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 *Kinect*™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. Kinect™

Kinect™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 *Kinect*™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 *RGB* que trae información de la profundidad de la escena y es similar a un mapa de disparidad estéreo.
Motor de inclinación	Ajustar hacía donde están dirigidas las cámaras.
Salida de adaptador *USB* 2.0	Conectar un cable para conectar vía *USB* el dispositivo.
Cámara *RGB*	Reconocer los tres colores básicos. Rojo, verde y azul.

Cuadro 1.1: Elementos Principales de *Kinect*™[2].

Además cuentan con:

- *Triple Core PowerPC* 970*,* 3.2*GHz, 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 diseño 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 agrupa 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 cuáles 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 *Kinect*$^{\text{TM}}$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 tenia 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 *ARToolKit 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 grafica 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].

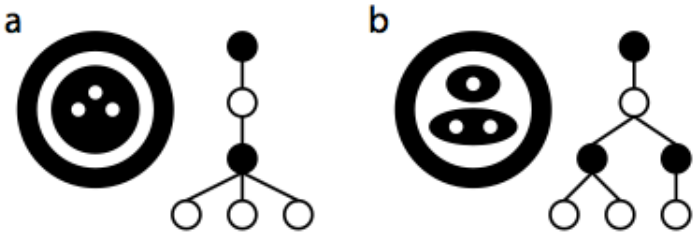

Figura 1.2: Algunas simples topologías y su correspondiente gráfico de región adyacente.

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 hacía 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 esta 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 platilla 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 proporcioná:

- 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 interactué directamente con un dispositivo electrónico, junto con esto querer desarrollar algún sistema que probara que se podía utilizar *Kinect*™junto con la computadora personal.

1.7.2. General

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

1.7.3. Particulares

- Lograr una configuración para usar *Kinect*™con la computadora personal, para desarrollar una aplicación.
- Eliminar uso del *mouse* en nuestro sistema.
- Eliminar uso del teclado en nuestro sistema.
- Reconocer una imagen binaria la cual puede ser rotada para diferentes acciones.
- Implementar una interfaz sin involucrar dispositivos electronicos como una pantalla tactil.

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*™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*™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*™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*™, *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*™, 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. Debído a que era una tecnología reciente.[ex]istía 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 interes con el cual se van a hacer los trazos además de seleccionar el color y agregar otra

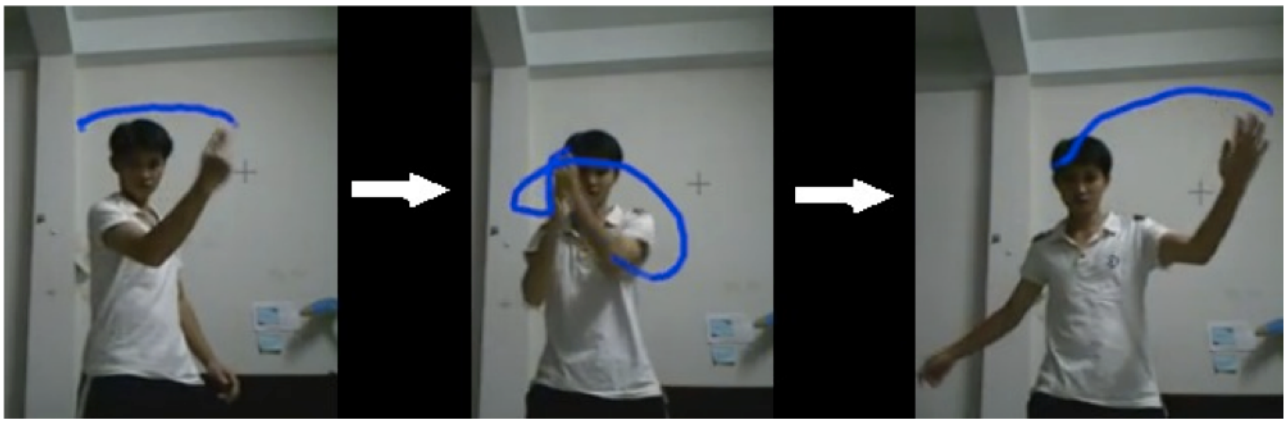

Figura 1.5: Hand Tracking.

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].

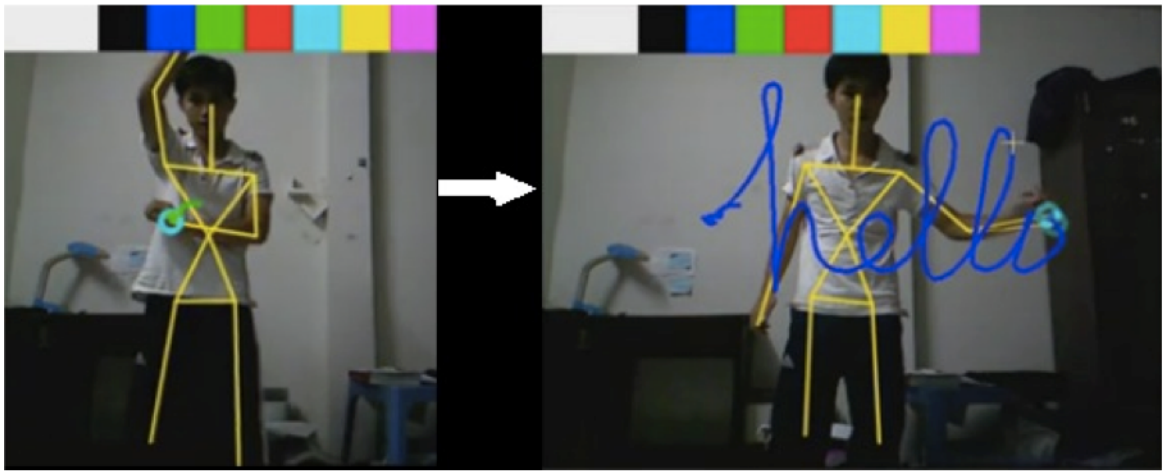

Figura 1.6: Reconocimiento del cuerpo.

Kinect Active Projection Mapping

Es una aplicación que trabaja con *Kinect*^{TM}*, OpenCV y OpenNI*, además comparte la idea de mantener un área de trabajo y una proyeción, 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 detras de el usuario[16].

Esta aplicación ha sido desarrollada en el *Computer Fusion Laboratory* como parte del programa de ingenieria 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 imágen sobre un área de trabajo.

Aldebaran Nao Kinect Controller

Es un proyecto donde se controla por medio de *Kinect*™a 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.

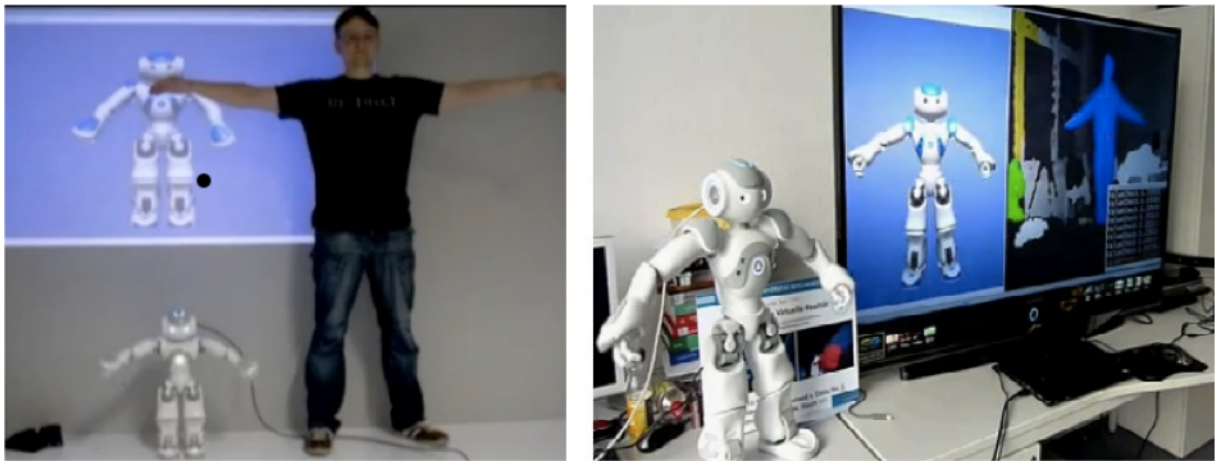

Figura 1.8: Interacción con robot automáta programable.

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 realizo 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, Pn 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*™, 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 Kinect™

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

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*™.

Problemas: No poder reconocer el dispositivo.

Detección de Imagen

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

Descripción: Se capturarán los datos de profundidad de la escena por medio de la camara 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*™se captura como se desplaza el punto de interés siguiendolo 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*™, 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 *Kinect*™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 selcciona 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 *Kinect*$^{\text{TM}}$, 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 *Kinect*$^{\text{TM}}$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 *Kinect*$^{\text{TM}}$, 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 (Tablas 2.1, 2.2 y 2.3) que se utilizaron para la realización del sistema, asi como las caracteristicas del kinect(Tabla 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 módelo 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 módelo dv4.

- Dispositivo *Kinect*™

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 adapatador 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 utlizarón se mencionan en la Tabla 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 hizó una tabla comparativa(Tabla 2.6) de 3 sistemas diferrentes, *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, asi 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 (Tabla 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 desarrollo 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.

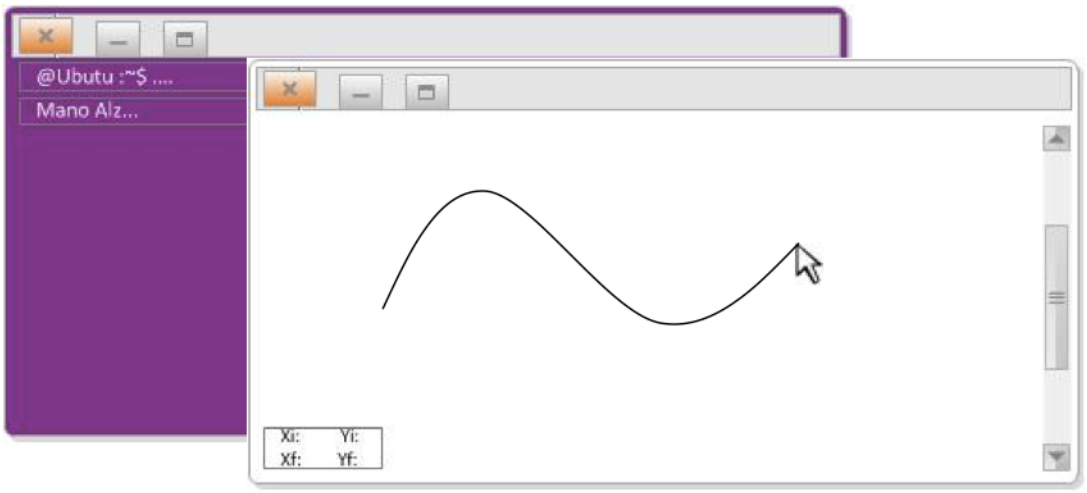

Figura 2.1: Módulo 1 - Dibujar a Mano Alzada.

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.

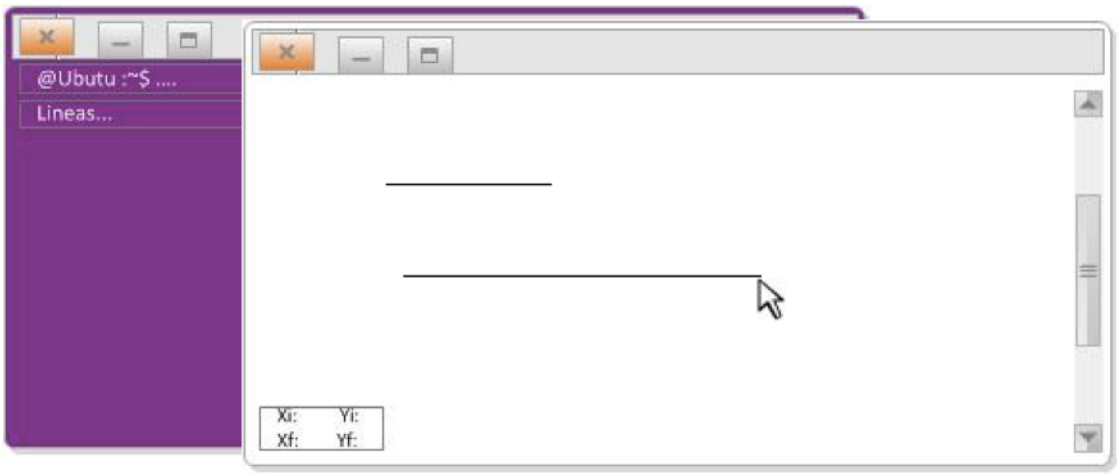

Figura 2.2: Módulo 1 - Líneas de Diferente Longitud.

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.

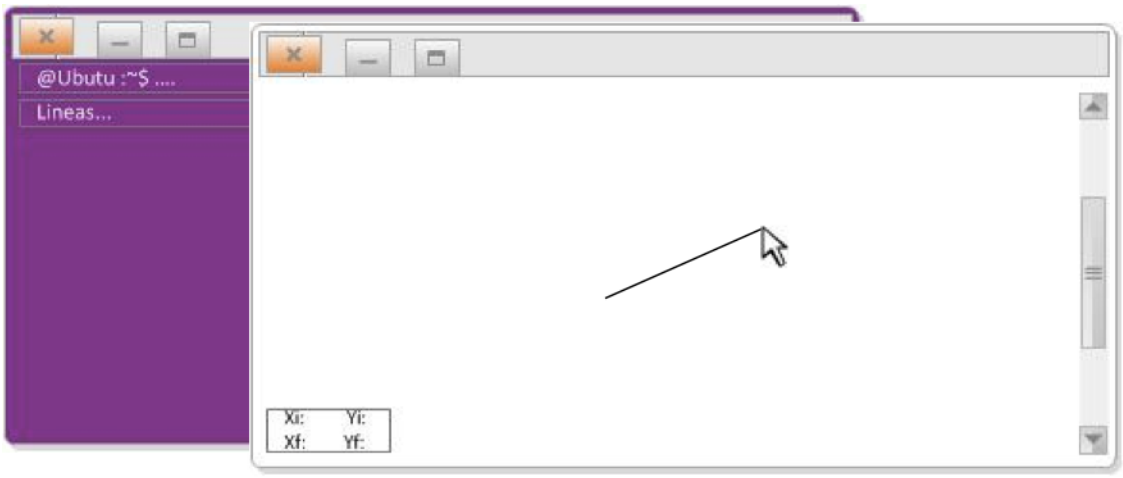

Figura 2.3: Módulo 1 - Líneas en Diferentes Direcciones.

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.

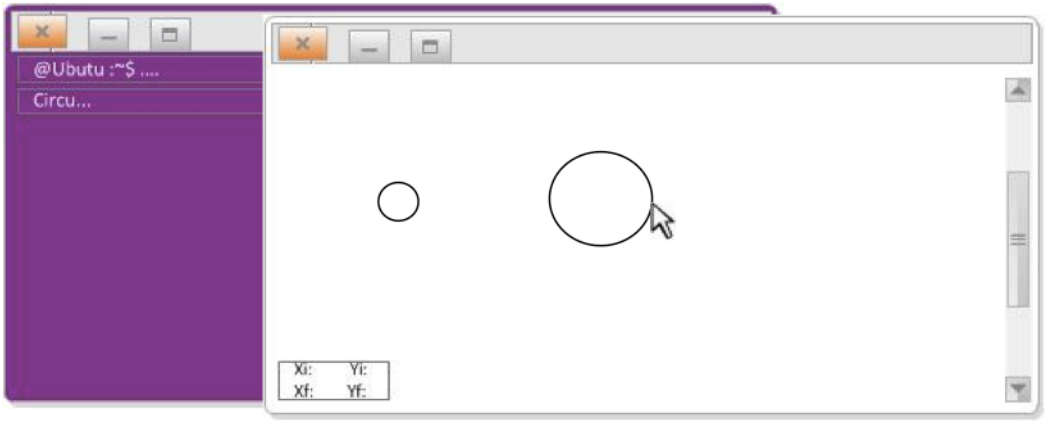

Figura 2.4: Módulo 1 - Circunferencias de Distintos Radios.

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.

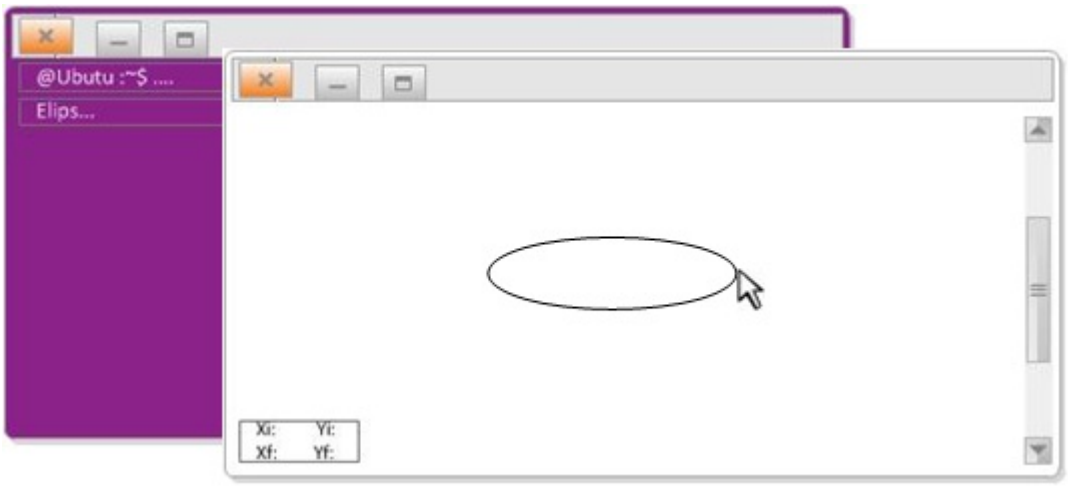

Figura 2.5: Módulo 1 - Elipse de distinta dimensión.

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.

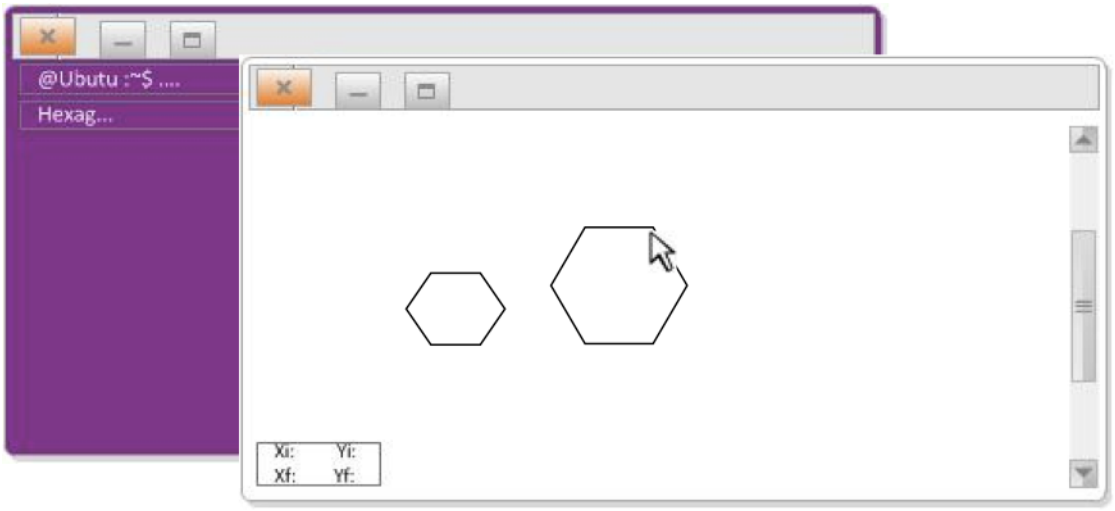

Figura 2.6: Módulo 1 - Poligonos de Diferentes Tamaños.

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.

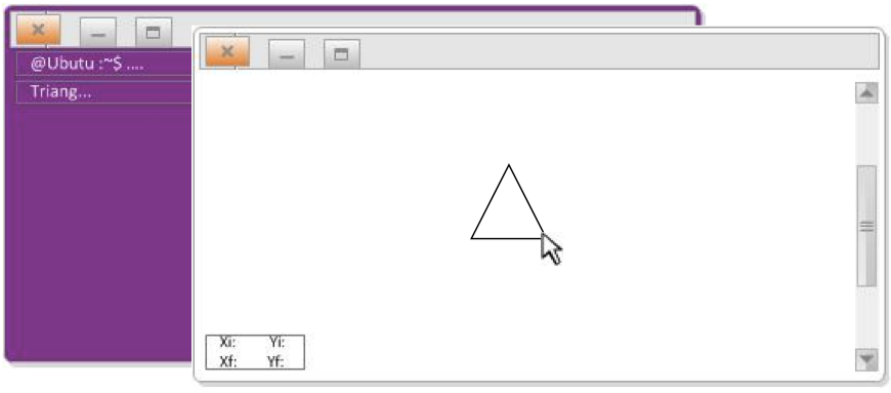

Figura 2.7: Módulo 1 - Selección del tipo de Polígono.

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

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

Descripción: La computadora reconocerá la identificación del *Kinect*™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 *Kinect*™.

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 *Kinect*™.

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

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

Datos de salida: Reconocimiento del dedo índice.

Pre-condiciones: Instalación del *API, Kinect*™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 *Kinect*^{TM}.

Descripción: Detectar el desplazamiento que hará el dedo índice mediante el *Kinect*^{TM}la imagen, siendo más especifico en este Módulo será el dedo índice que capturará el *Kinect*^{TM}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, Kinect*^{TM}conectado.

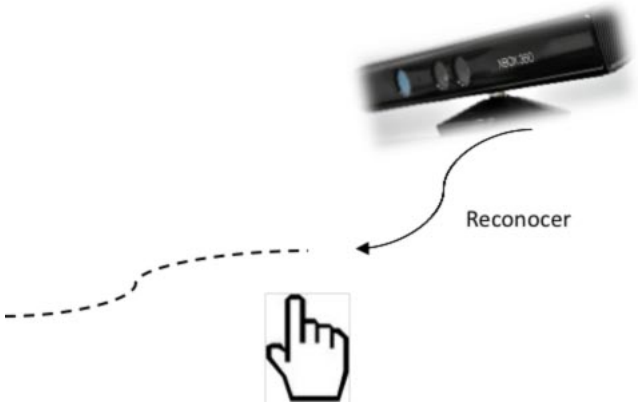

Figura 2.10: Módulo 2 - Reconocer Desplazamiento del dedo índice.

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, Kinect*™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 *Kinect*™, como se muestra en la Figura 2.12.

Datos de entrada: posición inicial.

Datos de salida: Trazo realizado.

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

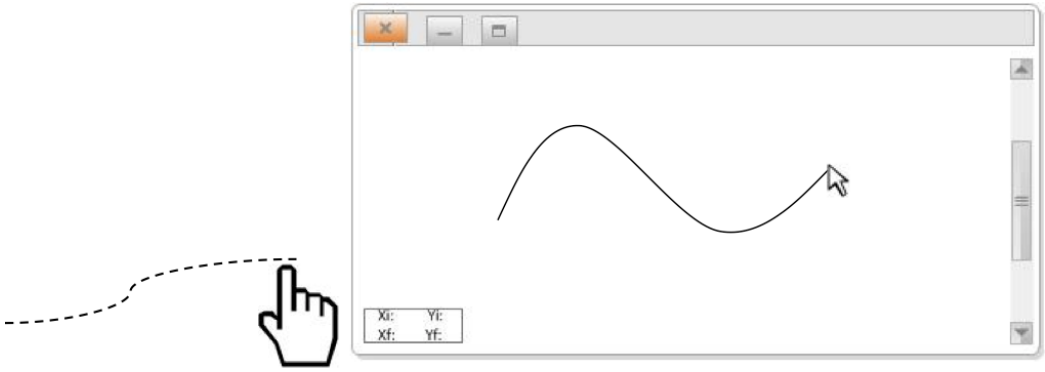

Figura 2.12: Módulo 2 - Proyección en el editor básico de dibujo.

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.

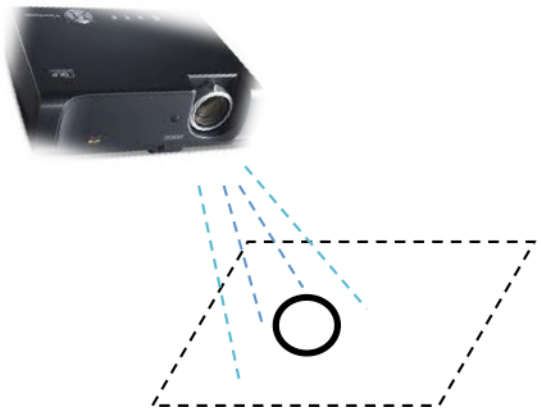

Figura 2.14: Módulo 3 - Proyección sobre el área de trabajo.

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 sobra 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, Kinect*™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 *Kinect*™reconozca la herramienta (*tag*).

Descripción: Poder hacer que se reconozca la herramienta mediante el *Kinect*™, 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*™conectado, 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 ($\widehat{\text{tag}}$) 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.

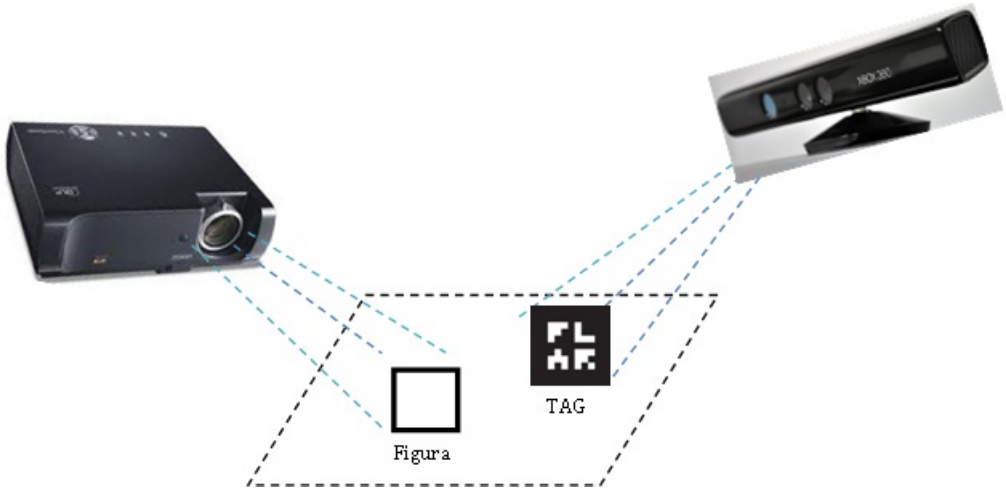

Figura 2.17: Módulo 4 - Dibujar utilizando la Herramienta (tag).

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*^{TM}conectado, imagen binaria.

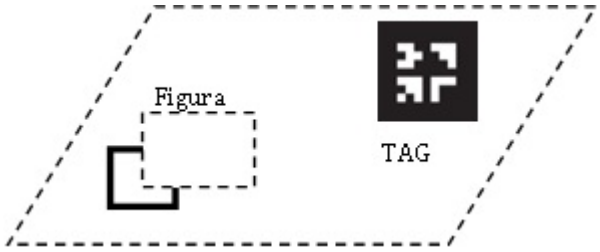

Figura 2.18: Módulo 4 - Escalar Figura.

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*™conectado, imagen binaria.

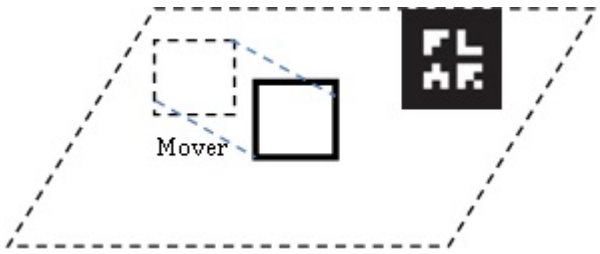

Figura 2.19: Módulo 4 - Mover Figura.

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*™conectado, imagen binaria.

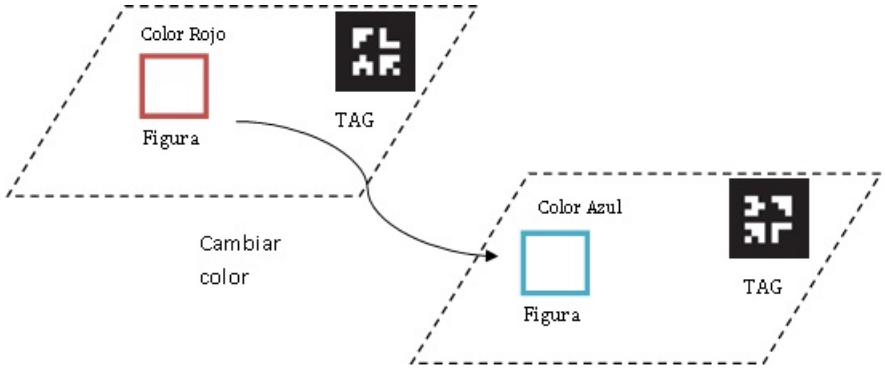

Figura 2.20: Módulo 4 - Cambiar de color.

Capítulo 3

Diseño

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.

Capítulo 5

Pruebas

Se presentan las pruebas realizadas al sistema con el objetivo de verificar su funcionamiento. También se muestras los resultados de dichas pruebas.

Capítulo 6

Conclusiones

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

6.1. Generales

6.2. Individuales

6.3. Trabajo a futuro

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 ^{TM}. 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. Acaemic Press 2° 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. reacTIVison, 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/

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 distrubución 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 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
```