DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE INTERACCIÓN ENTRE ROBOTS E-PUCKS DIRIGIDOS POR UN USUARIO Y OBJETOS VIRTUALES

ANDRES CAMILO CUBIDES FRANCO

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2016

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE INTERACCIÓN ENTRE ROBOTS E-PUCKS DIRIGIDOS POR UN USUARIO Y OBJETOS VIRTUALES

ANDRES CAMILO CUBIDES FRANCO

Proyecto de grado para optar al título de Ingeniero Mecatrónico

Director
JESÚS ALFONSO LÓPEZ
Ingeniero Electricista

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2016

Nota de aceptación:

Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente para optar al título de Ingeniero Mecatrónico.

JIMMY TOMBE ANDRADE
Jurado

VICTOR ROMERO
Jurado

Santiago de Cali, 03 de marzo de 2017

AGRADECIMIENTOS

Agradezco a Dios por guiarme y darme la oportunidad de estudiar en esta Universidad para obtener un título profesional que me ha permitido labrar un camino lleno de aprendizajes, experiencias, oportunidades, conociendo personas increíbles que han aportado a mi crecimiento tanto personal como profesional.

Los verdaderos protagonistas en este proceso han sido mis padres, este es nuestro triunfo, no lo hubiera alcanzado sin ellos, he sido yo quien ha asistido a las clases, sin embargo, han sido mis padres quienes han trabajado por este logro desde mucho antes que ingresara a la universidad, han sido mi ejemplo a seguir, me han inculcado valores y un gran compromiso que hacen de mí el hombre que soy hoy. Mis padres han sido los trabajadores incansables detrás de bambalinas que me han estado apoyando, quienes me han brindado su ayuda e ideas cuando me encontraba estancado con un trabajo, quienes se han aguantado mi cansancio y mal genio después de un largo día de Universidad que no terminó muy bien, quienes están detrás de mí para que a pesar de la carga y los trabajos me alimente bien, quienes han sido pacientes y comprensivos conmigo cuando en la época de exámenes solo me dedicaba a estudiar y no ayudaba en el hogar ni participaba de las actividades familiares, quienes me han brindado su experiencia, consejos y ánimos cuando más lo necesitaba motivándome siempre a dar todo de mí y lograr mis metas con éxito, por todo esto y mucho más, mil gracias Pa, mil gracias Ma, los amo.

A mi hermano gracias por su hermandad, por cubrirme con las tareas del hogar, cuando estaba muy ocupado con mis trabajos, por su espontaneidad que brinda más alegría a los momentos que hemos vivido en familia, por ser ejemplo de determinación y perseguir lo que a uno le apasiona, gracias hermanito, te amo.

A mis compañeros de universidad y amigos, David Álvarez y Maria Camila Álvarez, gracias por hacer de esta experiencia universitaria una vivencia increíble, porque el equipo de trabajo que formamos fue excelente y divertido, el desempeño y aprendizaje que obtuve en la universidad no habría sido posible sin ustedes, gracias por los momentos de risas y distensión (por nuestros apodos), por las conversaciones reflexivas e interesantes, por la paciencia y comprensión con mis 15 minutos de sueño, ustedes son personas que admiro como seres humanos y profesionales, gracias por su amistad.

A mis profesores de la universidad gracias por su vocación y escoger una de las profesiones más importantes para el futuro del mundo, por tener la paciencia de responder mi gran cantidad de preguntas, por compartir sus conocimientos y

motivarme en cada una de sus áreas transmitiéndome un poco de su pasión por la materia.

A mi director de tesis y profesor Jesús Alfonso López muchas gracias por su apoyo y guía desde el principio de la carrera, por motivar y alimentar mi interés en la mecatrónica, por su paciencia y tiempo para atender todas mis inquietudes en la universidad, gracias por compartir conmigo sus conocimientos y experiencias, y promover mi curiosidad e interés en las redes neuronales. Gracias por confiar y apostar en mis conocimientos y habilidades brindándome la oportunidad de realizar el intercambio en suiza, una de las experiencias más increíbles y enriquecedores que he vivido y que incluso ha cambiado mi rumbo.

A Andrés Pérez y Héctor Satizábal muchas gracias por su guía, apoyo y colaboración en el desarrollo de mi tesis en Suiza, trabajar en su laboratorio fue una experiencia muy agradable, amena y enriquecedora, sus consejos, ideas, comentarios y disposición fueron esenciales para la culminación exitosa de mi proyecto.

Gracias a la Universidad Autónoma de Occidente porque me brindó una educación profesional en la cual aprendí a utilizar excelentes herramientas y desarrolle habilidades y metodologías que me ayudarán a desenvolverme como profesional, no muchas personas tienen acceso a este nivel de educación sin embargo la UAO me permitió vivir esta experiencia y adquirir estos conocimientos al ofrecerme la oportunidad de estudiar toda la carrera becado.

CONTENIDO

Pág.

RESUMEN 8

INTRODUCCIÓN 9

1. JUSTIFICACIÓN 11

2. ANTECEDENTES 12

3. OBJETIVOS 18

3.1 OBJETIVO GENERAL 18

3.2 OBJETIVOS ESPECIFICOS 18

4. MARCO TEORICO 19

4.1 PROCESAMIENTO DE IMAGEN 20

4.2 CONTROL PID 27

4.3 CONCEPTOS PARA EL DESARROLLO DE VIDEOJUEGOS 28

5. RESULTADOS 29

5.1 VISIÓN GLOBAL DEL PROYECTO IMPLEMENTADO 29

5.2 IMPLEMENTACIÓN Y RESULTADOS ESPECÍFICOS 32

5.2.1 Lógica del juego 32

5.2.2 Configuración del proyector y el Kinect 33

5.2.3 Programación del Kinect 34

5.2.4 Configuración y software de los E-pucks 35

5.2.5 Detección de los E-pucks 35

5.2.6 Detección del usuario 42

5.2.7 Desplazamiento de los E-pucks 47

5.2.8 Programación con Pygame 53

6. CONCLUSIONES 57

7. MEJORAS A FUTURO 61

BIBLIOGRAFÍA 62

RESUMEN

Actualmente vivimos en un mundo inmerso en lo digital con el fin de entretenernos, aprender, trabajar, conocer personas y lugares, debido a esta tendencia se están desarrollando nuevas maneras de acercar a los usuarios cada vez más a este mundo digital y aumentar su capacidad de interacción, una opción para lograr esto consiste en proyectar información directamente en el mundo físico y permitir al usuario interactuar con elementos virtuales a través de diferentes interfaces o incluso robots. El presente proyecto de tesis tiene como objetivo diseñar e implementar un sistema de interacción entre robots e-pucks dirigidos por un usuario y objetos virtuales, usando técnicas de visión artificial y de detección de colisiones. Para el desarrollo de la visión artificial se hizo uso de un sensor Kinect y debido a que, tanto este sensor como los robots e-pucks, son programables en Python se utilizó Pygame para el desarrollo del videojuego y así un solo lenguaje de programación. Como resultado de esta tesis el videojuego desarrollado consiste en limpiar la basura del mar reciclando correctamente los desechos por medio de cuatro robots que se mueven sobre un escenario virtual proyectado en una mesa. El usuario debe señalar a un E-puck para activarlo y luego señalar una basura del tipo que ese robot especifico puede limpiar (Papel, latas, plástico o vidrio) para que este se dirija a desecharla. El usuario deberá realizar la recolección de basuras rápidamente para evitar que la contaminación llegue a niveles críticos y pierda el juego pero teniendo en cuenta que si causa una colisión entre los robots el juego también acabara. Por medio del emisor infrarrojo del Kinect se realizó la detección del brazo del usuario para poder calcular un vector en el espacio que indique a donde apunta y así posteriormente implementar técnicas de procesamiento de imagen en los mapas de profundidad, obtenidos del sensor, para detectar a los robots. Una vez detectadas las áreas de los e-puck se procedió a calcular la posición y orientación de estos permitiendo la implementación de un control PID para dirigirlos correctamente hasta las basuras que deben recolectar.

PALABRAS CLAVE: Robótica. Visión artificial. Procesamiento de imagen. Interfaz natural de usuario. Realidad virtual. Videojuego.

INTRODUCCIÓN

A lo largo de la historia podemos observar como a medida que se van desarrollando nuevas tecnologías de hardware el procesamiento digital de imágenes (PDI) va avanzando. Por medio de los diferentes lenguajes de programación y sistemas operativos el PDI se ha extendido a un gran número de áreas y aplicaciones como imágenes astronómicas, satelitales, médicas para la detección de enfermedades y anomalías, también en seguridad para alarmas activadas por medio de videos de vigilancia o aplicaciones industriales, entre muchos otros campos.

La constante miniaturización del computador ha permitido el desarrollo de dispositivos vestibles (Wearable Devices) dotados de sensores, capaces de procesar la información sensada y de brindar una interfaz multimodal con el usuario. Por ejemplo, un smartwatch es capaz de vibrar, emitir sonidos y está dotado de una pantalla que puede visualizar cierta información. Sin embargo, el tamaño de dichos dispositivos limita el tipo y la cantidad de información que se puede comunicar a través de su pantalla, por esta razón el desarrollo de nuevas interfaces de usuario se encuentra actualmente en una etapa de bastante investigación y desarrollo para permitir a los usuarios interactuar con la información de una manera diferente como por ejemplo haciendo uso de dispositivos vestibles como anillos para controlar la música o televisores y sensores para la detección del cuerpo y el movimiento como lo es el Kinect para el control de videojuegos.

Con todos estos avances tecnológicos en este momento vivimos en un mundo inmerso en lo digital, hacemos uso del mundo digital para entretenernos, para aprender, para trabajar, para conocer personas y lugares, entre muchas otras funcionalidades que nos brinda esta tecnología. Una de las opciones que se han pensado para permitir al usuario acercarse cada vez más a este mundo digital y aumentar su capacidad de interacción con este consiste en proyectar información directamente en el mundo físico y permitir al usuario interactuar con elementos virtuales en el ambiente a través de diferentes interfaces o incluso robots. Para utilizar robots como interfaz de interacción entre el usuario y los elementos virtuales es necesario abarcar la problemática de detección de colisiones la cual es utilizada en diferentes áreas como en la robótica móvil, para evitar obstáculos, en la animación, para diseñar aplicaciones coherentes con el ambiente virtual, y por último en el área de la salud en simulaciones de operaciones quirúrgicas.

Por todo lo anterior en este proyecto se desarrolló un videojuego que fomenta el reciclaje adecuado de los desechos permitiendo a los niños interactuar con objetos virtuales y robots E-pucks. La detección del área de trabajo y los cuatro robots e-

pucks se realizó a través de un sensor Kinect debido a que permite tener una visión artificial a través de los datos obtenidos por medio del proyector infrarrojo, por esta razón el sensor Kinect se debe ubicar en el techo paralelo a la mesa de trabajo para obtener una correcta visión del espacio de trabajo y los gestos del usuario. Para acercar el usuario al mundo virtual y permitir la interacción con los robots se utilizó un proyector para sobreponer la imagen del juego sobre la superficie de trabajo donde se encontraran los E-pucks. El entorno virtual consiste en un mar que se va contaminando con el tiempo por lo cual el usuario debe dirigir a los robots y así limpiar los desechos virtuales. Para lograr que el usuario controle los e-pucks por medio de comandos y movimientos naturales e intuitivos no se utilizaron mandos o controles, en su lugar se procesan los datos de profundidad del brazo del usuario, cuando está apuntando a un objetivo, y así brindar una agradable experiencia por medio de gestos sencillos. La implementación de un PID permitió a través de la visión artificial monitorear la velocidad y rotación de los robots de tal manera que llegara a la posición de su objetivo y generar una interacción coherente con las imágenes proyectadas en el juego.

1. JUSTIFICACIÓN

En la actualidad, vivimos en un mundo donde el ser humano cada vez hace más uso del mundo digital con diferentes fines, educación, entretenimiento, ocio, conocer personas, trabajar, etc. Por esta razón se han desarrollado diferentes dispositivos que permitan al usuario acercarse cada vez más a este mundo digital por ejemplo celulares, consolas de juegos, gafas de realidad aumentada, etc. Una de las ideas que busca aumentar la capacidad de interacción con el usuario, consiste en proyectar información en el mundo físico y permitir al usuario interactuar con elementos virtuales en el ambiente, por esta razón este proyecto apunta en esa dirección con el fin de acercar al usuario a un mundo virtual a través del uso de visión artificial y una interfaz de control natural. Para esto es necesario investigar y ampliar los conocimientos de procesamiento digital de imágenes de tal manera que la implementación de visión artificial permita al usuario interactuar con objetos virtuales de una manera más natural, adicionalmente para desarrollar un videojuego como resultado de la interacción del usuario con el mundo digital se debe adquirir conocimientos en el desarrollo y programación de juegos. Los dispositivos vestibles y detectores de movimiento, como los sensores kinect y la PlayStation Camera, se encuentran dotados de una variedad de sensores que permiten obtener información a partir de la cual se puede desarrollar una mayor interacción de una manera intuitiva y sencilla para brindar una agradable experiencia de usuario. La implementación de robots en la interacción con el entorno virtual brindara una experiencia más dinámica e implicara investigación y desarrollo en el área de la robótica móvil y en cómo integrarla de manera coherente y eficiente con visión artificial, detección de colisiones y el desarrollo y diseño de video juegos con interfaces naturales y agradables para el usuario, de tal manera que los conocimientos adquiridos y la aplicación desarrollada, pueden servir como punto de partida para otros estudiantes interesados en la investigación y desarrollo de la robótica móvil y nuevas formas de interacción con el usuario.

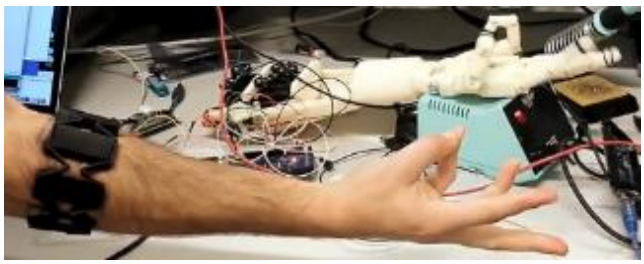
Debido a que este proyecto se realiza en conjunto entre la universidad Autónoma de Occidente y Haute Ecole d'Ingénierie et de Gestion du Canton Vaud (HEIG-VD), el desarrollo de esta tesis promoverá la cooperación y el trabajo colaborativo entre profesores y estudiantes de diferentes lugares del mundo, obteniendo así conocimientos y puntos de vista desde otras perspectivas que enriquecen nuestro camino de aprendizaje como estudiantes. Esto fortalecerá y afianzará los vínculos entre universidades, lo cual permite promover la movilidad de estudiantes entre estas instituciones brindando un abanico de nuevas oportunidades a diferentes estudiantes.

2. ANTECEDENTES

Actualmente el procesamiento de imágenes y las interfaces naturales de usuarios son utilizadas en un sinnúmero de aplicaciones y áreas, el uso de sensores vestibles o detectores de movimiento como interfaz para el control de dispositivos electrónicos y robots se encuentran en un gran auge generando así una gran oferta de dispositivos que buscan brindar al usuario nuevas maneras de interactuar con la información y el mundo digital, para este tipo de aplicaciones se pueden encontrar en el mercado diferentes opciones como Myo, Nod, Shimmer, Kinect entre otros.

La empresa Thalmic Labs busca disminuir la brecha entre la tecnología y los humanos, con el fin de conectar el mundo real y el digital por medio de los dispositivos vestibles y la computación ubicua[1]. Por esta razón esta empresa lanzó al mercado Myo, un brazalete que cuenta con bluetooth, un sensor de electromiografía, un giroscopio, un acelerómetro y un magnetómetro de tres ejes (IMU)[2]. Este dispositivo es utilizado en diferentes aplicaciones como en el control de videos, música, presentaciones, videojuegos, entre otros; en la robótica la empresa ha desarrollado aplicaciones para controlar el Parrot AR.Drone, la rueda robotica Orbotix Sphero, el Parrot Bebop, entre otros, actualmente los desarrolladores trabajan en diferentes proyectos para el control de otras interfaces robóticas como humanoides, prótesis (Figura 1), carros de control remoto, entre otros.

Figura 1. Control de una prótesis por medio del Myo.



Fuente: Myo - An Armband That Gives You Superpowers [video]. THALMIC LABS, 2015 [consultado 28 de Septiembre 2015] Disponible en internet: https://www.youtube.com/watch?t=49&v=UL_pDatlLOg

[1] About [en linea]. THALMIC LABS, 2013 [consultado 28 de Septiembre de 2015]. Disponible en Internet: https://www.thalmic.com/about

[2] Tech Specs [en linea]. THALMIC LABS, 2013 [consultado 28 de Septiembre 2015]. Disponible en internet: https://www.myo.com/techspecs

Otro de los productos utilizados para acercar al usuario al mundo digital y brindarle una mayor interacción con dispositivos electrónicos es el anillo Nod. A través de este dispositivo se ha llevado a cabo también el control del Parrot AR.Drone (Figura 2) en el área de la robótica, y adicionalmente el control de computadores, Smartphones y Smart Tvs[3], en el área de videojuegos se han desarrollado otras aplicaciones como realidad virtual por medio de las Oculus.

Figura 2. Gesto para el control de un drone por medio del anillo Nod.

Fuente: Nod enables natural drone control [en linea]. NOD, 2015 [consultado 28 de Septiembre de 2015]. Disponible en Internet: https://nod.com/

El sensor Shimmer es un dispositivo desarrollado por la empresa del mismo nombre y la última versión de este cuenta con 10 grados de medición inercial por medio de un giroscopio, un acelerómetro y un magnetómetro, adicionalmente cuenta también con un altímetro y conexiones para módulos de expansión de electromiografía y electrocardiograma[4]. Este sensor es bastante utilizado y se han realizado una gran variedad de aplicaciones en medicina, neuro-marketing, robótica y motion sensing. En el área de la medicina se ha utilizado para el monitoreo de personas con epilepsia, personas en rehabilitación y deportistas de alto rendimiento. A través del dispositivo se puede monitorear las actividades del usuario y en caso de un ataque o lesión, notificar a la persona a cargo de brindarle asistencia, en el caso de los deportistas brinda datos importantes para mejorar el desempeño de este. En el área de robótica se ha realizado el control del Parrot ARDrone (Figura 3) al igual que los otros dos dispositivos, esta aplicación es la que se desea realizar en este caso en el control de un robot móvil como el e-puck.

[3] Nod enables natural drone control [en línea]. NOD, 2015 [consultado 28 de Septiembre de 2015]. Disponible en Internet: https://nod.com/

[4] Wireless Sensor Platform [en línea]. SHIMMER, 2013 [consultado 28 de Septiembre de 2015]. Disponible en internet: http://www.shimmersensing.com/images/uploads/docs/Shimmer3_Spec_1.4.pdf

Figura 3. Control del Parrot ARDrone por medio del sensor shimmer.

Fuente: PEREZ URIBE, Andrés. Gesture control of a quadcopter [video]. 2012 [consultado 28 de Septiembre de 2015]. Disponible en internet: https://www.youtube.com/watch?v=dClxYuMmi4s#t=151

Por ultimo tenemos el sensor Kinect de Microsoft el cual es un dispositivo con tecnología para la detección de profundidad debido a que cuenta con una cámara a color 1080p, un emisor infrarrojo, y micrófonos, permitiendo detectar la ubicación, el movimiento y las voces de los usuarios[5] abriendo así la puerta a un gran abanico de posibilidades para el desarrollo de nuevas formas de interacción con el mundo digital. El sensor kinect salió al mercado con el Xbox con el principal propósito de mejorar e innovar la manera en que los usuarios controlan los videojuegos, sin embargo hoy en día este sensor es utilizado en muchas áreas del conocimiento totalmente diferentes como la medicina, la pedagogía, el mercadeo, la robótica, entre otras. En la medicina encontramos proyectos como GestSure, el cual busca permitir a los doctores tener acceso y manipular imágenes de tomografías y resonancias magnéticas durante cirugías solo a través de gestos sencillos (Figura 4) evitando así la necesidad de tocar teclados y computadores los cuales no se encuentran esterilizados implicando que el doctor invierta tiempo vital de la cirugía en desinfectarse de nuevo las manos.

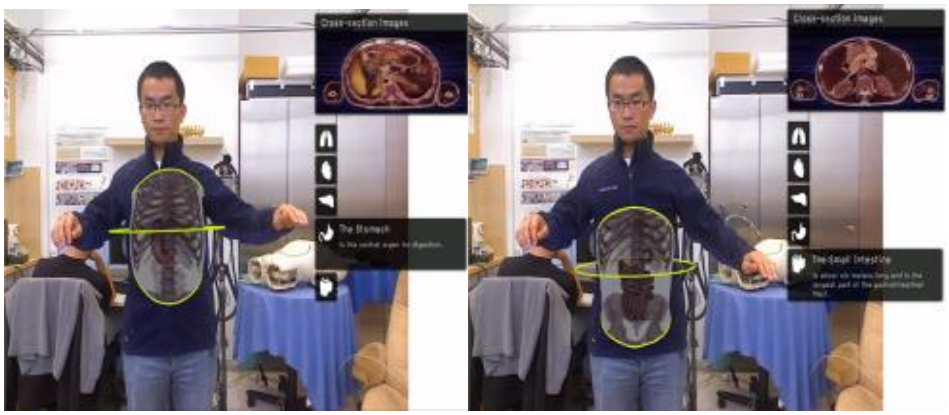
[5] Kinect hardware [en línea]. Microsoft, 2017 [consultado 20 de Enero de 2017]. Disponible en Internet: https://developer.microsoft.com/en-us/windows/kinect/hardware

Figura 4. Control de imágenes con Kinect.

Fuente: Watch the expanded gestsure story here [video]. GestSure [consultado 28 de Octubre 2016] Disponible en internet: http://www.gestsure.com/

En la Figura 5 se puede observar el proyecto Magic Mirror realizado en la Universidad técnica de Munich, el cual consiste en utilizar el kinect para mostrar en una pantalla la imagen en tiempo real del usuario y a través de la detección del cuerpo sobreponer imágenes de tomografías y mostrar las secciones del cuerpo en la derecha de la pantalla, todo esto es controlado por la mano del usuario quien indica la sección del cuerpo que desea ver, este sistema busca brindar un nuevo método para facilitar el aprendizaje de la Anatomía.

Figura 5. Kinect como método de aprendizaje de anatomía.



Fuente: FALLAVOLLITA, Pascal et al. Augmented Reality Magic Mirror using the Kinect [video]. Technische Universität München, 2015 [consultado 29 de Octubre 2016]. Disponible en internet: http://campar.in.tum.de/Chair/ProjectKinectMagicMirror

Durante el evento EXPO Milano 2015, se presentó el futuro distrito de comida en el cual las personas podían entrar a un supermercado donde encima de los productos hay pantallas y kinects (Figura 6) que permiten al usuario apuntar a uno de estos y obtener información detallada como, donde se cultivó, que químicos utilizaron, donde lo almacenan, hace cuanto fueron recogidos del suelo, entre otros datos, los desarrolladores de este proyecto, las empresas Accenture y Avanade, decidieron utilizar el kinect debido a la detección de personas a una distancia media y la alta confiabilidad del uso de sus sensores durante 24 horas seguidas.

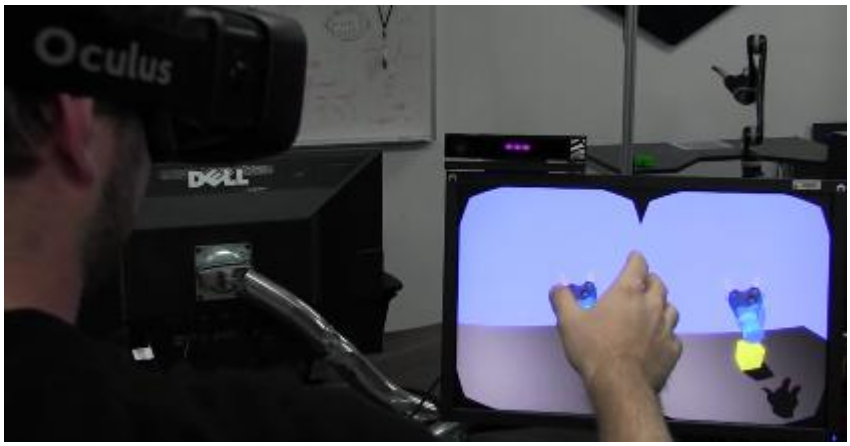
Figura 6. Supermercado interactivo en Milán.

Fuente: Kinect powers informed shoppers [en línea]. Microsoft, 8 de Julio de 2015 [consultado 29 de Octubre 2016]. Disponible en internet: https://blogs.msdn.microsoft.com/kinectforwindows/2015/07/08/kinect-powers-informed-shoppers/

Para la NASA las naves espaciales robóticas representan un área importante y critica en el futuro de la exploración espacial por lo cual se encuentran investigando opciones para mejorar la eficiencia y facilitar el control a distancia de estas naves. Por esta razón en el Jet Propulsion Laboratory de la NASA, en el instituto de tecnología de California, se está desarrollando un proyecto para tele operar robots en marte de una manera más sencilla y natural a través de la implementación de la segunda versión del kinect y las oculus rift. El nuevo sensor del kinect permite una mayor precisión al momento de detectar los movimientos reproduciendo correctamente la rotación de las muñecas de un usuario, en la Figura 7 se puede observar un usuario que controla el brazo robótico en frente de

él, con el fin de recoger el objeto en la mesa el cual puede ver en un mundo virtual utilizando las gafas Oculus y el kinect se encarga de reproducir los movimientos del usuario en el robot.

Figura 7. Kinect como herramienta de teleoperación.



Fuente: ARCHAMBAULT, Michael. NASA uses Microsoft's Kinect 2 and Oculus Rift to control space robots [en línea]. Windows Central, 27 de Diciembre de 2013 [consultado 29 de Octubre 2016]. Disponible en internet: http://www.windowscentral.com/nasa-uses-microsoft-kinect-2-and-oculus-rift-control-space-robots

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Diseñar e implementar un sistema de interacción entre robots e-pucks dirigidos por un usuario y objetos virtuales, usando técnicas de visión artificial y algoritmos de detección de colisiones.

3.2 OBJETIVOS ESPECIFICOS

3.2.1 Implementar el procesamiento de imágenes obtenidas a través de un kinect, para una correcta detección de los objetos reales que deben interactuar con los virtuales proyectados.

3.2.2 Programar comportamientos en los e-pucks para interactuar con el ambiente virtual y detectar colisiones.

3.2.3 Implementar un sistema de interacción con el usuario por medio de una interfaz natural.

3.2.4 Validar el sistema desarrollado implementando un juego donde el usuario interactúe con los e-pucks y estos con el entorno proyectado.

4. MARCO TEORICO

La integración de objetos y mundos reales y virtuales, da origen al área de realidad mezclada, la cual se basa en diferentes estrategias y tecnologías de visualización para permitir una interactividad. Esta área se encuentra dividida en dos, si el ambiente es principalmente virtual y se le agregan objetos virtuales y reales, se trata de realidad virtual, en el caso contrario donde el entorno dominante es real y se agregan objetos virtuales, se habla de realidad aumentada. Para la realidad aumentada es necesario tomar diferentes señales del mundo real (video o audio) las cuales se procesan para realizar diferentes acciones, dependiendo de la aplicación, y así lograr complementar el entorno real de tal manera que coexista y responda de manera coherente ante objetos del mundo virtual[6].

En este proyecto los objetos reales que interactuaran con los virtuales son los robots e-pucks, estos son el resultado de un proyecto desarrollado en la escuela politécnica federal de Lausana con el fin de proporcionar un robot móvil miniatura que funcione como herramienta de desarrollo y enseñanza en diferentes áreas de la ingeniería. Los robots e-pucks presentan un diámetro de 75 mm, en la Figura 8 se puede observar el tamaño de este comparado a un mouse y un teclado, este pequeño robot cuenta con una variedad de sensores y actuadores, tales como ocho sensores de proximidad alrededor de su estructura, un acelerómetro de tres ejes, tres micrófonos para capturar el sonido, una cámara a color con resolución 640x480, dos motores paso a paso, un parlante y ocho diodos rojos alrededor de su estructura.

[6] HERAS LARA, Lizbeth. La realidad aumentada: una tecnología en espera de usuarios [en línea]. En: Revista Digital Universitaria. 10 de agosto 2004, vol. 5, no. 7, p. 4-5 [consultado 28 de Octubre de 2016]. Disponible en Internet: http://www.revista.unam.mx/vol.8/num6/art48/jun_art48.pdf

Figura 8. E-puck

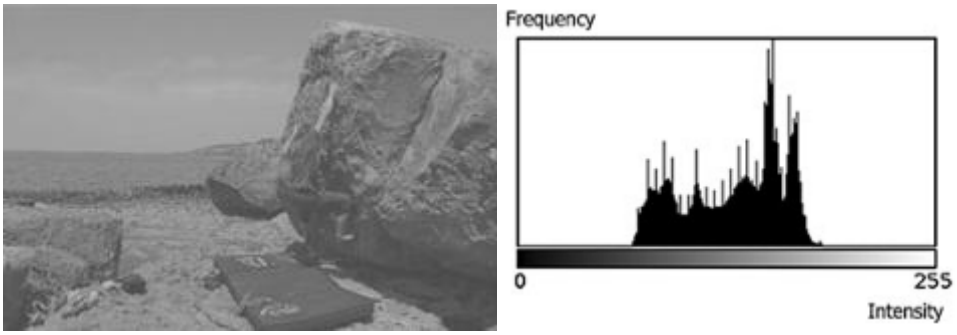
Fuente: Robot [en línea]. e-puck, 24 de Noviembre de 2010 [consultado 10 de Junio 2016]. Disponible en internet: http://www.e-puck.org/index.php?option=com_content&view=article&id=2&Itemid=8

4.1 PROCESAMIENTO DE IMAGEN

Para la detección de estos e-pucks y demás objetos, en la etapa de procesamiento se deberá utilizar diferentes técnicas para resaltar aspectos de la imagen y así extraer propiedades y características con diferentes objetivos como por ejemplo, la sustitución o sobre posición de objetos reales por virtuales o el desplazamiento de objetos virtuales a causa del movimiento de los robots de tal manera que sea coherente, es decir que se encuentren en la posición y orientación adecuada, desarrollando así una interacción lógica entre estos elementos la cual se encontrara dirigida por un usuario.

En la Figura 9 se puede observar el histograma de la imagen a la izquierda, el cual representa la frecuencia o número de pixeles (eje Y) en cada valor del nivel de intensidad de grises (eje X), en esta imagen se observa un histograma estrecho concentrado en el centro del rango por lo cual la imagen presenta un bajo contraste debido a que se utilizan pocos niveles y sus valores se encuentran muy cercanos.

Figura 9. Histograma estrecho

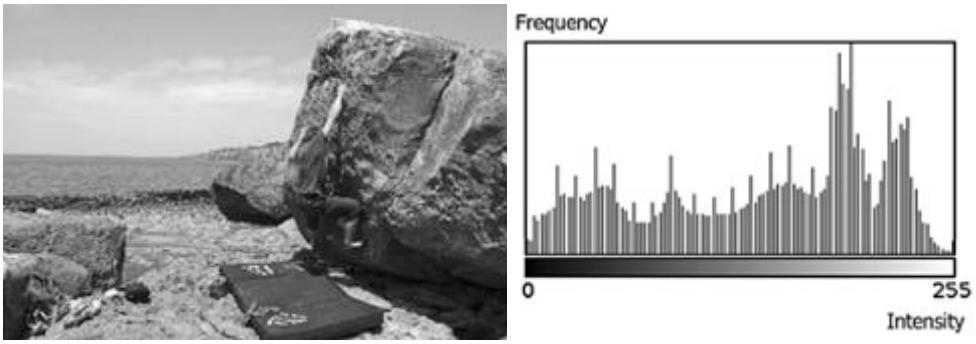


Fuente: Point Processing (Introduction to Video and Image Processing) Part 2 [en línea]. What-When-How [consultado 10 de Junio 2016]. Disponible en internet: http://what-when-how.com/introduction-to-video-and-image-processing/point-processing-introduction-to-video-and-image-processing-part-2/

La ecualización de histograma es una técnica del procesamiento de imagen que busca aprovechar todo el rango de valores de intensidad disponible en una imagen, esto causa que el histograma sea más uniforme distribuyendo las gamas de tonos que más aparecen en todo el rango de intensidades, como resultado se obtiene una grafico en el que las frecuencias, en todos los niveles de intensidad, son lo más similares posible generando así un mayor contraste al visualizar más intensidades en la imagen[7] lo que resalta detalles que antes no se veían como se puede observar en la Figura 10.

[7] RUIZ DEL SOLAR, Javier et al. Filtrado de Imágenes y ecualización de histograma [en línea]. Universidad de Chile, 24 de Septiembre 2015, p. 3-5 [consultado 28 de Octubre de 2016]. Disponible en Internet: https://www.u-cursos.cl/usuario/b4eb6d37062854338662ba7470704112/mi_blog/r/EL7008___Tarea_1.pdf

Figura 10. Histograma ecualizado



Fuente: Point Processing (Introduction to Video and Image Processing) Part 2 [en línea]. What-When-How [consultado 10 de Junio 2016]. Disponible en internet: http://what-when-how.com/introduction-to-video-and-image-processing/point-processing-introduction-to-video-and-image-processing-part-2/

Por otro lado la aplicación de filtros permite resaltar o suprimir, de manera selectiva, información o detalles de una imagen, para este proceso de filtrado se utiliza una matriz de tamaño NxN la cual se aplica a cada uno de los pixeles de la imagen generando un nuevo valor en función del valor del pixel actual y sus vecinos. Entre los filtros más comunes se encuentran los paso bajo (suavizan la imagen), paso alto (aumentan el contraste), filtros direccionales (detectan en la imagen estructuras que siguen una determinada dirección) y los de detección de bordes. Para este proyecto nos enfocaremos en los filtros paso bajo como método para reducir el ruido o valores indeseados en las imágenes procesadas debido a que al suavizar la imagen la gran variabilidad o valores atípicos de la imagen son eliminados, algunos ejemplos de filtro paso bajo son:

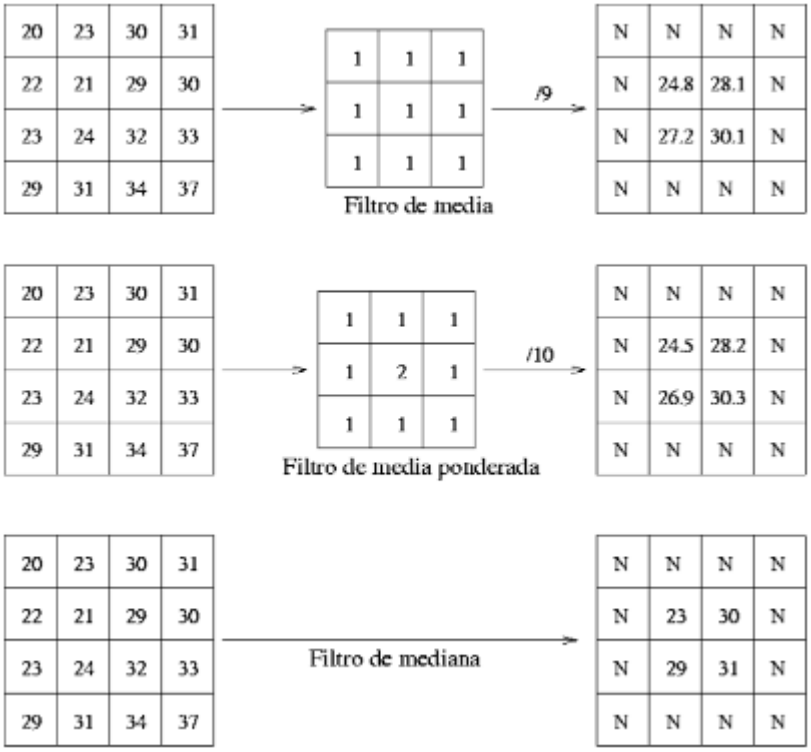
- **Filtro de la media o uniforme:** Este filtro se encarga de asignar al pixel la media o valor promedio de todos los pixeles incluidos en la ventana o matriz auxiliar, por esto la matriz está compuesta por unos y posteriormente se divide por el total de elementos en la ventana.

- **Filtro de media ponderada:** En este filtro a diferencia del anterior, no todos los elementos de la ventana presentan un valor de uno, debido a que se le otorga mayor peso a alguna posición de la matriz, usualmente es el pixel central obteniendo así una imagen más similar a la original.

- **Filtro de la mediana:** Este filtro asigna al pixel un valor real de la imagen debido a que no realiza un promedio sino que ordena los valores incluidos en la ventana y selecciona el valor central, sin embargo este resulta más complejo de calcular debido a que toma mayor tiempo reordenar los diferentes valores[8].

En la Figura 11 se puede observar un ejemplo de los tres filtros previamente mencionados utilizando una ventana de 3x3 en cada caso.

Figura 11. Histograma ecualizado



Fuente: Técnicas de filtrado [en línea]. Universidad de Murcia, p. 69 [consultado 8 de Enero de 2017]. Disponible en Internet: http://www.um.es/geograf/sigmur/teledet/tema06.pdf

Otra herramienta muy útil en el procesamiento de imagen son las operaciones morfológicas binarias, las cuales pueden simplificar datos de una imagen que contiene únicamente valores en ceros y unos, preservando características

[8] Técnicas de filtrado [en línea]. Universidad de Murcia, p. 67-69 [consultado 8 de Enero de 2017]. Disponible en Internet: http://www.um.es/geograf/sigmur/teledet/tema06.pdf

importantes o eliminando elementos indeseados, dependiendo de las formas detectadas en la imagen. Para realizar estas operaciones se utilizan imágenes binarizadas y un elemento estructural o kernel el cual funciona como las ventanas de los filtros, son matrices que se operan y desplazan por toda la imagen, entre estas operaciones las más comunes son:

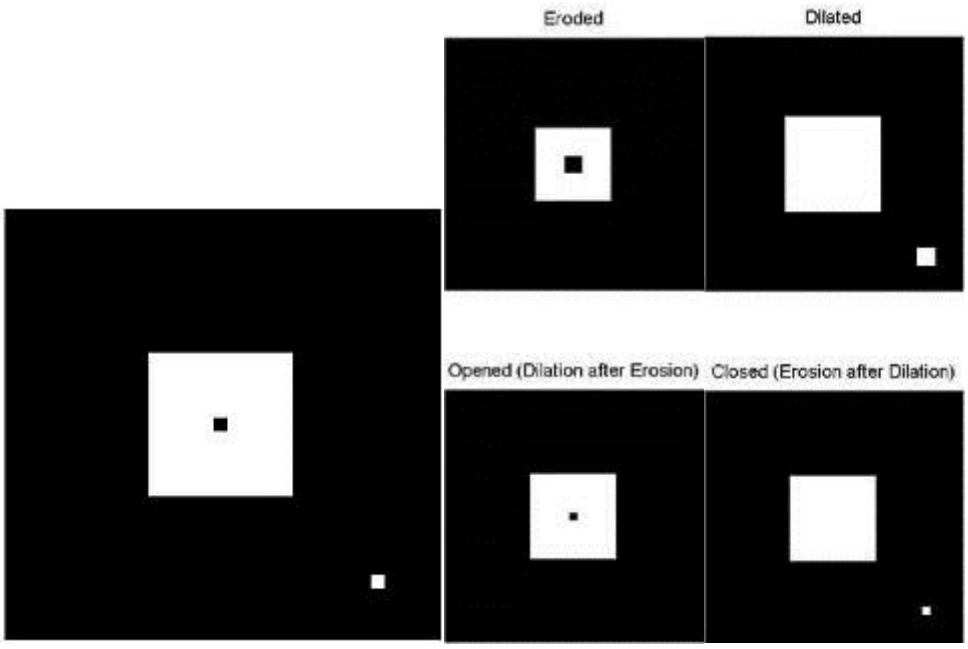
- **Erosión (Erode):** En esta operación el pixel que se está operando se considera uno (blanco) solo si todos los pixeles bajo el kernel también lo son de lo contrario es erosionado y se convierte en cero (negro), esto causa que los bordes de los elementos grandes se reduzcan o separen en secciones y que los elementos pequeños o ruidos se eliminen dependiendo del tamaño del kernel.

- **Dilatación (Dilation):** Esta operación es totalmente opuesta a la anterior, en este caso el pixel es asignado como uno (blanco) si hay por lo menos un pixel bajo el kernel que lo sea, por esta razón los elementos se expanden y los agujeros o ceros (negro) cercanos al objeto tienden a desaparecer dependiendo del kernel utilizado.

- **Apertura (Opening):** Este caso es la combinación de las operaciones anteriores utilizando primero la erosión para eliminar pequeños ruidos y luego la dilatación para restituir el tamaño de los objetos restantes. Como se observa en la Figura 12 el pequeño ruido de la esquina inferior derecha desaparece al realizarse la erosión y posteriormente al dilatar la imagen el cuadrado blanco recupera su tamaño y el agujero se disminuye nuevamente a su tamaño original.

- **Cierre (Closing):** Al contrario de la apertura esta operación primero realiza una dilatación para eliminar pequeños agujeros en grandes objetos y posteriormente se realiza una erosión para reducir el tamaño de los objetos nuevamente al original. En la Figura 12 se observa como al dilatar la imagen el agujero en el centro del rectángulo es eliminado y posteriormente al realizar la erosión ambos rectángulos blancos se reducen a su tamaño original.

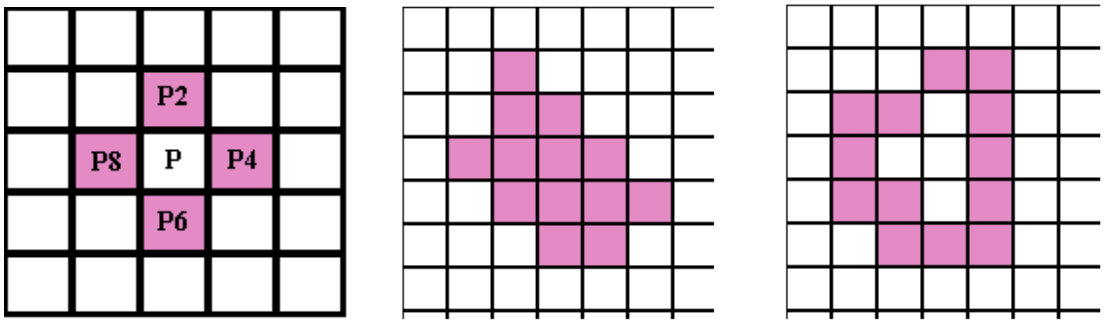
Figura 12. Operaciones morfológicas binarias



Fuente: Sensory Systems/Computer Models/Descriptive Simulations of Visual Informtion Processing [en línea]. Wikibooks, 2 de Diciembre de 2016 [consultado 8 de Enero de 2017]. Disponible en Internet: https://en.wikibooks.org/wiki/Sensory_Systems/Computer_Models/Descriptive_Simulations_of_Visual_Information_Processing

La conectividad entre pixeles es un concepto muy útil para etiquetar, definir o delimitar regiones de objetos en una imagen, la conectividad de dos pixeles se basa en la adyacencia, la vecindad y los niveles de gris. En la Figura 13.a se observa una vecindad tipo cuatro en donde se consideran solo los pixeles adyacentes de manera horizontal y vertical (P2, P4, P6 y P8), en caso de imágenes binarias, se revisa si estos pixeles presentan el mismo valor, cero o uno, del pixel central P para definir si pertenecen al mismo objeto, las Figuras 13.b y 13.c son ejemplos de objetos detectados con vecindad tipo cuatro.

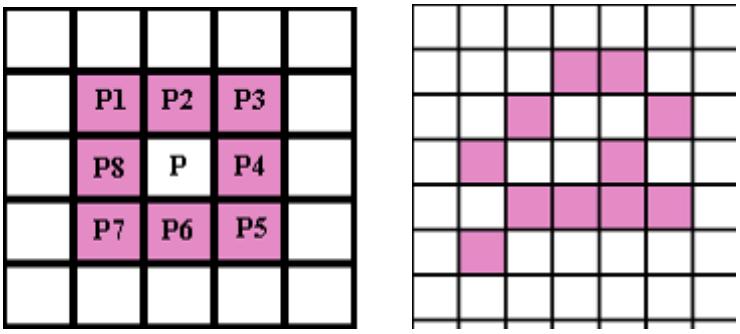
Figura 13. a) Vecindad tipo 4. b) y c) Ejemplos detectados



Fuente: Defining Connectivity [en línea]. Contour Tracing [consultado 08 de enero de 2017]. Disponible en Internet: http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/connectivity.html

En una vecindad tipo 8 como se observa en la Figura 14.a se tienen en cuenta los pixeles adyacentes no solo de manera horizontal y vertical sino también diagonal incluyendo los pixeles P1, P3, P5 y P7, de esta manera los objetos como el de la Figura 14.b que presentan secciones únicamente diagonales siguen perteneciendo a un mismo elemento debido a la vecindad seleccionada.

Figura 14. a) Vecindad tipo 8. b) Ejemplo detectado

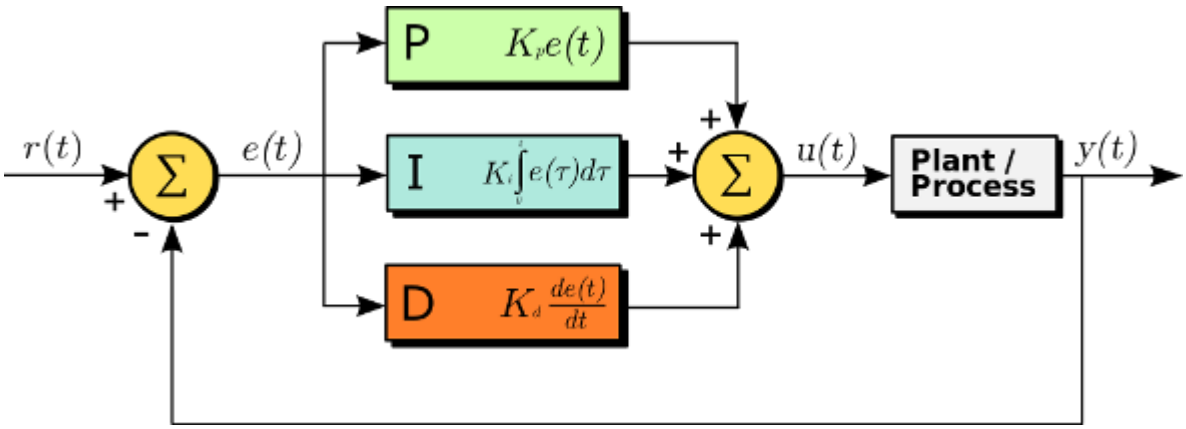


Fuente: Defining Connectivity [en línea]. Contour Tracing [consultado 08 de enero de 2017]. Disponible en Internet: http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/connectivity.html

4.2 CONTROL PID

Este tipo de control de lazo cerrado es el más utilizado en el control industrial debido a su buen desempeño en una gran variedad de condiciones, este control consiste en recibir un error como entrada, usualmente la diferencia entre una referencia y la lectura de un sensor, para posteriormente calcular, a través de la suma de sus tres componentes proporcional, integral y derivativa, la salida adecuado para obtener el comportamiento deseado en un actuador, como se observa en la Figura 15 la salida del actuador (Plant/Process) $y(t)$ es comparada con el valor de referencia para obtener el error, esta realimentación hace de este control uno de lazo cerrado y es aquí donde usualmente se utiliza un sensor intermedio para medir la salida y así calcular el error.

Figura 15. Diagrama de bloques de un PID.



Fuente: PID controller [en línea]. Wikipedia [consultado 08 de enero de 2017]. Disponible en Internet: https://en.wikipedia.org/wiki/PID_controller

Cada uno de los componentes del PID presenta una función diferente:

- **Proporcional:** Este componente calcula un valor proporcional al error $e(t)$ utilizando una ganancia ajustable K_p la cual si es muy alta puede generar oscilaciones que inestabilicen el sistema.

- **Integral:** La salida de este componente es proporcional en un valor K_I al error acumulado por lo cual incluso un pequeño error causa que este término aumente eliminando por completo el error en estado estacionario.

- **Derivativo:** Este término es proporcional a la tasa de cambio de la variable a controlar, se encarga de hacer más rápida la acción de control produciendo una corrección significativa antes de que el error crezca demasiado y disminuyendo la salida en caso que la variable de control este aumentando muy rápido, sin embargo debido a que amplifica señales de ruido que podrían provocar la saturación en el actuador su coeficiente K_D generalmente es muy bajo[9].

4.3 CONCEPTOS PARA EL DESARROLLO DE VIDEOJUEGOS

- **Sprites:** Son objetos que aparecen en el video juego, desarrollado usualmente con una variedad de atributos de los cuales los esenciales son una imagen y una posición en pantalla (coordenadas). Estos sprites pueden se estáticos, como un árbol u objeto decorativo en el juego, o dinámicos como un personaje controlado por el usuario o un enemigo controlado por la computadora. El movimiento de estos sprites dinámicos funciona igual que el de la animación, es decir una secuencia de imágenes dibujadas en secuencia en la pantalla, el número de imágenes por unidad de tiempo, usualmente cuadros por segundo, es el factor que definirá que tan fluido es el movimiento.

- **Superficie (Surface):** Una superficie hace referencia a un espacio de memoria en el cual se almacena una imagen, es decir que consiste en una estructura o arreglo de datos que contienen cada uno de los pixeles que componen una imagen, este término no se debe confundir con un sprite ya que este último contiene como atributo una superficie.

- **Blitting (Block Image Transfer):** Es una operación que permite transferir una superficie de un espacio de la memoria a otro, es decir renderizar o dibujar las imágenes de los sprites sobre un fondo en la pantalla del videojuego. A través de esta operación es que se puede crear el movimiento de objetos utilizando las coordenadas de origen y destino del sprite.

[9] PID Theory Explained [en línea]. National Instruments, 29 de Marzo de 2011 [consultado 08 de enero de 2017]. Disponible en Internet: http://www.ni.com/white-paper/3782/en/

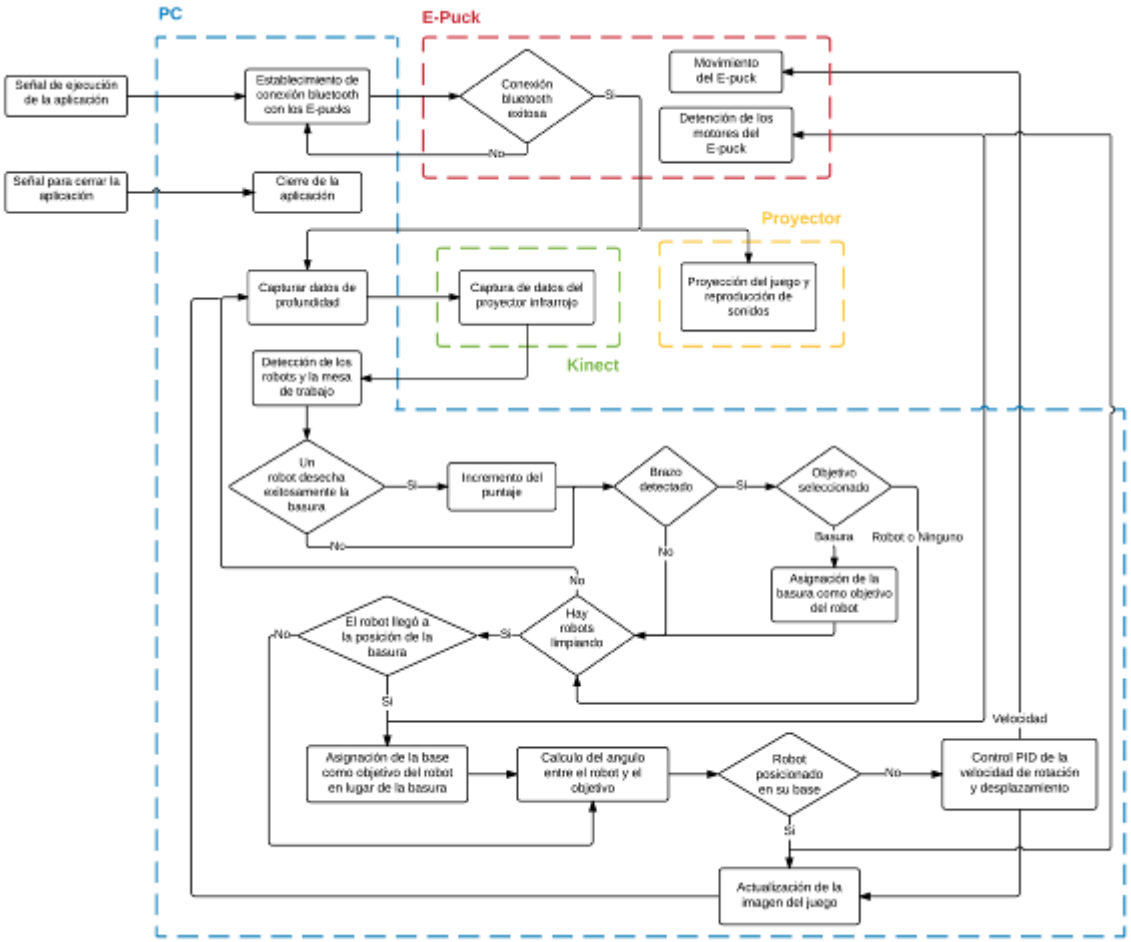
5. RESULTADOS

5.1 VISIÓN GLOBAL DEL PROYECTO IMPLEMENTADO

La aplicación desarrollada es un juego que fomenta el adecuado reciclaje de los desechos permitiendo a los niños interactuar con objetos virtuales y robots E-pucks. Para este desarrollo se hará uso de un sensor Kinect, un proyector, cuatro robots E-pucks y un computador. El sensor Kinect es esencial para este proyecto debido a que permite tener una visión artificial a través de los datos obtenidos por medio del proyector infrarrojo, de esta manera con el mapa de profundidades obtenido se puede detectar el área de trabajo y cada uno de los robots E-pucks, por esta razón el sensor Kinect se debe ubicar en el techo paralelo a la mesa de trabajo para obtener una correcta visión del espacio de trabajo y los gestos del usuario. El proyector es utilizado para sobreponer la imagen del juego sobre la superficie de trabajo donde se encontraran los robots E-pucks, de esta manera el mundo virtual se acerca al usuario y permite a los robots interactuar con este. Los robots E-pucks son los trabajadores dirigidos por el usuario que se encargaran de limpiar el mar de los desechos virtuales y por último el computador es el que permite integrar los diferentes dispositivos sirviendo como puente entre el mundo virtual (escenario proyectado) y el mundo real (robots y usuario) a través de diferentes algoritmos y cálculos realizados con base a la información del proyector y los datos obtenidos del Kinect.

A continuación se presenta el diagrama de flujo implementado en el juego desarrollado (Figura 16). En este diagrama se puede observar de manera general los procesos que se llevan a cabo para el funcionamiento de la aplicación.

Figura 16. Diagrama Flujo.



La interfaz inicial con la que debe interactuar el usuario es el computador para correr la aplicación desarrollada, después de esto la interacción del usuario será únicamente a través de movimientos detectados por el sensor Kinect. Por medio de bluetooth se establece la conexión para poder transmitir datos de velocidad a los cuatro robots e-pucks, haciendo uso del protocolo RFCOMM. Una vez la conexión con los cuatro robots se ha establecido, el escenario y las animaciones del juego, con las cuales interactuara el usuario, son proyectadas y los sonidos reproducidos para posteriormente obtener del sensor Kinect los datos de profundidad, es en este punto donde comienza el ciclo repetitivo de la aplicación debido a que en el inicio de cada ciclo se obtienen los datos de profundidad y se puede observar el cambio de posición de los robots. A partir de este punto se utiliza técnicas de procesamiento de imagen para poder detectar del mapa de profundidades donde se encuentra la mesa (espacio de trabajo) para así eliminar

información innecesaria y calcular límites basados en los datos obtenidos de la mesa que faciliten la detección de los robots e-puck. Una vez se han detectado las cuatro áreas de los robots es posible calcular las coordenadas (en pixeles) y la orientación debido a que utilizando una cubierta encima de cada uno de los robots se le proporciona un marcador con una saliente que sobresale de la circunferencia de los e-pucks permitiendo calcular el ángulo en el que se encuentran. Debido a que se tiene la posición exacta de todos los robots se debe revisar si alguno se encontraba en proceso de recoger una basura y la ha desechado correctamente para así poder retroalimentar al usuario con su progreso por medio del puntaje de aciertos y a su vez actualizando el nivel de contaminación en el agua, el cual depende de la cantidad de basuras no recogidas y el tiempo transcurrido.

En la siguiente etapa se calculan límites a partir de los datos de profundidad de la mesa para poder detectar si se encuentra en el área de trabajo el brazo de un usuario entre las alturas delimitadas y de esta manera, exclusivamente con el área del brazo, se calcula un vector 3D con el cual se halla la coordenada en pixeles a la que el usuario está apuntando. Una basura será seleccionada en caso de que se encuentre dentro de un radio delimitado de pixeles alrededor de la coordenada calculada, de ser así esta será asignada al robot previamente activado (seleccionado por el usuario) para ser desechada, por otro lado si ningún brazo es detectado no se realizaran los procesos sino que se procederá directamente a revisar si un robot se encuentra actualmente en proceso de limpieza. En caso de no encontrar ningún robot a cargo de una basura el ciclo se reinicia para obtener los nuevos datos de profundidad y realizar todos los cálculos de nuevo, sin embargo en caso contrario de que si se encuentre uno o más robots en proceso de limpieza es necesario continuar el ciclo para monitorear el proceso. Se procede a revisar cada uno de los robots activos para saber si ha alcanzado las coordenadas donde se encuentra la basura que tiene como objetivo para así detener el robot ya que ahora debe desplazarse en otra dirección y actualizar el objetivo al que debe dirigirse sustituyendo las coordenadas de la basura por las de la base del robot en donde se desechan los objetos recolectados. Una vez se han actualizado los objetivos necesarios se detecta si algún robot se encuentra en la base con la basura recolectada, de ser así se detiene el e-puck, se actualiza el escenario proyectado y se reinicia el ciclo, en caso contrario se procede a calcular el ángulo entre cada e-puck y su objetivo, utilizando este dato y la distancia entre los dos para calcular por medio de un PID la velocidad de desplazamiento y rotación. Esta velocidad se envía por medio del bluetooth al e-puck y se actualiza el escenario del juego para empezar de nuevo el ciclo principal y obtener los datos de profundidad actualizados.

5.2 IMPLEMENTACIÓN Y RESULTADOS ESPECÍFICOS

A continuación se presentan las herramientas, procedimientos y cálculos finalmente implementados para obtener las características deseadas y el mejor desempeño en el resultado final de la aplicación. Este proyecto fue desarrollado en Python debido a que este lenguaje de programación presenta librerías para programar los robots E-puck, el sensor Kinect y a la vez desarrollar videojuegos por medio de Pygame, facilitando de esta manera la programación de esta aplicación al trabajar solo con un lenguaje para integrar los diferentes dispositivos.

5.2.1 Lógica del juego. La aplicación desarrollada consiste en limpiar la basura del mar reciclando correctamente los desechos haciendo uso de cuatro robots. El usuario debe señalar a un E-puck para activarlo y luego señalar una basura del tipo que ese robot especifico puede limpiar (Papel, latas, plástico o vidrio) para que este se dirija a desecharla. Si la basura seleccionada y el robot no son del mismo tipo de desecho el robot no podrá recogerla y el puntaje no aumentara. A medida que pasa el tiempo las basuras aparecen cada vez más rápido e irán contaminando el mar aumentando el nivel de contaminación. El juego se termina cuando el usuario envía a dos robots a limpiar en una trayectoria en la que podrían colisionar o cuando la contaminación llega un nivel crítico, el cual está actualmente establecido a 40 puntos de contaminación. El usuario gana si sobrevive 3 minutos jugando sin alcanzar los niveles críticos de contaminación. En la Figura 17 se puede observar en el escenario del juego los diferentes elementos con los que se interactúan.

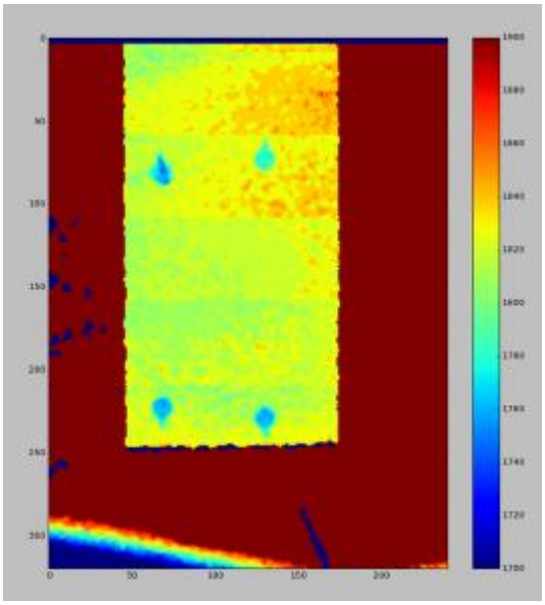
Figura 17. Escenario del juego.

5.2.2 Configuración del proyector y el Kinect. El sensor Kinect debe ubicarse en el techo con la cámara dirigida y paralela a la mesa de trabajo para poder obtener las imágenes de profundidad de toda la superficie de trabajo donde se encuentran los robots y donde se encontrara el brazo del usuario para interactuar con la aplicación. El proyector se debe ubicar también en el techo paralelo a la mesa para evitar una distorsión de la imagen proyectada y en una posición donde el sensor Kinect no interfiera con la trayectoria de la proyección como se puede observar en la Figura 18.b. Al ubicar la mesa debajo del proyector esta se debe posicionar de tal manera que la esquina superior izquierda de la imagen proyectada coincida con la esquina de la mesa (Figura 18.a) para que el punto de referencia cero del Kinect y de la pantalla este ubicado en el mismo punto.

Figura 18. a) Montaje general. b) Orientación del proyector y el sensor Kinect.

El sensor Kinect debe encontrarse lo más paralelo posible a la mesa de trabajo para obtener datos de profundidad homogéneos sobre la superficie de está y de esta manera lograr una mayor diferenciación entre la mesa y los robots como se observa en la Figura 19.

Figura 19. Imagen de profundidad de la mesa con 4 robots.



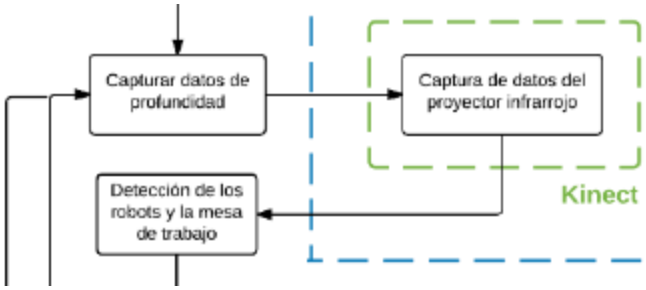
5.2.3 Programación del Kinect. *SDK 1.8:* Para tener acceso a la información obtenida por medio del sensor Kinect y desarrollar aplicaciones que soporten el reconocimiento de gestos y de voz, es necesario instalar el kit de desarrollo de software para Windows el cual contiene los drivers necesarios para el uso del sensor.

Pykinect: Para programar el Kinect en Python se utilizó un paquete llamado Pykinect el cual permite acceder al sensor y utilizar las cámaras del Kinect como detector de esqueletos, video cámara normal o cámara de profundidad. Debido a que los ejemplos y demos de Pykinect no se encontraban actualizados generando errores en la obtención de datos de profundidad se utilizó como base el código encontrado en el perfil de Bitbucket Vito Gentile, el cual permite la correcta conexión con el receptor infrarrojo.

5.2.4 Configuración y software de los E-pucks. Para controlar los e-pucks desde el computador es necesario descargar al robot el firmware estándar para después poder programarlo haciendo uso de la librería ePuck for Python desarrollada por ITRB Labs la cual permite leer los sensores de los robots y utilizar sus actuadores. La conexión entre el computador y un robot e-puck puede tomar tiempo por lo cual a veces puede haber un error debido a una falla en la conexión, por esta razón fue necesario editar la librería para permitir que el computador reintente conectar aun después de varios intentos fallidos, de esta manera se puede realizar la conexión de varios e-puck exitosamente sin tener que correr la aplicación cada vez que la conexión falle. Adicionalmente se agregaron unos métodos y variables para permitir el adecuado control de velocidad de cada e-puck por medio de un PID.

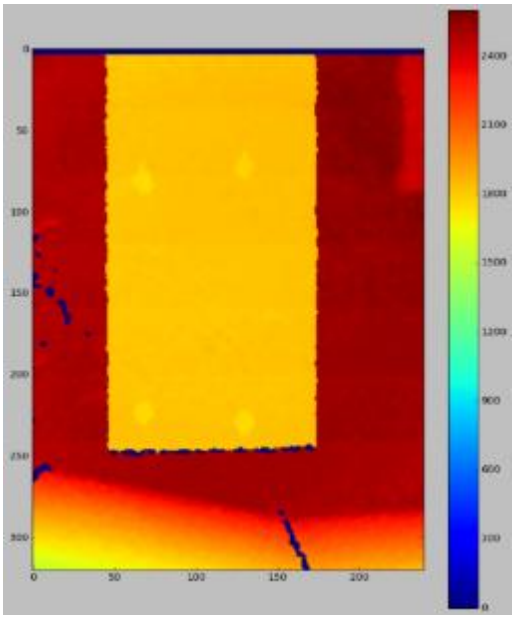
5.2.5 Detección de los E-pucks. En este numeral nos enfocaremos en la sección del diagrama de flujos que se puede observar en la Figura 20, debido a que se utilizaran los datos obtenidos por medio del emisor infrarrojo para poder detectar los robots y la mesa de trabajo.

Figura 20. Sección del diagrama de flujo.



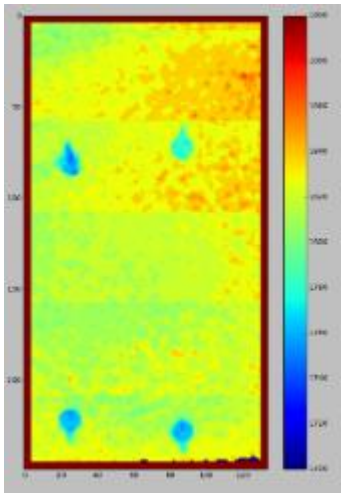
Los valores de profundidad obtenidos del Kinect son menores al estar más cerca de la cámara de este (el techo) y mayores al estar más cerca del suelo, sin embargo para esta aplicación nuestro rango de interés se encuentra entre la cámara del sensor y la mesa, el resto de objetos por debajo de la mesa se pueden ignorar. En la Figura 21 se puede observar la imagen inicial de profundidad obtenida del Kinect, debido a que el rango es bastante amplio e incluye el suelo, los robots no se pueden diferenciar tan evidentemente.

Figura 21. Imagen inicial de profundidad de la mesa.



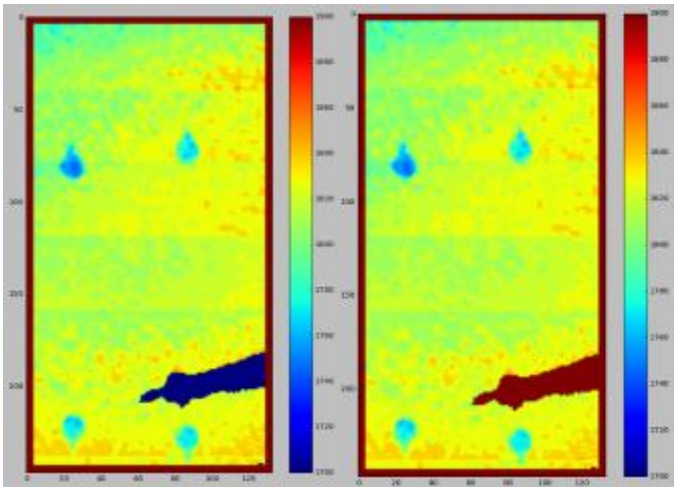
Como se observa en la Figura 16 el ciclo principal del juego que se repite inicia con la actualización de los datos de profundidad por esta razón el procesamiento de imagen debe optimizarse para que tome el menor tiempo posible y de esta manera lograr un control más preciso de la posición de los e-pucks y una agradable experiencia de usuario al disminuir el tiempo de respuesta a sus acciones haciendo el juego más fluido y natural. Por lo anterior se recortó la imagen para analizar solo el área donde se encuentra la mesa como se observa en la Figura 22, para realizar este recorte se extrajo un rectángulo donde se encontraron valores de profundidad en el rango deseado, este rango se creó utilizando como referencia el valor promedio de los datos en el suelo (distancia > 2100 mm) y a partir de este dato se calcularon dos valores proporcionales que determinan entre que valores de distancia se encuentra la mesa y los robots, esto se realizó con el fin de que la detección de la mesa no se vea afectada en caso de que el sensor no se encuentre siempre ubicado exactamente a la misma altura (el Kinect debe estar ubicado a un máximo de dos metros sobre la mesa) o en caso de que los valores de profundidad varíen un poco debido a que la cámara infrarroja se puede ver afectada por la luz y no entrega valores muy estables.

Figura 22. Recorte de la mesa de trabajo.



Para la detección de los robots estos deben ser los objetos más cercanos al sensor Kinect, sin embargo el brazo del usuario siempre se detectara por encima de los robots por lo cual se define un rango, proporcional a la profundidad del suelo, que se encuentra por encima de los e-pucks, a todos los objetos detectados en este rango se les asigna el valor del suelo (Figura 23) y de esta manera el brazo no será erróneamente detectado como un robot.

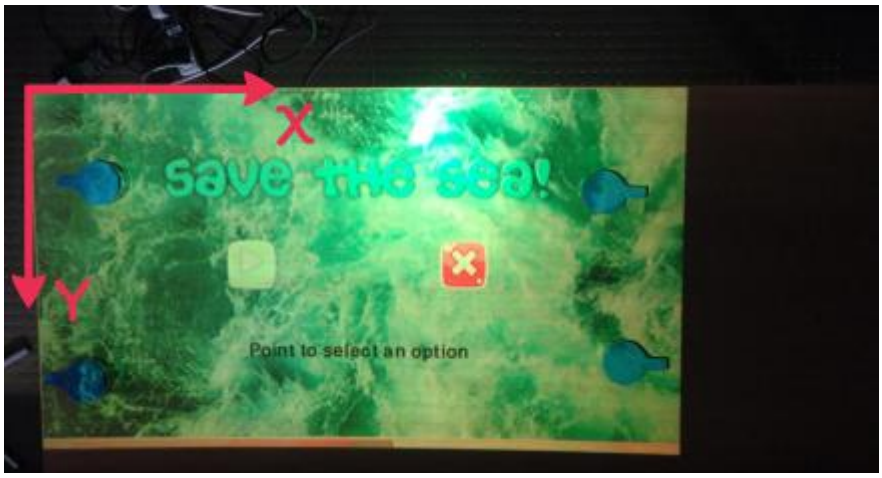
Figura 23. Eliminación del brazo del usuario.



Debido a que el punto de referencia cero de la mesa está ubicado en la parte superior izquierda, donde se encuentran reunidos los e-pucks (Figura 24), es

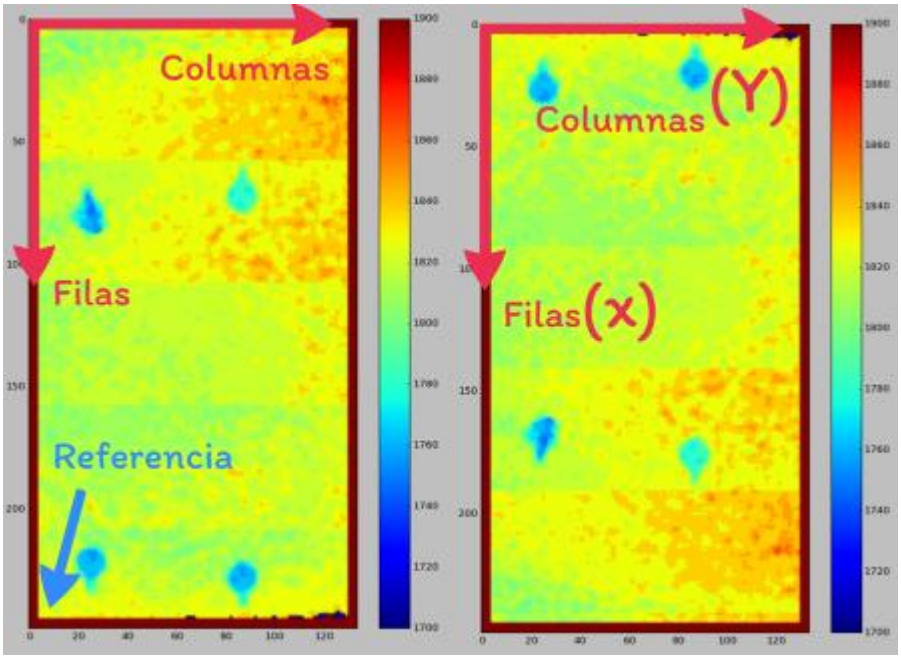
necesario mover el punto de referencia en la imagen del kinect para que estos coincidan.

Figura 24. Imagen de la mesa invertida verticalmente.



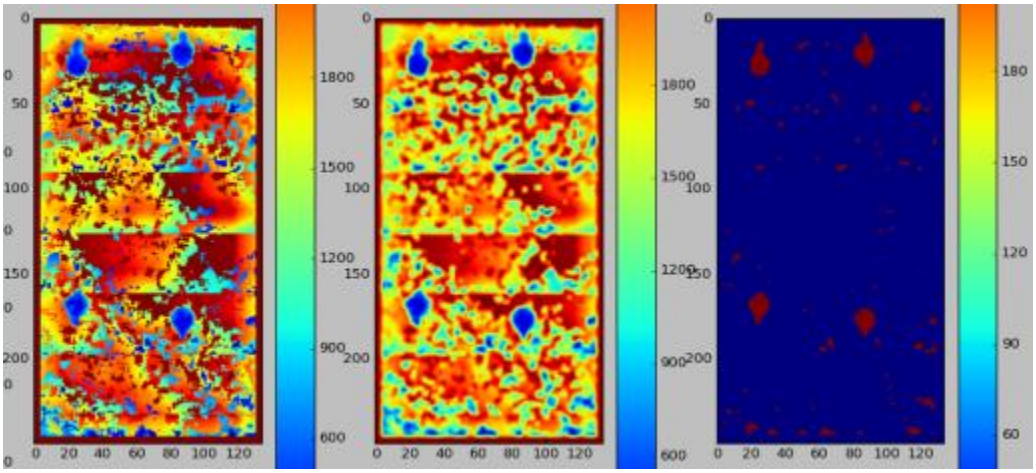
Como se observa en la Figura 25.a el punto de referencia cero de la mesa se encuentra en una esquina diferente a la deseada pues los e-pucks se encuentran reunidos en la base de la imagen siendo la esquina inferior izquierda donde debería encontrase ubicado el punto de referencia cero. Adicionalmente al mostrar la imagen de profundidad por medio de la pantalla de Pygame se observa que la imagen aparece invertida, esto se debe a que las matrices de la imagen se indexan [filas,columnas] mientras que la pantalla se indexa en coordenadas [X,Y] utilizando el sistema de referencia de la Figura 24, por esta razón al utilizar la pantalla de Pygame las filas se indexan en el eje X y las columnas en el eje Y resultando en una imagen o matriz transpuesta. Para unir los dos sistemas de referencia es necesario invertir la imagen de profundidad verticalmente para obtener el punto de referencia en la esquina adecuada (Figura 25.b), donde se encuentran los e-pucks, y de esta manera al indexar la imagen de profundidad en la pantalla de Pygame las filas coinciden correctamente con el eje X y las columnas con el eje Y.

Figura 25. a) Mesa con referencia errónea. b) Mesa invertida verticalmente.



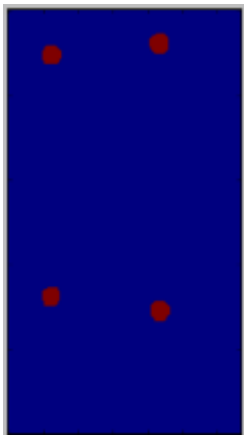
Una vez el brazo del usuario ha sido ignorado y la imagen invertida, para reubicar el punto de referencia cero, se puede proceder a separar los robots de la mesa, primero se realiza una ecualización de histograma local con un kernel circular de tal manera que al distribuir más uniformemente los valores de profundidad, se genere un mayor contraste permitiendo realzar el área de los e-pucks como se puede observar en la Figura 26.a. Sin embargo debido a la ecualización se generan ruidos de menor área que los robots los cuales deben eliminarse, por esta razón se aplica un filtro local que promedia los valores dentro de una ventana 3x3 (Figura 26.b) y luego se saca la profundidad promedio sobre la mesa la cual al dividirse por un valor hallado experimentalmente permite definir un límite para eliminar todos los datos por encima de este y asignar el valor de 255 a los objetos restantes detectados obteniendo así la Figura 26.c.

Figura 26. a) Resultado de la ecualización. b) Filtro de la media. c) Binarización.



Una vez se han separados los robots del fondo se procede a hacer uso de los módulos de procesamiento de imagen de Python, Scikit-Image y Scipy.ndimage. Fue necesario hacer uso de las operaciones morfológicas binarias apertura y cierre, en este mismo orden, con el fin de eliminar el ruido detectado como objetos y posteriormente aumentar el tamaño de los robots para obtener un área similar al tamaño real del e-puck como se observa en la Figura 27 la cual no presenta áreas indeseadas.

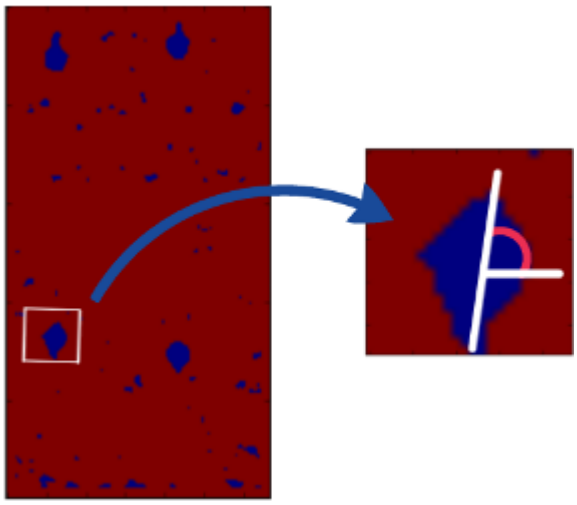
Figura 27. Imagen resultante después de las operaciones morfológicas.



Al obtener la imagen de los robots lo más limpia posible se procede a utilizar la conectividad de los pixeles utilizando una vecindad tipo 4 para identificar y

etiquetar cada una de las áreas restantes. Debido a la fluctuación en los datos de profundidad, en ciertos casos aun después de las operaciones morfológicas se presentan pequeñas áreas indeseadas por esta razón fue necesario crear un último filtro, se obtuvo la mayor área de todos los objetos detectados, la cual siempre pertenecerá a un robot, y se filtran todas las áreas restantes con la condición de que todas las áreas menores al 60% del área máxima serán eliminadas y el resto serán definidas como e-pucks. Una vez se obtienen las coordenadas de los cuatro robots se procede a extraer la orientación de una imagen previamente binarizada con un límite mayor permitiendo más áreas indeseadas pero a su vez obteniendo el área completa de los robots incluyendo el marcador con la saliente, el cual es necesario para determinar la orientación del e-puck. Como se observa en la Figura 28 utilizando las coordenadas halladas se extrae un rectángulo de la imagen con áreas indeseadas, debido a que en el rectángulo extraído puede presentarse ruido, como se observa en la parte superior derecha del rectángulo de la Figura 28.b, se selecciona la mayor área como el robot para extraer la orientación por medio de la función Regionprops del módulo Scikit-Image el cual permite extraer datos como las coordenadas del centroide, el área y la orientación, esta última se calcula mediante el ángulo entre el eje más largo que atraviesa el área y el eje x de la mesa, como se observa en la Figura 28.b.

Figura 28. a) Extracción del e-puck. b) Obtención de la orientación del robot.



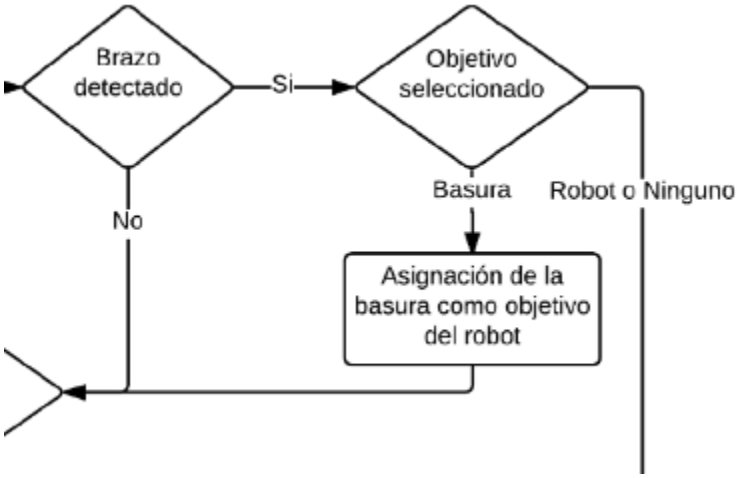
Para poder trasladar las coordenadas de los e-pucks de la imagen del kinect a las coordenadas de la pantalla es necesario crear la siguiente ecuación para la conversión:

$$R = \frac{A_p}{P_m * (m_p/m)} \qquad (1)$$

Donde R es la relación entre las coordenadas de los pixeles del kinect y la pantalla del juego; A_p es el ancho en pixeles de la pantalla del juego; P_m es el ancho de la imagen proyectada sobre la mesa en mm; m_p es el alto de la mesa en pixeles detectado con el Kinect y m es el alto de la mesa en mm. Los valores del alto de la mesa y el ancho de la proyección en milímetros deben ser suministrados al principio del código en las variables globales para así poder calcular este factor y lograr que la proyección del juego sea coherente con las posiciones de los robots.

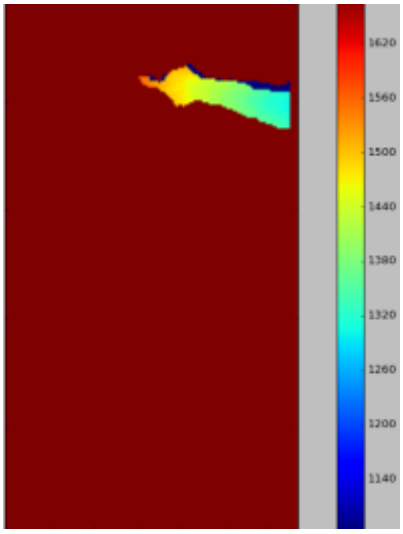
5.2.6 Detección del usuario. Debido a la posición del Kinect la única parte visible del usuario son los brazos cuando este se encuentra seleccionando un robot o basura, este numeral se enfoca en la sección del diagrama de flujos en la Figura 29, la cual incluye la detección del brazo y el proceso de selección para poder realizar el cálculo de las coordenadas a las que señala el usuario.

Figura 29. Sección del diagrama de flujo.



Para detectar el brazo del usuario se utiliza la imagen original del recorte de la mesa antes de aplicar cualquier filtro (Figura 22) para así filtrar la imagen creando un rango de profundidades proporcional al suelo el cual detecta los objetos por encima de los robots obteniendo así únicamente el brazo del usuario en caso de que se encuentre seleccionando como se observa en la Figura 30.

Figura 30. Detección del brazo.

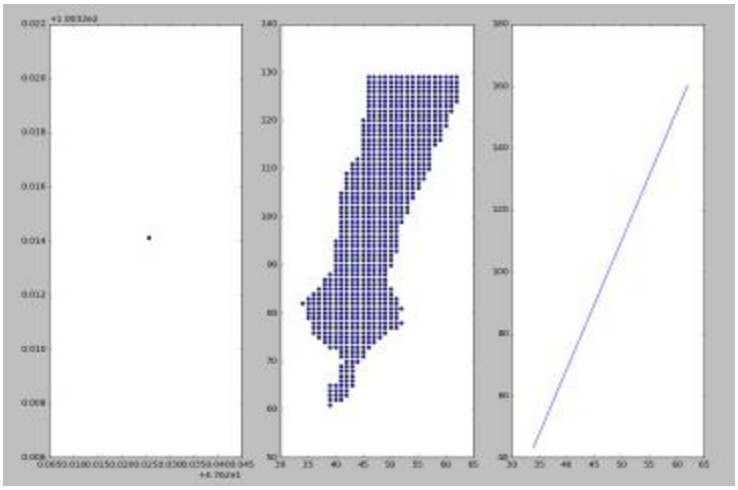


Por medio del uso de Regionprops se obtiene la orientación y el centroide del área detectada como el brazo para así proceder a calcular la ecuación de la recta (Ecuación 2), que describe el brazo en el plano XY, obteniendo la pendiente a partir del ángulo (Ecuación 3) y despejando el intercepto al sustituir las coordenadas X, Y del centroide y la pendiente hallada, de esta manera se obtiene el vector que se observa en la Figura 31.c.

$$y = mx + b \qquad (2)$$

$$m = \tan(\theta) \qquad (3)$$

Figura 31. a) Centroide. b) Área del brazo. c) Recta del brazo.



Debido a que los valores en Z que se tienen del brazo se encuentran en mm es necesario pasar estos datos a pixeles para poder hacer una regresión lineal en 3D, para esto se utiliza la siguiente ecuación:

$$z_p = (h - z) * \frac{m_p}{m} \qquad (4)$$

Donde z_p es la altura de los puntos en el brazo en pixeles; h es la altura a la que se encuentran los e-pucks en mm; z es la altura de los puntos en el brazo en mm; m_p es el alto de la mesa en pixeles detectado con el Kinect y m es el alto de la mesa en mm. El punto de referencia cero de la profundidad se encuentra en el Kinect y los valores de profundidad aumentan a medida que el brazo se acerca a los e-pucks por esta razón en la Ecuación 4 se restan los datos de profundidad al valor de profundidad al que se encuentran los e-pucks, para que el nuevo punto de referencia cero sea la superficie de los e-pucks.

Una vez se tienen todos los datos X, Y, Z en pixeles, se crea un diccionario con estos valores y así, por medio del módulo StatsModels de Python, se calcula una regresión lineal con dos variables independientes para así poder obtener un plano en el espacio (3D), de esta regresión se obtienen los coeficientes A, B y C que describen la Ecuación 6 que caracteriza el plano en el que se encuentra el vector que se desea hallar. Las coordenadas a las que se encuentra apuntando el vector 3D obtenido del brazo del usuario se obtienen al despejar de la ecuación de la recta, hallada en el plano XY (Figura 31), la coordenada X y posteriormente

sustituir el resultado (Ecuación 5) en la Ecuación 6 que representa un plano, obteniendo así la Ecuación 7 que se observa a continuación:

$$y = mx + b \qquad (2)$$

$$x = \frac{y - b}{m} \qquad (5)$$

$$z = Ax + By + C \qquad (6)$$

$$0 = A\left(\frac{y - b}{m}\right) + By + C$$

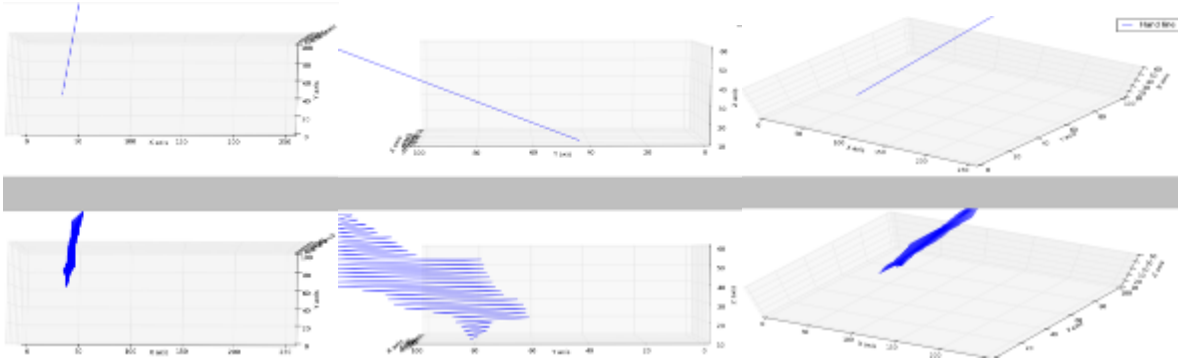
$$By = \frac{-Ay}{m} + \frac{Ab}{m} - C$$

$$y\left(\frac{Bm + A}{m}\right) = \frac{Ab - Cm}{m}$$

$$y = \frac{Ab - Cm}{Bm + A} \qquad (7)$$

Debido a que lo valores X,Y de entrada a la ecuación del plano pertenecen a la recta que describe el brazo (Figura 31), la Figura obtenida en el espacio es un vector 3D y no un plano. Al definir el valor en Z igual a cero se asegura que la coordenada X, Y (Ecuación 5 y 7 respectivamente) que se halle sea la proyección del vector del brazo hasta la superficie de los robots, en la Figura 32 se puede observar el vector 3D obtenido, en la parte superior de la Figura, comparándolo a la superficie del brazo detectado, en la parte inferior, desde diferentes perspectivas.

Figura 32. Vector 3D del brazo.

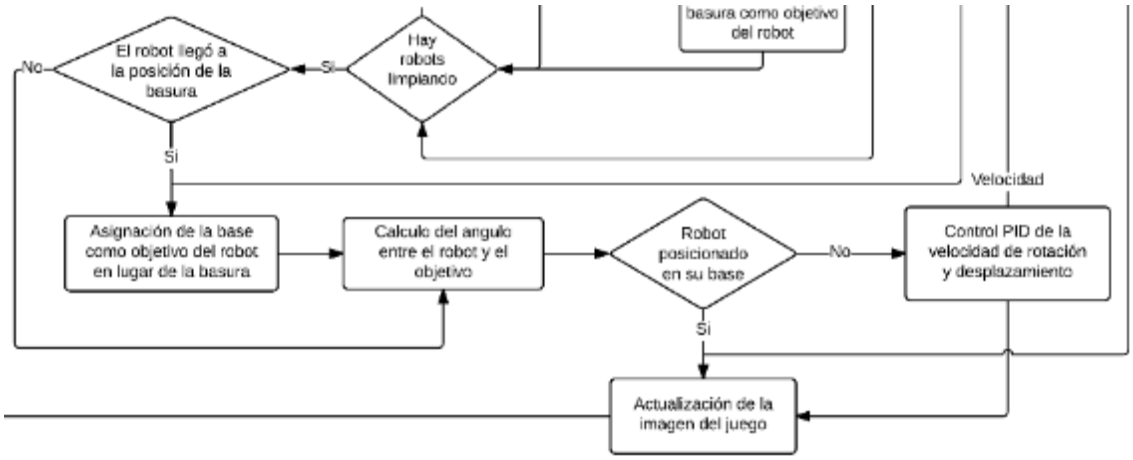


Con las Ecuaciones 5 y 7 se obtienen las coordenadas a las que apunta el usuario y de esta manera se calcula la distancia desde este punto a todos los posibles objetivos a seleccionar (robots o basuras), los cuales se encuentran ya almacenados en un arreglo. Una vez se tienen todas las distancias se extrae la mínima y debido a que el cálculo del vector no es muy exacto se da un margen de error de 50 pixeles, es decir si la distancia mínima hallada es menor a 50 pixeles se acepta la selección del objeto, en caso contrario se toma como que el usuario no está seleccionando. Debido a que el usuario debe desplazar su brazo por la mesa hasta apuntar al objeto deseado, no se puede tomar la primera detección del brazo como el objetivo definitivo a seleccionar por esta razón siempre se proyecta un puntero que indica al usuario a donde está apuntando y en un vector se va almacenando las selecciones efectivas que el usuario ha realizado de manera consecutiva. Para activar un robot o la recolección de una basura la selección se divide en dos etapas, pre-selección y selección definitiva, en la pre-selección primero se revisa que el objetivo al que se apunta no sea el que fue definitivamente seleccionado en el ciclo anterior ya que en este caso el usuario aún no ha bajado el brazo de su selección previa, posteriormente si el usuario ha seleccionado diez veces algún objetivo, lo cual puede tomar alrededor de 1,5 segundos, y de estas diez selecciones siete son un mismo objetivo se pre-selecciona este objetivo fijando el puntero a este aunque el usuario presente un leve movimiento o vibración en el brazo, en caso de que el objetivo que más se repite en el vector de selección no se haya seleccionado un mínimo de siete veces el contador de selecciones se reinicia y el proceso empieza de nuevo. Una vez se ha fijado el puntero en un objetivo específico empieza la etapa de selección definitiva, en esta se compara si después de doce selecciones más, es decir un total de veintidós selecciones, el objetivo más seleccionado sigue siendo el pre-seleccionado, en caso de que el usuario haya cambiado de decisión sobre cual objetivo seleccionar y en medio del proceso apunta a otro objetivo, la selección es cancelada y el proceso de selección empieza desde la primera etapa con un nuevo objetivo, por otro lado en caso de que el objetivo más seleccionado siga

siendo el mismo objetivo de la etapa anterior y supere dieciséis selecciones de las veintidós realizadas en total, la selección definitiva es correcta y se activan los leds del robot seleccionado y se asigna la basura escogida al robot previamente activado como se observa en sección del diagrama de flujos en la Figura 31.

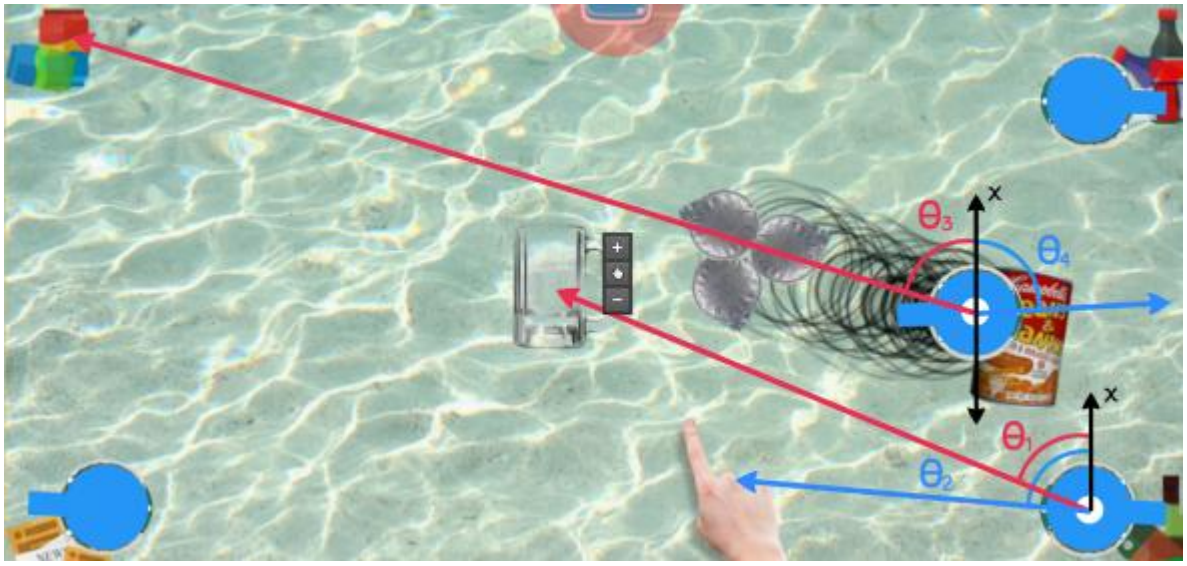
5.2.7 Desplazamiento de los E-pucks. A continuación se observa en la Figura 33 la sección del diagrama de flujo en la que se centrara este numeral, ya que se observara como se realizan los cálculos de los ángulos parar el desplazamiento y el PID para el control de velocidad y orientación de los robots.

Figura 33. Sección del diagrama de flujo.



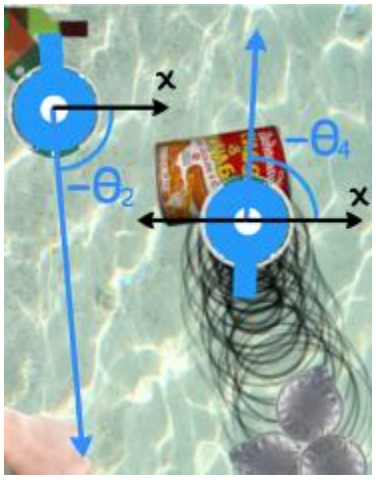
Para el desplazamiento de los robots es necesario saber la orientación del robot para definir en qué sentido es más rápido girar para dirigirse al objetivo deseado, como se observa en la Figura 34 hay un vector de color rojo que se crea desde el centro del robot al objetivo al que debe desplazarse el cual forma un ángulo θ_1 con el eje X, el cual se encuentra vertical debido a que el sistema de referencia está definido por la imagen del Kinect donde la mesa se encuentra vertical (Figura 25.b), por otro lado se tiene la orientación del e-puck θ_2, la cual fue calculada previamente (Figura 28), al restar estos ángulos se obtiene el ángulo que el robot debe girar para ubicarse de frente a su objetivo.

Figura 34. Ángulos necesarios para el desplazamiento.



Debido a que en la Figura 28 la imagen de profundidad del kinect se encuentra invertida cuando se calcula la orientación del e-puck por lo tanto continuando con el ejemplo de la Figura 34 las orientaciones de los robots seria la que se puede observar en la Figura 35.

Figura 35. Sistema de referencia para la orientación de los e-pucks.



Por esta razón se multiplica θ_1 por -1 para pasar el ángulo a este sistema de referencia y de esta forma utilizar la siguiente ecuación para calcular la diferencia entre los ángulos:

$$\theta_d = \theta_e - \theta_o \qquad (8)$$

Donde θ_d es la diferencia de los angulos deseada; θ_e es la orientación del e-puck y θ_o es el ángulo entre el objetivo y el eje x multiplicado por -1.Asi de esta manera, asignando valores a los ángulos del ejemplo en la Figura 34 donde $\theta_1 = 65°, \theta_2 = 85°, \theta_3 = 70°$ y $\theta_4 = -88°$ se obtendrían los siguientes resultados:

$$\theta_{d1} = -85 - (-1 * 65) = -20$$

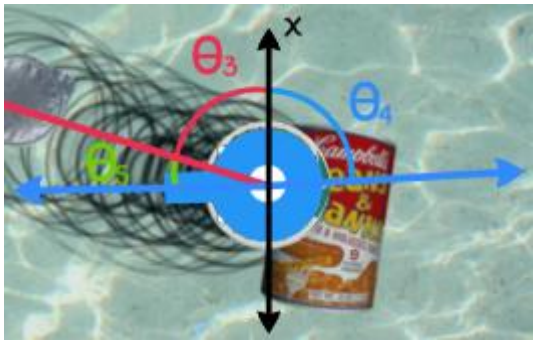
$$\theta_{d2} = 88 - (-1 * 70) = 158$$

Como se observa en los resultados para el segundo caso el ángulo a girar es muy grande y definitivamente no es el sentido más corto para girar, por esta razón cuando el valor absoluto de la diferencia entre los ángulos es mayor a noventa grados ($|\theta_d| > 90°$) es necesario hallar el ángulo complementario con el signo opuesto con el fin de que el giro se de en sentido contrario, para esto se utiliza la siguiente ecuación:

$$\theta_c = \theta_d \pm 180 \qquad (9)$$

Donde θ_c es el ángulo complementario; θ_d es la diferencia de los ángulos y el signo de 180 es negativo si θ_d es positivo o es positivo si θ_d es negativo de tal manera que el ángulo resultado presente el sentido contrario. En el segundo e-puck de la Figura 34 la diferencia de los ángulos es $158°$ por lo cual el ángulo complementario seria $\theta_c = \theta_5 = -22°$ el cual se puede observar en la Figura 36.

Figura 36. Angulo complementario.

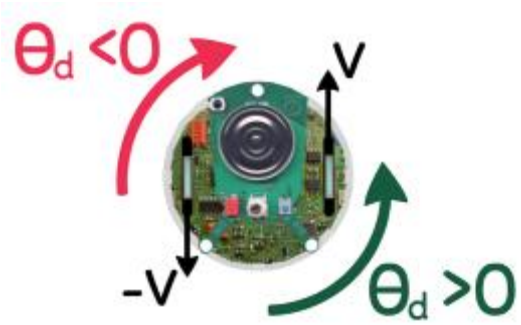


Una vez se tiene calculado el ángulo en el que el robot debe moverse se procede a utilizar un PID para calcular la velocidad del e-puck. La variable de entrada al PID es la diferencia de los ángulos θ_d o en su defecto su complemento θ_c y la salida es la velocidad a la que debe rotar cada llanta del robot la cual se calcula través de la Ecuación 10:

$$S = k_p P + k_i I + k_d D \quad (10)$$

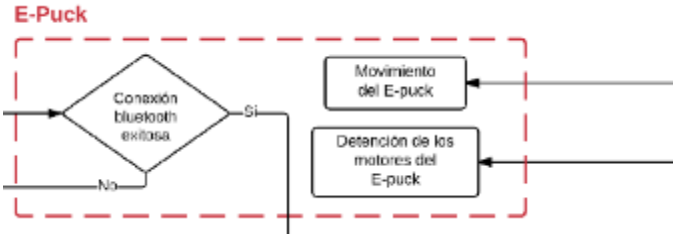
Donde S es la velocidad de cada llanta; P es el ángulo θ_d; I es la suma acumulada de todos los θ_d; D es la resta del error actual (θ_d) con el anterior y k_p, k_i, k_d son los coeficientes hallados experimentalmente para ajustar el resultado del PID. Debido a que la velocidad máxima de los motores del e-puck es mil pasos por segundo, lo cual equivale a una rotación completa de la llanta por segundo, es necesario limitar la salida del PID entre +1000 y -1000. Una vez se obtiene la salida del PID este valor se asigna al motor derecho y se multiplica por -1 para asignarlo al motor izquierdo y de esta manera al tener signos opuestos en la velocidad de las llantas el robot no se desplaza pero gira en su propio eje como se observa en la Figura 37. La salida del PID siempre presentara el mismo signo de la entrada (θ_d) por lo cual utilizando los ejemplos de la Figura 34 en cuyo caso ambos ángulos resultantes son negativos (θ_d y θ_c) las velocidades obtenidas del PID también serán negativas causando que los e-pucks giren a la derecha siguiendo la lógica de la Figura 37.

Figura 37. Sentido de giro del e-puck.



En cada ciclo se calcula el nuevo ángulo del robot y se repite el proceso anterior hasta que el error o diferencia entre los ángulos ($|\theta_d|$) sea menor o igual a $0.5°$ caso en el cual el robot se encuentra correctamente orientado por lo que se detiene la rotación del e–puck para empezar a desplazarse, el envío de datos a los motores de los robots y la señal de detención se pueden observar en la sección del diagrama de flujo de la Figura 38.

Figura 38. Sección del diagrama de flujos.



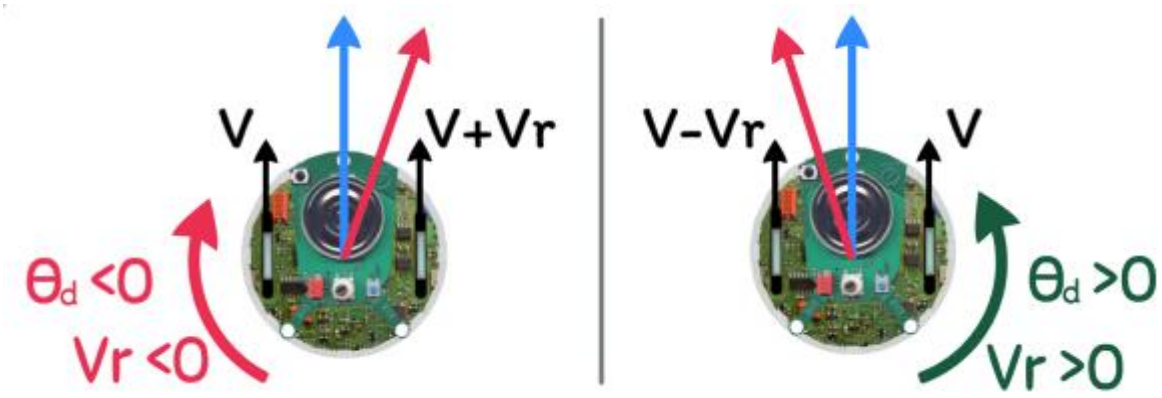
Para iniciar el desplazamiento primero se calcula la distancia entre las coordenadas del e-puck y del objetivo, debido a que este valor será el error en la posición que el PID debe corregir utilizando la misma Ecuación 10:

$$S = k_p P + k_i I + k_d D \quad (10)$$

Donde S es la velocidad de cada llanta; P es la distancia entre el e-puck y su objetivo; I es la suma acumulada de todas las distancias; D es la resta del error actual (distancia al objetivo) con el anterior y k_p, k_i, k_d son los coeficientes hallados experimentalmente para ajustar el resultado del PID. El resultado de este PID es la velocidad para desplazarse en línea recta hasta el objetivo, sin embargo debido a que la orientación del e-puck no es exacta esta debe ser corregida a medida que

el robot se mueve, por esta razón se calcula otro PID cuya entrada P es la velocidad hallada multiplicada por el error en la orientación ($V * \theta_d$) y la salida S es la velocidad que se debe restar a una llanta para hacer al e-puck girar, corrigiendo así su orientación. En la Figura 39 se observa como a la velocidad en línea recta (V) se le resta la velocidad de rotación (V_r) hallada con el segundo PID, debido a que la velocidad de rotación es proporcional a la velocidad y al error en la orientación, el resultado será siempre del mismo signo del error (θ_d) definiendo el sentido de giro (Figura 39) y a su vez a medida que la distancia disminuye la velocidad en línea recta disminuye causando que a sí mismo la velocidad de rotación que se debe restar sea menor.

Figura 39. Sentido de giro del e-puck mientras se desplaza.



En algunos casos el menor ángulo para rotar y orientar rápidamente al e-puck en línea con el objetivo resulta en que el e-puck se encuentre dirigido en sentido contrario al objetivo como en el ejemplo de la Figura 39, en este caso después de que el robot se encuentre correctamente orientado debe desplazarse hacia atrás. El marcado con saliente en la parte trasera del e-puck permite calcular la orientación pero no diferenciar el frente de la parte de atrás por medio de la imagen del Kinect por lo cual todos los robots inician su desplazamiento siempre hacia el frente, sin embargo después de cuatro ciclos del juego se compara la distancia actual al objetivo con la inicial para saber si el robot se aleja o se acerca al objetivo, de esta manera si la distancia ha aumentado el robot invierte su dirección de desplazamiento y se dirige hacia atrás.

Cuando se detecta que el robot se encuentra a dos o menos pixeles de llegar a las coordenadas de la basura asignada, se detiene el robot y se asigna como nuevo objetivo las coordenadas de su base o posición inicial, de esta manera empieza todo el proceso de desplazamiento desde el principio orientando correctamente el robot para moverse con la basura en caso de ser el tipo de desecho correcto. Por

último se detiene definitivamente al robot cuando se detecta que se encuentra a dos o menos pixeles de las coordenadas de su base o posición inicial desechando así la basura.

5.2.8 Programación con Pygame. Debido a que el posicionamiento de la basura en el mar o pantalla se realiza de manera aleatoria es necesario detectar colisiones entre cada imagen de basura y los demás objetos en pantalla para evitar que algunas imágenes se sobrepongan y el juego pierda coherencia. Por esta razón todas las basuras son un tipo de sprite de esta manera se evita que una basura se ubique en el mismo lugar de otra, adicionalmente cada robot tiene dos sprites, uno que se queda siempre ubicado en su base, para evitar que una basura se ubique en ese espacio cuando el robot no se encuentre, y otro sprite invisible (no presenta imagen en la pantalla) que se mueve todo el tiempo por la pantalla al mismo tiempo que el robot real se mueve, esto se realiza con el fin de detectar colisiones reales entre dos robots y así detener los e-pucks e indicar al usuario que ha perdido. En la Figura 40.A se puede observar el diseño de la pantalla cuando el usuario pierde debido al nivel de contaminación y en la Figura 40.B se observa la implementación real del juego cuando se presenta una colisión de los robots.

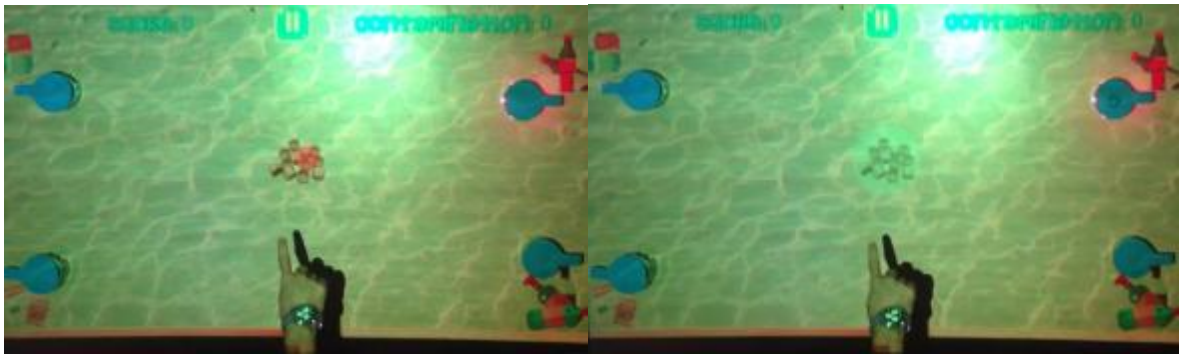
Figura 40. Fin del juego.

Los demás objetos en la pantalla como los botones y los puntajes también utilizan sprites para evitar que algún desecho sea sobrepuesto, por otro lado el nivel de contaminación en el agua se refleja en el juego por medio de dos indicadores, el puntaje en pantalla y una imagen de agua contaminada cuya transparencia disminuye a medida que aumenta la contaminación, en la Figura 41.A se puede observar la pantalla del juego cuando la contaminación es cero y en la Figura 41.B la contaminación ha aumentado a 24 dificultando la visualización de las basuras debido a la imagen del agua contaminada.

Figura 41. a) Pantalla descontaminada. b) Alto nivel de contaminación.

Para facilitar la selección de los robots y las basuras es necesario implementar un puntero que indique al usuario claramente donde está apuntando, por esta razón todo el tiempo se proyecta un círculo rojo en la coordenada a la que apunta el vector 3D del brazo como se observa en la Figura 42.A, en caso de que el puntero se encuentre sobre un objetivo ya sea robot o basura, el área del puntero empieza a aumentar para brindar una realimentación de que está realizando una selección la cual es completada cuando el puntero alcanza su área máxima y cambia a color verde como se observa en la Figura 42.B.

Figura 42. a) Puntero. b) Realimentación visual de una selección exitosa.



En la Figura 43 se puede observar la primera pantalla que se proyecta al iniciar el juego, en la parte inferior se despliega un mensaje que indica al usuario que debe apuntar para seleccionar alguna opción, el tamaño de letra de este mensaje varía en cada ciclo del juego dando la sensación de que el mensaje se mueve con el fin de llamar la atención del usuario y así servir como guía.

Figura 43. Menú de inicio.

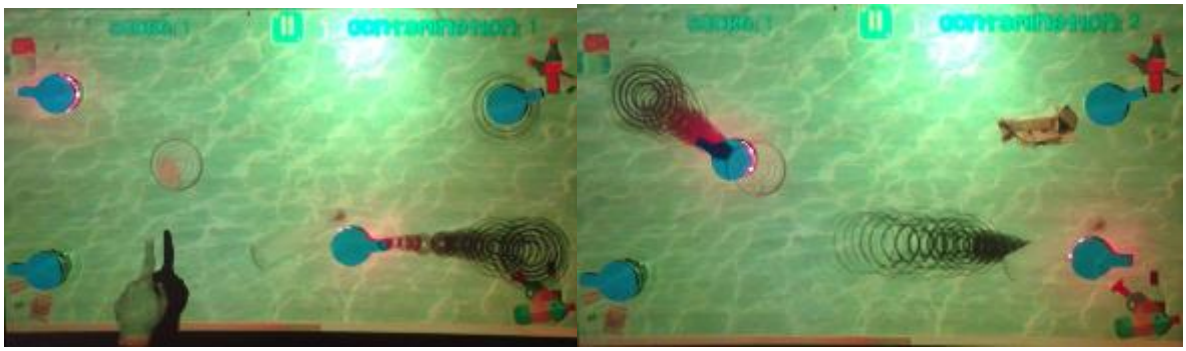
Para el desarrollo del juego se utilizaron diferentes bases de datos para los sonidos, botones y tipografías, por medio de FreeSound se descargaron sonidos que indicaran el inicio del juego, el acierto en un desecho, una colisión o el fin del juego. Por otro lado de la base de datos Game Art 2D se obtuvo los botones implementados y del generador de textos CoolText se obtuvo las tipografías utilizadas cuyo resultado se puede observar en la Figura 44, donde la imagen a es el menú que se despliega al pausar el juego y la imagen b es la pantalla que se proyecta al ganar el juego.

Figura 44. a) Menú de pausa. b) Pantalla ganadora.

Una vez un robot ha alcanzado su basura objetivo, si esta es del tipo de desecho correcto, el robot la desplaza a su base, este desplazamiento es coherente debido a que en cada ciclo del juego las coordenadas del sprite de la basura son re-calculadas basadas en el cambio de las coordenadas del e-puck halladas con el kinect, de esta misma manera funcionan las ondas de agua generadas en el desplazamiento del robot, dependiendo de la posición del robot se reproduce en

secuencia unas imágenes de círculos cuya área aumenta imitando así el movimiento del agua como se puede observar en la Figura 45.

Figura 45. Ejemplo de movimiento de la basura y las ondas de agua.



6. CONCLUSIONES

El uso de técnicas de procesamiento de imagen, como la ecualización del histograma, filtros y operaciones morfológicas binarias, permitió obtener una correcta detección de los robots e-pucks y así proceder a calcular sus coordenadas y orientación para permitir la interacción con objetos proyectados. Esta interacción entre los robots y el ambiente virtual se desarrolló satisfactoriamente al programar que los desechos virtuales se muevan acorde al desplazamiento de los e-pucks, los cuales a su vez dejan el rastro de sus movimientos por medio de ondas de agua proyectadas en el juego. El sistema de interacción desarrollado se validó exitosamente a través de la implementación de un juego que fomenta el adecuado reciclaje de los desechos y brinda al usuario una experiencia agradable por medio de una interfaz natural y sencilla, debido a que el control del juego se realiza mediante un movimiento intuitivo y básico, como lo es apuntar, sin necesidad de mandos o controles. De esta manera se logró alcanzar cada uno de los objetivos específicos planteados en el proyecto obteniendo como resultado el videojuego Save the sea.

La implementación del sensor Kinect como cámara de profundidad facilitó la detección de los robots debido a la diferencia de altura entre la mesa y estos, sin embargo debido a que esta cámara trabaja con un rayo infrarrojo la radiación infrarroja de la luz solar afecta la imagen obtenida haciendo que esta no sea muy exacta o estable debido al ruido provocado. El nivel de interferencia depende de que tan directa sea la luz solar recibida[10], para la detección de los e-pucks y del brazo del usuario se realizaron pruebas en habitaciones con bastante iluminación del sol pero no de manera directa y los resultados fueron exitosos, sin embargo en espacios al aire libre no es posible la implementación del juego debido a la iluminación directa del sol, este inconveniente se podría solucionar en un futuro cambiando el sensor por el nuevo Kinect V2 con infrarrojo activo. Por otro lado a pesar de que el sensor funciona en habitaciones con iluminación solar la implementación del juego en estas áreas dependerá de la potencia del proyector, debido a que la imagen del proyector que se tenía disponible en este caso no era muy visible en áreas tan iluminadas, el área de prueba final tuvo que ser en una habitación oscura. Al momento de utilizar el procesamiento de imagen para la detección de los robots es necesario tener en cuenta que esta detección debe realizarse rápidamente para dar la sensación de que es en tiempo real evitando que el robot pase de largo sus objetivos o pueda caerse de la mesa, inicialmente

[10] SIRIGNANO, Carmine. Range of Windows Kinect, and outdoor sun Infrared interference? [en línea]. Microsoft Developer Network, 17 de Septiembre 2013 [consultado 15 de enero de 2017]. Disponible en Internet: https://social.msdn.microsoft.com/Forums/en-US/97dbe40f-e962-42c6-8e01-b4c882958011/range-of-windows-kinect-and-outdoor-sun-infrared-interference?forum=kinectsdk

se utilizó un filtro local calculando la desviación estándar en una ventana de 5x5 en toda la mesa de trabajo, a pesar de que la aplicación de este filtro permitió detectar correctamente los robots, el tiempo de procesamiento de esta operación era muy alto, aproximadamente dos segundos, para realizar un adecuado control de velocidad y posición de los e-pucks, por esto fue necesario recurrir a métodos de procesamiento de imagen más rápidos como la ecualización del histograma y las operaciones morfológicas. Durante el procesamiento de imagen para la detección de la mesa, los robots y el brazo, fue necesario utilizar diferentes parámetros para realizar la toma de decisiones, tales como valores de profundidad, cantidad de pixeles, el tamaño de áreas detectadas, entre otros, estos parámetros se utilizaron para extraer la mesa de trabajo, para crear rangos de altura donde se encuentran los robots o el brazo del usuario, para detectar que áreas son seleccionadas como robots, para determinar cuándo un robot ha llegado a su objetivo y en otros procesos realizados a lo largo del código. El uso de estos parámetros hallados experimentalmente hacen que la aplicación sea muy dependiente de una variedad de factores por lo cual no es robusta, al realizar diferentes pruebas en estos procesos, era necesario re-ajustar constantemente las alturas de los rangos y las áreas a detectar como robots para que el juego funcionara correctamente, por esto es importante que estos parámetros hallados experimentalmente no sean valores exactos sino factores proporcionales a una variable de tal manera que a pesar de que las condiciones cambien el juego siga funcionando, de esta manera los rangos de altura se crearon a partir de la profundidad a la que se encuentra el suelo mientras que el área mínima para detectar los robots se definió como el 60% del área máxima hallada ($AreaMax * 0{,}6$), como cambios a futuro la distancia utilizada para detectar cuando un robot ha llegado a su objetivo se podría cambiar de una cantidad de pixeles exacta, actualmente dos, por un factor proporcional al radio en pixeles del área detectada como un e-puck y así de esta manera un cambio en la altura del kinect no cambiaría sustancialmente esta distancia.

Debido a que el comportamiento de los e-pucks se ve dirigido por el usuario y la cámara del kinect, estos no presentan autonomía en decisiones causando que deban estar siempre conectados por bluetooth al computador. Al iniciar el desarrollo del juego se planteó la idea de utilizar ocho e-pucks de tal manera que estos trabajaran en parejas para mover una basura, sin embargo al intentar conectar varios e-pucks a un mismo computador se evidencio que a medida que aumentaban los robots se dificultaba más establecer la conexión bluetooth y a pesar de que se programó el código para realizar una reconexión en caso de errores en la conexión, debido al tiempo de espera, no se logró conectar más de 7 e-pucks. El objetivo de utilizar dos e-pucks por basura era generar la sensación de cooperación, sin embargo esta habría sido solo simulada ya que un robot se encargaría solamente de imitar el movimiento del otro manteniendo siempre una distancia entre sí, finalmente se decidió utilizar solo cuatro robots ya que la

cooperación simulada no era un objetivo del proyecto y así lograr que la conexión se estableciera de manera más rápida.

Inicialmente se había planteado el uso de un sensor shimmer como interfaz de usuario para controlar los robots, sin embargo la implementación se realizó utilizando un sensor Kinect ya que se concluyó que este brinda una mayor comodidad al usuario y una interacción más natural al no necesitar ningún tipo de dispositivo en el cuerpo, adicionalmente elimina la programación de un dispositivo adicional debido a que el Kinect era igualmente necesario en el juego para la detección de los robot por medio de la cámara de profundidad. Microsoft presenta diferentes ejemplos y librerías que ayudan a la detección de gestos del usuario utilizando el kinect, sin embargo estos ejemplos utilizan la detección completa del esqueleto del usuario para poder identificar los movimientos de las extremidades, debido a que en esta aplicación la única parte visible del cuerpo son los brazos no es posible utilizar la detección de gestos de las librerías ya existentes. En un principio se había propuesto utilizar gestos de agarre para detectar cuando un usuario quiere seleccionar un robot o recoger una basura sin embargo este procedimiento no fue el adecuado ya que se utilizaron diferentes métodos reorientando los brazos, hallando la sección más ancha, calculando el área y segmentando las imágenes pero debido a las diferentes manos y posibles variaciones en los gestos no fue posible diferenciar correctamente las manos abiertas de las cerradas, una solución para esto podría ser descargar o crear una base de datos con una gran cantidad de imágenes de diferentes brazos con las manos cerradas y abiertas para así entrenar una red neuronal que distinga una mano de la otra pero por razones de tiempo este procedimiento no se llevó a cabo.

El uso de sprites en la programación del juego facilitó la detección de colisiones ya que cada robot y basura, al estar representado por un sprite, presenta un cuadro que lo delimita y los estos ya presentan una función encargada de comparar las coordenadas de los cuadros delimitadores para detectar alguna colisión, por lo cual lo único necesario del procesamiento de imagen para crear los sprites y detectar el choque entre robots es la coordenada donde se encuentran ubicados. Los sprites presentan otras bondades como el facilitar el control de los objetos haciéndolo más ordenado por medio de la creación de grupos como por ejemplo botones, basuras, robots o bases, esto permite definir el orden en el que se copian las imágenes a la pantalla logrando que las ondas de agua proyectadas siempre aparezcan debajo de las basuras o que el puntero siempre aparezca por encima de los botones y los desechos. Es importante resaltar que los grupos de sprites no se pueden indexar como un vector, el cual se encuentra ordenado y la posición de cada elemento no cambia, ya que la iteración de los sprites se realiza de manera aleatoria lo cual puede causar errores si no se tiene en cuenta. En el desarrollo de la aplicación se presentó un error en la selección de objetivos debido a que el vector ordenado con las coordenadas de los objetivos disponibles se editaba en

cada ciclo del juego por medio de la iteración del grupo de sprites de los desechos, esto generaba que cada ciclo del juego la posición de una basura en la pantalla cambiara de índice en el vector de objetivos, y es este índice el que indica la selección del usuario, de esta manera cuando una persona se encontraba seleccionando un desecho de repente el puntero cambia de objetivo y seleccionaba una basura distinta a la deseada, este problema se solucionó evitando la actualización de todo el vector en cada ciclo del juego y la iteración de un grupo de sprites y utilizando en su lugar un vector auxiliar para concatenar cada nuevo objeto que se debía posicionar en la pantalla dejando intactas las posiciones de los objetos previamente proyectados en el juego.

7. MEJORAS A FUTURO

En el desarrollo del juego se logró exitosamente permitir la interacción de los e-pucks con basuras virtuales y a su vez detectar la interacción entre los robots en caso de una colisión, debido a que el choque entre los e-pucks causa que se pierda la partida, es necesario que el usuario tenga esto en cuenta al momento de dirigir los robots para que sus trayectorias no se crucen, sin embargo hace falta ofrecer al usuario un mecanismo para evitar la colisión una vez los robots ya se encuentren en movimiento, para esto se propone como mejora a futuro la implementación de comandos de voz como otro método de interacción entre los usuarios y los e-pucks, de esta manera utilizando procesamiento de señales se podría detectar cuando el usuario diga “Detener robot uno” para evitar la colisión y permitir el paso del otro e-puck y posteriormente reanudar la trayectoria del robot detenido por medio del comando “Reanudar robot uno”. Por otro lado la implementación del algoritmo A* evitaría el uso de comandos y retiraría esta tarea del usuario calculando la ruta a su objetivo teniendo en cuenta la posición de los robots y las basuras en pantalla. En este momento el desarrollo del juego permite que los robots pasen por encima de las basuras proyectadas mientras se dirige a su objetivo, sin embargo la implementación de este algoritmo permitiría mejorar la lógica visual del juego causando que los robots esquiven los desechos, este algoritmo no se utilizó en el video juego debido a que su implementación debe ser muy eficiente de tal manera que el procesamiento no tome mucho tiempo, lo cual causaría errores en el control de velocidad y posición de los robots, pero que a su vez recalcule constantemente la ruta a su objetivo teniendo en cuenta los otros robots que se encuentran en movimiento, por esta razón la implementación del algoritmo en el desarrollo del juego requeriría de mucho tiempo el cual estaba restringido a cuatro meses ya que era el resultado final del proceso de intercambio con la universidad Haute Ecole d’Ingenerie et du Gestion du Canton Vaud en Suiza.

Otro aspecto a mejorar en el juego es brindar la opción de un menú de configuración en el cual se pueda seleccionar el nivel de dificultad, cambiando la velocidad de contaminación del agua y adicionalmente brindar la opción de seleccionar el modo de juego, de tal manera que el usuario seleccione si desea jugar a contrarreloj, perdiendo si se cumple el tiempo antes de alcanzar cierto puntaje de limpieza, o si desea jugar hasta la saturación de basura en la pantalla.

BIBLIOGRAFÍA

About [en línea]. THALMIC LABS, 2013 [consultado 28 de Septiembre de 2015]. Disponible en Internet: https://www.thalmic.com/about

ALBORNOZ FIGUEROA, Roberto. Conceptos Básicos para el Desarrollo de Videojuegos 2D [en línea]. Losers Juego, 10 de Enero de 2007 [consultado 08 de Enero de 2017]. Disponible en Internet: http://www.losersjuegos.com.ar/referencia/articulos/conceptos_basicos

BONANI, Michael et al. The e-puck, a Robot Designed for Education in Engineering [en línea]. Infoscience EPFL, 2009, p. 1-5 [consultado 7 de Febrero de 2016]. Disponible en Internet: https://infoscience.epfl.ch/record/135236/files/epuck-robotica2009.pdf?version=2

CHÁCON MURGUÍA, Mario Ignacio et al. Percepción visual aplicada a la robótica [en línea]. Mexico D.F.: Alfaomega, 2015. 3 p. [consultado 8 de Enero de 2017]. Disponible en Internet: https://books.google.com.co/books?id=pCHaDAAAQBAJ&pg=SA1-PA5&lpg=SA1-PA5&dq=conectividad+pixeles&source=bl&ots=n8ojtRULzm&sig=cqiqUyzwU3_njTHx9s3NUi0vHlc&hl=es&sa=X&ved=0ahUKEwit1ez4otTRAhXMMSYKHb6vAQA4ChDoAQg7MAg#v=onepage&q=conectividad%20pixeles&f=false

Depth data visualization with PyKinect and PyGame [en línea]. Atlassian Bitbucket, 21 de mayo de 2015 [consultado 20 de abril de 2016]. Disponible en Internet: https://bitbucket.org/snippets/VitoGentile/Eoze/depth-data-visualization-with-pykinect-and

Ecualizado [en línea]. Opto-Electronic Image Processing Group, Universitat de Valéncia [consultado 28 de Octubre de 2016]. Disponible en Internet: http://www.uv.es/gpoei/eng/Pfc_web/realzado/ecualiza/ecuali.htm

E-Puck [en línea]. GCtronic: 2 Software [consultado 25 de febrero de 2016]. Disponible en Internet: http://www.gctronic.com/doc/index.php/E-Puck#Software

FALLAVOLLITA, Pascal et al. Augmented Reality Magic Mirror using the Kinect [en línea]. Technische Universität München, 2015 [consultado 29 de Octubre 2016]. Disponible en internet: http://campar.in.tum.de/Chair/ProjectKinectMagicMirror

Freebies [en línea]. Game Art 2D, 2016 [consultado 02 de julio de 2016]. Disponible en Internet: http://www.gameart2d.com/freebies.html

Future Robotic Interfaces [en línea]. Jet Propulsion Laboratory, California Institute of Technology [consultado 29 de Octubre 2016]. Disponible en Internet: http://www.hi.jpl.nasa.gov/projects/#controlling-space-missions

HERAS LARA, Lizbeth. La realidad aumentada: una tecnología en espera de usuarios [en línea]. En: Revista Digital Universitaria. 10 de agosto 2004, vol. 5, no. 7, p. 4-5 [consultado 28 de Octubre de 2016]. Disponible en Internet: http://www.revista.unam.mx/vol.8/num6/art48/jun_art48.pdf

How it works [en línea]. GestSure [consultado 28 de Octubre 2016]. Disponible en internet: http://www.gestsure.com/

Kinect for Windows SDK v1.8 [en línea]. Microdoft Download Center, 13 de Septiembre de 2013 [consultado 15 de abril de 2016]. Disponible en Internet: https://www.microsoft.com/en-us/download/details.aspx?id=40278

Kinect hardware [en línea]. Microsoft, 2017 [consultado 20 de Enero de 2017]. Disponible en Internet: https://developer.microsoft.com/en-us/windows/kinect/hardware

Kinect powers informed shoppers [en línea]. Microsoft, 8 de Julio de 2015 [consultado 29 de Octubre 2016]. Disponible en internet: https://blogs.msdn.microsoft.com/kinectforwindows/2015/07/08/kinect-powers-informed-shoppers/

Linear Regression [en línea]. StatsModels Statistics in Python, 2013 [consultado 9 Junio de 2016]. Disponible en Internet: http://statsmodels.sourceforge.net/devel/regression.html

Logos [en línea]. CoolText: Graphics generator [consultado 16 de julio de 2016]. Disponible en Internet: https://cooltext.com/

MCKIERNAN, Sean J. Find a Way [en línea]. Pygame, 24 de Junio de 2013 [consultado 19 de Julio de 2016]. Disponible en Internet: http://pygame.org/project-Find+a+Way-2094-.html

Module: measure [en línea]. Scikit-Image: image processing in python [consultado 09 de junio de 2016]. Disponible en Internet: http://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.compare_ssim

Morphological Transformations [en línea]. OpenCV [consultado 8 de Enero de 2017]. Disponible en Internet: http://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html

Multi-dimensional image processing [en línea]. SciPy, 19 de Septiembre de 2016 [consultado 13 de Mayo de 2016]. Disponible en Internet: https://docs.scipy.org/doc/scipy/reference/ndimage.html

Myo - An Armband That Gives You Superpowers [video]. THALMIC LABS, 2015 [consultado 28 de Septiembre 2015]. Disponible en internet: https://www.youtube.com/watch?t=49&v=UL_pDatlLOg

Nod enables natural drone control [en línea]. NOD, 2015 [consultado 28 de Septiembre de 2015]. Disponible en Internet: https://nod.com/

Operaciones Morfológicas en Imágenes Binarias [en línea]. Universidad Nacional de Quilmes, Agosto de 2005, p. 1-6 [consultado 8 de Enero de 2017]. Disponible en Internet: http://iaci.unq.edu.ar/materias/vision/archivos/apuntes/Operaciones%20Morfol%C3%B3gicas%20en%20Im%C3%A1genes%20Binarias%20-%20parte%201.pdf

PID Theory Explained [en línea]. National Instruments, 29 de Marzo de 2011 [consultado 08 de enero de 2017]. Disponible en Internet: http://www.ni.com/white-paper/3782/en/

Pykinect 2.1 [en línea]. Python, 8 de Noviembre de 2014 [consultado 20 de abril de 2016]. Disponible en Internet: https://pypi.python.org/pypi/pykinect

RUIZ DEL SOLAR, Javier et al. Filtrado de Imágenes y ecualización de histograma [en línea]. Universidad de Chile, 24 de Septiembre 2015, p. 3-5 [consultado 28 de Octubre de 2016]. Disponible en Internet: https://www.u-cursos.cl/usuario/b4eb6d37062854338662ba7470704112/mi_blog/r/EL7008___Tarea_1.pdf

SIRIGNANO, Carmine. Range of Windows Kinect, and outdoor sun Infrared interference? [en línea]. Microsoft Developer Network, 17 de Septiembre 2013 [consultado 15 de enero de 2017]. Disponible en Internet: https://social.msdn.microsoft.com/Forums/en-US/97dbe40f-e962-42c6-8e01-b4c882958011/range-of-windows-kinect-and-outdoor-sun-infrared-interference?forum=kinectsdk

SOLEM, Jan Erick. Programming Computer Vision with Python [en línea]. O'Reilly, 2012 [consultado 21 de Mayo de 2016]. Disponible en Internet: https://www.safaribooksonline.com/library/view/programming-computer-vision/9781449341916/ch01.html

Sounds [en línea]. Freesound [consultado 16 de julio de 2016]. Disponible en Internet: https://www.freesound.org/

Tech Specs [en línea]. Obtenido de THALMIC LABS, 2013 [consultado 28 de Septiembre 2015]. Disponible en internet: https://www.myo.com/techspecs

Técnicas de filtrado [en línea]. Universidad de Murcia, p. 67-69 [consultado 8 de Enero de 2017]. Disponible en Internet: http://www.um.es/geograf/sigmur/teledet/tema06.pdf

Wireless Sensor Platform [en línea]. SHIMMER, 2013 [consultado 28 de Septiembre de 2015]. Disponible en internet: http://www.shimmersensing.com/images/uploads/docs/Shimmer3_Spec_1.4.pdf