

Índice general

1. Introducción	11
1.1. Kinect™	11
1.2. OpenNI	12
1.3. OpenCV	12
1.4. Reconocimiento de Patrones	12
1.5. Etiquetas (Tag´s)	13
1.5.1. reacTIVision	13
1.5.2. Código QR	14
1.5.3. ARToolkit	15
1.6. Metodología	16
1.6.1. Paradigma	16
1.7. Objetivos	16
1.7.1. Problemática	16
1.7.2. General	16
1.7.3. Particulares	17
1.8. Justificación	17
1.8.1. Estado del arte	17
2. Análisis	21
2.1. Módulo 1: Editor básico de dibujo	21
2.1.1. Introducción	21
2.1.2. Objetivo	21
2.1.3. Análisis y descripción de procesos	21
2.2. Módulo 2: Reconocimiento de trazos a mano alzada	23
2.2.1. Introducción	23
2.2.2. Objetivo	23
2.2.3. Análisis y descripción de procesos	23
2.3. Módulo 3: Proyección sobre el área de trabajo	25
2.3.1. Introducción	25
2.3.2. Objetivo	25
2.3.3. Análisis y descripción de procesos	25
2.3.4. Mostrar trazos realizados	25
2.4. Módulo 4: Implementación de herramientas físicas	25
2.4.1. Introducción	25
2.4.2. Objetivo	25
2.4.3. Análisis y descripción de procesos	25
2.5. Estudio de Factibilidad.	26
2.5.1. Factibilidad operativa	26

2.5.2. Factibilidad técnica	26
2.5.3. Software actual	28
2.6. Requerimientos	29
2.7. Especificación de los Requerimientos	30
3. Diseño	43
3.1. Arquitectura del Sistema	43
3.2. Diagramas Generales del Sistema	44
3.2.1. Diagrama General de Casos de Uso.	44
3.2.2. Diagrama General de Clases.	64
3.2.3. Diagrama General de Secuencia.	65
3.2.4. Diagrama General de Estados.	66
3.3. Diagramas del Módulo 1	67
3.3.1. Diagrama de Casos de Uso - Módulo 1.	67
3.3.2. Diagrama de clases - Módulo 1.	68
3.3.3. Diagrama de Secuencia - Módulo 1.	69
3.3.4. Diagrama de Estados - Módulo 1.	70
3.4. Diagramas del Módulo 2	71
3.4.1. Diagrama de Casos de Uso - Módulo 2.	71
3.4.2. Diagrama de Clases - Módulo 2.	72
3.4.3. Diagrama de Secuencias - Módulo 2.	73
3.4.4. Diagrama de Estados - Módulo 2.	74
3.5. Diagramas del Módulo 3	75
3.5.1. Diagrama de Casos de Uso - Módulo 3.	75
3.5.2. Diagrama de Secuencia - Módulo 3.	76
3.5.3. Diagrama de Estados - Módulo 3.	77
3.6. Diagramas del Módulo 4	78
3.6.1. Diagrama de Casos de Uso - Módulo 4.	78
3.6.2. Diagrama de Secuencia - Módulo 4.	79
3.6.3. Diagrama de Estados - Módulo 4.	80
4. Desarrollo	81
4.1. Editor de dibujo básico.	81
4.2. Reconocimiento de trazos a mano alzada.	82
4.3. Implementación de herramientas físicas.	82
4.3.1. Elección de tag	82
4.3.2. Reconocimiento de Tag's	83
5. Pruebas	87
5.1. Pruebas - Módulo 1: Editor básico de dibujo	87
5.2. Pruebas - Módulo 2: Reconocimiento de trazos a mano alzada	89
5.3. Pruebas - Módulo 3: Proyección sobre el área de trabajo	90
5.4. Pruebas - Módulo 4: Implementación de herramientas físicas	91
6. Conclusiones	93
6.1. Generales	93
6.2. Individuales	93
6.3. Trabajo a futuro	94
7. Bibliografía	95

8. Anexo. Manual de Instalación OpenNI/OpenCV	99
8.1. Introducción	99
8.2. Requisitos previos	99
8.2.1. Sistema Operativo	99
8.2.2. Gestor de paquetes	100
8.2.3. Actualización de Repositorios	100
8.2.4. Instalación de Paquetería	100
8.3. Instalación de OpenNI y OpenCv	102
8.3.1. Pasos para la instalación de OpenNI	102
8.3.2. Pasos para la Instalación de OpenCV	109

Índice de cuadros

1.1. Elementos Principales de <i>Kinect</i> TM [2].	11
1.2. Capacidad de datos del código QR.	14
1.3. Capacidad de corrección de errores Código QR[9].	14
2.1. Descripción técnica Apple MacBook.	27
2.2. Descripción técnica Dell Inspiron modelo 1545.	27
2.3. Descripción técnica HP Pavillion modelo dv4.	27
2.4. Descripción técnica Kinect[2].	28
2.5. Software para el desarrollo del proyecto.	28
2.6. Comparativa entre sistemas operativos para desarrollar el proyecto.	28
2.7. Comparativa entre Windows y Ubuntu.	29
3.1. Inicializar Sistema.	45
3.2. Rotar Herramienta.	46
3.3. Reconocer Herramienta.	47
3.4. Seleccionar.	48
3.5. Herramienta Invalida.	49
3.6. Mover.	50
3.7. Cambiar Color.	51
3.8. Escalar.	52
3.9. Reconocer Rotación.	53
3.10. Realizar Rotación.	54
3.11. Proyectar.	55
3.12. Reconocer Desplazamiento.	56
3.13. Realizar Desplazamiento.	57
3.14. Dibujar.	58
3.15. Mano Alzada.	59
3.16. Línea Recta.	60
3.17. Elipse.	61
3.18. Circunferencia.	62
3.19. Polígono.	63
5.1. Pruebas realizadas al Módulo 1.	88
5.2. Pruebas realizadas al Módulo 2.	89
5.3. Pruebas realizadas al Módulo 3.	90
5.4. Pruebas realizadas al Módulo 4.	91
8.1. Paquetes de OpenNI.	102

8.2. Paquetes de OpenCV.	109
----------------------------------	-----

Índice de figuras

1.1. Ejemplos de marcadores para reacTable.	13
1.2. Algunas simples topologías y su correspondiente gráfico de región adyacente.	13
1.3. Ejemplo de Código QR.	14
1.4. Ejemplo de ARToolKit Marker.	15
1.5. Hand Tracking.	18
1.6. Reconocimiento del cuerpo.	18
1.7. Proyección de imágen sobre un área de trabajo.	19
1.8. Interacción con robot automática programable.	19
2.1. Módulo 1 - Dibujar a Mano Alzada.	30
2.2. Módulo 1 - Líneas de Diferente Longitud.	31
2.3. Módulo 1 - Líneas en Diferentes Direcciones.	32
2.4. Módulo 1 - Circunferencias de Distintos Radios.	32
2.5. Módulo 1 - Elipse de distinta dimensión.	33
2.6. Módulo 1 - Polígonos de Diferentes Tamaños.	34
2.7. Módulo 1 - Selección del tipo de Polígono.	34
2.8. Módulo 2 - Integración con Kinect.	35
2.9. Módulo 2 - Detección del Dedo.	36
2.10. Módulo 2 - Reconocer Desplazamiento del dedo índice.	36
2.11. Módulo 2 - Dibujar a Mano Alzada con el Dedo Índice.	37
2.12. Módulo 2 - Proyección en el editor básico de dibujo.	37
2.13. Módulo 3 - Trabajo Colectivo de los Dispositivos.	38
2.14. Módulo 3 - Proyección sobre el área de trabajo.	39
2.15. Módulo 3 - Trabajar a pesar de la sombra.	39
2.16. Módulo 4 - Reconocimiento de la herramienta (Tag).	40
2.17. Módulo 4 - Dibujar utilizando la Herramienta (tag).	41
2.18. Módulo 4 - Escalar Figura.	41
2.19. Módulo 4 - Mover Figura.	42
2.20. Módulo 4 - Cambiar de color.	42
3.1. Arquitectura - abstracción de capas.	43
3.2. Diagrama General de Casos de Uso.	44
3.3. Diagrama General de Clases.	64
3.4. Diagrama General de Secuencia.	65
3.5. Diagrama General de Estados.	66
3.6. Diagrama de Casos de Uso - Módulo 1.	67
3.7. Diagrama de clases - Módulo 1.	68
3.8. Diagrama de Secuencia - Módulo 1.	69

3.9. Diagrama de Estados - Módulo 1	70
3.10. Diagrama de Casos de Uso - Módulo 2	71
3.11. Diagrama de Clases - Módulo 2	72
3.12. Diagrama de Secuencias - Módulo 2	73
3.13. Diagrama de Estados - Módulo 2	74
3.14. Diagrama de Casos de Uso - Módulo 3	75
3.15. Diagrama de Secuencia - Módulo 3	76
3.16. Diagrama de Estados - Módulo 3	77
3.17. Diagrama de Casos de Uso - Módulo 4	78
3.18. Diagrama de Secuencia - Módulo 4	79
3.19. Diagrama de Estados - Módulo 4	80
4.1. Editor de dibujo básico	81
4.2. Vista de las Tag's	83
4.3. Tag segmentada	86
8.1. gestor de paquetes	100
8.2. actualizar repositorios	100
8.3. build-essential	101
8.4. Pagina de Descargas de OpenNi	103
8.5. Crear carpeta Kinect	103
8.6. Crear carpeta Kinect	103
8.7. Descarga Sensor Kinect	104
8.8. Sensor Kinect	104
8.9. Install.sh OpenNI	105
8.10. Install.sh Sensor	105
8.11. Capeta Data	105
8.12. Archivo xml sin licencia	106
8.13. Archivo xml con licencia	106
8.14. install.sh de Nite	107
8.15. RedistMaker	107
8.16. Permisos para RedistMaker	107
8.17. Permisos y ejecución del archivo install.sh	108
8.18. Ejecutar NiViewer	108
8.19. Mapa de profundidad	108
8.20. rmmod gspca kinect	109
8.21. Remover las versiones de ffmpeg y x264	110
8.22. Instalación de paquetes necesarios	110
8.23. Instalación de gstreamer	110
8.24. Configuración de x264	111
8.25. Compilación de x264	111
8.26. Configuración de ffmpeg	112
8.27. Compilación de ffmpeg	112
8.28. Instalación de gtk	113
8.29. Paquetes de QT	113
8.30. Paquete de vl4 panel	113
8.31. Paquetes de vl4	114
8.32. cmake-gui	115
8.33. Parte 1 Configuración de cmake	115
8.34. Parte 2 Configuración de cmake	116

8.35. Parte 3 Configuración de cmake.	116
8.36. Crear archivo.	117
8.37. Agregar líneas al archivo.	118

Capítulo 1

Introducción

El presente trabajo muestra el desarrollo de un sistema *multi-touch* para crear figuras básicas bidimensionales.

La interacción con el sistema contará con objetos físicos que representan herramientas para dibujar, además se cuenta con otros objetos como herramientas básicas para la edición del dibujo.

El desarrollo del sistema está enfocado en la integración de tecnologías, en éste caso particular se utilizará la herramienta *KinectTM* de *Microsoft®* conectada a una computadora personal, además el sistema contará con reconocimiento de patrones para la selección de herramientas de dibujo o de edición del mismo.

1.1. KinectTM

KinectTM es un dispositivo que combina una cámara *RGB*, un sensor de profundidad y un arreglo de micrófonos[1].

Como podemos observar en el Cuadro 1.1, se listan los elementos principales de *KinectTM* junto con su función.

Elemento	Función
Arreglo de micrófonos	Detecta las voces y las aísla del ruido ambiental
Proyector de luz infrarroja	Dispara luz infrarroja.
Sensor de profundidad cámara infrarroja	Detecta la luz que lanza el proyector de luz infrarroja y genera un canal extra al <i>RGB</i> que trae información de la profundidad de la escena y es similar a un mapa de disparidad estéreo.
Motor de inclinación	Ajustar hacia donde están dirigidas las cámaras.
Salida de adaptador <i>USB</i> 2.0	Conectar un cable para conectar vía <i>USB</i> el dispositivo.
Cámara <i>RGB</i>	Reconocer los tres colores básicos. Rojo, verde y azul.

Cuadro 1.1: Elementos Principales de *KinectTM*[2].

Además cuentan con:

- *Triple Core PowerPC 970, 3.2GHz, Hyperthreaded, 2 threads/core.*
- *500 MHz ATI graphics card.*
- *512 MB RAM.*

1.2. OpenNI

Se utiliza *OpenNI framework (Open Source Natural Interaction)* versión 1.3.4.6 que es una *API* desarrollada por *OpenNI organization*, que provee una interfaz para dispositivos que ofrecen una interfaz natural para operar los componentes del *middleware*[3], en nuestro sistema lo utilizamos para poder comunicar la computadora personal con *Kinect™* y así utilizar de la mejor manera para lo que se diseña el dispositivo *Kinect™*, que es eliminar los dispositivos físicos para controlar, en este caso, la computadora personal.

Posee una biblioteca de código abierto para poder trabajar con casi todas las capacidades de *Kinect™* en *Windows*, *Linux* y *Mac*.

1.3. OpenCV

OpenCV (Open Source Computer Vision Library) Es una biblioteca *GPL*, orientada a la computación visual en tiempo real, utilizada principalmente en el campo de la Interacción Computadora - Humano por sus siglas en inglés *HCI (Human Computer Interaction)* creada y soportada por *Intel* desde 1999, y tiene amplia documentación.

Actualmente trabajamos con la versión 2.3.1 de la librería *OpenCV*.

Una de las características más importantes de *OpenCV* es que las funciones están totalmente optimizadas para los procesadores de arquitectura *Intel*. Existen versiones para diferentes arquitecturas de procesadores y para diferentes sistemas operativos[4].

1.4. Reconocimiento de Patrones

Para intentar definir que es el Reconocimiento de Patrones (RP) definiremos algunos conceptos[5].

Reconocimiento: Proceso de clasificación de un objeto en una o más clases.

Objeto: Es un concepto con el cual representamos los elementos sujetos a estudio. Pueden ser concretos o abstractos.

Patrón: Tras los procesos de segmentación, extracción de características y descripción, cada objeto queda representado por una colección (posiblemente ordenada y estructurada) de descriptores.

Clase: Es un conjunto de objetos. Al agrupar en clases, se puede hacer de dos formas distintas:

- *Por pertenencias duras:* Un objeto pertenece o no a una clase.
- *Por pertenencias difusas:* Los objetos pertenecen parcialmente a una clase. Existen clases con intersecciones no vacías.

Existen varios intentos para definir al reconocimiento de patrones.

- La disciplina dedicada a la clasificación de objetos y el pronóstico de fenómenos.[6]
- Rama del conocimiento, de carácter multidisciplinario, cuyo objeto de estudio son los procesos de identificación, caracterización, clasificación y reconstrucción sobre conjuntos de objetos o fenómenos, así como el desarrollo de teorías, tecnologías y metodologías relacionadas con dichos procesos.[6]

- Es la ciencia que se ocupa de los procesos sobre ingeniería, computación y matemáticas relacionados con objetos físicos y/o abstractos, con el propósito de extraer información que permita establecer propiedades de o entre conjuntos de dichos objetos.[6]

Con respecto al reconocimiento de patrones, se cuenta con objetos físicos los cuales hacen la función de botones permitiendo eliminar éstos de la interfaz, éstos objetos tienen asignada una imagen binaria que denominamos *tag*(etiqueta), así la cámara de *KinectTM* captura la imagen del escenario, entonces el sistema procesa esa captura donde si se encuentra uno de los objetos físicos se decodifica la imagen y se activa la acción que se tenga asignada a dicha *tag*. Algunas de las *tags* tienen más de una acción que se activan con un giro en cierto ángulo esto hace que cambie la función que al principio tenía designada.

1.5. Etiquetas (Tag's)

Denominamos *tag's*(etiquetas) a un conjunto de imágenes binarias en colores blanco y negro que indica el uso de una herramienta.

Analizamos tres tipos de imágenes binarias candidatas, las cuales son: *reacTIVision fiducials*, Código QR y *AR-ToolKit Marker*.

1.5.1. reacTIVision

Son marcadores especiales (Figura 1.1) que se desarrollaron en conjunto con el sistema *reacTIVision* que es un *software* creado especialmente para el rastreo de éstos marcadores que a grandes rasgos son imágenes binarias, específicamente, en blanco y negro.



Figura 1.1: Ejemplos de marcadores para reacTable.

Para la identificación de cada marcador se basa en una región gráfica adyacente y los rectángulos de delimitación. El método combina coincidencias de patrones binarios de gráficos topológicos (Figura 1.2) para el reconocimiento y la identificación con simples técnicas geométricas para calcular la ubicación y orientación de los marcadores[7].

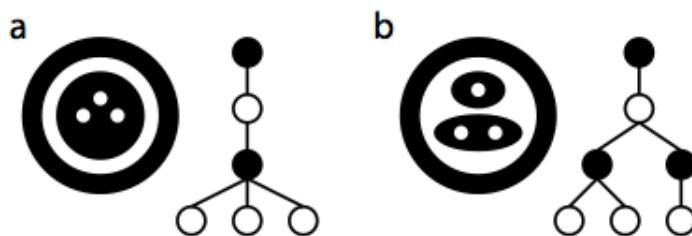


Figura 1.2: Algunas simples topologías y su correspondiente gráfico de región adyacente.

Se utilizan algoritmos genéticos para la identificación de cada marcador, para más detalles consultar[7].

Éstos marcadores están disponibles en PDF para su impresión así como el sistema *reacTIVision* en la página del proyecto[8] y no es necesario producir nuevos marcadores.

1.5.2. Código QR

El código de barras de respuesta rápida por sus siglas en inglés *QR code** (*Quick Response Barcode*, Figura 1.3) es un sistema que permite almacenar información en un código de barras bidimensional, esto quiere decir que tiene un patrón de arriba hacia abajo, de izquierda a derecha, y puede almacenar alrededor de 7,000 dígitos (véase el Cuadro 1.2) mucho más que un código de barras convencional, además con la ayuda de una cámara y un programa especial podemos recuperar la información de cada código. Éste código está estandarizado *ISO/IEC 18004*.



Figura 1.3: Ejemplo de Código QR.

Numérico	Máximo 7,089 caracteres.
Alfanumérico	Máximo 4,296 caracteres.
Binario	Máximo 2,953 caracteres.
Kanji/Kana	Máximo 1,817 caracteres.

Cuadro 1.2: Capacidad de datos del código QR.

Existen versiones del código QR desde la 1 hasta la 40 y cada una tiene diferentes números de módulos (módulo se refiere a los puntos blancos y negros que conforman el código QR)[9].

* QR code es una marca registrada por DENSO WAVE INCORPORATED

Tiene la capacidad de corrección de errores (véase el Cuadro 1.3), si una parte del código está dañada, manchada o doblada puede ser interpretado de igual forma.

QR Code Error Correction Capability	
Level L	Approx.7 %
Level M	Approx.15 %
Level Q	Approx.25 %
Level H	Approx.30 %

Cuadro 1.3: Capacidad de corrección de errores Código QR[9].

La decodificación del código *QR* puede seguir varios algoritmos a continuación se describe un algoritmo general que puede utilizarse para algunos código de barras en 2D.

1. Binarización de la imagen.
Método de Otsu[10].
2. Corrección de la inclinación.
3. Corrección geométrica de la imagen.
4. Obtención de los cuatro vértices de la imagen.
5. Obtención de los nuevos valores de los vértices.
6. Obtener el valor en cada nuevo pixel.
7. Normalización de la imagen.

Existen distintos sistemas que ofrecen la creación de códigos *QR* así como la decodificación del mismo como *ZXing*[11].

1.5.3. ARToolkit

Son plantillas de forma cuadrada, que se componen de cuadrado negro con un cuadrado blanco cuatro veces más pequeño que su centro y un dibujo sencillo en el interior del cuadrado blanco, como se muestra en la figura 1.4.

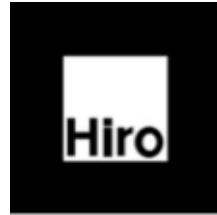


Figura 1.4: Ejemplo de ARToolKit Marker.

Para identificación de la plantilla está basada en la detección de las esquinas con ayuda del algoritmo de *fast pose estimation*[12]. Los pasos para el tratamiento de estas plantillas son los siguientes:

1. La imagen capturada se transforma a una imagen binaria.
2. Identificamos el marco de color negro.
3. Extraemos los patrones del dibujo que se encuentra en el interior del marco negro.
4. Almacenamos los patrones.
5. Repetimos los primeros tres pasos.
6. Comparamos los patrones extraídos con los almacenados.
7. Aplicamos funcionalidad de la imagen.

1.6. Metodología

El desarrollo del proyecto se realizó aplicando el modelo incremental.[13] Esta metodología tiene la ventaja de ser dinámica y flexible, además permite usar las salidas de las etapas precedentes, como entradas en las etapas sucesivas y facilita corregir cualquier error detectado o llevar a cabo mejoras en los distintos productos que se generan a lo largo de su aplicación[13].

Esta metodología, se basa en la metodología en cascada. El uso de esta metodología dentro del desarrollo del proyecto proporciona:

- Definición de actividades a llevarse a cabo en el tiempo de realización del Trabajo Terminal.
- Unificación de criterios en la organización para el desarrollo del proyecto.
- Puntos de control y revisión.
- Seguimiento de secuencias ascendentes o descendentes en las etapas del desarrollo.
- Cumplimiento de etapas o fases en paralelo, por lo que es más flexible que la estructurada.

1.6.1. Paradigma

El paradigma será Orientado a Objetos, porque la *API* usada de *OpenCV* está en lenguaje *C++*, que permite la manipulación de objetos, ya que primero definen objetos, para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

El uso del paradigma proporciona:

- No modela la realidad, sino la forma en que las personas comprenden y procesan la realidad.
- Es un proceso ascendente basado en una abstracción de clases en aumento.
- Se basa en identificación de objetos, definición y organización de librerías de clases, y creación de macros para aplicaciones específicas.
- Utiliza menor cantidad de código.
- Es reutilizable.

1.7. Objetivos

1.7.1. Problemática

Ofrecer una alternativa al teclado y mouse con una interfaz de usuario natural donde ya no se interactúe directamente con un dispositivo electrónico, junto con esto querer desarrollar algún sistema que probara que se podía utilizar *KinectTM* junto con la computadora personal.

1.7.2. General

Desarrollar un sistema con una interfaz de usuario natural, asistido con la herramienta de entretenimiento *KinectTM* y reconocimiento de patrones.

1.7.3. Particulares

- Lograr una configuración para usar *Kinect*TM con la computadora personal, para desarrollar un sistema.
- Implementar una interfaz sin involucrar dispositivos electronicos como una pantalla tactil.
- Eliminar uso del *mouse* en nuestro sistema.
- Eliminar uso del teclado en nuestro sistema.

1.8. Justificación

Una de las principales características sobre la dificultad del desarrollo de los sistemas *multi-touch* basado en tecnología “reciente” en el caso de *Kinect*TM era la falta de documentación fiable al momento de plantear este proyecto (Octubre - Diciembre 2011) ya que no se contaba con drivers capaces de explotar todas las características de *Kinect*TM ni un entorno de desarrollo estable por parte de *MS* o de la comunidad de *software libre*. El desarrollo de éste sistema busca contribuir a la creación de documentación formal que permitirá que futuras generaciones tengan mayor cantidad de fuentes fiables y por lo tanto se interesen por la creación de sistemas basados en movimientos, logrando ser un aportador más al crecimiento de dichos entornos.

Además, permitir que los alumnos de la Escuela Superior de Cómputo que se encuentren cursando o tengan interés en el reconocimiento de patrones o semejantes, trabajen con los actuales dispositivos de captura de imagen, siendo en nuestro caso *Kinect*TM de *Microsoft®*, dejando a un lado su complejidad y crear un mayor interés, buscando cambiar el enfoque de dicha herramienta en donde el alumno no la vea como un proyecto de Trabajo Terminal sino como prácticas semestrales, lo que brindará mayor competitividad e integración de nuevas tecnologías.

Esta integración de tecnologías ofrece una alternativa al *mouse* y al teclado pudiendo así evitar enfermedades ya conocidas causadas por éstos dispositivos como lo es el síndrome de túnel carpiano[14].

1.8.1. Estado del arte

Actualmente no se cuenta con un sistema de dibujo como el que se pretende realizar, a la fecha de la documentación del estado del arte (Marzo 2012) existen otros trabajos que también manejan *Kinect*TM, *OpenCV* y *OpenNI*, elementos con los que se llevará acabo el desarrollo de nuestro sistema, de los cuales vamos a mencionar algunos a continuación.

Gracias a la aparición de *drivers* que permiten la interacción entre el dispositivo *Kinect*TM, que primeramente era exclusivo para la consola de videojuegos *Xbox 360* de *Microsoft®*, y la computadora, se comenzaron a realizar aplicaciones que permiten al usuario tener una interacción más natural, permitiendo ser ellos mismos el control de la aplicación. Debido a que era una tecnología “reciente” existía poca documentación formal acerca de proyectos relacionados.

Hand Tracking - Kinect with OpenCV 2.2 and OpenNI

Es una aplicación sencilla que en primera instancia reconoce la mano de un usuario, como un punto permitiendo realizar trazos a mano alzada (Figura 1.5), la aplicación puede cambiar el punto con que se realizar el trazo entre una mano y otra juntando las manos para volverlas a separar así queda realizado el cambio. Esta aplicación también permite la identificación del cuerpo (Figura 1.6) con lo que se toman a ambas manos como puntos de interés, uno de ellos se encarga de realizar el trazo y con el otro se puede seleccionar el color.

La relación que existe entre este trabajo y el que se pensó realizar es que ambos deben poder generar dibujos identificando un punto de interés con el cual se van a hacer los trazos además de seleccionar el color y agregar otra

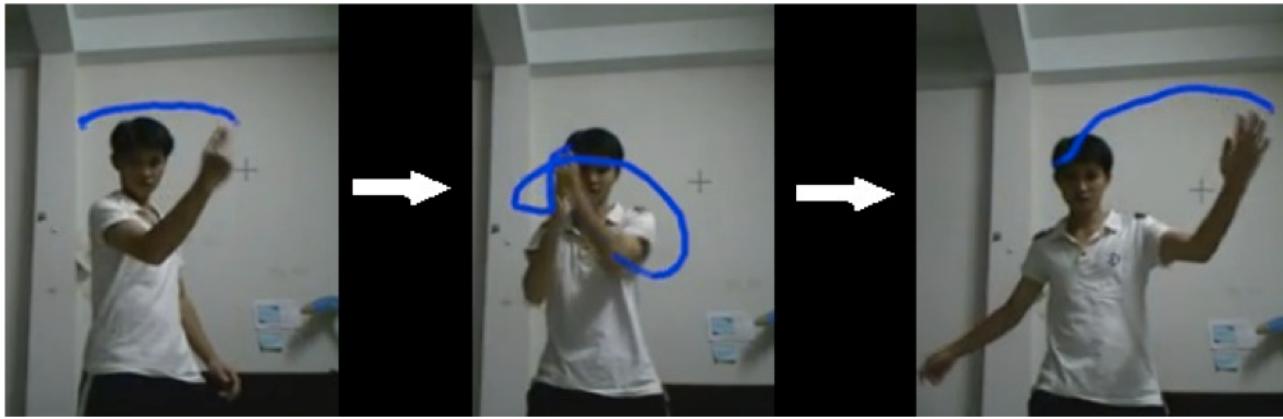


Figura 1.5: Hand Tracking.

funcionalidades. Éste sistema no cuenta con una documentación y lo único que se puede obtener es lo que se visualiza en un video subido a la red[15].

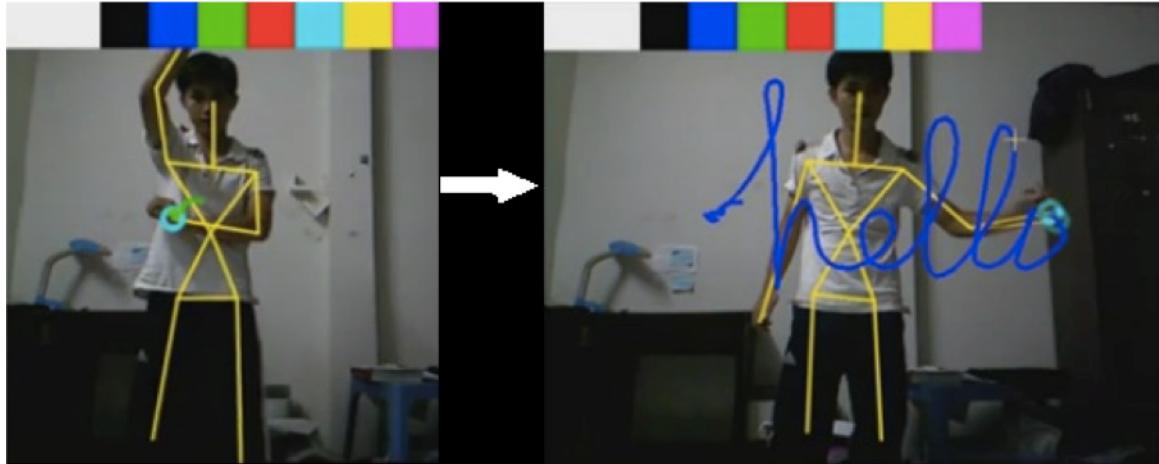


Figura 1.6: Reconocimiento del cuerpo.

Kinect Active Projection Mapping

Es una aplicación que trabaja con *KinectTM*, *OpenCV* y *OpenNI*, además comparte la idea de mantener un área de trabajo y una proyección, de tal modo el usuario interactúa directamente sobre el área designada (Figura 1.7).

En este proyecto se utiliza el kinect para reconocer el cuerpo, la posición de las manos principalmente. El sistema crea un efecto visual en sobre las manos y entre ellas por medio de una imagen que es proyectada sobre una pantalla detrás de el usuario[16].

Esta aplicación ha sido desarrollada en el *Computer Fusion Laboratory* como parte del programa de ingeniería de la *Temple University*. La página donde se dan más detalles del proyecto se encuentra aún en construcción. Y por el momento los recursos no están disponibles.



Figura 1.7: Proyección de imagen sobre un área de trabajo.

Aldebaran Nao Kinect Controller

Es un proyecto donde se controla por medio de *Kinect*TMa un robot (Figura 1.8), organismo autónomo programable y de mediana estatura desarrollado por la empresa Francesa *Aldebaran Robotics*. Esta aplicación ha sido desarrollada en *Technical University Bergakademie Freiberg* en Alemania por Erik Berger y Heni Ben Amor. Existe información adicional de este proyecto en la página oficial de la universidad [17], Pero se encuentra en idioma Alemán.

Este proyecto no tiene mucha relación en cuanto a la funcionalidad del trabajo que se desea realizar, pero se considera por el hecho de también emplear los elementos que utilizaremos en nuestro sistema.



Figura 1.8: Interacción con robot automática programable.

Capítulo 2

Análisis

En este capítulo se describe en análisis realizado para la creación del sistema. El análisis se presenta según los cuatro módulos que se reconocieron en un principio:

1. Editor Básico de Dibujo
2. Reconocimiento de trazos a mano alzada
3. Proyección sobre el Área de trabajo
4. Implementación de las herramientas físicas para el dibujo.

Con el análisis correspondiente a los módulos tres y cuatro se fijarán las especificaciones necesarias, además se mencionarán todas aquellas problemáticas detectadas en los procesos.

Una vez que se tiene los procesos por módulo, se mostrará el estudio de factibilidad que se realizó al correspondiente proyecto, para ver la disponibilidad de los recursos que necesitaron en la realización del sistema, posteriormente se expondrá la definición de requerimientos, y finalizar este capítulo con la especificación de requerimientos.

2.1. Módulo 1: Editor básico de dibujo

2.1.1. Introducción

En las sub-secciones siguientes, se definen los procesos que se llevan a cabo en el modulo uno, el cual se estructura de funciones básicas de dibujo, las cuales son: mano alzada, línea, circunferencia, elipse y polígonos (3 a 6 lados), así como detectar los problemas que puedan surgir.

2.1.2. Objetivo

Analizar los procesos necesarios para el desarrollo de un editor de dibujo y detectar posibles problemas por medio de la realización de pruebas que se documentan para precisar los requerimientos y redactar una propuesta con bases firmes y confiables.

2.1.3. Análisis y descripción de procesos

A continuación se describen los procesos correspondientes al módulo uno

Trazo a Mano Alzada

Objetivo: Realizar un trazo a mano alzada, es decir a pulso.

Descripción: Es un trazo que no requiere de reglas ni herramientas de medición exactas o auxiliares, solo con el movimiento de tu muñeca o pulso, P_n son el conjunto de puntos que conforman el trazo. Dado el conjunto de puntos $P_n(x_n, y_n)$ se encenderá un pixel por cada punto.

Datos de entrada: $P_n(x_n, y_n)$

Datos de salida: Trazo a mano alzada.

Problemas: la velocidad de desplazamiento al aumentar reduce el conjunto de puntos.

Trazar una Línea recta

Objetivo: Realizar trazos rectos a partir de dos puntos.

Descripción: Es la sucesión continua de puntos en una misma dirección, donde $P_1(x_1, y_1)$ es la posición de partida y $P_2(x_2, y_2)$ es la posición final. Dados los puntos $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$ se traza una línea entre estos por medio de la siguiente ecuación.

$$y - y_1 = \frac{(y - y_1)}{(x - x_1)}(x - x_1)$$

Datos de entrada: $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$

Datos de Salida: Línea recta entre $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$

Problemas: Se puede apreciar una pequeña deformación en la línea, al no poder segmentar un pixel.

Trazar una Circunferencia

Objetivo: Dibujar una superficie plana limitada por una circunferencia.

Descripción: Es una curva cerrada y plana en la que todos sus puntos están a la misma distancia, de otro punto fijo, que llamamos centro. Dados los puntos $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$ se traza un lugar geométrico de los puntos de un plano que equidistan de otro punto fijo y coplanario llamado centro en una cantidad constante llamado radio.

$$(y - y_1)^2 + (x - x_1)^2 = r^2$$

Datos de entrada: $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$

Datos de salida: Circunferencia entre $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$

Problemas: Se puede apreciar una deformación mayo cuando la distancia entre los puntos sea menor, al ser pixeles de forma cuadrada.

Trazar un Polígono

Objetivo: Dibujar una figura plana compuesta por una secuencia de segmentos rectos.

Descripción: Es una figura plana compuesta por una secuencia finita de segmentos rectos consecutivos que cierran una

región en el espacio. Dados dos puntos $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$, se genera una circunferencia la cual se divide entre el numero de lados obteniendo así intersecciones llamadas vértices, las cuales se unen atreves de segmentos de línea recta.

Datos de entrada: $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$

Datos de salida: Polígono entre $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$

Problemas: Se puede apreciar una deformación, cuando se le da una dimensión, al no poder segmentar un pixel.

Trazar una elipse

Objetivo: Es una circunferencia aplastada, una curva simétrica cerrada. Dibujar una curva plana y cerrada, simétrica respecto a dos ejes perpendiculares entre sí.

Descripción: dados los puntos $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$ se traza una curva plana cerrada que es simétrica respecto a dos ejes, los cuales constan de focos (puntos fijos), eje focal (recta que pasa por los focos), centro (punto de intersección de los focos) y de los ejes con la siguiente ecuación.

$$\frac{(x-x_1)^2}{a^2} + \frac{(y-y_1)^2}{b^2} = 1$$

Datos de entrada: $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$

Datos de salida: Elipse entre $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$

Problemas: Se puede apreciar una deformación mayor cuando la distancia entre los puntos sea menor, al ser pixeles de forma cuadrada.

2.2. Módulo 2: Reconocimiento de trazos a mano alzada

2.2.1. Introducción

En las sub-secciones de este apartado, se definen los procesos que se llevan a cabo en el modulo dos, donde se reconoce el trazo a mano alzada por medio de *Kinect*TM, dicho trazo será realizado por el usuario en el área de trabajo que estará delimitada, así como detectar los problemas que se puedan manifestar en los procesos.

2.2.2. Objetivo

Analizar los problemas que surjan en los procesos que involucra la realización de éste módulo, de igual manera se documentarán las pruebas hechas con esto corregiremos errores para cumplir con los requerimientos establecidos.

2.2.3. Análisis y descripción de procesos

Este apartado describe los procesos pertinentes para la implementación del módulo dos.

Integración del KinectTM

Objetivo: la computadora debe reconocer el dispositivo *Kinect*TM.

Descripción: Mediante el *driver OpenNI* se hará la comunicación entre el dispositivo y la computadora personal, para poder realizar capturas de imágenes, utilizar y procesar la información recibida.

Datos de entrada: Datos recibidos por el sensor.

Datos de salida: Detección exitosa del *Kinect*TM.

Problemas: No poder reconocer el dispositivo.

Detección de Imagen

Objetivo: Detectar la imagen infrarroja mediante el dispositivo *Kinect*TM.

Descripción: Se capturarán los datos de profundidad de la escena por medio de la cámara infrarroja.

Datos de entrada: Datos de profundidad.

Datos de salida: Imagen de la escena con profundidad.

Problemas: Si la imagen se sale del rango para el reconocimiento no se podrá detectar.

Reconocimiento del desplazamiento

Objetivo: Detectar que un punto de interés realiza un desplazamiento.

Descripción: Mediante el *Kinect*TM se captura como se desplaza el punto de interés siguiéndolo con un marcador que enfoque esa área.

Datos de entrada: $P_1(x_1, y_1)$

Datos de salida: $P_n(x_n, y_n)$

Problemas: si la velocidad de desplazamiento aumenta drásticamente se reduce la captura de los puntos.

Dibujar a mano alzada con el dedo

Objetivo: Poder plasmar el movimiento del área de interés en un trazo.

Descripción: Se levanta la mano y se inicia un movimiento libre dentro del rango de visión de *Kinect*TM, procesándolos y reflejándolos en el encendido de los pixeles del trazo sobre el editor.

Datos de entrada: $P_1(x_1, y_1)$

Datos de salida: $P_n(x_n, y_n)$, trazo realizado.

Problemas: Si la velocidad de desplazamiento varía de la velocidad de procesamiento, el trazo realizado podría no ser igual al movimiento del dedo.

2.3. Módulo 3: Proyección sobre el área de trabajo

2.3.1. Introducción

En la sub-sección del modulo 3, se definen los procesos que se llevan a cabo en el modulo, el cual se refiere a la imagen que nos proporcionará un proyector en el área de trabajo, donde el usuario podrá visualizar los trazos realizados. Así mismo se pretende detectar problemas en los procesos.

2.3.2. Objetivo

Analizar los procesos pertinentes del módulo 3 por medio de distintas configuraciones tanto del proyector como de *KinectTM* para el desarrollo del mismo cumpliendo completamente con el requerimiento establecido.

2.3.3. Análisis y descripción de procesos

Este apartado describe los procesos correspondientes al módulo tres.

2.3.4. Mostrar trazos realizados

Objetivo: Poder vizualizar en el área de trabajo los trazos realizados.

Descripción: Se capturan los movimientos del dedo en una secuencia de video continuo procesándolos y reflejándolos en el encendido de los pixeles de acuerdo al trazo sobre el editor.

Datos de entrada: Secuencia de video continúo.

Datos de salida: $P_n(x_n, y_n)$, trazo realizado.

Problemas: Si la velocidad de desplazamiento varía de la velocidad de procesamiento, el trazo realizado podría no ser igual al movimiento del dedo.

2.4. Módulo 4: Implementación de herramientas físicas

2.4.1. Introducción

En la sub-sección siguiente, se definen los procesos que se llevan a cabo en el modulo 4, en el cual se implementan las *tags*, con las que el usuario puede realizar un trazo ó hacer cambios en el dibujo, dichas opciones de edición son: cambiar de color, mover de posición y escalar el dibujo. Así como detectar los problemas que surjan en los procesos.

2.4.2. Objetivo

Analizar los procesos involucrados para el reconocimiento de las *tags* aplicado lo investigado sobre distintos algoritmos para reconocer la imagen binaria elegida y tener una implementación completa de acuerdo a los requerimientos.

2.4.3. Análisis y descripción de procesos

Este apartado describe los procesos correspondientes al desarrollo del módulo cuatro.

Selección de herramienta de trabajo

Objetivo: Identificar la *tag* colocada para iniciar un trazo o la edición de un dibujo.

Descripción: Se captura imagen de la *tag* y se procesa, de acuerdo a que *tag* este colocada se internamente se selecciona la opción correspondiente para iniciar el trazo de un dibujo o la edición del mismo.

Datos de entrada: Secuencia de video continuo.

Datos de salida: No hay datos de salida visibles al usuario, internamente queda seleccionada una opción.

Problemas: La *tag* puede no estar posicionada correctamente para su identificación.

2.5. Estudio de Factibilidad.

Después de definir la problemática presente y establecer las consideraciones de hardware y software, es conveniente realizar un estudio de factibilidad para el desarrollo del sistema, se muestra el análisis técnico y operativo que implica la implementación del sistema.

2.5.1. Factibilidad operativa

Para un mejor alcance como proyecto se desarrollará una aplicación de fácil uso, de tal forma que sin mucha dificultad, un usuario no experimentado, pueda adaptarse y aprovechar al máximo las facilidades que este brinde. Los usuarios podrán visualizar la información que ellos soliciten, al interactuar con el editor y las herramientas conjuntamente con el *KinectTM*, claro que solo con las operaciones mencionadas, no se podrá hacer algo que no esté indicado en el sistema.

Al ir implementando módulo por módulo, facilitará el trabajo para el avance de los posteriores módulos, y darse una idea de las posibles *tag's* que se necesitarán usar para el Trabajo Terminal. Estamos conscientes de aceptar los cambios y mejoras que el trabajo terminal ofrezca dentro del periodo de realización, llegando a la conclusión de que el sistema es factible operativamente, ya que se cuenta con la aceptación y la tecnología para desarrollar el sistema con éxito.

2.5.2. Factibilidad técnica

Actualmente para la realización del proyecto se cuenta con tres *laptops*, un *KinectTM* y un proyector, prestado por la Escuela Superior de Computo (ESCOM). Estos equipos de cómputo se utilizaron para desarrollar y hacer pruebas colectivamente con el *KinectTM*, aunque en cuestiones con el dispositivo solo es necesario un equipo de cómputo.

De acuerdo a la tecnología necesaria para la implementación del Sistema se evaluó bajo dos enfoques: *hardware* y *software*.

En cuanto a *hardware* existente, no se requirió realizar inversión inicial para la adquisición del mismo, ya que con lo que se contaba satisfacía los requerimientos establecidos, tanto para el desarrollo y puesta en funcionamiento del sistema propuesto.

A continuación se muestra la descripción de las computadoras personales (Cuadros 2.1, 2.2 y 2.3) que se utilizaron para la realización del sistema, así como las características del *kinect*(Cuadro 2.4):

- Una *Apple MacBook*

Procesador	Intel Core 2 Duo
Velocidad del Procesador	2,26 GHz
Numero de Procesadores	1
Memoria RAM	2 GB

Cuadro 2.1: Descripción técnica Apple MacBook.

- Una *Dell Inspiron 1545*

Procesador	Pentium Dual-Core
Velocidad del Procesador	2,20 GHz
Numero de Procesadores	1
Memoria RAM	3 GB

Cuadro 2.2: Descripción técnica Dell Inspiron modelo 1545.

- Una *HP Pavilion dv4*

Procesador	AMD Athlon II Dual Core
Velocidad del Procesador	2,00 GHz
Numero de Procesadores	1
Memoria RAM	3 GB

Cuadro 2.3: Descripción técnica HP Pavillion modelo dv4.

- Dispositivo *Kinect*TM

Triple Core PowerPC 970, 3,2GHz, Hyperthreaded, 2 threads/core.
500 MHz ATI graphics card.
512 MB RAM
Arreglo de micrófonos
Proyector de luz infrarroja
Sensor de profundidad cámara infrarroja
Motor de inclinación
Salidad de adaptador USB
Camara RGB

Cuadro 2.4: Descripción técnica Kinect[2].

2.5.3. Software actual

En cuanto al software, no amerita una inversión alguna, ya que el sistema operativo, el *driver* y la *API* son versiones libres, las versiones que se utilizarán se mencionan en el Cuadro 2.5:

Sistema operativo:	Linux Ubuntu 11,04 y versiones posteriores
Driver:	OpenNI versión 1,3,4,6
API para visión por computadora:	OpenCv versión 2,3,1

Cuadro 2.5: Software para el desarrollo del proyecto.

Comparativa de Sistemas Operativos (S.O)

En cuánto al sistema operativo, se hizo una tabla comparativa (Cuadro 2.6) de 3 sistemas diferentes, *Windows 7*, *Ubuntu(Linux)* y *MacOSX*, donde se notan las herramientas, *driver*, *API*'s y lenguajes que pueden ser usados en cada sistema, y ver cuál se acopla a lo que se hará, así como al *API* y *driver* seleccionados.

Sistema Operativo	Driver OpenNi	Middleware o utilidades	IDE	Lenguaje	API para visión por computadora
Windows 7	✓	Nite	Visual Studio 2008 en adelante, Eclipse, CodeBlocks	C ++, Phyton	Processing
Ubuntu	✓	Nite	QT, Eclipse, (Otros IDE's)	C ++, Phyton	OpenCV
Mac OSX	✓	Nite	XCode	C ++, Phyton	OpenCV

Cuadro 2.6: Comparativa entre sistemas operativos para desarrollar el proyecto.

Windows vs Ubuntu (Linux)

Al tener esta comparativa de los sistemas operativos, se empezó a descartar opciones, para lo cual el sistema de *Mac* fue el primero, ya que los equipos de computo con los que se contaba, solo se tenía un equipo de *Mac*, por lo cual se hizo otra tabla (Cuadro 2.7) con las características de los otros dos sistemas, y ver cual tenía más ponderación.

Característica	Windows	Linux Ubuntu
Familiaridad	Si	Si
Seguridad	No	Si
Precio	No	Si
Drivers	Si	Si

Cuadro 2.7: Comparativa entre Windows y Ubuntu.

En conclusión se utilizará la distribución *Ubuntu* del sistema operativo *Linux*, ya que es un software libre y no requiere de herramientas propietarias; el *Driver OpenNi* que nos proporciona el middleware Nite, el cual es soportado por la empresa *PRIMESENSE*, la cual desarrolló el dispositivo *Kinect™* y *OpenCv* que es una *API* para visión por computadora y se utilizado en proyectos con *Kinect™*.

2.6. Requerimientos

R1: El módulo 1 del sistema permite dibujar a mano alzada.

R2: El módulo 1 del sistema permite dibujar líneas de diferente longitud.

R3: El módulo 1 del sistema permite crear líneas en diferente dirección (ubicación).

R4: El módulo 1 del sistema permite dibujar círculos de diferente circunferencia.

R5: El módulo 1 del sistema permite dibujar una elipse de distintas dimensiones.

R6: El módulo 1 del sistema permite dibujar polígonos de diferentes tamaños.

R7: El módulo 1 del sistema permite dibujar polígonos de 3 a 6 lados dependiendo de la *tag* utilizada.

R8: El módulo 2 del sistema permite la integración con *Kinect™*.

R9: El módulo 2 del sistema permite la detección del dedo índice.

R10: El módulo 2 del sistema permite reconocer el desplazamiento del dedo índice.

R11: El módulo 2 del sistema permite dibujar a mano alzada con el dedo índice.

R12: El módulo 2 del sistema permite proyectar en el editor básico de dibujo la acción realizada por el dedo índice.

R13: El módulo 3 del sistema permite trabajar conjuntamente proyector y *Kinect™*.

R14: El módulo 3 del sistema permite proyectar sobre el área de trabajo.

R15: El módulo 3 del sistema permite trabajar sin interferencia de la sombra que produzca la mano.

R16: El módulo 4 del sistema permite reconocer la herramienta (*tag*).

R17: El módulo 4 del sistema permite dibujar con la herramienta (*tag*) de la figura a usar.

R18: El módulo 4 del sistema permite realizar la selección de escalar una figura.

R19: El módulo 4 del sistema permite realizar la selección de mover la figura.

R20: El módulo 4 del sistema permite realizar la selección de cambiar el color de la figura.

2.7. Especificación de los Requerimientos

R1: El módulo 1 del sistema permite dibujar a mano alzada.

Objetivo: El módulo permite al usuario realizar un trazo a mano alzada, es decir a pulso.

Descripción: Permite el dibujo a mano alzada, el cual se realiza simplemente con el dedo, u otro instrumento que lo simule, es dibujar a pulso, como se muestra en la Figura 2.1.

Datos de entrada: posición inicial y final.

Datos de salida: Trazo realizado.

Pre-condiciones: Posición de partida.

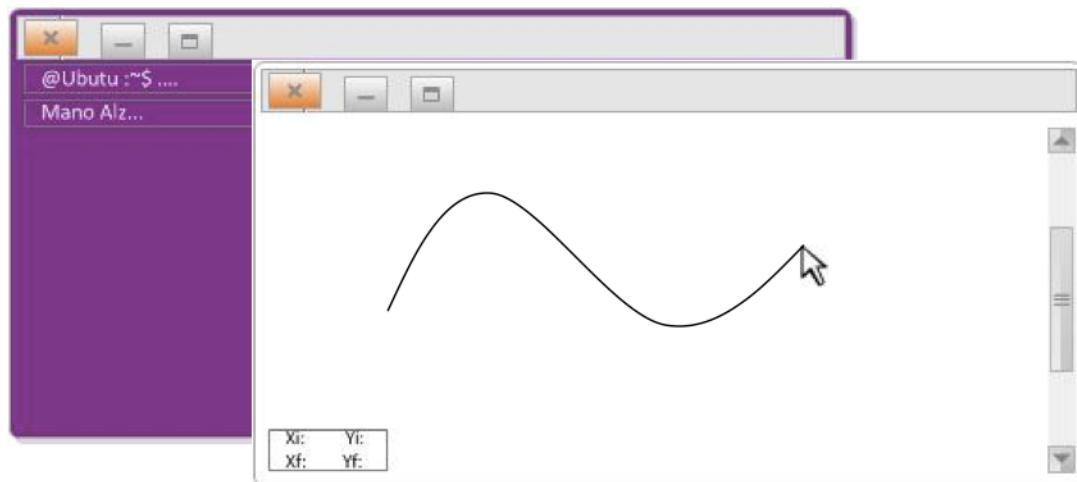


Figura 2.1: Módulo 1 - Dibujar a Mano Alzada.

R2: El módulo 1 del sistema permite dibujar líneas de diferente longitud.

Objetivo: Realizar trazos rectos sueltos y dinámicos .

Descripción: Permite el dibujo de una sucesión continua de puntos (trazado) con diferentes longitudes, según la desea por el usuario, como se muestra en la Figura 2.2, el cual se realiza simplemente con el dedo, u otro instrumento que lo simule, es dibujar a pulso.

Datos de entrada: posición inicial y final.

Datos de salida: Trazo realizado (línea).

Pre-condiciones: Posición de partida.

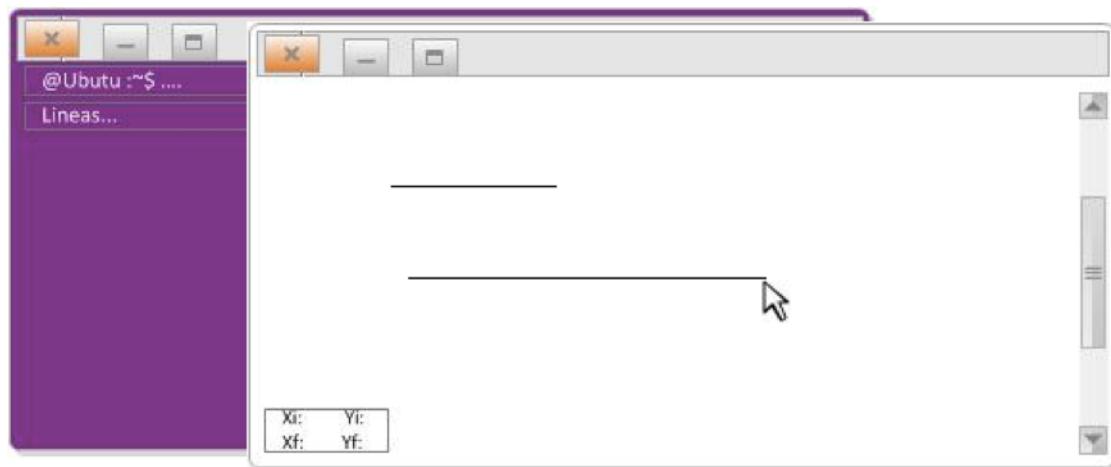


Figura 2.2: Módulo 1 - Líneas de Diferente Longitud.

R3: El módulo 1 del sistema permite crear líneas en diferente dirección (ubicación).

Objetivo: Realizar un trazo recto suelto en una cierta posición.

Descripción: Permite el dibujo de una sucesión de puntos (trazado) en cierta posición de coordenadas (X_1, Y_1) iniciales y (X_2, Y_2) finales, según donde se sitúe el usuario, como se muestra en la Figura 2.3, el cual se realiza simplemente con el dedo, u otro instrumento que lo simule, es dibujar a pulso.

Datos de entrada: posición inicial y final.

Datos de salida: Trazo realizado (línea).

Pre-condiciones: Posición de partida.

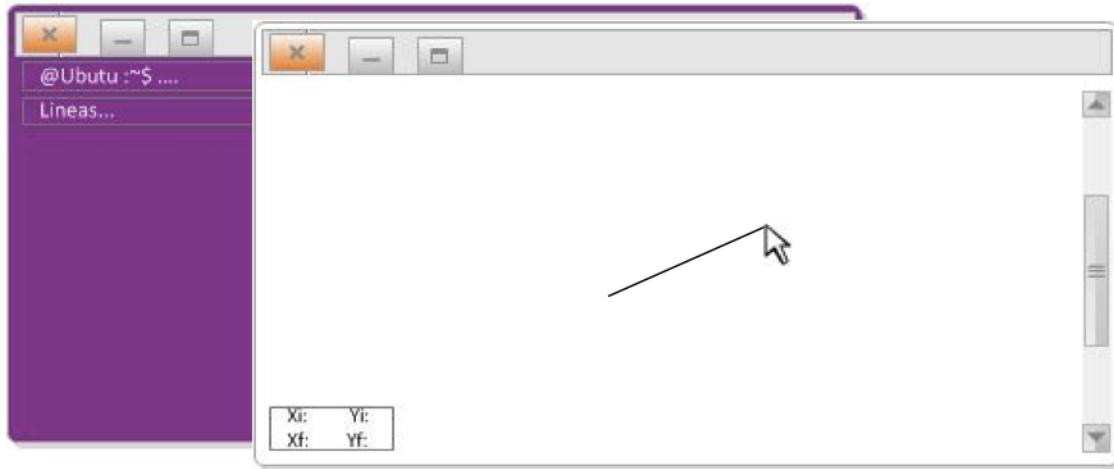


Figura 2.3: Módulo 1 - Líneas en Diferentes Direcciones.

R4: El módulo 1 del sistema permite dibujar circunferencias de distintos radios.

Objetivo: Dibujar una superficie plana limitada por una circunferencia.

Descripción: Es el lugar geométrico de los puntos de un plano que equidistan de otro punto fijo y coplanario llamado centro en una cantidad constante llamada radio, el usuario decidirá el tamaño de la circunferencia, como se muestra en la Figura 2.4.

Datos de entrada: posición inicial y final.

Datos de salida: Circunferencia realizada.

Pre-condiciones: Posición de partida.

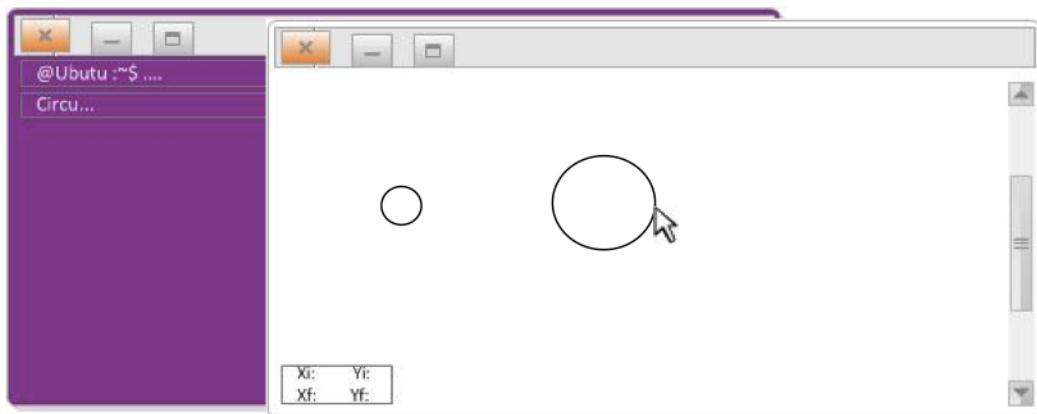


Figura 2.4: Módulo 1 - Circunferencias de Distintos Radios.

R5: El módulo 1 del sistema permite dibujar una elipse de distintas dimensiones.

Objetivo: El módulo del sistema permite dibujar una superficie curva plana simétrica a 2 ejes.

Descripción: El módulo del sistema permite el dibujo de una región curva del plano simétrica a 2 ejes (elipse), la cual el usuario establecerá la dimensión con el movimiento de su dedo, como se muestra en la Figura 2.5.

Datos de entrada: posición inicial y final.

Datos de salida: elipse realizada.

Pre-condiciones: Posición de partida.

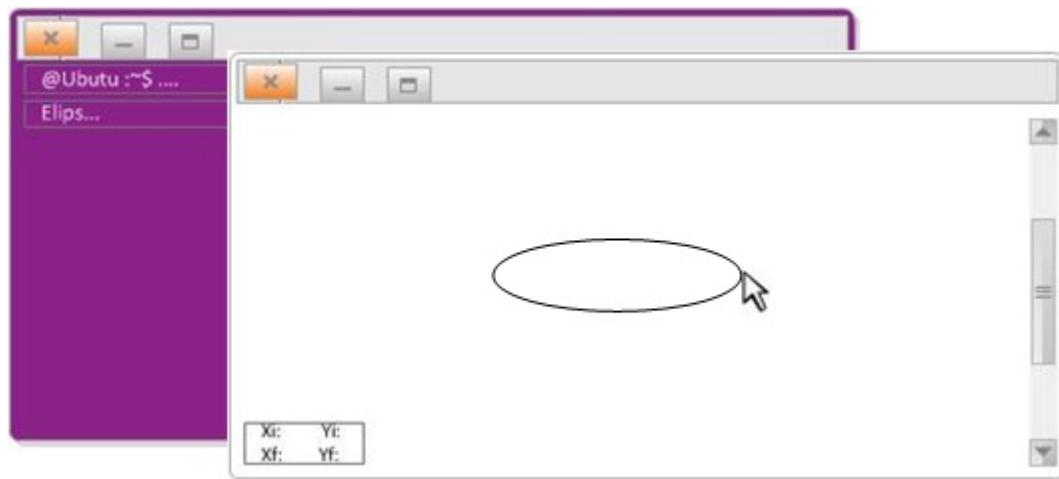


Figura 2.5: Módulo 1 - Elipse de distinta dimensión.

R6: El módulo 1 del sistema permite dibujar polígonos de diferentes tamaños.

Objetivo: el módulo del sistema permite dibujar una figura plana compuesta por una secuencia de segmentos rectos.

Descripción: El módulo del sistema permite al usuario dibujar una figura plana compuesta por lados (segmentos), clasificándolos por su número de lados, la cual puede ser de distinto tamaño según el usuario, como se muestra en la Figura 2.6.

Datos de entrada: posición inicial y final.

Datos de salida: figura plana realizada (polígono).

Pre-condiciones: Posición de partida.

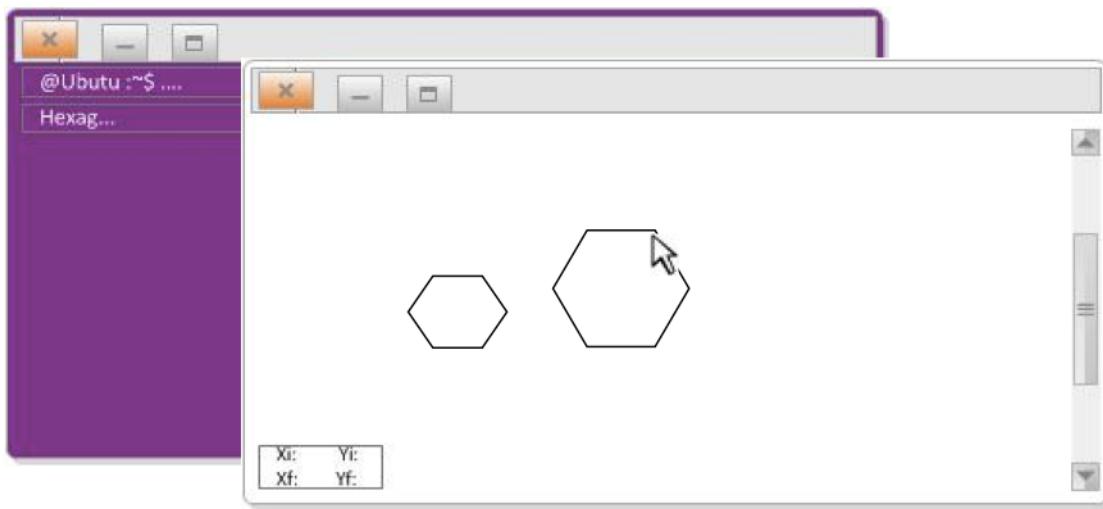


Figura 2.6: Módulo 1 - Polígonos de Diferentes Tamaños.

R7: El módulo 1 del sistema permite dibujar un polígono de diferentes números de lados (3 a 6 lados) dependiendo de la *tag* utilizada.

Objetivo: El módulo permite dibujar una figura plana compuesta por una secuencia de segmentos rectos.

Descripción: El módulo del sistema permite trazar el dibujo de una figura plana compuesta por lados (segmentos), clasificándolos por su número de lados (de 3 a 6), el cual se seleccionará dando el tipo de polígono para hacer la figura deseada, como se muestra en la Figura 2.7.

Datos de entrada: posición inicial y final.

Datos de salida: figura plana realizada (polígono).

Pre-condiciones: Posición de partida.

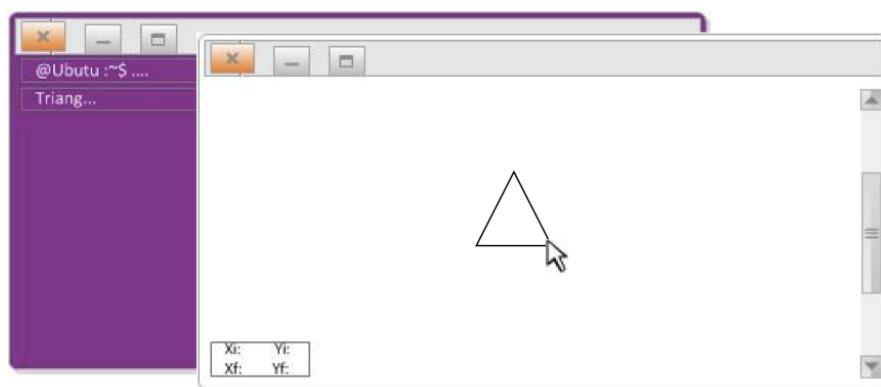


Figura 2.7: Módulo 1 - Selección del tipo de Polígono.

R8: El módulo 2 del sistema permite la integración con *KinectTM*.

Objetivo: Poder integrar el *KinectTM* con ayuda del *Driver OpenNi* a la computadora y lo reconozca.

Descripción: La computadora reconocerá la identificación del *KinectTM* con el uso del *Driver (OpenNi)*, para capturar, utilizar y procesar la información recibida del sensor. Figura 2.8.

Datos de entrada: Información recibida del sensor.

Datos de salida: Reconocimiento del *KinectTM*.

Pre-condiciones: Instalación del *Driver (OpenNi)*.



Figura 2.8: Módulo 2 - Integración con Kinect.

R9: El módulo 2 del sistema permite la detección del dedo índice.

Objetivo: Poder detectar la imagen que capturará *KinectTM*.

Descripción: Se detectará la imagen, siendo más específico en este Módulo será el dedo índice que capturará el *KinectTM* para poderlo procesar, como se muestra en la Figura 2.9.

Datos de entrada: Información recibida del sensor de *KinectTM*.

Datos de salida: Reconocimiento del dedo índice.

Pre-condiciones: Instalación del *API*, *KinectTM* conectado.



Figura 2.9: Módulo 2 - Detección del Dedo.

R10: El módulo 2 del sistema permite reconocer el desplazamiento del dedo índice.

Objetivo: Poder detectar el desplazamiento con el *KinectTM*.

Descripción: Detectar el desplazamiento que hará el dedo índice mediante el *KinectTM* la imagen, siendo más específico en este Módulo será el dedo índice que capturará el *KinectTM* para poder ser procesado por la PC, como se muestra en la Figura 2.10.

Datos de entrada: Posición inicial del dedo.

Datos de salida: Posición final del dedo.

Pre-condiciones: Instalación del *API*, *KinectTM* conectado.

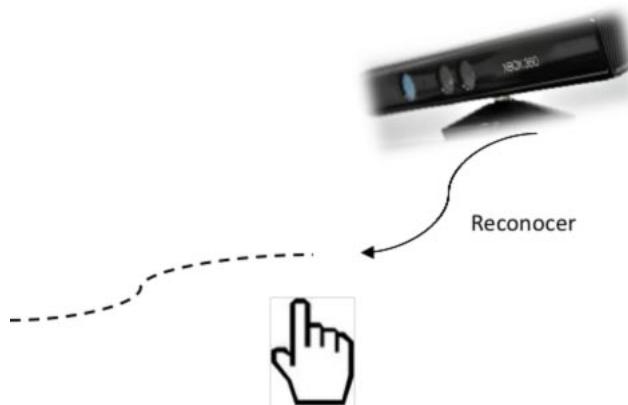


Figura 2.10: Módulo 2 - Reconocer Desplazamiento del dedo índice.

R11: El módulo 2 del sistema permite dibujar a mano alzada con el dedo índice.

Objetivo: Poder plasmar el trazo realizado con el dedo.

Descripción: Poder ver reflejado la acción del dedo, con la realización del trazo a mano alzada, como se muestra en la Figura 2.11.

Datos de entrada: posición inicial.

Datos de salida: Trazo realizado.

Pre-condiciones: Instalación del *API, KinectTM* conectado.



Figura 2.11: Módulo 2 - Dibujar a Mano Alzada con el Dedo Índice.

R12: El módulo 2 del sistema permite proyectar en el editor básico de dibujo la acción realizada por el dedo índice.

Objetivo: Poder plasmar el trazo realizado con el dedo en el editor de dibujo.

Descripción: Poder ver reflejado la acción del dedo, con la realización del trazo a mano alzada en el editor de dibujo básico con la colaboración del *KinectTM*, como se muestra en la Figura 2.12.

Datos de entrada: posición inicial.

Datos de salida: Trazo realizado.

Pre-condiciones: Instalación del *API, KinectTM* conectado.

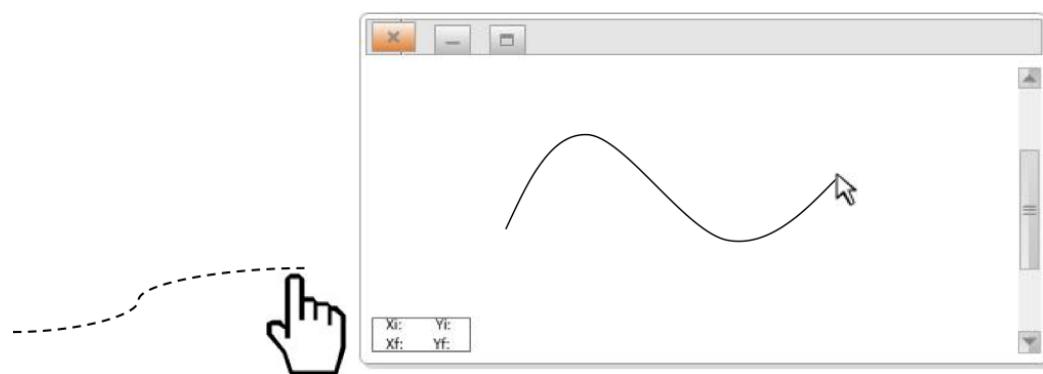


Figura 2.12: Módulo 2 - Proyección en el editor básico de dibujo.

R13: El módulo 3 del sistema permite trabajar conjuntamente proyector y *Kinect*TM.

Objetivo: Poder trabajar simultáneamente el proyector y el *Kinect*TM.

Descripción: Tener trabajando colectivamente los dos dispositivos, como se muestra en la Figura 2.13, evitando interferencias (ruido) entre ambos.

Datos de entrada: Información recibida del sensor del *Kinect*TM.

Datos de salida: Reconocimiento de los dispositivos.

Pre-condiciones: Instalación del *API*, dispositivos conectados.



Figura 2.13: Módulo 3 - Trabajo Colectivo de los Dispositivos.

R14: El módulo 3 del sistema permite proyectar sobre el área de trabajo.

Objetivo: Poder plasmar la imagen con el proyector sobre el área de trabajo.

Descripción: Con el proyector se podrá mostrar la imagen en el área de trabajo que se indicará ajustando el dispositivo en una cierta posición, como se muestra en la Figura 2.14.

Datos de entrada: ninguno.

Datos de salida: Imagen proyectada.

Pre-condiciones: Instalación del *API*, *Kinect*TM y proyector conectado.

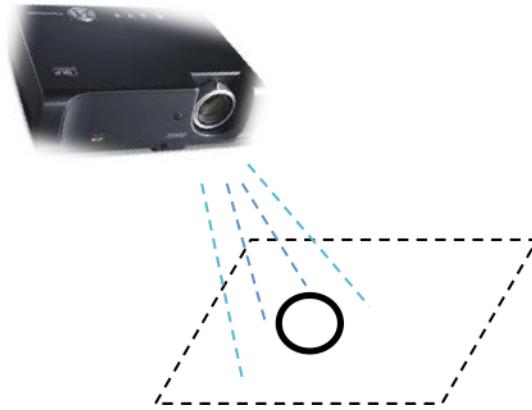


Figura 2.14: Módulo 3 - Proyección sobre el área de trabajo.

R15: El módulo 3 del sistema permite trabajar sin interferencia de la sombra que produzca la mano.

Objetivo: Poder trabajar con la mano, a pesar de la sombra que genere no afectará el producto deseado.

Descripción: Trabajar con la mano, a pesar de la sombra que genere, no afectará el producto deseado (figura o acción) proyectado sobre el área de trabajo, como se muestra en la Figura 2.15.

Datos de entrada: Procesamiento de imagen.

Datos de salida: Imagen proyectada.

Pre-condiciones: Instalación del *API*, *KinectTM* y proyector conectado.

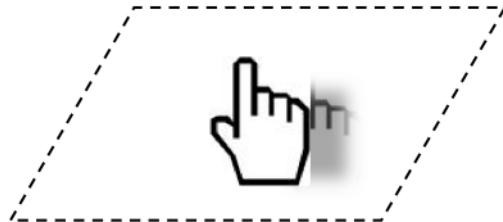


Figura 2.15: Módulo 3 - Trabajar a pesar de la sombra.

R16: El módulo 4 del sistema permite reconocer la herramienta (*tag*).

Objetivo: Que el *KinectTM* reconozca la herramienta (*tag*).

Descripción: Poder hacer que se reconozca la herramienta mediante el *KinectTM*, dicha herramienta la llamamos *tag*, la cual es una imagen binaria, como se muestra en la Figura 2.16.

Datos de entrada: imagen binaria.

Datos de salida: herramienta reconocida.

Pre-condiciones: Instalación del *API*, *Kinect*TMconectado, imagen binaria.



Figura 2.16: Módulo 4 - Reconocimiento de la herramienta (Tag).

Nota: la imagen binaria (tag) utilizada es solo un ejemplo

R17: El módulo 4 del sistema permite dibujar con la herramienta (tag) de la figura a usar.

Objetivo: Se utiliza la herramienta (*tag*) para la acción que tiene prevista, la cual es dibujar.

Descripción: Poder dibujar alguna de las figuras ya mencionadas, simplemente con el hecho de poner la tag, y que los usuarios (máximo 2 usuarios) desplacen el dedo para crear la figura, reconociendo que debe dibujar la figura seleccionada por la tag que le corresponde a la figura, como se muestra en la Figura 2.17.

Datos de entrada: imagen binaria.

Datos de salida: herramienta reconocida.

Pre-condiciones: Instalación del *API*, *Kinect* conectado, imagen binaria.

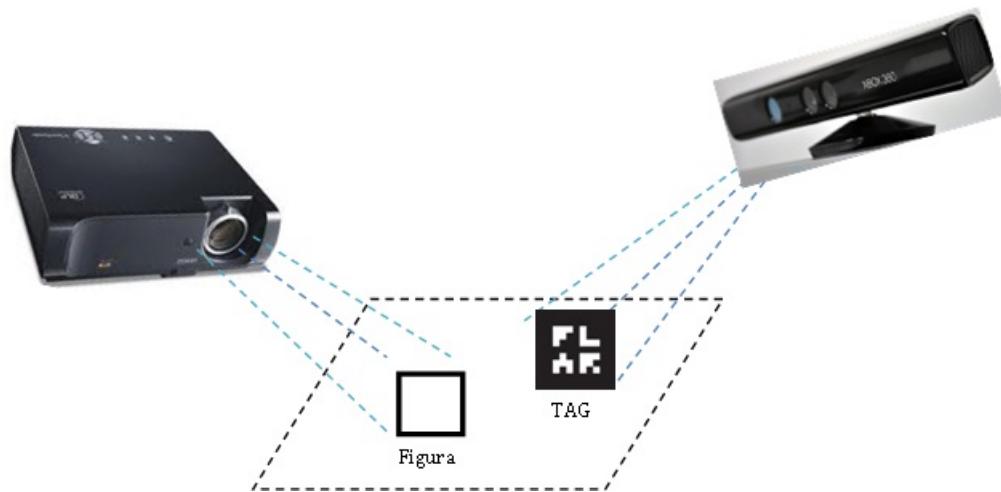


Figura 2.17: Módulo 4 - Dibujar utilizando la Herramienta (tag).

R18: El módulo 4 del sistema permite realizar la selección de escalar una figura.

Objetivo: Se utiliza la herramienta (*tag*) para la acción que tiene prevista, la cual es cambiar de tamaño la figura.

Descripción: Poder realizar cambios a alguna de las figuras ya mencionadas, simplemente con el hecho de poner la *tag* y que reconozca que debe cambiar de tamaño la figura seleccionada, girando la *tag* que le corresponde a la acción de escalar, como se muestra la Figura 2.18.

Datos de entrada: imagen binaria.

Datos de salida: herramienta reconocida.

Pre-condiciones: Instalación del *API*, *Kinect*TMconectado, imagen binaria.



Figura 2.18: Módulo 4 - Escalar Figura.

R19: El módulo 4 del sistema permite realizar la selección de mover la figura.

Objetivo: Se utiliza la herramienta (*tag*) para la acción que tiene prevista, la cual es mover la figura.

Descripción: Poder realizar cambios a alguna de las figuras ya mencionadas, simplemente con el hecho de poner la *tag* y que reconozca, que debe mover de posición la figura seleccionada por la *tag* que le corresponde a la acción de mover, como se muestra en la Figura 2.19.

Datos de entrada: imagen binaria.

Datos de salida: herramienta reconocida.

Pre-condiciones: Instalación del *API*, *Kinect*TMconectado, imagen binaria.

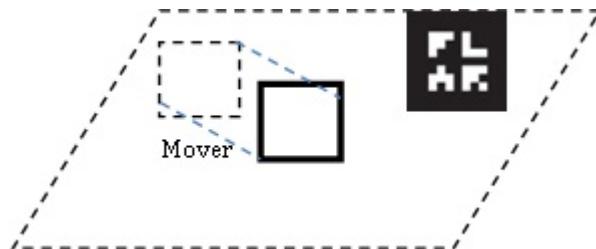


Figura 2.19: Módulo 4 - Mover Figura.

R20: El módulo 4 del sistema permite realizar la selección de cambiar el color de la figura.

Objetivo: Se utiliza la herramienta (*tag*) para la acción que tiene prevista, la cual es cambiar de color la figura.

Descripción: Poder realizar cambios a alguna de las figuras ya mencionadas, simplemente con el hecho de poner la tag y que reconozca que debe cambiar de color la figura seleccionada, girando la *tag* que le corresponde a la acción de cambiar color figura 2.20.

Datos de entrada: imagen binaria.

Datos de salida: herramienta reconocida.

Pre-condiciones: Instalación del *API*, *Kinect*TMconectado, imagen binaria.

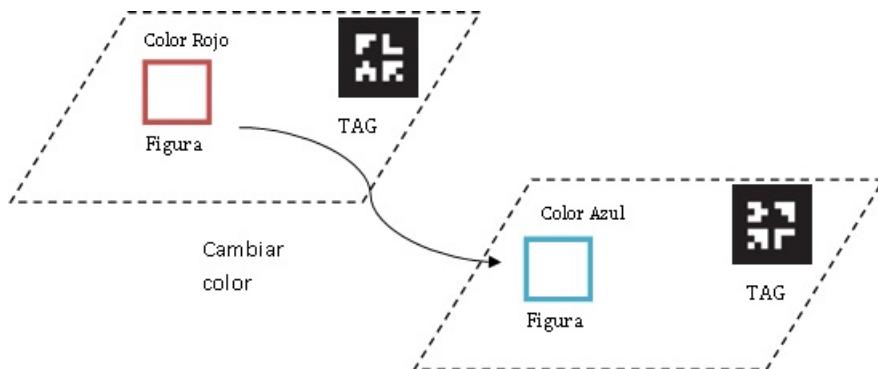


Figura 2.20: Módulo 4 - Cambiar de color.

Capítulo 3

Diseño

3.1. Arquitectura del Sistema

A continuación se muestra la estructura o esqueleto del sistema, para ello se utilizo la abstracción de capas [18],[19] el cual cuenta con tres capas, una capa superior, una intermedia y una inferior, las cuales se describen enseguida(Figura 3.1):

- La capa superior: Representa el *software* que implementa el sistema, con el que interactúa el usuario.
- La capa intermedia: Representa la *API* (*OpenCV*) y el *driver* (*OpenNI*), para la comunicación entre el hardware y el sistema.
- La capa inferior: Muestra los dispositivos (*hardware*), que se encargan de la captura y proyección de la imagen.

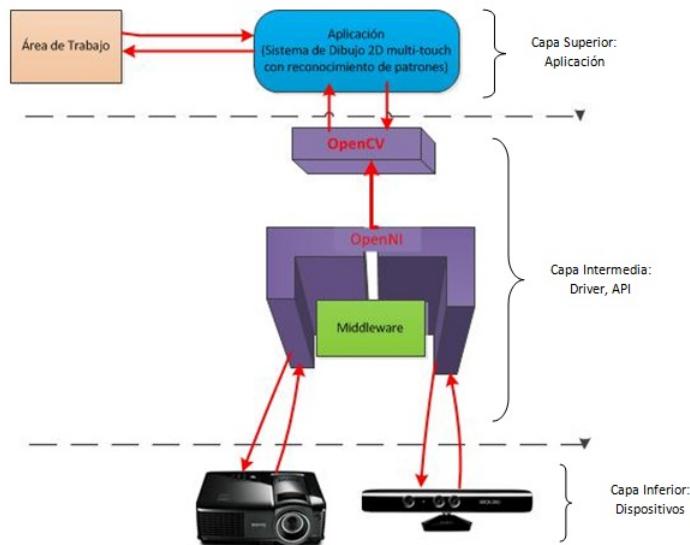


Figura 3.1: Arquitectura - abstracción de capas.

3.2. Diagramas Generales del Sistema

En esta sección se mostrarán los diagramas generales, que se llevaron acabo para la realización del sistema, los cuales son: de casos de uso(Figura 3.2), de clases(Figura 3.3), de secuencia(Figura 3.4) y de estados(Figura 3.5).

3.2.1. Diagrama General de Casos de Uso.

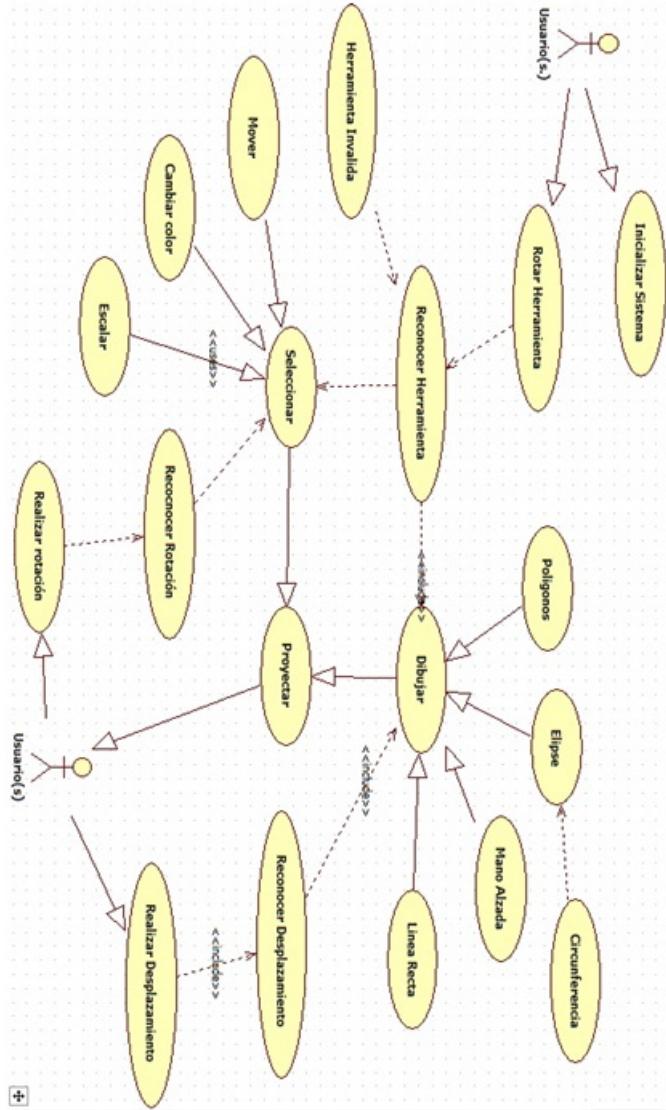


Figura 3.2: Diagrama General de Casos de Uso.

Especificación de casos de uso

En este aparto se describen los casos de uso del sistema, donde se explica cada caso de uso correspondiente a su Cuadro, en el cual se tiene su descripción, el actor, las condiciones con las que se debe contar, su flujo, así como anotaciones y/o excepciones.

Abreviaturas:

E.C.U.: Especificación de Caso de Uso

TT-2011-B007: Trabajo Terminal con No. de Registro 2011-B007

Caso de Uso: Inicializar Sistema

Especificación del Caso de Uso:	Inicializar Sistema
ID:	E.C.U-1
Nombre:	Inicializar el sistema
Descripción:	El usuario inicializara la aplicación del sistema
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivos conectados (kinect, proyector), Driver y API instalados
Pos Condición:	Sistema inicializado con éxito
Flujo Normal de eventos:	Iniciar: 1. El actor iniciara el sistema de dibujo en 2D multi-touch con reconocimiento de patrones 2. Una vez iniciado, el usuario podrá interactuar con el sistema
Flujos Alternos:	Sistema ya iniciado: 1. Si ya se inicio, se procederá con el paso 2
Excepciones:	Ninguna
Anotaciones:	Ninguna

Cuadro 3.1: Inicializar Sistema.

Caso de Uso: Rotar Herramienta

Especificación del Caso de Uso:	Rotar Herramienta
ID:	E.C.U-2
Nombre:	Rotar Herramienta
Descripción:	El usuario rotará la herramienta llamada TAG, en el área de trabajo
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivo kinect conectado
Pos Condición:	La herramienta estará situada en el área de trabajo
Flujo Normal de eventos:	Colocación de la Herramienta: 1. El actor seleccionara la herramienta que desea utilizar 2. Se rotará la herramienta en el área de trabajo
Flujos Alternos:	Posición de Herramienta: 1. Después de rotar la herramienta se acomodara en una posición para hacer algo distinto en el editor, si se trata de una herramienta para hacer cambios
Excepciones:	Herramienta mal posicionada: 1. Si la herramienta no está del lado indicado ó de la posición correcta, no se podrá trabajar con ella
Anotaciones:	Solo se puede utilizar una herramienta en el área a la vez

Cuadro 3.2: Rotar Herramienta.

Caso de Uso: Reconocer Herramienta

Especificación del Caso de Uso:	Reconocer Herramienta
ID:	E.C.U-3
Nombre:	Reconocer Herramienta
Descripción:	Cuando ya se encuentre colocada la tag , kinect tomará video del escenario el cual será procesado por la PC y se detecta la tag que este ubicada en el área de trabajo
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivo kinect conectado
Pos Condición:	Quedará habilitada la acción correspondiente a la tag.
Flujo Normal de eventos:	<p>Detección de imagen:</p> <ol style="list-style-type: none"> 1. Una vez colocada la herramienta en el área, el kinect captura video del escenario para procesarla 2. Una vez que se procese, se podrá hacer la acción que tiene prevista la herramienta 3. El actor podrá seleccionar ó dibujar en el área de trabajo
Flujos Alternos:	<p>Posición de Herramienta:</p> <ol style="list-style-type: none"> 1. Si se trata de una herramienta para hacerle cambios a lo dibujado, entonces se moverá de posición la herramienta(giro), para tener dichos cambios, esto solo aplica en escalar y color
Excepciones:	<p>Imagen incorrecta:</p> <ol style="list-style-type: none"> 1. Si la tag no corresponde con las indicadas en el sistema ,no se podrá reconocer
Anotaciones:	Solo se puede utilizar una herramienta en el área a la vez

Cuadro 3.3: Reconocer Herramienta.

Caso de Uso: Seleccionar

Especificación del Caso de Uso:	Seleccionar
ID:	E.C.U-4
Nombre:	Seleccionar
Descripción:	Selección del dibujo para poder realizar algún cambio sobre éste.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivo kinect conectado, dibujo previo
Pos Condición:	Se selecciono con éxito el dibujo.
Flujo Normal de eventos:	<p>Selección:</p> <ol style="list-style-type: none"> 1. Ya que se tenga el dibujo, el actor, seleccionará la imagen mediante el uso de su dedo 2. Para por ultimo poder hacer algún cambio en la imagen
Flujos Alternos:	<p>Dibujo seleccionado:</p> <ol style="list-style-type: none"> 1. Una vez que se está en el paso 1 del flujo normal se puede cambiar el tipo de acción para modificar el dibujo
Excepciones:	<p>Selección de dibujo:</p> <ol style="list-style-type: none"> 1. Trazos a mano alzada no podrán ser seleccionados.
Anotaciones:	Solo se puede utilizar una herramienta en el área a la vez, el usuario solo puede utilizar un dedo de la mano.

Cuadro 3.4: Seleccionar.

Caso de Uso: Herramienta Invalida

Especificación del Caso de Uso:	Herramienta Invalida
ID:	E.C.U-5
Nombre:	Herramienta Invalida
Descripción:	Si se llega a poner otro objeto que no se tenga considerado como herramienta del sistema, entonces el sistema no reconocerá dicha herramienta.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivo kinect conectado
Pos Condición:	El objeto es una herramienta invalida para el sistema.
Flujo Normal de eventos:	Herramienta Errónea: 1. Si el actor llegará a poner otro objeto que no se considero en el sistema, entonces no se podrá reconocer el objeto
Flujos Alternos:	Verificar Herramienta: 1. Si pasara el paso 1 del evento, entonces se verificaría si la herramienta es la correcta o está colocada correctamente.
Excepciones:	Ninguna
Anotaciones:	Solo se puede utilizar una herramienta en el área a la vez.

Cuadro 3.5: Herramienta Invalida.

Caso de Uso: Mover

Especificación del Caso de Uso:	Mover
ID:	E.C.U-6
Nombre:	Mover
Descripción:	Esta herramienta nos permite mover de posición alguna figura, para ello debe estar seleccionada.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivo kinect conectado
Pos Condición:	Reconocimiento de la herramienta, poder mover con éxito el dibujo.
Flujo Normal de eventos:	Mover: 1. El actor seleccionará el dibujo que desea mover 2. El actor procederá a mover el dibujo seleccionado sobre el área.
Flujos Alternos:	Ninguno.
Excepciones:	Selección de dibujo: 1. Trazos a mano alzada no podrán ser seleccionados.
Anotaciones:	Solo se puede utilizar una herramienta en el área a la vez , el usuario solo puede utilizar un dedo de la mano.

Cuadro 3.6: Mover.

Caso de Uso: Cambiar Color

Especificación del Caso de Uso:	Cambiar Color
ID:	E.C.U-7
Nombre:	Cambiar Color
Descripción:	Esta herramienta nos permite cambiar de color alguna figura seleccionada.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivo kinect conectado
Pos Condición:	Reconocimiento de la herramienta, poder cambiar de color con éxito el dibujo.
Flujo Normal de eventos:	<p>Color:</p> <ol style="list-style-type: none"> 1. El actor moverá las herramientas que se tiene para la gama de colores(64 colores), una por R, otra por G y una de B, para dar dichas combinaciones 2. Se rotará la herramienta hasta conseguir el color deseado en el dibujo. 3. El actor dibujara su trazo con el color escogido
Flujos Alternos:	Ninguno.
Excepciones:	<p>Selección de dibujo:</p> <ol style="list-style-type: none"> 1. Trazos a mano alzada no podrán ser seleccionados.
Anotaciones:	Solo se puede utilizar una herramienta en el área a la vez , el usuario solo puede utilizar un dedo de la mano.

Cuadro 3.7: Cambiar Color.

Caso de Uso: Escalar

Especificación del Caso de Uso:	Escalar
ID:	E.C.U-8
Nombre:	Escalar
Descripción:	Esta herramienta nos permite cambiar de tamaño alguna figura seleccionada.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivo kinect conectado
Pos Condición:	Reconocimiento de la herramienta, poder cambiar de tamaño con éxito el dibujo.
Flujo Normal de eventos:	Tamaño: 1. Se colocará la herramienta de Escalar en el área de trabajo 2. El actor seleccionará el dibujo que desea cambiar de tamaño. 3. Se rotara la herramienta hasta conseguir el tamaño deseado en el dibujo
Flujos Alternos:	Ninguno.
Excepciones:	Selección de dibujo: 1. Trazos a mano alzada no podrán ser seleccionados.
Anotaciones:	Solo se puede utilizar una herramienta en el área a la vez , el usuario solo puede utilizar un dedo de la mano.

Cuadro 3.8: Escalar.

Caso de Uso: Reconocer Rotación

Especificación del Caso de Uso:	Reconocer Rotación
ID:	E.C.U-9
Nombre:	Reconocer Rotación
Descripción:	El dispositivo kinect transmitirá la información de la escena de la rotación de la tag a la PC, al cambiar de posición (girar) se procesa para aplicar cierto efecto.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivo kinect conectado
Pos Condición:	Reconocimiento de la acción de la herramienta.
Flujo Normal de eventos:	<p>Reconocimiento de rotación:</p> <ol style="list-style-type: none"> 1. Una vez colocada la herramienta en el área, el kinect captura el escenario para procesarlo. 2. El actor rotara la imagen para obtener la acción designada.
Flujos Alternos:	<p>Rotación:</p> <ol style="list-style-type: none"> 1. Si ya se tiene el paso 1 del evento, entonces se procede a rotar la herramienta las veces que permite el sistema, para obtener la acción designada
Excepciones:	Ninguna.
Anotaciones:	Solo se puede utilizar una herramienta en el área a la vez , el usuario solo puede utilizar un dedo de la mano.

Cuadro 3.9: Reconocer Rotación.

Caso de Uso: Realizar Rotación

Especificación del Caso de Uso:	Realizar Rotación
ID:	E.C.U-10
Nombre:	Realizar Rotación
Descripción:	El dispositivo kinect capturará el escenario y se identificará la rotación de la tag, la cual al cambiar de posición, podremos visualizar la acción designada.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivo kinect conectado
Pos Condición:	Reconocimiento de cambio de la herramienta.
Flujo Normal de eventos:	Reconocimiento de rotación: 1. Giro de la tag por parte del usuario. 2. Se aplica el cambio al dibujo de acuerdo a la acción designada para la tag.
Flujos Alternos:	Rotación: 1. Giro incompleto de la tag.
Excepciones:	Selección de dibujo: 1. Trazos a mano alzada no podrán ser seleccionados. 2. La herramienta tiene un límite de giros.
Anotaciones:	Solo se puede utilizar una herramienta en el área a la vez , el usuario solo puede utilizar un dedo de la mano.

Cuadro 3.10: Realizar Rotación.

Caso de Uso: Proyectar

Especificación del Caso de Uso:	Proyectar
ID:	E.C.U-11
Nombre:	Proyectar
Descripción:	El proyector nos dará la visualización del área de trabajo
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivos: kinect y proyector conectados
Pos Condición:	Proyección del área de trabajo.
Flujo Normal de eventos:	Proyección: 1. Una vez iniciado el sistema se proyectará el área de trabajo. 2. El usuario puede empezar a dibujar o colocar herramientas para dibujar.
Flujos Alternos:	Ninguno.
Excepciones:	Ninguno.
Anotaciones:	Al inicio del sistema solo se visualizara el área de trabajo.

Cuadro 3.11: Proyectar.

Caso de Uso: Reconocer Desplazamiento

Especificación del Caso de Uso:	Reconocer Desplazamiento
ID:	E.C.U-12
Nombre:	Reconocer Desplazamiento
Descripción:	El dispositivo kinect capturará el desplazamiento del dedo para que lo procese la PC.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivos: kinect conectado
Pos Condición:	Reconocimiento del dedo con éxito.
Flujo Normal de eventos:	Proyección: 1. El actor realizará el desplazamiento de su dedo. 2. El kinect capturará la escena y la envía a la PC para procesar el movimiento que se realice.
Flujos Alternos:	Ninguno.
Excepciones:	Ninguno.
Anotaciones:	Los usuarios solo pueden usar un solo dedo índice por usuario.

Cuadro 3.12: Reconocer Desplazamiento.

Caso de Uso: Realizar Desplazamiento

Especificación del Caso de Uso:	Realizar Desplazamiento
ID:	E.C.U-13
Nombre:	Realizar Desplazamiento
Descripción:	El actor hará el desplazamiento del dedo hasta la posición que se desea.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Ninguna.
Pos Condición:	Reconocimiento del dedo con éxito.
Flujo Normal de eventos:	<p>Desplazamiento:</p> <ol style="list-style-type: none"> 1. El actor desplazará su dedo por el área de trabajo de un punto a otro. 2. El kinect captura y la PC procesara el movimiento que realice el dedo.
Flujos Alternos:	Ninguno.
Excepciones:	Ninguno.
Anotaciones:	Los usuarios solo pueden usar un solo dedo índice por usuario.

Cuadro 3.13: Realizar Desplazamiento.

Caso de Uso: Dibujar

Especificación del Caso de Uso:	Dibujar
ID:	E.C.U-14
Nombre:	Dibujar
Descripción:	El actor podrá realizar trazos dentro del área de trabajo.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivos: kinect y proyector, tags en caso de requerirlo..
Pos Condición:	Visualización de trazo realizado.
Flujo Normal de eventos:	Dibujar con herramienta: 1. El actor pondrá la herramienta de la figura deseada. 2. Desplazara su dedo índice para formar la figura.
Flujos Alternos:	Dibujar sin herramienta: 1. Si el actor quiere dibujar un trazo a mano alzada simplemente realizara el trazo en el área de trabajo.
Excepciones:	Ninguno.
Anotaciones:	Los usuarios solo pueden usar un solo dedo índice por usuario.

Cuadro 3.14: Dibujar.

Caso de Uso: Mano Alzada

Especificación del Caso de Uso:	Mano Alzada
ID:	E.C.U-15
Nombre:	Mano Alzada
Descripción:	El actor podrá realizar ciertos trazos a mano alzada dentro del área de trabajo.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivos: kinect y proyector , Posición inicial.
Pos Condición:	Trazo realizado.
Flujo Normal de eventos:	Dibujar a mano alzada: 1. El actor usara su dedo con un movimiento libre de un punto inicial a un punto final, dentro del área de trabajo. 2. El kinect captura la escena y se reflejará el trazo realizado.
Flujos Alternos:	Ninguno.
Excepciones:	Velocidad de desplazamiento: 1. Si la velocidad de desplazamiento varía de la velocidad de procesamiento, el trazo realizado podría no ser igual al movimiento del dedo
Anotaciones:	Los usuarios solo pueden usar un solo dedo índice por usuario.

Cuadro 3.15: Mano Alzada.

Caso de Uso: Línea Recta

Especificación del Caso de Uso:	Línea Recta
ID:	E.C.U-16
Nombre:	Línea Recta
Descripción:	El actor podrá realizar un trazo recto a partir de dos puntos.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivos: kinect y proyector , Posición inicial.
Pos Condición:	Trazo realizado.
Flujo Normal de eventos:	Dibujar a mano alzada: 1. El actor usará la herramienta para líneas. 2. El actor usara su dedo con un movimiento libre dentro del área de trabajo. 3. El kinect capturará la escena y se reflejará el trazo realizado en el área
Flujos Alternos:	Interacción de los actores: 1. El desplazamiento que realiza el actor ,puede variar con lo que se dibuja
Excepciones:	Segmentación de Pixel: 1. Se puede apreciar una pequeña deformación en la línea, al no poder segmentar un pixel.
Anotaciones:	Los usuarios solo pueden usar un solo dedo índice por usuario.

Cuadro 3.16: Línea Recta.

Caso de Uso: Elipse

Especificación del Caso de Uso:	Elipse
ID:	E.C.U-17
Nombre:	Elipse
Descripción:	El actor podrá dibujar una curva plana y cerrada, simétrica respecto a dos ejes perpendiculares entre sí.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivos: kinect y proyector , Posición inicial.
Pos Condición:	Elipse realizada.
Flujo Normal de eventos:	Dibujar elipse: 1. El actor utilizará la herramienta para el elipse. 2. El actor usara su dedo con un movimiento libre dentro del área de trabajo. 3. El kinect lo capturará y se reflejara el trazo realizado en el área
Flujos Alternos:	Interacción de los actores: 1. Cuando se encuentre el paso 1, los actores podrán realizar su trazo respectivamente dentro del área de trabajo
Excepciones:	Distancia de puntos de la figura: 1. 1. Se puede apreciar una deformación mayor cuando la distancia entre los puntos sea menor, al ser pixeles de forma cuadrada.
Anotaciones:	Los usuarios solo pueden usar un solo dedo índice por usuario.

Cuadro 3.17: Elipse.

Caso de Uso: Circunferencia

Especificación del Caso de Uso:	Circunferencia
ID:	E.C.U-18
Nombre:	Circunferencia
Descripción:	El actor podrá dibujar una superficie plana limitada por una circunferencia.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivos: kinect y proyector , Posición inicial.
Pos Condición:	Circunferencia realizada.
Flujo Normal de eventos:	Dibujar circunferencia: 1. El actor utilizará la herramienta para la circunferencia. 2. El actor usara su dedo con un movimiento libre dentro del área de trabajo. 3. kinect lo capturará y se reflejara el trazo realizado en el área
Flujos Alternos:	Interacción de los actores: 1. Cuando se encuentre el paso 1, los actores podrán realizar su trazo respectivamente dentro del área de trabajo
Excepciones:	Distancia de puntos de la figura: 1. Se puede apreciar una deformación mayor cuando la distancia entre los puntos sea menor, al ser pixeles de forma cuadrada.
Anotaciones:	Los usuarios solo pueden usar un solo dedo índice por usuario.

Cuadro 3.18: Circunferencia.

Caso de Uso: Polígono

Especificación del Caso de Uso:	Polígono
ID:	E.C.U-18
Nombre:	Polígono
Descripción:	El actor podrá dibujar una figura plana compuesta por una secuencia de segmentos rectos , que se puede elegir entre 3 a 6 lados.
Autor:	TT-2011-B007
Actores:	Usuarios (Máximo dos usuarios)
Precondición:	Dispositivos: kinect y proyector , Posición inicial.
Pos Condición:	Polígono realizado.
Flujo Normal de eventos:	<p>Dibujar polígono:</p> <ol style="list-style-type: none"> 1. El actor utilizará la herramienta del polígono deseado. 2. El actor usara su dedo con un movimiento libre dentro del área de trabajo. 3. El kinect lo capturará y se reflejara el trazo de la circunferencia.
Flujos Alternos:	<p>Interacción de los actores:</p> <ol style="list-style-type: none"> 1. Cuando se encuentre el paso 1, los actores podrán realizar su trazo respectivamente dentro del área de trabajo
Excepciones:	<p>Dimensión de la figura:</p> <ol style="list-style-type: none"> 1. Se puede apreciar una deformación, cuando se le da una dimensión, al no poder segmentar un pixel. 2. Solo se puede hacer polígonos de 3 a 6 lados.
Anotaciones:	Los usuarios solo pueden usar un solo dedo índice por usuario.

Cuadro 3.19: Polígono.

3.2.2. Diagrama General de Clases.

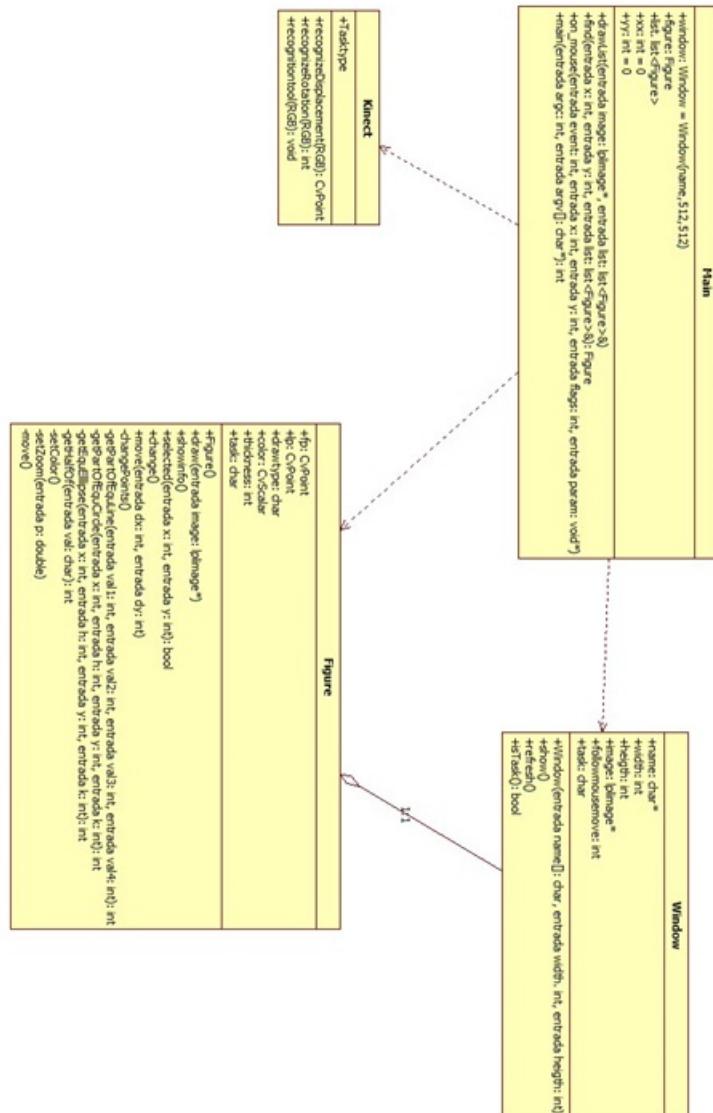


Figura 3.3: Diagrama General de Clases.

3.2.3. Diagrama General de Secuencia.

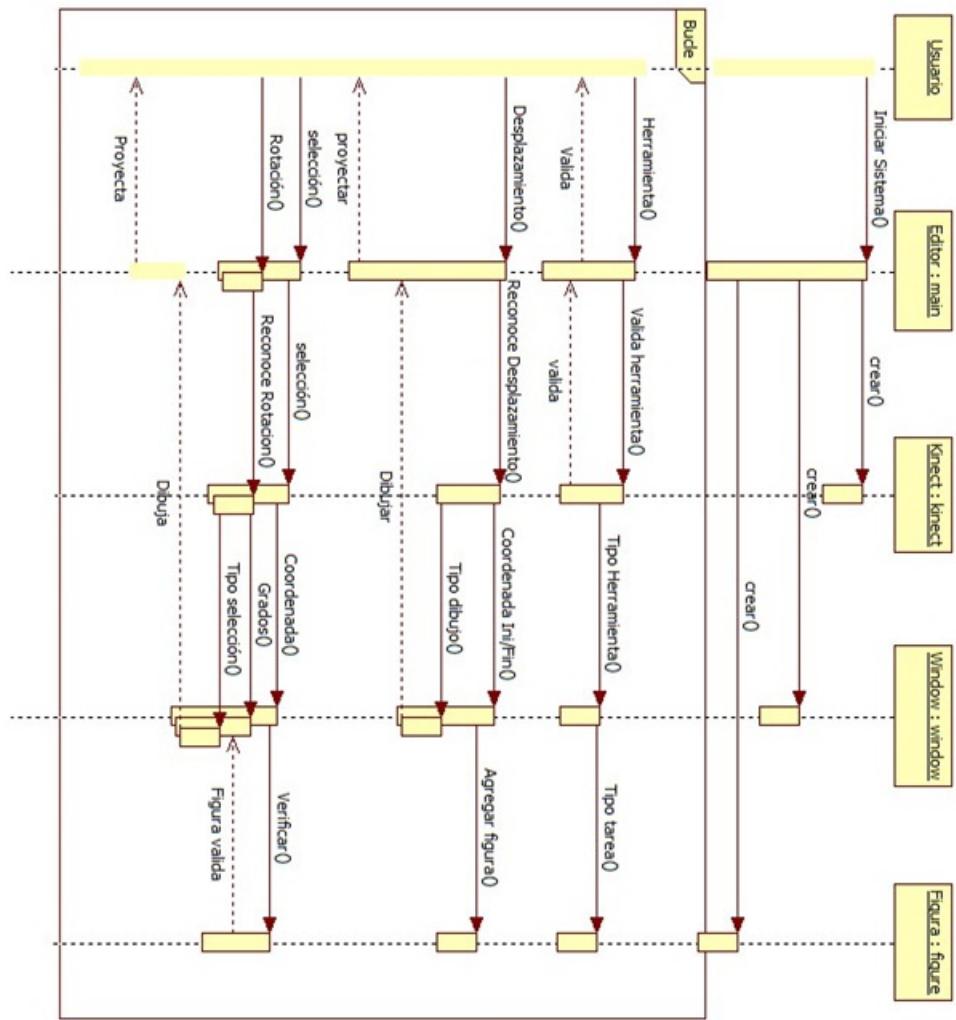


Figura 3.4: Diagrama General de Secuencia.

3.2.4. Diagrama General de Estados.

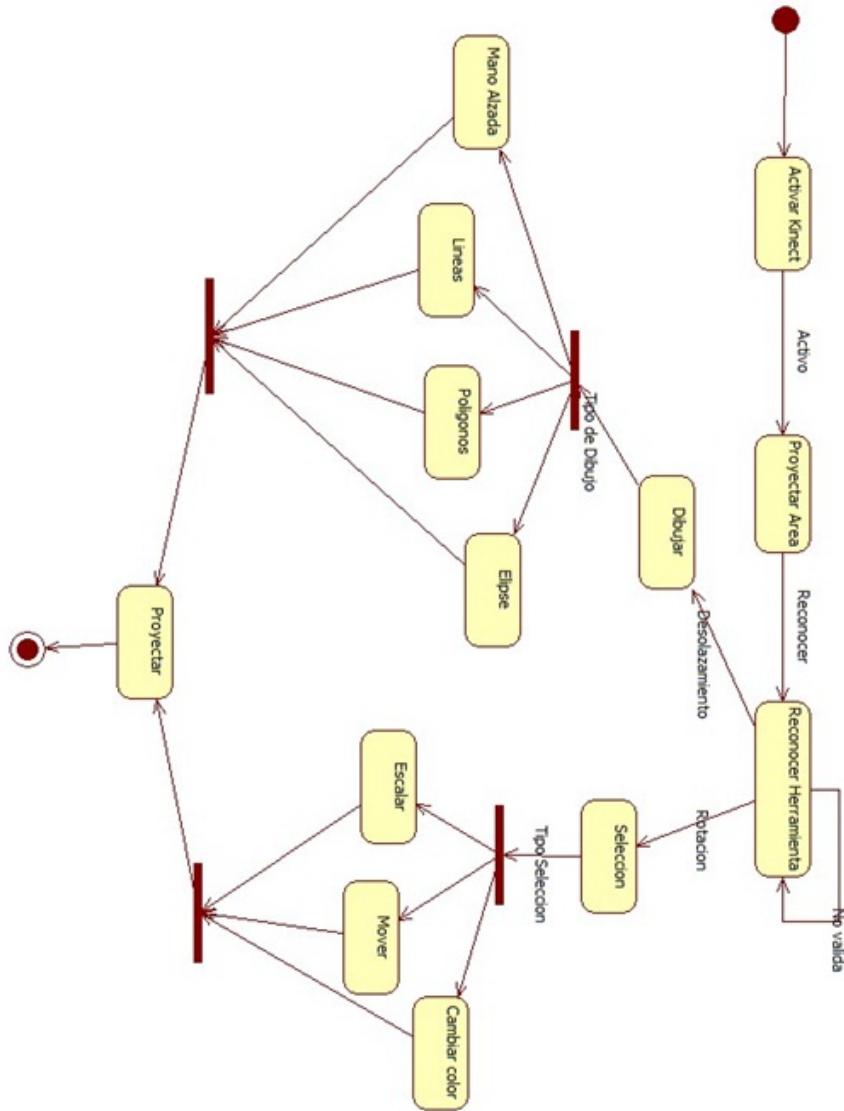


Figura 3.5: Diagrama General de Estados.

3.3. Diagramas del Módulo 1

En esta sección se mostrarán los diagramas del Módulo 1, que se llevaron acabo para la realización del módulo, los cuales son: de casos de uso(Figura 3.6), de clases(Figura 3.7), de secuencia(Figura 3.8) y de estados(Figura 3.9).

3.3.1. Diagrama de Casos de Uso - Módulo 1.

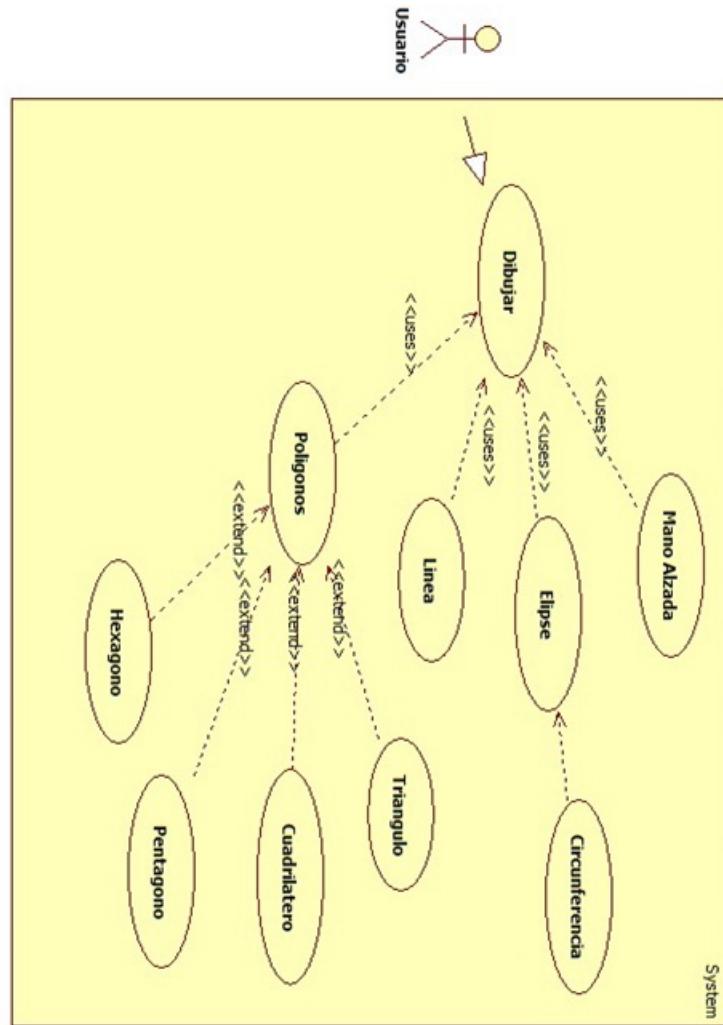


Figura 3.6: Diagrama de Casos de Uso - Módulo 1.

3.3.2. Diagrama de clases - Módulo 1.

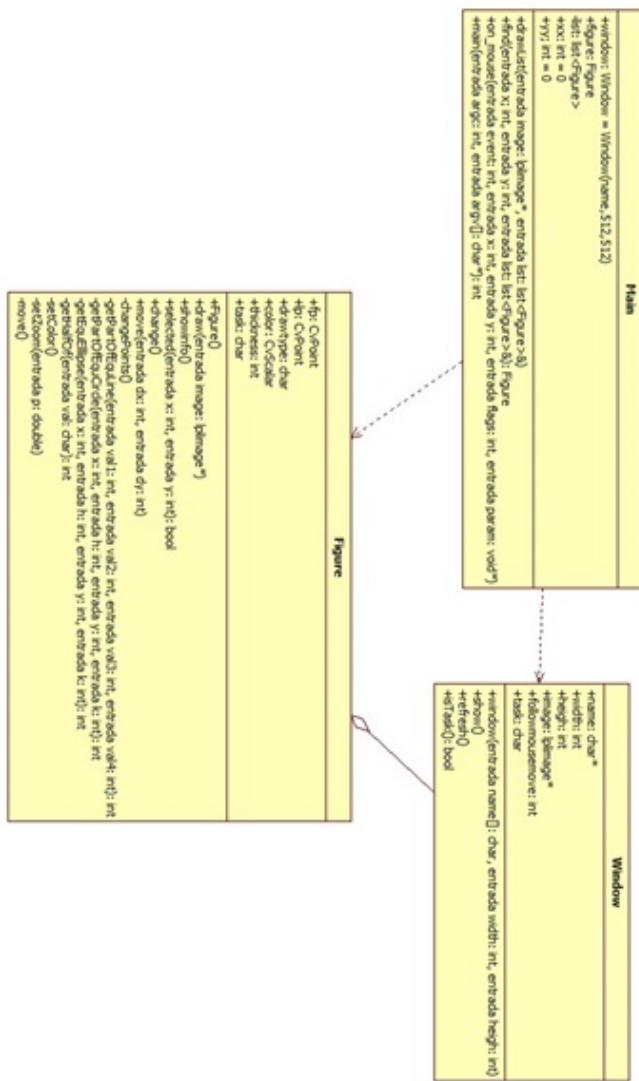


Figura 3.7: Diagrama de clases - Módulo 1.

3.3.3. Diagrama de Secuencia - Módulo 1.

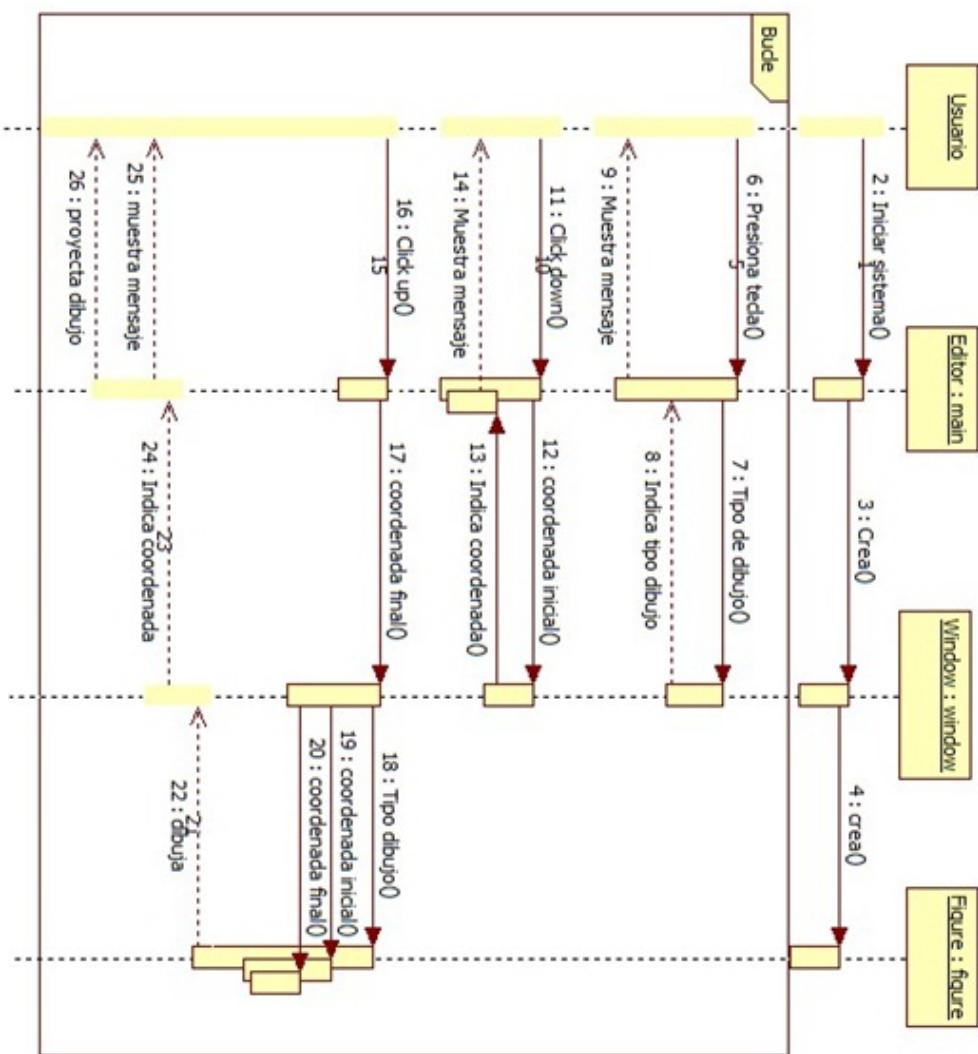


Figura 3.8: Diagrama de Secuencia - Módulo 1.

3.3.4. Diagrama de Estados - Módulo 1.

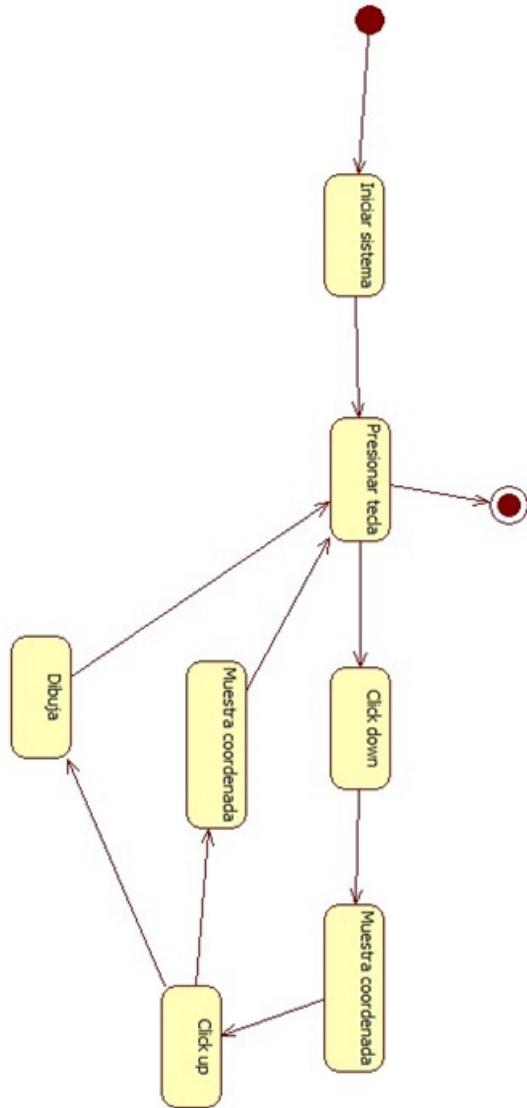


Figura 3.9: Diagrama de Estados - Módulo 1.

3.4. Diagramas del Módulo 2

En esta sección se mostrarán los diagramas del Módulo 2, que se llevaron acabo para la realización del módulo, los cuales son: de casos de uso(Figura 3.10), de clases(Figura 3.11), de secuencia(Figura 3.12) y de estados(Figura 3.13).

3.4.1. Diagrama de Casos de Uso - Módulo 2.

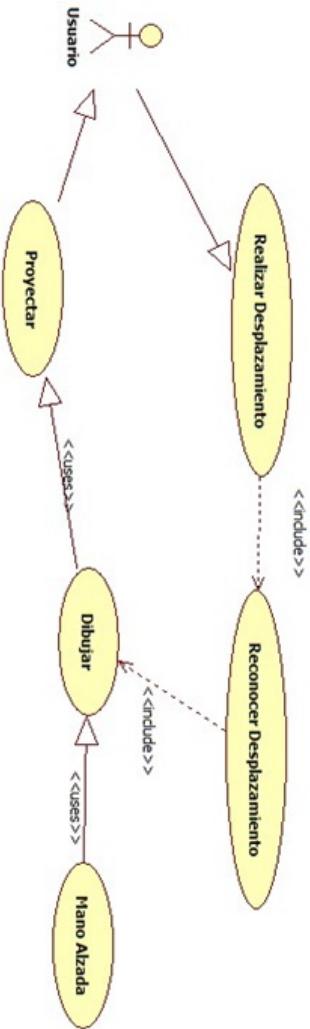


Figura 3.10: Diagrama de Casos de Uso - Módulo 2.

3.4.2. Diagrama de Clases - Módulo 2.

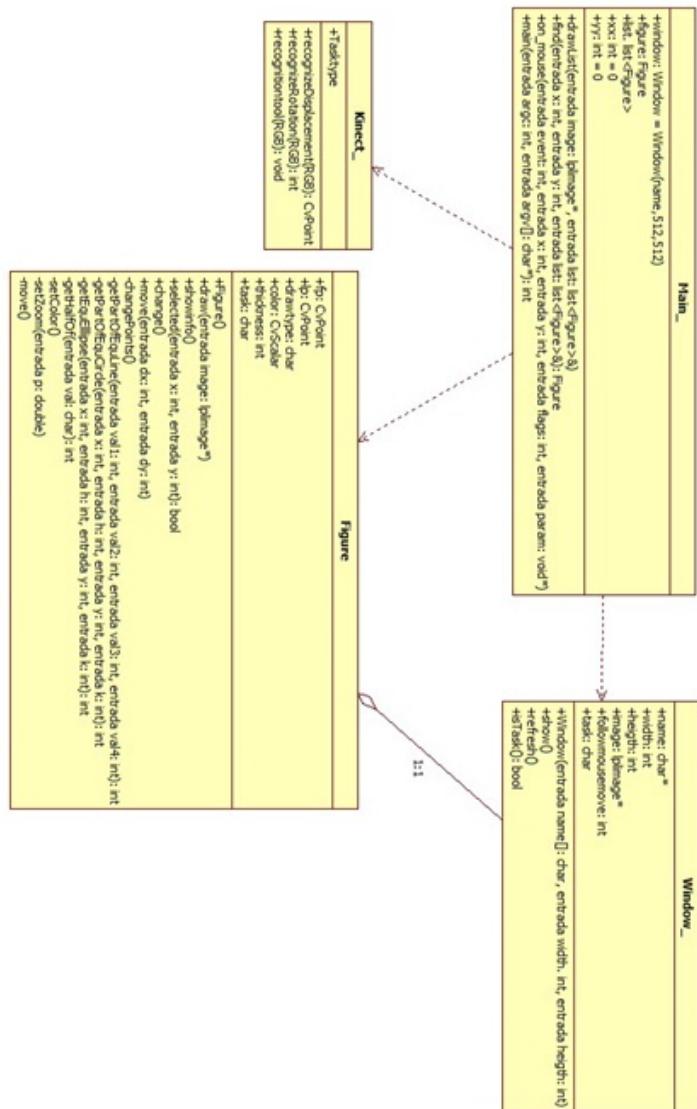


Figura 3.11: Diagrama de Clases - Módulo 2.

3.4.3. Diagrama de Secuencias - Módulo 2.

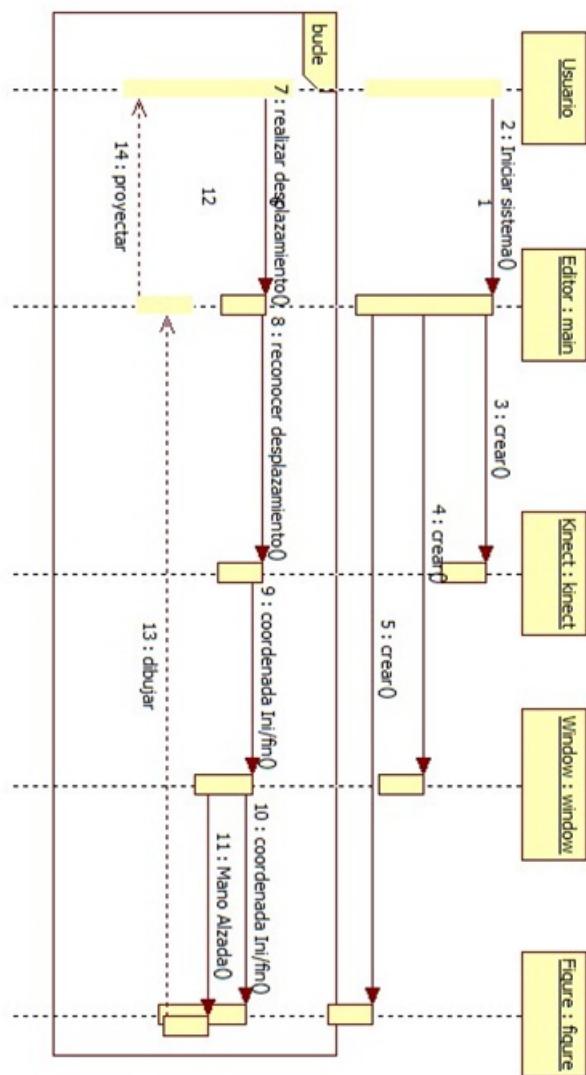


Figura 3.12: Diagrama de Secuencias - Módulo 2.

3.4.4. Diagrama de Estados - Módulo 2.

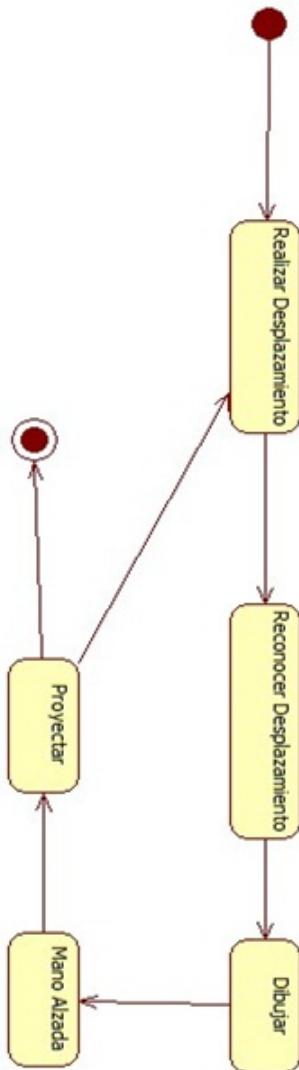


Figura 3.13: Diagrama de Estados - Módulo 2.

3.5. Diagramas del Módulo 3

En esta sección se mostrarán los diagramas del Módulo 3, que se llevaron acabo para la realización del módulo, los cuales son: de casos de uso(Figura 3.14), de secuencia(Figura 3.15) y de estados(Figura 3.16).

3.5.1. Diagrama de Casos de Uso - Módulo 3.

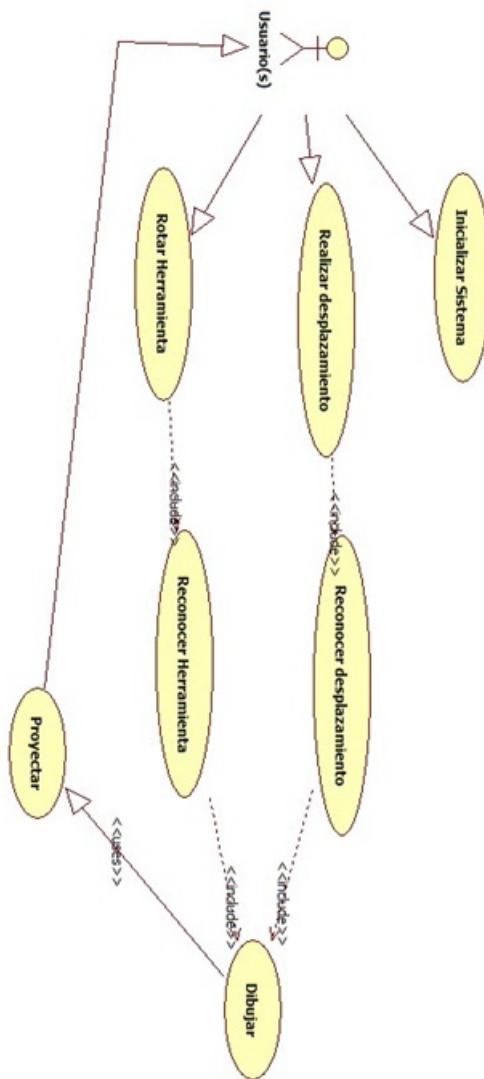


Figura 3.14: Diagrama de Casos de Uso - Módulo 3.

3.5.2. Diagrama de Secuencia - Módulo 3.

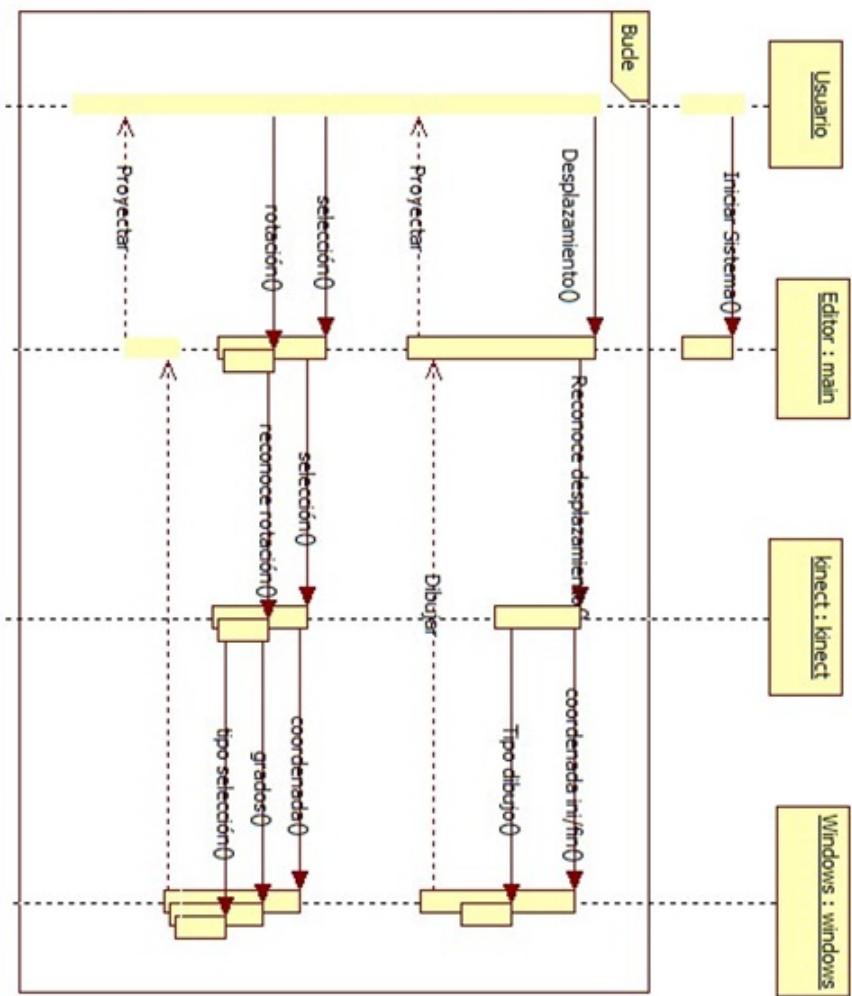


Figura 3.15: Diagrama de Secuencia - Módulo 3.

3.5.3. Diagrama de Estados - Módulo 3.

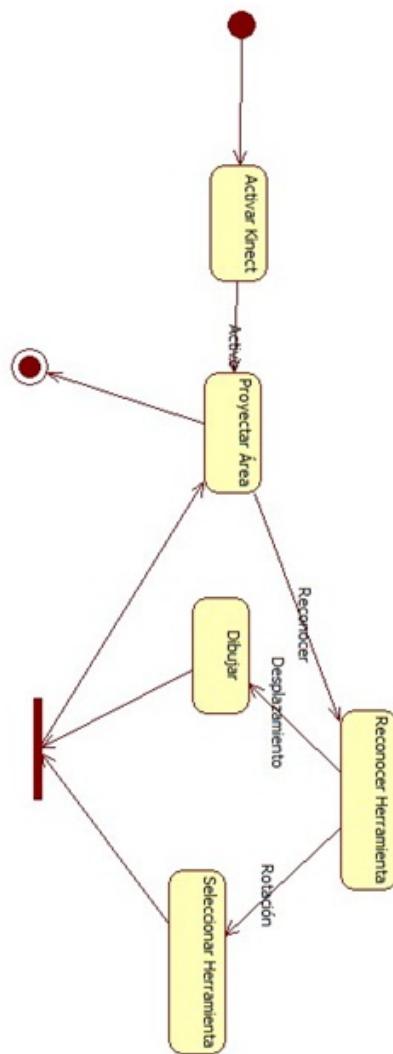


Figura 3.16: Diagrama de Estados - Módulo 3.

3.6. Diagramas del Módulo 4

En esta sección se mostrarán los diagramas del Módulo 4, que se llevaron acabo para la realización del módulo, los cuales son: de casos de uso(Figura 3.17), de secuencia(Figura 3.18) y de estados(Figura 3.19).

3.6.1. Diagrama de Casos de Uso - Módulo 4.

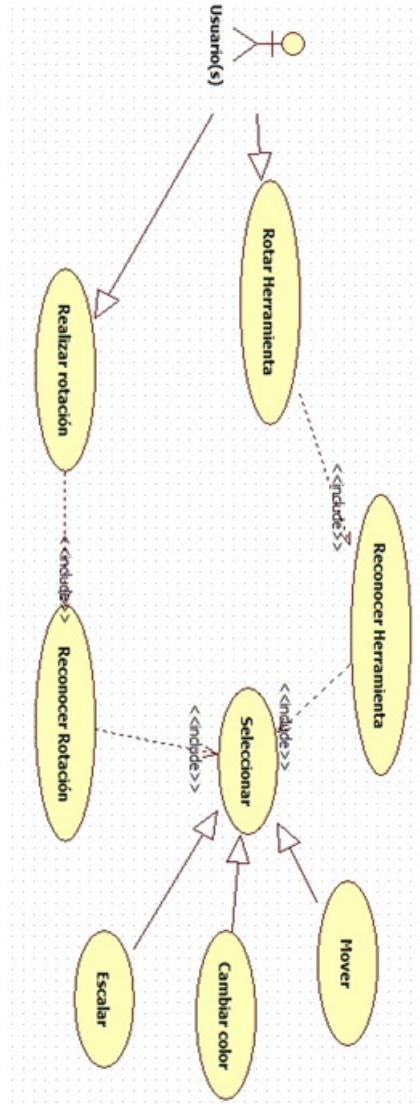


Figura 3.17: Diagrama de Casos de Uso - Módulo 4.

3.6.2. Diagrama de Secuencia - Módulo 4.

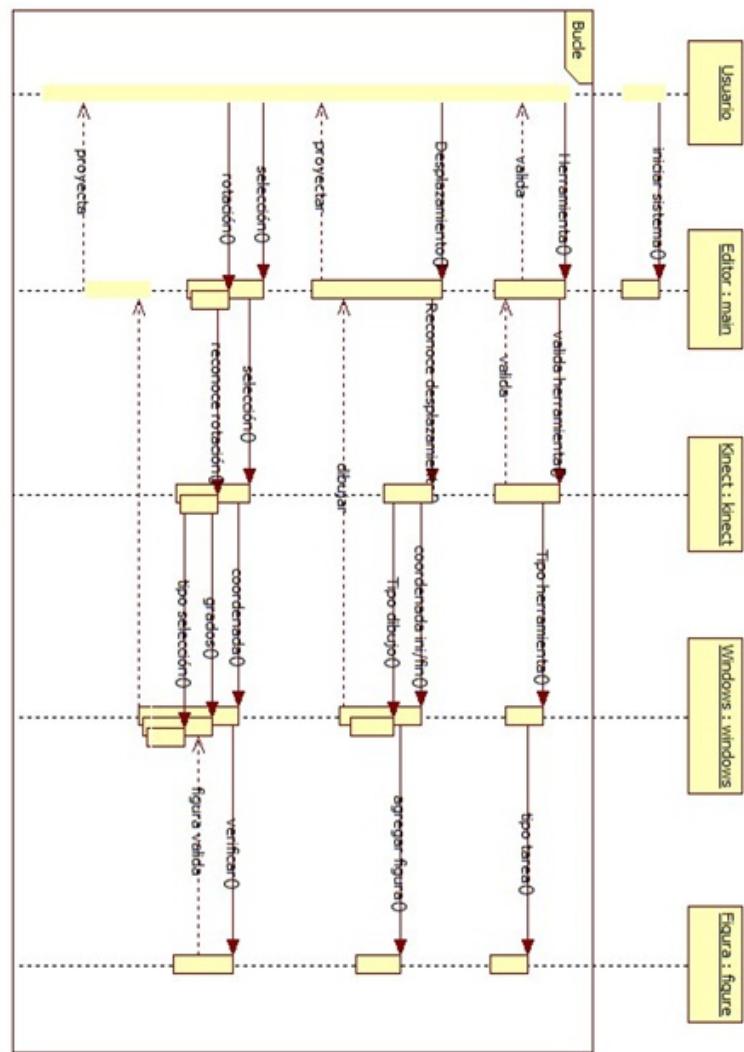


Figura 3.18: Diagrama de Secuencia - Módulo 4.

3.6.3. Diagrama de Estados - Módulo 4.

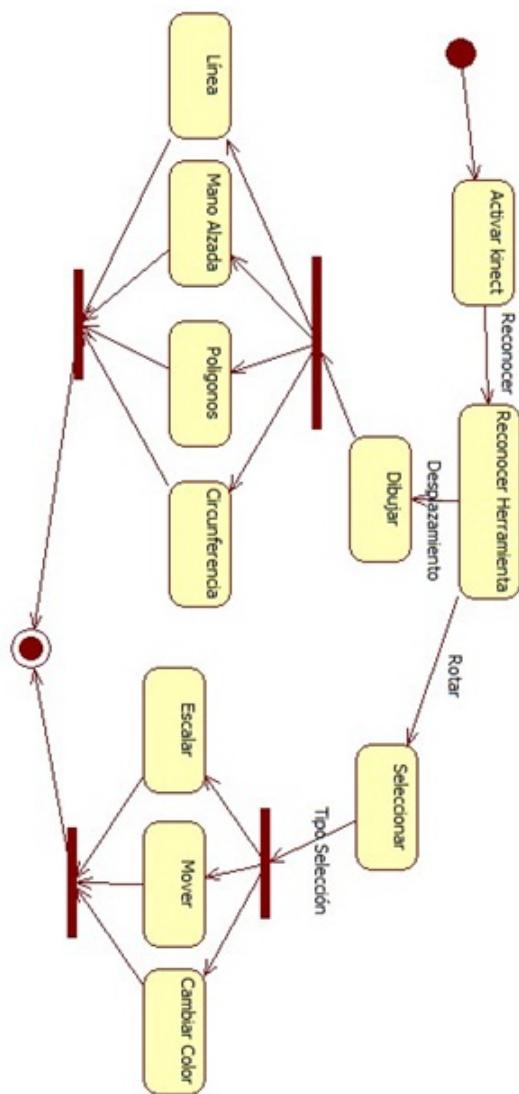


Figura 3.19: Diagrama de Estados - Módulo 4.

Capítulo 4

Desarrollo

Este capítulo explica el desarrollo del sistema en cada uno de los módulos. Se explica como se hizo el reconocimiento de las etiquetas (*Tags*) y el seguimiento del dedo (*fingertracking*) para poder realizar trazos.

4.1. Editor de dibujo básico.

La realización de este módulo no llevo mayor problema, ya que la *API (OpenCV)* que utilizamos, ofrece algunas funciones tanto para el dibujo de ventanas para la interfaz como para el dibujo de líneas y rectángulos, que fueron utilizadas para que se realizaran los trazos.

Para visualizar los trazos se tomaron dos puntos, un inicial y un final, que por medio de estos; para los trazos de línea recta (función *cvLine*) y rectángulo (función *cvRectangle*) se pasan como parámetros a las funciones respectivas, mientras que para las demás figuras se toman los mismos dos puntos de inicio-fin y se procedio a hacer los cálculos descritos en la sección de análisis y descripción de procesos del módulo uno. De esta manera se determinan puntos que se unen con líneas con las funciones que provee *OpenCV*: *cvPoint* y *cvLine*(Figura 4.1).

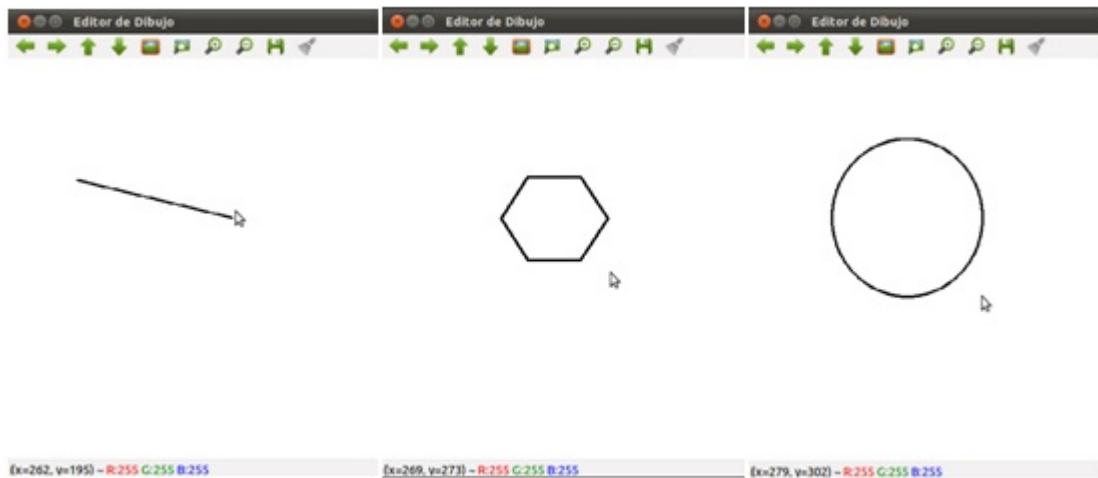


Figura 4.1: Editor de dibujo básico

4.2. Reconocimiento de trazos a mano alzada.

Para realizar el reconocimiento de los trazos con el *KinectTM*, se calculan las características de la mano del usuario con los métodos *cvMoments* y *cvHuMoments*.

Los momentos son propiedades numéricas que se pueden obtener de una determinada imagen. Tienen en cuenta todos los pixeles de la imagen, no solo los bordes[20][21].

El momento que se utilizó fue el momento central que hace referencia al área (m00).

Una vez calculados los momentos, calculamos los momentos invariantes de *Hu*, un conjunto de siete momentos invariantes. Estos momentos se mantienen invariantes ante rotaciones, traslaciones y cambios de escalas de objetos. Se define mediante una serie de ecuaciones[21]. Del cual solo usamos el momento invariante uno, correspondiente a la rotación.

Con el momento de área y el primer invariante de *Hu* (m00 y hu1) comparamos estos datos para poder identificar el dedo índice y conforme a esto se realiza los trazos. Lo que se le llama un reconocimiento de patrones supervisado.

4.3. Implementación de herramientas físicas.

Esta sección, se enfoca en explicar cual fue la tag elegida de las analizadas y cómo se hace el reconocimiento de la misma para la selección de alguna opción del editor de dibujo.

4.3.1. Elección de tag

La *tag* elegida contiene rasgos mínimos de los *reactIVision fiductials*. La cual está formada por dos círculos uno contenido dentro del otro, podemos notar que el círculo interior se encuentra separado del perímetro del círculo exterior, evitando colocarlo en el centro, esto permitirá identificar si la tag sufrió alguna rotación, así saber que funciones (color, dibujos, selección o zoom) se están trabajando y por consecuente ver reflejado esta funcionalidad en el área de trabajo.

La *tag* está formada por dos caras, la inferior que contiene los rasgos para realizar el reconocimiento y la superior que contiene pequeños dibujo representativos en la funcionalidad del sistema.

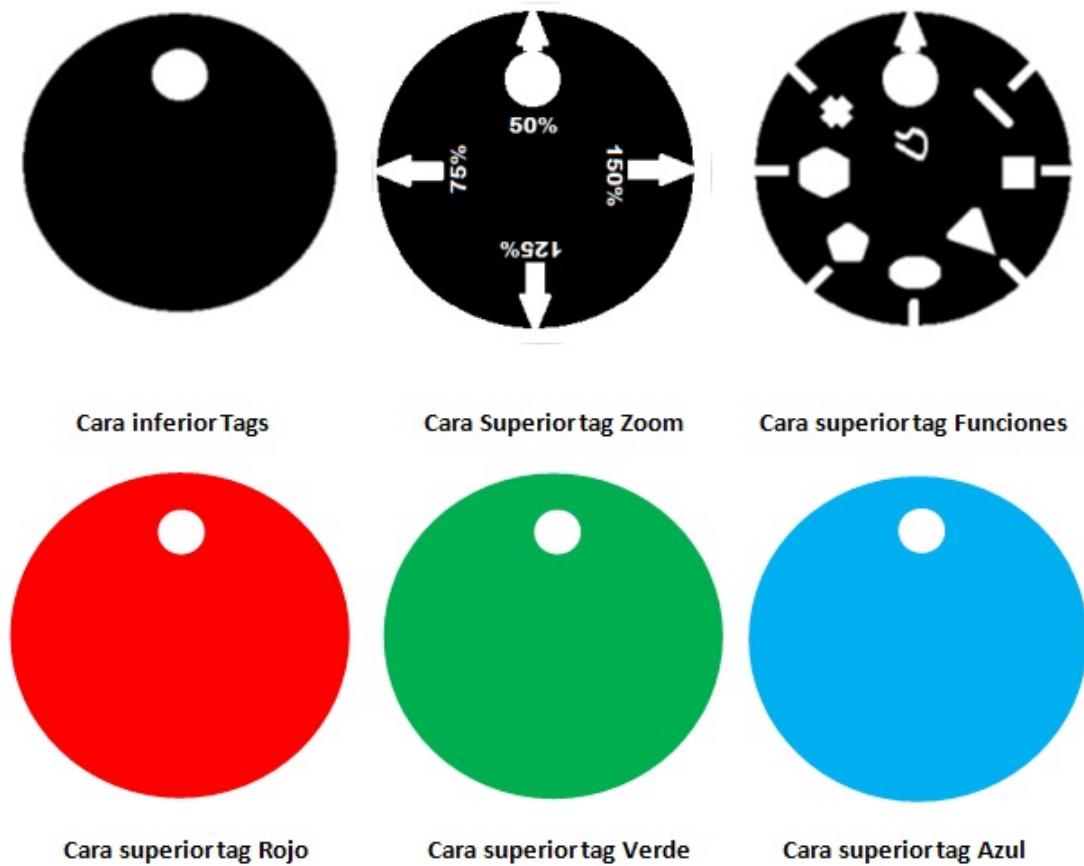


Figura 4.2: Vista de las Tag's

4.3.2. Reconocimiento de Tag's

Para hacer el reconocimiento de la imagen se realizaron los siguientes procesos.

1. La captura de la imagen es con ayuda del *KinectTM*, al cual se le indica que la imagen a capturar será del tipo *RGB*.
2. La imagen capturada la cambiamos a grises con ayuda del método *cvCvtColor*, el cual multiplica la tonalidad del pixel con los siguientes valores R*0.299, G*0.587 y B*0.114 los cuales se suman y se asignan al mismo pixel [22].
3. La imagen ya en escalas grises se le aplica un suavizado *Gaussiano*, para realizar este proceso se utilizó el método *cvSmooth* al cual se le indica sea de tipo gaussiano con *CV_GAUSSIAN*.

El operador de suavizado *Gaussiano* es un operador de convolución bidimensional que es usado para difuminar imágenes, eliminar detalles y remover el ruido en la imagen. La convolución es realizada por una máscara que representa la función de distribución Gaussiana. La función de distribución Gaussiana unidimensional tiene la forma[23](1):

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma^2} e^{\frac{(-x^2)}{(2\sigma^2)}} \quad (1)$$

Donde σ es la desviación estándar de la distribución. En dos dimensiones tenemos la forma (2):

$$G(x, y) = \frac{1}{\sqrt{2\pi}\sigma^2} e^{\frac{(-x^2-y^2)}{(2\sigma^2)}} \quad (2)$$

La idea del suavizado Gaussiano es usar esta distribución bidimensional como una función de “punto de propagación” y esta es llevada a cabo mediante una operación de convolución. Ya que una imagen es almacenada como una colección discreta de píxeles necesitamos realizar una aproximación discreta de la función antes de poder ejecutar la convolución.[23] El grado de suavizado esta determinado por la desviación estándar σ

- Después de suavizar la imagen la Binarizamos, es decir que únicamente tendremos dos colores de la imagen, a Blanco y Negro. Para poder realizar este proceso debemos de calcular un umbral “T” el cual nos permitirá identificar si un pixel cambia a Blanco o Negro(3).[24]

$$g(x, y) = \begin{cases} 1 & \text{si } f(x, y) > T \\ 0 & \text{si } f(x, y) \leq T \end{cases} \quad (3)$$

Para realizar este proceso se utilizó el método *cvThreshold* del tipo *OTSU*.

La umbralización es una técnica de segmentación ampliamente utilizada en las aplicaciones industriales. Se emplea cuando hay una clara diferencia entre los objetos a extraer respecto del fondo de la escena.[25]

Al aplicar un umbral, T, la imagen en escala de grises, $f(x,y)$, quedará binarizada; etiquetando con '1' los píxeles correspondientes al objeto y con '0' aquellos que son del fondo.[25]

Una imagen es una función bidimensional de la intensidad del nivel de gris, y contiene N píxeles cuyos niveles de gris se encuentran entre 1 y L. El número de píxeles con nivel de gris i se denota como f_i , y la probabilidad de ocurrencia del nivel de gris i en la imagen está dada por (4). [25]

$$p_i = \frac{f_i}{N} \quad (4)$$

En el caso de la umbralización en dos niveles de una imagen (a veces llamada binarización), los píxeles son divididos en dos clases: C_1 , con niveles de gris $[1, \dots, t]$; y C_2 , con niveles de gris $[t+1, \dots, L]$. Entonces, la distribución de probabilidad de los niveles de gris para las dos clases son[25]:

$$\begin{aligned} C_1 : & \frac{p_1}{\omega_1(t)}, \dots, \frac{p_t}{\omega_1(t)} \\ C_2 : & \frac{p_{t+1}}{\omega_2(t)}, \frac{p_{t+2}}{\omega_2(t)}, \dots, \frac{p_L}{\omega_2(t)} \end{aligned}$$

Donde

$$\begin{aligned} \omega_1(t) &= \sum_{i=1}^t p_i \\ \omega_2(t) &= \sum_{i=t+1}^L p_i \end{aligned}$$

La media para la clase $C1$ y la clase $C2$

$$\mu_1 = \sum_{i=1}^t \frac{ip_i}{\omega_1(t)}$$

$$\mu_2 = \sum_{i=t+1}^L \frac{ip_i}{\omega_2(t)}$$

Sea μ_T la intensidad media de toda la imagen se demuestra que

$$\begin{aligned}\omega_1\mu_1 + \omega_2\mu_2 &= \mu_T \\ \omega_1 + \omega_2 &= 1\end{aligned}$$

Otsu definió la varianza entre clases de una imagen umbralizada como

$$\sigma_B^2 = \omega_1(\mu_1 - \mu_T)^2 + \omega_2(\mu_2 - \mu_T)^2$$

Para una umbralización de dos niveles, *Otsu* verificó que el umbral óptimo $t*$ se elige de manera que σ_B^2 sea máxima; esto es

$$\begin{aligned}t* &= \text{Max}_t\{\sigma_B^2(t)\} \\ 1 \leq t &\leq L\end{aligned}$$

5. Después de tener la imagen binarizada, se obtienen todos sus contornos es decir todos aquellos elementos que son de color blanco dentro de imagen, para obtener todos los contornos y sus elementos dentro del se utiliza el método `cvFindContours`, este método utiliza el método de contornos de *Suzuki*[26].
6. Se calculan las características de cada contorno con los métodos `cvMoments` y `cvHuMoments`

Los momentos son propiedades numéricas que se pueden obtener de una determinada imagen. Tienen en cuenta todos los pixeles de la imagen, no solo los bordes[20][21].

El momento que se utilizó fue el momento central que hace referencia al área ($m00$).

Una vez calculados los momentos, calculamos los momentos invariantes de Hu, un conjunto de siete momentos invariantes. Estos momentos se mantienen invariantes ante rotaciones, traslaciones y cambios de escalas de objetos. Se define mediante las siguientes ecuaciones[21]. Del cual solo usamos el momento invariante uno.

7. Con el momento de área y el primer invariante de *Hu* ($m00$ y $hu1$) comparamos estos datos para poder identificar el tipo de *Tag*. Lo que se le llama un reconocimiento de patrones supervisado
8. Dentro de la imagen que se capturó y se realizó todo el proceso anterior, identificaremos 5 tags con la misma característica, para seleccionar las diferentes funcionalidades que cada una contendrá.
9. Una vez procesada la imagen capturada e identificado cada tag, se segmentará para poder identificar la posición que ocupa el circulo interior, esto permitirá especificar la tarea asignada a dicha posición.(Figura 4.3)



Figura 4.3: Tag segmentada.

Capítulo 5

Pruebas

Se presentan las pruebas realizadas al sistema con el objetivo de verificar su funcionamiento.

También se muestran los resultados de dichas pruebas. Para ello se uso un plan de pruebas del tipo caja negra, en la cual se tiene un conjunto de entradas y se verificaba si la salida es la correcta, ya que el proceso interno de las funciones no es complejo se descarto la posibilidad de evaluar con un plan de pruebas del tipo caja blanca[27].

5.1. Pruebas - Módulo 1: Editor básico de dibujo

En esta sección se muestran las pruebas que se llevaron a cabo en el Modulo 1, para determinar si cumple con todos los requerimientos y condiciones de satisfacción, para verificar si la funcionalidad es la correcta. En el Cuadro 5.1 se muestra los resultados de las pruebas que se llevaron a cabo para el modulo y las correcciones para cumplir con las condiciones y requerimientos:

No. de Prueba	Requerimientos Probados	Entrada	Resultados	Correciones
1	R1:Trazo Mano Alzada	Posición del puntero	Al aumentar la velocidad del desplazamiento del mouse, no traza en algunos intervalos	Cambiar la forma en que se obtienen las coordenadas del puntero
2	R2:Trazar Líneas Rectas R3:Trazar líneas rectas en varias direcciones	Posición del puntero al presionar y Posición del puntero al soltar	Trazo correcto	Ninguna
3	R4:Trazar Circunferencia	Posición del puntero al presionar y Posición del puntero al soltar	No traza circunferencia si el trazo empieza de abajo hacia arriba	Validar para cualquier posición
4	R5:Trazar Elipse	Posición del puntero al presionar y Posición del puntero al soltar	Elipse incompleta	Modificar los arcos a 180 grados de cada eje
5	R6:Trazar Polígonos R7:Tipo de polígono de 3-6 lados	Posición del puntero al presionar y Posición del puntero al soltar	Al trazar polígonos de 5-6 lados, la figura se deforma si es trazo es de abajo hacia arriba	Validar para cualquier posición y precisar un poco más los vértices

Cuadro 5.1: Pruebas realizadas al Módulo 1.

5.2. Pruebas - Módulo 2: Reconocimiento de trazos a mano alzada

En esta sección se muestran las pruebas que se llevaron a cabo en el Modulo 2, para determinar la funcionalidad y el cumplimiento de los requerimientos, para verificar si la funcionalidad es la correcta. En el Cuadro 5.2 se muestra los resultados de las pruebas que se llevaron a cabo para el modulo y las correcciones para cumplir con las condiciones y requerimientos:

No. de Prueba	Requerimientos Probados	Entrada	Resultados	Correcciones
1	R8:Integracion de kinect	Verificar la comunicación de kinect a PC	El dispositivo kinect y la PC no se comunican entre si	Verificar si el driver y sus respectivos paquetes están instalados en las rutas correctas
2	R9:Detección del dedo índice	Verificar si al colocar la mano dentro del área de visión del kinect se identifica el dedo índice	El algoritmo no funciona correctamente únicamente se obtiene la palma de la mano como punto de referencia	Modificar el cálculo al segmentar la imagen y verificar si el contorno de la mano está completamente cubierto
3	R10:Renocer Desplazamiento del dedo índice	Verificar la posición del dedo índice al desplazarse dentro de la visión del kinect	Únicamente se tiene el seguimiento de la palma.	Corregir el R9, ya que se depende del mismo
4	R11:Dibujar a mano alzada con el dedo índice	Desplazar el dedo índice dentro de la visión del kinect simulando un trazo a mano alzada	Se dibujo la trayectoria, sin embargo el seguimiento fue de la palma	Corregir el R9
5	R12:Dibujar dentro del módulo 1	Proyectar el trazo obtenido en el R11 dentro del modulo 1	Error en la Conexión	Verificar el envío de paquetes

Cuadro 5.2: Pruebas realizadas al Módulo 2.

5.3. Pruebas - Módulo 3: Proyección sobre el área de trabajo

En esta sección se muestran las pruebas que se llevaron a cabo en el Modulo 3, para determinar la funcionalidad y el cumplimiento de los requerimientos, para verificar si la funcionalidad es la correcta. En el Cuadro 5.3 se muestra los resultados de las pruebas que se llevaron a cabo para el modulo y las correcciones para cumplir con las condiciones y requerimientos:

No. de Prueba	Requerimientos Probados	Entrada	Resultados	Correcciones
1	R13: Trabajar conjuntamente proyector y Kinect	Vidrio	Se puede trabajar conjuntamente sin problema alguno.	Ninguna
2	R14: Proyectar sobre el área de trabajo	Vidrio	La imagen de salida traspasa el vidrio	Cambiar la superficie
3	R14: Proyectar sobre el área de trabajo	Vidrio polarizado	La captura de la cámara se reduce	Cambiar superficie
4	R15: Trabajar sin interferencia de la sombra que produce la mano	Vidrio Polarizado	Se puede trabajar pero se reduce la velocidad de captura de la tag por la condición de luz	Cambiar superficie.
5	R14: Proyectar sobre el área de trabajo	Pared	Imagen proyectada, no es la más apta, solución temporal	Identificar otro tipo de superficie adecuado y de bajo costo
6	R15: Trabajar sin interferencia de la sombra que produce la mano	Pared	No afecta la sombra ya que no se proyecta directamente en el área de trabajo	Identificar otro tipo de superficie adecuado y de bajo costo.

Cuadro 5.3: Pruebas realizadas al Módulo 3.

5.4. Pruebas - Módulo 4: Implementación de herramientas físicas

En esta sección se muestran las pruebas que se llevaron a cabo en el Modulo 4, para determinar la funcionalidad y el cumplimiento de los requerimientos, para verificar si la funcionalidad es la correcta. En el Cuadro 5.4 se muestra los resultados de las pruebas que se llevaron a cabo para el modulo y las correcciones para cumplir con las condiciones y requerimientos:

No. de Prueba	Requerimientos Probados	Entrada	Resultados	Correcciones
1	R16:Reconocimiento de la herramienta	Tag de alguna herramienta	Las herramientas se reconocieron a excepción de línea y mano alzada	Cambiar el grosor del dibujo interno de la tag
2	R17:Dibujar con el uso de la herramienta	Colocar la tag frente a la cámara y después realizar desplazamiento de mouse sobre el área de trabajo	Trazo correcto	Ninguna
3	R18,R19,R20:Cambios de la figura con el uso de la herramienta	Colocar la tag que indica modificación y realizar un click dentro de la figura	No se reconoció las tags de modificación	Modificar dibujo interior de las herramientas
4	R16:Reconocimiento de la herramienta	Tag Mano alzada	Reconocida	Ninguna.
5	R16:Reconocimiento de la herramienta	Tag Linea	Reconocida	Ninguna
6	R18,R19,R20:Cambios de la figura con el uso de la herramienta	Colocar la tag que indica modificación y realizar un click dentro de la figura	Reconoció tag pero no identifico giro	Modificar método de reconocimiento de rotación.

Cuadro 5.4: Pruebas realizadas al Módulo 4.

Capítulo 6

Conclusiones

Se presentan las conclusiones a las que se llegaron después del desarrollo del trabajo terminal.

6.1. Generales

Actualmente las interfaces de usuario natural están acaparando el mercado de la tecnología desde teléfonos móviles hasta computadoras personales intentando hacer de manera más sencilla la interacción entre ellas y los usuarios, eliminando dispositivos intermediarios y que el usuario pueda manejar programas o sistemas como si estuviera haciendo la actividad regular, como si no contara con el dispositivo, por ejemplo escribir o dibujar.

Con la realización de este trabajo terminal se ofrece un sistema con una alternativa al mouse y teclado convencionales y una interacción diferente ya que la interfaz no cuenta con botones sino con herramientas físicas para la selección de las distintas opciones.

Trabajando con *Kinect*TM nos dimos cuenta que tan rápido avanza la tecnología y el desarrollo de la industria computacional ya que cuando se planteó el proyecto no se contaba con documentación fiable ni con los drivers oficiales por parte de *Microsoft*® y en tan solo unos meses ya había bastantes desarrollos con este dispositivo que aumentaron con la liberación del entorno de desarrollo y los drivers oficiales.

6.2. Individuales

- En el presente trabajo terminal, se puede hacer mención del conocimiento adquirido en el reconocimiento de patrones, configuración de dispositivos que fueron diseñados con un fin diferente y la integración de distintas herramientas que permiten cumplir con un cierto fin, sin embargo no debemos olvidar mencionar la parte administrativa del proyecto, en la cual se encuentra involucrada un grupo de personas con capacidades y cualidades distintas, las cuales deben de trabajar en conjunto para un fin común, asignar a cada miembro en el área que más destaque y al mismo tiempo compartir ese conocimiento con aquellas que carecen del mismo, permitiendo retroalimentar al equipo de trabajo, el cual tendrá como efecto ir la mismo ritmo.

También cabe mencionar la experiencia adquirida en ponderar un proyecto, referenciar conocimientos ajenos, segmentarlo en pequeños módulos, los cuales permitirán atacar el problema en sus particularidades dando como resultado una solución general a partir de la suma de las anteriores. El conjunto de todos estos conocimientos nos da como resultado una mejor integración laboral, dicho de otra manera, la capacidad de trabajar con personas

que comparten diferentes conocimientos.

Fernández Hernández Daniel.

- Durante el trascurso del desarrollo del sistema, fortalecí los conocimientos que fui adquiriendo en la carrera, así mismo obtuve nuevos, que sirvieron en mi desarrollo como persona, no solamente en el ámbito profesional, sino también para ampliar mi breviario cultural, ya que al inicio parte del conocimiento me era ambiguo. Además mejorar los escasos ó pocos conocimientos que involucra la organización de un sistema, desde lo técnico hasta lo administrativo, qué conlleva la planeación para cumplir con lo que se esta comprometiendo el sistema en la fecha indicada.

Concluyo que se está ante un sistema, que puede servir como base para otras personas, que quieran incursionar con dispositivos como el kinect, que ha tenido está evolución, y que ha pasado de ser un simple dispositivo para videojuegos, a tener diferentes alternativas en distintos ámbitos, que puede, no solo servir como entretenimiento, si no para el avance en diferentes de áreas.

Hernández Guerrero Javier Irving.

- Al finalizar el trabajo terminal, primeramente se ha adquirido el conocimiento para realizar un análisis y diseño para el desarrollo de un sistema de cómputo real, posteriormente se aprendió a realizar la configuración entre el dispositivo kinect y la computadora con lo cual se pudieron demostrar ciertas ventajas del dispositivo kinect además de descubrir algunas nuevas, como el hecho de que permite abrir varias capturas de las cámaras simultáneamente, inclusive desde procesos diferentes, cualidad que permite dividir los sistemas en módulos aun mas pequeños para un desarrollo mas rápido y de fácil integrar. Se adquirieron conocimientos acerca del tratamiento de imágenes mediante filtros que permiten facilitar el reconocimientos de patrones. Fuera de la parte técnica, se adquirió experiencia sobre el trabajo en equipo y el beneficio de compartir ideas con otros compañeros. Se pudo demostrar la capacidad que tienen los alumnos para hacer frente a complicaciones al momento de realizar el sistema, experiencia que se integra a la formación profesional.

Hernández Hernández Alex.

- Durante el desarrollo de este trabajo terminal conocimos alternativas al mouse y teclado pero no alguna que ofrezca una interacción sin tener contacto con algún dispositivo electrónico. Esta investigación que realizamos para ofrecer una interfaz de usuario natural es un paso más para que se realicen sistemas donde ya no se tenga que trabajar con un mouse o teclado, espero que este trabajo funcione como base para que en un futuro próximo las interfaces naturales sean las dominantes en el mercado.

Además aprendimos la organización que se debe de tener para el desarrollo de un sistema, nos sirvió de experiencia para en el futuro sepamos cómo realizar desde una investigación si es que no contamos con los conocimientos o es una tecnología reciente como lo fue Kinect hasta la implementación de un sistema con las nuevas tecnologías.
Sánchez Ramírez Gustavo Rafael.

6.3. Trabajo a futuro

Se deja como base un trabajo de investigación para la correcta instalación y configuración de una computadora personal para trabajar con *KinectTM* y poder realizar una aplicación o un sistema más robusto, ya que una de las finalidades de este trabajo era investigación acerca de cómo manejar *KinectTM* con una computadora.

Dirigirlo hacia un sector en específico de la población solucionando un problema o proponiendo una alternativa sólida para desplazar poco a poco a los *hardware* directos, teclado y mouse, y no tener que depender en su totalida de ellos en un sistema.

Capítulo 7

Bibliografía

Bibliografía

- [1] Online. Systems, Intelligence, Robotics and Perception, Pontificia Universidad Javeriana Bogotá <http://www.gruposirp.org/sirp/wiki/doku.php> <http://gruposirp.org/sirp/sirp-home.html> WebSite 2011.
- [2] Presentación Kinect Bruno Capuano Visual Studio ALM <https://mvp.support.microsoft.com/profile=800D1522-8788-4A34-8985-233DBBF2A40A> Conference Cidetec IPN 2011 cortesía Ing. Naím Rivera.
- [3] Online. OpenNI™. <http://75.98.78.94/About.aspx>
- [4] Open Source Computer Vision Library. Reference Manual. Edic. Intel Corporation.
- [5] "Pattern Recognition" Sergios Theodoquidis, Konstantinos Kourtroumbus Ed. Academic Press 2º Edición 2003.
- [6] "Reconocimiento de Patrones: Enfoque Lógico Combinatorio" José Ruiz Shulcloper Ed. Instituto Politécnico Nacional.
- [7] "The Design and Evolution of Fiducials for the reacTIVision System", Ross Bencina and Martin Kaltenbrunner, Universitat Pompeu Fabra, Barcelona, España, 2005.
- [8] Online. reacTIVision, <http://reactivision.sourceforge.net/>
- [9] Online. QR code, versions 1 - 40. <http://www.denso-wave.com/qrcode/vertable1-e.html>
- [10] Online. Otsu Thresholding, The Lab bookpages <http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html>
- [11] Online. ZXing web page. <http://code.google.com/p/zxing/>
- [12] Online. ARToolkit Documentation. <http://www.hitl.washington.edu/artoolkit/documentation/userarwork.htm>
- [13] Roger S. Pressman (adaptado por Darrel Ince), Ingeniería del Software un enfoque práctico, 5ta ed. Ed. McGraw-Hill, pp. 23-24.
- [14] Síndrome del túnel carpiano — Causas y factores de riesgo <http://familydoctor.org/familydoctor/es/diseases-conditions/carpal-tunnel-syndrome/causes-risk-factors.html>
- [15] Online. Hand Tracking (Kinect with OpenCV and OpenNI). http://www.youtube.com/watch?v=qNH_MqqOPX0
- [16] Online. Kinect Active Projection Mapping. <http://www.youtube.com/watch?v=hkHUGxP3ecI>
- [17] Online. Technical University Bergakademie Freiberg <http://www.informatik.tu-freiberg.de/>
- [18] Online. Arquitectura basada en capas. Posted on May 26, 2009 by Juan Peláez <http://www.juanpelaez.com/geek-stuff/arquitectura/arquitectura-basada-en-capas/>
- [19] Online. Estilo arquitectural en capas (N-Layer) http://www.ecured.cu/index.php/Estilo_arquitectural_en_capas_28N-Layer29

- [20] "Image Description Using Moments", Dr. S. Belkasim.
- [21] "Momentos Wavelet del Tipo B-Spline Cúbicos para Clasificación de Objetos", Judith Pérez Marcial, Benemérita Universidad Autónoma de Puebla, 2005.
- [22] Online. Manual de referencia OpenCV. http://www.seas.upenn.edu/bensapp/opencvdocs/ref/opencvref_cv.htm
- [23] J. R. Parker., Algorithms for Image Processing and Computer Vision, Wiley,1997.
- [24] Online. "Técnicas de procesamiento digital de imágenes y reconocimiento de patrones." http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/nieto_b_d/capitulo2.pdf
- [25] Online "Segmentación por Umbralización", Universidad Nacional de Quilmes, 2005 <http://iaci.unq.edu.ar/materias/vision/archivos/apuntes/SegmentaciC3B3n20por20umbralizaciC3B3n20-20MC3A9todo20de20Otsu.pdf>
- [26] "Optimizing Connected Component Labeling Algorithms" Kensheng Wu, Ekow Otoo and Arie Shoshani, Lawrence Berkeley National Laboratory, Berkeley, CA, USA.
- [27] Online Prueba de Programas José A. Mañas 16 de marzo, 1994 <http://www.lab.dit.upm.es/lprg/material/apuntes/pruebas/test>

Capítulo 8

Anexo. Manual de Instalación OpenNI/OpenCV

8.1. Introducción

El objetivo de este manual es explicar la instalación del *driver OpenNi* y del *framework OpenCV*, para la utilización del dispositivo kinect, en cualquier equipo de cómputo, en este documento se citan los requerimientos previos con los que el usuario debe contar.

También se explican los pasos necesarios para una correcta instalación. El *driver* y *framework* son multiplataforma, con soporte para *Linux*, *Windows* ó *Mac OS*. Este manual sólo describe el proceso de instalación y configuración para una distribución de *Linux*. El manual incluye las versiones de los paquetes que se usaron, otras versiones pueden no funcionar, pero puede servir como previo para la búsqueda de información.

8.2. Requisitos previos

Tener instalada una distribución de *Linux*. Este manual fue probado con *Ubuntu 11.04* en 32 bits.

8.2.1. Sistema Operativo

En caso de no tener una distribución Linux los pasos básicos se pueden resumir en:

1. Se inserta el disco y se procede a instalar
2. Después nos dará una serie de opciones, de la cual usaremos la opción de algo mas
3. Se escoge la partición mas grande con la que se cuenta y se reduce el espacio
4. Nos quedara un espacio libre, el cual fue reducido en la partición, este espacio se usara para crear una nueva partición.
5. Se crea la partición, que sea de tipo lógica
6. Por Último, se le da instalar y seguimos las instrucciones del asistente

8.2.2. Gestor de paquetes

Las distribuciones incluyen gestores de paquetes (Figura 8.1) que ayudan en la instalación de bibliotecas de funciones, pero en caso de no tener un gestor instalador se pueden seguir los siguientes pasos:

1. Se abre una terminal.
2. Se escribe el siguiente comando:

```
sudo apt-get install synaptic
```

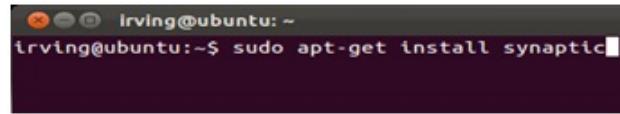


Figura 8.1: gestor de paquetes.

8.2.3. Actualización de Repositorios

Es necesario actualizar los repositorios (Figura 8.2) de la distribución, para saber si no faltan paquetes ó si hay mejoras de algunos, para ello se siguen los siguientes pasos:

1. Se abre una terminal.
2. Se escribe el siguiente comando:

```
sudo apt-get update
```

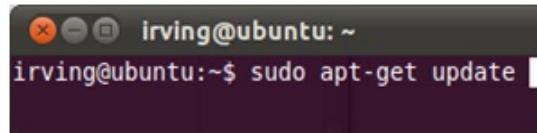


Figura 8.2: actualizar repositorios.

8.2.4. Instalación de Paqueteria

Esto son parte de los paquetes que se debe instalar, para poder trabajar con el *drive* y el *API*. para ello se siguen los siguientes pasos:

1. Se abre una terminal.
2. Se escribe el siguiente comando (Figura 8.3):

```
sudo apt-get install build-essential mercurial linux-headers-'uname -r'
```

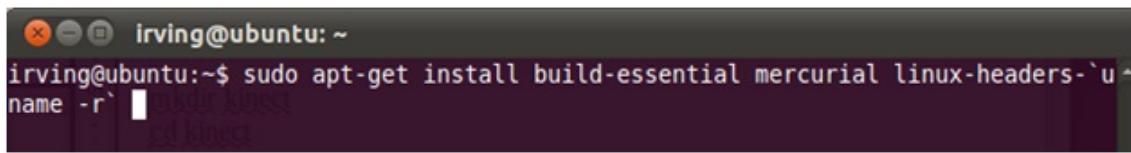
A screenshot of a terminal window titled "irving@ubuntu: ~". The user has typed the command "sudo apt-get install build-essential" and is in the process of pressing the Enter key. The terminal is dark-themed.

Figura 8.3: build-essential.

También se debe instalar *gcc*, *python*, *libusb*, *freeglut3*, *jdk*, *graphviz*, *mono*, *doxygen*. La forma que se puede instalar se puede checar directamente desde la página de *OpenNI*, pero para facilitar la tarea, a continuación se describen los comandos que se deben seguir:

1. GCC 4.x

De: <http://gcc.gnu.org/releases.html>

O a través de apt:

```
sudo apt-get install g++
```

2. Python 2.6 + / 3.x

De: <http://www.python.org/download/>

O a través de apt:

```
sudo apt-get install python
```

3. LibUSB 1.0.8

De: <http://sourceforge.net/projects/libusb/>

O a través de apt:

```
sudo apt-get install libusb-1.0-0-dev
```

4. freeglut3

De: <http://freeglut.sourceforge.net/index.php> download

O a través de apt:

```
sudo apt-get install freeglut3-dev
```

5. JDK 6.0

De: <http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u26-download-400750.html>

O a través de apt:

```
sudo add-apt-repository "deb socio http://archive.canonical.com/ lúcido"
sudo apt-get update
sudo apt-get install sun-java6-jdk
```

6. Doxygen

De: <http://www.stack.nl/~dimitri/doxygen/download.html> latestsrc

O a través de apt:

```
sudo apt-get install doxygen
```

7. GraphViz

De: http://www.graphviz.org/Download_linux_ubuntu.php

O a través de apt:

```
sudo apt-get install graphviz
```

8. Mono

De: <http://www.go-mono.com/mono-downloads/download.html>

O a través de apt:

```
sudo apt-get install mono-complete
```

8.3. Instalación de OpenNI y OpenCv

8.3.1. Pasos para la instalación de OpenNI

Paso 1 Descargue los archivos desde el siguiente link:

<http://www.openni.org/Downloads/OpenNIModules.aspx>

Se recomienda que use estos paquetes(Cuadro 8.1):

Paquete	Versión
OpenNI	openni-bin-dev-linux-x86-v1.5.2.23
Nite	nite-bin-linux-x86-v1.5.2.21
Sensor	sensor-bin-linux-x86-v5.1.0.41
SensorKinect	avin2-SensorKinect-faf4994 (Version 5.1.0.25 version - Dec 18th 2011)

Cuadro 8.1: Paquetes de OpenNI.

Todos los paquetes deben ser versiones estables, en la siguiente Figura 8.4 se muestra como se visualiza la página, así mismo estos son los nombres con los que encontraremos los paquetes *openni*, *nite* y *sensor*:

- Openni - OpenNI Binaries
- Nite - OpenNI Compliant Middleware Binaries
- Sensor - OpenNI Compliant Hardware Binaries



Figura 8.4: Pagina de Descargas de OpenNi.

- Hacer una nueva carpeta llamada Kinect (Figura 8.5).

```
$ cd
$ mkdir Kinect
$ cd Kinect
```

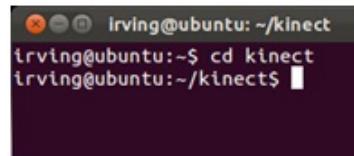


Figura 8.5: Crear carpeta Kinect.

Y extraer los archivos descargados en la carpeta *kinet*

Paso 2 Actualizar los repositorios (Figura 8.6).

```
sudo apt-get update
```

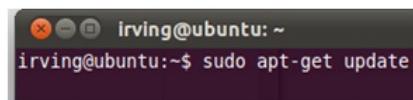


Figura 8.6: Crear carpeta Kinect.

- Instalar estos archivos que son necesarios para la correcta instalación de los *drivers*

```
$ sudo apt-get install mono-complete
$ sudo apt-get install libusb-1.0-0-dev
$ sudo apt-get install-dev freeglut3
```

Paso 3 De las carpetas extraídas en el paso 1, se les cambiaron el nombre a *OpenNI*, *Sensor*, *Nite*, respectivamente, para la parte de *SensorKinect* descarguelo del siguiente link: <https://github.com/avin2/SensorKinect>

- Dándole en la pestaña *dowloads* y extraer el paquete en la carpeta “kinect” (Figura 8.7).

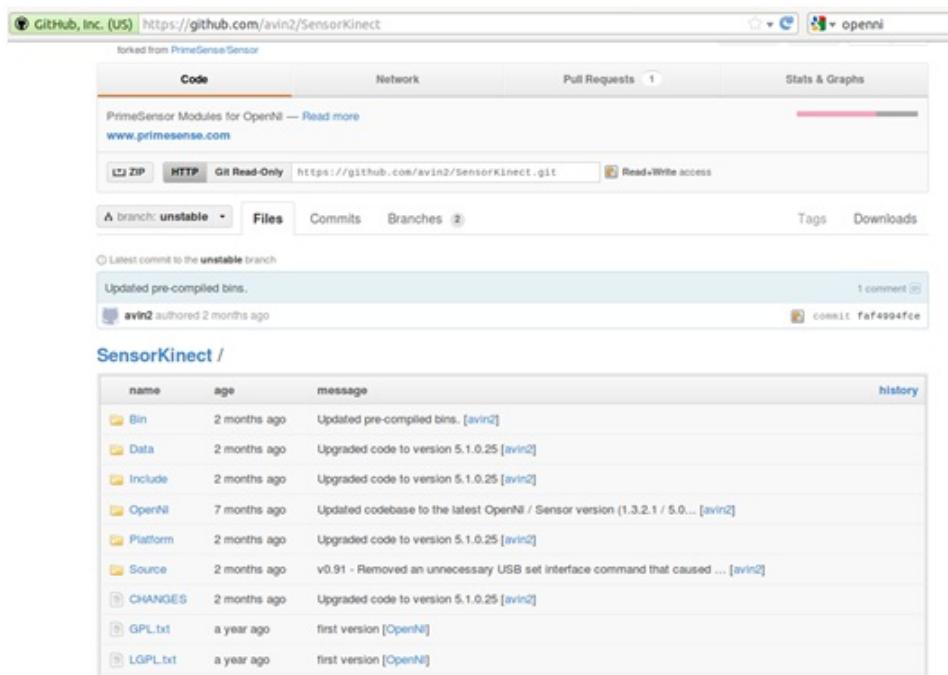


Figura 8.7: Descarga Sensor Kinect.

- Cambiar el nombre del folder a *SensorKinect* (Figura 8.8).



Figura 8.8: Sensor Kinect.

Paso 4 Ir a la carpeta OpenNI, Sensor, Nite y ejecutar sudo ./install.sh (Figura 8.9).

```
$ cd
$ cd Kinect
$ cd OpenNI
$ sudo ./Install.sh
```

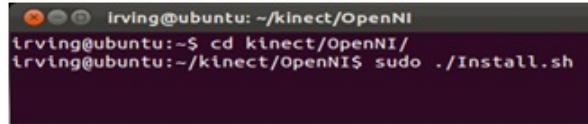


Figura 8.9: Install.sh OpenNI.

```
$ cd .. / Sensor
$ sudo . / install.sh
```

(Figura 8.10)



Figura 8.10: Install.sh Sensor.

- Antes de ejecutar *install.sh de Nite*, cambiar en los archivos *.xml* la linea de la licencia y escribir la licencia que se indica “0KOIk2JeIBYClPWVnMoRKn5cdY4 =”, para ello entrar en la carpeta de *kinect*, despues la de *Nite* y posteriormente a la de *Data* donde se encuentran los archivos, como se muestra en las figuras (Figura 8.11, Figura 8.12 y Figura 8.13).



Figura 8.11: Capeta Data.

```

1 <OpenNI>
2     <Licenses>
3         <license vendor="PrimeSense" key="insert key here"/>
4     </Licenses>
5     <Log writeToConsole="true" writeToFile="false">
6         <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error
7 (default) -->         <LogLevel value="3"/>
8     <Masks>
9         <Mask name="ALL" on="false"/>
10    </Masks>
11    <Dumps>
12    </Dumps>
13 </Log>
14 <ProductionNodes>
15     <Node type="Depth">
16         <Configuration>
17             <Mirror on="true"/>
18         </Configuration>
19     </Node>
20     <Node type="Scene" />
21 </ProductionNodes>
22 </OpenNI>

```

XML ▼ Ancho de la tabulación: 8 ▼ Ln 9, Col 54 INS

Figura 8.12: Archivo xml sin licencia.

```

1 <OpenNI>
2     <Licenses>
3         <license vendor="PrimeSense"
4             key="OKOIk2JeIBYClPWVnMoRKh5cdY4="/>
5     </Licenses>
6     <Log writeToConsole="true" writeToFile="false">
7         <!-- 0 - Verbose, 1 - Info, 2 - Warning, 3 - Error
8 (default) -->         <LogLevel value="3"/>
9     <Masks>
10    <Mask name="ALL" on="false"/>
11    <Dumps>
12    </Dumps>
13 </Log>
14 <ProductionNodes>
15     <Node type="Depth">
16         <Configuration>
17             <Mirror on="true"/>
18         </Configuration>
19     </Node>
20     <Node type="Scene" />
21 </ProductionNodes>
22 </OpenNI>

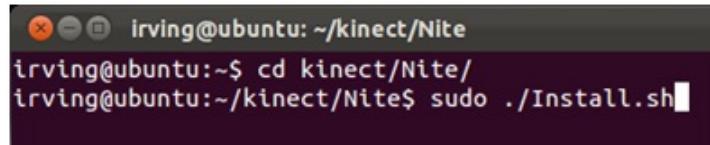
```

XML ▼ Ancho de la tabulación: 8 ▼ Ln 8, Col 24 INS

Figura 8.13: Archivo xml con licencia.

```
$ cd .. / Nite
$ sudo . / Install.sh
```

(Figura 8.14)

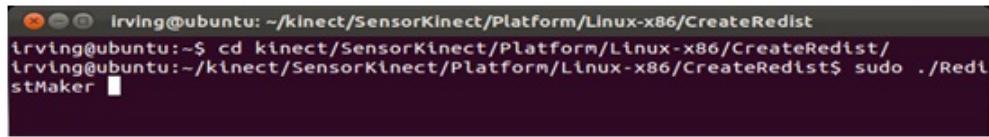


A terminal window titled "irving@ubuntu: ~/kinect/Nite". The command entered is "sudo ./Install.sh".

Figura 8.14: install.sh de Nite.

- Utilice esta licencia cuando se le preguntó durante la instalación:
"0KOIk2JeIBYClPWVnMoRKn5cdY4 ="
- Para el SensorKinect sea los siguiente desde una terminal: (Figura 8.15)

```
$ cd .. / SensorKinect/Platform/Linux-x86/CreateRedist/
$ sudo . / RedistMaker
```



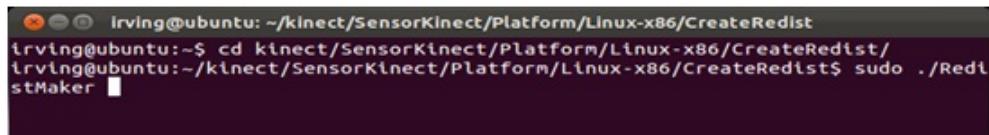
A terminal window titled "irving@ubuntu: ~/kinect/SensorKinect/Platform/Linux-x86/CreateRedist". The command entered is "sudo ./RedistMaker".

Figura 8.15: RedistMaker.

- Para poder ejecutar este comando, se deben cambiar los permisos de ese archivo, como se muestra a continuación:

```
$sudo chmod 755 RedistMaker
```

(Figura 8.16)



A terminal window titled "irving@ubuntu: ~/kinect/SensorKinect/Platform/Linux-x86/CreateRedist". The command entered is "\$sudo chmod 755 RedistMaker".

Figura 8.16: Permisos para RedistMaker.

- Despues se hará lo siguiente:

```
$cd.. /kinect/SensorKinect/Platform/Linux/Redist/Sensor-Bin-Linux-x86-v5.1.0.25
$sudo ./install.sh

sudo chmod 755 install.sh
```

(Figura 8.17)

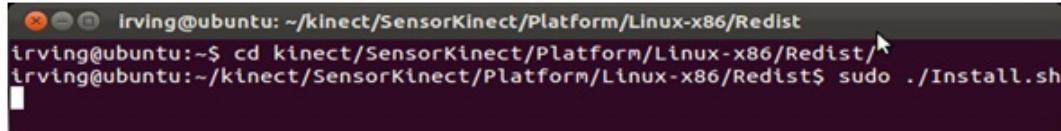


Figura 8.17: Permisos y ejecución del archivo install.sh.

Paso 5 Comprobando si todo está funcionando bien

- Conecte que *Kinect* y ejecutar los ejemplos de la carpeta *OpenNI*

```
$ cd ~ /Kinect/OpenNI/Samples/bin/Release/
```

- Suponiendo que la carpeta de *Kinect* se encuentra en la carpeta de inicio, de lo contrario ir a la carpeta respectiva (Figura 8.18).

```
$. /NiViewer
```

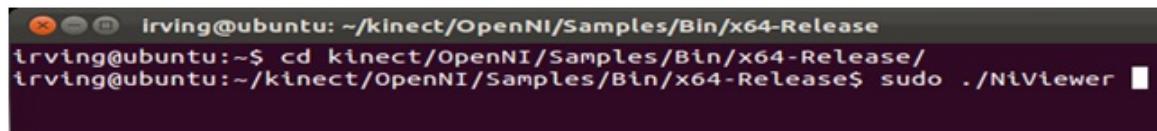


Figura 8.18: Ejecutar NiViewer.

Usted debe obtener el mapa de profundidad y la secuencia de imágenes en su ventana (Figura 8.19).



Figura 8.19: Mapa de profundidad.

- Si en caso de que salga “*Open failed: Failed to set USB interface!*”, escribir el siguiente comando como se muestra en la Figura 8.20:

```
$ sudo rmmod gspca_kinect
```

- Y volver a ejecutar el ejemplo

```
irving@ubuntu:~/kinect/OpenNI/Samples/Bin/x64-Release
irving@ubuntu:~/kinect/OpenNI/Samples/Bin$ cd x64-Release/
irving@ubuntu:~/kinect/OpenNI/Samples/Bin/x64-Release$ ls
libSample-NiSampleModule.so          Sample-NiAudioSample
Log                                Sample-NiBackRecorder
NiViewer                            Sample-NiConvertXToONI
org.OpenNI.jar                      Sample-NiRead
org.OpenNI.Samples.SimpleRead        Sample-NiRecordSynthetic
org.OpenNI.Samples.SimpleRead.jar    Sample-NiSimpleCreate
org.OpenNI.Samples.SimpleViewer     Sample-NiSimpleRead
org.OpenNI.Samples.SimpleViewer.jar  Sample-NiSimpleViewer
org.OpenNI.Samples.UserTracker      Sample-NiUserTracker
org.OpenNI.Samples.UserTracker.jar   irving@ubuntu:~/kinect/OpenNI/Samples/Bin/x64-Release$ sudo ./NiViewer
Open failed: Failed to set USB interface!
Press any key to continue . . .

irving@ubuntu:~/kinect/OpenNI/Samples/Bin/x64-Release$ rmmod gspca_kinect
ERROR: Removing 'gspca_kinect': Operation not permitted
irving@ubuntu:~/kinect/OpenNI/Samples/Bin/x64-Release$ sudo V
sudo: V: command not found
irving@ubuntu:~/kinect/OpenNI/Samples/Bin/x64-Release$ sudo rmmod gspca_kinect
irving@ubuntu:~/kinect/OpenNI/Samples/Bin/x64-Release$ sudo ./NiViewer
irving@ubuntu:~/kinect/OpenNI/Samples/Bin/x64-Release$ sudo ./NiViewer
irving@ubuntu:~/kinect/OpenNI/Samples/Bin/x64-Release$
```

Figura 8.20: rmmod gspca kinect.

Después de concluir la guía anterior, se deben poder correr los ejemplos de *OpenNI* sin problema.

8.3.2. Pasos para la Instalación de OpenCV

Se recomienda que use estos paquetes(Cuadro 8.2):

Paquete	Versión
x264	x264-snapshot-20111122-2245-stable
ffmpeg	ffmpeg-0.8.12
vl4	v4l-utils-0.8.3-1
opencv	OpenCV-2.3.1

Cuadro 8.2: Paquetes de OpenCV.

- Todos los paquetes que se instalan deben ser usados recién desempaquetados, ya que si se utilizan un paquete previamente descomprimido, puede que el paquete ya haya sido alterado por otra configuración. Por lo que no se puede saber que ocurra.

Para instalar y configurar *OpenCV 2.3.1*, realice los siguientes pasos:

Paso 1 Quitar todas las versiones instaladas de *ffmpeg* y *x264* introduciendo el siguiente comando como se muestra en la Figura 8.21:

```
sudo apt-get remove ffmpeg x264 libx264-dev
```

Figura 8.21: Remover las versiones de *ffmpeg* y *x264*.

Paso 2 Obtenga todas las dependencias para *x264* y *ffmpeg* mediante la introducción de los siguientes comandos como se muestra en la Figura 8.22:

```
sudo apt-get update
```

```
sudo apt-get install build-essential checkinstall git cmake libfaac-dev  
libjack-jackd2-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev  
libsdl1.2-dev libtheora-dev libva-dev libvdpau-dev libvorbis-dev libx11-dev  
libxfixes-dev libxvidcore-dev texi2html yasm zlib1g-dev
```

Figura 8.22: Instalación de paquetes necesarios.

Paso 3 Descargar e instalar gstreamer introduciendo el siguiente comando (Figura 8.23):

```
sudo apt-get install libgstreamer0.10-0 libgstreamer0.10-dev gstreamer0.10-tools  
gstreamer0.10-plugins-base libgstreamer-plugins-base0.10-dev  
gstreamer0.10-plugins-good gstreamer0.10-plugins-ugly gstreamer0.10-plugins-bad  
gstreamer0.10-ffmpeg
```

Figura 8.23: Instalación de gstreamer.

Paso 4 Descargue e instale x264.

- Descargar una instantánea reciente estable de x264 de <ftp://ftp.videolan.org/pub/videolan/x264/snapshots/>
La versión exacta no parece importar, he utilizado la versión x264-snapshot-20110808-2245-stable.tar.bz2.

- Configurar y compilar las bibliotecas x264 introduciendo los siguientes comandos (Figura 8.24 y Figura 8.25):

```
./configure --enable-static
make
sudo make install
```

```
irving@ubuntu:~/kinopencv/x264-snapshot-20110808-2245-stable$ ./configure --enable-static
bash: ./: Es un directorio
irving@ubuntu:~/kinopencv/x264-snapshot-20110808-2245-stable$ make
make: No se hace nada para «default».
irving@ubuntu:~/kinopencv/x264-snapshot-20110808-2245-stable$ sudo make install
[sudo] password for irving:
```

Figura 8.24: Configuración de x264.

```
ar rc libx264.a common/mc.o common/predict.o common/pixel.o common/macroblock.o
common/frame.o common/dct.o common/cpu.o common/cabac.o common/common.o common/o
sdep.o common/rectangle.o common/set.o common/quant.o common/deblock.o common/vl
c.o common/mvpred.o common/bitstream.o encoder/analyse.o encoder/me.o encoder/r
atecontrol.o encoder/set.o encoder/macroblock.o encoder/cabac.o encoder/cavlc.o e
ncoder/encoder.o encoder/lookahead.o common/threadpool.o common/x86/mc-c.o commo
n/x86/predict-c.o common/x86/const-a.o common/x86/cabac-a.o common/x86/dct-a.o c
ommon/x86/deblock-a.o common/x86/mc-a.o common/x86/mc-a2.o common/x86/pixel-a.o
common/x86/predict-a.o common/x86/quant-a.o common/x86/cpu-a.o common/x86/dct-64
.o common/x86/bitstream-a.o common/x86/sad-a.o
ranlib libx264.a
gcc -o x264 x264.o input/input.o input/timecode.o input/raw.o input/y4m.o output
/raw.o output/matroska.o output/matroska_ebml.o output/flv.o output/flv_bytestre
am.o filters/filters.o filters/video/video.o filters/video/source.o filters/vide
o/internal.o filters/video/resize.o filters/video/cache.o filters/video/fix_vfr_
pts.o filters/video/select_every.o filters/video/crop.o filters/video/depth.o in
put/thread.o input/lavf.o libx264.a -L. -pthread -L/usr/local/lib -lavformat -l
avcodec -ldl -lx11 -lxext -lxfixes -lva -ljack -lasound -lSDL -lxvidcore -lx264
-lvorbisenc -lvorbis -ltheoraenc -ltheoradec -logg -lopencore-amrwb -lopencore-a
mrnb -lmp3lame -lfaac -lz -lwscale -lavutil -lm -L/usr/local/lib -lwscale -
lavutil -lm -lpthread
irving@ubuntu:~/kinopencv/x264-snapshot-20110808-2245-stable$ sudo make install
install -d /usr/local/bin
install x264 /usr/local/bin
```

Figura 8.25: Compilación de x264.

Paso 5 Descargue e instale ffmpeg.

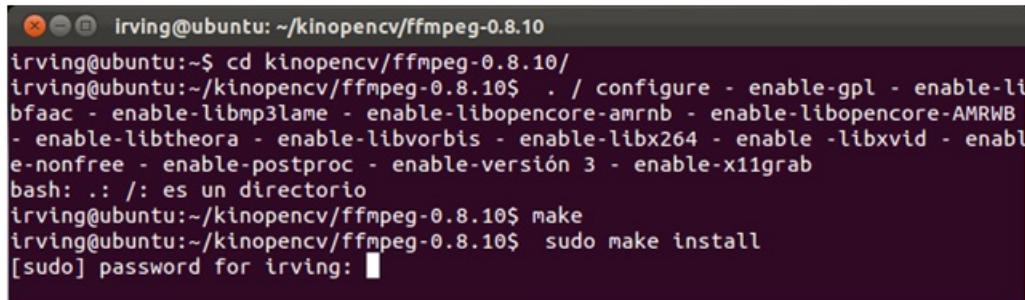
- Descargar la versión 0.8.x de <http://ffmpeg.org/download.html>

Las versiones de OpenCV anteriores a 2.3.1 requieren ffmpeg 0.7.x.

- Configurar y compilar ffmpeg mediante la introducción de los siguientes comandos en una terminal (Figura 8.26 y Figura 8.27):

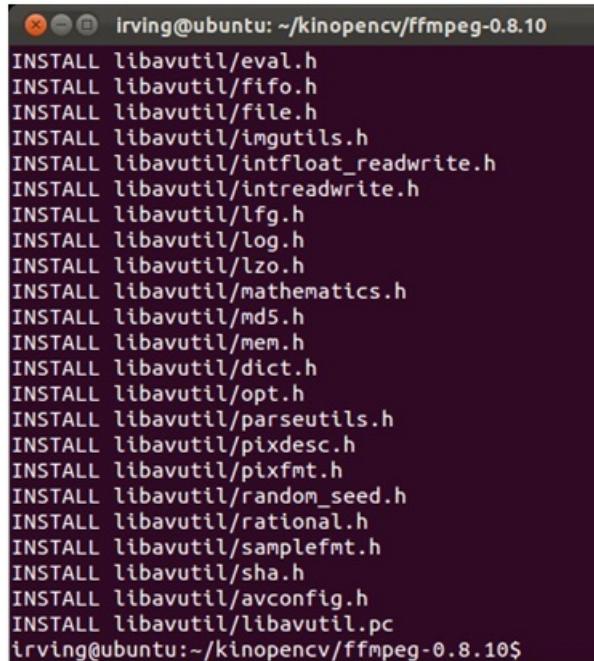
```
./configure --enable-gpl --enable-libfaac --enable-libmp3lame
--enable-libopencore-amrnb --enable-libopencore-amrwb --enable-libtheora
--enable-libvorbis --enable-libx264 --enable-libxvid --enable-nonfree
--enable-postproc --enable-version3 --enable-x11grab
make
```

```
sudo make install
```



```
irving@ubuntu:~/kinopencv/ffmpeg-0.8.10$ ./configure --enable-gpl --enable-libfaac --enable-libmp3lame
--enable-libopencore-amrnb --enable-libopencore-amrwb --enable-libtheora
--enable-libvorbis --enable-libx264 --enable-libxvid --enable-nonfree
--enable-postproc --enable-version3 --enable-x11grab
make
```

Figura 8.26: Configuración de ffmpeg.



```
irving@ubuntu:~/kinopencv/ffmpeg-0.8.10$ make
[...]
[ 1%] Generating libavutil/eval.h
[ 2%] Generating libavutil/fifo.h
[ 3%] Generating libavutil/file.h
[ 4%] Generating libavutil/imgutils.h
[ 5%] Generating libavutil/intfloat_readwrite.h
[ 6%] Generating libavutil/intreadwrite.h
[ 7%] Generating libavutil/lfg.h
[ 8%] Generating libavutil/log.h
[ 9%] Generating libavutil/lzo.h
[10%] Generating libavutil/mathematics.h
[11%] Generating libavutil/md5.h
[12%] Generating libavutil/mem.h
[13%] Generating libavutil/dict.h
[14%] Generating libavutil/opt.h
[15%] Generating libavutil/parsseutils.h
[16%] Generating libavutil/pixdesc.h
[17%] Generating libavutil/pixfmt.h
[18%] Generating libavutil/random_seed.h
[19%] Generating libavutil/rational.h
[20%] Generating libavutil/samplefmt.h
[21%] Generating libavutil/sha.h
[22%] Generating libavutil/avconfig.h
[23%] Generating libavutil/libavutil.pc
[24%]
```

Figura 8.27: Compilación de ffmpeg.

Paso 6 Descargar e instalar gtk introduciendo el siguiente comando (Figura 8.28):

```
sudo apt-get install libgtk2.0-0 libgtk2.0-dev
```

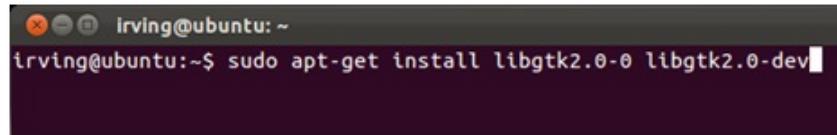


Figura 8.28: Instalación de gtk.

Paso 7 Descargue e instale *libjpeg* introduciendo el siguiente comando:

```
sudo apt-get install libjpeg62 libjpeg62-dev
```

Paso 8 Descargar *QT*, se puede hacer desde el centro de *software de Ubuntu*, se debe de instalar todos los paquetes que aparecen con una paloma, como se muestra en la Figura 8.29.



Figura 8.29: Paquetes de QT.

Paso 9 Desde el centro de *software de Ubuntu*, se instalara el paquete *vl4 panel*, como se muestra en la Figura 8.30.



Figura 8.30: Paquete de vl4 panel.

Paso 10 Para la instalación de los paquetes referentes a v4l que se muestran en la Figura 8.31, utilizaremos *synaptic*, puede buscar el ícono de la aplicación ó desde una terminal escribir el siguiente comando:

```
sudo synaptic
```

E	Paquete	Versión instalada	Última versión	Descripción
<input checked="" type="checkbox"/>	dv4l	1.0-3	1.0-3	Redirect V4L API to access a camcorder from a V4L program
<input type="checkbox"/>	kradio4		4.0.2-2	comfortable radio application for KDE
<input checked="" type="checkbox"/>	v4l-utils	0.8.3-1	0.8.3-1	Collection of command line video4linux utilities
<input type="checkbox"/>	libvideo-capture-v4l-perl		0.902-2ubuntu1	Perl interface to the Video4linux framegrabber interface
<input type="checkbox"/>	arista		0.9.6~repack-3	codificador multimedia para el escritorio GNOME
<input checked="" type="checkbox"/>	v4l2ucp	2.0.2-4	2.0.2-4	Vídeo para el panel de control universal de Linux 2
<input type="checkbox"/>	xserver-xorg-video-v4l		1:0.2.0-5build2	X.Org X server – Video 4 Linux display driver
<input checked="" type="checkbox"/>	v4l-conf	3.95.dfsg.1-8.1ubuntu1	3.95.dfsg.1-8.1ubuntu1	tool to configure video4linux drivers
<input type="checkbox"/>	libpt2.4.5		2.4.5-1	Portable Tools Library
<input type="checkbox"/>	libpt2.6.7		2.6.7-1ubuntu1	Portable Tools Library
<input type="checkbox"/>	libvideo-ivtv-perl		0.13-6	Perl extension for using V4l2 in the ivtv perl scripts
<input type="checkbox"/>	libpt-1.10.10-plugins-v4l		1.10.10-3ubuntu1	Portable Windows Library Video Plugin for Video4Linux
<input type="checkbox"/>	vgrabbj		0.9.6-3.2	grabs a image from a camera and puts it in jpg/png format
<input type="checkbox"/>	xfce4-radio-plugin		0.4.4-0ubuntu3	v4l radio control plugin for the Xfce4 panel
<input type="checkbox"/>	motion		3.2.12-2	V4L capture program supporting motion detection
<input checked="" type="checkbox"/>	libv4l-0	0.8.3-1	0.8.3-1	Collection of video4linux support libraries
<input checked="" type="checkbox"/>	libv4l-dev	0.8.3-1	0.8.3-1	Collection of video4linux support libraries (development files)
<input checked="" type="checkbox"/>	v4l2loopback-source	0.3.1-1	0.3.1-1	source for the v4l2loopback driver
<input type="checkbox"/>	qv4l2	0.8.3-1	0.8.3-1	Graphical Qt v4l2 control panel
<input type="checkbox"/>	fswebcam		20100622-1	Tiny and flexible webcam program

Figura 8.31: Paquetes de v4l.

Paso 11 Desde *synaptic*, buscar e instalar los siguientes paquetes:

```
libavformat-dev
libjasper-dev
libjasper-runtime
```

Paso 12 Descargar e instalar *OpenCV*

- Descargar la version 2.3.1 de *OpenCV* desde el siguiente enlace:
<http://sourceforge.net/projects/opencvlibrary/files/>
- Pasamos el paquete a la carpeta de home, y a partir de una terminal usamos los siguientes comandos:

```
tar xvf OpenCV-2.3.1.tar.bz2
cd OpenCV-2.3.1/
mkdir OpenCV_Bin
cd OpenCV_Bin
```

Paso 13 Instalar cmake-gui desde synaptic como se ve en la Figura 8.32:

E	Paquete	Versión instalada	Última versión	Descripción
■	cmake-qt-gui	2.8.3-3ubuntu7	2.8.3-3ubuntu7	Interfaz de usuario basada en qt4 para CMake (cmake-gui)
■	cmake-curses-gui	2.8.3-3ubuntu7	2.8.3-3ubuntu7	interfaz de usuario basada en curses para CMake (ccmake)

Figura 8.32: cmake-gui.

Paso 14 Se entra a cmake-gui como administrador, para ello nos encontramos en el directorio de *OpenCV_Bin*

En las siguientes figuras (Figura 8.33, Figura 8.34 y Figura 8.15) se muestra como debe estar configurado cmake:

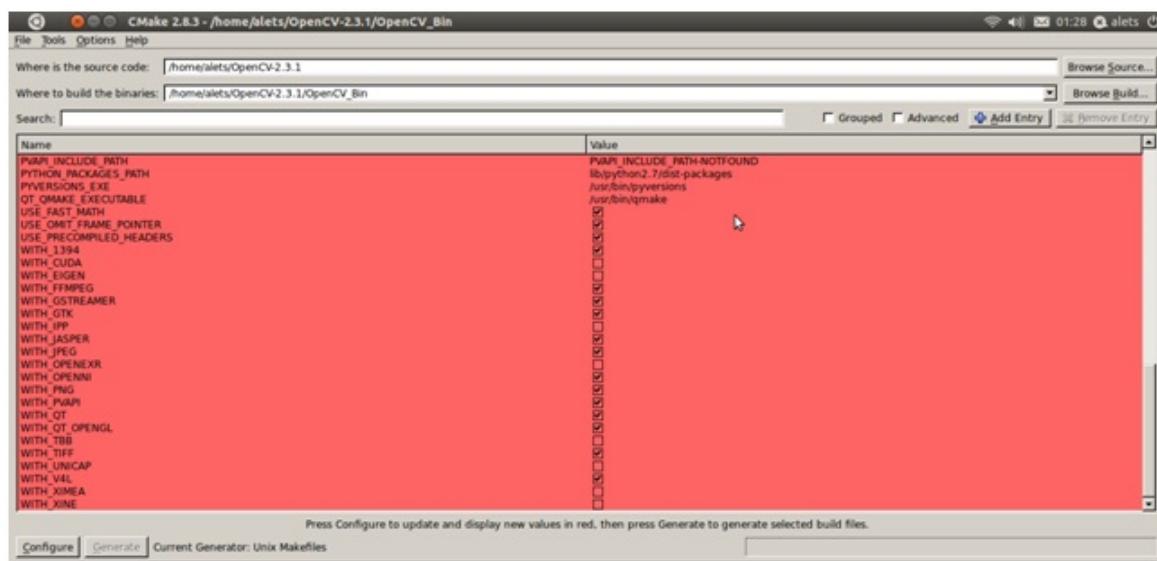


Figura 8.33: Parte 1 Configuración de cmake.

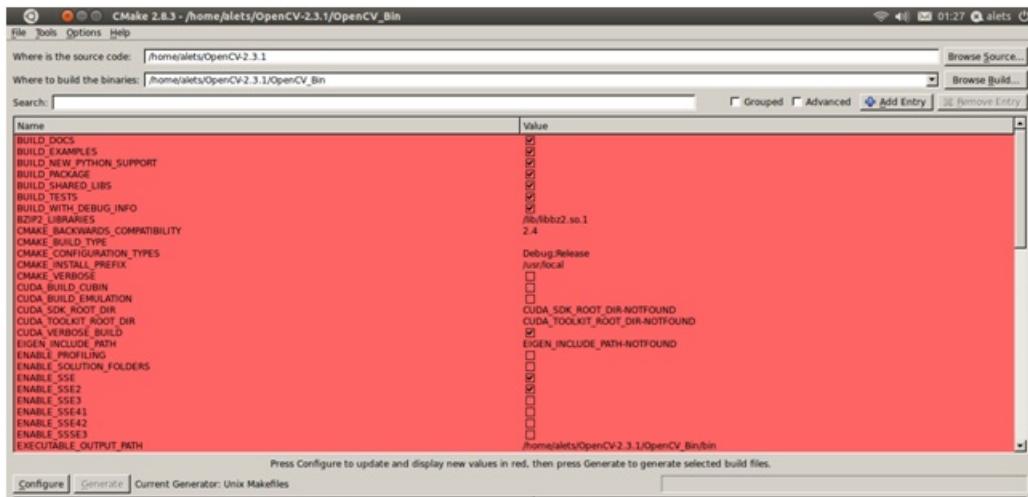


Figura 8.34: Parte 2 Configuración de cmake.

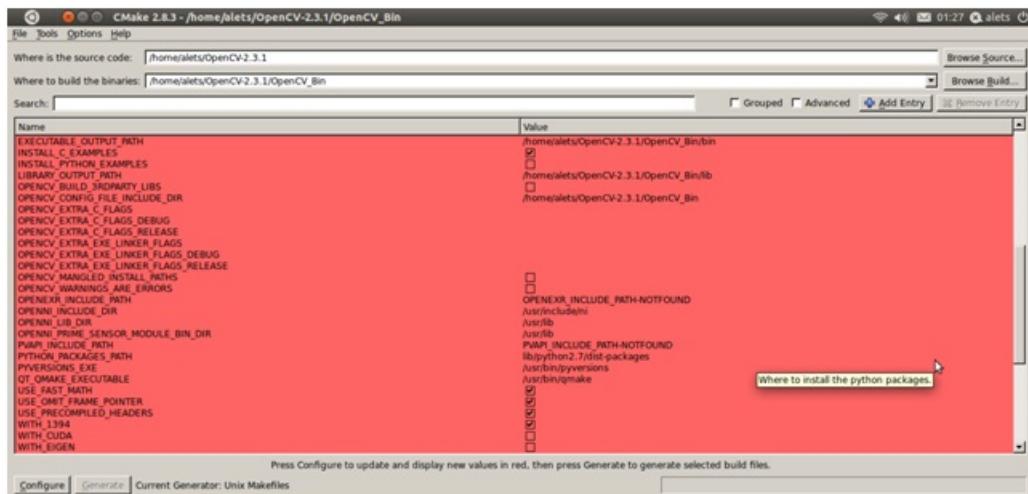


Figura 8.35: Parte 3 Configuración de cmake.

- Una vez que se tiene *cmake*, como se muestra en las imágenes de arriba se le da al botón de configurar y después al de generar.

Tal vez algunas casillas estén apagadas, después de que lo genere podrá marcarlas vuelva a repetir el paso de configurar y al de generar, para que aparezcan las nuevas opciones de *Openni*

Paso 15 Despues de que se genero cerramos *cmake*, proseguimos a escribir los siguientes comandos en una terminal desde la ruta ..//OpenCV-2.3.1/OpenCV_Bin:

```
sudo make
sudo make install
```

Si no se omitió algún paso, no tendrá ningún problema.

Paso 16 Crear un archivo, desde una terminal se escribirá los siguientes comandos:

```
sudo gedit /etc/ld.so.conf.d/opencv.conf
```

- En el documento que apareció, escribir la siguiente línea, como se muestra en la Figura 8.36.

```
/usr/local/lib
```

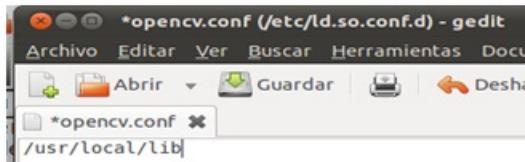


Figura 8.36: Crear archivo.

- Se guarda y se cierra el documento

Paso 17 Configurar las librerías, para ello desde una terminal debemos estar en la siguiente ruta: ..//OpenCV-2.3.1/OpenCV.Bin y usar la siguiente linea de comando(Figura 8.37):

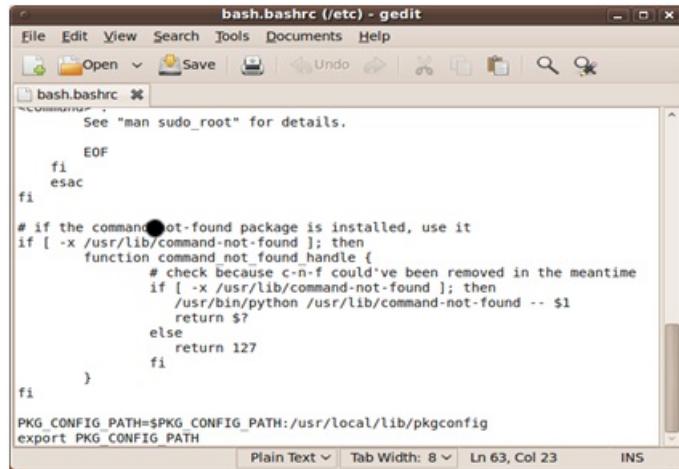
```
sudo ldconfig
```

- Por último, agregamos dos líneas a el documento:

```
sudo gedit /etc/bash.bashrc
```

- Hasta el final del documento ponemos las líneas

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig  
export PKG_CONFIG_PATH
```



The screenshot shows a window titled "bash.bashrc (/etc) - gedit". The file content is a shell script. At the bottom of the script, two new lines are being added:

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
export PKG_CONFIG_PATH
```

The status bar at the bottom right indicates "Plain Text" and "Ln 63, Col 23".

Figura 8.37: Agregar líneas al archivo.

Guardamos, cerramos, y reiniciamos nuestro computador, ahora estamos listos para utilizar y compilar programas con opencv y openni. Para compilar usamos la línea

```
g++ ‘pkg-config opencv --cflags’ my_code.cpp -o my_code ‘pkg-config opencv --libs’
```