Universidad Autónoma de Querétaro
Facultad de Ingeniería
Campus San Juan del Río

Segmentación de imágenes obtenidas a través de un sensor Kinect con criterios morfológicos y atributos visuales de profundidad

Presenta

Marco Antonio Garduño Ramón

No. de expediente: 154526

Asesor: Dr. Luis Alberto Morales Hernández

San Juan del Río, Qro., Mayo de 2014.

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias (Mecatrónica)

Segmentación de imágenes obtenidas a través de un sensor Kinect con criterios morfológicos y atributos visuales de profundidad

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Ciencias (Mecatrónica)

Presenta:
Marco Antonio Garduño Ramón

Dirigido por:
Dr. Luis Alberto Morales Hernández

SINODALES

Dr. Luis Alberto Morales Hernández Presidente	Firma
Dr. Roque Alfredo Osornio Rios Secretario	Firma
Dr. Juan Primo Benítez Rangel Vocal	Firma
Dr. Jesus Rooney Rivera Guillén Suplente	Firma
Dr. Iván Ramón Terol Villalobos Suplente	Firma

Dr. Aurelio Domínguez González
Director de la Facultad

Dr. Irineo Torres Pacheco
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Mayo de 2014
México

Resumen

El procesamiento digital de imágenes ha pasado de ser una herramienta auxiliar a una vital en múltiples procesos industriales. El desarrollo tecnológico ha permitido la creación de equipos de visión que permiten medir la temperatura de los sistemas, otros que pueden capturar imágenes a altas velocidades, y finalmente aquellos que pueden cuantificar la profundidad a la que se encuentran los diferentes objetos de una escena, entre otros. Estos últimos utilizan una tecnología basada en emisores y sensores de luz infrarroja, donde dicha luz reflejada en los objetos permite realizar una estimación de la profundidad en milímetros a la que se encuentran. Uno de los problemas inherentes a estos equipos es la alta susceptibilidad al ruido que tienen, lo que se traduce en perdida de información. El sensor Kinect es un ejemplo de este tipo de dispositivos y ha venido a significar una revolución por las características que posee y sobre todo por su bajo costo. En este trabajo de tesis se presenta una metodología para resolver el problema de ruido y huecos presente en la información de profundidad que entrega el Kinect mediante filtros morfológicos y estadísticos, al mismo tiempo se lleva a cabo segmentación de imágenes por distancias en imagen de profundidad y de color comparándolo contra procesamientos de segmentación tradicionales analizando sus ventajas y desventajas. Se utilizan atributos visuales para representar la información de profundidad que entrega el sensor y se aplica una técnica de geometría proyectiva para empatar la información de profundidad y de color que entrega el sensor, todo esto con el fin de acercar las funcionalidades de este dispositivo de bajo costo a aquellas que ofrecen los equipos de nivel industrial. Como caso particular de estudio se utilizó el sensor Kinect y su función de monitoreo de articulaciones de usuario como una herramienta auxiliar de la gente de fisioterapia de la Universidad en la realización de análisis posturales.

(**Palabras clave:** Kinect, Morfología Matemática, Filtrado Morfológico, Interfaz Gráfica de Usuario.)

Abstract

The digital image processing has grown from an assistant to a vital tool in many industrial processes. Technological development has enabled the creation of vision equipment to measure the temperature of a system, others which can capture images at high speeds, and finally those who can quantify the depth at you will find different objects in a scene, among other. This latter uses a technology based in infrarred light emitters and sensors, where this reflected light in the objects allow to make an estimation of objects depth in milimeters at which they are. One of the problems inherent to these devices is the high susceptibility to noise with which results in loss of information. The Kinect sensor is an example of this type of devices and has come to mean a revolution for the features it has and especially for its low cost. This thesis presents a methodology to solve the problem of noise and holes present in depth information delivered by the Kinect using morphological and statistical filters, simultaneously we perform image segmentation by distances in depth and color images comparing it to traditional segmentation processing analyzing their advantages and disadvantages. Visual attributes are used to represent the depth information delivered by the sensor and a technique of projective geometry is applied to tie the depth and color information that sensor delivers, all this in order to bring the same functionality to this low cost device to those that offer industrial grade equipment. As a particular case of study, Kinect sensor and its articulation user detection function was used as auxiliary tool for physiotherapy people at the University to make postural analisys.

(**Keywords:** Kinect, Mathematical Morphology, Morphological Filtering, Graphical User Interface.)

Dedicatorias

A mis padres Eugenio y Cristina (q.e.p.d)
por inculcarme el amor al estudio
e impulsarme a ser cada día mejor.
A mi hermano Carlos
por ser mi mejor amigo.

Agradecimientos

Quiero agradecer a mi padre y a mi hermano por apoyarme incondicionalmente en todos sentidos durante estos dos años.

Al Dr. Luis Alberto Morales Hernández por su asesoría brindada para la realización de este trabajo de investigación, pero sobre todo por su amistad.

Al Dr. Roque Alfredo Osornio Rios por brindarme la confianza y la oportunidad para enrolarme en este curso de posgrado, así como por sus consejos y asesorías para lograr un mejor trabajo de investigación.

Al M. en C. Jesús Iván Sánchez Gómez por su ayuda, consejos pero sobre todo por su amistad.

A mis compañeros de maestría, a mis profesores y al personal de la Universidad Autónoma de Querétaro con los que conviví durante éstos dos años.

Al Consejo Nacional de Ciencia y Tecnología por la beca otorgada para la realización de mis estudios de maestría en la Universidad Autónoma de Querétaro.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El desarrollo científico y tecnológico han permitido al ser humano mejorar sus condiciones y niveles de vida de manera significativa desde los comienzos de la historia. Múltiples áreas se han visto beneficiadas por este desarrollo siendo la Ingeniería pieza vital de estos avances. A nivel industrial podemos encontrar algunos de los avances más significativos, siendo los sistemas automatizados uno de los aportes de mayor interés debido a su complejidad. Tenemos, por ejemplo, sistemas que ensamblan autos, otros que forjan el acero y algunos que envasan alimentos, todos ellos con una gran eficiencia. Muchos de éstos cuentan con herramientas de visión artificial los cuales dotan a las máquinas con uno de los sentidos de mayor importancia del ser humano, obvio es, con sus debidas limitantes. Los sistemas de procesamiento de imágenes son aplicados para verificar diferentes aspectos, por ejemplo: dimensiones de piezas, niveles de llenado, calidad de maquinado, número de piezas producidas, entre otros. Precisamente una de las principales restricciones de estos sistemas es la dificultad para medir, cuantificar y distinguir la diferencia de profundidad entre varios objetos en una imagen, tarea que para el ser humano es relativamente sencilla. Se han desarrollado métodos y equipos que llevan a cabo esta tarea como la visión estéreo, sin embargo, su alto costo debe ser tomado en cuenta. Un país en vías de desarrollo necesita crear su propia tecnología a fin de ofrecer una opción de costo accesible para las pequeñas, medianas y micro empresas que se abren a diario en él. El presente trabajo presenta un sistema que aborda el problema de profundidad en los sistemas de visión haciendo uso del sensor Kinect, ofreciendo además una opción de costo

accesible.

1.1. Antecedentes

El procesamiento de imágenes es usado comúnmente para dos fines principales, el primero de ellos es el de mejora para facilitar su interpretación a los seres humanos y, por otra parte, su procesamiento con el fin de obtener información valiosa por medio de sistemas automáticos. A nivel industrial podemos encontrar múltiples sistemas que hacen uso del procesamiento de imágenes, siendo esta herramienta pieza vital en la línea de producción de muchos productos en los cuales permiten verificar dimensiones, calidad de maquinado, niveles de llenado, cantidades de producto, entre otros. Sus bondades y posibilidades han permitido que esta área genere una gran cantidad de tecnologías e investigaciones en múltiples áreas (Ingeniería, Robótica, Control, Computación, Medicina, Biología, entre otras), donde las posibilidades de desarrollo aún son bastante amplias dado el crecimiento exponencial que están teniendo y tendrán los sistemas computacionales. En la Universidad Autónoma de Querétaro (UAQ) se han desarrollado trabajos relacionados con procesamiento de imágenes abordando diversas aplicaciones entre ellos:

Cayetano (2010) desarrolló un sistema para la medición de la capa endurecida del acero, analizó la huella dejada por el identador del durómetro según el estándard internacional ASTM, para determinar sus bordes y área que se puede relacionar con su profundidad y así calcular su dureza, haciendo uso de procesamiento de imágenes implementado en un software propio desarrollado en Visual Studio C++ con el formato de imágenes BMP (mapa de bits, por sus siglas en inglés). **Morales et al. (2009)** desarrollaron un sistema que permite caracterizar las propiedades del aluminio mediante el conteo de poros. Para esto se analiza la imagen digital de la muestra a través de diversos tratamientos a fin de ajustar su umbralizado y obtener una imagen segmentada. En esta imagen se aprecian los poros en color negro sobre un fondo blanco, permitiendo implementar un sistema automático que mida su proporción con respecto al tamaño de la muestra. Este sistema fue implementado mediante el uso de Visual Studio C++ y usa el formato de imágenes BMP. **Solís (2008)** desarrolló un software para medir el porcentaje de grafito incrustado en una muestra metalo-

gráfica, aplicando técnicas de umbralizado y filtros morfológicos a fin de facilitar la identificación del grafito. Estas técnicas permiten diferenciar estructuras de grafito, contarlas y relacionarlas con el tamaño de la muestra. El sistema hace uso de Visual C++ y formato BMP. **Garduño (2012)** desarrolló, mediante software libre y lenguaje C, un sistema de mejora de contraste para imágenes obtenidas a través de un microscopio tomadas bajo malas condiciones de luminosidad a fin de dar uso a dichas muestras en procesos de segmentación y mediciones de microdureza. Trabajó en el espacio de color $u'v'Y'$ para aplicar procesamiento sólo al canal de luminancia. Implementó los métodos de ecualización de histograma, balance de blancos, balance de blancos con valores neutros y retinex. **Razo et al. (2011)** implementaron un sistema eficiente para la detección de rostros en imágenes mediante filtros morfológicos y segmentación aplicado a imágenes en espacio de color RGB de 24 bits de profundidad en lenguaje C++. Adicionalmente aplican procesamientos complementarios como binarización y eliminación de ruido. **Pacheco et al. (2011)** propusieron un sistema para la detección de error de posición de barrenos en piezas maquinadas y dimensiones de la misma para control de calidad. Implementó su sistema en Matlab (Matrix Laboratory) diseñando además, una interfaz de usuario, aplicó operaciones morfológicas y umbralizado obteniendo buenos resultados, sin embargo, no es posible aplicar su sistema para verificar la correcta profundidad de los barrenos. **Rico et al. (2011)** propusieron un sistema de reconstrucción 3D mediante Matlab, una cámara digital estándar y una lente telecéntrica para objetos de menos de 50 mm de diámetro. Aplica un proceso de homografía a fin de relacionar las características de una imagen con un patrón de calibración comercial tipo tablero de ajedrez. Trabajó en imágenes en escala de grises en formato JPG con resolución de 1280 x 960 pixeles. El resultado obtenido es aceptable aunque se puede mejorar la exactitud de la reconstrucción. **Mejía et al. (2012)** desarrollaron un sistema la selección automática de la herramienta de trabajo de una máquina CNC aplicando binarizado de imágenes a fin de determinar la herramienta adecuada para realizar el trabajo indicado sobre el borde detectado, la pieza a realizar cargada a través de un archivo tipo DXF. Esta tarea anteriormente se basaba meramente en la experiencia del diseñador, se observa que este sistema toma decisiones adecuadas con respecto a la herramienta a usar.

Por otro lado, un dispositivo que está cobrando auge para la realización de múltiples investigaciones y proyectos a nivel nacional e internacional dadas las ventajas que ofrece es el sensor Kinect

de Microsoft (`http://kinectforwindows.org/`), algunos trabajos publicados que hacen uso de dicho sensor se mencionan a continuación. **Cruz et al. (2012)** y **Han et al. (2013)** llevaron a cabo una revisión muy completa sobre el sensor Kinect, hacen una semblanza histórica acerca de él, mencionan sus principales características, sus ventajas y desventajas, además enumeran trabajos realizados a nivel académico que hace uso del sensor. Ambos documento sirven de base para la realización de trabajos que usen este dispositivo. Deben resaltarse los problemas de ruido y huecos que presentan las imágenes adquiridas por el equipo, así como la diferencia existente entre las imágenes RGB y de profundidad que entrega el sensor. **Chen et al. (2012)**, lograron calcular parámetros como lo son índice de área y el ángulo de distribución de la hoja del maíz haciendo uso sólo de la información de profundidad provista por el sensor Kinect. La información de profundidad forma una nube de puntos con la cual llevan a cabo un proceso de reconstrucción mediante una red de triangulación irregular. Para llevar a cabo su proceso hacen uso de Visual Studio C++ y OpenGL (Open Graphics Library). Mediante este método obtienen resultados con un error de hasta 25 porciento comparando la medición real contra la obtenida a través del sensor. El combinar su uso con la información de color provista por el mismo sensor Kinect podría mejorar la precisión de la medición e incluso hacer mas intuitivo el uso del sistema para el usuario. **Hernández et al. (2012)** propusieron la implementación de un algoritmo para la detección de objetos en ambientes cerrados usando técnicas de segmentación color en el espacio CIE-Lab y de profundidad a imágenes adquiridas a través del sensor Kinect. Aplican además un proceso de homografía para sincronizar la imagen RGB y de profundidad provistas por el sensor Kinect. La detección de objetos funciona adecuadamente, sin embargo, para aplicaciones específicas contar con información como la distancia a la que se encuentran los objetos a modo de retroalimentación podría brindar un mundo de oportunidad en robótica y control. **Peralta (2012)** desarrolló, una interfaz para interpretar el lenguaje natural del ser humano usando el sensor Kinect. Para realizar su trabajo utilizó el lenguaje C/C++ y OpenNI (Open Natural Interface) bajo sistema operativo Linux. A fin de verificar el correcto funcionamiento de su sistema desarrollado, implementó tres diferentes entornos, primero un ratón virtual, un sistema de menús y una interfaz para manipulación de objetos virtuales. Para corregir el problema de ruido presente en las imágenes entregadas por el sensor Kinect hizo uso de un filtro Kalman. **Díaz et al. (2012)** realizaron dentro de la UAQ, una comparativa entre dos

aplicaciones enfocadas al reconocimiento de patrones en video para el control virtual de un mouse, una desarrollada bajo la aplicación de software libre Processing y la otra mediante C# en el SDK (Software Development Kit) de Microsoft para Kinect. Implementaron un sistema para la detección del usuario y asignan funciones del mouse a los gestos obtenidos mediante el rastreo de sus manos. Mencionaron que debido a que el SDK de Microsoft es de uso específico para el Kinect, éste puede sacar todo el provecho de dicho hardware para desarrollar una gran cantidad de aplicaciones, aunque no es una limitante total para no hacer uso de software libre. **De León et al. (2012)**, llevaron a cabo una comparativa para determinar el potencial del sensor Kinect contra medios tradicionales para el control de un vehículo móvil. El control de la dirección del vehículo se realiza mediante un volante virtual que son las manos del usuario simulando a éste. Mediante el Kinect se detecta la posición de estas y se determina la dirección y la velocidad del vehículo. Si bien acostumbrarse a esta nueva forma de control es difícil al principio, basta un rato de pruebas para lograr resultados similares a los obtenidos contra un sistema de control físico tradicional.

Como se puede ver, dentro de la UAQ se han desarrollado una gran cantidad de trabajos dentro del área de procesamiento de imágenes, sobre todo en el área de metalografía, además, es común que los proyectos realizados cuenten con una interfaz gráfica de usuario, la mayoría de estas desarrolladas bajo la plataforma de programación Visual Studio desarrollada por Microsoft, aunque de igual forma se ha empezado a trabajar con software libre, todo esto encaminado a evitar la dependencia de software propietario de terceros, lo cual implica reducir e inclusive eliminar la necesidad de pagar costos de licencias y por capacitación. Es importante resaltar el hecho de que en la actualidad se están realizando muchos trabajos, tanto a nivel nacional como internacional, que hacen uso del sensor Kinect como sistema de adquisición de imágenes y de mapas de profundidad; con aplicaciones en áreas tan diversas como biología, medicina, robótica, computación, procesamiento de imágenes, control, etc. Particularmente en la UAQ se ha empezado a trabajar con dicho sistema, principalmente para la generación de interfaces de control de PC o de sistemas móviles, tanto con el SDK de Microsoft como con herramientas libres, por lo que aun falta camino por recorrer para explotar por completo las capacidades del dispositivo en otro tipo de aplicaciones a nivel local.

1.2. Objetivos e hipótesis

1.2.1. Objetivos generales y particulares

1.2.1.1. Objetivo general

Implementar un sistema de captura, representación y filtrado a partir de la información dada por el sensor Kinect mediante técnicas de segmentación a fin de cuantificar parámetros de profundidad.

1.2.1.2. Objetivos particulares

- Desarrollar una interfaz mediante Visual Studio para el procesamiento y análisis de las imágenes RGB y de profundidad obtenidas mediante el sensor Kinect.
- Aplicar métodos de pre-procesamiento a las imágenes de profundidad obtenidas a través del sensor Kinect mediante técnicas de segmentación para eliminación de ruido y huecos.
- Compensar mediante homografía la distancia física existente entre la cámara RGB y el sensor de profundidad a fin de mejorar la precisión referente a la información de profundidad de la imagen.
- Definir el espacio de color de trabajo más adecuado para llevar a cabo la segmentación, aplicando las matrices de transformación pertinentes e implementando métodos simples de procesamiento a las imágenes entregadas por el sensor Kinect.
- Determinar los criterios adecuados de acuerdo al funcionamiento del sistema para aplicar la segmentación por formas, color y distancia.
- Generar una librería con los métodos de segmentación propuestos a imágenes obtenidas a través del sensor Kinect en software para aplicaciones en la universidad y trabajos posteriores.

1.2.2. Hipótesis

La información de profundidad proporcionada por el sensor Kinect, en conjunto con el procesamiento mediante técnicas de segmentación morfológica, permite cuantificar y caracterizar parámetros dimensionales que en una cámara convencional no son posibles.

1.3. Justificación

El sensor Kinect de Microsoft ofrece la posibilidad de cuantificar parámetros de profundidad en una imagen a un bajo costo combinando una cámara RGB junto con un sensor de profundidad. Este dispositivo entrega una imagen 2D y un mapa de profundidad, combinación que se ha definido en la literatura como imágenes RGB-D, la cual, combinada con técnicas de procesamiento de imágenes, permite caracterizar aspectos de la escena capturada con la posibilidad de aplicar segmentación no sólo por forma y color, sino también por distancia. No es necesario hacer uso del SDK del Kinect proporcionado por Microsoft para su plataforma de desarrollo Visual Studio. Si bien este software incluye varias herramientas básicas para el Kinect, como la esqueletización del usuario, existen herramientas de desarrollo libres que ofrecen la mayoría de las características que tiene el software de la compañía establecida en Redmon aunque con ciertas limitantes; ejemplos de estas herramientas son OpenNI y Processing. Esto se traduce, por tanto, en mayor ahorro y libertad de desarrollo, sin embargo, recordemos que Visual Studio ofrece licencias gratuitas para estudiantes y profesores universitarios siempre y cuando la aplicación no sea para fines comerciales.

Los trabajos desarrollados en la UAQ en el área de procesamiento de imágenes, han mostrado un gran enfoque hacia la programación modular, es decir, cada aporte realizado por las diversas investigaciones es adaptable en otro sistema. Esta misma filosofía debe ser seguida a fin de que el software final sea flexible, así como de fácil actualización para adecuarse a las necesidades específicas del cliente o usuario final, además, se debe tomar en cuenta que localmente es la primera vez que se trabaja con un sensor de este tipo, por lo que el resultado de esta investigación deberá poderse aplicar fácilmente en futuros trabajos.

1.4. Planteamiento general

A continuación se muestra un diagrama que muestra la metodología a seguir para el desarrollo de la aplicación así como la descripción de cada uno de sus bloques (Fig. 1.1). Dicho diagrama ejemplifica de igual manera cómo será el funcionamiento del sistema final trabajando.

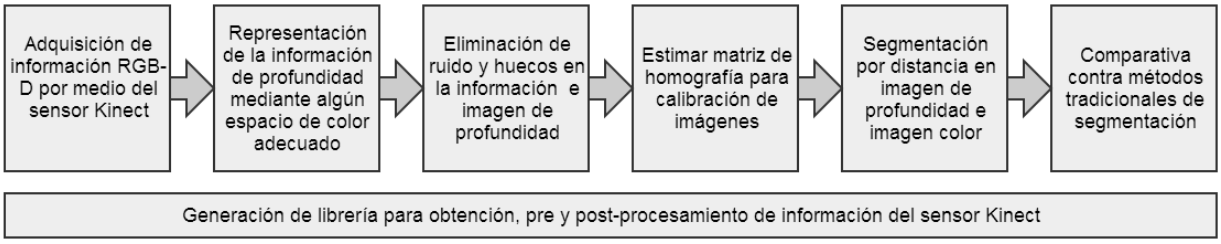


Figura 1.1: Diagrama a bloques de la metodología a seguir.

Adquisición de información RGB-D por medio del sensor Kinect: Esta es la parte de inicio del proyecto, una vez comprado el sensor, se deberán tomar varias imágenes, bajo diferentes condiciones de iluminación, en ambientes cerrados y abiertos, a fin de analizar qué pasa con la información de profundidad del sensor.

Representación de la información de profundidad mediante algún espacio de color adecuado: Es necesario hacer pruebas de conversión de espacios de color para determinar el espacio de color más adecuado para representar la información de profundidad que proporciona el sensor a fin de facilitar el proceso de segmentación.

Eliminación de ruido y huecos en la imagen: Las imágenes adquiridas por medio del sensor Kinect presentan ruido y huecos, a fin de evitar esto, es posible aplicar filtros como él de media e inclusive morfología matemática. Esto permitirá que la información no esté incompleta evitando problemas con las mediciones.

Estimar matriz de homografía para calibración de imágenes: Para cada conjunto de imágenes adquiridas se deberá compensar la distancia física que existe entre la cámara RGB y el sensor de profundidad del Kinect, con el fin de que la información de color y profundidad sea correspondiente punto a punto, esto se hace mediante un proceso de homografía, a cada píxel de la imagen le

debe corresponder su información de profundidad.

Segmentación por forma, color y distancia: Es necesario determinar los operadores adecuados para realizar la segmentación por forma, color y distancia, sobre todo de esta última, ya que no es común trabajar con dispositivos que entreguen información de profundidad a la par de una imagen color.

Comparativa contra métodos tradicionales de segmentación: Se deben contrastar los resultados obtenidos a partir de la metodología de segmentación propuesta contra los métodos y técnicas de segmentación utilizados tradicionalmente a fin de establecer ventajas y desventajas que sirvan para futuros trabajos.

Generación de librería con procesos de segmentación: Todo el trabajo realizado deberá ser contenido en una librería que pueda servir para investigaciones y trabajos posteriores dentro de la universidad.

Capítulo 2

Revisión de la literatura

2.1. Estado del arte

El procesamiento de imágenes se ha convertido en una herramienta valiosa dentro de múltiples procesos industriales, así como un instrumento muy útil dentro de laboratorios y centros de investigación universitarios. La generación y desarrollo de sensores de captura de imágenes cada vez más baratos así como un mayor interés por el desarrollo de software propio ha permitido una gran cantidad de sistemas relacionados con esta área. Si bien el sensor Kinect es relativamente de reciente aparición, gran cantidad de trabajos han sido realizados al respecto con la intención de ofrecer una alternativa de bajo costo combinando una cámara RGB y un sensor de profundidad. En el presente capitulo se presentan la herramientas necesarias para la realización del trabajo propuesto en esta tesis.

2.2. Sensor Kinect

De acuerdo a la página web de Microsoft `http://kinectforwindows.org/` el sensor Kinect apareció en Noviembre de 2011 como un accesorio de la consola Xbox 360. Fue desarrollado por la compañía Prime Sense en colaboración con Microsoft. En Febrero de 2012 fue lanzada

una versión especialmente desarrollada para el sistema operativo Windows 7. El sensor Kinect (Fig. 2.1) tiene una cámara RGB así como un emisor y sensor de luz infrarroja, en conjunto permiten capturar imágenes a color y entregar la información de profundidad de cada píxel en la escena (Cruz, 2012).

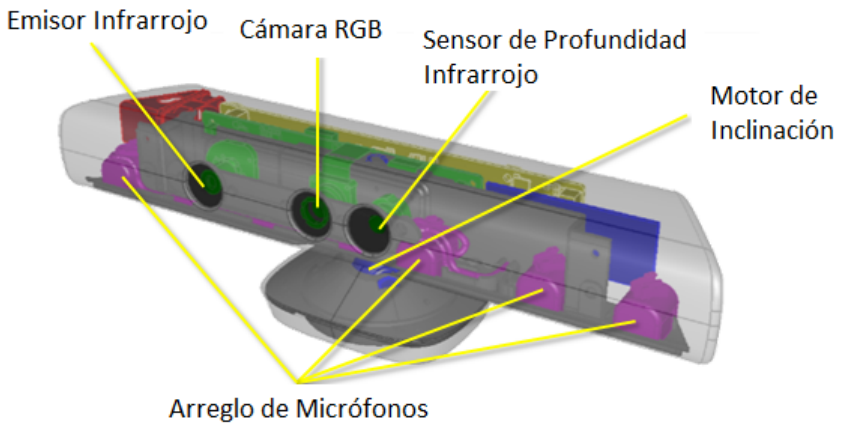


Figura 2.1: Elementos del sensor Kinect (Fuente: http://kinectforwindows.org/).

La cámara RGB opera a 30 Hz y entrega imágenes de 640x480 pixeles con 8 bits por canal. El campo de visión (FOV, por sus siglas en inglés) para la imagen a color es de 57 grados en horizontal y 43 grados en vertical. Es posible cambiar la resolución de la cámara a 1280x1024 pixeles pero corriendo a 10 cuadros por segundo. Por su parte la cámara infrarroja opera a 30 Hz y entrega imágenes con 1200x960 pixeles, estas son disminuidas a 640x480 pixeles con 11 bits, con lo cual provee 2048 niveles de sensibilidad. El FOV del sistema es de 57 grados en horizontal y 43 grados en vertical. El rango de operación estándar es de entre 0.8 y 4 m. La medición de profundidad se basa en un método de luz estructurada para medirla, un patrón de puntos conocido es proyectado por el emisor infrarrojo, el sensor infrarrojo los captura y compara con el patrón conocido. Cada píxel contiene la distancia cartesiana en mm desde el plano de la cámara hasta el objeto más cercano en esa coordenada particular (x,y), esto se muestra en la Fig. 2.2.

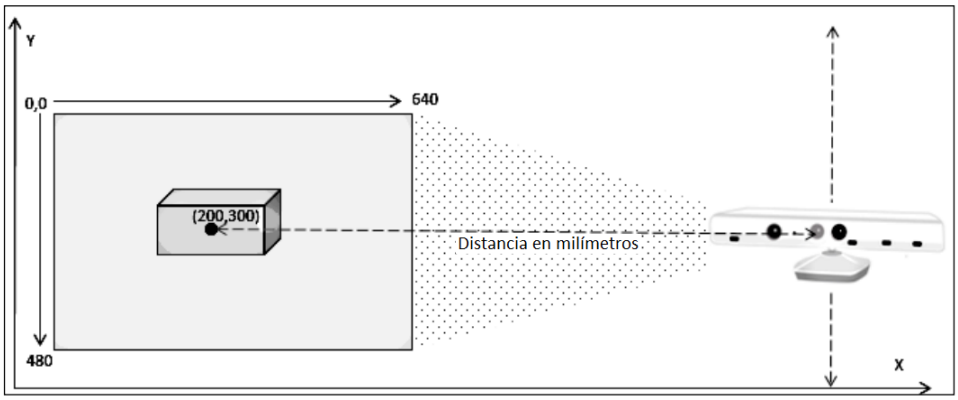


Figura 2.2: Medición de profundidad.

El sensor de profundidad cuenta con dos rangos, estos se muestran en la (Fig. 2.3). El primero es conocido como rango de default (default range) y el otro como rango cercano (near mode), ambos están disponibles en el Kinect para Windows no así para el de Xbox 360 el cual no cuenta con la información del rango cercano.

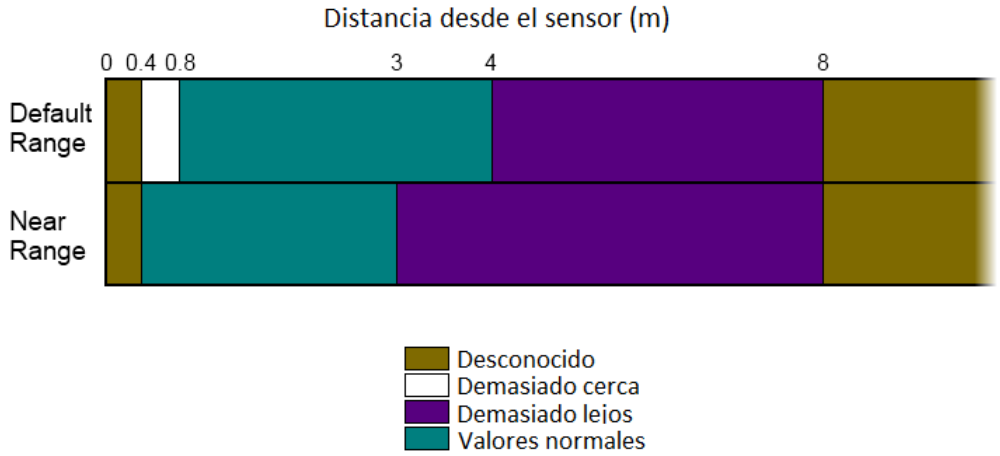


Figura 2.3: Rango del espacio de profundidad (Fuente: http://kinectforwindows.org/).

Probablemente la función mas conocida de este dispositivo es la que tiene que ver con la detección de articulaciones de usuario, esta función es medular para el objetivo original del Kinect que es la de proveer control de vídeo juegos si la necesidad de un mando físico. El dispositivo puede reconocer hasta 6 usuarios dentro del campo de visión del sensor, Fig. 2.4.

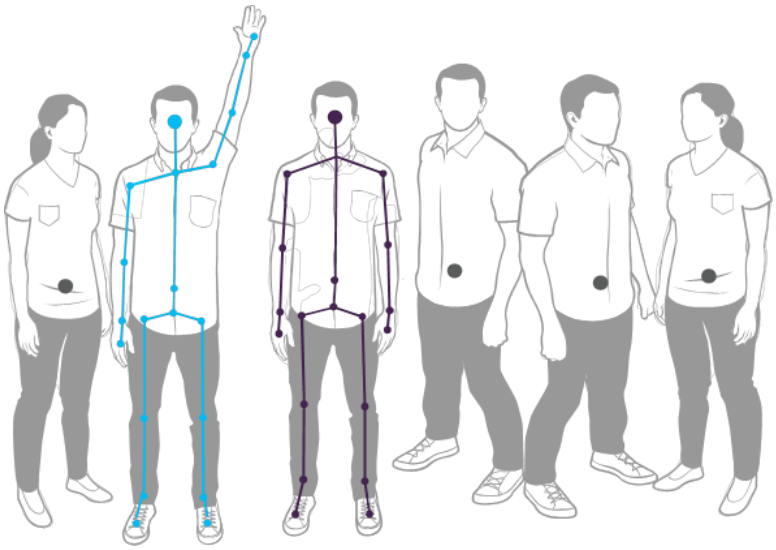


Figura 2.4: Detección de articulaciones con Kinect (Fuente: http://kinectforwindows.org/).

Esta función del sensor permite monitorear 20 articulaciones de dos usuarios, para esto, utiliza la información del sensor infrarrojo y de este mod estima la posición de las articulaciones, Fig. 2.5.

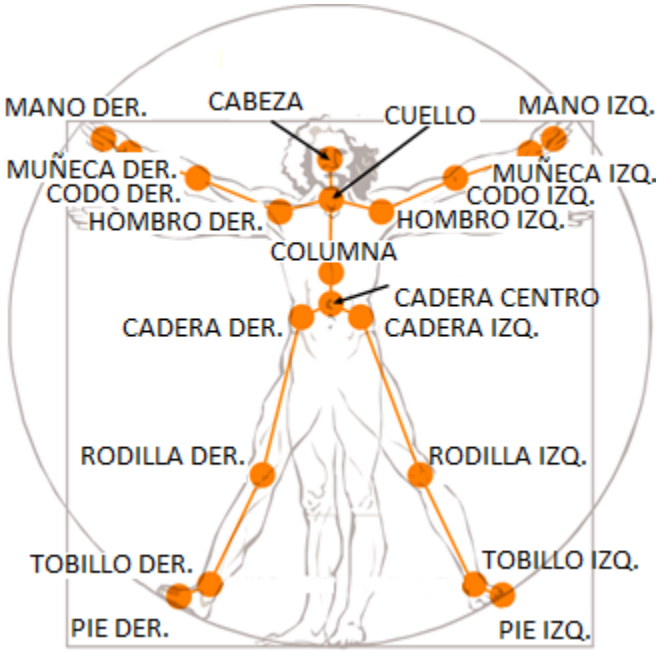


Figura 2.5: Articulaciones detectadas por el Kinect (Fuente: http://kinectforwindows.org/).

2.3. Atributos visuales

Un atributo visual es un atributo visualmente perceptible de un elemento gráfico. El concepto de atributo visual ha sido ampliamente discutido en la literatura. Se dividen en espaciales y de llenado de área, (Fig. 2.6). Los primeros abarcan la orientación, forma, tamaño y posición. La textura y color corresponden a los segundos (von Engeldardt, 2002).

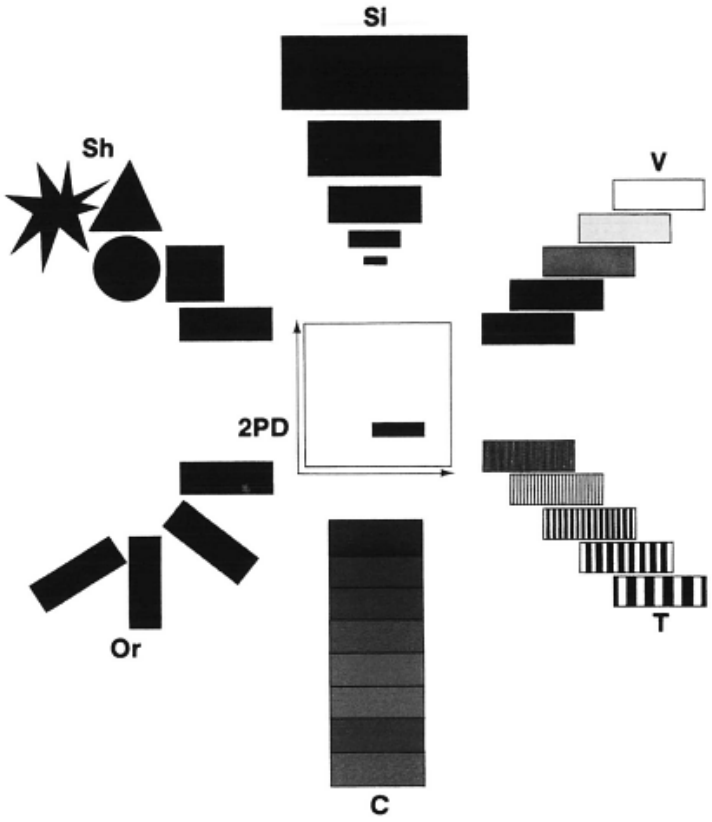


Figura 2.6: Diagrama de Bertin mostrando los diferentes atributos visuales.

Juntos, el espacio y el objeto gráfico así como sus atributos visuales pueden considerados como los “ingredientes“ de una representación gráfica. Las relaciones gráficas corresponden a la forma en que estos “ingredientes“ se combinan en estructuras sintéticas. De esta forma es posible establecer relaciones objeto a objeto y objeto a espacio, con lo cual se puede decir por ejemplo, que “un objeto es mas grande que otro“, “ambos objetos tiene el mismo color“, etc.

Por su parte el SIGGRAPH (Special Interest Group on GRAPHichs and Interactive Tecniques) en su tratado acerca de la percepción de elementos visuales mencionan el color, el matiz, la saturación, el brillo, la textura, la orientación, atributos de profundidad y movimiento como atributos visuales, ademas sugiere como deben ser tratados.

Para los atributos de profundidad se dice que la utilización de estos son para mejorar la percepción de estructuras tridimensionales y para su manejo se incluyen técnicas como:

- Variación del brillo para mostrar el incremento de profundidad.
- Utilización de geometría perspectiva.
- Colocación de referencias a distancias conocidas para distinguir frente y fondo.
- Utilización de objetos trasparentes o translucidos.
- Modificación del brillo (sombreado) para simular superficies.
- Rotación para mejorar la percepción 3D.
- Efectos stereo (gafas 3D).

2.4. Espacios de color

Sirven para facilitar la especificación de los colores de una manera estándar donde cada color se representa por un punto. El espacio de color RGB (Red, Green and Blue), es un espacio cartesiano cúbico, en el que las señales rojas, verdes y azules son independientes y pueden ser sumadas para producir cualquier color dentro del cubo (Russ, 2011). El color negro se encuentra en el origen y el blanco en la arista opuesta a él, por su parte la escala de grises se extiende desde el color negro hasta el blanco a través de la diagonal que los une. Los diferentes colores en este modelo son puntos dentro del cubo definidos por vectores que se extienden desde el origen. Por conveniencia se asume que todos los valores de colores han sido normalizados por lo que el cubo de la (Fig. 2.7) es un cubo unitario (González, 2004).

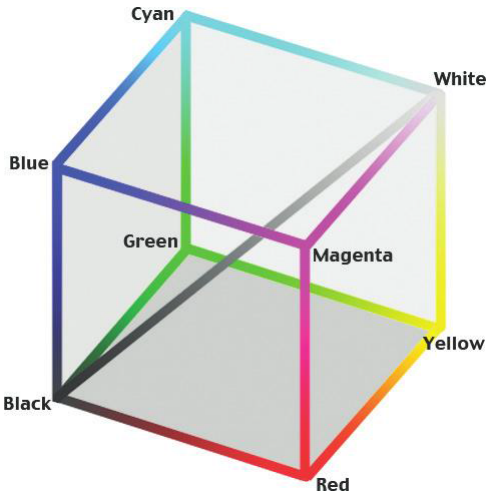


Figura 2.7: Espacio de color RGB (Fuente: Russ ,2011).

El siguiente espacio es el HSI (Hue, Saturation and Intensity, por su denotación en inglés). Cuando el ser humano observa un color no lo describe dando un porcentaje de cada uno de los colores primarios que lo componen, lo hace describiendo su matiz, saturación y brillo (González, 2004b). El matiz es un atributo que describe un color puro, lo que la gente entiende por color; la saturación es la cantidad de color que está presente (Russ, 2011). El brillo es un descriptor subjetivo que es prácticamente imposible de medir, representa la notación acromática de la intensidad y es uno de los factores claves en la descripción de sensación de colores, sin embargo, la intensidad es un descriptor más útil de imágenes monocromáticas, además es un parámetro medible y de fácil interpretación (González, 2004b). Espacios relacionados al HSI son el HSV y el HSL. El espacio en el cual los tres valores son representados puede ser mostrado como un cono circular o hexagonal e incluso como un doble cono como en la (Fig. 2.8), siendo esta última la representación más útil, en el cual el eje del cono representa la escala de grises desde el negro hasta el blanco, la distancia desde el eje central es la saturación y la dirección es el matiz. Los colores primarios RGB se ubican con una separación de 120°entre ellos.

La conversión entre el espacio RGB y HSI se puede llevar a cabo de diversas formas dependiendo de la forma del espacio HSI utilizada, esfera cilindro o doble cono como en la (Fig. 2.8). En todas ellas el eje de intensidad corresponde a la diagonal principal del cubo RGB donde se ubica la escala de grises, sin embargo, ninguna de las geometrías de representación del espacio HSI se ajus-

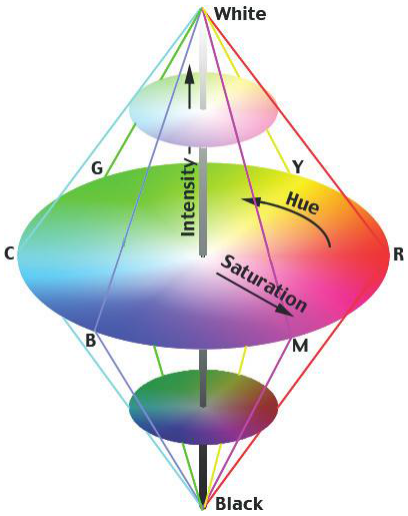


Figura 2.8: Espacio de color HSI (Fuente: Russ ,2011).

tan a la forma del cubo. Esto significa que para representar colores en ambos espacios los valores de saturación se distorsionan en algún punto de la conversión. La forma de pasar del espacio RGB a HSI considerando una representación de doble cono se muestra en las Ecuaciones 2.1 , 2.2 , 2.3 y 2.4.

$$I = \frac{R + G + B}{3} \tag{2.1}$$

$$S = 1 - \frac{3 * Min(R, G, B)}{R + G + B} \tag{2.2}$$

$$H = \begin{cases} cos^{-1}(z) & \text{si } G \geq R \\ 2\pi - cos^{-1}(z) & \text{si } G \leq R \end{cases} \tag{2.3}$$

$$z = \frac{2B - G - R}{2\sqrt{(B - G)^2 + \frac{B-R}{G-R}}} \tag{2.4}$$

Los espacios HSI son útiles para procesamiento de imágenes porque separan la información de color en formas que corresponden al sistema de visión humana y también porque los ejes corres-

ponden a muchas características físicas de especímenes. Las Ecuaciones 2.5 , 2.6 , 2.7 , 2.8 , 2.9 y 2.10 muestran como pasar de RGB a los espacios HSL, las tres primeras, y a HSV, las tres ultimas, según sea necesario.

$$L = \frac{Max(R,G,B) + Min(R,G,B)}{2} \tag{2.5}$$

$$S = \begin{cases} \dfrac{Max - Min}{Max + Min} & \text{si } L \leq 0{,}5 \\ \dfrac{Max - Min}{2 - Max - Min} & \text{si } L \geq 0{,}5 \end{cases} \tag{2.6}$$

$$H = \begin{cases} \dfrac{G - B}{Max - Min} & \text{si } R = Max \\ \dfrac{B - R}{Max - Min} & \text{si } R = Max \\ \dfrac{R - G}{Max - Min} & \text{si } R = Max \end{cases} \tag{2.7}$$

$$V = Max(R,G,B) \tag{2.8}$$

$$S = \begin{cases} 0 & \text{si } Max = 0{,}5 \\ 1 - \dfrac{Min}{Max} & \text{si } Max \neq 0{,}5 \end{cases} \tag{2.9}$$

$$H = \begin{cases} no\,definido & \text{si } Max = Min \\ 60° \times \dfrac{G-B}{Max-Min} + 0° & \text{si } R = Max \text{ y } G \geq B \\ 60° \times \dfrac{G-B}{Max-Min} + 360° & \text{si } R = Max \text{ y } G < B \\ 60° \times \dfrac{B-R}{Max-Min} + 120° & \text{si } G = Max \\ 60° \times \dfrac{B-R}{Max-Min} + 240° & \text{si } B = Max \end{cases} \tag{2.10}$$

2.5. Filtros

A veces la calidad de la imagen no es lo suficiente buena, de forma que no se puede extraer la información adecuadamente, ello implica el tener que utilizar ciertas técnicas de mejora de la calidad de la imagen original (González, 2004). La aplicación de filtros en procesamiento de imágenes permite eliminar el ruido que se llega a presentarse en las imágenes durante la captura de estas. Entre los filtros más comunes tenemos los estadísticos como el de media y mediana.

2.5.1. Filtro de media aritmética

Éste es uno de los filtros más sencillos de aplicar. La fórmula para implementar el filtro de media aritmética se muestra en la Ecuación 2.11. Este filtro reemplaza el valor de un píxel central por el nivel promedio de valores de intensidad de los pixeles vecinos a este y de él mismo (González, 2004).

$$f(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t) \tag{2.11}$$

Donde mxn es la dimensión de la matriz del filtro asociada, generalmente n=m, siendo m=3 el valor más común pero con la posibilidad de variarse con incrementos de 2 (3, 5, 7, 9, etc.), a fin de garantizar un píxel central el cual será modificado. El ruido es disminuido como consecuencia del difuminado.

2.5.2. Filtro de mediana

Probablemente el filtro estadístico más conocido, el cual como su nombre lo dice reemplaza el valor del píxel central por la media de los niveles de intensidad en sus alrededores, Ecuación 2.12.

$$f(x,y) = mediana_{(s,t)\in S_{x,y}} g(s,t) \tag{2.12}$$

El valor del píxel original se incluye en el cálculo de la mediana. Este tipo de filtros son bastante populares por sus excelentes capacidades para la reducción de ruido generando un menor desdibujado a la imagen (González, 2004).

2.6. Segmentación y morfología matemática

La segmentación es la técnica mediante la cual se extrae de la imagen cierta información para su posterior uso, se basa en dos principios, discontinuidad y similitud. Es conveniente entonces hacer mención de la segmentación orientada a bordes (discontinuidad) y de la orientada a regiones (similitud). Una región es un área de la imagen en la que sus pixeles poseen propiedades similares (intensidad, color, etc.), mientras que un borde es una línea que separa dos regiones, por tanto de diferentes propiedades. Tanto la detección de bordes como la de región implican una manipulación de la imagen original, que supone en definitiva una transformación de la imágenes originales de

forma que los valores de los pixeles son modificados mediante ciertas funciones de transformación u operadores (Pajares, 2008).

Algunos operadores basados en la primera derivada para la detección de bordes son, gradiente de una imagen, operadores de Sobel, operador de Prewitt, operador de Roberts, entre otros. Otros operadores pero basados en la segunda derivada serían el operador Laplaciano, operador Laplaciano de Gaussiana y diferencia de Gaussianas. Todos estos han sido ampliamente estudiados en la literatura, además es importante notar que no hay un método efectivo para todas las aplicaciones, más bien el tipo de aplicación determina el método a usar.

2.6.1. Umbralización

Debido a sus propiedades intuitivas, simplicidad de implementación y rapidez computacional, el umbralizado de imágenes juega un papel central en aplicaciones de segmentación de imágenes. Supongase que se tiene una imagen compuesta de objetos claros - blancos sobre un fondo oscuro de modo que los pixeles correspondientes a los objetos y al fondo tenga valores de intensidad agrupados en dos dominios. Una forma sencilla de separar los objetos del fondo consiste en seleccionar un valor de umbral T, que separe dichos valores. Por lo tanto, cualquier punto (x, y) en la imagen en la cual $f(x, y) > T$ se le llama punto de objeto o interés, de lo contrario, se le llama punto correspondiente al fondo. En otras palabras la imagen segmentada $g(x, y)$ es dada por la Ecuación 2.13.

$$g(x,y) = \begin{cases} 1 & \text{si } f(x,y) > T \\ 0 & \text{si } f(x,y) <= T \end{cases} \tag{2.13}$$

Donde T es una constante aplicable sobre toda la imagen. El algoritmo del umbralizado puede ser modificado de acuerdo a nuestros requerimientos, esto incluye segmentar sólo un rango de valores seleccionando un par de umbrales $T1$ y $T2$ como se ve en la Ecuación 2.14.

$$g(x,y) = \begin{cases} 1 & \text{si } f(x,y) > T1 \text{ y } f(x,y) < T2 \\ 0 & \text{si } f(x,y) <= T1 \text{ y } f(x,y) >= T2 \end{cases} \tag{2.14}$$

Es posible inclusive modificar dicho algoritmo para segmentar con diferentes colores las áreas de interés como en la (Fig. 2.9.

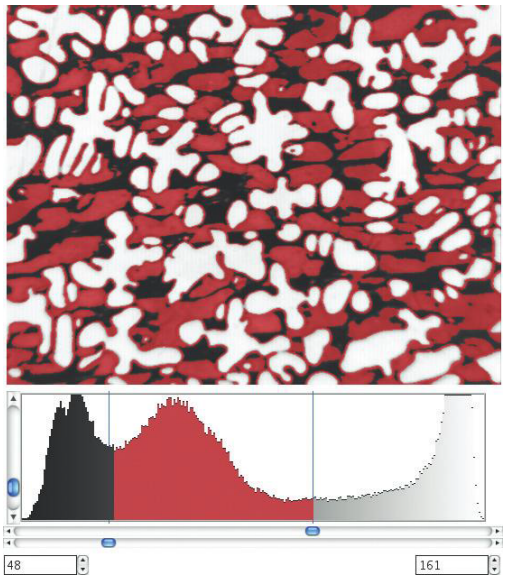


Figura 2.9: Umbralizado con color.

La morfología por su parte se relaciona con la estructura o la forma de objetos. Las operaciones morfológicas se utilizan en el proceso de dividir imágenes en segmentos para facilitar la búsqueda de objetos de interés. Los usos más comunes de los operadores morfológicos incluyen el llenado de agujeros pequeños, la separación de adyacente de objetos levemente traslapados, y el ensamble de límites rotos en segmentos continuos (Morales et al., 2007).

2.6.2. Elementos estructurantes

La morfología matemática tiene como objeto la descripción de una imagen X, uniendo patrones similares en diversos puntos de la misma. Esto se lleva a cabo conceptualmente mediante la introducción de los elementos estructurantes (EE). La comparación de este elemento con la estruc-

tura a tratar (imagen) debe verificar relaciones conjuntistas. Es en particular sobre este concepto, el elemento estructural, que pondremos un interés particular en esta sección. Las transformaciones básicas en la morfología matemática son la erosión y dilatación morfológicas. Cabe mencionar que el elemento estructural puede tener diferentes formas, como por ejemplo un hexágono, un disco; depende de la aplicación que se le requiera dar.

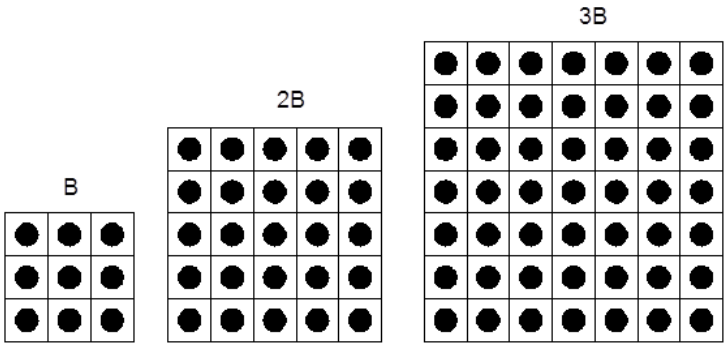


Figura 2.10: Ejemplo de elementos estructurantes.

2.6.3. Erosión y dilatación

Son las dos transformaciones morfológicas más importantes desde el punto de vista del procesamiento digital de imágenes. Su utilidad es importante no solo como operadores aislados, sino como base para crear transformaciones más complejas.

La transformación morfológica dilatación $\oplus$, Ecuación 2.15, combina dos conjuntos utilizando la adición de vectores (Fig. 2.11).

$$A \oplus B = \{d \in E^2 \colon d = x + b \text{ para cada } x \in X \text{ y } b \in B\} \tag{2.15}$$

La erosión $\ominus$, Ecuación 2.16, combina dos conjuntos utilizando las sustracción de vectores (Fig. 2.12), es dual a la dilatación, ninguno de los dos son transformaciones reversibles.

$$A \ominus B = \{d \in E^2 \colon d + b \in X \text{ para cada } b \in B\} \tag{2.16}$$

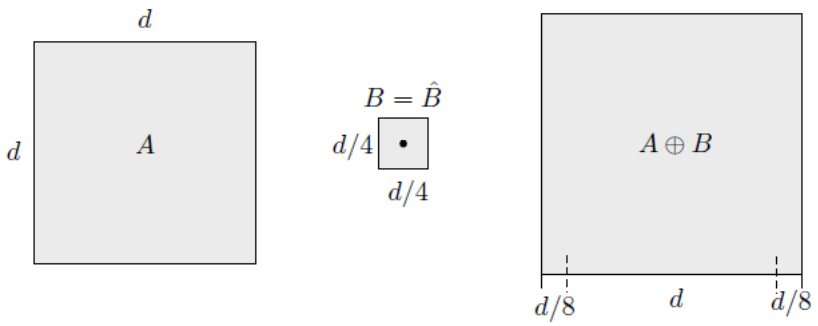


Figura 2.11: Dilatación de A por el elemento B.

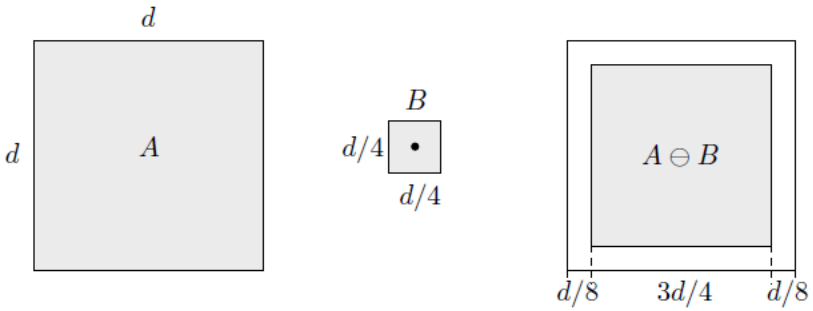


Figura 2.12: Erosión de A por el elemento B.

2.6.4. Apertura y cerradura

Las transformaciones Dilatación y Erosión son la base de un conjunto de filtros más complejos en el análisis de imágenes, la apertura y cerradura.

La erosión seguida de una dilatación crea una transformación morfológica importante llamada apertura (Ecuación 2.17), la operación apertura se puede ver en la (Fig. 2.13) .

$$A \circ B = (X \ominus B) \oplus B \tag{2.17}$$

Mientras que la dilatación seguida de una erosión crea una transformación morfológica llamada cierre (Ecuación 2.18), dicha transformación puede ser vista en la en la (Fig. 2.14).

$$A \bullet B = (X \oplus B) \ominus B \tag{2.18}$$

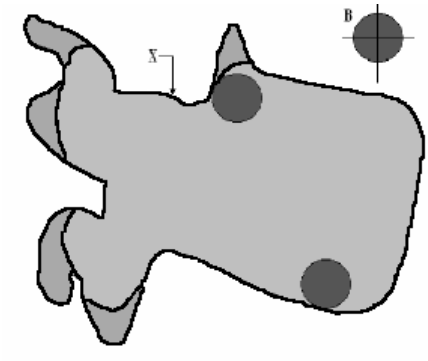


Figura 2.13: Apertura de X por el elemento B.

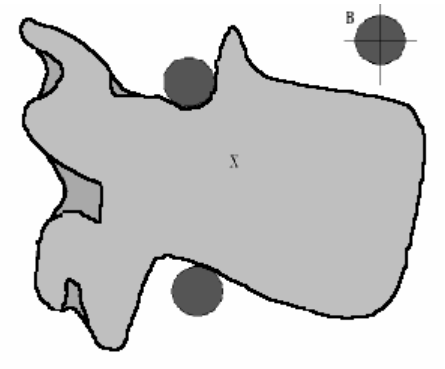


Figura 2.14: Cerradura de X por el elemento B.

Al igual que la apertura y cerradura morfológicas son obtenidas por composición de transformaciones morfológicas básicas como la erosión y la dilatación, otros filtros morfológicos pueden ser obtenidos por composición de la apertura y cerradura. Sabemos que la cerradura morfológica trabaja eliminando regiones blancas de la imagen, mientras que la cerradura morfológica trabaja sobre las regiones obscuras.

2.7. Homografía

Los puntos de una imagen desde un punto de vista están relacionados con los puntos de otra vista por medio de una homografía plana. La homografía juega un papel importante en la geometría

de múltiples vistas ya que relaciona vistas planas (Flores, 2008). Puede ser determinada a partir de 4 pares de puntos correspondientes de dos vistas. Cada par de puntos correspondientes entrega dos restricciones para dos grados de libertad. Por lo tanto, el número mínimo de datos para calcular una homografía son cuatro pares correspondientes desde dos vistas de cámaras distintas. Una homografía también es llamada una transformación proyectiva de plano la cual describe qué ocurre con las posiciones de objetos cuando el observador cambia su posición. Puede ser representada por medio de una matriz no singular de 3x3 como la mostrada en la Ecuación 2.19, que a su vez puede ser reescrita de manera más simple como en la Ecuación 2.20.

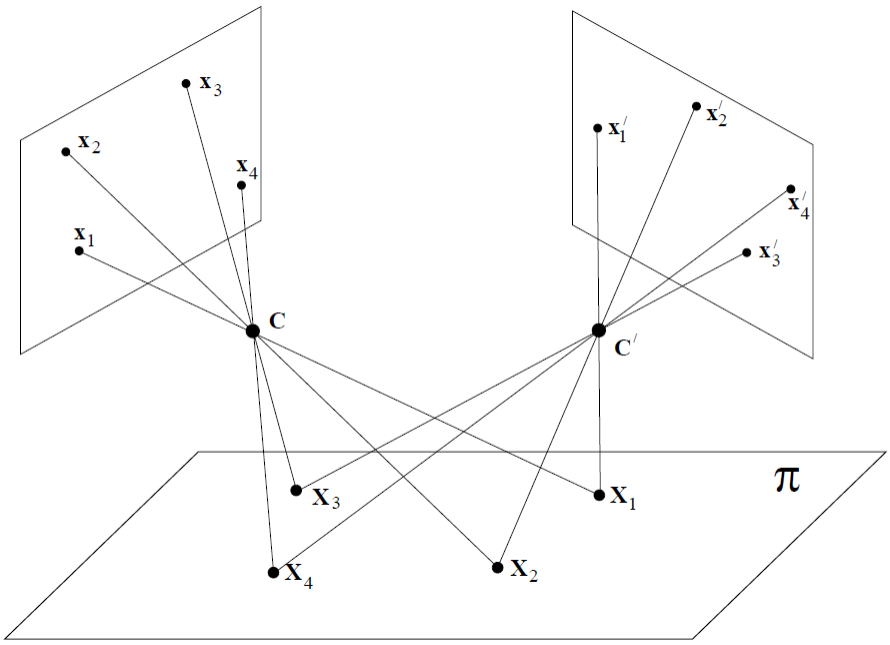


Figura 2.15: Dos puntos de vista de un plano pi realizado con dos sensores.

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{2.19}$$

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = H \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{2.20}$$

Si consideramos x, y e x', y' como los puntos correspondientes en un plano, la transformación proyectiva puede ser reescrita de forma no homogénea como se muestra en la Ecuaciones 2.21 y 2.22.

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{2.21}$$

$$y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{2.22}$$

Finalmente, una característica de las homografías es que conserva la naturaleza de los elementos transformados, es decir, un punto se transforma en otro punto. En los anexos se muestra la implementación del algoritmo de homografia en Matlab desarrollado por David Young (2008) traducido a Python.

2.8. Python

Python es un lenguaje de programación moderno desarrollado por Guido van Rossum en la década de los 90's. Tiene ese nombre por el famoso grupo de comedia. A pesar de que Python no es perfecto para todo tipo de aplicaciones, por ejemplo aquellas en que la velocidad es fundamental, sus fortalezas lo convierten en una gran opción para múltiples situaciones (Ceder, 2010).

Python es un lenguaje multiplaformas que corre lo mismo en Windows, Linux/UNIX y plataformas Macintosh, abarcando desde super computadoras hasta celulares. Puede ser usado para el desarrollo de aplicaciones pequeñas y prototipos rápidos, con la ventaja de poderse escalar fácilmente hacia aplicaciones de mayor tamaño. Cuenta con una interfaz gráfica de usuario poderosa y fácil de usar, además hay disponibles en la red múltiples librerías de programación para usarlas en conjunto. Lo mejor de todo es que es libre.

Las ventajas de Python se enumeran a continuación:

- Es fácil de usar. Los programadores familiarizados con lenguajes tradicionales aprenden rápido Python. Una aplicación en Python toma alrededor de una quinta parte de tiempo de lo que toma desarrollar esa misma aplicación en C o Java.

- Es expresivo. Una simple línea de código en Python hace más que en la mayoría de lenguajes. Entre menos lineas de código se escriban más rápidamente se completa un proyecto.

- Es fácil de leer. Python tiene la filosofía de que entre más fácil de entender sea un código, este será más fácil de depurar, mantener y modificar.

- Incluye todo lo necesario para trabajar. La librería estandard de Python incluye múltiples módulos para trabajar por ejemplo con mails, páginas web, bases de datos, llamados del sistema operativo, desarrollo de interfaces de usuario y más. Si esto no fuera suficiente en la red existen múltiples librerías que expanden la posibilidades de Python aún mas.

- Es múltiplaforma. Windows, Mac, Linux, UNIX, y más.

- Es libre. Esta desarrollado bajo el modelo de código fuente abierto, disponible gratuitamente. Con cualquier versión de Python es posible desarrollar aplicaciones personales o comerciales sin tener que pagar por licencias.

Python es ampliamente utilizado por empresas como Google, Rackspace, Industrial Light & Magic, Honeywell, etc.

Capítulo 3

Metodología

En este capitulo se presentan los pasos implementados para la realización de este trabajo de investigación, la metodología es concentrada en un par de diagramas a bloques que son mostrados en las Figs. 3.1 y 3.2. Ambos diagramas a bloques pueden ser interpretados como dos interfaces de usuario diferentes, la primera de ellas permite obtener información del dispositivo y en la segunda se implementan las técnicas de pre-procesamiento necesarias.

El proyecto, de manera general, se divide en dos partes, en la primera se hace incapie en entender los elementos necesarios para lograr el correcto funcionamiento del dispositivo lo que implica la instalación de drivers y software de desarrollo, la obtención de la información de color y de profundidad que proporciona el sensor y los modos de despliegue de éstas, así como las estructuras de programación necesarias para su almacenaje para procesamiento posterior.

Para la segunda parte y ya con la información que proporciona el sensor almacenada, se debe analizar las necesidades de filtrado de los datos de profundidad así como posibilidades de mejora, se deben implementar los filtros estadísticos y morfológicos, el proceso de homografía y la aplicación de falso color a la información de profundidad; finalmente analizar qué operadores permiten de mejor manera la segmentación por distancia.

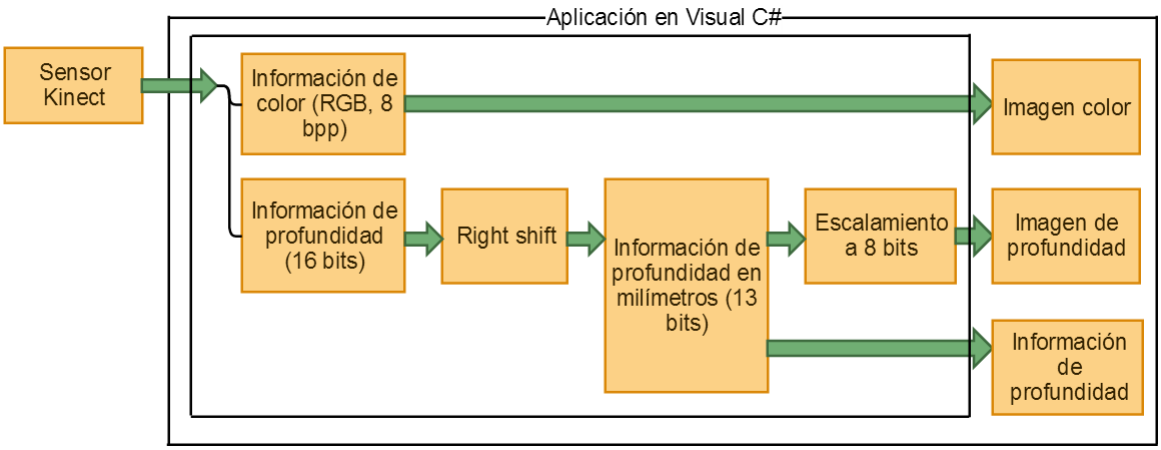


Figura 3.1: Diagrama a bloques de la metodología.

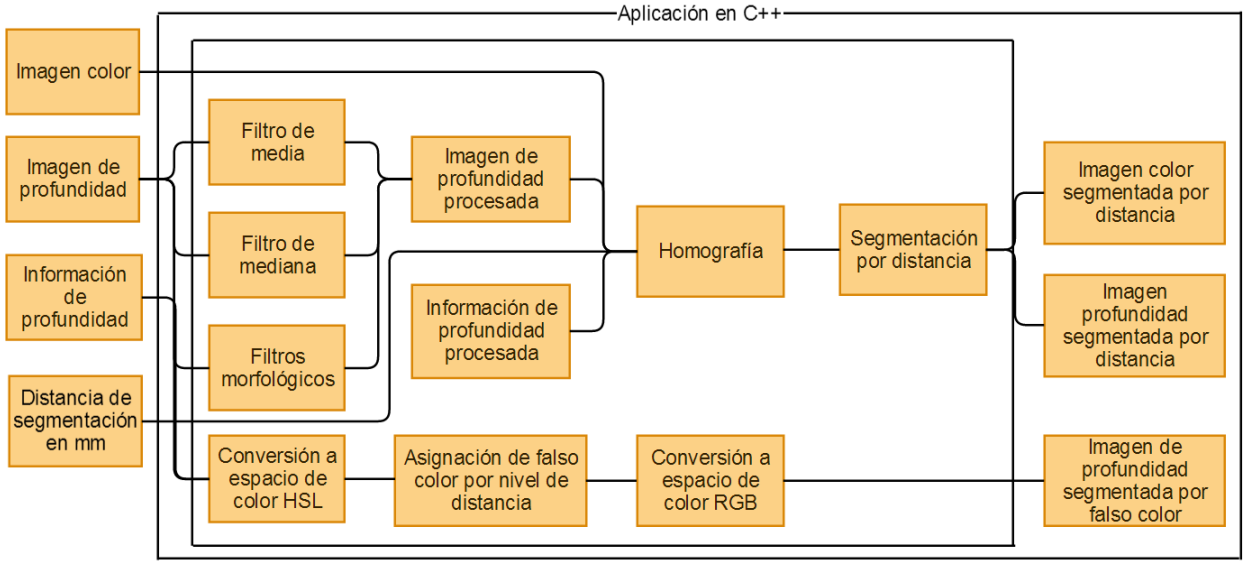


Figura 3.2: Diagrama a bloques de la metodología (segunda parte).

3.1. Instalación y configuración del sensor Kinect

Esta primer etapa requiere de una serie de pasos sencillos que abarcan desde la adquisición del dispositivo hasta la parte de captura de primeras imágenes. Estos se muestran en el diagrama a bloques de la Fig. 3.3.

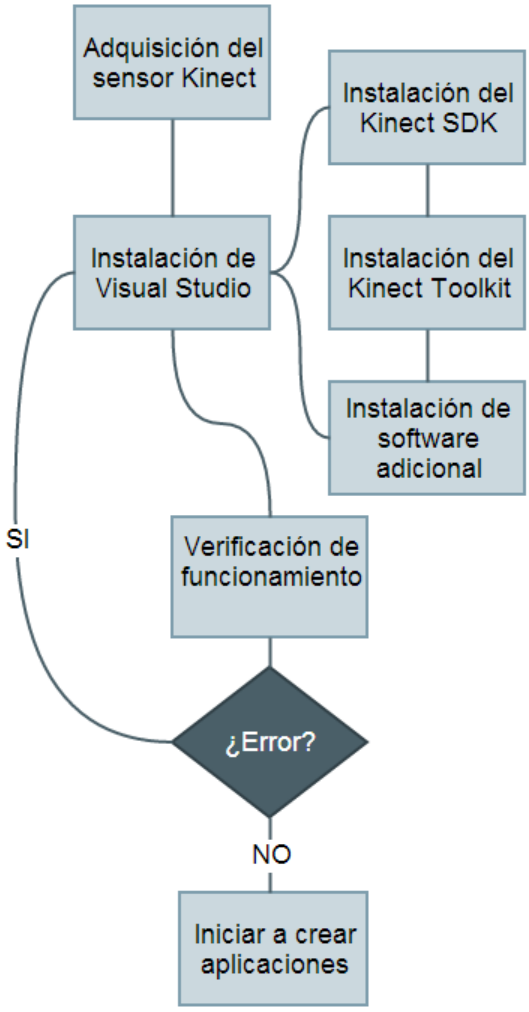


Figura 3.3: Pasos para la instalación y configuración del sensor.

Adquisición del sensor Kinect: El sensor Kinect para Windows incluye el sensor Kinect y un cable "Y" de alimentación con conexión USB, dichos elementos son mostrados en la (Fig. 3.4). Como se mencionó en la sección anterior, hay dos versiones de sensor Kinect, una que se utiliza en la consola Xbox 360 y una especialmente diseñada para la plataforma de Windows, esta última incluye el conector "Y" que permite conectar el dispositivo a la PC y a su vez a la alimentación de voltaje.

Figura 3.4: Elementos que contiene el sensor Kinect para Windows.

Instalación del IDE de desarrollo Visual Studio: Para el desarrollo de aplicaciones de Kinect es necesario descargar e instalar la IDE (Integrated Delevelopment Environment, Ambiente Integrado de Desarrollo por sus siglas en inglés) de Visual Studio la cual puede ser a partir de su versión 2010 en adelante (Fig. 3.5). Microsoft pone de manera gratuita a la disposición de estudiantes universitarios versiones profesionales de Visual Studio en su sitio web `www.dreamspark.com`, basta pasar por un proceso de verificación a través de la cuenta de correo electrónico institucional. Igualmente es posible utilizar las ediciones "Express" de este software para desarrollar aplicaciones de Kinect, esta herramienta limitada en algunos aspectos, es gratuita para todo el público interesado en el desarrollo de software.

Visual Studio

Figura 3.5: Logo de Visual Studio 2012 y 2013.

Instalación del SDK de Kinect para Windows: En aras de lograr la completa funcionalidad del sensor Kinect se deben descargar e instalar los controladores y demás elementos de software que proporciona el mismo Microsoft a través de su página web `www.kinectforwindows.com`. El primer paquete de software que Microsoft recomienda instalar es el Kinect for Windows SDK (kit

de desarrollo de software) que, al momento de escribir este texto, se encuentra en su versión 1.8. Precisamente, en este se incluyen todos los drivers necesarios para el funcionamiento de la cámara de vídeo, del emisor y sensor infrarrojo, de los micrófonos, del acelerómetro así como del motor de orientación vertical del dispositivo.

Instalación del conjunto de herramientas para desarrollador del sensor Kinect: A continuación se debe instalar el Kinect for Windows Developer Toolkit (conjunto de herramientas para desarrollador windows para Kinect) en su versión 1.8. Aquí se incluyen ejemplos básicos de uso del sensor Kinect y de programación desarrollados en Visual Studio abarcando los lenguajes Visual Basic, Visual C++ y Visual C#. Se incluyen, además, un ejemplo que utiliza Matlab y uno más que hace uso de la librería de procesamiento de imágenes libre de Intel OpenCV junto con C++ puro.

Instalación de software opcional: El Microsoft Speech Plataform SDK es una herramienta extra que permite desarrollar aplicaciones con reconocimiento de voz y esta disponible para gran cantidad de idiomas incluido el español. En este trabajo no se hace uso de esta herramienta por lo que su instalación se omite, sin embargo, en caso de que se requiera hacer uso de ellas Microsoft recomienda instalarlo posterior al conjunto de herramientas para desarrollador. Finalmente y dado el caso de que solamente se requiera ejecutar una aplicación de Kinect en una PC, como por ejemplo en un equipo de un usuario final o similar, basta con instalar el Kinect Runtime el cual únicamente incluye los drivers del sensor Kinect así como el ambiente de tiempo de ejecución necesarios para ejecutar dichas aplicaciones.

Verificación de funcionamiento: Una vez instalado el software anterior, se debe conectar el sensor Kinect a la alimentación y posteriormente a la PC por medio de cable USB. La primera vez que se conecte el sensor la PC detectara el dispositivo e iniciara la configuración de drivers, cuando todo el proceso termine se vera una ventana como la de la (Fig. 3.6). Además, el sensor Kinect deberá mostrar el led que tiene en la parte frontal iluminado de color verde sin parpadear.

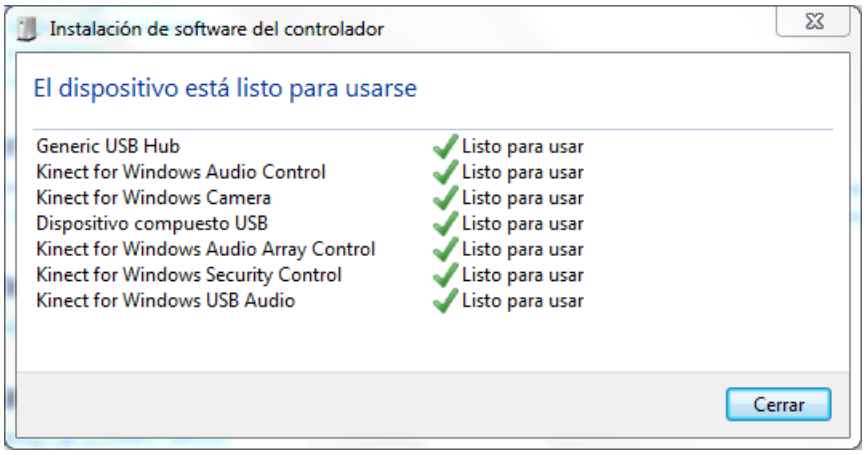


Figura 3.6: Primera conexión del sensor Kinect a la PC.

En caso de que dicha ventana no aparezca o el led frontal del sensor se ilumine de color rojo se debe verificar en el administrador de dispositivos de Windows que los diferentes componentes del Kinect hayan sido reconocidos e instalados correctamente, ver Fig. 3.7. En caso afirmativo se recomienda desconectar el sensor, reiniciar la computadora y repetir los pasos anteriores. En el caso contrario y en el de no lograr un funcionamiento correcto del sensor se debe desconectar el sensor y desinstalar todo software relacionado con el sensor Kinect, reiniciar la computadora y volver a iniciar con la instalación del software. En un caso último en que bajo ninguna circunstancia se logre echar a andar el dispositivo, se debe probar en otra PC y descartar un mal funcionamiento del sensor o de la misma computadora y, en un caso extremo, solicitar soporte al distribuidor del sensor Kinect y al fabricante.

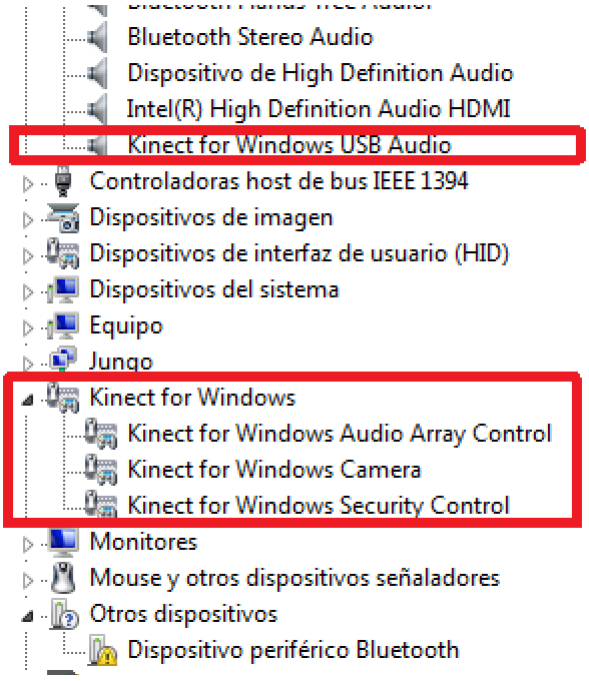


Figura 3.7: Dispositivos instalados correctamente.

3.2. Aplicación básica para sensor Kinect

Dentro del conjunto de herramientas para desarrollador del Kinect se incluyen varios ejemplos de programas utilizando Visual C++ y Visual C#. Lo primero que salta a la vista al contrastar las diferencias entre ambos lenguajes de programación es la longitud de los programas y la cantidad de archivos necesarios, siendo los de Visual C# más compactos y por ende entendibles. Otra característica a considerar es que Visual C# permite desarrollar aplicaciones mediante código administrado lo cual permite implementaciones mas seguras, con mejor manejo de errores y recolección de basura. A continuación se describen de manera general los pasos para generar una aplicación básica haciendo uso del sensor Kinect y el lenguaje de programación Visual C# a modo de diagrama de bloques.

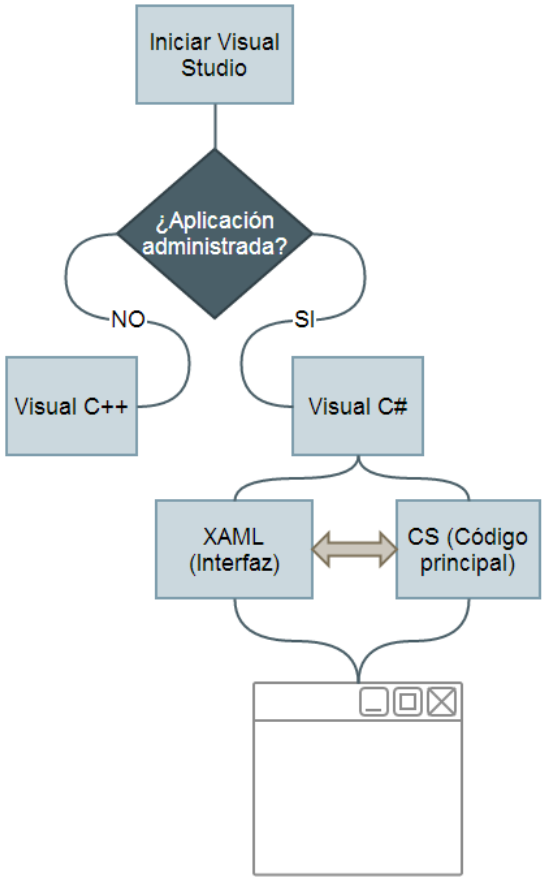


Figura 3.8: Pasos para generar una aplicación que use el sensor Kinect.

Los pasos para crear la aplicación que haga uso del sensor Kinect se enumeran a continuación:

1. Iniciar Visual Studio 2010, 2012, ó 2013.
2. Seleccionar new project.
3. En templates, seleccionar el lenguaje de programación Visual C#.
4. En el menú de apps de C# seleccionar WPF Application.
5. Indicar un nombre para el proyecto.
6. Dar click en OK.
7. Agregar la referencia Microsoft.Kinect.dll.

8. Diseñar la interfaz en XAML:

   - Agregar ventanas para mostrar imagen color, imagen de profundidad o de usuario esqueletizado según se necesite.
   - Agregar botones y demás controles WPF según sea necesario.

9. En el codigo cs:

   - Iniciar Kinect.
   - Habilitar flujo de información color, profundidad, etc.
   - Convertir la información de color, de profundidad, etc., a mapa de bits (BMP) para desplegar en la interfaz.
   - Almacenar información de color, de profundidad o la que se requiera.
   - Deshabilitar flujo de información color, profundidad, etc.
   - Detener Kinect.

Con Visual Studio abierto, los pasos del dos al seis de la lista anterior se muestran en la Fig. 3.9.

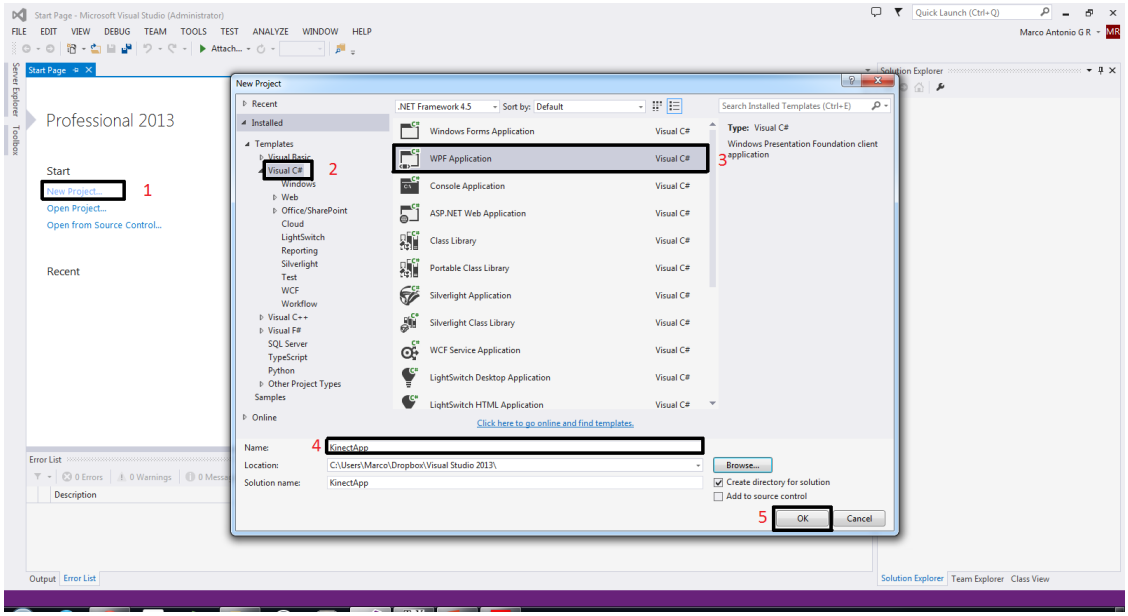


Figura 3.9: Pasos iniciales para crear una aplicación de Kinect.

Como paso 7, a continuación en el Solution Explorer es necesario hacer click derecho en References y después en Add Reference, esto se muestran en la Fig. 3.10. En la ventana que se abre (Fig. 3.11) hay que dar click en botón Browse y en la nueva ventana que aparece se tiene que localizar la DLL (Dynamic-Link Library) Microsoft.Kinect.dll que por lo general se encuentra en la ruta C:\Program Files\Microsoft SDK's\Kinect\v1.8. A continuación se debe dar click en Add (Fig. 3.12), y finalmente en OK (Fig. 3.13).

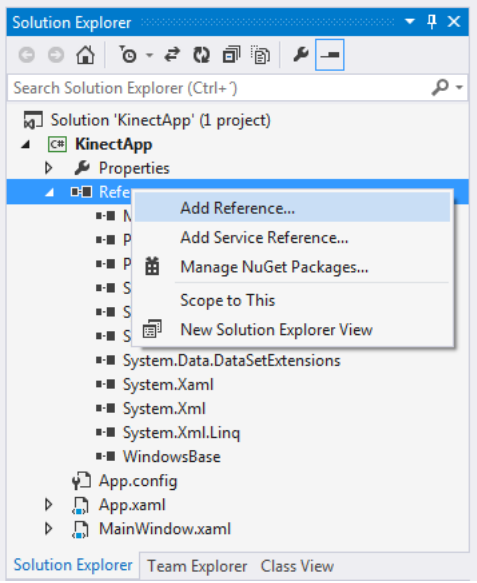


Figura 3.10: Agregando referencia (parte 1).

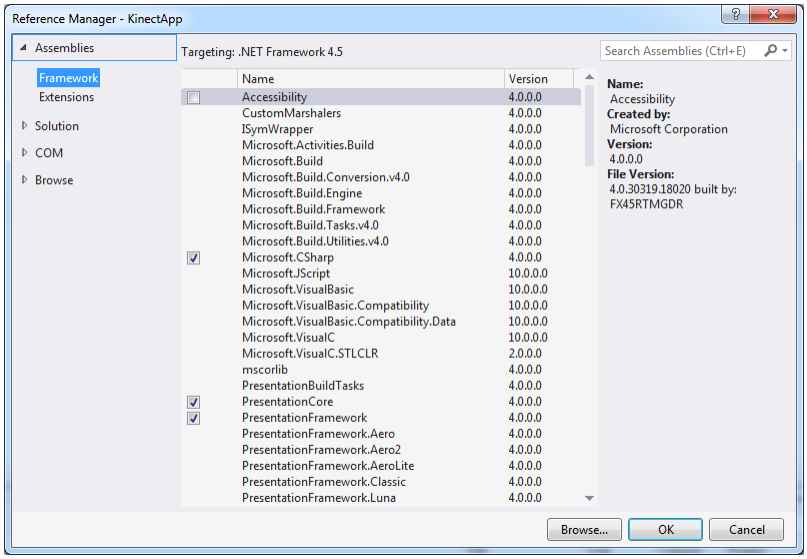


Figura 3.11: Agregando referencia (parte 2).

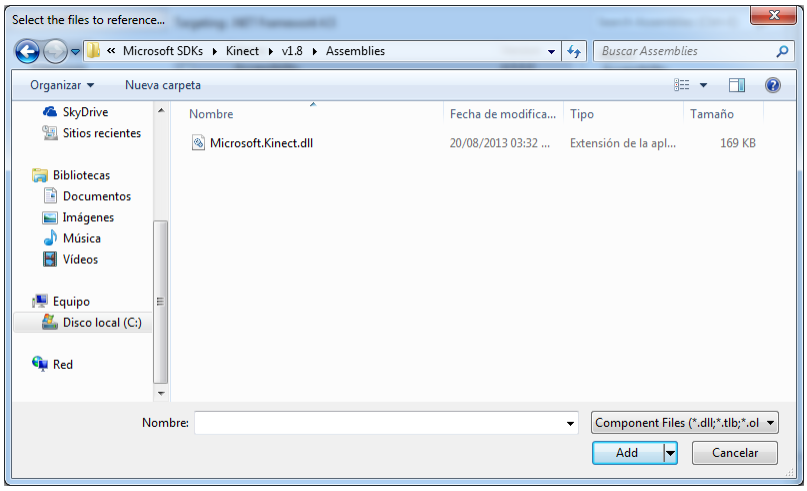


Figura 3.12: Agregando referencia (parte 3).

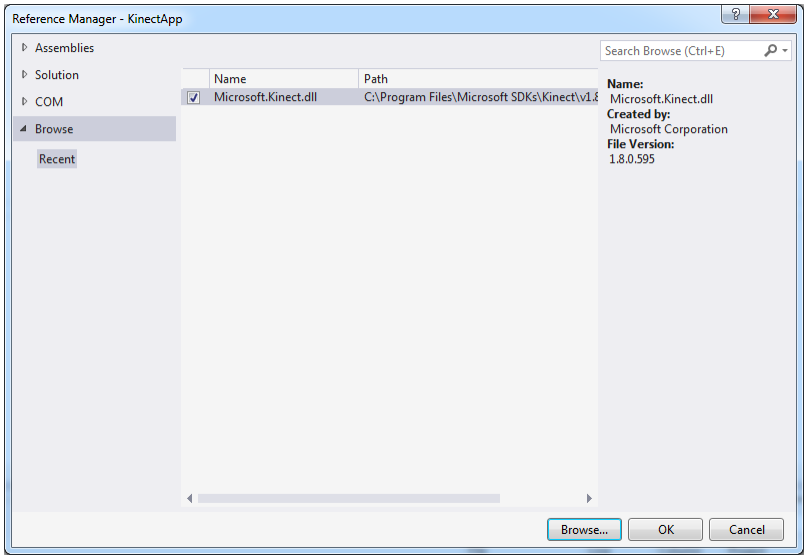


Figura 3.13: Agregando referencia (parte 4).

Ahora es posible empezar a desarrollar una aplicación que haga uso del sensor Kinect. En la pantalla principal de Visual Studio se podrán ver abiertos dos archivos en dos pestañas diferentes. El primero es el MainWindow.xaml, en este se diseña la interfaz gráfica de usuario agregando botones, cuadros de texto, etc., todo esto mediante la opción de arrastar botones o agregándolos directamente mediante código XAML (eXtensible Application Markup Language) que es una versión modificada por Microsoft de XML (eXtensible Markup Language). En la interfaz, por ejemplo, es necesario crear un elemento tipo Imagen (Image) que es dónde se mostrará la imagen a color, de profundidad

o de esqueleto de usuario.

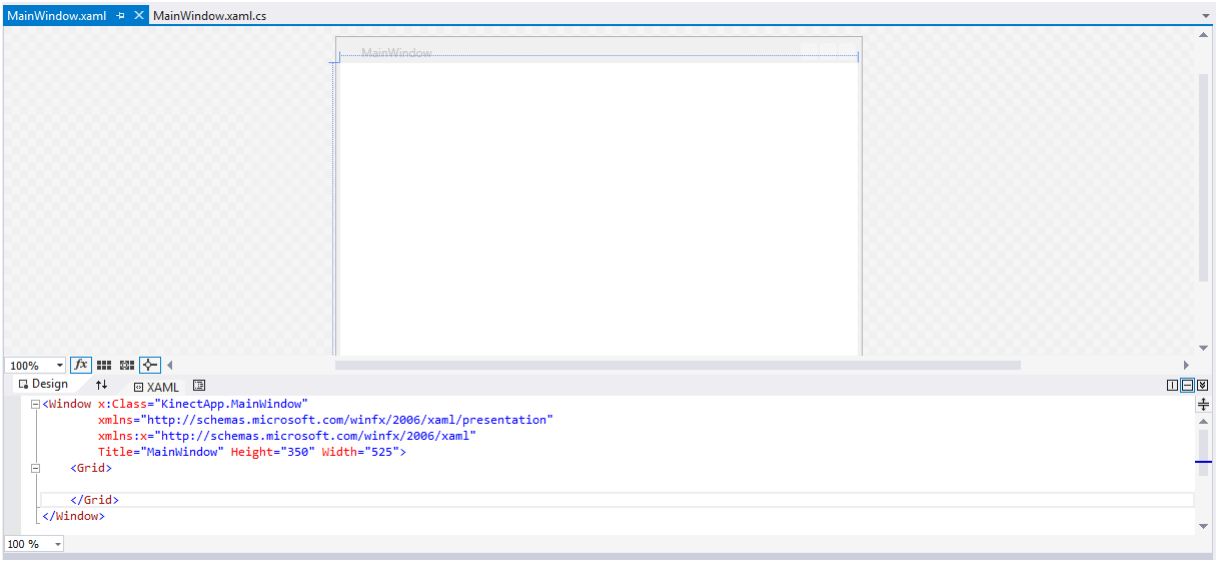


Figura 3.14: Archivo MainWindow.xaml en Visual Studio.

Junto al archivo MainWindow.xaml se encuentra el archivo MainWindow.xaml.cs que es donde a modo de código se programaran todas las funciones internas que utiliza nuestro programa. Esto sólo es el esqueleto básico de una aplicación general de Visual Studio C#. Para trabajar con el sensor Kinect es necesario agregar la linea de código mostrada en la Fig. 3.15.

```
1  using Microsoft.Kinect;
```

Figura 3.15: Dependencia elemental del Kinect para Visual Studio.

Como miscelanea hay varios puntos a tomar en consideración. Se debe declarar una variable tipo sensor y habilitar el flujo de imágenes a color, profundidad o esqueleto en el formato apropiado (RGB, YUV, formato crudo, etc.) eligiendo tambien la resolución (como 320×240, 640×480) deseada. Ademas se debe crear un evento que maneje el flujo de imágenes que entrega el sensor y finalmente activar el sensor. El manejador de eventos debe lidiar con la obtención de 30 imágenes por segundo en nuestro caso, así como convertir la información proporcionada por el sensor en una imagen tipo BMP de 8, 16 o 32 bits de profundidad y desplegarla en la ventana Image. Finalmente, al cerrar la aplicación, es necesario detener el sensor de manera segura. Con esto se evita que el programa se trabe o se haga lento en algún punto. Un ejemplo completo se muestra en los anexos.

3.3. Implementación de filtros estadísticos y morfológicos

Tanto la implementación de los filtros estadísticos como morfológicos se puede realizar de manera numérica muy sencilla utilizando para ello un arreglo de ordenamiento burbuja. El pseudocódigo de dicho ordenamiento de muestra en la Fig. 3.16. Dependiendo del tamaño de kernel seleccionado se incrementará el numero de elementos a ordenar, por ejemplo, un filtro de kernel tamaño 1, tiene 3 elementos por lado con lo cual se requiere ordenar 9 elementos, uno de tamaño 2, tiene 5 elementos por lado, con lo que ahora se tienen que ordenar 25 elementos y así sucesivamente.

```
procedimiento DeLaBurbuja (a_0, a_1, a_2, ..., a_(n-1))
  para i ← 1 hasta n hacer
    para j ← 0 hasta n − i hacer
      si a_(j) > a_(j+1) entonces
        aux ← a_(j)
        a_(j) ← a_(j+1)
        a_(j+1) ← aux
      fin si
    fin para
  fin para
fin procedimiento
```

Figura 3.16: Pseudocódigo para el arreglo burbuja.

En la Fig.3.17 se muestra un diagrama a bloques sobre el proceso de implementación de los diferentes filtros. En un inicio se debe seleccionar el tamaño de filtro o kernel esto determina el numero de elementos a entrar al ordenamiento burbuja. A continuación se debe seleccionar el tipo de filtro a implementar ya sea mediana, apertura o cerradura. De los elementos que entran al arreglo burbuja se debe tomar el valor central del arreglo burbuja para un filtrado tipo mediana, el valor máximo y luego el mínimo para el de apertura o el valor mínimo y luego el máximo para el de cerradura. Para el filtro de media solo basta obtener el promedio de los elementos y reemplazar el píxel central de la ventana por ese valor calculado.

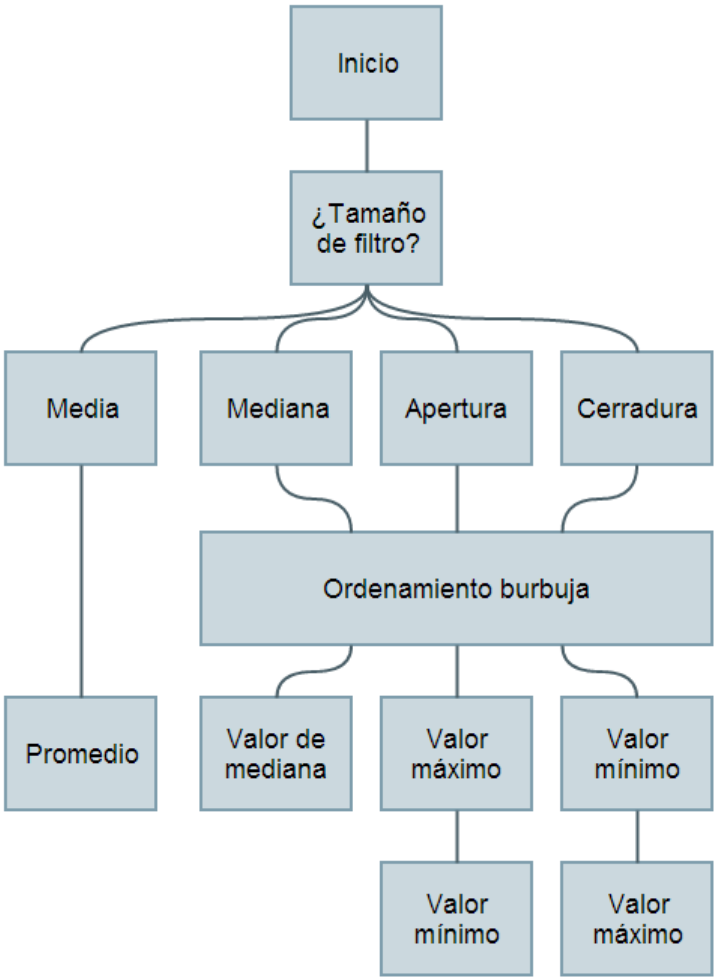
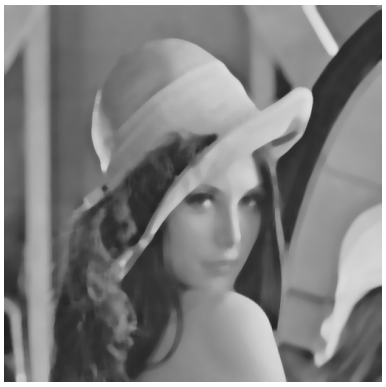


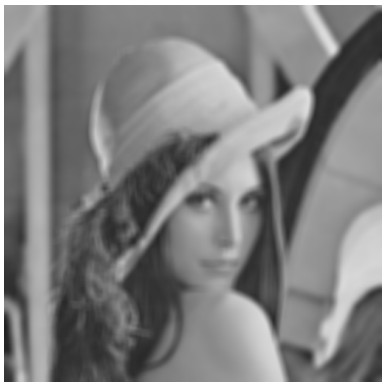
Figura 3.17: Proceso de fitrado.

Para probar la implementación tanto de los filtros estadísticos como morfológicos se utilizó la clásica imagen de Lena en escala de grises, ver Fig. 3.18. Los resultados se muestran en las Figs. 3.19 y 3.20.

Figura 3.18: Imagen de Lena en escala de grises.



(a) Mediana kernel 11x11.



(b) Media kernel 11x11.

Figura 3.19: Filtros etadísticos kernel 11x11.

(a) Cerradura kernel 11x11.

(b) Apertura kernel 11x11.

Figura 3.20: Filtros morfológicos kernel 11x11.

3.4. Análisis del proceso de homografía

El proceso de homografía funciona con mínimo cuatro pares de puntos. Antes de aplicar el algoritmo en las imágenes proporcionadas por el sensor Kinect es necesario probarlo en ejemplos mas sencillos a fin de verificar su funcionamiento. Como ejemplo una vez mas se usó la clásica imagen de Lena esta vez en dos tamaños, 512x512 y 256x256, la primera como imagen de entrada y la segunda como salida. La Fig. 3.21 muestra la imagen en ambas resoluciones con los puntos de homografía seleccionados para esta primer prueba.

Figura 3.21: Puntos de homografía seleccionados para la imagen de Lena.

Los puntos seleccionados se muestran en la Tab. 3.1.

lena (512x512)	**lena(256x256)**
415, 115	208, 57
266, 349	133, 176
60, 360	30, 180
382, 512	191, 255

Tabla 3.1: Puntos de entrada y salida homografía Lena.

La matriz de homografía resultante se muestra en la Ec. 3.1.

$$H = \begin{bmatrix} 0{,}0334 & 0{,}0022 & -0{,}7906 \\ 0{,}0006 & 0{,}0348 & -0{,}6072 \\ 0{,}0000 & 0{,}0000 & 0{,}0624 \end{bmatrix} \tag{3.1}$$

Para comprobar el efecto de la matriz de homografía calculada se propone mapear ahora dos puntos, que serán los ojos de lena, primero el ojo derecho de la imagen de Lena con el tamaño de 512x512 cuya posición (264,266) se convierte en la coordenada (134,134) aplicando la matriz de homografía, se ve claramente en la Fig. 3.22 que corresponde fielmente al ojo derecho de Lena en la imagen de resolución mas pequeña.

Figura 3.22: Ejemplo de homografía ojo derecho.

Se hace lo mismo ahora para el ojo izquierdo de Lena con posición (327,365) que con la matriz de homografía se mapea al punto (163,134). Al igual que en el ejemplo anterior el punto corresponde exactamente al ojo izquierdo de Lena en la imagen con resolución de 256x256 (Fig. 3.23).

Figura 3.23: Ejemplo de homografía ojo izquierdo.

3.5. Aplicación de falso color

La información de profundidad proporcionada por el sensor Kinect viene en paquetes de 640x480 donde cada elemento tiene una longitud de 12 bits. Una forma sencilla de representar dicha información es escalando a 8 bits para mostrarla como una imagen en escala de grises. Si bien la variación del nivel de intensidad de gris es suficiente de acuerdo a los atributos visuales para dar el efecto de profundidad, resulta mas intuitivo desplegarla mediante la utilización de un falso color.

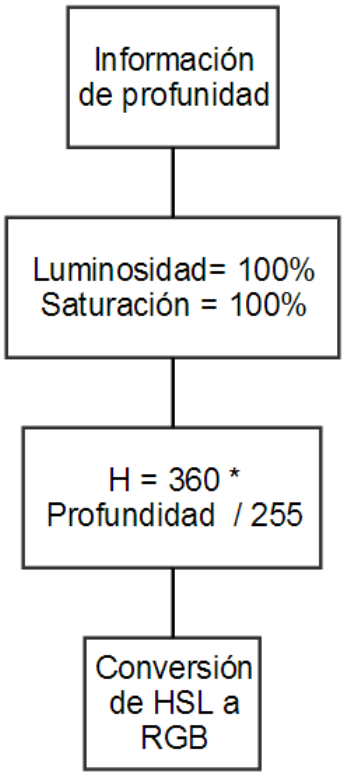


Figura 3.24: Proceso para la asignación de falso color.

De manera general y como se muestra en diagrama a bloques de la Fig. 3.24, como entrada se tiene la información de profundidad o incluso los niveles de intensidad de gris, la saturación y la luminosidad se establecen ambos al 100 %. A cada valor de nivel de intensidad se le asigna una valor de tono entre 0°y 360°multiplicándolo por 360 y dividiendo entre 255. Los tres valores H, S, L con transformados entonces al espacio de color RGB para desplegar.

(a) Lena en escala de grises. (b) Imagen de Lena con falso color.

Figura 3.25: Aplicación de falso color.

Una vez tomamos a Lena para aplicarle el algoritmo propuesto de falso color, Fig. 3.25. El efecto que se busca al aplicar este procesamiento a la información de profundidad que entrega el sensor es una variación de color tenue conforme los objetos se alejan de la cámara, precisamente para dar un efecto de profundidad mediante este atributo visual cuya escala correspondería a la Fig. 3.26.

Figura 3.26: Escala de color HSL.

Capítulo 4

Resultados

Éste capítulo describe los resultados obtenidos a partir de la metodología descrita en el capítulo anterior. Aquí se muestra el resultado final de las interfaces de usuario desarrolladas así como el de los pre-procesamientos utilizando los filtros estadísticos y morfológicos comparándolos directamente. También se muestra el proceso de homografía implementado sobre las imágenes de color y de profundidad y la matriz de calibración resultante. A continuación se muestran ejemplos de segmentación por distancias y color comparándolos contra métodos tradicionales. Finalmente se muestra un caso de estudio particular en que utilizó la función de detección de articulaciones con que cuenta el sensor como auxiliar de los expertos del área de rehabilitación de la universidad.

4.1. Interfaz de usuario

Se diseñaron dos interfaces de usuario, la primera desarrollada en Visual C# que permite conectarse con el sensor Kinect, leer la imagen color y de profundidad del dispositivo y almacenarlas (Fig. 4.1). La imagen color es guardada en formato PNG (Portable Network Graphics, por sus siglas en inglés) con una resolución de 640x480. En el caso de la información de profundidad esta es almacenada de dos formas, en primer lugar como un archivo de texto conteniendo la información de profundidad en crudo en 16 bits (de los cuales 12 son de utilidad), formato que no es de fácil

representación por medio de una imagen, además del archivo de texto la información de profundidad es almacenada como una imagen RGB en escala de grises de 32 bits de profundidad o 8 bits por canal en formato PNG, esto significa que la información ha sido escalada de 12 a 8 bits para poder ser representada por medio de una imagen. Mediante la misma interfaz de usuario es posible activar o desactivar el modo de corto alcance de medición de profundidad (near mode), así como la forma en la imagen de 8 bits es desplegada (Fig. 4.2).

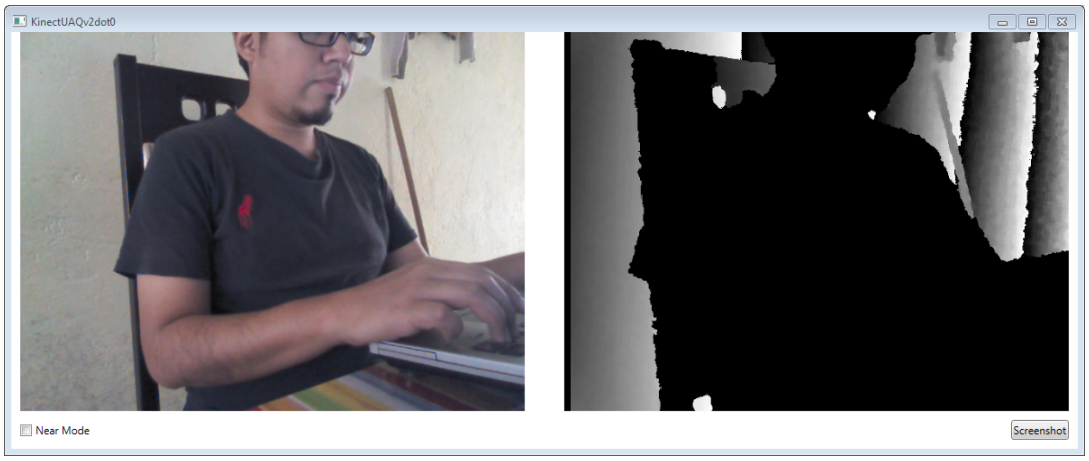


Figura 4.1: Interfaz con modo de profundidad normal activado.

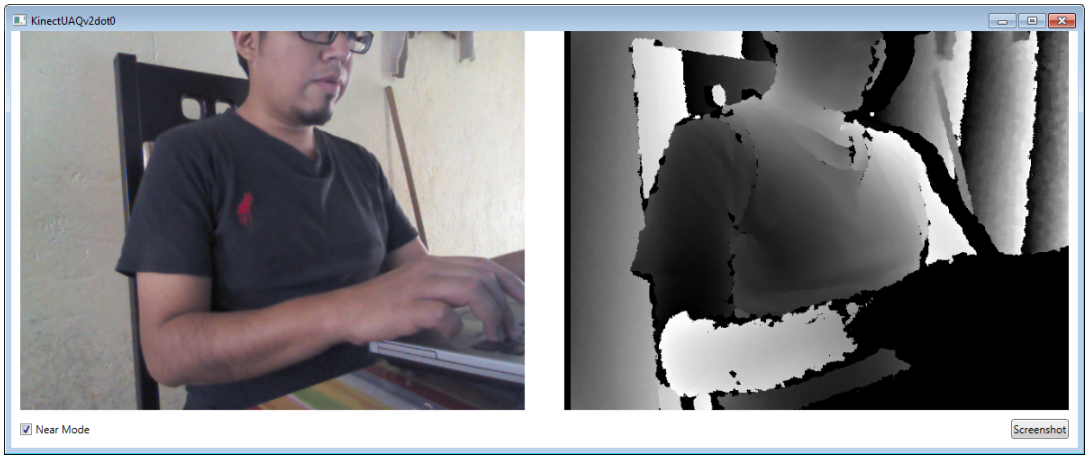


Figura 4.2: Interfaz con modo de profundidad cercano activado.

La segunda interfaz está desarrollada en C++ con ayuda de Gtk y linkeada a la librería de imágenes OpenCV que en este caso se usa solamente para cargar la imagen. En esta otra interfaz,

que es más elaborada, se agregan los diferentes procesamientos propuestos para este trabajo de tesis (Fig. 4.3).

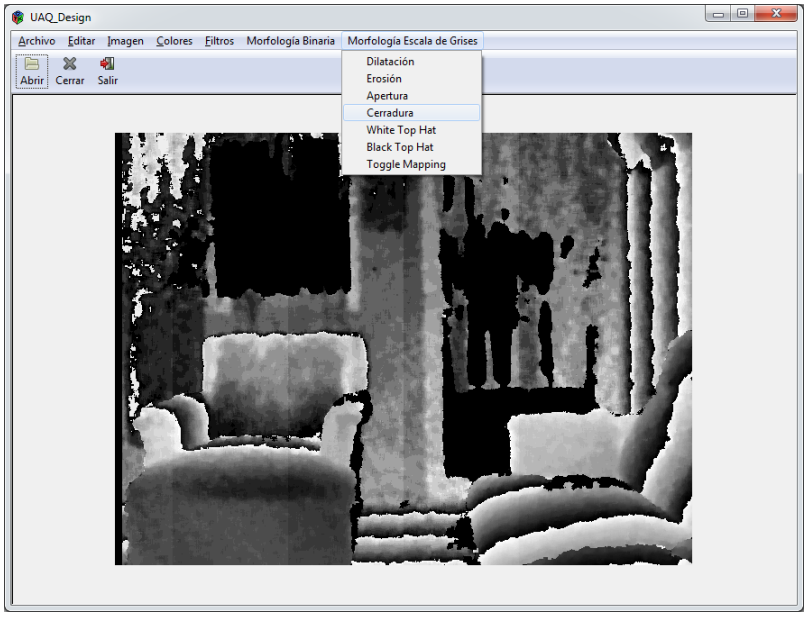


Figura 4.3: Interfaz en C++ para aplicar procesamientos.

Se desarrollaron dos interfaces principalmente para facilitar el crecimiento modular de la misma en cuanto a la adición de mas herramientas de procesamiento de imágenes dado que se tiene mas experiencia programando en C++ como ocurre con la segunda aplicación. La primer interfaz por su parte permite entender de mejor manera como adquirir la información de color y de profundidad del sensor. A futuro se deben combinar ambas aplicaciones para que funcionen de manera conjunta.

4.2. Pre-procesamiento de imágenes de profundidad

La información de profundidad proporcionada por el sensor Kinect tiene problemas de ruido y huecos que la hacen altamente susceptible de mejora. A continuación se muestra una imagen de profundidad adquirida por medio del sensor Kinect donde se observa de manera muy notoria la presencia de ruido y huecos así como una pobre definición de los objetos que contiene (Fig. 4.4). La escena corresponde a un usuario en la sala de su casa con un brazo extendido a 1.5 metros del

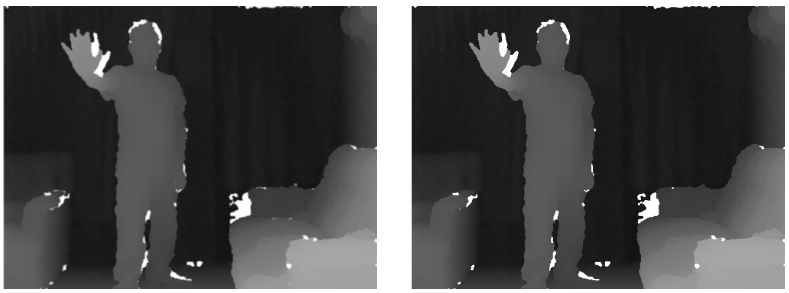
sensor.

Figura 4.4: Imagen de profundidad con ruido.

Las regiones blancas pueden significar zonas fuera del rango del sensor o áreas donde la información de profundidad no esta definida de manera correcta. Claramente el usuario al ubicarse a 1.5 metros del sensor está dentro del rango de medición de éste, de modo que las zonas blancas, en este caso, se relacionan mas bien a ruido debido a la tecnología de censado que usa el Kinect.

4.2.1. Filtros estadísticos (media y mediana)

Los filtros estadísticos son muy populares por su fácil implementación y en general por los buenos resultados que llegan a ofrecer. Las Figs. 4.5, 4.6, 4.7, 4.8 y 4.9 muestran la aplicación de los filtros de media y mediana variando el tamaño del kernel del filtro desde 3 hasta 11, siguiendo la formula 2n - 1 ya que el filtro sólo puede ser de tamaño impar.



(a) Media kernel 3x3. (b) Mediana kernel 3x3.

Figura 4.5: Filtros etadísticos kernel 3x3.

El resultado del filtro de media se muestra en los incisos a) de las Figs. 4.5, 4.6, 4.7, 4.8 y 4.9, éste entrega un característico efecto de borroneado o desenfoque que se hace más notorio conforme el tamaño del Kernel se incrementa. Este filtro en vez de atacar el problema de los huecos y ruido de la imagen hace que se pierda definición de los diferentes elementos esta.

(a) Media kernel 5x5.

(b) Mediana kernel 5x5.

Figura 4.6: Filtros etadísticos kernel 5x5.

(a) Media kernel 7x7.

(b) Mediana kernel 7x7.

Figura 4.7: Filtros etadísticos kernel 7x7.

Los incisos b) de las Figs. 4.5, 4.6, 4.7, 4.8 y 4.9 muestran por su parte el efecto del filtro de mediana con kernels desde 3 hasta 11. Este filtro ataca el problema de ruido y huecos de una manera muy ligera por lo que se hace evidente la necesidad de aplicar filtros de mediana con kernels de mayor tamaño.

(a) Media kernel 9x9.

(b) Mediana kernel 9x9.

Figura 4.8: Filtros etadísticos kernel 9x9.

(a) Media kernel 11x11.

(b) Mediana kernel 11x11.

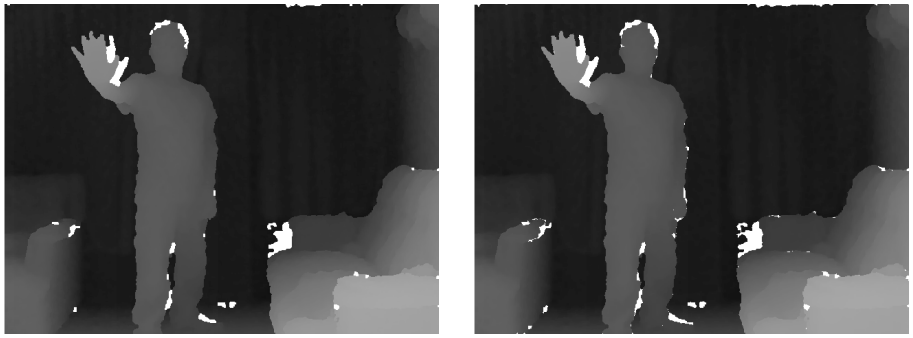
Figura 4.9: Filtros etadísticos kernel 11x11.

Finalmente y en base a pruebas variando el tamaño del kernel, se logra eliminar completamente las regiones blancas al menos alrededor del usuario usando un kernel tamaño 27x27, Fig. 4.10. En general se logra definir de una mejor forma la silueta del usuario y de los objetos alrededor de él, sin embargo, esto no ocurre en los detalles finos del usuario principalmente los dedos de las manos y el espacio entre las piernas, de los cuales se pierde total detalle.

Figura 4.10: Mediana kernel 27x27.

4.2.2. Filtros morfológicos (apertura y cerradura)

De la misma manera se aplicaron los filtros morfológicos a la Fig. 4.4 variando el tamaño del elemento estructurante del filtro, los resultados de filtrado de un tamaño de EE 1 hasta 5 se muestran en las Figs. 4.11, 4.12, 4.13, 4.14, 4.15 incisos a) y b).



(a) Apertura EE 3x3. (b) Cerradura EE 3x3.

Figura 4.11: Filtros morfológicos EE 3x3.

De manera particular los resultados utilizando el filtro de apertura se muestran en los incisos a) de las Figs. 4.11, 4.12, 4.13, 4.14, 4.15. Los resultados son buenos y se observa como gradualmente se van reduciendo los huecos blancos de la imagen, hasta que un filtro tamaño 11x11 da resultados equiparables a los del filtrado de media con kernel tamaño 27x27.

(a) Apertura EE 5x5.

(b) Cerradura EE 5x5.

Figura 4.12: Filtros morfológicos EE 5x5.

(a) Apertura EE 7x7.

(b) Cerradura EE 7x7.

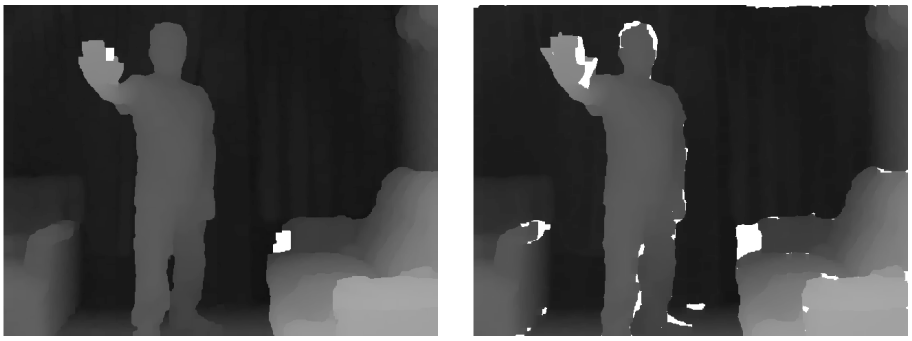
Figura 4.13: Filtros morfológicos EE 7x7.

El filtrado por cerradura no da buenos resultados, Figs. 4.11, 4.12, 4.13, 4.14, 4.15 incisos a). La modificación es mínima y ni siquiera equiparable al filtrado con una apertura del mismo tamaño.

(a) Apertura EE 9x9.

(b) Cerradura EE 9x9.

Figura 4.14: Filtros morfológicos EE 9x9.



(a) Apertura EE 11x11. (b) Cerradura EE 11x11.

Figura 4.15: Filtros morfológicos EE 11x11.

Un filtro morfológico de apertura tamaño 13x13, de hecho, remueve casí por completo las regiones blancas de la imagen, este resultado se muestra en la Fig. 4.16.

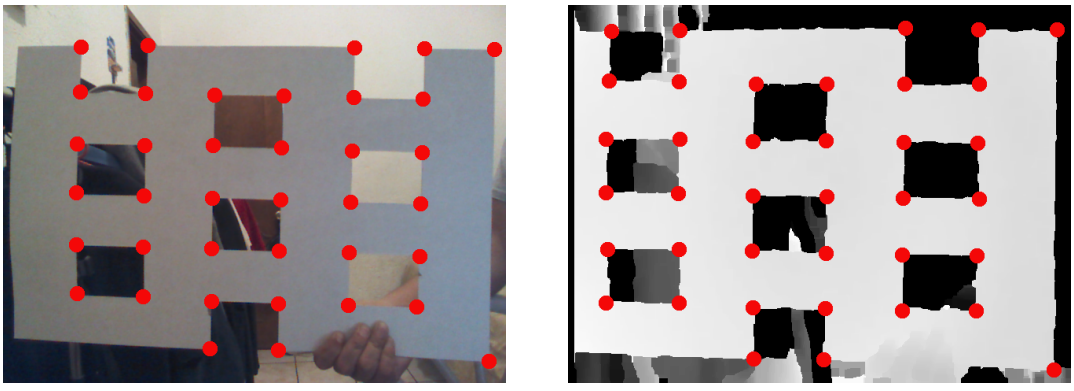
Figura 4.16: Apertura EE 13x13.

Un aspecto bastante interesante de la morfología es la dualidad entre sus operaciones, en vista de los resultados entregado por la apertura sobre la Fig. 4.4 y que la cerradura en realidad no tuvo efecto sobre ella, se aplica un negativo a dicha imagen y se aplica el filtro de cerradura tamaño 13x13 y se obtiene un resultado análogo Fig. 4.17.

Figura 4.17: Cerradura EE 13x13.

4.3. Proceso de homografía

Una vez probada la implementación de este algoritmo en el capitu anterior, es necesario aplicarlo a las imágenes que entrega el Kinect. Para eso se diseño un pequeño tablero tipo ajedrez, con algunas casillas perforadas de modo que fueran visibles en la imagen de profundidad. Tanto la imagen color como la de profundidad se pueden observar en la Fig. 4.18. Claramente se ve como el tablero diseñado se observa por completo en la imagen color mientras que en la de profundidad se ve entrecortado, esto se debe a los diferentes campos de visión que tienen la cámara de vídeo y el sensor infrarrojo.



(a) Imagen color. (b) Mapa profundidad.

Figura 4.18: Puntos para homografía.

Los puntos seleccionados para la homografía se muestran en la Tabla 4.1.

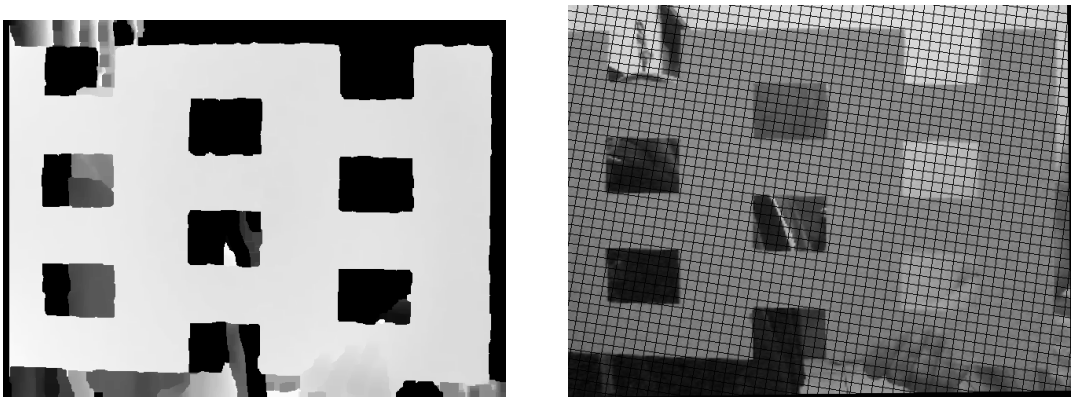
Imagen color	**Imagen profundidad**	**Imagen color**	**Imagen profundidad**
99, 53	55, 33	266, 245	235, 242
184, 51	142, 32	353, 247	329, 243
447, 54	429, 29	442, 250	426, 245
536, 55	522, 31	531, 252	523, 246
625, 56	622, 31	93, 304	50, 310
98, 110	52, 96	178, 307	141, 310
181, 112	141, 97	265, 309	234, 312
269, 114	239, 100	351, 312	328, 315
357, 115	329, 100	440, 314	425, 317
446, 117	428, 100	530, 319	520, 318
534, 119	523, 100	94, 366	48, 378
94, 176	48, 170	178, 370	141, 378
180, 178	142, 170	265, 376	235, 384
267, 178	235, 173	350, 379	327, 385
354, 181	328, 172	439, 381	425, 388
444, 185	427, 174	526, 383	519, 388
533, 187	521, 175	263, 436	236, 449
93, 238	48, 238	350, 438	325, 450
179, 242	140, 239	618, 542	618, 463

Tabla 4.1: Puntos de entrada y salida homografía imágenes Kinect.

El algoritmo de Homografía entrega la matriz mostrada en la Ec. 4.1.

$$H = \begin{bmatrix} -1{,}77974101e-02 & -2{,}53534883e-04 & 9{,}32279664e-01 \\ 2{,}24044932e-04 & -1{,}79672532e-02 & 3{,}60479749e-01 \\ -3{,}00533852e-08 & -1{,}28984197e-07 & -1{,}64102173e-02 \end{bmatrix} \tag{4.1}$$

Al proyectar los puntos de la imagen de color a los de la imagen de profundidad con sus respectivos niveles de intensidad como se ven en la Fig. 4.19, observamos una mejor correspondencia entre ambas imágenes.



(a) Imagen profundidad. (b) Nueva imagen color.

Figura 4.19: Resultado de homografía.

4.4. Segmentación por color y distancia

La información de profundidad del sensor Kinect esta contenida en un arreglo de 640x480 pixeles, cada uno con 12 bits de información relevante, lo cual para sistemas automáticos y digitales, es suficiente para operar, Fig. 4.20.

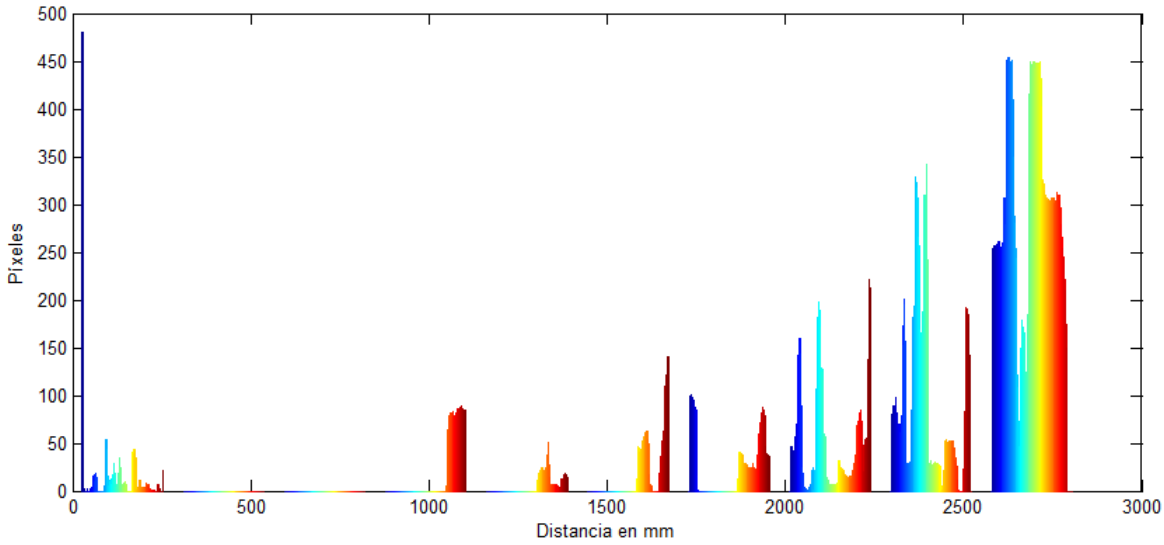


Figura 4.20: Histograma de distancias.

La representación de dicha información en una imagen en escala de grises da una mejor apre-

ciación al usuario final de como se distribuyen los diferentes elementos de la imagen mediante un efecto de profundidad sutil. La Fig.4.21 muestra, por ejemplo, la representación de la imagen de profundidad tomando en cuenta solamente los 8 bits menos significativos de información de profundidad de los 12 que entrega el sensor. Debido a esto es notorio que los niveles de intensidad de gris se repiten dando un efecto de bandas.

Figura 4.21: Información de profundidad (8 bits menos significativos).

Las Figs. 4.22a y 4.22b representan por su parte el ajuste del rango de medición del sensor Kinect a lo largo de la escala de grises desde el nivel 0 hasta el nivel 255. Para la primer imagen, el nivel 0 se asocia a lo mas cercano o distancia 0 mm al sensor, hasta 255 que sería para los píxeles cuya profundidad sea la mayor al sensor. En el caso de la segunda imagen ocurre completamente lo contrario

(a) Información de profundidad en el rango de 0 a 255.

(b) Información de profundidad en el rango de 255 a 0.

Figura 4.22: Representación de profundidad por niveles de gris.

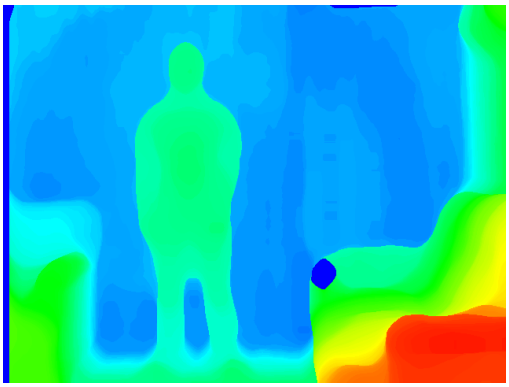
A pesar de que en su caso las tres imágenes anteriores (Figs. 4.21, 4.22a y 4.22b) resultan de mucha utilidad para lograr el efecto de profundidad, la variación tan brusca de la primer imagen y la apenas visible variación en la segunda y tercera, hacen que sea necesaria otra forma de representar el efecto de profundidad, esto se hace mas notorio después de procesar las imágenes Figs. 4.23a y 4.23b. La mejor opción es la aplicación de un falso color a la imagen cuidando que dicha variación sea lo suficientemente útil al usuario final. El problema es que el espacio de color RGB, como se vio en el Capítulo 2, al usar 8 bits por canal puede representar hasta 16, 777, 216 colores, con lo cual lograr variaciones sutiles o significativas de color se dificulta. Es ahí donde entra en juego el uso de los espacios de color HS, ya que el parámetro hue (H) o tono, varia de 0 a 360 grados, con lo que es posible lograr una variación de color más uniforme. Las Figs. 4.24a y 4.24b muestran el resultado de aplicar un algoritmo de falso color a las imágenes procesadas mediante el filtro de mediana y de apertura.

(a) Imagen procesada con filtro de mediana.

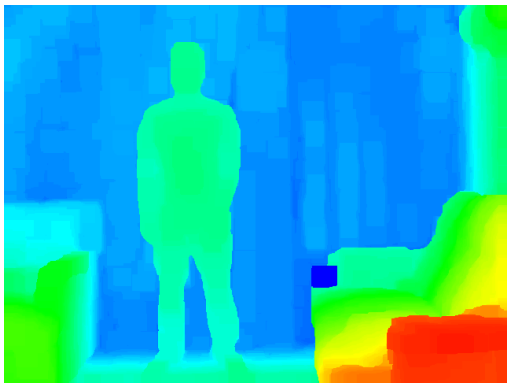
(b) Imagen procesada con filtro de apertura.

Figura 4.23: Imagenes de profundidad procesadas en escala de grises.

Sea cual sea la forma de representar la información de profundidad que entrega el sensor, es importante conservar, y en su caso procesar a la par, los datos en crudo que nos proporciona en 12 bits, de este modo aseguramos no perder la sensibilidad del dispositivo al hacer el ajuste a 8 bits.



(a) Imagen procesada con filtro de mediana representada en falso color.



(b) Imagen procesada con filtro de apertura representada en falso color.

Figura 4.24: Representación de profundidad por falso color.

A continuación con la imagen procesada es posible segmentar por distancia utilizando la información de profundidad y representando por medio de una imagen en escala de grises o binaria el resultado, esto con ayuda de un umbralizado con dos niveles de umbral, no por nivel de gris, sino por distancias en milímetros. Un ejemplo de umbralización por distancia con umbrales 2200 y 2400 mm se muestra en las Figs. 4.25a y 4.25b.

(a) Imagen segmentada con objetos de interés en grises.

(b) Imagen segmentada con objetos de interés en blanco.

Figura 4.25: Segmentación de usuario por distancia.

4.5. Segmentación por distancia contra métodos tradicionales

Como se pudo ver en el capitulo 2 de esta tesis una de las herramientas de segmentación mas utilizadas es la de umbralización. En ella, basicamente, mediante un análisis del histograma de la imágen, generalmente en escala de grises, se establecen los niveles de umbral a partir de los cuales se mandan a 0 o a 255 los tonos. De entrada, una de las principales problematicas de este método es que se deben tener ciertas consideraciones a la hora de capturar la imagen como cuidar la iluminación y que hasta cierto punto haya cierto contraste entre los objetos que se quieren segmentar. Las Figs. 4.26 y 4.27 ponen de manifiesto esto en la tarea segmentar los dedos de la mano.

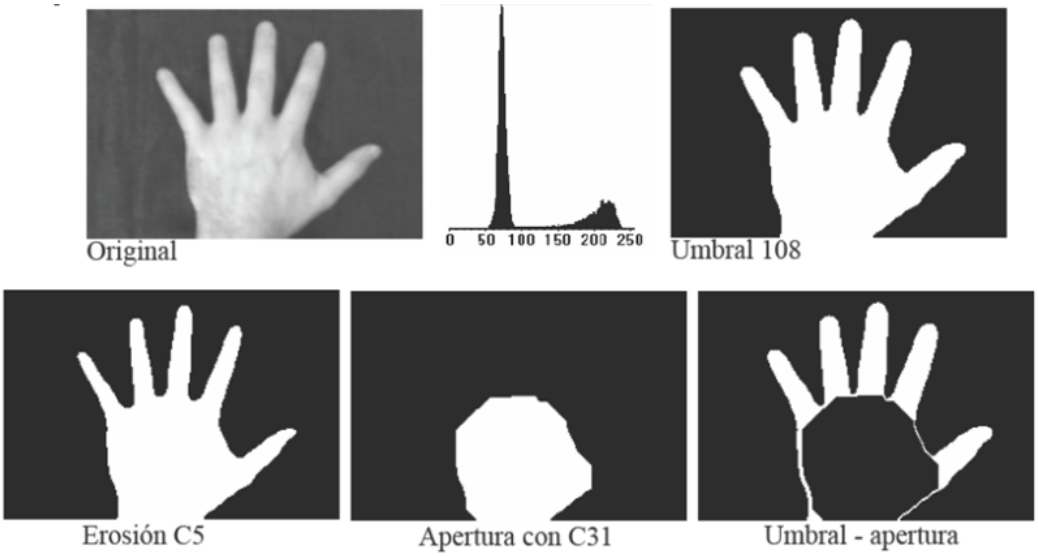


Figura 4.26: Segmentación de una mano es escala de grises.

Teniendo la imagen es escala de grises se observa claramente que la mano contrasta del fondo. El análisis de histograma hace esto mas evidente al observarse dos picos de histograma a diferentes niveles de intensidad. Los mas oscuros correspondiendo al fondo de la imagen, mientras que los mas claros pertenecen a la mano de la persona. La umbralización con un nivel 108 manda el fondo a un color ngro y la mano a un tono blanco. Mediante varias pasadas con herramientas morfológicas es posible eliminar la palma de la mano y dejar solamente los dedos del usuario. En total se contabilizan 9 pasos para segmentar los dedos.

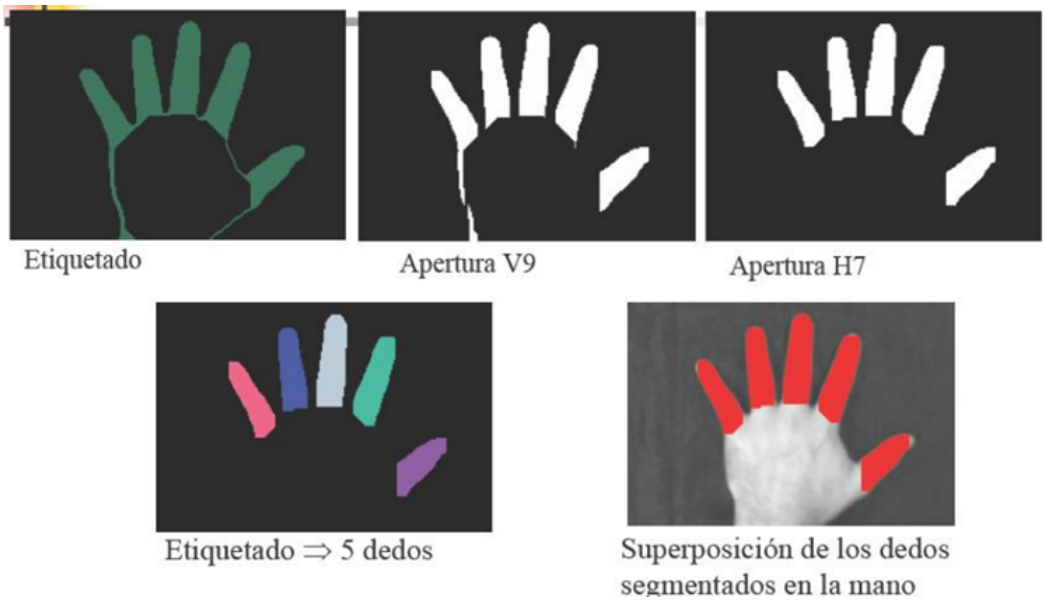


Figura 4.27: Segmentación de una mano es escala de grises (cont.).

Por su parte, una de las grandes ventajas del sensor Kinect, aparte de su bajo costo, comparado contra otros dispositivos que permiten estimar la profundidad de una escena, es la de contar con una cámara color. En la sección anterior se resolvió el problema de la estimación de la matriz de homografía para empatar ambas informaciones. Con esto en mente y a la par de segmentar la información de profundidad y representarla por medio de una imagen en escala de grises o en falso color, resulta mas intuitivo para el usuario saber que regiones o objetos están a una distancia determinada tal y como él los percibe. Tómese como ejemplo la Fig. 4.28a, en ella se muestra el brazo de un usuario, dicha imagen color no nos dice en ningún momento como es que los elementos de ella están ubicados con respecto a la cámara. Por su parte la Fig. 4.28b muestra lo que ve el sensor infrarrojo del sensor Kinect mejorada por medio de un filtro morfológico de cerradura tamaño 5x5. El ojo experimentado notara las sutiles variaciones de nivel de gris en ella para percibir el efecto de profundidad, pero aún así esto no es algo revelador y significante.

(a) Imagen color de brazo de usuario.

(b) Imagen de profundidad de brazo de usuario.

Figura 4.28: Segmentación de mano y brazo por distancia.

Segmentar por profundidad la Fig. 4.28b con un umbral de distancia de entre 450 y 650 mm, permite segmentar la mano del usuario y al empatar dicha información mediante la matriz de homografía, es posible sobreponer dicha información sobre la imagen colo como se muestra en la Fig. 4.29a. Mas aún si lo que se requiere es segmentar el brazo completo del usuario se deben establecer como niveles de umbral de distancia 450 y 850 mm para obtener los resultados mostrados en la Fig. 4.29b

(a) Imagen color con mano segmentada.

(b) Imagen color con brazo segmentado.

Figura 4.29: Segmentación de mano y brazo por distancia (cont.).

4.6. Caso de estudio: Análisis de postura mediante el sensor Kinect

De manera específica y a la par de trabajar con la parte de segmentación de imágenes por medio de atributos visuales y de profundidad, se desarrolló un sistema automático de análisis postural con la asesoría de la Facultad de Enfermería de la Universidad, que hace uso del sensor Kinect y su función de detección de articulaciones del cuerpo humano, Fig. 4.30. El primer paso consistió en un análisis de repetibilidad de medición del sensor que se presento en el CIINDET 2014. En ese trabajo se realizaron 5 mediciones entre elementos articulares de un individuo joven a fin de detectar diferencias de distancias entre estos, se pudo determinar que el sensor llega a tener una desviación estándar de entre 8.6 y 8.9 mm. Además se resalto la ventaja de almacenar la información tanto de color, como de profundidad así como las coordenadas de las articulaciones entregadas por el sensor para análisis posteriores. El extenso el trabajo mencionado se puede revisar en uno de los Anexos de este trabajo.

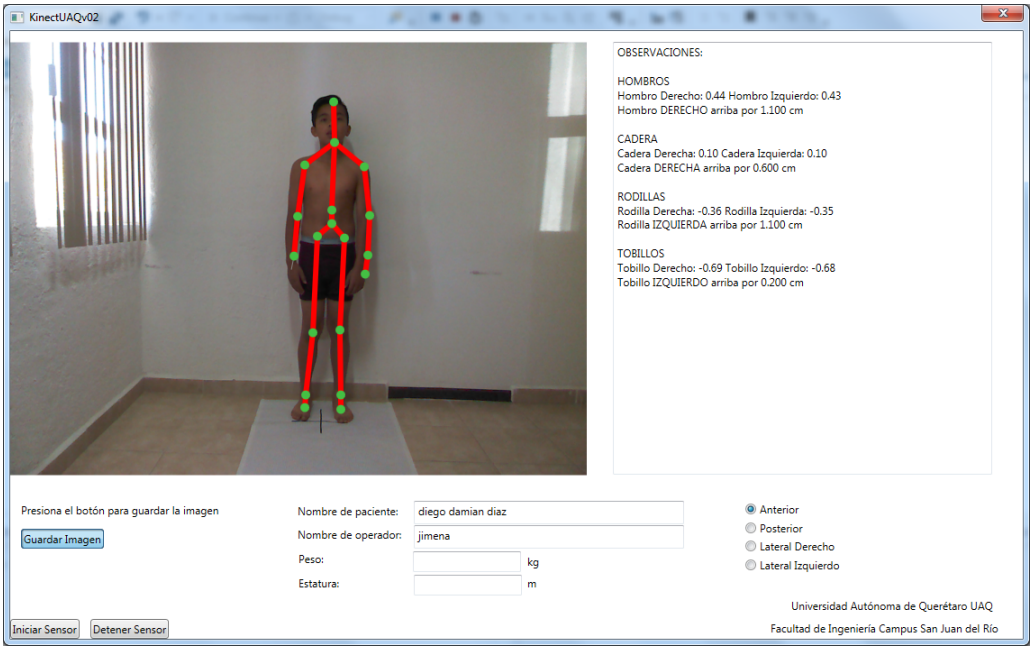


Figura 4.30: Interfaz de usuario Postural Kinect v0.4.

Uno de los principales problemas a los que se enfrentan los especialistas del área de fisioterapia

es la gran incertidumbre que existe en los análisis de postura por método tradicional ya que es meramente observacional y depende en gran medida de la experiencia del especialista. Para ello se basan en el Formato de Observación Sistemática de la Alineación Corporal (FOSAC, Fig. 4.31). En dicho formato el especialista analiza los perfiles posterior, derecho, izquierdo y anterior, cada perfil tiene entre 14 a 18 puntos que debe observar, con lo cual dicho análisis se hace tedioso y en el cual es difícil garantizar que el paciente conserve o mantenga la misma postura siempre. La opción de los software comerciales que procesan imagen existentes en el mercado no mejoran mucho la situación, por un lado se tiene su alto costo y por otro se ha probado que muchos de ellos detectan problemas posturales en el 100 de las muestran que analizan, ademas de que requieren de la colocación de puntos de referencia en el paciente con lo cual el tiempo de la prueba se incrementa.

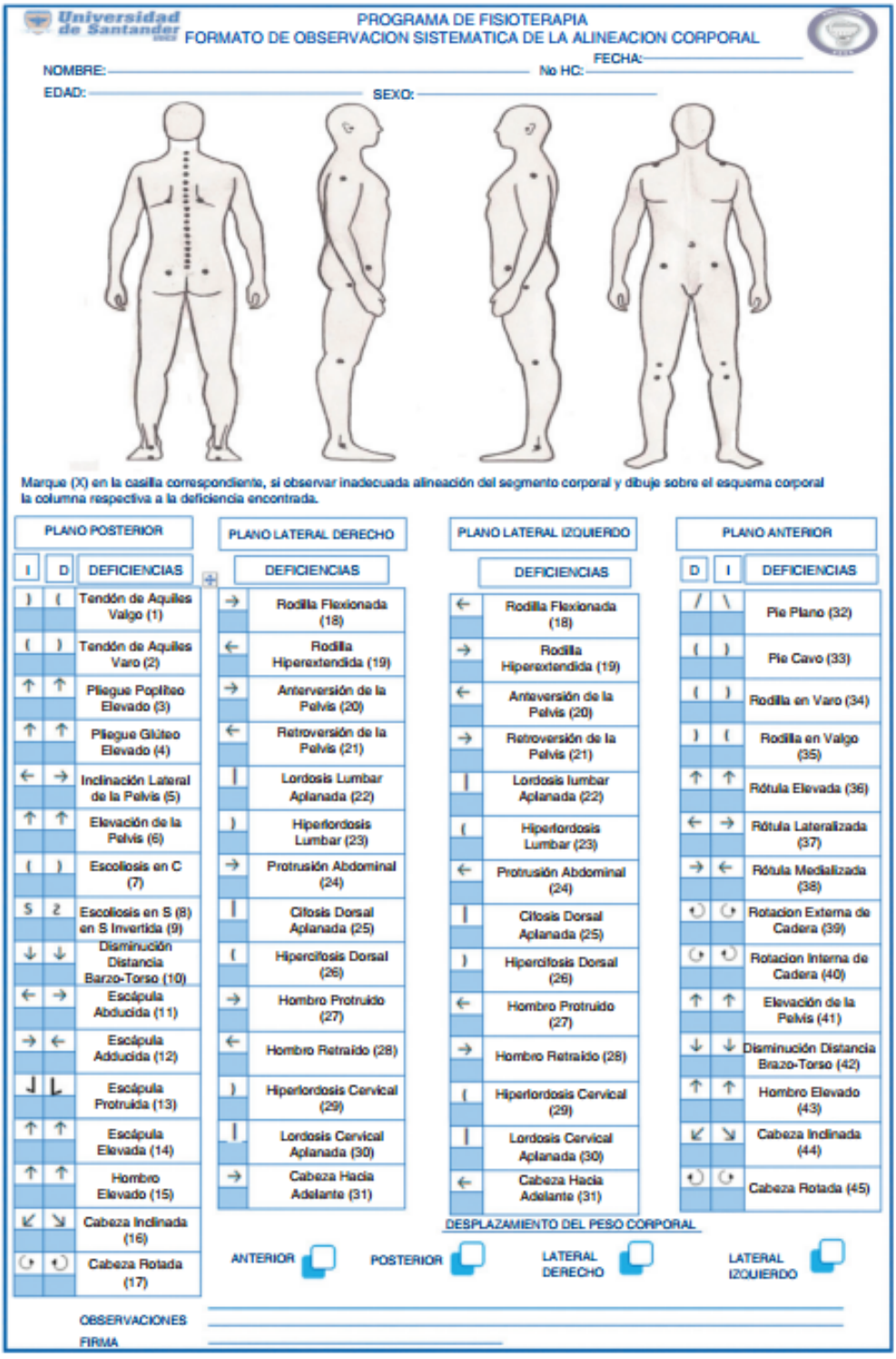


Figura 4.31: FOSAC.

El siguiente paso de este caso particular de estudio consistió en contrastar los análisis posturales

entre un especialista haciendo uso del formato FOSAC contra los que permite obtener el Kinect. Específicamente se contrastaron los referentes a la detección de desniveles entre hombros, caderas, rodillas y tobillos. Los resultados se muestran en la Tablas 4.2 y 4.3.

Paciente	**Vista Anterior (FOSAC)**	**Vista Anterior (Kinect)**
Gilberto Cardoso Martinez	Hombro derecho elevado, Rótula lateralizada (ambas)	Hombro derecho elevado (1.1 cm), Cadera derecha elevada (0.1 cm), Rodilla derecha elevada (2.2 cm), Tobillo izquierdo elevado (0.6 cm)
Nicole Morales Delgado	Hombro izquierdo elevado, Pie plano (ambos), Rótula medializada (ambas)	Hombro derecho elevado (1.3 cm), Cadera derecha elevada (0.4 cm), Rodilla izquierda elevada (1.2 cm), Tobillo izquierdo elevado (0.6 cm)
Daniela Diaz Urueta	Pie plano (izquierdo), Rotula medializada (izquierdo), Disminución distancia brazo-torso (izquierdo)	Hombro derecho elevado (0.9 cm), Cadera derecha elevada (0.2 cm), Rodilla izquierda elevada (0.9 cm), Tobillo derecho elevado (0.9 cm)
Gabriel Hernandez	Pie plano (derecho), Hombro derecho elevado	Hombro derecho elevado (2.5 cm), Cadera derecha elevada (0.5 cm), Rodilla izquierda elevada (1.2 cm), Tobillo derecho elevado (0.9 cm)
Diego Damian Diaz	Pie plano (ambos), Rodilla en varo (ambos), rotula Lateralizada (ambos), Disminución distancia brazo-torso (derecho), Hombro derecho elevado, Cabeza inclinada (derecho)	Hombro derecho elevado (1.1 cm), Cadera derecha elevada (0.6 cm), Rodilla izquierda elevada (2.2 cm), Tobillo izquierdo elevado (0.2 cm)

Tabla 4.2: Contrastación análisis frontal especialista vs Kinect.

Paciente	Vista Anterior (FOSAC)	Vista Anterior (Kinect)
Cinthya Jimena Alarcon	Rótula medializada (ambos), Disminución distancia brazo-torso (izquierdo)	Hombro derecho elevado (0.9 cm), Cadera derecha elevada (0.4 cm), Rodilla derecha elevada (1.3 cm), Tobillo derecho elevado (1.2 cm)
Saul Neftali Aldabalde	Pie cavo (izquierdo), Rodilla en varo (ambas), Disminución distancia brazo-torso (derecho), Hombro derecho elevado, Cabeza inclinada (derecho)	Hombro derecho elevado (1.2 cm), Cadera derecha elevada (0.3 cm), Rodilla izquierda elevada (0.1 cm), Tobillo izquierdo elevado (0.1 cm)
Jose Emmanuel Aguilar	Pie plano (ambos), Dismunicion distancia brazo-torso (izquierdo), Hombro izquierdo elevado	Hombro derecho elevado (0.4 cm), Cadera derecha elevada (0.4 cm), Rodillas al mismo nivel, Tobillo izquierdo elevado (4 cm)
Luis Angel Huerta de la Torre	Pie plano (izquierdo), Rótula medializada (ambos), Disminución distancia brazo-torso (derecho), Hombro izquierdo elevado	Hombro derecho elevado (0.6 cm), Cadera derecha elevada (0.2 cm), Rodilla izquierda elevada (0.3 cm), Tobillo derecho elevado (0.2 cm)
Ana Alexia Becerra	Pie plano (izquierdo), Rótula medializada (ambos), Disminución distancia brazo-torso (izquierdo), Hombro derecho elevado	Hombro derecho elevado (1.5 cm), Cadera derecha elevada (0.2 cm), Rodilla izquierda elevada (1.5 cm), Tobillo izquierdo elevado (0.5 cm)

Tabla 4.3: Contrastación análisis frontal especialista vs Kinect (cont.).

De las Tablas 4.2 y 4.3 es posible inferir varias cosas. De entrada, es visible que por sus características el sensor Kinect puede entregarnos información cuantitativa de lo que esta observando, esto es, dar un resultado numérico de por cuantos centímetros una articulación está por encima de la otra, esto es algo que para el especialista no es posible. Sin embargo, el especialista puede detectar una mayor cantidad de problemáticas, dónde el sensor Kinect se ve limitado porque sólo detecta 20 puntos del cuerpo humano. De una manera puntual en el análisis del desnivel de hombros se encontraron 5 coincidencias, 3 diferencias y 2 casos en donde el sensor detecto desnivel pero el

especialista no. Del análisis de cadera, rodillas y tobillos el especialista observo otros parámetros que no son cuantificables por el sensor, como el pie plano, rodillas en varo () o en valgo)(, por lo que la información adicional servirá de complemento para sus análisis. En general, se propone continuar utilizando el análisis postural tradicional y enriquecerlo con la información que proporciona el sensor Kinect.

Capítulo 5

Conclusiones

Al llevar a cabo la comparación de segmentación mediante técnicas tradicionales contra segmentación por distancia se observa claramente una enorme ventaja de ahorro en pasos necesarios para llevar a cabo esta ultima. Ademas condiciones como iluminación no afectan este tipo de segmentación y es posible segmentar objetos o cuerpos que por otros métodos no es posible o donde a pesar de múltiples pre y post procesamientos no se logran buenos resultados.

Las información de profundidad proporciona por el sensor Kinect se puede traducir en una imagen en escala de grises la cual es altamente susceptible de técnicas de pre-procesamiento, esto debido principalmente que la tecnología que usan este este tipo de dispositivos los hace muy susceptibles al ruido. Se puede observar que la utilización del filtro de mediana y filtros morfológicas permiten reducir y atenuar la presencia de ruido en la imagen de profundidad final lo que se traduce en la recuperación de información perdida. El filtro estadístico requiere de un tamaño de kernel grande para dar resultados aceptables dado el tamaño de ruido presente en las imágenes de profundidad del sensor, esto no ocurre así con los filtros morfológicos donde ademas se tiene la ventaja de una mejor preservación de los detalles de ésta.

Comparado con otros sensores de tiempo de vuelo, el Kinect presenta una gran ventaja al contar además con una cámara color, con lo que la interacción usuario – interfaz se vuelve sencilla e intuitiva, sin embargo, la imagen color no es de gran calidad por lo que es necesario, al igual que sucede

con la imagen de profundidad, someterla a técnicas de pre-procesamiento. En ese mismo sentido resulta mejor para el usuario final mostrarle directamente sobre la imagen color los objetos de interés segmentados por profundidad que sobre la representación de la información de profundidad en una imagen en escala de grises.

Finalmente, en el caso de estudio particular se abordaron las ventajas de utilizar el sensor Kinect como auxiliar de los expertos del área de fisioterapia dando buenos resultados.

Capítulo 6

Prospectivas

En este primer acercamiento con el sensor Kinect se ha trabajado en la parte de mejora y segmentación de imágenes o mapas de profundidad, queda abierta la posibilidad de procesar la imagen de color la cual, como se pudo ver en este trabajo, no es de mucha calidad. Es necesario además implementar los algoritmos propuestos en este trabajo para imagen de profundidad así como los nuevos algoritmos sugeridos a imágenes color mediante librerías rápidas (como OpenCV para C++ o EmguCV para C #) o incluso en hardware para procesar en tiempo real, es decir, trabajar dentro de la frecuencia de operación del sensor que llega a ser de hasta 33ms. De modo que el siguiente paso consiste en operar en el análisis de sistemas dinámicos (tanto robots fijos como móviles) y reconstrucción 3D de entornos de trabajo. Se deben además considerar el uso de herramientas de software libres para adquirir la información del sensor y la utilización de múltiples sensores al mismo tiempo para aplicaciones futuras.

Referencias

[1] Cayetano García, R. 2010. Sistema de Medición para determinar la microdureza en acero por visión artificial. Tesis de ingeniería. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

[2] Ceder, V. L. 2010. The Quick Python Book. Second Edition. Manning Publications Co, Sound View Court 3B, Greenwich, CT 06830.

[3] Chen, Y., Zhang, W., Yan, K. Li, X. and zhou, G. 2012. Extracting corn geometric structural parameters using Kinect. 978-1-4673-1159-5/12/ C 2012 IEEE. Beijing, China.

[4] Cruz, L., Lucio, D., and Velho, L. 2012. Kinect and RGBD Images: Challenges and Applications. 2012 XXV SIBGRAPI Conference on Graphics, Patterns and Image Tutorials. 978-0-7695-4830-2/12 2012 C IEEE. DOI 10.1109/SIBGRAPI-T.2012.13. Rio de Janeiro, Brazil.

[5] De León Cuevas, A., Tovar Arriaga, S., Vargas Soto, J. E. y Pérez Arreguín, J. I. 2012. Telecontrol de un robot móvil por medio de un sensor de proximidad RGB-D. Octavo Congreso Internacional de Ingeniería. Universidad Autónoma de Querétaro. Facultad de Ingeniería. Primera Edición. pp. 258-268. ISBN 978-607-513-021-7. Querétaro, México.

[6] Díaz Guerrero, R. E., Hernández Barrón, L. A., Pedraza Ortega, J. C., Gorrostieta Hurtado, E. y Vargas Soto, E. 2012. Análisis comparativo entre aplicaciones de reconocimiento de patrones mediante el uso el hardware Kinect TM. Octavo Congreso Internacional de Ingeniería. Universidad Autónoma de Querétaro. Facultad de Ingeniería. Primera Edición. pp. 8-23. ISBN 978-607-513-021-7. Querétaro, México.

[7] Flores Guerra, J. 2008. Homography. Based Multiple Camera Detection of Camouflaged Targets. Department of Electrical & Computer Engineering, ECE. University of Texas at El Paso. Thesis.

[8] Garduño Ramón, M. A. 2012. Mejora de contraste para imágenes de microscopio. Tesis de ingeniería. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

[9] Giorio, C., and Fascinari, M. 2013. Kinect in Motion - Audio and Visual Tracking by Example. First Edition. Packt Publishing Ltd, Livery Place, 35 Livery Street, Birmingham B3 2PB, UK.

[10] Gonzalez, R. C., Woods, R. E. and Eddins, S. L. 2004. Digital Image Processing. Second Edition. Pearson Prentice Hall, Upper Saddle River, New Jersey 07458.

[11] Gonzalez, R. C., Woods, R. E. and Eddins, S. L. 2004b. Digital Image Processing Using Matlab. First Edition. Pearson Prentice Hall, Upper Saddle River, New Jersey 07458.

[12] Han, J., Shao, L., Xu, D., and Shotton, J. 2013. Enhanced Computer Vision with Microsoft Kinect Sensor: A Review. IEEE Transactions oc Cybernetics. Vol. 43, No. 5, October, 2013. Identifier 10.1109/TCYB.2013.2265378.

[13] Hansard, M., Lee, S., Choi, O. and Horaud, R. 2012. Time-of-Flight Cameras: Principles, Methods and Applications. Uknown Edition. Springer, Upper Saddle River, New Jersey 07458.

[14] Hartley, R. and Zisserman, A. 2004. Multiple View Geometry in Computer Vision. Second Edition. Cambridge University Press, The Edinburg Building, Cambridge CB2 2RU, UK.

[15] Hernández López, J. J., Quintanilla Olvera, A. L., López Ramírez, J. L., Rangel Butanda, F. J., Ibarra Manzano, M. A. and Almanza Ojeda, D. L. 2012. Detecting objects using color and depth segmentation with Kinect sensor. The 2012 Iberoamerican Conference on Electronics Engineering and Computer Science. 2212-0173 C 2012. Published by Elsevier Ltd. doi:10.1016/j.protcy.2012.03.021. Guanajuato, Mexico.

[16] Jana, A. 2012. Kinect for Windows SDK Programming Guide. First Edition. Packt Publishing Ltd, Livery Place, 35 Livery Street, Birmingham B3 2PB, UK.

[17] Mejía Ugalde, M., Domínguez González, A., Trejo Hernández, M., Morales Hernández, L. A., Benítez Rangel, J. P. y Montes Olvera, L. M. 2012. Una nueva propuesta para la selección automática de la herramienta en torno de CNC aplicando procesamiento de imágenes. Octavo Congreso Internacional de Ingeniería. Universidad Autónoma de Querétaro. Facultad de Ingeniería. Primera Edición. pp. 269-286. ISBN 978-607-513-021-7. Querétaro, México.

[18] Morales Hernández, L. A., Manriquez Guerrero, F., Terol Villalobos, I., Domínguez González, A. y Herrera Ruíz, G. 2007. Desarrollo de sistema automático de segmentación morfológica de inclusiones de grafito. IEEE Quinto Congreso Internacional de Innovación y Desarrollo Tecnológico CIINDET. Cuernavaca, Morelos. México.

[19] Morales Hernández, L. A., Barajas Alarcón, M. R., Benítez Rangel, J. P., Ortiz Vargas, W., 2009. Caracterización de metalografías de aluminio por medio de visión por computadora. Quinto Congreso Internacional de Ingeniería. Universidad Autónoma de Querétaro. Facultad de Ingeniería. Primera Edición. Querétaro, México.

[20] Pacheco Estrada, A. H., Rivas Araiza, E. A., Vargas Vázquez, D., Bazan Trujillo, R. D. y Lozano Dorantes, R. 2011. Control de calidad de piezas maquinadas basado en procesamiento de imágenes. Séptimo Congreso Internacional de Ingeniería. Universidad Autónoma de Querétaro. Facultad de Ingeniería. Primera Edición. pp. 681-689. ISBN 978-607-7740-77-3. Querétaro, México.

[21] Pajares Martinsanz G. y de la Cruz García J. M., 2004. Visión por Computador. Imágenes Digitales y Aplicaciones 2008, Segunda Edición. Editorial Alfaomega. México.

[22] Peralta Benhumea, S. H. 2012. Interfaz de lenguaje natural usando Kinect. Tesis de maestría. Cetro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional. Unidad Zacatenco. Departamento de Computación. México, Distrito Federal.

[23] Razo Montes, C. A., Jaen Cuellar, A. Y., Muñoz Barrón, B., Morales Hernández, L. A. y Osornio Rios, R. A. 2011. Método para la detección de rostros en imágenes con modelo de color RGB basado en segmentación y operaciones elementales. Séptimo Congreso Internacional de

Ingeniería. Universidad Autónoma de Querétaro. Facultad de Ingeniería. Primera Edición. pp. 907-917. ISBN 978-607-7740-77-3. Querétaro, México.

[24] Rico Espino, J. G., González Barbosa, J. J. y Gómez Loenzo, R. A. 2011. Sistema de visión de reconstrucción 3D con lente telecéntrica. Séptimo Congreso Internacional de Ingeniería. Universidad Autónoma de Querétaro. Facultad de Ingeniería. Primera Edición. pp. 997-1013. ISBN 978-607-7740-77-3. Querétaro, México.

[25] Russ J. C. 2007. The Image Processing Handbook. Fifth Edition. Taylor & Francis. USA.

[26] Solís Arellano J. O., 2008. Análisis de la microestructura para detectar incrustaciones de grafito en el acero por medio de análisis de imágenes. Tesis de Ingeniería. Universidad Autónoma de Querétaro-Facultad de Ingeniería.

[27] von Engelhardt J., 2002. The Language of Graphics: A Framework for the Analysis of Syntax and Meaning in Maps, Charts and Diagrams. Universiteit van Amsterdam. Institute for Logic, Language and Computation.

[28] Young, D. 2013. Homography matrix solution using Matlab. `http://www.mathworks.com/matlabcentral/answers/26141-homography-matrix`. Marzo, 2013.

[29] Computing the plane to plane homography `http://www.robots.ox.ac.uk/~vgg/presentations/bmvc97/criminispaper/node3.html`. Marzo, 2013.

[30] Kinect for Windows `http://kinectforwindows.org/`. Marzo, 2013.

Apéndice A

Artículo CIINDET 2014

IEEE Sección Morelos

CIINDET
HACIA UN DESARROLLO SUSTENTABLE CON TECNOLOGÍA PROPIA

CONSTANCIA

Artículo: ***"Determinación de parámetros dimensionales en imágenes de líneas gravitacionales de referencia postural por medio de un sensor Kinect"***

Autores: **Marco Antonio Garduño Ramón, Arely Guadalupe Morales Hernández, Luis Alberto Morales Hernández, Irving Armando Cruz Albarrán, Roque Alfredo Osornio Ríos**

Id. artículo: **301**
Área: **Sistemas Computacionales**

El Comité Técnico del XI Congreso Internacional Sobre Innovación y Desarrollo Tecnológico CIINDET 2014, que se llevó a cabo en la ciudad de Cuernavaca, Morelos, México, del 2 al 4 de abril de 2014, hace constar que el artículo citado fue presentado de acuerdo con el programa técnico del congreso e incluido en las memorias del mismo.

La presente constancia se expide para los fines legales que a los autores convengan.

Cuernavaca, Morelos, México a 4 de Abril de 2014.

Atentamente

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC. FOUNDED NEW YORK 1884

Dr. Jorge Guillermo Calderón Guízar
Presidente del Comité Técnico CIINDET 2014

25 AÑOS 1989-2014 IEEE SECCION MORELOS ANIVERSARIO

CONGRESO INTERNACIONAL SOBRE INNOVACIÓN Y DESARROLLO TECNOLÓGICO

La Sección Morelos del Instituto de Ingenieros en Electricidad y en Electrónica
Otorga la presente

Constancia

A

Marco Antonio Garduño Ramón

Por su participación como

CONGRESISTA

Durante el XI Congreso Internacional sobre Innovación
y Desarrollo Tecnológico, realizado del 2 al 4 de abril del 2014,
en la ciudad de Cuernavaca, Morelos, México.

Dr. Rafael Castellanos Bustamante
Presidente IEEE Sección Morelos

ABRIL 2014
Cuernavaca • Morelos • México

CIINDET 2014
HACIA UN DESARROLLO SUSTENTABLE CON TECNOLOGÍA PROPIA

Seminario de Redes Inteligentes SRI 2014

CIINDET 2014

XI Congreso Internacional sobre Innovación y Desarrollo Tecnológico, 2 al 4 de abril de 2014, Cuernavaca Morelos, México.

Determinación de parámetros dimensionales en imágenes de líneas gravitacionales de referencia postural por medio de un sensor Kinect

M. A. Garduño-Ramón, *Graduate Student Member, IEEE*, A. G. Morales-Hernández, L. A. Morales-Hernández, I. A. Cruz-Albarrán y R. A. Osornio-Rios.

Resumen: Los expertos en rehabilitación física se auxilian de diferentes técnicas y métodos para detectar algún problema postural, algunos hacen uso de ciertos equipos y herramientas que les facilitan su trabajo, sin embargo, un diagnóstico acertado se basa en la experiencia del especialista. Es necesario contar con una herramienta accesible que ayude a detectar alguna problemática y que además facilite compartir la información obtenida durante un diagnóstico de un paciente con algún colega más experimentado en caso de duda. En este trabajo se presenta una metodología que permite cuantificar parámetros dimensionales en imágenes de líneas de referencia postural haciendo uso de un sensor Kinect de bajo costo, con la ventaja de que permite almacenar las imágenes y datos resultantes.
Palabras Clave: líneas gravitacionales, referencia postural, sensor Kinect.
Abstract: Physical rehabilitation experts are aided by different techniques and methods for detecting a posture problem, some of them make use of certain equipment and tools to facilitate their work, however, an accurate diagnosis is based on the experience of the specialist. It's necessary to have an accessible tool that help to detect some problem and also facilitate sharing information obtained during a diagnosis of a patient with a more experienced colleague in case of doubt. In this paper we present a methodology to quantify dimensional parameters in lines of postural reference images using a low cost sensor Kinect, with the advantage that it can store images and resulting data.
Keywords: gravitational lines, postural reference, Kinect sensor.

Ing. Marco Antonio Garduño Ramón, mgarduno01@alumnos.uaq.mx
Lic. Arely Guadalupe Morales Hernández,
Dr. Luis Alberto Morales Hernández, luis_morah@yahoo.com
Ing. Irving Armando Cruz Albarrán, irving_cruz93@hotmail.com
Dr. Roque Alfredo Osornio Rios, raor@uaq.mx
Universidad Autónoma de Querétaro, Av. Río Moctezuma 249 C.P. 76808, San Juan del Río, Qro, México.

Introducción

Durante muchos años, los expertos en el área de rehabilitación física, han estudiado los problemas posturales como una de las problemáticas más comunes en lo referente a los padecimientos físicos. Para lo anterior, se han auxiliado de diversas metodologías genéricas documentadas en manuales que les apoyen en su diagnóstico por ejemplo algunos de estas se pueden encontrar en [1]. Una de las herramientas utilizadas consiste en el análisis de los parámetros dimensionales de líneas gravitacionales de referencia postural donde se analiza el cuerpo ante el efecto de la gravedad y las dimensiones entre las diferentes articulaciones del paciente. La detección de algún problema físico, sin embargo, muchas veces se torna en un problema trivial, donde se depende en gran manera de la experiencia del kinesiólogo. Esto agrega cierto nivel de incertidumbre y se ha probado que donde un especialista con experiencia detecta cierta problemática, uno joven la pasa de largo. Además se dice que no existe un método de exploración de alineación corporal bípeda estática reconocida como prueba de oro para determinar las deficiencias posturales [2]. Esto por lo tanto refleja la necesidad de complementar este tipo de estudios con el uso de metodologías y herramientas tecnológicas novedosas para que el especialista pueda proporcionar un diagnóstico más acertado y que además le permita llevar un registro en el que pueda auxiliarse en caso de que tenga alguna duda con un colega más experimentado.

Existen equipos comerciales desarrollados para visualizar la posición de la línea del centro de gravedad del cuerpo o línea de carga de una persona de pie que usan tecnología láser, estos son usados por ejemplo, en los centros de rehabilitación infantil TELETON y algunos otros centros especializados [3]. Este tipo de equipos resultan no siempre accesibles económicamente, lo que hace que su utilización en zonas marginadas sea

prácticamente imposible. Otra forma de abordar esta problemática es mediante el uso de imágenes. Hoy día, el procesamiento digital de imágenes se utiliza como una herramienta de apoyo prácticamente en casi todas las áreas técnicas que requieran un diagnóstico visual [4]. La utilización de dispositivos de captura de imágenes son una gran herramienta en el entendimiento de múltiples fenómenos y procesos por tanto resultan un gran auxiliar dentro de las diferentes especialidades médicas. En ese sentido también se han desarrollado sistemas que integran dispositivos para captura y procesamiento de imágenes para llevar a cabo un análisis semi-automático y en algunos casos automático de la postura de una persona. Sin embargo, como ya se comentó, la limitante de los altos costos que hay que pagar por estos sistemas es prohibitiva para su integración en el diagnóstico, por ejemplo, algunos requieren de la adquisición múltiples sistemas de captura de imágenes y en la mayoría de los casos no son fácilmente escalables por el usuario mismo.

Otro aspecto importante es que algunos de estos equipos comerciales solo entregan la información de dos dimensiones, que proporciona una imagen común y corriente, la cual no es suficiente para algunas aplicaciones. La cuantificación de parámetros dimensionales de una cierta escena abre un mundo de posibilidades, anteriormente esto sólo era posible mediante la adquisición dispositivos costosos o con la utilización de múltiples cámaras [5] El sensor Kinect ha venido a revolucionar el desarrollo de aplicaciones dentro del área del procesamiento de imágenes debido a que este permite obtener no solo la información de una imagen en dos dimensiones, además es posible obtener la información de profundidad de una escena [6]. El Kinect se ha utilizado en los años recientes como auxiliar en procesos de rehabilitación física obteniéndose buenos resultados donde se resalta su bajo costo y tamaño compacto [7-10]. En este trabajo se presenta una metodología que permite hacer uso de la información de profundidad proporcionada por el sensor Kinect para cuantificar parámetros dimensionales entre articulaciones de un usuario y se complementa con un análisis estadístico a fin de verificar la repetibilidad que ofrece el sensor.

Este trabajo está organizado de la siguiente manera. En la sección de conceptos básicos se exponen las características fundamentales del sensor Kinect y la diferente información que proporciona. Posteriormente se hace mención de los procedimientos para la medición de distancias en imágenes y las herramientas estadísticas utilizadas. A continuación es expuesta la metodología usada, se discuten los resultados y finalmente se muestran las conclusiones y prospectivas.

Conceptos básicos

Sensor Kinect

El sensor Kinect apareció en Noviembre de 2010 como un accesorio de la consola Xbox 360. Fue desarrollado por la compañía Prime Sense en colaboración con Microsoft. El sensor Kinect (**Figura 1**) tiene una cámara de video así como un sensor de profundidad, en conjunto permite capturar imágenes a color y entregar la información de profundidad de cada pixel en la escena [6]. En Febrero de 2012 fue lanzada una versión específicamente para Windows a un precio de US$ 249 junto con un SDK disponible de manera gratuita en la red a fin de que los usuarios pudieran desarrollar aplicaciones propias.

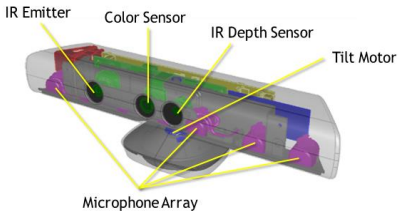


Fig. 1 Sensor Kinect

Imagen de color del sensor Kinect

El sensor Kinect cuenta con una cámara RGB que opera a 30 *Hz* y entrega imágenes de 640x480 pixeles con 8 bits por canal [6]. Es posible cambiar la resolución de la cámara a 1280x960 pixeles corriendo a 12 cuadros por segundo; con un FOV (campo de visión, por sus siglas en inglés) horizontal y vertical de 62° y 48.6° respectivamente. Se dice que es RGB porque precisamente hace uso de dicho espacio de color (Red, Green and Blue).

Sensor infrarrojo del sensor Kinect

La cámara infrarroja opera a 30 *Hz* y entrega imágenes con 1200x960 pixeles, estas son disminuidas a 640x480 pixeles con 11 bits, con lo cual provee 2048 niveles de sensibilidad [6]. El rango de operación es de entre 0.8 y

3.5 *m* aunque se recomienda trabajar en el rango entre 1.2 y 3 *m* para mejores resultados (**Figura 2**). Tiene un FOV horizontal de 58.5° y un FOV vertical de 45.6°. Cada pixel contiene la distancia cartesiana en *mm* desde el plano de la cámara hasta el objeto más cercano en esa coordenada particular (*x*,*y*).

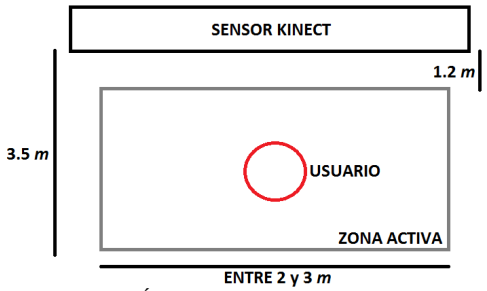


Fig. 2 Área de trabajo del sensor Kinect

Imagen de usuario esqueletizada

El sensor Kinect contiene una función interna la cual permite la detección de usuario además de la posición de 20 articulaciones o partes del cuerpo (**Figura 3**) de este junto con sus coordenadas con respecto al sensor Kinect [11]. Para esto, hace uso de la imagen de color y de la información obtenida con el sensor infrarrojo. A las uniones entre esos puntos se les conocen como líneas gravitacionales de referencia postural

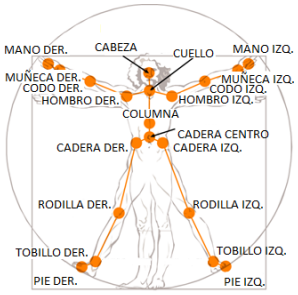


Fig. 3 Articulaciones y elementos del cuerpo detectados por el sensor Kinect

Medición de distancias en imágenes 2D y 3D

Una imagen puede ser definida como una función bidimensional *f(x,y)*, donde *x* e *y* son coordenadas espaciales en el plano (**Figura 4**), y la amplitud de f en cualquier par de coordenadas *(x,y)* es la intensidad de la imagen en ese punto (0 ó 1 para imágenes binarias; entre 0 y 255 para imágenes con niveles de intensidad de 8 bits).

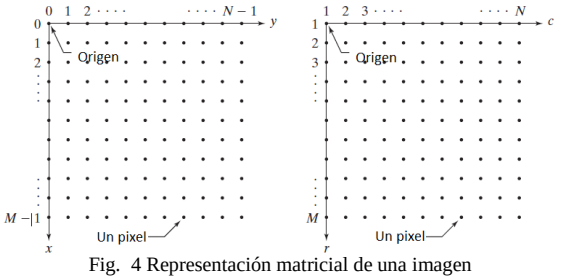


Fig. 4 Representación matricial de una imagen

De acuerdo con lo anterior, por lo tanto, es posible determinar la distancia entre dos puntos en una imagen en pixeles y, mediante la utilización de reglillas o algún otro instrumento parecido, relacionarlo para obtener la distancia en alguna unidad dimensional entre objetos en una imagen. Para los pixeles *p*, *q* y *z* con coordenadas *(x,y)*, *(s,t)* y *(v,w)* respectivamente, *D2* es una función distancia o métrica sí (1).

$$\begin{aligned} &a)\ D2(p,q) >= 0 \\ &\quad (D2(p,q) = 0 \text{ sí y solo sí } p = q) \\ &b)\ D2(p,q) = \quad D2(q,p)\ y \\ &c)\ D2(p,z) <= D2(p,q) + D2(q,z) \end{aligned} \tag{1}$$

La distancia euclidiana *D2* entre *p* y *q* se calcula por medio (2).

$$D2 = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2}$$

Este método es ampliamente utilizado [12], sin embargo, como se mencionó anteriormente el sensor Kinect entrega la información de profundidad de una escena, esto es, las coordenadas (*x*,*y*,*z*) de cualquier punto. De este modo podemos llevar (2) a su modo para espacios tridimensionales, mediante (3).

$$D3 = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_1)^2} \tag{3}$$

Media y desviación estándar

La media ($\bar{x}$) es una medida de tendencia central. Es el resultado de la suma de todos los elementos (x_i) divididos sobre el número total de estos (n). Se calcula mediante (4).

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (4)$$

La desviación estándar (S ó σ) es una medida de dispersión. Se define como la raíz cuadrada de la varianza, la cual se calcula con (5). De manera general, la desviación estándar dice que tan alejados están los datos de la media.

$$s^2 = \frac{\sum_{i=1}^{n} x_i^2 - n\bar{x}}{n} \quad (5)$$

Es de gran importancia hacer uso de estas herramientas estadísticas a fin de lograr una mejor comprensión del fenómeno estudiado, en este trabajo por ejemplo, corroborar que las mediciones hechas entre las diferentes articulaciones del cuerpo humano por medio del sensor Kinect sean iguales o cercana es cada una de las diferentes pruebas realizadas.

Metodología

La metodología propuesta se basa en el diagrama a bloques que se muestra en la **Figura 5**. El sistema adquiere datos por medio de un sensor Kinect conectado a una PC. Dentro de la computadora y en una aplicación desarrollada en Visual C# se separa la diferente información en una imagen color y en una imagen de usuario esqueletizado. Posteriormente ambas imágenes son sumadas y el resultado es desplegado como una imagen de color en la interfaz ofreciendo además la posibilidad de guardarla en formato PNG (Gráficos de Red Portátiles por sus siglas en inglés).

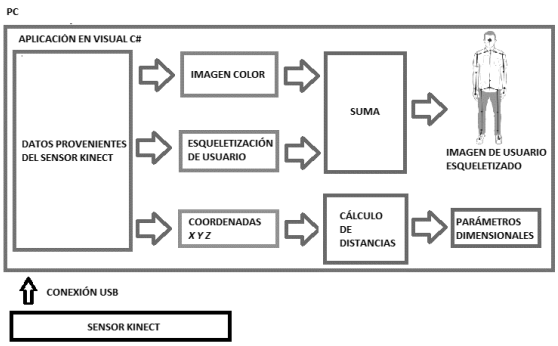


Fig. 5 Metodología propuesta

Por otro lado, mediante el uso de estructuras de programación se extrae las coordenadas de cada articulación detectada. El paso siguiente consiste en aplicar el algoritmo (3) para llevar a cabo el cálculo de la distancia entre un par de articulaciones. Este último paso puede ser realizado internamente dentro de la misma aplicación o de manera externa de acuerdo a las necesidades del usuario. Esto es posible porque los datos de posición de cada coordenada se pueden almacenar en un archivo de texto. De esta manera es posible complementar la información proporcionada por las imágenes con las distancias estimadas entre las líneas gravitacionales de referencia postural, a fin de proporcionar un diagnóstico más acertado.

Experimentación y resultados

Para implementar la metodología propuesta se colocó a un individuo que fungió como paciente a una distancia de 2.2 *m* del sensor Kinect, el cual se ubicó sobre una superficie plana a una altura de 0.95 m sobre el nivel del suelo. Esta configuración se muestra en la **Figura 6**.

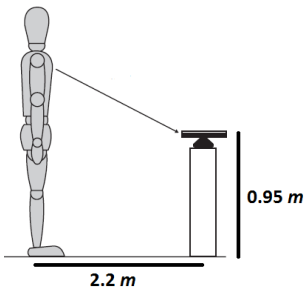


Fig. 6 Experimento propuesto para probar la metodología

Se le solicito al individuo que adoptara una postura erguida y viera de frente al sensor, a continuación se almacenaron tanto la imagen como las coordenadas obtenidas por el sensor. Este paso se repitió 5 veces con un intervalo de 1 min entre cada prueba, a fin de verificar la repetabilidad de las dimensiones estimadas. La imagen resultante de la suma de la imagen a color con la imagen de usuario esqueletizado entregado por el sistema se puede observar en las imágenes de la **Figura 8**. Por cuestiones de espacio en la **Tabla 1** sólo se muestran las coordenadas obtenidas de algunas articulaciones correspondientes a la prueba 1 que es la **Figura 8 a)**. Las **Figuras 8 c)** y **d)** permiten observar cierta desviación entre los puntos que unen el cuello, columna y el centro de la cadera, lo cual pudiera ser indicativo de algún problema físico o postural.

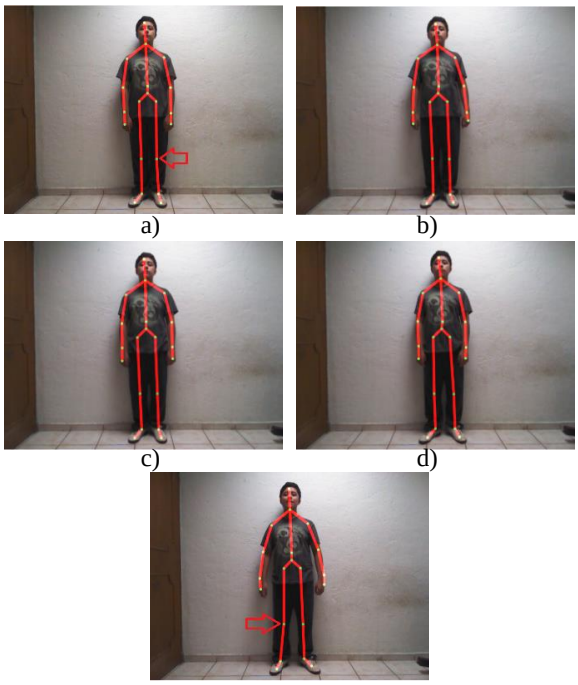


Fig. 8 Imagen esqueletizada de usuario

Se observa, además, una muy ligera desviación hacia adentro de la rodilla izquierda en la **Figura 8 a)** y una desviación más notoria hacia adentro de la rodilla derecha en la **Figura 8 e)**. Dado que en las **Figuras 8 a)**, **b)** y **e)** no se observa la desviación entre el cuello, columna y centro de la cadera, y en las **Figuras 8 b)**, **c)** y **d)** no se ve la desviación de las rodillas, por lo cual se hace evidente la necesidad de complementar la información proporcionada por las imágenes con la distancia entre las líneas de referencia postural a fin de mejorar el diagnóstico a ofrecer.

Tabla 1: Coordenadas (*x*,*y*,*z*) de la prueba 1

Elemento/Articulación	*x*(m)	*y*(m)	*z*(m)
Cabeza	-0.00	0.79	2.25
Hombro (izq.)	-0.15	0.45	2.25
Hombro (der.)	0.16	0.46	2.22
Cadera (izq.)	-0.05	0.06	2.19
Cadera (der.)	0.09	0.06	2.19
Codo (izq.)	-0.24	0.22	2.31
Muñeca (izq.)	-0.26	-0.02	2.21
Codo (der.)	0.26	0.24	2.30
Muñeca (der.)	0.30	0.00	2.18
Rodilla (izq.)	-0.05	-0.44	2.30
Tobillo (izq.)	-.083	-.781	2.27
Rodilla (der.)	.124	-.444	2.29
Tobillo (der.)	.138	-.777	2.34

De manera externa se aplicó el algoritmo para el cálculo de dimensiones con 3 coordenadas. Con base a lo observado en las imágenes de la **Figura 8**, se propuso calcular la distancia existente entre las articulaciones relacionadas con los problemas detectados (**Tabla 2)**. Finalmente se calculó el promedio así como la desviación estándar de cada dimensión estimada para las cinco pruebas. Estos datos son mostrados en la **Tabla 3**.

Tabla 2: Dimensiones entre articulaciones prueba 1, 2, 3, 4 y 5

	m	m	m	m	m
Cuello – Columna	.392	.391	.375	.378	.394
Columna – Cadera (c.)	.091	.090	.087	.087	.091
Cadera – Rodilla (izq.)	.521	.543	.531	.525	.529
Cadera – Rodilla (der.)	.522	.541	.532	.524	.541
Rodilla – Tobillo (izq.)	.337	.314	.324	.323	.321
Rodilla – Tobillo (der.)	.337	.326	.330	.334	.321

Tabla 3: Media y desviación estándar de las dimensiones calculadas

	$\bar{x}$	σ
Cuello - Columna	0.3864	0.0086
Columna – Cadera (c.)	0.0895	0.0019
Cadera – Rodilla (izq.)	0.5302	0.0083
Cadera – Rodilla (der.)	0.5323	0.0089
Rodilla – Tobillo (izq.)	0.3244	0.0086
Rodilla – Tobillo (der.)	0.3301	0.0062

Con la información de la **Tabla 2** es notorio el hecho de que en las pruebas 3 y 4 correspondientes a la **Figura 8 c)** y **d)** respectivamente, la distancia entre el cuello-columna y columna-cadera (c, centro) varían entre 2 y 3 *cm* con respecto a las otras tres pruebas. En base a los datos de la **Tabla 3** es posible detectar, por ejemplo, una diferencia de 2 *mm* entre la distancia de la cadera-rodilla derecha y la cadera-rodilla izquierda así como una diferencia de 5 *mm* en la distancia comprendida entre la rodilla-tobillo derecho y la rodilla-tobillo del lado izquierdo. Se observa en este conjunto de distancias, en particular, una desviación estándar de la distancia

calculada de entre 8.6 y 8.9 *mm*. La ventaja del sistema desarrollado es que permite compartir esta información con otro especialista a fin de hacer un diagnóstico más acertado, haciéndole notar los detalles encontrados en las imágenes y en los datos numéricos.

Conclusiones y prospectivas

Se desarrolló una metodología que permite obtener las imágenes de líneas gravitacionales y cuantificar parámetros dimensionales en ellas haciendo uso para del sensor de bajo costo Kinect. La metodología fue probada en un individuo joven en el que se detectaron problemas posturales al analizar en conjunto las imágenes y datos numéricos que entrega el sistema. El paso siguiente de este trabajo consistiría resaltar de manera automática aquellos elementos posturales donde se detecte algún problema, además de entregar la información de orientación entre los diferentes elementos articulares que detecta el sistema.

Referencias

[1] S. B. Brotzman, and R. C. Manske, *Clinical Orthopaedic Rehabilitation. An evidence-based approach*, PA, USA, ELSEVIER MOSBY, 2011.

[2] Y. Alfonso-Peñaloza et al, "Reproducibilidad interevaluador del Formato de Observación Sistemática de la Alineación Corporal en estudiantes universitarios", *Fisioterapia*, vol. 35 (4), pp 154-166, July-August 2013.

[3] S. Lillo, "Innovaciones tecnológicas en rehabilitación infantil y juvenil", *Boletín Técnico - Instituto de Rehabilitación Infantil Teletón Chile*, vol 6, pp 32-34, Diciembre 2008

[4] G. Pajares y J. M. De la Cruz, *Visión por computador. Imágenes digitales y aplicaciones*, Ciudad de México, México, Alfaomega, 2008.

[5] J. C. Russ, *The Image Processing Handbook*, Raleigh, NC, USA. CRC Press, 2011.

[6] L. Cruz, D. Lucio and L. Velho, "Kinect and RGBD Images: Challenges and Applications, Rio de Janeiro", *25th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, pp 36-49, August 2012

[7] N. A. Borghese, M. Pirovano, R. Mainetti and P. L. Lanzi, "An Integrated Low-Cost System for At-Home Rehabilitation", *18th International Conference on Virtual Systems and Multimedia (VSMM)*, pp 553-556, September 2012.

[8] N. Kitsunezaki, E. Adachi, T. Masuda and J. Mizusawa, "KINECT Applications for The Physical Rehabilitation", *IEEE International Symposium on Medical Measurements and Applications Proceedings (MeMeA)*, pp 294-299, May 2013.

[9] S. Obdrzálek, G. Kurillo, F. Ofli, R. Bajcsy et al, "Accuracy and Robustness of Kinect Pose Estimation in the Context of Coaching of Elderly Population", *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp 1188-1193, Sept. 2012

[10] Z. Xiao, F. Mengyin, Y. Yi and L. Ningyi, "3D Human Postures Recognition Using Kinect", *4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, pp 344-347, August 2012

[11] J. Webb and J. Ashley, *Beginning Kinect Programming with the Microsoft Kinect SDK*, New York, NY, USA. Springer Science+Business Media, 2012

[12] R. C. Gonzalez, and R. E. Woods, Digital Image Processing, Upper Saddle River, NJ, USA. Pearson Prentice Hall, 2008.

Currículo corto de los autores

Marco Antonio Garduño-Ramón: Ingeniero Electromecánico con Línea Terminal en Mecatrónica egresado de la Universidad Autónoma de Querétaro, México en 2010.

Arely Guadalupe Morales-Hernández: Licenciada en Quiropráctica egresada de la Universidad Estatal del Valle de Ecatepec. Con estancia en Parker College of Chiropractic en Dallas, Texas, EUA, en 2009.

Luis Alberto Morales-Hernández: Ingeniero Electromecánico, Maestro en Ciencias y Doctor en Ingeniería egresado de la Universidad Autónoma de Querétaro.

Irving Armando Cruz-Albarrán: Ingeniero Electromecánico egresado del Tecnológico de Estudios Superiores de Jocotitlán en 2012.

Roque Alfredo Osornio-Rios: Ingeniero Eléctrico egresado del Instituto Tecnológico de Querétaro, obtuvo el grado de Maestro y Doctor por parte de la Universidad Autónoma de Querétaro con honores en 2007.

Apéndice B

Artículo CONNIN 2014

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

10° CONGRESO INTERNACIONAL DE INGENIERÍA
12-16 MAYO 2014

Santiago de Querétaro, Querétaro, 10 de marzo de 2014

No. De referencia: **Auto-22**

Garduño-Ramón M. A.
Universidad Autónoma de Querétaro
PRESENTE

Me permito informarle, que su resumen titulado **"Morphological filters applied to Kinect depth images for noise removal as pre-processing stage"** fue recibido y después de ser revisado por el comité científico responsable del congreso, ha sido **ACEPTADO *(con correcciones menores)*** para su presentación en el 10° Congreso Internacional de Ingeniería, en la modalidad de **Presentación Oral**.

Adjunto a esta carta reciba la hoja del dictamen de evaluación emitido por el comité científico. Su trabajo en extenso lo debe enviar antes 11 de abril del año en curso, para poder ser publicado en el libro del congreso.

Su presentación será el día *miércoles 14 de mayo de las 10:20 a las 10:40 horas en campus San Juan del Río*. Le solicitamos estar al inicio de la sesión, para poder tener su material de proyección, en el lugar indicado.

Sera un placer contar con su presencia, le reiteramos nuestro agradecimiento por su aporte al éxito de este congreso.

ATENTAMENTE
"EL INGENIO PARA CREAR, NO PARA DESTRUIR"

Dr. Eduardo Arturo Elizalde Peña
COORDINADOR DEL COMITÉ CIENTIFICO
Facultad de Ingeniería, UAQ

CONCYTEQ

creandoconciencia

Universidad Autónoma de Querétaro
Cerro de las Campanas s/n. Colonia Las Campanas C.P. 76010
Santiago de Querétaro, Qro. México.

10° CONGRESO INTERNACIONAL DE INGENIERÍA
12-16 MAYO 2014

creandoconciencia

La **UNIVERSIDAD AUTÓNOMA DE QUERÉTARO**
a través de la **FACULTAD DE INGENIERÍA,**
Otorga la presente

CONSTANCIA

a:

Garduño-Ramón M. A., Morales-Hernández L. A., Osornio-Ríos R. A.

Por su participación con la **presentación oral** del trabajo:

"Morphological filters applied to Kinect depth images for noise removal as pre-processing stage"

en el Congreso Internacional de Ingeniería en su décima edición
realizado del 12 al 16 de mayo de 2014
en la ciudad de Santiago de Querétaro, México

CONCYTEQ

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

Dr. Aurelio Domínguez González
Director

Morphological filters applied to Kinect depth images for noise removal as pre-processing stage

Garduño-Ramón M. A. [#1], Morales-Hernández L. A. [*2], Osornio-Rios R. A. [#3]

[#] *Facultad de Ingeniería, Campus San Juan del Río, Universidad Autónoma de Querétaro*
Av. Río Moctezuma 249, CP 76808, San Juan del Río, Querétaro, México

[1] mgarduno01@alumnos.uaq.mx
[2] luis_morah@yahoo.com
[3] raor@uaq.mx

Abstract—**The Microsoft Kinect combines a RGB (red, green and blue) camera with an infrared (IR) sensor making possible quantify depth attributes in an image at a low cost just relating both information at a pixel to pixel level. However infrared sensor is very susceptible to noise in the form of holes which are visible in the depth map, being translated in depth data loss. Because of this, is necessary to develop a pre-processing stage to Kinect depth output information in order to recover the missing information. Mathematical morphology (MM) is based in set theory, where set A is a structuring element (SE), and set B, is a portion of the image of the same size where similarities or differences are looked for in an iterative process. MM is related with the structure and form of objects. It is widely used for pre and post-processing of binary and gray scale images. MM has two basic operations, erosion and dilation, and two compound operations, opening and closing called morphological filters. In this paper a methodology is proposed and evaluated. MM filters for gray scale images are applied as pre-processing stage to Kinect depth output in C++. Statistics filters are also applied for comparison. Color and depth images are gotten using a Visual C# application and the Kinect SDK (Software Development Kit). Morphological filtering shows great results for Kinect depth images, where 4 or 5 iterations are enough to completely remove the holes present in this kind of images. It is important to consider the possible removal or creation of structures that are or not originally presented this images due to this filtering, and the computational time of implementation which increases because of its iterative nature. Future work include improve Kinect color image and then apply a homography process to correlate both images correctly.**

I. INTRODUCTION

Microsoft Kinect sensor is a low-cost device which provides color image and depth information of a scene. Originally developed as an Xbox 360 accessory, it represented a whole revolution enabling interaction between game and user without the need of a physical controller [1]. With the release of a specific version for Windows 7 and a free cost software development kit (SDK) a lot of applications have been developed using this sensor, for example, object tracking and recognition, human activities analysis (pose estimation), hand gesture analysis (hand detection, gesture classification), indoor 3D mapping (sparse feature matching, dense point matching), etc., [2-6]. Some advantages of working with this device are the easy programming, the small size and form factor, and the low cost. Using traditional 3-D (such as stereo cameras and Time of Flight (TOF) sensors) cameras meant a high use of monetary resources, which it's not always suitable for everyone. Obviously there are some disadvantages. For example, the low quality color images, and the high susceptibility to noise due to depth sensing technology used by this device. There are works dealing with both problems. In the case of the noise present in depth images one characteristic necessary is to modify as less as possible the information acquired, but at the same time, try to recover the most data in order to increase the precision of the information [7-11]. One possible solution to this problem could be apply mathematical morphology (MM) to this images. MM is a widely tool used in image processing for filtering, thinning, and pruning, and it is much related to shape [12-13]. Another powerful tools for noise removal are the statistics filters (like median or mean), which are very popular and easy to implement in software. In this paper a methodology to aboard the Kinect depth images problem using mathematical morphology is proposed. Also statistics median filters are applied in order to make a comparison of both techniques. In section II, basic concepts are described as Kinect specifications and characteristics, the median filter concept and mathematical morphology theory. In section III, the proposed methodology for this works is presented and explained. In section IV we test our methodology and discuss the results obtained. Finally, work conclusions are in section V.

II. BASIC CONCEPTS

A. Kinect Sensor

The Kinect sensor (Fig. 1) was launched in November 2010 as an Xbox 360 video game console accessory. It was developed by the company Prime Sense in collaboration with Microsoft. In February 2012 a specific version for Windows 7 was released at US$ 249 along with a free cost Software Development Kit (SDK). Kinect has a color and an infrared (IR) depth sensor (Fig. 2). Together allow to capture color image and know the distance information of each pixel in millimeters [1-2].

1) *Color Sensor:* Kinect RGB color video camera works at 30 frames per second with a resolution of 640x480 pixels with 8 bits per channel. It also operates at 12 frames per second with a resolution of 1280x960 pixels. Its field of vision (FOV) horizontal and vertical is 62° and 48.6°, respectively.

Fig. 1. Kinect sensor

2) *IR Depth Sensor:* Kinect infrared sensor works also at 30 frames per second with a resolution of 640x480 pixels with 16 bits of depth info. The last three bits belong to player index. From the 13 remaining bits the more significant bit is always zero, so the distance information in millimeters is contained in 12 bits. This means a sensibility of 4096 levels for the sensor. Its depth sense range is from 0.8 m to 4 m in default mode and 0.4 m to 3.5 m in near mode, (Fig. 3). Its field of vision (FOV) horizontal and vertical is 58.5° and 45.6°, respectively.

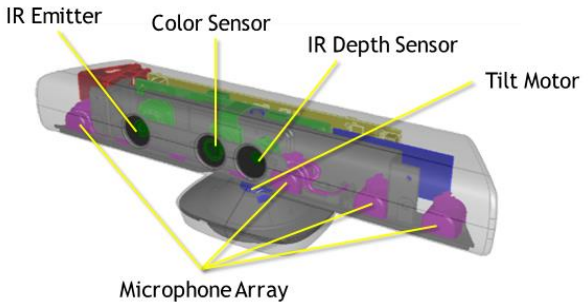


Fig. 2. Kinect sensor components

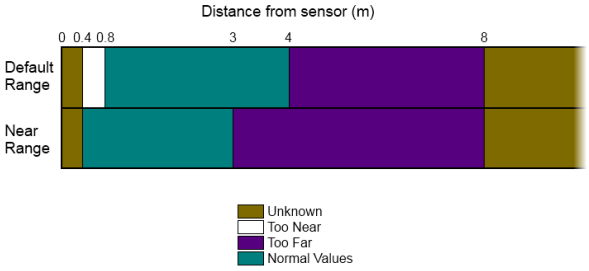


Fig. 3. Kinect sensor depth range

3) *Skeletal Tracking:* Skeletal Tracking allows Kinect to recognize people and follow their actions. Kinect can recognize up to six users in the field of view of the sensor. This function also allows to Kinect to monitor 20 human body articulations of two users, for this, it uses the depth information that acquires using the IR sensor and estimates the body joints, (Fig. 4). This is the most well-known function of this device and it is very popular in the control of user interfaces and devices [14].

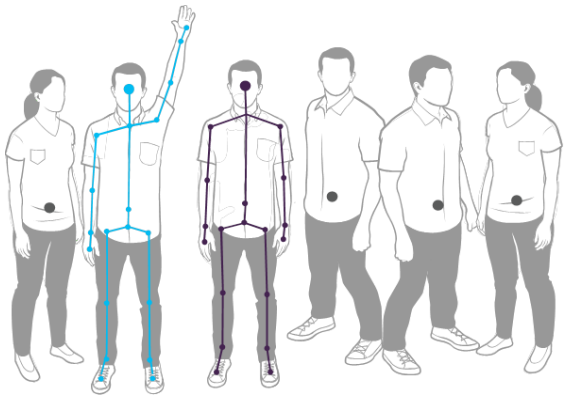


Fig. 4. User articulations detection by Kinect sensor

B. *Median Filter*

The best-know order-statistics filter is the median filter, Ec. 1, which, as its name implies, replaces the values of a pixel by the median of the gray levels in the neighborhood of that pixel [15].

$$f(x,y) = median_{(s,t)\in S_{x,y}} g(s,t) \quad (1)$$

The original value of the pixel is included in the computation of the median.

C. *Mathematical Morphology*

Morphology commonly denotes a branch of biology that deals with the form and structure of animals and plants. Mathematical morphology in image processing is a tool for extracting image components that are useful in the representation and description of region shape, such as boundaries, skeletons, and the convex hull. Another applications include pre- or post-processing stages as filtering, thinning, and pruning [15].

1) *Erosion and Dilation:* Erosion and dilation are the most important morphological transformations. Together are the base for more complex transformations.

Erosion combines two sets using vector addition, Ec. 2. One of the simplest uses of erosion is for eliminating irrelevant detail (in terms of size) from a binary image (Fig. 5).

$$A \ominus B = \{d \in E^2 : d + b \in X \; for \; each \; b \in B\} \quad (2)$$

Dilation combines two sets using vector subtraction, Ec. 3. One of the simplest applications of dilation is for bridging gaps (Fig. 6).

$$A \oplus B = \{d \in E^2 : d = x + b \; for \; each \; x \in X \; and \; b \in B\} \quad (3)$$

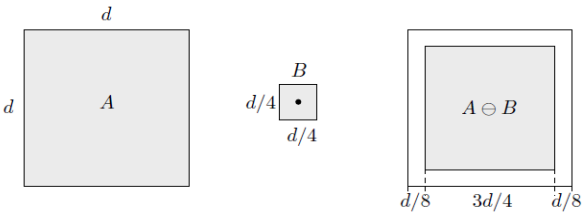


Fig. 5. Erosion of A by structuring element B

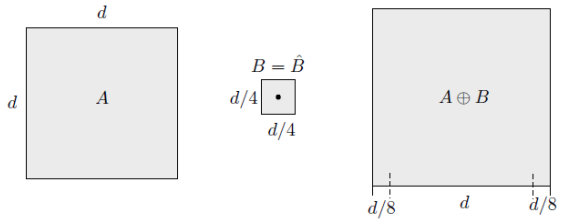


Fig. 6. Dilation of A by structuring element B

2) *Opening and Closing:* Eroding following by a dilating makes a morphological transformation called opening, Ec. 4. Opening generally smoothes the contour of an object, breaks narrow isthmuses, and eliminates thin protrusions (Fig. 7).

$$A \circ B = (X \ominus B) \oplus B \tag{4}$$

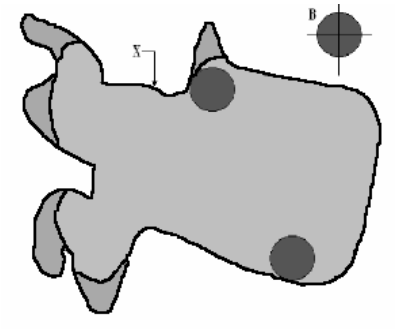


Fig. 7. Opening of X by structuring element B

Dilating following by an eroding makes a morphological transformation called closing, Ec. 5. Closing also tends to smooth sections of contours but, as opposed to opening, it generally fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour (Fig. 8).

$$A \circ B = (X \oplus B) \ominus B \tag{5}$$

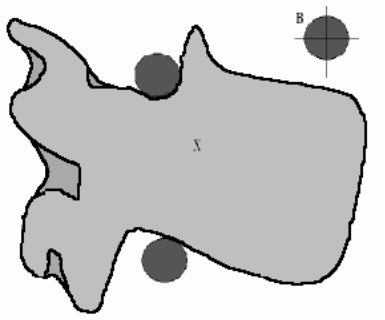


Fig. 8. Closing of X by structuring element B

III. METHODOLOGY

The proposed methodology for this paper is shown in the Fig. 9. From the Kinect sensor we get the color image and the depth information. We reserve the first one for posterior work. The depth information comes as a 16 bits data structure. From this 16 bits data structure, the last three bits belong to player index number, which are removed. The 13 bits remaining contain the distances in millimeters of every pixel. The more significant bit is always zero, so the relevant depth information is contained in 12 bits. This 12 bits structure then is scaled to an 8 bits in order to represent it as an 8 bits gray scale image. This image goes through mathematical morphology filtering, opening or closing filters, in an iterative process until the noise is fully removed.

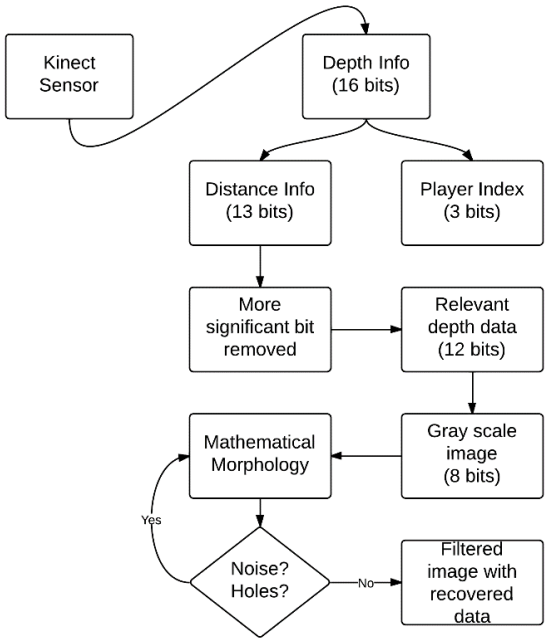


Fig. 9. Methodology

IV. EXPERIMENTS AND RESULTS

In order to probe the proposed methodology we read the depth information from a Kinect sensor of a user who is in his living room and in front of the device (Fig. 10). From this image

we can easily note the holes and not well defined structures in the chest and in the contour of the user. Also there are holes all over the background which could be a relevant problem, for example, in robotics applications. The black zones belong to undetermined distances or areas out of sensor range.

Fig. 10. Kinect depth image with noise and holes

The first 3 iterations of mathematical morphology filtering using a structuring element of 3x3 as basics, doesn't show good results, this is because of the size of the noise and holes present in the Kinect depth image, it was until iterations 4 and 5, which belong to a filter size 9x9 and 11x11, when good results were achieved. Figs. 11a and 11b show iterations 4 using both closing and opening filters. In this ones, we can see a better definition of structures in the user chest. In the case of closing filtering almost the whole background is unified. It is clear than opening filtering defines better the user contour without joining both user legs but the black shadow remains around him which doesn't occurred whit the closing filter.

a) Closing filter 9x9 b) Opening filter 9x9

Fig. 11. MM filter 9x9

Figs. 12a and 12b show iterations 5 using closing and opening, respectively. In this case, closing filter has removed almost every hole and noise signal in the image, the result is very uniform and consistent. One disadvantage is that the separation between both legs has decreased and the loss of definition in the user's fingers. For the opening filter we maintain the not uniform nature of background, the user silhouette is well defined, the low chest user depth data information is decreasing every iteration, and again, there is a loss of definition in user's fingers.

a) Closing filter 11x11 b) Opening filter 11x11

Fig. 12. MM filter 11x11

Median filter was also implemented. Figs. 13a and 13b belong to a kernel size 9 and 11 respectively. Full holes removal is not achieved, but the user silhouette, hand fingers and legs are well defined. The results are very similar to opening filter with some light advantages.

a) Median filter kernel size 9x9 b) Median filter kernel size 11x11

Fig. 13. Median filtering

V. CONCLUSIONS

In this paper mathematical morphological filters, opening and closing, were applied to Kinect depth images with good results. Closing filtering delivers better results because of its nature relative to filling holes and gaps. Opening filtering on the other hand, achieves a better contour definition of structures, but cannot fully deal with the problem of holes and noise. In both cases, there is a loss of definition in small details of the image, this was notorious in the fingers of the user's hand and disappear of certain structure is more notorious using the closing filter. Four and five iteration of a 3x3 structuring element was enough to get better results, this corresponds to apply directly a structuring element of 9x9 and 11x11, respectively. A single comparative against median filter was also done. Curiously, kernels size 9x9 and 11x11 gave better results, which were similar to opening filtering related to noise and holes problems. However, median filter shows better results defining the user contour and even the user fingers are better detailed.

REFERENCES

[1] L. Cruz, D. Lucio, and L. Velho, "Kinect and RGBD Images: Challenges and Applications," *25th SIBGRAPI Conference on Graphics, Patterns and Image Tutorials*, pp. 36–49, Aug. 2012.

[2] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced Computer Vision with Microsoft Kinect Sensor: A Review," *IEEE Transactions on Cybernetics*, vol. 43(5), pp. 1318–1334, Oct. 2013.

[3] Z. Xiao, F. Mengyin, Y. Yi, and L. Ningyi, "3D Human Postures Recognition Using Kinect," *4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 1, pp. 344–347 , Aug. 2012.

[4] L. Yong-Wan, L. Hyuk-Zae, Y. Na-Eun, and P. Rae-Hong, "3-D Reconstruction Using the Kinect Sensor and Its Application to a Visualization System", *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3361-3366, Oct. 2012.

[5] J. L. Raheja, A. Chaudhary, and K. Singal," Tracking of Fingertips and Centres of Palm using KINECT", *Third International Conference on Computational Intelligence, Modelling and Simulation (CIMSiM)*, pp. 248-252, Sept. 2012.

[6] Y. Chen, W. Zhang, K. Yan, X. Li, and G. zhou, "Extracting corn geometric structural parameters using Kinect", *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 6673-6676, July 2012.

[7] G. Danciu, S. M. Banu, and A. Caliman, "Shadow removal in depth images morphology-based for Kinect cameras," *16th International Conference on Systems Theory, Control and Computing (ICSTCC)*, pp. 1–6, Oct. 2012.

[8] K. Xu, J. Zhou, and Z. Wang, "A Method of Hole-filling for the Depth Map Generated by Kinect with Moving Objects Detection", *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1-5, June 2012.

[9] S. Matyunin, D. Vatolin, Y. Berdnikov, and M. Smirnov, "Temporal filtering for depth maps generated by Kinect depth camera," *3DTVC Conference: The True Vision – Capture, Transmission and Display 3D Video (3DTV-CON)*, pp. 1–4, May. 2011.

[10] K. Essmaeel, L. Gallo, E. Damiani, G. De Petro, and A. Dipanda," Temporal denoising of Kinect depth data", *Eighth International Conference on Signal Image Technology and Internet Based Systems (SITIS)*, pp. 47-52, Nov. 2011.

[11] C. V. Nguyen, S. Izadi, and D. Lovell, "Modelling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking," *Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pp. 524–530, Oct. 2012.

[12] J. Serra, *Image analysis and Mathematical Morphology*, 1st Ed., Academic Press, 1982.

[13] J. Serra, Ed., *Image analysis and Mathematical Morphology, Volume 2: Theoretical Advance*, 1st Ed., Academic Press, 1988.

[14] J. Webb, and J. Ashley, *Beginning Kinect Programming with the Microsoft Kinect SDK*, 1st Ed., 233 Springer Street, 6th floor, New York, NY 10013, Springer Science+Business Media, 2012.

[15] R. C. Gonzalez, and R. E. Woods, *Digital Image Processing*, 2nd Ed., Upper Saddle River, New Jersey 07458, Prentice Hall, 2002.

Apéndice C

Algoritmo de homografía en Python

La implementación del algoritmo de homografía en Matlab propuesto por David Young de la Universidad de Sussex resulta de enorme utilidad, sin embargo este software no es libre. Es por eso que para complementar el trabajo realizado en este trabajo de tesis se realizó en Python, el cual, como se mencionó en el Capitulo 2, cuenta con librerías igual de poderosas a las que incluye el software de Mathworks.

```
__author__ = 'Marco'

# HOMOGRAPHY_SOLVE finds a homography from point pairs
#   V = HOMOGRAPHY_SOLVE(PIN, POUT) takes a 2xN matrix of input vectors and
#   a 2xN matrix of output vectors, and returns the homogeneous
#   transformation matrix that maps the inputs to the outputs, to some
#   approximation if there is noise.
#
#   This uses the SVD method of
#   http://www.robots.ox.ac.uk/%7Evgg/presentations/bmvc97/criminispaper/node3.html
#
# Implementacion realizada en Python 2.7.6 usando numpy 1.8.0
#Ing. Marco Antonio Garduno Ramon, Universidad Autonoma de Queretaro, Febrero 2014

import numpy as np  # libreria numerica


# funcion hoografia
def homography(pin, pout):
    """

    :param pin: puntos de entrada de la homografia
    :param pout: puntos de salida de la homografia
    """
    if pin.size != pout.size:
        print("Matrices de puntos de diferentes tamanos")

    if pin.shape[0] != 2:
        print("Las matrices de puntos deben tener dos filas")

    n = pin.shape[1]

    if n < 4:
        print("Se necesitan al menos cuatro puntos correspondientes")

    # Resolucion de ecuaciones usando SVD

    x_in = pout[0, :]
    y_in = pout[1, :]
    x_out = pin[0, :]
    y_out = pin[1, :]

    rows0 = np.zeros((3, n))
    rows_xy = (-1) * np.vstack((x_out, y_out, np.ones((1, n))))

    hx = np.vstack((rows_xy, rows0, x_in * x_out, x_in * y_out, x_in))
    hy = np.vstack((rows0, rows_xy, y_in * x_out, y_in * y_out, y_in))

    h = np.hstack((hx, hy))

    if n == 4:
        u, s, v = np.linalg.svd(h)
    else:
        u, s, v = np.linalg.svd(h, full_matrices=False)

    h_matrix = np.reshape(u[:, 8], (3, 3))

    return h_matrix
```

```
__author__ = 'Marco'

# HOMOGRAPHY_TRANSFORM applies homographic transform to vectors
#   Y = HOMOGRAPHY_TRANSFORM(X, V) takes a 2xN matrix, each column of which
#   gives the position of a point in a plane. It returns a 2xN matrix whose
#   columns are the input vectors transformed according to the homography
#   V, represented as a 3x3 homogeneous matrix.

import numpy as np


def homography_transform(x, h_matrix):
    q = np.dot(h_matrix, np.vstack((x, np.ones((1, 1)))))
    p = q[2, :]
    y = np.array([[q[0, :]/p], [q[1, :]/p]])

    return y
```

```
__author__ = 'Marco'

import homography
import homography_solve
import numpy as np

# pin - puntos de entrada de la homografia
entrada = np.array([[99, 184, 447, 536, 625,  98, 181, 269, 357, 446, 534,  94, 180, 267, 354, 444, 533,  93, 179, 266,
                     353, 442, 531,  93, 178, 265, 351, 440, 530,  94, 178, 265, 350, 439, 526, 263, 350, 618],
                    [53,  51,  54,  55,  56, 110, 112, 114, 115, 117, 119, 176, 178, 178, 181, 185, 187, 238, 242, 245,
                     247, 250, 252, 304, 307, 309, 312, 314, 319, 366, 370, 376, 379, 381, 383, 436, 438, 452]])

# pout - puntos de salida de la homografia
salida = np.array([[55, 142, 429, 522, 622, 52, 141, 239, 329, 428, 523,  48, 142, 235, 328, 427, 521,  48, 140, 235,
                    329, 426, 523,  50, 141, 234, 328, 425, 520,  48, 141, 235, 327, 425, 519, 236, 325, 618],
                   [33,  32,  29,  31,  31, 96,  97, 100, 100, 100, 100, 170, 170, 173, 172, 174, 175, 238, 239, 242,
                    243, 245, 246, 310, 310, 312, 315, 317, 318, 378, 378, 384, 385, 388, 388, 449, 450, 463]])

# llamado de funcion homografia
H = homography.homography(entrada, salida)

# punto a proyectar
x = np.array([[179],
              [372]])

# calculo de punto proyectado
y = homography_solve.homography_transform(x, H)

print H
print y
```

Apéndice D

Aplicación básica para sensor Kinect

```
namespace Microsoft.Samples.Kinect.ColorBasics
{
    using System;
    using System.Globalization;
    using System.IO;
    using System.Windows;
    using System.Windows.Media;
    using System.Windows.Media.Imaging;
    using Microsoft.Kinect;

    public partial class MainWindow : Window
    {
        private KinectSensor sensor;
        private WriteableBitmap colorBitmap;
        private byte[] colorPixels;

        public MainWindow()
        {
            InitializeComponent();
        }

        private void WindowLoaded(object sender, RoutedEventArgs e)
        {
            foreach (var potentialSensor in KinectSensor.KinectSensors)
            {
                if (potentialSensor.Status == KinectStatus.Connected)
                {
                    this.sensor = potentialSensor;
                    break;
                }
            }

            if (null != this.sensor)
            {
                this.sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
                this.colorPixels = new byte[this.sensor.ColorStream.FramePixelDataLength];
                this.colorBitmap = new WriteableBitmap(this.sensor.ColorStream.FrameWidth, this.
                    sensor.ColorStream.FrameHeight, 96.0, 96.0, PixelFormats.Bgr32, null);
                this.Image.Source = this.colorBitmap;
                this.sensor.ColorFrameReady += this.SensorColorFrameReady;

                try
                {
                    this.sensor.Start();
                }
                catch (IOException)
                {
                    this.sensor = null;
                }
            }
            if (null == this.sensor)
            {
```

```
                this.statusBarText.Text = Properties.Resources.NoKinectReady;
            }
        }
        private void WindowClosing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (null != this.sensor)
            {
                this.sensor.Stop();
            }
        }

        private void SensorColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)
        {
            using (ColorImageFrame colorFrame = e.OpenColorImageFrame())
            {
                if (colorFrame != null)
                {
                    colorFrame.CopyPixelDataTo(this.colorPixels);
                    this.colorBitmap.WritePixels(
                        new Int32Rect(0, 0, this.colorBitmap.PixelWidth, this.colorBitmap.
                            PixelHeight),
                        this.colorPixels,
                        this.colorBitmap.PixelWidth * sizeof(int),
                        0);
                }
            }
        }

        private void ButtonScreenshotClick(object sender, RoutedEventArgs e)
        {
            if (null == this.sensor)
            {
                this.statusBarText.Text = Properties.Resources.ConnectDeviceFirst;
                return;
            }

            BitmapEncoder encoder = new PngBitmapEncoder();
            encoder.Frames.Add(BitmapFrame.Create(this.colorBitmap));
            string time = System.DateTime.Now.ToString("hh'-'mm'-'ss", CultureInfo.
                CurrentUICulture.DateTimeFormat);
            string myPhotos = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);
            string path = Path.Combine(myPhotos, "KinectSnapshot-" + time + ".png");

            try
            {
                using (FileStream fs = new FileStream(path, FileMode.Create))
                {
                    encoder.Save(fs);
                }

                this.statusBarText.Text = string.Format(CultureInfo.InvariantCulture, "{0} {1}",
                    Properties.Resources.ScreenshotWriteSuccess, path);
            }
            catch (IOException)
            {
                this.statusBarText.Text = string.Format(CultureInfo.InvariantCulture, "{0} {1}",
                    Properties.Resources.ScreenshotWriteFailed, path);
            }
        }
    }
}
```

```
<Window x:Class="Microsoft.Samples.Kinect.ColorBasics.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Color Basics" Height="735" Width="770" Loaded="WindowLoaded" Closing="WindowClosing">

    <Window.Resources>
        <SolidColorBrush x:Key="MediumGreyBrush" Color="#ff6e6e6e"/>
        <SolidColorBrush x:Key="KinectPurpleBrush" Color="#ff52318f"/>
        <SolidColorBrush x:Key="KinectBlueBrush" Color="#ff00BCF2"/>
        <Style TargetType="{x:Type Image}">
            <Setter Property="SnapsToDevicePixels" Value="True"/>
        </Style>
        <Style TargetType="{x:Type Button}" x:Key="SnapshotButton" >
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type Button}">
                        <Grid>
                            <StackPanel Orientation="Horizontal" Background="Transparent">
                                <TextBlock x:Name="SnapText" Text="{TemplateBinding Content}"
                                    TextAlignment="Left"  VerticalAlignment="Center" Foreground="{
                                    StaticResource KinectPurpleBrush}" FontSize="15" />
                                <Grid Margin="9,0,0,0">
                                    <Image x:Name="SnapNormal" Source="Images\SnapNormal.png"
                                        Stretch="None" HorizontalAlignment="Center"/>
                                    <Image x:Name="SnapHover" Source="Images\SnapHover.png"
                                        Stretch="None" HorizontalAlignment="Center" Visibility="
                                        Collapsed"/>
                                </Grid>
                            </StackPanel>
                        </Grid>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsMouseOver" Value="true">
                                <Setter Property="Visibility" Value="Collapsed" TargetName="
                                    SnapNormal"/>
                                <Setter Property="Visibility" Value="Visible" TargetName="
                                    SnapHover"/>
                                <Setter Property="Foreground" Value="{StaticResource
                                    KinectBlueBrush}" TargetName="SnapText"/>
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
    </Window.Resources>

    <Grid Name="layoutGrid" Margin="10 0 10 0">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <DockPanel Grid.Row="0" Margin="0 0 0 20">
            <Image DockPanel.Dock="Left" Source="Images\Logo.png" Stretch="Fill" Height="32" Width
                ="81" Margin="0 10 0 5"/>
            <TextBlock DockPanel.Dock="Right" Margin="0 0 -1 0" VerticalAlignment="Bottom"
                Foreground="{StaticResource MediumGreyBrush}" FontFamily="Segoe UI" FontSize="18">
                Color Basics</TextBlock>
            <Image Source="Images\Status.png" Stretch="None" HorizontalAlignment="Center" Margin="
                0 0 0 5"/>
        </DockPanel>
```

```
        <Viewbox Grid.Row="1" Stretch="Uniform" HorizontalAlignment="Center">
            <Image Name="Image" Width="640" Height="480"/>
        </Viewbox>
        <Button Grid.Row="2" Style="{StaticResource SnapshotButton}" Content="Screenshot" Height="
            Auto" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10 10 0 10" Name=
            "buttonScreenshot" Click="ButtonScreenshotClick" />
        <StatusBar Grid.Row="3" HorizontalAlignment="Stretch" Name="statusBar" VerticalAlignment="
            Bottom" Background="White" Foreground="{StaticResource MediumGreyBrush}">
            <StatusBarItem Padding="0 0 0 10">
                <TextBlock Name="statusBarText" Margin="-1 0 0 0">Press 'Screenshot' to save a
                    screenshot to your 'My Pictures' directory.</TextBlock>
            </StatusBarItem>
        </StatusBar>
    </Grid>
</Window>
```