genere por cada pose de mano. Ubicado en el capítulo 4.7
- OE-4: Verificar que el código obtenido no presente defectos de escritura de programación. Ubicado en el capítulo 5.1.1

En el objetivo 1 (OE – 1) se definió un proceso que permite generar código a partir de la obtención de una pose/gesto de mano. El gesto se captura mediante una cámara que cuenta con un sensor de profundidad, el sensor proporciona datos de lectura, algunos datos pasan por dos sistemas difusos y a otros se les aplica la ecuación de distancia euclidiana para un espacio tridimensional. Después se obtienen valores cualitativos que describen el comportamiento de la pose/gesto de mano, que son almacenados en una base de datos, a continuación, se utiliza una gramática de libre contexto para darle estructura a los valores de la base de datos, y posteriormente se transforman los valores cualitativos en fragmentos de código. Por lo tanto, se observó que el uso de la lógica difusa puede implementarse en este tipo de trabajos para obtener los resultados esperados respecto a la información que el sensor de profundidad arrojó en cada captura de las manos. Que además el uso de gramáticas libres de contexto sirve de manera correcta cuando se requiere tratar con lenguajes de programación.

El objetivo 2 (OE – 2) consta del desarrollo de un sistema de software que permite llevar a cabo las tareas para la creación de fragmentos de código de gestos de mano mediante su captura por un sensor de profundidad *RealSense SR300*. De esta manera, se logra implementar la visión que está teniendo el sensor, a pesar de no poder mostrar los puntos infrarrojos, se apoya de la cámara para tener notificado al usuario de que el sensor se encuentra en funcionamiento, capturando los movimientos.

El objetivo 3 (OE - 3) mediante el sistema de software desarrollado en el objetivo 2, se logra crear el archivo que contendrá todos los fragmentos de código generados que fueron almacenados en la base de datos. En este punto, además de crear un archivo de texto plano se logra crear un archivo en formato XML, el cual facilita su uso directamente para ser utilizado en alguna interfaz natural de usuario desarrollada con la librería Project Gesture, y así ejecutar los gestos de ese archivo.

El objetivo 4 (OE - 4) se desarrolló una pequeña interfaz donde se aplica el compilador del lenguaje *C#*, para realizar la lectura del archivo contenedor de fragmentos de código, que, mediante la agregación de palabras reservadas, llaves y librerías del mismo lenguaje se ponen a prueba los fragmentos de código en busca de defectos de programación, logrando una nula presencia de defectos en la compilación.

Con el cumplimiento de estos cuatro objetivos específicos, se logra completar de manera general lo propuesto en un inicio, el cual es: Detectar los movimientos de la mano derecha y generar el código fuente correspondiente a las posiciones de esa mano, reduciendo los defectos en la etapa de su programación.

## **6.1. Trabajos futuros**

A continuación, se muestran los trabajos futuros relacionados con este trabajo de investigación.

- Diseñar, desarrollar y probar sistemas que definan gestos con movimiento
- Investigar nuevas tecnologías para el desarrollo de sistemas de reconocimiento de gestos con mayor dificultad de detección.
- Investigar la posibilidad de crear un sistema de reconocimientos de gestos compatible con las tecnologías de almacenamiento en la nube donde solo se requiera disponer de una cámara.
- Desarrollar sistemas capaces de reconocer en tiempo de real las dos manos para la manipulación de entornos tridimensionales.
- Investigar nuevas tecnologías para desarrollar sistemas de reconocimiento de gestos que no requieran tener un dispositivo en frente para ser detectados.
- Investigar la posibilidad de unir el reconocimiento de los gestos de la mano con gestos faciales en un mismo sensor de profundidad.

## Bibliografía

## 7. Bibliografía

- Alexander, M. & Abid, K. (2013). Template Matching. OpenCV-Python-Tutorials. Obtenido de: [https://opencv-python-tutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_imgproc/py\\_template\\_matching/py\\_template\\_matching.html](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html)
- Amanquah, N., S. N. (2017). Code Generation on Mobile Devices for Mobile Apps. "Communications of the IIMA: Vol. 15: Iss. 2, Article 2.
- Andrew, R., R. C. (2017). Teleoperation of a Concentric Tube Robot through Hand Gesture Visual Tracking. Brisbane, Australia: IEEE.
- Battaglia, N. (2019). Integración de una herramienta CASE en un entorno académico colaborativo para la enseñanza de ingeniería de software. Universidad Abierta Interamericana. Obtenido de: <http://200.32.31.164:9999/ojs/index.php/RAIA/article/view/206>
- Barrios, J. (2019). La matriz de confusión y sus métricas. Health BIG DATA. Obtenido de: <https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/>
- Bellino, A. et al. (2016). Touch&Screen: Widget Collection for Large Screens Controlled through Smartphone. MUM, 11.
- Ceruzzi, P. (2018). Breve historia de la computación. Ciudad de México, México: Fondo de cultura Económica
- Cheon, Y. and Barua, A. (2017). Sudoku App: Model-Driven Development of Android Apps Using OCL? Departmental. Technical Reports (CS). 1183.
- Clarke, C. et al. (2016). TraceMatch: a Computer Vision Technique for User Input by Tracing of Animated Controls. UbiComp, 6.
- Cook, S. (2013). Libraries and SDK. 10.1016/B978-0-12-415933-4.00010-7.
- Dardan, M., A. M. (2016). Application Interface for Gesture Recognition with Kinect Sensor. Thessaloniki, Greece: IEEE International Conference on Knowledge Engineering and Applications.
- Deshpande, A. et al (2019). Finger Gesture recognition using laser line generator and camera. International research journal of engineering and technology. International Research Journal of Engineering and Technology (IRJET). e-ISSN: 2395-0056. p-ISSN: 2395-0072. Volume: 06 Issue: 04.
- EcuRed (05 de febrero del 2021). Distancia euclídea. ECURED. Obtenido de: [https://www.ecured.cu/Distancia\\_eucl%C3%ADdea](https://www.ecured.cu/Distancia_eucl%C3%ADdea)
- Estela, M. (2020). Concepto de Gramatica. Conceptode. Enlace: <https://concepto.de/gramatica/>
- Esther, M. & González R. (2011). Manos con voz: Diccionario de lenguaje de señas mexicana. 01780 México, D.F.: Libre Acceso, A.C., 11590 México, D.F.: Consejo Nacional para Prevenir la Discriminación
- Ensari, T. et al. (2019). Automatic HTML Code Generation from Mock-Up Images Using Machine Learning Techniques. 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), 2019, pp. 1-4, doi: 10.1109/EBBT.2019.8741736.
- Feng, Y., et al. (2014). MagicWatch: interacting & segueing. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication

- (UbiComp '14 Adjunct). Association for Computing Machinery, New York, NY, USA, 315–318. DOI: <https://doi.org/10.1145/2638728.2638848>
- Freyja (2017). How much could software errors be costing your company? RAYGUN. Obtenido de: <https://raygun.com/blog/cost-of-software-errors/>
  - Gallego, A. (2008). La jerarquía de Chomsky y la facultad del lenguaje: consecuencias para la variación y la evolución. Vol. XXVII/2, 2008, pp. 47-60 ISSN: 0210-1602
  - Giang, T. (2017). Intel RealSense SR300. GrabCad Community. Obtenido de: <https://grabcad.com/library/intel-realsense-sr300-3>
  - Gregoire, M. (2018). Writing Generic Code with Templates. 10.1002/9781119421276.ch12.
  - García, R. et al. (2017). Diseño de una estrategia de control difuso aplicada al proceso de ultracongelación de alimentos. Ingeniare. Rev. chil. ing. vol.25 no.1 Arica ene. 2017. Obtenido de: [https://scielo.conicyt.cl/scielo.php?pid=S0718-33052017000100070&script=sci\\_arttext](https://scielo.conicyt.cl/scielo.php?pid=S0718-33052017000100070&script=sci_arttext)
  - Hao, L. Negulescu, M. y Li, Y. (2014). Gesturemote: Interacting with Remote Displays through Touch Gestures. AVI, 4.
  - Humphrey, W. (2000). The Personal Software Process (PSP). Obtenido de Software Engineering Institute: <https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5283>
  - IEEE (2008). IEEE Standard for Software and System Test Documentation. IEEE Computer Society Sponsored by the Software & Systems Engineering Standards Committee. IEEE Std 829™-2008.
  - Iftekhhar, N. et al. (2019). Reverse Engineering of Relational Database Schema to UML Model. IEEE. 1-6
  - IONOS. (2019). Regex o expresiones regulares: la manera más sencilla de describir secuencias de caracteres. Digital Guide IONOS by 1&1. Obtenido de: <https://www.ionos.mx/digitalguide/paginas-web/creacion-de-paginas-web/regex/>
  - Intel. (2020). Cámara Intel® RealSense™ SR300. Intel Corporation.
  - Ivanov, P. et al. (2016). US Patente n° US 9,465,590 B2.
  - Jimenez J., A. M. (2017). Mexican Sign Language Alphanumerical Gestures Recognition using 3D Haar-like Features. Yucatan, Merida, México: IEEE LATIN AMERICA TRANSACTIONS.
  - Krasner, H. (2018). The Cost of Poor-Quality Software in the US: A 2018 Report. CISQ. 28-33.
  - Krupka, E. et al. (2017). Toward realistic hands gesture interface: Keeping it simple for developers and machines. Microsoft Research. CHI'17, May 6–11, 2017, Denver, USA. Copyright © 2017 ACM ISBN/14/04. DOI
  - Lara-Valencia, L. et al. (2015). Uso de lógica difusa para la administración de un sistema disipador de energía en estructuras compuesto por amortiguadores magnetoreológicos. Rev. Fac. Ing. Univ. Antioquia N. °74 pp. 151-164, Marzo ,2015.
  - Lingling, F. et al. (2019) Automated Cross-Platform GUI Code Generation for Mobile Apps. 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile), pp. 13-16, doi: 10.1109/AI4Mobile.2019.8672718.
  - Marin. (2019). El lenguaje corporal como medio de comunicación no verbal. Communications Media Conference Paper: ResearchGate.
  - Marouane, B., A. M. (2018). Machine Learning for Hand Gesture Recognition Using Bag-of-words. Rabat, Marruecos: IEEE.

- Mi-Eun, K. and Y. B. Park, (2017). Automatic Generation of UML Model-Based Image Processing Source Code in Hadoop Platform, 2017 International Conference on Platform Technology and Service (PlatCon), pp. 1-4, doi: 10.1109/PlatCon.2017.7883690.
- Mohiminul, S. (2017). Real Time Hand Gesture Recognition Using Different Algorithms Based on American Sign Language. Sylhet, Bangladesh: IEEE.
- Romero, L. (2017). Generación de palabras a partir de la lengua de señas mexicana (tesis de maestría). Centro Nacional de Investigación y Desarrollo Tecnológico.
- Saini, A, Security Consultant y Cigital/Synopsys. (2017). The cost of fixing bugs throughout the SDLC. CBR. Obtenido de: <https://www.cbronline.com/enterprise-it/software/cost-fixing-bugs-sdlc/>
- Sanket. (2019). Exponential cost of fixing bugs. DeepSource. Obtenido de: <https://deepsources.io/blog/exponential-cost-of-fixing-bugs/>
- Sen, C. et al. (2019). Automated Cross-Platform GUI Code Generation for Mobile Apps. IEEE.
- Siju, W., Amine, C., and Samir, O. (2014). FingerOscillation: clutch-free techniques for 3D object translation, rotation and scale. In Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology (VRST '14). Association for Computing Machinery, New York, NY, USA, 225–226. DOI: <https://doi.org/10.1145/2671015.2671117>
- Standish Group. (2015). CHAOS REPORT 2015. Standish Group.
- Sunitha, E. V. & Philip, S. (2019). Automatic code generation from UML state chart diagrams. IEEE Access.18p
- Supaartagorn, C. (2017). Web application for automatic code generator using a structured flowchart, 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017, pp. 114-117, doi: 10.1109/ICSESS.2017.8342876.
- Tang, J. et al. (2019). Position-free hand gesture recognition using single shot multibox detector based neural network. IEE International conference on mechatronics and automation. 6p
- Osmolik, V. (2019). Desarrollo de aplicación web para gestionar un campeonato de futbol usando metodología snail y lenguaje de programación Python. (Examen complejo) Universidad Técnica de Machala. Obtenido de: <http://repositorio.utmachala.edu.ec/bitstream/48000/13599/1/ECUAIC-2019-SIS-DE00005.pdf>
- Tomasz, H., M. R. (2014). Rule-based approach to recognizing human body poses and gestures in real time. Kraków, Polonia: Springer.
- Krupka, E., et al. (2017). Toward Realistic Hands Gesture Interface: Keeping it Simple for Developers and Machines. ACM. Obtenido de: <https://www.microsoft.com/en-us/research/publication/toward-realistic-hands-gesture-interface-keeping-simple-developers-machines/>
- Lozada, R. & Molina, F. & Ecriba, L. (2014). Interfaces de Usuario Natural. 10.13140/RG.2.1.5092.2324.
- Marin, R. (2019). El lenguaje corporal como medio de comunicación no verbal. Obtenido de: [https://www.researchgate.net/publication/333703741\\_El\\_lenguaje\\_corporal\\_como\\_medio\\_de\\_comunicacion\\_no\\_verbal](https://www.researchgate.net/publication/333703741_El_lenguaje_corporal_como_medio_de_comunicacion_no_verbal)



- Molina, J. (2018). Reconocimiento de movimientos de manos para la manipulación de una interfaz computacional (tesis de maestría). Centro Nacional de Investigación y Desarrollo Tecnológico.
- uOttawa (2018). Umple Online. uOttawa. Obtenido de: <http://cruise.eecs.uottawa.ca/umpleonline>
- Zamora, G. (2015). Defusificación para un Controlador Difuso - Hackeando Tec. Youtube: Hackeando Tec. Obtenido de <https://www.youtube.com/watch?v=G0KJA3JUrn>

# Anexos

# Anexos

## Anexo 1. Publicación de artículo

Se realizó la publicación de un artículo en el Congreso de Academia Journals Oaxaca 2021.

Memorias del Congreso Internacional de Investigación Academia Journals Oaxaca 2021

© Academia Journals 2021

Oaxaca de Juárez, Oaxaca, México  
9, 10, y 11 de junio de 2021

### Revisión de Dispositivos de Interacción Humano-Computadora para el Reconocimiento de Gestos

Primer autor: Ing. Humbertino Avilez Carpintero<sup>1</sup>  
Autor correspondiente: Dr. Máximo López Sanchez<sup>2</sup>  
Dr. Juan Gabriel González Serna<sup>3</sup> Dr. Dante Mújica Vargas<sup>4</sup> Dr. Noé Alejandro Castro Sanchez<sup>5</sup>

**Resumen**— En la actualidad existen varios dispositivos para llevar a cabo el reconocimiento de gestos. Cada dispositivo cuenta con características diferentes, unos enfocados para grandes distancias y otros por el contrario a distancias más cortas, una característica es que permiten enfocarse en algunas o varias áreas del cuerpo humano, que, mediante una serie de métodos y técnicas puedan crear modelos para la detección y el reconocimiento de movimientos corporales tales como: las manos, el rostro, dedos e incluso el cuerpo mismo. El enfoque actual de esta tecnología se ha expandido para analizar y estudiar estos dispositivos aplicándolos en otros ámbitos diferentes al del entretenimiento, demostrando su uso para el control de dispositivos o comunicación del lenguaje.

**Palabras clave**—dispositivos, gestos, reconocimiento, distancias.

#### Introducción

El reconocimiento de gestos es un enfoque que se ha ido implementando en el área de la visión artificial, el cual mediante dispositivos y algoritmos matemáticos, se encarga de entender el comportamiento del cuerpo humano para crear un puente de comunicación y control del tipo humano-computadora. Los movimientos que comúnmente se toman en cuenta para explotar estas técnicas son los realizados: por las manos, el rostro y el cuerpo completo. De esta manera se han llegado a desarrollar múltiples sistemas, lo que tienen como control principal los movimientos corporales, estos sistemas llevan por nombre “interfaces naturales de usuario” (por sus siglas en inglés NUI: Natural User Interface) donde también se incluye el reconocimiento de voz. En la actualidad, el ámbito en el cual se ha visto el uso de esta tecnología es principalmente en la del entretenimiento, siendo más específicos, en el área de los videojuegos y reproducción multimedia, un claro ejemplo es el uso del Kinect en la consola xbox360, que, mediante gestos del cuerpo el usuario interactúa en el video juego, otro ejemplo es el uso del dispositivo Leap Motion Controller, el cual utiliza el reconocimiento de gestos de las manos en los videojuegos de realidad virtual, haciendo que el usuario tenga una experiencia más inmersiva. Pero a pesar de ello se han realizado investigaciones para utilizar en otros ámbitos estos dispositivos y métodos de reconocimiento, tales como: lenguaje de señas, control de vehículos, manejo de robots, entre otros.

#### Dispositivos para reconocimiento de gestos

En la actualidad existen diversos dispositivos para llevar a cabo la tarea del reconocimiento de gestos que permiten controlar sistemas con los movimientos corporales. Desde dispositivos que alcanzan distancias amplias hasta aquellos que solo cuentan con algunos centímetros de visión funcional para llevar a cabo la tarea de manera exitosa. Los dispositivos con un alto rango de distancia funcional, comúnmente se enfocan en el reconocimiento de gestos hechos considerando el cuerpo humano completo. Muy contrario a los dispositivos de corta distancia, que se enfocan en áreas más pequeñas como lo son: las manos, el rostro o los dedos; siendo más versátiles al querer reconocer movimientos más complejos, como los movimientos del rostro o la interacción que tienen los dedos de las manos unos con otros. Entre los dispositivos más utilizados para llevar a cabo este tipo de tarea se encuentran los siguientes:


**Para largas distancias**

**Kinect:** Es la primera versión de Kinect la cual es un complemento de la Xbox 360, integra cámaras, sensores de profundidad y una serie de micrófonos que detectan los movimientos corporales y puede interpretarlos en acciones. Microsoft recomienda un “rango óptimo” de 1.2m a 3.5m del sensor, así como un rango angular de 57°

<sup>1</sup> Ing. Humbertino Avilez Carpintero. TecNM/ CENIDET. humbertino.avilez@cenidet.edu.mx  
<sup>2</sup> Dr. Máximo López Sanchez. TecNM/ CENIDET. maximo.ls@cenidet.tecnm.mx  
<sup>3</sup> Dr. Juan Gabriel González Serna. TecNM/ CENIDET. gabriel.gs@cenidet.tecnm.mx  
<sup>4</sup> Dr. Dante Mújica Vargas. TecNM/ CENIDET. dante.mv@cenidet.tecnm.mx  
<sup>5</sup> Dr. Noé Alejandro Castro Sanchez. TecNM/ CENIDET. noe.cs@cenidet.tecnm.mx

ISSN online 1946-5351  
Vol. 13, No. 5, 2021

17



ACADEMIA JOURNALS  
CENTRO DE INVESTIGACIÓN DE OAXACA

x 43° (horizontal x vertical) y se puede extender verticalmente con ayuda de un motor de inclinación con un rango de 54°. Sus dimensiones son de 28cm x 6cm x 7.5cm. Es capaz de detectar hasta 6 personas, pero solo puede rastrear perfectamente a 2. En cada persona rastreada puede identificar 20 articulaciones. Trabaja con la tecnología de luz estructurada y solo se puede usar en interiores (Smisek J. et al, 2013). En la literatura se encuentran varios trabajos realizados con el dispositivo Kinect, uno de ellos se presenta en:

“Machine Learning for Hand Gesture Recognition Using Bag-of-words” en el cual se propone un método de aprendizaje automático para el reconocimiento de 16 gestos en tiempo real. El procedimiento inicia con la obtención de las imágenes obtenidas por el Kinect, utilizando una distancia óptima de 0.8m a 2.5m y obteniendo 500 imágenes por gesto. Utiliza un enfoque de bolsa de palabras obtenidas mediante los descriptores SURF y SIFT, forman histogramas para el entrenamiento y pruebas, las que alimentan un clasificador máquina de soporte vectorial (Benmoussa M. & Mahmoudi A, 2018).

**Kinect v2:** En esta versión el dispositivo opera en un rango óptimo de 0.5m a 4.5m, con un campo de visión de 70.6° x 60° (horizontal x vertical). Con dimensiones de 249mm x 66mm x 67mm. El dispositivo es capaz de capturar imágenes de profundidad precisas a una mayor velocidad, pudiendo detectar hasta 6 personas y rastrear a todos los individuos, donde cada rastreo de persona identifica 25 articulaciones. A comparación de la versión original, este dispositivo trabaja con el principio de medición del tiempo de vuelo y es capaz de funcionar en exteriores (Fankhauser P. et al., 2015). Unos de los trabajos realizados con este dispositivo son:

“Hand Gesture Recognition Using K-Means Clustering and Support Vector Machine” se ponen a prueba dos métodos para el reconocimiento de gestos hechos por la mano, k-means y máquina de soporte vectorial. El trabajo se enfoca en reconocer 4 gestos para controlar a un robot. Se realizaron pruebas desde tres diferentes distancias: 2m, 3m y 4m. El proceso de reconocimiento rastrea las articulaciones del cuerpo con el dispositivo, para enviarlas a un pre-procesamiento usando técnicas de estadística y obtener las distancias de las articulaciones del cuerpo. Después la información se usará para los dos métodos: k-means y máquina de soporte vectorial. Este proceso se repite para la parte de entrenamiento y prueba (Maharani D. et al, 2018).

“Hand Gesture for Elderly Care Using a Microsoft Kinect” propone el reconocimiento de 5 gestos de manos para indicar ciertas peticiones como: “necesito agua”, “quiero comer”, “quiero ir al baño”, “necesito ayuda”, “necesito medicina”. El trabajo se dirige al cuidado de personas mayores de edad. El proceso comienza con la obtención de imágenes en tiempo real con el dispositivo Kinect a una distancia de 1 a 1.5 m. Después se hace la segmentación de la mano. Finalmente se hace un conteo de los dedos para identificar el gesto del usuario. El gesto es transmitido a un mini controlador que manda el mensaje a un celular, a través de un pequeño modem. Con una precisión aproximada el 94% (Oudah M. & Abdulelah A., 2020).

**WI-FI:** En este caso no se está hablando de un dispositivo, sino de una tecnología, que es utilizada para la interconexión de dispositivos. El alcance que posee esta tecnología va desde los 15m a 45m en interiores aproximadamente, y de 30m a 90m en exteriores aproximadamente. Esta distancia varía por diferentes factores como: material del dispositivo, protocolos, interferencias de radio, obstáculos físicos o la posición de las antenas (Martínez S., 2005). El WI-FI utiliza ondas electromagnéticas y dentro de estas son microondas que junto con las ondas de radio pertenecen a las radiofrecuencias. La tecnología WI-FI comúnmente hace uso de la banda de frecuencia 2.4 GHz, que se encuentra dentro del rango de las microondas, aunque también existe el uso de la banda de frecuencia 5 GHz (Pedreño O. et al., 2015). Este dispositivo se utilizó en:

“Zero-Effort Cross-Domain Gesture Recognition with WI-FI” en este trabajo se hace uso de dispositivos WI-FI COTS para el reconocimiento de gestos del cuerpo humano en diferentes lugares y entornos. Proponen un sistema llamado Widar3.0. Widar3.0 describe la distribución de energía sobre diferentes velocidades, en la cual las partes del cuerpo están involucradas al momento de realizar algún gesto corporal. Estas señales generan un “perfil de velocidad de coordenadas corporales” para la etapa de reconocimiento de gestos, esta última etapa es un modelo híbrido, en el cual se incluye una red neuronal convolucional para la extracción de características espaciales y una red neuronal recurrente para un modelado temporal y lograr el reconocimiento de gestos corporales (Zheng Y. et al, 2019).

**Cámaras web:** Son dispositivos digitales con dimensiones muy cortas, y al igual que una cámara convencional, las cámaras web en la actualidad pueden capturar imágenes y video de muy buena calidad, además de que pueden transmitirlos por medio de la red. El “alcance” en estos dispositivos varían respecto a la resolución que estos puedan manejar, en la actualidad en el mercado una cámara web puede alcanzar fácilmente una resolución de 1920 x 1080, proporcionando imágenes de calidad del cuerpo humano. Uno de los trabajos relacionados con cámaras web para el reconocimiento de gestos es el siguiente:

“Gesture Recognition in RGB Videos Using Human Body Keypoints and Dynamic Time Warping” propone un método que se basa en la captura de video RGB y requiere una cámara que grabe con una calidad razonable, para hacer el reconocimiento de gestos más accesible. Después se extraen las poses utilizando el framework OpenPose y

a continuación se hace una normalización de escalado y de traslación, así, de los ejemplos de los gestos junto con los puntos clave dados por el clasificador de un vecino más cercano, se aplica un preprocesamiento de la información para obtener las distancias de deformación para ser clasificadas con el clasificador utilizado (Schneider P. et al, 2019).

#### *Para cortas distancias*

**Leap Motion Controller:** El rango de eficiencia del dispositivo contempla aproximadamente de 1 pulgada a 2 pies por encima del dispositivo. Es un dispositivo capaz de detectar y rastrear manos, dedos y objetos similares a los dedos, mostrando datos discretos de movimiento y posición. Con dimensiones de 7.6cm x 3cm x 1.2cm. El dispositivo opera un aproximado de 200 cuadros por segundo. Utiliza dos cámaras infrarrojas de alta precisión y tres leds para capturar información de la mano dentro de su rango activo. El dispositivo utiliza un sistema de coordenadas cartesianas diestras (Mohandes M. et al., 2014). El Leap Motion Controller ha tenido varias aplicaciones desde su lanzamiento, entre ellos está el siguiente trabajo:

“Dynamic Hand Gesture Recognition Based on Leap Motion Controller and Two-Layer Bidirectional Recurrent Neural Network” en este trabajo una de las pruebas estuvo enfocada con el lenguaje de señas americano ofreciendo una exactitud aproximada del 95.2%. El proceso consta de la captura de las manos por el Leap Motion en tiempo real, para proceder con un procesamiento de las imágenes, así como identificar el inicio y final de la ejecución de un gesto. Después se procede con la aplicación de una red neuronal recurrente bidireccional de dos capas para llevar a cabo la clasificación de los gestos (Yang, L. et al, 2019).

“Hand Gesture Recognition with Leap Motion” en este trabajo se ponen a prueba 10 gestos, en el reconocimiento de gestos de la mano. El proceso consiste en la obtención de datos del Leap Motion, como la información de rastreo, y las imágenes capturadas. Después se hace una fusión de características, de las características obtenidas del rastreo, y de las características obtenidas por un histograma de gradiente orientado aplicado a las imágenes capturadas del dispositivo. Al conjunto fusionado de características se le aplica un análisis de componentes principales para poder alimentar un clasificador de máquina de soporte vectorial multiclase, y así llevar a cabo el reconocimiento de gestos. Esta combinación de características permitió alcanzar una exactitud del 99.42% (Youchen D. et al, 2017).

**RealSense SR300:** El dispositivo cuenta con un intervalo de operación óptima de 0.3m a 2m. Es capaz de realizar seguimiento de manos y dedos, análisis y reconocimiento facial, reconocimiento de gestos, escaneo 3D, segmentación de fondo y un modo de cursor que permite usar el dedo como cursor. Con dimensiones de 110mm x 12.5mm x 3.75mm. Este dispositivo tiene una resolución de 1080p para transmitir a 30 cuadros por segundo, y una resolución de 720p para transmitir a 60 cuadros por segundo, además cuenta con un proyector láser de infrarrojos y con capacidades de profundidad 3D. También incluye micrófonos de matriz dual para comandos de voz y control (Patil J. et al, 2016 y MouserElectronics, 2017). Algunos de los trabajos relacionados con este dispositivo son los siguientes:

“Joint-based Hand Gesture Recognition Using RealSense” en este trabajo se reconocen gestos mediante el uso de articulaciones de la mano. El dispositivo RealSense proporciona información de las articulaciones de la mano para la extracción de características: distancia de los dedos, ángulos de articulaciones, distancia de dedos adyacentes, ángulos de dedos adyacentes. El siguiente paso es la selección de características, en el cual se aplica el F-score para seleccionar las mejores características y poder ser usadas por el clasificador de máquina de soporte vectorial multiclase. Obteniendo una exactitud de aproximadamente el 96% (Yun W. et al, 2019).

“Toward realistic hands Gesture interface: Keeping it simple for developers and machines” propone un lenguaje simple para describir poses y gestos de las manos y un algoritmo capaz de reconocerlos con una alta exactitud a través del dispositivo. El lenguaje está basado en un conjunto de proposiciones básicas por ejemplo “el índice y medio no están tocándose”, las proposiciones toman en cuenta la pose de la palma y los dedos. El algoritmo para la estimación de la pose de mano consiste en 6 etapas y cada etapa hace uso de un conjunto de predictores de tablas convolucionales. Los resultados arrojaron una detección arriba del 90% de exactitud (Krupka E. et al, 2017).

**Cámaras:** Tanto en las cámaras digitales como en las cámaras web, su “alcance” se ve afectado en la resolución que estas puedan manejar, en el mercado actual estos dispositivos fácilmente se encuentran con resoluciones de 1920 x 1080 e incluso con mayores valores, otorgando imágenes más claras para detectar aspectos más específicos como lo son las manos o el rostro. Las cámaras digitales son capaces de poder conectarse a una computadora y tener el mismo desempeño o incluso superior que una cámara web. Algunas poseen funciones especiales tales como: el uso de visión nocturna, seguimiento de rostro, recepción de audio mediante micrófonos o resistencia al agua. Sus dimensiones varían respecto a los modelos que en la actualidad se comercializan, como las cámaras de seguridad que suelen ser pequeñas o las cámaras profesionales que suelen usar los fotógrafos, que son de dimensiones más grandes y de un mayor peso.

Uno de los trabajos donde se pone a prueba una cámara digital para el reconocimiento de gestos es el siguiente:

“Position-Free Hand Gesture Recognition Using Single Shot MultiBox Detector Based Neural Network” utiliza una cámara simple para el reconocimiento de gestos de mano. Captura imágenes en tiempo real para ser procesadas, detectar y delimitar una mano en la escena, posteriormente la parte delimitada es segmentada para pasarla como entrada en una red neuronal convolucional y predecir el gesto realizado. El sistema fue entrenado con 9 gestos de mano. El modelo puede detectar la mano con fondos complejos, la segmentación de la mano no es suficientemente exacto, perdiendo información en el traslape de dedos. Los resultados se compararon con otros modelos, dando como resultado un 99.85% de exactitud (Jingwei T., et al. 2019).

**Teléfonos inteligentes:** Los teléfonos inteligentes o mejor conocidos como smartphone cuentan con una herramienta muy característica de estos dispositivos la cual es una pantalla táctil, además de un sistema operativo y la conectividad a internet. El uso de pantallas táctiles permite que el usuario utilice sus dedos para interactuar con el sistema. Estas pantallas táctiles en la mayoría de los teléfonos inteligentes de la actualidad, son capaces de reconocer desde dos dedos hasta un máximo de diez dedos o más, permitiendo que los desarrolladores puedan utilizar esto a su favor, facilitando ciertas funciones en sus aplicaciones con la ejecución de un simple gesto de dedos sobre la pantalla (Lü H, 2014). En la actualidad la detección de gestos en pantallas táctiles es muy utilizada, por ejemplo, al realizar zoom en un smartphone u ocultar ventanas en una computadora. En algunos casos se busca ampliar el uso de estos, como en el siguiente trabajo:

“Touch&Screen: Widget Collection for Large Screens Controlled through Smartphones” propone un conjunto extenso de técnicas para el control remoto de widgets, utilizado en pantallas grandes mediante teléfonos inteligentes. Trabaja relacionando áreas del smartphone con widgets en la pantalla grande (p. ej. elementos con el mismo color). Utilizan ejecuciones animadas para tener al usuario notificado de lo que están haciendo o en donde está ubicado. Desarrollaron 8 técnicas de interacción. El trabajo se evaluó en la experiencia del usuario y la comparación de las técnicas con otros sistemas como los controladores de cursores a distancia (Bellino A. et al, 2016).

**Superficie de electromiografía:** Registra la actividad eléctrica muscular y por tanto constituye una extensión de la exploración física y prueba la integridad del sistema motor. En otras palabras, es el análisis electromiográfico que permite recoger la señal eléctrica de un músculo en un cuerpo en movimiento, esto quiere decir que se necesita que el cuerpo este en contacto con la superficie para recolectar información de movimiento. Esta herramienta se puede utilizar para las acciones que implican movimiento, pero también es aplicable al estudio de acciones estáticas que requieren un esfuerzo muscular de carácter postural (Massó N et al. 2010). Esta herramienta fue utilizada en el siguiente trabajo para el reconocimiento de gestos:

“Surface EMG hand Gesture recognition system based on PCA and GRNN” se basa en el análisis de componentes principales y en la aplicación de una red neuronal de regresión generalizada para la reducción de información redundante que presentan las señales de una superficie de electromiografía. Se toman como ejemplo 9 gestos de mano para evaluar su sistema. El proceso comienza con la extracción de características que son obtenidas por la superficie de electromiografía (p. ej. dominio del tiempo, dominio de frecuencia, entre otras.). Se aplica en un análisis de componentes principales y poder construir una red neuronal de regresión generalizada para la clasificación de gestos. La exactitud del reconocimiento de gestos alcanzó un 95.1%, y el tiempo para calcular fue de 0.19s, lo que indica que es posible aplicar el proceso en tiempo real (Zheng Y. et al, 2019).

**Gautes con sensores de medición:** Este tipo de dispositivos suelen tener un cierto número de sensores para llevar a cabo la medición de los movimientos que pudiese realizar la mano. Entre los sensores que comúnmente se utilizan están los siguientes:

**Acelerómetro:** el cual es un dispositivo electromecánico que mide las fuerzas de aceleración, estos se encuentran a escalas de décimas de micrones con niveles de sensibilidad y error muy pequeños. La unidad de medida son la gravedad ( $g$ 's). Existen modelos compuestos por uno, dos y tres ejes de detección. Estos dispositivos son utilizados para obtener los cambios de velocidad con respecto al tiempo y para el control de mecanismos de vibración, se utilizan para medir la actividad sísmica, la inclinación, la vibración de las máquinas, la distancia dinámica y la velocidad con o sin la influencia de la gravedad (Rincon-Jara et al., 2010).

**Giroscopios:** es un sensor que mide o mantiene el movimiento rotacional. Los giroscopios de sistema micro electromecánico, son sensores pequeños que miden la velocidad angular. Las unidades de velocidad angular se miden en grados por segundo ( $^{\circ}/s$ ) o revoluciones por segundo (RPS). Se pueden utilizar para medir la rotación de la posición balanceada y enviar las correcciones a un motor (Paguayo, 2019).

**Magnetómetro:** es un sensor enfocado a cuantificar la fuerza de la señal magnética de una muestra. En dispositivos como smartphone se implementan para que se puedan encender y apagar sin necesidad de pulsar ningún botón, utilizando un imán también conocido como sensor Hall. Se puede utilizar como brújula haciendo uso correcto

en la medición del campo magnético de nuestro planeta, para facilitar coordenadas de orientación (Torres J. et al. 2007).

Estos dispositivos al ser colocados en guantes, se enfocan en el reconocimiento de gestos de las manos, como se presenta en el siguiente trabajo:

“MagicWatch: interacting y Segueing” trabajo enfocado en detectar los gestos de la mano del usuario, usando varios sensores de medición. Su capacidad le permite funcionar como un apuntador, control remoto; y un portal para conseguir información desde la nube. El reloj inteligente está conformado por varios sensores, algunos de ellos son: acelerómetro, giroscopio, magnetómetro, GPS, bluetooth, entre otros. Se controla mediante el sistema operativo Android. El proceso de reconocimiento de gestos está basado en la aceleración, se recolectan los datos del acelerómetro al realizar un gesto para ser representado en un descriptor y obtener información discriminativa. Esta información sirve para construir un clasificador multiclase de máquina de soporte vectorial y realizar el reconocimiento de gestos. La experimentación muestra un 95% de exactitud en las pruebas (Feng Y. et al, 2014).

#### Conclusión

Como se puede apreciar existen varios dispositivos que facilitan la tarea de la interacción humano-computadora mediante el reconocimiento de gestos, tanto para cortas y largas distancias, que muestran buenos resultados, ofreciendo como otra opción de controlador de sistemas al mismo cuerpo humano. Además, esta tecnología comienza a desarrollarse cada vez más y en un futuro se podrían tener dispositivos con mayores ventajas y mejores resultados de los que posee cualquier herramienta mecánica de control en la actualidad y así poder aplicarla en: un control para un vehículo, ejecución tareas domésticas, traductores de lenguaje de señas, hasta diseño 3d virtualmente como en la ciencia ficción.

#### Sugerencias de uso

Se tienen dispositivos que poseen la ventaja de trabajar a largas distancias, los que permiten trabajar con todo el cuerpo e incluso se podría aplicar con partes específicas tales como la mano, pero si no se tiene un área física lo suficientemente amplia el uso de estos dispositivos podría perder su versatilidad a la hora de manipularlos. Por otro lado, los dispositivos que operan a cortas distancias se enfocan en partes específicas del cuerpo, ya que sus capacidades no les permiten una mayor visión, pero en este caso, no se requiere un área muy grande, ya que estos dispositivos suelen tener dimensiones pequeñas, como en el caso del Leap Motion Controller.

#### Agradecimientos

Agradecemos el apoyo a: el CONACYT, el Tecnológico Nacional de México y en especial al campus CENIDET y al proyecto: 10437.21-P Identificación de alteraciones en el iris del ojo para detectar patrones que permitan medir si existe correlación con personas diagnosticadas con diabetes mellitus

#### Referencias

- Bellino A., Cabitza F., De Michelis G. & De Paoli F., “Touch&Screen: Widget Collection for Large Screens Controlled through Smart”, Proceedings of the 15th International Conference on Mobile and Ubiquitous Multimedia (MUM '16), December 12–15, 2016, Rovaniemi, Finland, <http://dx.doi.org/10.1145/3012709.3012736>
- Benmoussa M. & Mahmoudi A., “Machine Learning for Hand Gesture Recognition Using Bag-of-words”, IEEE, 2018, 978-1-5386-4396-9/18/
- Fankhauser P., Bloesch M., Rodriguez D., Kaestner R., Hutter M. & Siegwart R., “Kinect v2 for Mobile Robot Navigation: Evaluation and Modeling”, 2015 International Conference on Advanced Robotics (ICAR), Istanbul, 2015, pp. 388-394, doi: 10.1109/ICAR.2015.7251485
- Feng Y., Shugang W., Shijian L., Gang P. & Runhe H., “MagicWatch: Interacting & Segueing”, UbiComp'14 Adjunct, September 13-17, 2014, Seattle, WA, USA. <http://dx.doi.org/10.1145/2638728.2638848>
- Jingwei T., Xingtian Y., Xin K., Shun N. & Fuji R., “Position-Free Hand Gesture Recognition Using Single Shot MultiBox Detector Based Neural Network”, IEEE, International Conference on Mechatronics and Automation August 4 - 7, Tianjin, China, 2019, 978-1-7281-1699-0/19/
- Krupka E., Freedman D., Leichter I., Karnon K., Gurvich I., Smolin Y., Bloom N., Hurvitz A., Tzairi Y., Vinnikov A & Hillel A., “Toward realistic hands gesture interface: Keeping it simple for developers and machines”, CHI Conference on Human Factors in Computing Systems May 2017 Pages 1887–1898 <https://doi.org/10.1145/3025453.3025508>
- Lü H., Negulescu M. & Li Y., “Gesturermote: Interacting with Remote Displays through Touch Gestures”. Proceedings of the Workshop on Advanced Visual Interfaces AVI. 10.1145/2598153.2598195. Mayo 2014.

- Maharani D., Fakhruroja H., Riyanto & Machbub C., "Hand gesture recognition using K-means clustering and Support Vector Machine," 2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, 2018, pp. 1-6, 10.1109/ISCAIE.2018.8405435.
- Martínez S., "Análisis de la calidad de señal en una red wifi con la herramienta netstumbler". *Umbral Científico*, (en línea). 2005(7) 61-71, ISSN: 1692-3375.
- Massó N., Rey F., Romero D., Gual G., Costa L., y Germán A. "Aplicaciones de la electromiografía de superficie en el deporte", *Elsevier España*, 5 de febrero de 2010.
- Mohandes M., Aliyu S. and Deriche M. "Arabic sign language recognition using the leap motion controller," 2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE), Istanbul, pp. 960-965, doi: 10.1109/ISIE.2014.6864742. 4 de junio del 2014
- MouserElectronics "Intel® RealSense™ Camera SR300 - New Product Brief | Mouser Electronics" [Archivo de video]. Youtube. 20 de septiembre del 2017.
- Oudah M. & Abdulelah A., "Hand Gesture for Elderly Care Using a Microsoft Kinect", *Nano Biomedicine and Engineering*, 12. 197-204. 10.5101/nbe.v12i3.p197-204., Julio 2020.
- Paguayo. "Giroscopio", MCI capacitación- MCI Electronics (en línea). 18 de junio del 2019.
- Patil J. & Bilke P. "Real Time Facial Expression Recognition using RealSense camera and ANN", *International Conference on Inventive Computation Technologies (ICICT)*, Coimbatore, 2016, pp. 1-6, doi: 10.1109/INVENTIVE.2016.7824820. 2016
- Pedreño O. & Villamor P. "wifi y salud", universidad de Valladolid, facultad de enfermería, 2015.
- Rincon-Jara R., Ambrosio R. & Mireles J. "Análisis y caracterización de un acelerómetro capacitivo fabricado con tecnología polimump's", *Sociedad Mexicana de Ciencia y Tecnología de Superficies y Materiales Superficies y Vacío* 23 (3) 26-31. septiembre de 2010
- Schneider P., Memmesheimer R., Kramer J. & Paulus D., "Gesture Recognition in RGB Videos Using Human Body Keypoints and Dynamic Time Warping", Chalup S., Niemueller T., Suthakorn J., Williams MA. (eds) *RoboCup 2019: Robot World Cup XXIII. RoboCup 2019. Lecture Notes in Computer Science*, vol 11531. Springer, Cham. [https://doi.org/10.1007/978-3-030-35699-6\\_22](https://doi.org/10.1007/978-3-030-35699-6_22), 2019
- Smisek J., Jancosek M. & Pajdla T. "3D with Kinect" Consumer Depth Cameras for Computer Vision. *Advances in Computer Vision and Pattern Recognition*. Springer, London. [https://doi.org/10.1007/978-1-4471-4640-7\\_1](https://doi.org/10.1007/978-1-4471-4640-7_1), 2013
- Torres J., Cruz B. y Villegas C. "Criterios para la instrumentación de un magnetómetro hall en corriente continua", *Scientia et Technica Año XIII*, No 37. Diciembre del 2007.
- Yang, L., Chen, J. & Zhu, W. "Dynamic Hand Gesture Recognition Based on a Leap Motion Controller and Two-Layer Bidirectional Recurrent Neural Network". *Sensors* 2020, 20, 2106.
- Youchen D., Shenglan L., Lin F., Menghui C. & Jie W., "Hand Gesture Recognition with Leap Motion", Cornell University, arXiv:1711.04293v1, 12 Nov 2017.
- Yun W., Dong-chen H., Wei-Chang D., Meng-ke W. & Chun-Zhen L., "Joint-based Hand Gesture Recognition Using RealSense", *Journal of Computers* Vol. 31 No. 2, 2020, pp. 141-151 doi:10.3966/199115992020043102013, 6 de Marzo del 2019
- Zheng Y., Zhang Y., Qian K., Zhang G., Liu Y., Wu C. & Yang Z., "Zero-Effort Cross-Domain Gesture Recognition with Wi-Fi", *MobiSys '19*, June 17-21, 2019, Seoul, Republic of Korea, Association for Computing Machinery, <https://doi.org/10.1145/3307334.3326081>



## Anexo 2. Publicación de artículo

Se realizó la publicación de un artículo en la Xi Congreso Internacional De Computacion México – Colombia Cicom 2021.



**XI CONGRESO INTERNACIONAL DE COMPUTACION  
MÉXICO - COLOMBIA  
CICOM 2021**

Octubre 20, 21 y 22 de 2021, Colombia

### Identificación de gestos realizados por la mano utilizando una gramática libre de contexto

**Ing. Humbertino Avilez Carpintero**  
TecNM/ CENIDET  
humbertino.avilez@cenidet.edu.mx

**Dr. Máximo López Sánchez**  
TecNM/ CENIDET  
maximo.ls@cenidet.tecnm.mx

**Dr. Juan Gabriel González Serna**  
TecNM/ CENIDET  
gabriel.gs@cenidet.tecnm.mx

**Dr. Dante Mújica Vargas**  
TecNM/ CENIDET  
dante.mv@cenidet.tecnm.mx

**Dr. Noé Alejandro Castro Sánchez**  
TecNM/ CENIDET  
noe.cs@cenidet.tecnm.mx

**RESUMEN**

Las gramáticas son un conjunto de reglas que se deben seguir para dar sentido y estructura a un determinado lenguaje. Existen diferentes tipos de gramáticas según la jerarquía de Chomsky, utilizándose la que exprese mejor el lenguaje que se quiere representar. En este artículo se hace uso de una gramática formal libre de contexto, la que comúnmente se utiliza cuando se expresan lenguajes de programación. El propósito de utilizar esta gramática, es para dar sentido a un conjunto de atributos obtenidos de un proceso de transformación de datos obtenidos de un sensor de profundidad, y así poder facilitar la continuación de un proceso de generación de código. Esta generación de código se enfocaría en la descripción del comportamiento estático, que está teniendo una pose de mano capturada por el sensor de profundidad.

**ABSTRACT**

Grammars are a set of rules that must be followed to give meaning and structure to a certain language. There are different types of grammars according to the Chomsky hierarchy, using the one that best expresses the language to be represented. This article makes use of a formal context-free grammar, which is commonly used when expressed in programming languages. The purpose of using this grammar is to make sense of a set of attributes obtained from

El permiso para hacer copias digitales o impresas en parte o en la totalidad de este artículo, se otorga sin tener que cubrir una contribución financiera, siempre y cuando sea para uso personal o en el aula, que las copias que se realicen o se distribuyan no sean con fines de lucro o ventaja comercial y que las copias conserven este aviso y los datos de la cita completa en la primera página. Para otro tipo de copias, o volver a publicar el artículo, para almacenarlos en servidores o redistribuirlo en listas de correo, se requiere una autorización previa de los autores y/o una posible cuota financiera.

XI Congreso Internacional de Computación CICOM 2021, (20 al 22 de octubre del 2021). Sede virtual: Colombia. Copyright 2021 Universidad Distrital Francisco José de Caldas.

**Categorías y Descriptores Temáticos**  
Ing. de Software--Ciencia e Ingeniería de la Computación

**Términos Generales**  
Gramáticas formales, reconocimiento de gestos, generación de código

**Palabras clave**  
Gramática, generación, código, gestos, detección

**Keywords**  
Grammar, generation, code, gestures, detection

**INTRODUCCIÓN**

En este documento se hace uso de una gramática formal del tipo 2 de la jerarquía de Chomsky, con el propósito de definir una estructura específica y así favorecer el desarrollo de un proceso de generación de código de programación en un evento posterior. El código obtenido está enfocado a reconocer los gestos que se realicen con la mano, los que son capturados mediante una cámara que cuenta con un sensor de profundidad.

La información que se recabe a través de este proceso deberá transformar los datos puntuales en atributos para obtener datos descriptivos, que mediante una serie de reglas de comportamientos puedan realizar con mayor facilidad la generación de código, por lo que es importante implementar una gramática que mediante sus reglas de producción ayuden a identificar de mejor manera los gestos de la mano que sean posibles de realizar.



## XI CONGRESO INTERNACIONAL DE COMPUTACION MÉXICO - COLOMBIA CICOM 2021

Octubre 20, 21 y 22 de 2021, Colombia

Un ejemplo de la aplicación de gramáticas formales es en el uso que se le ha dado para describir la sintaxis de un lenguaje de programación de alto nivel al momento de compilar un programa, es parte de la verificación de que un programa esté escrito correctamente para su transformación en lenguaje de bajo nivel.

### OBJETIVO

Identificar gestos realizados por la mano mediante una gramática libre de contexto para permitir la generación de código

### METODOLOGÍA Y PROCESOS DE DESARROLLO

#### Tipos de gramática

De acuerdo a la literatura [1] se habla de que el uso de una gramática es útil para poder presentar pensamientos o ideas de una manera clara, de acuerdo a un conjunto de reglas y especificaciones que conlleve dicha gramática que se está abordando. Para esto existen diferentes tipos de gramáticas, las cuales son:

**Gramática prescriptiva o normativa:** Este gramático maneja lo que es un "esquema" o una manera correcta de lo que es el idioma a tratar, para poder guiar al hablante a formular y desarrollar correctamente sus oraciones.

**Gramática descriptiva:** A diferencia de la anterior, no juzga como "correcta" o "incorrecta" la manera en que distintos hablantes hacen uso del idioma, sino que aspira a comprender cómo es el uso real de las normas del idioma dentro de una comunidad o unas comunidades determinadas.

**Gramática tradicional:** Se trata del conjunto histórico de documentos e ideas heredadas de civilizaciones anteriores en torno a lo que la gramática es.

**Gramática funcional:** Aspira a ser una gramática general del lenguaje natural, o sea, un conjunto de normas básicas aplicables a diferentes idiomas dotados de gramáticas distintas.

**Gramáticas formales:** Se llaman así a las gramáticas abstractas, que pueden aplicar su lógica a lenguajes no verbales, como los lenguajes de programación informáticos.

#### Jerarquía Chomsky

A partir de esta información se hace la selección del tipo de gramática con la cual se trabajará, la que permitirá definir una gramática que ayudará en la generación de código, la cual es la gramática formal. Para esto, se encontró lo que es la jerarquía de Chomsky la cual habla que las gramáticas formales se dividen en cuatro tipos y la diferencia entre cada uno de ellas se basa en el comportamiento que tienen sus respectivas producciones. [2]

De este modo, ya sea una gramática con la estructura de la siguiente ecuación:

$$G = (\Sigma_T, \Sigma_N, S, P)$$

Donde:

- $\Sigma_T$  : Es un conjunto de terminales
- $\Sigma_N$  : Un conjunto de no terminales
- $S$  : Producción inicial

- $P$  : Conjunto de producciones

Los tipos de gramáticas formales de acuerdo a la jerarquía de Chomsky [2] son los siguientes:

- Tipo 0: Lenguajes recursivos

$$\frac{u ::= v}{u, v \in (\Sigma_T \cup \Sigma_N)^*}$$

- Tipo 1: Lenguajes sensibles al contexto

$$\frac{xAy ::= xvy}{A \in \Sigma_N, x, y \in (\Sigma_T \cup \Sigma_N)^*, v \in (\Sigma_T \cup \Sigma_N)^+}$$

- Tipo 2: Lenguajes libres de contexto

$$\frac{A ::= v}{v \in (\Sigma_T \cup \Sigma_N)^*, A \in \Sigma_N}$$

- Tipo 3: Lenguajes regulares

Lineales por la izquierda:

$$\frac{A ::= a}{A ::= VaS ::= \lambda}$$

Lineales por la derecha

$$\frac{A ::= a}{A ::= aVS ::= \lambda}$$

Donde  $a \in \Sigma_T$ ,  $A, V, S \in \Sigma_N$ ,  $S$  es el axioma de la gramática.

La representación de las producciones de la jerarquía de Chomsky, presentan algunos símbolos que denotan la cantidad de repeticiones que se pueden presentar haciendo uso de un símbolo (\*, +), estos símbolos pertenecen a las expresiones regulares. Las expresiones regulares son las unidades de descripción de los lenguajes regulares, que se incluyen en los denominados lenguajes formales (véase tabla 1). [3]

Tabla 1: caracteres especiales de expresiones regulares.

Caracter	Significado
*	El número del carácter, de la clase o del grupo situado antes del asterisco puede ser aleatorio (cero incluido)



XI CONGRESO INTERNACIONAL DE COMPUTACION MEXICO - COLOMBIA CICOM 2021

Octubre 20, 21 y 22 de 2021, Colombia

Table with 2 columns: Caracter and Significado. It defines symbols like +, ?, [], (), and {n} in the context of grammar rules for gesture recognition.

De acuerdo al comportamiento de las producciones se optó por el tipo 2 que son las de libre de contexto, ya que se trata de un lenguaje de programación y, por otra parte, la estructura de los fragmentos de código que se pretenden generar sigue este patrón.

Análisis de gramáticas

Se llevaron a cabo el análisis de dos tipos de gramáticas, una con un mayor número de restricciones, por lo cual significa que tiene un mayor número de producciones. Y otra con un número menor de restricciones. Ambas gramáticas tienen la siguiente forma:

G = (NT, T, Gesto, P)

Donde:

- T : Es un conjunto de terminales
• NT : Un conjunto de no terminales
• Gesto : Producción inicial
• P : Conjunto de producciones

La primera gramática contiene lo siguiente

NT= (mano, orientación, dedos, dedo, direccion1, direccion2, direccion3, direccion4, direccion5, direccion6, reglas1, reglas2, reglas3, reglas4, reglas5, reglas6, reglas7, reglas8, reglas9, reglas10, reglas11, reglas12, condicion1, condicion2, condicion3, condicion4, condicion5, condicion6, pulgar1, pulgar2, pulgar3, flexión, conjuntoDedos, tipoflexión, distancia, relación)
T = (izquierda, derecha, abajo, arriba, adelante, atrás, meñique, anular, medio, índice, pulgar, tocando, notocando, abierto, cerrado)
P = Producciones (véase figura 1)

```
gesto ::= mano orientacion dedos
mano ::= [1][2][3][4]
orientacion ::= [1][2][3][4][5][6]
dedos ::= [1][2][3][4][5][6]
```

Figura 1: Producciones de la primera gramática (parte parcial)
La segunda gramática utiliza las mismas tuplas y estas se conforman de la siguiente manera:

NT= (gesto, mano, palma, orientación, dirección, dedos, posición, flexión, tipoflexión, distancia, relación, dedo, dedos)
T= (izquierda, derecha, abajo, arriba, adelante, atrás, meñique, anular, medio, índice, pulgar, tocando, notocando, abierto, cerrado)
P = Producciones (véase Figura 2)

```
gesto ::= mano <palma> <dedos>*
mano ::= [1][2][3][4][5][6]
palma ::= [1][2][3][4][5][6]
orientacion ::= [1][2][3][4][5][6]
direccion ::= [1][2][3][4][5][6]
dedos ::= [1][2][3][4][5][6]
posicion ::= [1][2][3][4][5][6]
flexion ::= [1][2][3][4][5][6]
tipoflexion ::= [1][2][3][4][5][6]
distancia ::= [1][2][3][4][5][6]
relacion ::= [1][2][3][4][5][6]
```

Figura 2: Producciones de segunda gramática.

El usar una gramática muy extensa (véase figura 1) ayuda a tener un mayor control en las restricciones de lo que se quiere generar de terminales como resultado. Y el usar una gramática muy reducida (véase Figura 2) puede propiciar a valores que semánticamente no estarían correctos, pero a pesar de ello, añadir meñique arriba abierto tocando anular notocando med se tiene la parte de programación para arreglar ciertas incongruencias o código extra que se podría llegar a generar. Por esto se escogió la segunda gramática para ponerla a prueba respecto a los posibles gestos a tomar en cuenta.

Para poner a prueba la gramática existen varias herramientas que facilitan el uso al implementar gramáticas en lenguajes de programación, una de estas y la que se utilizó es la herramienta ANTLR4 enfocado en el lenguaje de programación C#.

ANTLR

De acuerdo a su sitio oficial ANTLR es un potente generador de analizadores sintácticos para leer, procesar, ejecutar o traducir texto estructurado o archivos binarios. Se usa ampliamente para crear lenguajes, herramientas y frameworks. A partir de una gramática, ANTLR se genera un analizador que puede construir y recorrer árboles de análisis. [4]



**XI CONGRESO INTERNACIONAL DE COMPUTACION  
MÉXICO - COLOMBIA  
CICOM 2021**

Octubre 20, 21 y 22 de 2021, Colombia

**Cadenas de producción**

Los gestos fueron capturados mediante un sensor de profundidad llamado RealSense SR300 (véase figura 3) el cual está diseñado para poder llevar a cabo la detección de manos. Este dispositivo proporciona un número de información que describe el comportamiento de las manos, dicha información previo a ser analizada por la gramática establecida, se desarrolló un proceso para hacer una transformación, en el cual se convierten los datos puntuales a cadenas de texto, que describen en palabras el comportamiento de la mano.



Figura 3: Sensor RealSense SR 300. [5]

Algunos ejemplos de producciones que son generadas por la gramática, se ejemplifican en las siguientes pruebas.

Prueba 1: La Figura 4 muestra el gesto a reconocer por medio de la gramática propuesta.



Figura 4: Ejemplo P1 de gesto capturado

La producción que se genera es la siguiente (véase tabla 2):

Tabla 2: Producciones de prueba 1

Producciones	
<Gesto>	::= <mano><palma><dedos>*
<mano>	::= derecha
<palma>	::= <orientacion>?<direccion>?
<orientacion>	::= adelante
<direccion>	::= arriba
<dedos>	::= <posicion> <interaccion>*

<posicion>	::= <dedo><direccion><flexion>
<dedo>	::= indice
<direccion>	::= arriba
<flexion>	::= abierto
<interaccion>	::= <relacion><dedo>*
<relacion>	::= tocando
<dedo>	::= medio
<interaccion>	::= <relacion><dedo>*
<relacion>	::= notocando
<dedo>	::= anular
<dedo>	::= menique
<dedo>	::= pulgar
<dedos>	::= <posicion> <interaccion>*
<posicion>	::= <dedo><direccion><flexion>
<dedo>	::= medio
<direccion>	::= arriba
<flexion>	::= abierto
<interaccion>	::= <relacion><dedo>*
<relacion>	::= tocando
<dedo>	::= indice
<dedo>	::= anular
<interaccion>	::= <relacion><dedo>*
<relacion>	::= notocando
<dedo>	::= menique
<dedo>	::= pulgar
<dedos>	::= <posicion> <interaccion>*
<posicion>	::= <dedo><direccion><flexion>
<dedo>	::= menique
<direccion>	::= arriba
<flexion>	::= abierto
<interaccion>	::= <relacion><dedo>*
<relacion>	::= tocando
<dedo>	::= anular
<interaccion>	::= <relacion><dedo>*
<relacion>	::= notocando
<dedo>	::= medio
<dedo>	::= indice
<dedo>	::= pulgar
<dedos>	::= <posicion> <interaccion>*
<posicion>	::= <dedo><direccion><flexion>
<dedo>	::= menique
<direccion>	::= arriba
<flexion>	::= abierto
<interaccion>	::= <relacion><dedo>*
<relacion>	::= tocando
<dedo>	::= anular
<interaccion>	::= <relacion><dedo>*
<relacion>	::= notocando
<dedo>	::= medio
<dedo>	::= indice
<dedos>	::= <posicion> <interaccion>*
<posicion>	::= <dedo><direccion><flexion>
<dedo>	::= pulgar
<direccion>	::= izquierda
<flexion>	::= abierto



**XI CONGRESO INTERNACIONAL DE COMPUTACION  
MÉXICO - COLOMBIA  
CICOM 2021**

Octubre 20, 21 y 22 de 2021, Colombia

```

<interaccion> ::= <relacion><dedo>*
<relacion> ::= tocando
<interaccion> ::= <relacion><dedo>*
<relacion> ::= notocando
<dedo> ::= indice
<dedo> ::= medio
<dedo> ::= anular
<dedo> ::= menique
    
```

Prueba 2: La Figura 5 muestra el gesto a reconocer por medio de la gramática propuesta.



Figura 5: Ejemplo P2 de gesto capturado

La producción que se genera es la siguiente (véase tabla 3):

Tabla 3: Producciones de prueba 2

Producciones	
<Gesto>	::= <mano><palma><dedos>*
<mano>	::= derecha
<palma>	::= <orientacion>?<direccion>?
<orientacion>	::= arriba
<direccion>	::= adelante
<dedos>	::= <posicion> <interaccion>*
<posicion>	::= <dedo><direccion><flexion>
<dedo>	::= indice
<direccion>	::= adelante
<flexion>	::= abierto
<interaccion>	::= <relacion><dedo>*
<relacion>	::= tocando
<dedo>	::= medio
<interaccion>	::= <relacion><dedo>*
<relacion>	::= notocando
<dedo>	::= anular
<dedo>	::= menique
<dedo>	::= pulgar
<dedos>	::= <posicion> <interaccion>*
<posicion>	::= <dedo><direccion><flexion>
<dedo>	::= indice
<dedo>	::= medio
<direccion>	::= adelante
<flexion>	::= abierto

```

<interaccion> ::= <relacion><dedo>*
<relacion> ::= tocando
<dedo> ::= indice
<interaccion> ::= <relacion><dedo>*
<relacion> ::= notocando
<dedo> ::= anular
<dedo> ::= menique
<dedo> ::= pulgar
<dedos> ::= <posicion> <interaccion>*
<posicion> ::= <dedo><direccion><flexion>
<dedo> ::= anular
<direccion> ::= atras
<flexion> ::= cerrado
<interaccion> ::= <relacion><dedo>*
<relacion> ::= tocando
<dedo> ::= menique
<interaccion> ::= <relacion><dedo>*
<relacion> ::= notocando
<dedo> ::= indice
<dedo> ::= medio
<dedo> ::= pulgar
<dedos> ::= <posicion> <interaccion>*
<posicion> ::= <dedo><direccion><flexion>
<dedo> ::= menique
<direccion> ::= atras
<flexion> ::= cerrado
<interaccion> ::= <relacion><dedo>*
<relacion> ::= tocando
<dedo> ::= anular
<interaccion> ::= <relacion><dedo>*
<relacion> ::= notocando
<dedo> ::= indice
<dedo> ::= pulgar
<dedos> ::= <posicion> <interaccion>*
<posicion> ::= <dedo><direccion><flexion>
<dedo> ::= derecha
<flexion> ::= abierto
<interaccion> ::= <relacion><dedo>*
<relacion> ::= tocando
<interaccion> ::= <relacion><dedo>*
<relacion> ::= notocando
<dedo> ::= indice
<dedo> ::= medio
<dedo> ::= anular
<dedo> ::= menique
    
```

De esta manera, las producciones generadas fueron analizadas con ayuda de la herramienta ANTLR mediante el uso de la gramática descrita anteriormente. Con la finalidad de obtener una estructura lo más adecuada posible para facilitar el proceso posterior que es la generación de código de dicho gesto de mano.



## XI CONGRESO INTERNACIONAL DE COMPUTACION MÉXICO - COLOMBIA CICOM 2021

Octubre 20, 21 y 22 de 2021, Colombia

### RESULTADOS

Cabe mencionar que las producciones de cada gesto deben de ir libres de puntuaciones y acentos, ya que el proceso del análisis es sensible a símbolos extras en los terminales.

Para evaluar la gramática se realizaron un total de 60 ejecuciones a 6 gestos distintos, cada uno conformado por 10 repeticiones, para verificar que las producciones fueran persistentes independientemente del número de ejecuciones (véase tabla 2).

Tabla 2: resultados de evaluación de gramática

Gesto	Ejecuciones	Errores	Aciertos
1	10	0	10
2	10	0	10
3	10	0	10
4	10	0	10
5	10	0	10
6	10	0	10

El porcentaje final de la evaluación contempla un 100% de exactitud en que la gramática esta aprobando las producciones que se están generando de los gestos capturados. En el caso de existir duplicaciones de palabras, la herramienta arroja errores de los terminales que se esperaban, obedeciendo así las condiciones establecidas en la gramática.

### CONCLUSIONES

El uso de una gramática formal libre de contexto de acuerdo a la jerarquía de Chomsky, es útil cuando se trabaja con lenguajes de programación de alto nivel.

A pesar de que la gramática definida es pequeña cumple con el funcionamiento que se requiere, ya que reacciona a caracteres

duplicados o atributos que no se esperan. Aun cuando pueda existir la posibilidad de presentar situaciones que semánticamente no tendrían sentido; queda pendiente la fase programable para auxiliar a la gramática en esas situaciones.

El enfoque que se le da a esta gramática es en el proceso de generación de código de gestos de las manos, debido a que donde se puede implementar esta forma de control humano-computadora es para poder desarrollar sistemas con interfaces naturales de usuarios, y así lograr diferentes áreas de aplicación como lo son: control de multimedia, operar sistemas de transporte, manejo de robots, atención médica, lenguaje de señas, diseño virtual 3d, videojuegos, entre otros.

### AGRADECIMIENTOS

Agradecemos el apoyo a: el CONACYT, el Tecnológico Nacional de México y en especial al campus CENIDET y al proyecto: 10437.21-P Identificación de alteraciones en el iris del ojo para detectar patrones que permitan medir si existe correlación con personas diagnosticadas con diabetes mellitus

### REFERENCIAS

- [1] Editorial Etecé (septiembre 6 2020). Gramática. Concepto. Obtenido de <https://concepto.de/gramatica/>
- [2] Gallego A. (2008). La jerarquía de Chomsky y la facultad del lenguaje: consecuencias para la variación y la evolución. Vol. XXVII/2, 2008, pp. 47-60 ISSN: 0210-1602
- [3] IONOS (diciembre 12 2019). Digital Guide IONOS. obtenido de: <https://www.ionos.mx/digitalguide/paginas-web/creacion-de-paginas-web/regex/>
- [4] Terence P. (2014). ANTLR. obtenido de: <https://www.antlr.org/>
- [5] Truong G. (diciembre 1 2017). grabcadcommunity. obtenido de: <https://grabcad.com/library/intel-realsense-sr300-3>