UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Trabajo de Fin de Grado

Detección de cubos de colores para valoración automática de destreza manual

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Autor: Luis Ángel García Astudillo

Tutor: Edwin Daniel Oña Simbaña

Título: *DETECCIÓN DE CUBOS DE COLORES PARA VALORACIÓN AUTOMÁTICA DE DESTREZA MANUAL*

Autor: *LUIS ÁNGEL GARCÍA ASTUDILLO*

Tutor: *EDWIN DANIEL OÑA SIMBAÑA*

La defensa del presente Trabajo de Fin de Grado se realizó el día 6 de Octubre de 2017; siendo calificada por el siguiente tribunal:

Presidente: *Horacio Lamela Rivera*

Secretario *Juan Carlos Burgos Díaz*

Vocal *Luis Santiago Garrido Bullón*

Habiendo obtenido la siguiente calificación:

Calificación:

Presidente **Secretario** **Vocal**

Agradecimientos

Este trabajo representa la culminación del que es, casi con seguridad, el mayor logro de mi vida. Me gustaría, por tanto, empezar agradeciendo y dedicando este trabajo a mis padres. Ellos me han educado y cuidado con abnegación, inculcándome valores y conocimientos sin los cuales no sería la persona que soy y, posiblemente, no estaría escribiendo estas líneas. Gracias.

A mi familia, a mi hermana, Patricia, a Nerea. Gracias por cuidar de mí, quererme y apoyarme incluso a pesar de mis nervios y defectos.

A todos mis amigos; los de Parla, los de Mozoncillo, los de la universidad. Gracias por compartir vuestro tiempo conmigo. Sólo lamento no tener tantas oportunidades de veros como antes.

Gracias a la universidad, los profesores y compañeros por todo el esfuerzo que han realizado. Gracias por satisfacer mi curiosidad y alentarme a buscar nuevas metas.

Me gustaría dar las gracias también a mi tutor, Edwin, por su ayuda, consejos y directrices en los momentos complicados. La confianza que depositó en mí ha llevado a lograr este resultado.

Por último, agradecer también a la gente que trabaja en el Parque Científico, en especial a Jorge y Bartek, su ayuda, atención y dedicación.

En definitiva, gracias a todo el mundo que, directa o indirectamente, ha contribuido a este trabajo. Gracias por darme la oportunidad de aprender de vosotros.

En Mozoncillo a 9 de septiembre de 2017

Luis

Poca gente es capaz de prever hacia dónde les lleva el camino hasta que llegan a su fin.

J.R.R. Tolkien

Con lluvia, este camino sería otro camino, este bosque, otro bosque.

Patrick Rothfuss

Espero que encuentres tu Torre, Roland, que entres y subas hasta lo más alto.

Stephen King

Índice general

Lista de Figuras

Lista de Tablas

Resumen

La adquisición de datos del exterior por medio de cámaras y su posterior procesado para obtener información valiosa para una Inteligencia Artificial (IA) es, en la actualidad, una de las líneas de investigación con mayor futuro y mayor número de aplicaciones. La visión artificial es un complemento importante a la hora de crear IAs capaces cada vez de tomar decisiones más complejas.

En el ámbito de la robótica asistencial, la visión artificial puede ser de gran ayuda, tanto para médicos como para pacientes, facilitando pruebas, operaciones, diagnósticos y rehabilitaciones.

El objetivo de este proyecto es implementar y evaluar el desempeño de varios algoritmos de reconocimiento con el objetivo de automatizar una prueba de destreza manual para pacientes afectados de parálisis lateral consistente en mover bloques con las manos. Para ello, nos valdremos de una cámara Kinect que permite adquirir imagen en color y de profundidad.

Palabras clave: robótica, asistencial, rehabilitación, visión artificial, Kinect, OpenCV.

Abstract

Data acquisition from the outside through video cameras and its subsequent processing to obtain valuable information that can be used by an Artificial Intelligence (AI) is a research area with a bright future in robotics. Computer vision is a very important tool in order to create robots capable of making complex decisions.

Computer vision can be a useful asset in robotic healthcare, for both doctors and patients, making tests, surgery, diagnosis and rehabilitations easier.

The main goal of this project is to implement and measure the efficiency of some recognition algorithms meant to automatize a manual dexterity test used by occupational therapist and doctors to evaluate physically handicapped individuals involving moving blocks with both hands. To do this a Kinect camera will be used, which will provide us with colour and depth images.

Keywords: robotics, healthcare, rehabilitation, computer vision, Kinect, OpenCV.

Capítulo 1

Introducción

Debido a los enormes avances médicos y sociales de las últimas décadas, la esperanza de vida ha aumentado progresiva y considerablemente en los países desarrollados. En España, uno de los países con mayor esperanza de vida de Europa, la media de edad ha aumentado en 14 años desde 1960 (Figura 1.1) [1].

Figura 1.1: Esperanza de vida en España en las últimas décadas [1].

Por desgracia, la edad trae consigo la aparición de enfermedades y dolencias asociadas al deterioro de los tejidos musculares, del cerebro o de otros órganos vitales. Dado que muchos de estos trastornos no tienen aún una solución médica efectiva, se hace patente la importancia de encontrar métodos de prevención y rehabilitación que aminoren la progresión de las enfermeda-

des, facilitando la vida a los enfermos. En España, las enfermedades cardiovasculares ya suponen la causa de muerte más habitual. Sólo en 2015, las muertes por infartos, enfermedades cerebrovasculares o insuficiencias cardíacas, entre otros tipos, alcanzaron el 29 % de las muertes totales en España [2]. Sin embargo, gracias a los avances médicos, la probabilidad de sobrevivir a uno de estos ataques es bastante alta, en torno al 70 % en el caso de los ataques cerebrovasculares, por poner un ejemplo. Esto supone que el número de enfermos que necesitan rehabilitación y ayuda es, en la actualidad, enorme, lo que conlleva un gran desembolso económico.

Las sesiones de rehabilitación y seguimiento de un trastorno médico requieren de personal cualificado, pero los últimos avances en electrónica y robótica permiten complementar la ayuda profesional, pudiendo realizar tareas sencillas y repetitivas, que de otra forma ocuparían mucho tiempo al médico, tiempo que se puede emplear en otro paciente, agilizándose así el proceso. La incorporación de la robótica a la medicina, se perfila, por tanto, como una opción de futuro para mantener e incluso incrementar la calidad del servicio sanitario en una sociedad avejentada.

1.1. Sobre las terapias de rehabilitación de pacientes

La Terapia Física y Rehabilitación es una especialidad médica centrada en la prevención, diagnóstico, rehabilitación y terapia de pacientes que sufren limitaciones funcionales producto de heridas, enfermedades o malformaciones. A pesar de que la especialidad comenzó como tal a principios del siglo XX, las bases de este campo surgieron en la Antigüedad y su historia se traza en diferentes culturas y geografías.

Los primeros usos que se conocen del ejercicio para aliviar dolores proceden de China, de la mano de las Artes Marciales tradicionales, en torno al 2000 a.C. [3]. La práctica de artes marciales incide en dos facetas: cuerpo y mente. Por un lado, prepara mentalmente a la persona que la practica para aguantar dolor. Y por otro, fortalece de forma progresiva todos los músculos del cuerpo, favoreciendo y acelerando la recuperación.

En el Antiguo Egipto contaban también con las herramientas para proporcionar rehabilitación efectiva a heridos y enfermos. Conocían los beneficios de los masajes y los utilizaban no sólo con fines estéticos, usaban aceites perfumados, sino también para aliviar dolores [4]. La Figura 1.2,

una reconstrucción de un grabado de la tumba de un sacerdote, muestra varios sirvientes dando masajes a sus señores. Además, de esta época datan algunas de las primeras prótesis funcionales encontradas. En la Figura 1.3 puede verse una prótesis para sustituir un dedo del pie amputado, que ayudaría a su dueña a conservar el equilibrio [5].

Figura 1.2: Grabado de la tumba de Ankh-mahor [4].

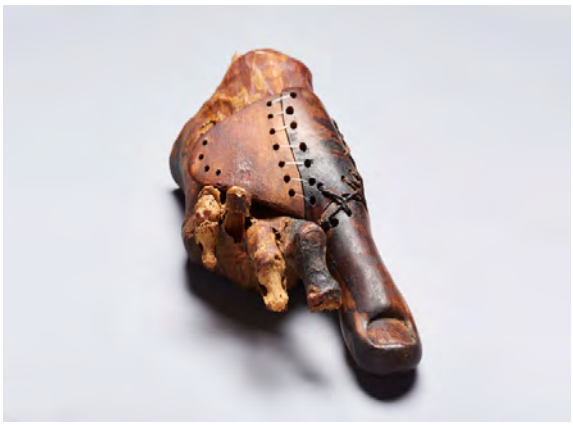


Figura 1.3: Prótesis de dedo del pie hallada en la tumba TT95 en Tebas [5].

La terapia de rehabilitación da un paso muy importante con Heródico de Selimbria, en torno al 500 a.C. Heródico propuso un método para prevenir y tratar enfermedades basado en un plan estricto de dieta y deporte y recopiló una gran cantidad de ejercicios beneficiosos. A pesar de que su método logró buenos resultados en ciertos pacientes, el propio Platón habla también de fracasos estrepitosos debidos a su alta exigencia [6].

Durante los siglos siguientes, el impulso en medicina vino por parte del mundo árabe. Alrededor del 1190 el judío cordobés Maimonides publica su «Guía para la buena salud», en la que expone sus ideas sobre prevención y tratamiento de enfermedades basándose en la dieta y el ejercicio saludable que propone el Talmud.

Ya en el Renacimiento, el médico italiano Jerónimo Mercurialis retoma las ideas de Heródico y Galeno y publica en 1569 «De arte gymnastica», un estudio acerca de cómo la dieta, el ejercicio y una higiene adecuada pueden llevar a curar y prevenir enfermedades y dolencias.

En 1674, el Rey Luis XIV de Francia ordena la construcción en París de la primera institución estatal destinada al cuidado y rehabilitación de personas, el «Hôtel Royal des Invalides» (Figura

1.4), donde se acogía y trataba a soldados heridos [7].

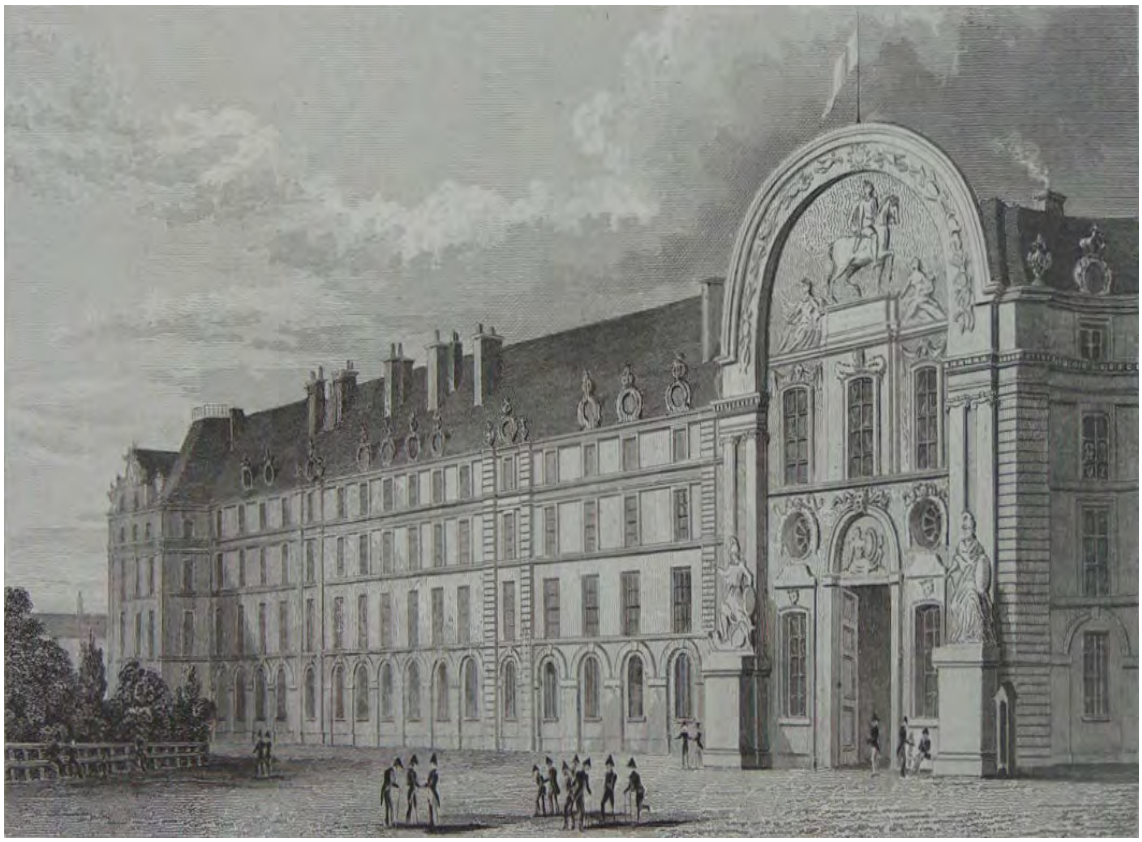


Figura 1.4: Grabado de la fachada del Palacio de los Inválidos. (W. Watkins, 1831) [8].

Los primeros tratados sobre «gimnasia médica» aparecen en el siglo XVIII y entre ellos cabe destacar «Gimnasia Medicinal y Quirúrgica», obra de Joseph Clément Tissot, un completo compendio sobre la manera de tratar e inmovilizar fracturas con ejercicios de rehabilitación específicos para cada caso. Gracias a las pruebas que ofrecía este tratado de las ventajas de la rehabilitación que involucra la movilidad del paciente, se dejó de considerar el reposo en cama como la mejor terapia de recuperación para heridos.

La investigación en torno al sistema nervioso se incrementó enormemente en el siglo XIX con la mejora de las herramientas y técnicas médicas. Investigadores de toda Europa se lanzaron a desentrañar el funcionamiento del cuerpo humano, en especial los procesos detrás de la propriocepción, la manera en que los animales percibimos en qué posición se encuentra cada parte de nuestro cuerpo sin utilizar los sentidos. Estas investigaciones condujeron a que el neurólogo Fulgence Raymond propusiera el concepto de «reeducación neuromuscular», base para gran

cantidad de las técnica de rehabilitación que se usan en la actualidad.

La primera mitad del siglo XX trajo consigo gran cantidad de novedades en este ámbito. Las Guerras Mundiales dejaron tras de sí gran cantidad de heridos y lisiados e impulsaron la investigación en todos los campos de la ciencia y la ingeniería. Como consecuencia se estudiaron nuevas técnicas de rehabilitación que incorporaban nuevas tecnologías como los rayos UV o la hidroterapia, se desarrollaron prótesis más eficaces y se mejoraron notablemente las prestaciones y ayudas a la rehabilitación por parte de los gobiernos mediante la creación de programas dedicados exclusivamente a estos menesteres [9].

Las causas por las que los pacientes necesitan rehabilitación han cambiado progresivamente desde mitad del siglo pasado, así como también las técnicas han evolucionado. La filosofía de usar movimientos predefinidos para cada dolencia ha experimentado un gran impulso, sobre todo desde que se sabe que la actuación inmediata tras la operación quirúrgica no tiene contraindicaciones. Esto es muy importante en una de las terapias de rehabilitación más comunes actualmente: la rehabilitación después de un problema cardiovascular. No sólo se logra la recuperación del paciente, sino que se mejora su condición y se previenen, por tanto, futuras complicaciones. Por otro lado, también se busca simplificar las terapias y dotarlas de ubicuidad; que los pacientes puedan realizar los ejercicios en casa, en cualquier momento, mejora su condición física y su bienestar psíquico.

1.2. La rehabilitación manual en la sociedad actual

Las enfermedades degenerativas como la enfermedad de Parkinson y los accidentes cerebrovasculares, también llamados ictus (trombos, embolias, aneurismas o malformaciones venosas), e infartos de cualquier tipo pueden producir en el paciente pérdida de movilidad, total o parcial [10]. La rehabilitación de esta condición se enfoca en tres áreas principalmente:

- **Física:** destinada a recuperar la fuerza y el equilibrio del paciente, ayudándole a caminar y mantenerse de pie.

- **Terapia Ocupacional:** consiste en practicar la psicomotricidad gruesa y fina para que el paciente pueda realizar tareas cotidianas como bañarse, peinarse, comer o vestirse por sí mismo.
- **Terapia del Habla y el Lenguaje:** se centra en la mejora de la vocalización, la ingesta de alimentos y los déficits cognitivos.

En este proyecto se va a tratar la terapia ocupacional para la rehabilitación de la destreza manual del paciente. La destreza manual se divide en gruesa y fina y existen ejercicios específicos para cada una de ellas. El objetivo de estos ejercicios es aprovechar la plasticidad del cerebro, su capacidad para rehacer conexiones neuronales después de un fallo, para reaprender los movimientos y recuperar la movilidad en la medida de lo posible [11].

En un afán por hacer más fácil la rehabilitación de los pacientes, la inclusión de la tecnología en estos ejercicios ha logrado acelerar la recuperación y permitir que la terapia sea continua, sin que el paciente necesite desplazarse al centro médico para realizar los ejercicios. Las Figuras 1.5, 1.6, 1.7 y 1.8 muestran algunos ejemplos.

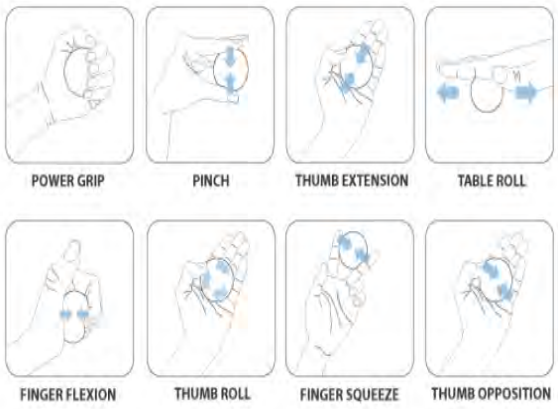


Figura 1.5: Ejercicios usando pelotas de goma de distinta dureza [12].

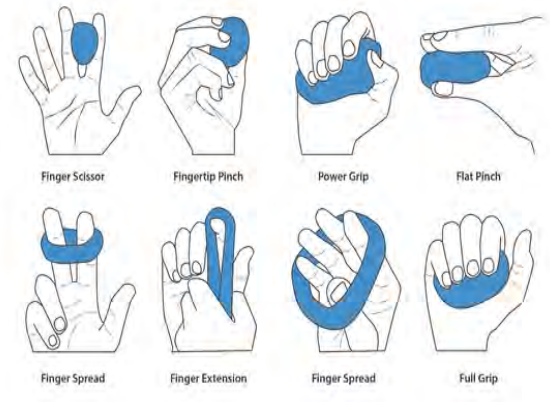


Figura 1.6: Ejercicios usando plastilina terapéutica [12].

Las maneras de evaluar el desempeño y el grado de avance del proceso también se han desarrollado enormemente y hay gran cantidad de test médicos para evaluar la psicomotricidad gruesa (Figura 1.9), la fina (Figura 1.10), aspectos específicos de una de ellas (Figura 1.11) o ambas a la vez (Figura 1.12).

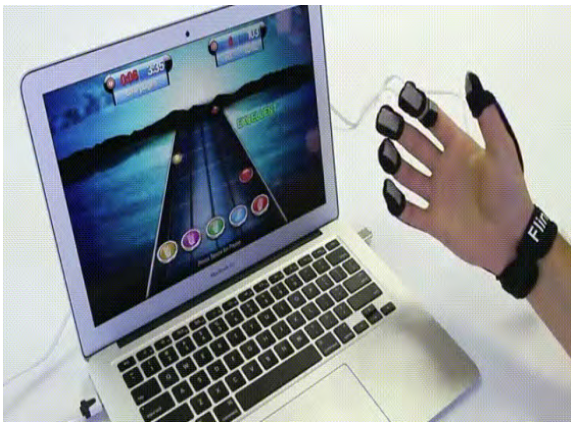


Figura 1.7: Juego musical controlado con un guante para rehabilitar los dedos [13].

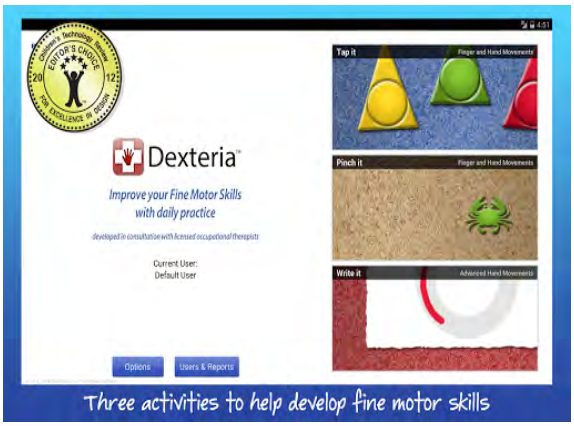


Figura 1.8: Aplicación para smartphone [14].

Figura 1.9: Test de Minnesota. El paciente debe colocar los cilindros en los huecos del tablero en el menor tiempo posible [15].

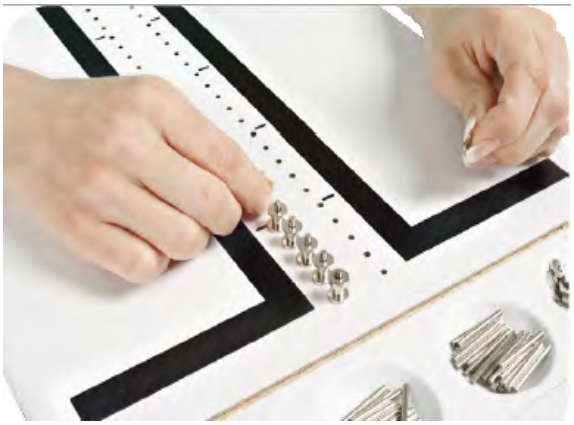


Figura 1.10: Test de Perdue Pegboard. El paciente debe colocar las varillas en los agujeros de la caja [16].

1.3. Objetivos

El objetivo de este proyecto es realizar la automatización del ensayo BBT (*Box and Blocks Test*). Esta prueba está dirigida a pacientes que sufren pérdida de movilidad en una o las dos extremidades superiores debido a accidentes cerebrovasculares, esclerosis múltiple, traumatismos

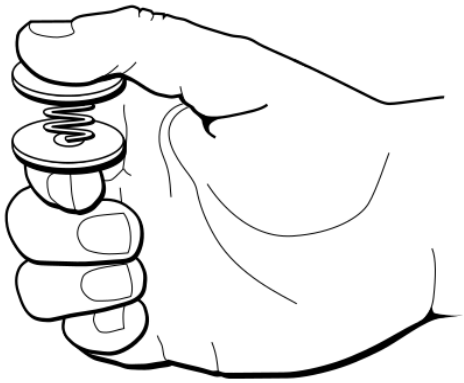


Figura 1.11: Test de Valero-Cuevas. Se evalúa la destreza y la fuerza de pellizco con la que el paciente el paciente comprime muelles de longitudes y durezas diferentes [17].

Figura 1.12: Test de Jebsen. El paciente debe superar varias pruebas cotidianas: escribir, mover latas, usar una cuchara... [18] [19].

craneoencefálicos, uso de prótesis y otros problemas que afectan a la destreza manual. El ensayo permite evaluar la psicomotricidad unilateral gruesa del paciente de una manera rápida y barata. [20]

El material del ensayo, que puede verse en la Figura 1.13, consiste en una caja rectangular de medidas 53.7 cm x 25.4 cm x 8.5 cm con una pantalla que la divide en dos partes cuadradas e iguales. En uno de los compartimentos se sitúan 150 cubos de 2.5 cm de lado que el paciente debe trasladar de uno en uno al otro compartimento. La duración del test es de 60 segundos, en los que el paciente debe mover el mayor número de cubos posible utilizando una sola mano. La punta de los dedos de dicha mano debe al menos cruzar la pantalla divisoria para que el cubo cuente como válido. Si el cubo no cae en la caja o se sale de ella una vez cruzada la pantalla también será contabilizado. Por otro lado, si el paciente coge más de un cubo en un movimiento, sólo se contabilizará uno de los cubos.

El paciente debe realizar el ejercicio con las dos manos, empezando primero con la mano del lado no afectado, si lo hubiera, cuya puntuación servirá de referencia [21]. Una puntuación más alta indicará mejor destreza manual y registrando las puntuaciones del paciente a lo largo del tiempo podemos fácilmente ver la progresión del tratamiento. Una comparativa de puntuaciones que puede servir de referencia para evaluar el desempeño del paciente fue realizada por Mathio-

Figura 1.13: Caja y cubos empleados en las pruebas.

wetz, Volland, Kashman y Weber en 1985 [22]. Este estudio estadístico separa a los pacientes por rangos de edad y determina sus puntuaciones medias con ambas manos como se puede ver en las Tablas 1.1 y 1.2, extraídas del citado estudio.

Para realizar la automatización del proceso se utilizará una cámara Kinect en su segunda versión, que se puede ver en la Figura 1.14. Kinect es una cámara RGB-D barata y accesible, con multitud de aplicaciones en el campo de la Robótica, que nos permitirá grabar el ensayo y posicionar en el espacio los cubos y las manos del paciente por medio de las cámaras de color y profundidad.

Aunque en un principio se planteó el uso de MatLab para el procesamiento y obtención de las imágenes, finalmente se decidió emplear otras herramientas que lograsen extraer todo el potencial de la cámara. Las imágenes se obtendrán mediante la API de Kinect [23], desarrollada por Microsoft, y el procesamiento se realizará utilizando las librerías de OpenCV para el lenguaje C++ [24].

El código completo del proyecto puede ser consultado en el portal GitHub en este enlace: https://github.com/LuisAngelGarcia/ProyectoBBT

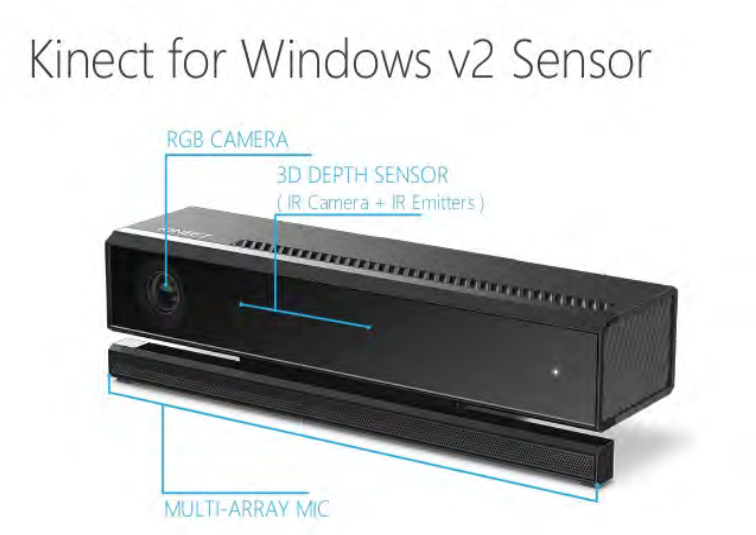


Figura 1.14: Cámara Kinect V2 [25].

Edad	Mano	Media	Desviación típica	Mínimo	Máximo
20 - 24	D	88	8.3	67	103
	I	83.4	7.9	66	99
25 - 29	D	86	7.4	63	96
	I	80.9	6.4	63	93
30 - 34	D	85.2	7.4	75	101
	I	80.2	5.6	66	92
35 - 39	D	84.8	6.1	71	95
	I	83.5	6.1	72	97
40 - 44	D	81.1	8.2	60	97
	I	79.7	8.8	57	97
45 - 49	D	82.1	7.5	68	99
	I	78.3	7.6	59	91
50 - 54	D	77.7	10.7	57	98
	I	74.3	9.9	53	93
55 - 59	D	74.7	8.9	56	94
	I	73.6	7.8	54	85
60 - 64	D	76.1	6.9	63	95
	I	73.6	6.4	62	86
65 - 69	D	72	6.2	60	82
	I	71.3	7.7	61	89
70 - 74	D	68.6	7	53	80
	I	68.3	7	51	81
+75	D	65	7.1	52	79
	I	63.6	7.4	51	81
Total	D	78.4	10.4	52	103
	I	75.8	9.5	51	99

Tabla 1.1: Desempeño en el BBT de mujeres no afectadas [22].

Edad	Mano	Media	Desviación típica	Mínimo	Máximo
20 - 24	D	88.2	8.8	70	105
	I	86.4	8.5	70	102
25 - 29	D	85	7.5	71	95
	I	84.1	7.1	69	100
30 - 34	D	81.9	9	68	96
	I	81.3	8.1	69	99
35 - 39	D	81.9	9.5	64	104
	I	79.8	9.7	56	97
40 - 44	D	83	8.1	69	101
	I	80	8.8	59	93
45 - 49	D	76.9	9.2	61	93
	I	75.8	7.8	60	88
50 - 54	D	79	9.7	62	106
	I	77	9.2	60	97
55 - 59	D	75.2	11.9	45	97
	I	73.8	10.5	43	94
60 - 64	D	71.3	8.8	52	84
	I	70.5	8.1	47	82
65 - 69	D	68.4	7.1	55	80
	I	67.4	7.8	48	86
70 - 74	D	66.3	9.2	50	86
	I	64.3	9.8	45	84
+75	D	63	7.1	47	75
	I	61.3	8.4	46	74
Total	D	76.9	11.6	45	106
	I	75.4	11.4	43	102

Tabla 1.2: Desempeño en el BBT de hombres no afectados [22].

Capítulo 2

Estado del Arte

En este capítulo se repasarán los avances realizados hasta la fecha en torno a la automatización del Ensayo de Caja y Cubos (BBT) por otros investigadores, qué objetivo perseguían con sus proyectos y de qué estrategias se han servido, así como su grado de acierto. Además, se expondrán algunos otros proyectos que han servido de ayuda e inspiración para este trabajo por compartir características similares.

2.1. Precedentes al caso de estudio

El problema propuesto, al que se ha tratado de encontrar una solución alternativa, ya ha sido investigado con anterioridad por otros autores utilizando aproximaciones distintas o con propósitos diferentes.

El precedente más inmediato y del que deriva la presente investigación es el sistema desarrollado por Oña et al. [26] [27] utilizando Matlab para el procesamiento de datos y la primera versión de Kinect. Este sistema se basaba en la segmentación por colores para realizar la cuenta de los cubos. La segmentación consiste en separar los colores objetivo de la imagen de color; así se crearían imágenes binarias en las que se resalten los píxeles de los colores especificados (rojo azul, verde y amarillo). Luego se buscan conjuntos de píxeles vecinos en cada una de ellas que

formen aproximadamente un cuadrado. La cuenta de cubos totales consiste en la suma de las cuentas de cada color. De este modo, se consiguen ratios de acierto muy altos con un número bajo de cubos. La principal limitación es que los cubos apilados pueden esconder el color del cubo inferior y que dos cubos muy próximos del mismo color pueden ser interpretados como el mismo cubo. Además, este método, que localiza continuamente los cubos, consume gran cantidad de recursos del procesador. Este proyecto contaba, además, con una interfaz gráfica muy intuitiva y con gran cantidad de información útil que se puede ver en la Figura 2.1.

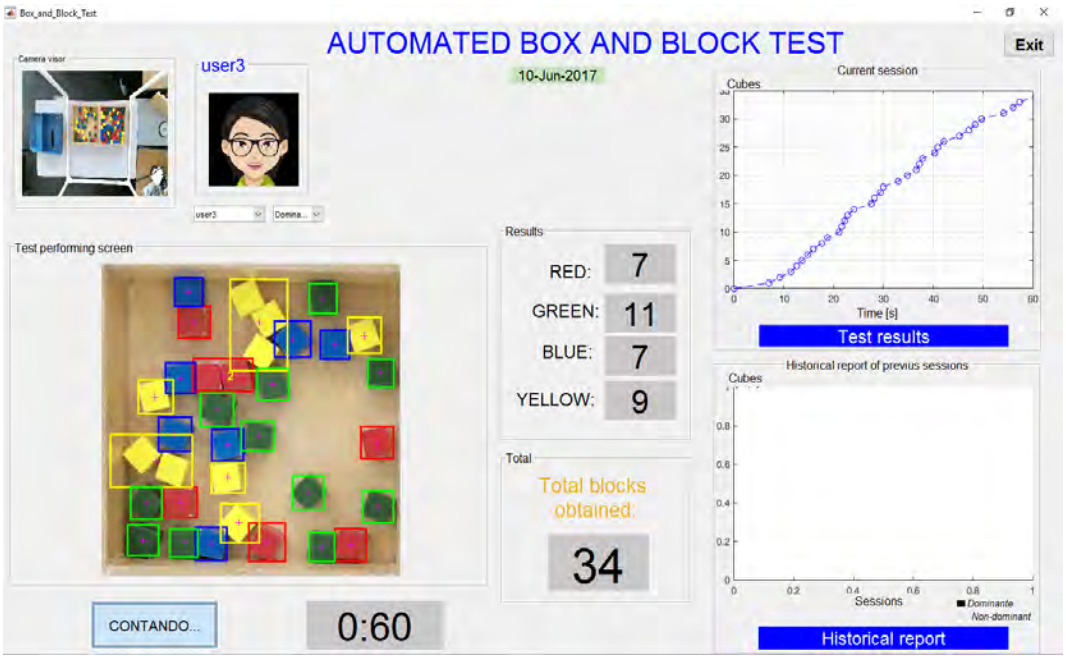


Figura 2.1: Interfaz gráfica del proyecto [26].

Otra aproximación interesante es la de Hsiao et al. [28]. Este proyecto, en lugar de basarse en la adquisición de imágenes de color, explota otra de las características más notables de la cámara Kinect, la obtención de mapas de profundidad. La estrategia utilizada para el cómputo de los cubos desplazados consiste en realizar «cortes», umbralización, en la imagen de profundidad a diferentes alturas prefijadas para cubrir las tres alturas posibles a las que puede encontarse un cubo y someterlos a detección de contornos para localizar las pilas de cubos (Figura 2.2). La suma de los resultados de cuenta de las distintas alturas consigue contabilizar con mucha precisión un gran número de cubos. Este proyecto, además, incluye una solución para otro de los requisitos

del ensayo. Como se ha explicado, los cubos no deben contabilizarse si parte de la mano no llega a pasar al otro lado de la pantalla. Esta propuesta lleva el seguimiento de la mano del paciente, para corroborar su paso correcto sobre la pantalla divisoria y recoger datos relativos a su movimiento, como se aprecia en la Figura 2.3.

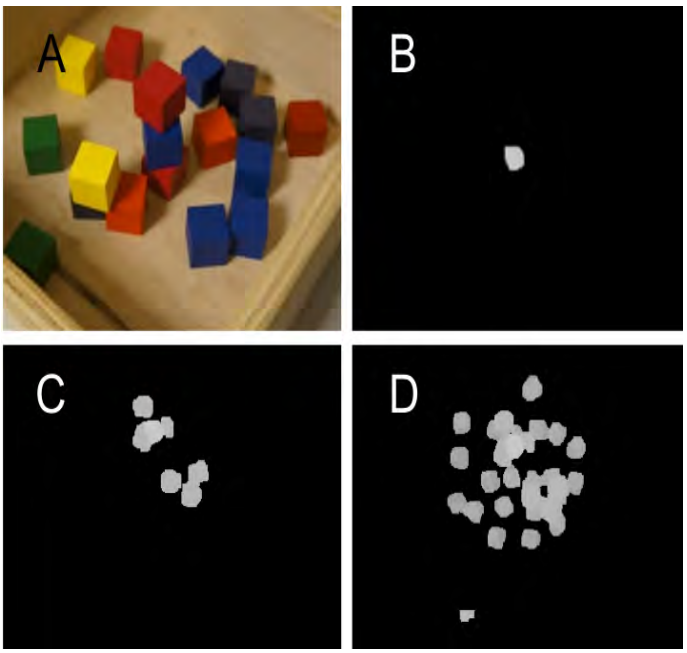


Figura 2.2: Cortes a tres alturas diferentes para localizar todos los cubos [28].

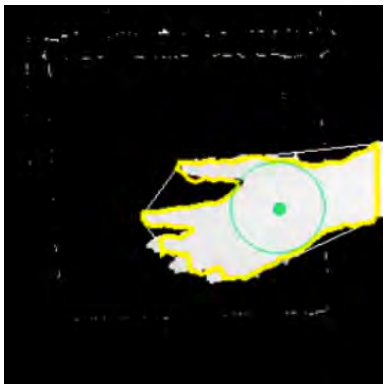


Figura 2.3: Localización y seguimiento de la mano usando Convex hull [28].

2.2. Otros sistemas de automatización

Existen, además, otros proyectos de automatización del ensayo cuyo fin último no es contar los cubos trasladados por el paciente. Estos proyectos se valen de la visión artificial para facilitar la rehabilitación y obtener información extra que pueda ayudar al médico a utilizar un tratamiento adecuado y personalizado para cada paciente.

Desde Microsoft se ha hecho una propuesta tratando el problema de una manera radicalmente distinta [29]. El sistema, que puede verse en la Figura 2.4, usa una cámara Kinect para simular en Realidad Virtual la caja y los cubos, de manera que una mano virtual en la pantalla sigue los movimientos hechos por el paciente. Con este enfoque se pretende por un lado simplificar la automatización de la prueba, teniendo sólo que localizar la mano del paciente, un caso ampliamente estudiado, y por otro lado se proporciona un medio más ameno y que se puede llevar a la casa del paciente para que este pueda realizar la prueba en cualquier momento y enviar los resultados de manera remota a su médico.

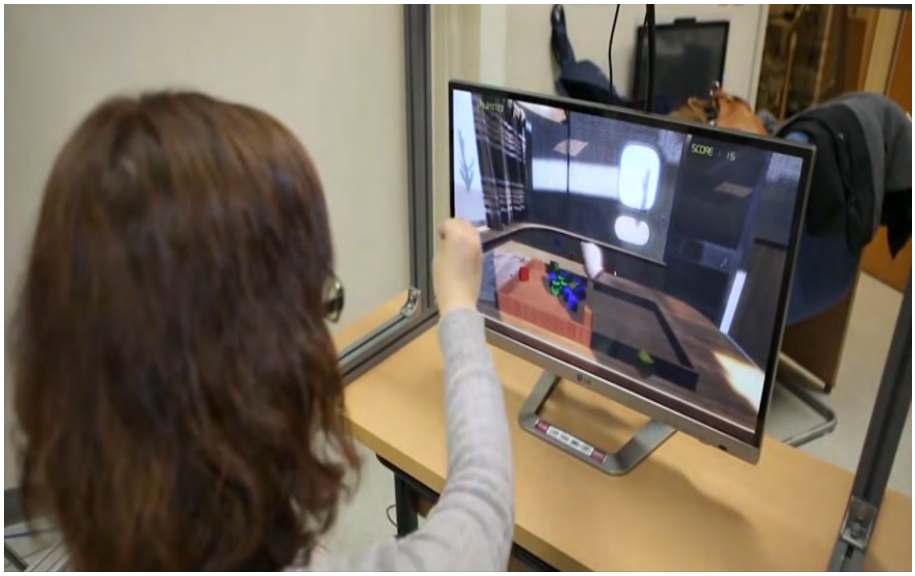


Figura 2.4: Solución en Realidad Virtual implementada por Microsoft [29].

Otros investigadores también se han hecho algunos avances centrados en la monitorización del movimiento de la mano y el brazo. No sólo de cara a la rehabilitación del paciente, sino también para comprobar el ajuste perfecto de prótesis [30] [31]. Estos proyectos se caracterizan

por el uso de marcadores colocados en posiciones estratégicas del brazo del paciente como la muñeca, el dorso de la mano o el codo (Figura 2.5), de manera que se puede llevar el seguimiento completo de velocidad y posición de diferentes articulaciones de una manera sencilla y efectiva.

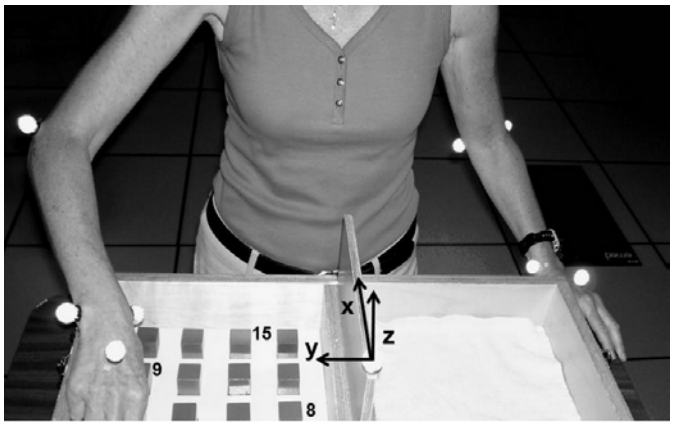


Figura 2.5: Detalle de la monitorización del movimiento de las articulaciones [30].

2.3. Proyectos indirectamente relacionados

Otro proyecto en el que se ha basado la presente investigación, a pesar de no tener que ver con el BBT, es el de Chen, Dietrich y Miller [32]. Consiste en el uso de una Kinect y algoritmos de *Template Matching* para localizar *Cubelets*, pequeños robots para niños que se construyen de manera modular, y permitir su ensamblaje automatizado por medio de un brazo robótico. Los módulos de los que se componen los *Cubelets* son cúbicos, como su propio nombre indica, y están esparcidos por el campo de visión de la cámara en cualquier ángulo de rotación; de ahí la similitud con este proyecto. Los buenos resultados obtenidos en este proyecto para la detección de cubos fueron un aliciente para decantarse por el *Template Matching* como método de generación de candidatos en esta versión del ABBT. En las Figuras 2.6 y 2.7 se pueden observar la aplicación de un algortimo de detección de bordes y el resultado final.

Especial mención merece también el proyecto [33], en el que se basa el código de discernimiento de candidatos con altas probabilidades de ser cubos y candidatos con menor probabilidad de ser cubos tras aplicar el *Template Matching* a la imagen obtenida de la cámara RGB. El

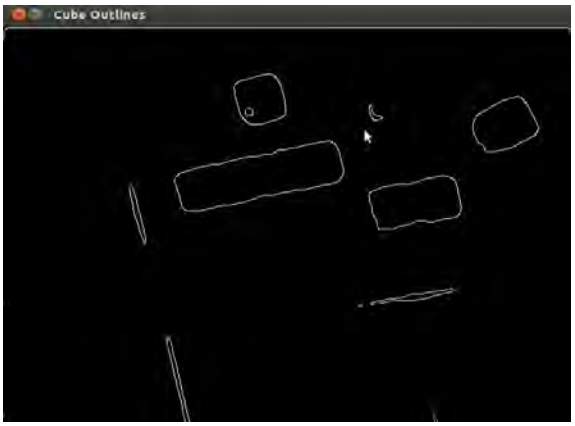


Figura 2.6: Aplicación de un filtro Canny a la imagen [32].

Figura 2.7: Resultados de la detección usando Template matching [32].

proyecto anteriormente citado consiste en el análisis de un blíster de pastillas para comprobar si alguno de los compartimentos no está correctamente lleno y proceder a la retirada de dicho blíster. El algoritmo debe, por tanto lidiar con la posibilidad de encontrar hasta 10 candidatos perfectos en el caso de que el blíster contenga las diez pastillas necesarias. La estrategia seguida consiste en recorrer la imagen resultado del Template Matching buscando el punto con valor máximo. Una vez encontrado, se almacena su posición y se «pinta de negro», se sobreescribe el valor del píxel por el valor mínimo de una imagen en escala de grises. A continuación se vuelve a buscar el siguiente punto máximo por encima de cierto umbral y se repite el proceso hasta que no queden más máximos que cumplan la condición umbral. De este modo se obtienen los candidatos perfectos hallados en la imagen y se localizan de manera rápida los huecos vacíos. En las Figuras 2.8 y 2.9 a continuación se describe el proceso:

Figura 2.8: Imagen original del blíster al que le faltan dos pastillas [33].

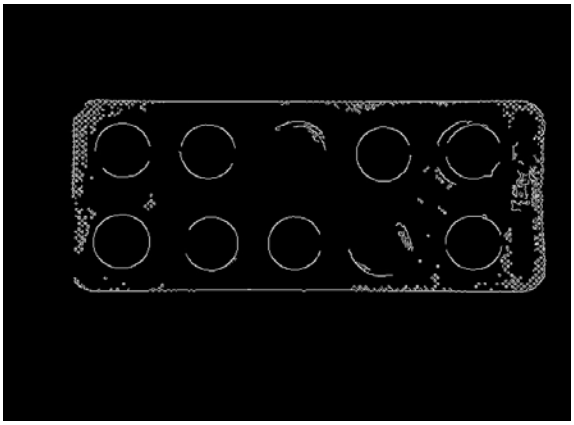


Figura 2.9: Imagen tras la detección de bordes [33].

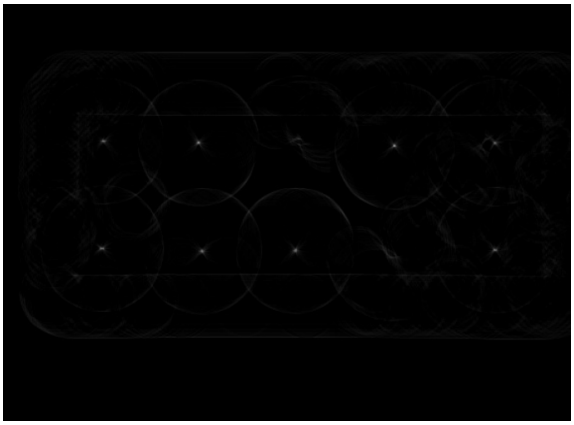


Figura 2.10: Resultados de la aplicación de Template matching [33].

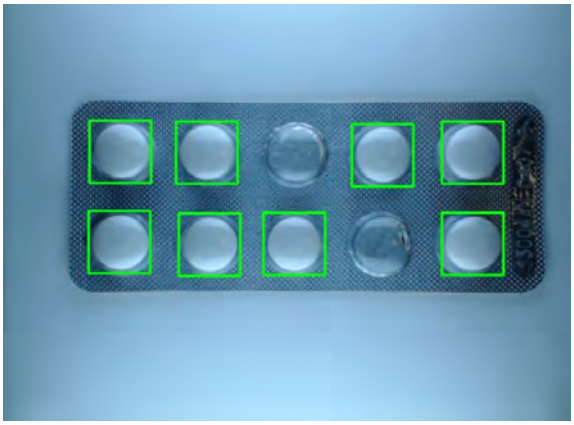


Figura 2.11: Resultados de la detección [33].

Capítulo 3

Conceptos teóricos

En este proyecto se incluyen varias técnicas de posicionamiento y reconocimiento de objetos que funcionan conjuntamente para conseguir unos buenos resultados utilizando reconocimiento basado en el color y reconocimiento por mapas de profundidad. La información teórica de este apartado ha sido obtenida de un curso online gratuito impartido por profesores de la Universidad Autònoma de Barcelona [34].

3.1. Tecnología y Aplicaciones

- **Espacios de color:** la manera habitual de adquirir y mostrar imágenes se apoya en el sistema RGB. La tecnología RGB basa su funcionamiento en las propiedades ópticas de la luz. Utiliza tres tipos de sensores, cada uno de los cuales es sensible a una longitud de onda de luz: luz roja (700 nm), luz verde (546.1 nm) y luz azul (435.8 nm). La combinación de estos colores en distintas intensidades (comprendidas entre 0 y 255 para una resolución estándar de 8 bits) puede dar lugar a todos los demás. Cada terna de sensores RGB compone un píxel, la unidad mínima de color variable de una imagen. Para detectar objetos nos valemos fundamentalmente del color de estos píxeles y de su relación con sus vecinos.

Las imágenes RGB se pueden transformar a otros sistemas de representación mediante fórmulas matemáticas. Cada uno de los diferentes espacios de color tiene diferentes usos, ventajas e inconvenientes. En este proyecto, además del RGB recogido por la cámara, se ha utilizado un cambio al espacio HSV. El espacio HSV utiliza tres canales (*Hue/Saturation/Value*) para representar los colores. El primer canal representa el matiz del color en una escala de 0 a 360 grados, esto es el color «real», independiente de las condiciones de luminosidad bajo las que fue tomada la imagen. El segundo canal contiene la saturación del color, su valor de blanco. Una saturación del 0 % correspondería a un color blanco, mientras que la saturación del 100 % sería el color puro indicado por el matiz. El tercer canal es el Valor, también llamado brillo, y representa el porcentaje de negro que tiene el color, donde los valores bajos contienen la mayor cantidad de negro. Este sistema de representación tiene, por un lado, la ventaja de que un determinado color puede ser clasificado utilizando para ello únicamente su valor en el canal *Hue*, simplificando mucho la clasificación y segmentación por colores y, por otro, que se reduce considerablemente la influencia de la luz ambiental en la clasificación.

La transformación de RGB a HSV se realiza mediante las siguientes expresiones:

$$H = \begin{cases} \text{Indefinido}, & \text{si } MAX = MIN \\ 60° \times \frac{G-B}{MAX-MIN} + 0°, & \text{si } MAX = R \\ 60° \times \frac{G-B}{MAX-MIN} + 0°, & \text{si } MAX = R \text{ y } G \geq B \\ 60° \times \frac{G-B}{MAX-MIN} + 360°, & \text{si } MAX = R \text{ y } G < B \\ 60° \times \frac{B-R}{MAX-MIN} + 120°, & \text{si } MAX = G \\ 60° \times \frac{R-G}{MAX-MIN} + 0°, & \text{si } MAX = B \end{cases} \tag{3.1}$$

$$S = \begin{cases} 0, & \text{si } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{en otro caso} \end{cases} \tag{3.2}$$

$$V = MAX \tag{3.3}$$

Donde MAX y MIN son el máximo y el mínimo de las componentes RGB del píxel que se quiere transformar.

Existen numerosos proyectos centrados en el reconocimiento y seguimiento de objetos y partes del cuerpo utilizando cámaras RGB que logran buenos resultados. [35] Al ser, además, sensores baratos y fáciles de conseguir y adaptar a las necesidades, su uso está bastante extendido. [36] En los últimos años, su uso ha quedado relegado a un segundo plano por la introducción de las cámaras RGB-D y la investigación en reconocimiento 3D, sin embargo, se siguen utilizando en smartphones. Dos de los usos más conocidos son la detección automática de caras y enfoque de las fotos [37] y el reconocimiento de códigos QR [38]. Hay, además, líneas de investigación muy interesantes en torno a la traducción del lenguaje de signos a varios idioma [39]. Una futura ampliación de este proyecto que podría ser interesante consistiría en monitorizar la seguridad con la que el paciente mueve la mano: velocidad, aceleración, temblores, precisión del movimiento... [40]

- **Profundidad:** existen dos tecnologías que nos permiten realizar mapas de profundidad. Por un lado tenemos la tecnología láser, que precisa de poco procesamiento, lo que permite una alta velocidad de captura de imágenes a baja resolución. Podemos distinguir dos métodos que utilizan láser. El de «tiempo de vuelo» (*time-of-flight*) proyecta un haz de luz y mide el tiempo que tarda en llegar al objeto, rebotar y volver; conocida la velocidad de la luz, es sencillo calcular la distancia entre el emisor y el objeto. Este método funciona bien incluso a distancias del orden de los kilómetros, aunque en distancias cortas no es demasiado preciso debido a la dificultad para medir tiempos pequenõs. El segundo método es la triangulación. El haz de luz se proyecta sobre la superficie y se localiza mediante una cámara adecuada. Conocido el ángulo bajo el cual se ve el punto luminoso desde la cámara, por trigonometría se halla fácilmente la distancia. La principal limitación de este método es que los objetos deben estar cerca del sensor. Por contrapartida, cuenta con mejor precisión que el *time-of-flight*.

  Por otro lado, la tecnología basada en en infrarrojos que incorpora Kinect utiliza también el método de triangulación, pero al ser luz infrarroja, fuera del espectro visible, evita problemas con la iluminación del entorno. Obtiene mejores resoluciones a costa de bajar la velocidad de procesamiento. Sin embargo, dado que la Kinect no es una simple cámara de profundidad, sino que funciona también como cámara RGB, es más versátil y más utilizada para aplicaciones en robótica que requieren análisis espacial. [41]

Además de las imágenes RGB y de profundidad, se pueden obtener otros tipos de imágenes como las de infrarrojo lejano, disparidad o flujo óptico. Estas tienen campos de aplicación mucho más específicos, requieren de otros elementos hardware diferentes para ser adquiridas y no tienen especial interés en este proyecto.

3.2. Algoritmos y técnicas de detección

Cada una de las tecnologías utilizadas lleva asociados unos procedimientos de tratamiento de la información, conocidos como algoritmos, que permiten la correcta interpretación de los datos. Además, dado que en el procesado de imágenes en muchas ocasiones se prescinde del color, muchos de los algoritmos utilizados para imágenes RGB tienen su equivalente en imágenes de profundidad. A continuación explicaremos el proceso básico de detección e introduciremos la terminología fundamental que se utiliza.

El primer paso de todo proceso de detección de objetos es analizar la imagen para extraer las características y obtener lo que se conoce como *descriptores*. Los descriptores son conjuntos de datos en forma de vector cuyas coordenadas describen cada una una pequeña parte de la imagen, que puede ser desde un píxel a un bloque de píxeles. Una vez obtenidos los descriptores de la imagen se pasa a la *generación de candidatos*, la fase en la que se analizan las imágenes en busca de *ventanas* que contienen descriptores similares a los del objeto que estamos buscando. Este proceso suele estar ligado al proceso de *clasificación* por un proceso iterativo.

Existen tres tipos de ventanas: ventanas positivas, que verdaderamente contienen el objeto, ventanas de falsos positivos y ventanas redundantes alrededor del objeto de estudio, pero ligeramente desplazadas.

En la siguiente etapa, la de *clasificación*, se busca aprender un modelo que permita distinguir automáticamente una ventana falsa de un correcta y discernir cuál de las ventanas con información redundante es la mejor. Para ello, se utiliza un proceso iterativo en el que una persona etiqueta algunas ventanas, normalmente ejemplos difíciles positivos o negativos, el ordenador aprende un modelo, se prueba, la persona corrige los fallos y anota alguna muestra más, se aprende otro modelo... Tras un número de iteraciones, el sistema es capaz de aprender los descriptores

de una ventana positiva y diferenciarla de una negativa.

La última fase del proceso de creación de un detector es la *evaluación*. Se toman unas muestras y se las somete a detección. Se contabilizan los positivos, los falsos positivos, los negativos y los falsos negativos y se construye la «Matriz de Confusión», de la que se pueden extraer la sensibilidad, especificidad, precisión y exactitud del detector, medidas de su desempeño.

La estructura básica de un detector puede verse en la Figura 3.1.

Figura 3.1: Proceso básico de detección [34].

3.2.1. Algoritmos utilizados en este proyecto

En esta sección se detallarán los procedimientos utilizados en el sistema y se explicará la base teórica que los fundamenta.

Algoritmos de extracción de características

- **Basados en el color**

  Las detecciones basadas en el color suelen ser bastante sensibles a cambios de luminosidad o color, por lo que se suele aplicar a las imágenes un tratamiento preliminar, ya sea un cambio del espacio de color, como a HSV, explicado anteriormente, o un tratamiento de *White-Patch*, normalizar la imagen en base al valor de máxima intensidad en cada canal (este filtro está implementado en el código, pero finalmente no se utilizó):

$$\text{White-patch: } (R \quad G \quad B) \rightarrow \left(\frac{255}{R_{max}} \cdot R \quad \frac{255}{G_{max}} \cdot G \quad \frac{255}{B_{max}} \cdot B \quad \right)$$

En el programa se utilizan dos métodos de extracción de características: la umbralización y el *Template matching*.

La umbralización consiste en crear una imagen binaria, blanca y negra, en la que los píxeles que cumplen ciertas características se «pintan» de color blanco y los demás, se etiquetan como fondo y se ponen de color negro.

Las características que definen los umbrales se refieren siempre a los valores que contienen los píxeles y, por tanto, la umbralización puede ser aplicada a cualquier tipo de imagen. En este proyecto se aplica la umbralización a la imagen de profundidad, una imagen de 8 bits, de un solo canal (escala de grises) para hallar los píxeles que pertenecen al borde de la caja, a la pantalla divisoria y para diferenciar el compartimento lleno del vacío. La umbralización también se utiliza con la imagen de color para hacer una segmentación por color de los cubos y lograr una detección más precisa.

La segunda opción, que no sólo es aplicable en detectores basados en color, es el *Template Matching*. Se toma una ventana que contiene uno de los objetos a detectar y se va deslizando por toda la imagen comparando píxel a píxel la diferencia entre la plantilla y la imagen y anotando el resultado en el píxel de la esquina superior izquierda del trozo de imagen examinado. Una vez se ha pasado la plantilla por toda la imagen se hallan los píxeles cuya diferencia es mínima. En estos píxeles se situarán las ventanas con candidatos. La diferencia se calcula de la siguiente manera:

$$D = \sum_{i=-x,j=-y}^{i=x,j=y} (T_{i,j} - I_{i,j})^2$$

Donde T e I son las intensidades de un píxel de la plantilla y de la imagen respectivamente.

- **Basados en características locales (Locale Features)**

  Consisten en deslizar por la imagen lo que se conocen como un kernel, una matriz cuadrada de números que, al hacer una convolución con las intensidades de la imagen son capaces de detectar cambios en el gradiente de intensidad, lo que típicamente significa que en un determinado punto hay un borde. Dependiendo de la orientación del borde que se quiere

detectar, los números del kernel varían de posición y valor. Estos son dos ejemplos sencillos para detectar bordes verticales (Tabla 3.1) y horizontales (Tabla 3.2):

-1	0	1
-1	0	1
-1	0	1

Tabla 3.1: Filtro de detección vertical

1	1	1
0	0	0
-1	-1	-1

Tabla 3.2: Filtro de detección horizontal

Para definir las fronteras de los objetos se trabaja con conjuntos de kernel de distintas formas, escalas y orientaciones que se adecúen a los distintos tipos de borde posibles.

El algoritmo utilizado en el sistema para la detección de bordes es el *Detector de Bordes de Canny*. Este algoritmo es una versión mejorada de un filtro tipo Sobel, que simplemente aplica un kernel a la imagen. El Canny busca diferencias de gradiente mediante un kernel y después clasifica los píxeles candidatos en función de dos umbrales como borde o no borde. De este modo se logra una detección de borde muy fina y precisa, tanto en buena localización como en ratio de falsos positivos y redundancias.

Algoritmos de anotación, refinación y clasificación

- **Ventana deslizante y pirámide**

  El primer paso consiste en la localización de posibles candidatos dentro de ventanas. Para ello, se desplaza una ventana de un tamaño canónico por la imagen con desplazamientos que suelen estar determinados por el tamaño de los bloques en los que se ha dividido la imagen para extraer sus descriptores. La ventana se desliza por la imagen y se anotan las posiciones en las que los descriptores de la imagen podrían representar al objeto buscado.

  Esto sólo nos permitiría localizar objetos que cupiesen en la ventana canónica. Para encontrar objetos más grandes se reescala la imagen, haciéndola gradualmente más pequeña, una *pirámide* de imágenes cada vez más pequeñas, mientras la ventana quepa en ella y repitiendo el deslizamiento de la ventana. Así nos aseguramos de captar posibles objetos que se encuentren en primer plano.

  En nuestro proyecto, los cubos tienen diferencias de tamaño despreciables, por lo que podemos prescindir de la pirámide. Las posiciones de las ventanas de candidatos nos vienen

dadas por los mínimos hallados en el *Template Matching* y el tamaño de la ventana canónica es el tamaño de la plantilla, un cuadrado de 44 píxeles de lado.

- **Refinación**

  El proceso de ventana deslizante genera muchas ventanas alrededor de posibles candidatos, por lo que es necesario descartar algunas de ellas. En el algoritmo de refinación se comienza ordenando los candidatos por orden de probabilidad de contener al objeto. Luego se coge la primera y se comprueba si se solapa más de un porcentaje fijado con alguna de las demás ventanas candidatas. De no ser así, se añadirá al grupo de candidatos finales. Repitiendo el proceso se logra tener candidatos no redundantes.

  En este proyecto, la refinación de candidatos y su clasificación están intimamente ligadas en varios algoritmos en cascada que se valen de características de color, bordes y profundidad para eliminar candidatos redundantes y falsas detecciones y añadir cubos a diferentes alturas, de manera que se logra un resultado bastante preciso, tal y como se explicará más adelante de manera detallada.

Algoritmos de evaluación

- **Hold-Out**

  El método Hold-Out utiliza un determinado porcentaje del total de muestras para hacer el entrenamiento del detector y las restantes para validar su funcionamiento. El proceso habitual de evaluación se basa en construir la matriz de confusión y calcular a partir de ella ciertos parámetros estadísticos. La matriz de confusión se implementa de la manera mostrada en la Tabla 3.3

		Resultado de la clasificación	
		Positivos	Negativos
Instancias reales (Ground Truth)	Positivos	Reales Positivos	Falsos Negativos
	Negativos	Falsos Positivos	Reales Negativos

Tabla 3.3: Matriz de Confusión

A partir de los resultados se puede calcular:

- *Exactitud:* medida de la proximidad entre el resultado y la clasificación perfecta.

$$Exactitud = \frac{RealesPositivos + RealesNegativos}{PrediccionesTotales}$$

- *Precisión:* medida de la calidad de la respuesta del clasificador.

$$Precisin = \frac{RealesPositivos}{RealesPositivos + FalsosPositivos}$$

- *Sensibilidad:* medida de la eficiencia de la clasificación de todos los elementos de la clase.

$$Sensibilidad = \frac{RealesPositivos}{RealesPositivos + FalsosNegativos}$$

- *Especificidad:* medida de la eficiencia de la clasificación de todos los elementos que no son de la clase.

$$Especificidad = \frac{RealesNegativos}{RealesNegativos + FalsosPositivos}$$

3.2.2. Otros algoritmos

En este apartado se analizarán algunos algoritmos importantes usados en el campo de la visión artificial que, aunque no se han utilizado en el proyecto, sí fueron considerados y estudiados durante el proceso de documentación.

Algoritmos de extracción de características

- **Local Binary Patterns (LBP)**

  Los Patrones Binarios Locales o *Local Binary Patterns* son un método de extracción de características de una imagen o una región. En los LBP a cada píxel se le asocia un número entre en función del valor de sus píxeles vecinos. Si el valor de la intensidad del vecino es mayor o igual que el del píxel central, al vecino se le asocia un 1 y un cero en caso contrario. Repitiendo el proceso con los 7 vecinos restantes, se obtiene un número binario. Pasándolo a decimal, el píxel central tendría un valor entre 0 y 255. Si se repite el proceso para cada

píxel de la imagen se podría construir un histograma con las frecuencias de aparición de cada número. Este histograma constituiría un descriptor de 256 dimensiones para nuestra imagen.

Ahora bien, 256 dimensiones son muchas para un descriptor. Es por eso que se suele recurrir al LBP Uniforme. Para construirlo se cuenta el número de transiciones de 1 a 0 y de 0 a 1 del código binario asociado a cada píxel. Esto resulta en 59 posibles combinaciones, y, por tanto, en un histograma de 59 dimensiones, mucho más manejable.

En la práctica, los histogramas de toda la imagen son poco precisos y son insensibles a ciertas modificaciones de la imagen por lo que arrojan falsos positivos. Para corregir este defecto se calculan los histogramas para bloques de la imagen, que pueden o no solaparse entre sí. El descriptor resultante sería la concatenación de todos los histogramas calculados.

- **Sobel**

  El operador de Sobel es una simplificación del cálculo del gradiente. Se utilizan dos kernel de 3x3 píxeles, uno para la dirección vertical y otro para la horizontal, que se convolucionan con la imagen. El resultado final se obtiene como la raíz cuadrada de la suma de los cuadrados de las dos imágenes convolucionadas. Sirve, por tanto, en la detección de bordes.

- **SIFT**

  El método SIFT (*Scale-Invariant Feature Transform*) consiste en la localización de ciertos puntos clave contenidos en el objeto a detectar que sean invariantes a cambios de posición, como son las esquinas. Estos puntos se almacenan y se tomarán como referencia para compararlos con las imágenes que se quieren analizar. Cuando el detector recibe las imágenes busca coincidencias entre los puntos de la base de datos y los de la imagen buscando la distancia mínima entre puntos vecinos. La detección de tres o más puntos clave indica la posible presencia de un objeto, por lo que esa zona se estudiará con detenimiento y se extraerá la probabilidad de la región de contener dicho objeto.

- **SURF**

  Este método es una evolución del SIFT que logra resultados más rápidos utilizando un método distinto para buscar los puntos clave. En lugar de calcular el gradiente de la manera habitual, se utilizan imágenes integrales y filtros de tipo Haar para aproximarlo.

Con este método se obtiene no sólo la posición del punto de interés, sino también su orientación. Como la orientación no es siempre necesaria, la variante U-SURF calcula simplemente la posición con un menor coste computacional.

Algoritmos de anotación, refinación y clasificación

- **Bootstrapping**

  El *bootstrapping* es un método de anotación de candidatos para que el sistema pueda realizar un modelo automático de clasificación. En este método las muestras anotadas son todas negativas (representan fondo) y a ser posible, difíciles. Esto ayuda a que la frontera se genere más rápidamente y sea más precisa.

- **Aprendizaje activo**

  En el aprendizaje activo se usan los mismos principios que en el *bootstrapping*, solo que aquí se anotan fundamentalmente muestras positivas relevantes, aunque también se pueden anotar algunas negativas.

- **Regresión Logística**

  La Regresión Logística es un proceso de clasificación basado en establecer una frontera entre las muestras positivas y las negativas minimizando el error cometido. La frontera debe ser tal que su producto escalar con el descriptor de una muestra positiva sea mayor que cero y menor que cero con muestras negativas. Sin embargo, clasificar las muestras como positivas o negativas es un sistema muy simple y puede llevar a errores. Por eso se introduce la función logística, una función probabilística que determina con qué probabilidad la muestra es positiva o negativa. De esta manera se puede determinar un umbral por debajo del cual clasificar una muestra como negativa y conseguir una mejor tasa de positivos.

Algoritmos de evaluación

- **Validación Cruzada (Cross Validation)**

  Una evolución del Hold-Out es la validación cruzada, con la que se consiguen resultados más exactos y con menor variabilidad. Consiste en dividir el total de muestras en k conjuntos,

luego utilizar k-1 conjuntos para entrenar y evaluar en el siguiente. Esto se repetiría k veces hasta utilizar todos los conjuntos en la evaluación. Los resultados finales se promedian.

3.2.3. Detectores completos

En esta sección se presentan algunos detectores con aprendizaje automático, compuestos por varios algoritmos de las categorías antes expuestas y que tienen aplicación directa en multitud de sistemas de visión artificial muy extendidos en la tecnología actual.

- **HOG/SVM**

  Estos detectores se basan en la combinación de un extractor de características HOG (*Histogram of Gradients*) y el algoritmo clasificador SVM (*Support Vector Machine*).

  Los HOG se basan en extraer para cada píxel la dirección y el módulo del vector gradiente calculado como:

$$dx = I(x+1, y) - I(x-1, y)$$

$$dy = I(x, y+1) - I(x, y-1)$$

  La imagen se divide en bloques, de una manera parecida a como se hace en LBP, y en cada bloque se calcula un histograma. La concatenación de los histogramas da lugar al descriptor. Para calcular el histograma se clasifican en grupos todos los vectores de un bloque según el ángulo de su dirección. El número de grupos n es un parámetro a fijar por el usuario y determina los intervalos de ángulos como $180^o/n$. En cada intervalo se suman los módulos de los vectores pertenecientes a éste.

  Ahora bien, este método es bastante sensible con vectores que se encuentran justo en la frontera entre dos intervalos o cerca de ella. Por eso se suelen hacer interpolaciones en orientación de tal manera que un vector contribuya simultáneamente a varios intervalos con diferentes pesos. De un modo similar, a píxeles cercanos a los bordes de un bloque se les aplica una interpolación espacial con los bloques vecinos.

  El algoritmo SVM se basa en encontrar una frontera lineal de separación entre las muestras positivas y las negativas de tal manera que la frontera maximice la distancia entre las dos

muestras de ambas clases más cercanas entre sí, las llamadas «vectores de soporte». En conjuntos no linealmente separables se pueden añadir tolerancias (variables de holgura) para relajar la condición de linealidad o utilizar el llamado «truco del kernel», llevar el espacio de características a un orden superior donde sí es posible separarlo linealmente.

Una descripción gráfica de las muestras, la frontera y los vectores de soporte puede consultarse en la Figura 3.2.

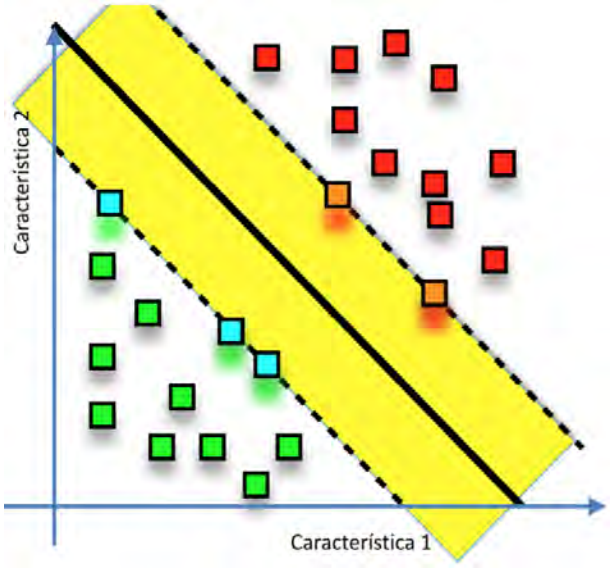


Figura 3.2: Clasificación de muestras según el método SVM [34].

- **Haar/Adaboost**

  Estos detectores se componen de un extractor de características con filtros de Haar y una cascada de clasificadores que utilizan el algoritmo Adaboost.

  Los filtros de Haar son combinaciones de rectángulos adyacentes del mismo tamaño, algunos ejemplos pueden verse en la Figura 3.3.

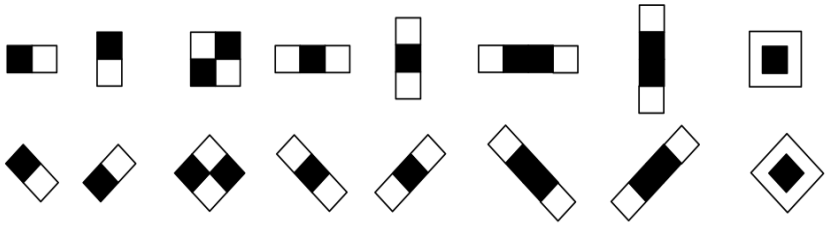


Figura 3.3: Diferentes formas de filtros Haar [34].

Los rectángulos negros son zonas de contribución positiva y los blancos, de contribución negativa. El resultado del filtro es la diferencia de la suma de los valores de los píxeles en las zonas negras y la suma de los píxeles de las zonas blancas. Los filtros se hacen pasar por toda la imagen a todas las escalas horizontales y verticales posibles y cada filtro a cada escala genera una característica de Haar. Este procedimiento genera una cantidad enorme de características incluso para una imagen pequeña, pero es poco costoso a nivel de cálculos debido al uso de imágenes integrales. La imagen integral es una imagen en la que cada píxel contiene la suma de los píxeles superior e izquierdo, de tal manera que calcular la suma de los píxeles de un rectángulo que empiece en la esquina superior izquierda de la imagen es tan sencillo como mirar el valor del píxel de su esquina inferior derecha. De este modo, al utilizar la imagen integral, para calcular el valor de la suma de un rectángulo cualquiera de la imagen sólo se necesitan tres sumas.

El método Adaboost saca partido de la gran cantidad de información generada por los filtros de Haar. Con cada una de las características de Haar se aprende un clasificador débil, no mucho mejor que una clasificación aleatoria. Para entrenar el clasificador se asigna a cada muestra un peso: las muestras mal clasificadas en la iteración anterior reciben un peso mayor que las que han sido correctamente clasificadas. El umbral se fija en el punto en el que el error cometido se hace mínimo y se vuelven a asignar unos pesos ponderados con el error y utilizando el nuevo umbral. El proceso se repite un número determinado de veces para cada característica de Haar, obteniéndose así un conjunto de clasificadores débiles. El clasificador global «fuerte» se obtiene como la suma ponderada con los errores de los clasificadores débiles.

A la hora de detectar objetos, podemos utilizar en cascada los clasificadores fuertes aprendidos para cada filtro de Haar. Esto significa que la ventana es evaluada por el primero de los clasificadores y si resulta positiva, pasa al siguiente. Si resultara negativa en cualquier clasificador de la cadena, se rechazaría. Debido a que la mayor parte de las ventanas de una imagen suelen ser fondo, este método permite analizar rápidamente gran cantidad de ventanas con bajo coste computacional, ya que muchas se pueden rechazar en los primeros niveles de la cascada.

- **Redes Neuronales Convolucionales (CNN)**

  Las Redes Convolucionales Neuronales (CNN o ConvNet en inglés) son algoritmos de extracción de características y clasificación muy utilizados en reconocimiento de imágenes, aunque bastante complejos. Están habitualmente formados por dos partes que actúan en serie. La primera parte extrae características de una imagen y la segunda, una red de neuronas totalmente conectadas, determina la probabilidad de que la imagen contenga el objeto buscado.

  La extracción de características se compone de dos pasos. En primer lugar se hacen convoluciones de la imagen con uno o varios kernel de tal manera que se obtienen tantas «versiones» de la imagen como filtros se hayan usado. El siguiente paso se conoce como *pooling* y consiste en reducir el tamaño de las imágenes. Esto se hace dividiendo las imágenes resultantes de la convolución en partes más o menos grandes (2x2, 3x3 píxeles...) y extrayendo de cada subgrupo un valor que puede ser el valor de intensidad más alto del grupo, una media de todos o cualquier otro método. Con cada valor extraído se forma una nueva imagen más pequeña, pero que conserva las características fundamentales de la original. Por lo general se suelen utilizar varias etapas de convolución y pooling sucesivas, no sólamente una, para reducir el tamaño de las imágenes lo máximo posible y minimizar los cálculos en la siguiente fase.

  Dicha segunda fase consiste en una red neuronal totalmente conectada. Las redes neuronales son conjuntos de objetos llamados nodos o neuronas organizados en capas.

  Cada neurona recibe una información, la procesa según una fórmula prefijada y la transmite a la siguiente capa de neuronas. Se dice que están totalmente conectadas si la salida de cada neurona va a parar a las entradas de todas las neuronas de la siguiente capa.

  La información de entrada se trata de diferente forma según de qué neurona proceda. La salida de una neurona tiene la forma

  $$y = \sum_{n=0}^{N} \quad f\left(w_n \cdot x_n\right)$$

  por lo que se puede ver que cada entrada tiene asociado un peso distinto. En la asignación de los pesos (aprendizaje) se utiliza la técnica de *backpropagation*.

  Para el entrenamiento del detector se empieza dando valores aleatorios a los pesos y se hace una primera prueba. En un caso ideal la respuesta del sistema ante una imagen positiva

será un 1 y un 0 ante imágenes negativas. En la práctica es imposible, menos aún usando pesos aleatorios. En este punto entra en juego la propagación hacia atrás de errores. Con los resultados de la primera prueba se calculan los errores cometidos y se realimentan al sistema para cambiar los pesos. Iterando varias veces con diferentes ejemplos se consigue entrenar al clasificador con los pesos que minimicen el error.

Capítulo 4

Herramientas utilizadas

En este capítulo se analizarán las características de los elementos utilizados para llevar a cabo el sistema propuesto. El proyecto ha sido elaborado y testado en el Parque Científico UC3M de Leganés, en el Laboratorio de Robótica Asistencial. Estos medios han sido proporcionados por la Universidad Carlos III de Madrid.

4.1. Hardware

- **Kit de ensayo BBT:** el kit, que puede verse en las Figuras 4.1 y 4.2, está compuesto de una caja de madera con dos compartimentos cuadrados de 25.4 cm de lado y 8.5 cm de alto. Los compartimentos están unidos por una bisagra que permite cerrar uno sobre el otro y almacenar en su interior la pantalla divisoria y los cubos. La pantalla se coloca de manera vertical en el hueco entre los dos compartimentos cuando la caja está completamente abierta. Los cubos son bloques de madera de unos 2.5 cm de lado pintados de rojo, verde, azul o amarillo. El color no tiene una relevancia especial en esta prueba, pero sí la tiene en otras pruebas de evaluación cognitiva que se realizan con estos bloques.

- **Bastidor de soporte:** el bastidor (Figura 4.3) es la estructura que sostiene la cámara Kinect por encima de la caja para no entorpecer la visión completa de los cubos o los

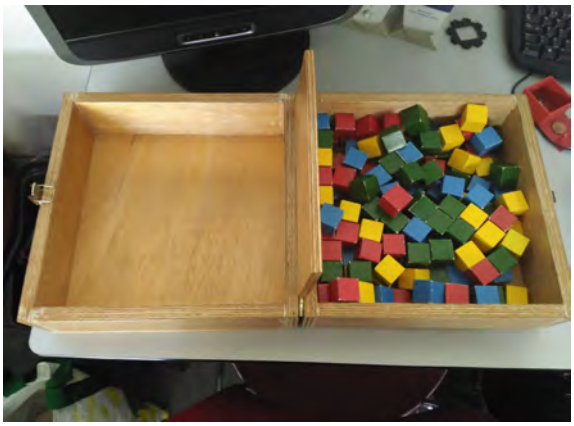


Figura 4.1: Kit para ensayo BBT preparado para la prueba.

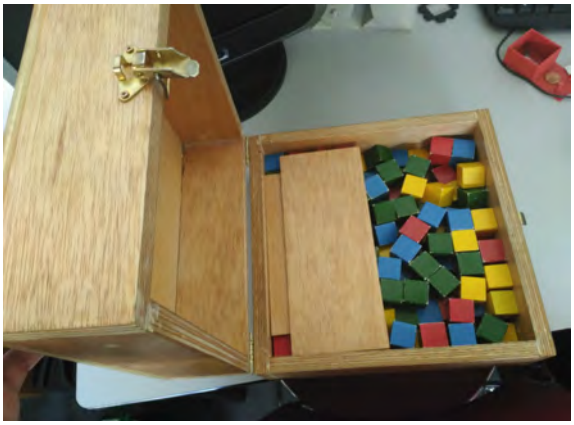


Figura 4.2: Kit desmontado para su almacenamiento.

movimientos del paciente. Está formado por 10 tubos de PVC de unos 65 cm de longitud y 32 mm de diámetro, unidos entre sí por codos y esquinas. La cámara se acopla al conjunto mediante una pieza fabricada por impresión 3D que se puede ver en la Figura 4.4.

Figura 4.3: Bastidor usado para sujetar la cámara sobre el paciente.

Figura 4.4: Detalle del acople de la cámara al bastidor.

- **Kinect V2:** el dispositivo empleado para la captura de imágenes. La segunda versión del popular invento de Microsoft cuenta con mejores características técnicas que su predecesora y ya está pensada para ser utilizada con otros dispositivos distintos de la XBOX; desde la compañía se proporciona software y guías de utilización para poder aprovechar todo su potencial con un ordenador en proyectos de investigación. La cámara cuenta con las

siguientes especificaciones [42]:

- **Dimensiones:** 24,9 cm x 6,6 cm x 6,7 cm
- **Imagen de color:** resolución de 1920 x 1080 px a 30Hz con buena iluminación.
- **Imagen de profundidad:** resolución de 512 x 424 px con un rango de los 50 cm a los 4.5 m y unos ángulos de apertura de 70 grados en horizontal y 60 en vertical.
- **Interfaz USB 3.0**
- **Matriz de 4 micrófonos**

- **MSI GT72 2QE:** el ordenador utilizado para la recepción y procesado de las imágenes. Es una máquina pensada originalmente para ejecutar sin esfuerzo juegos con requisitos muy altos, a nivel de procesado de datos y de procesado de vídeo. Sus características [43], que exceden los requerimientos de la cámara Kinect [44], son las siguientes:
  - **Procesador:** Intel Core i7-4720HQ (3.6GHz).
  - **RAM:** 16 GB DDR3.
  - **Almacenamiento:** Disco duro 1TB (7200 rpm) + 128 GB SSD.
  - **GPU:** nVidia Geforce GTX980M, 4GB GDDR5 compatible con DX 11.
  - **Puertos USB 3.0:** 6.
  - **Sistema Operativo:** Windows 10.

4.2. Software

- **Microsoft Software Development Kit para Kinect:** es un conjunto de librerías oficiales de Microsoft para extraer la información recogida por la cámara. La empresa decidió publicar sus propias librerías a raíz del hackeo que sufrió la Kinect V1 y las librerías libres (OpenNI, OpenKinect) que surgieron de esa investigación [45]. Las librerías oficiales incluyen códigos de ejemplo para cada una de las funcionalidades de la cámara: reconocimiento por voz, seguimiento de esqueletos y partes del cuerpo, lectura de imágenes de color, profundidad e infrarrojas... [23] Las librerías libres están construidas a partir de la ingeniería inversa realizada sobre las cámaras y, por tanto, no logran sacar todo el potencial

de las mismas. No obstante, son más flexibles en cuanto a los lenguajes de programación utilizados y a los entornos de desarrollo, disponibles, además, en varios sistemas operativos [46].

- **Microsoft Visual Studio 2015:** el entorno de desarrolo de aplicaciones de escritorio diseñado por Microsoft para crear contenido en Windows, iOS y Android. Dispone de gran cantidad de herramientas para facilitar la programación en varios lenguajes derivados de C y es totalmente compatible con las librerías oficiales de Kinect mencionadas anteriormente [47].

- **OpenCV 3.2:** librería de procesamiento de imágenes para Windows, Linux, MAC OS, iOS y Android. Soporta los lenguajes C, C++, Python y Java y está optimizada para sistemas multiprocesadores. La librería dispone de gran cantidad de funciones para el tratamiento de imágenes y algoritmos de detección que facilitan la implementación de sistemas con un alto rendimiento y, además, puede aprovecharse de la aceleración hardware de la plataforma en la que se esté utilizando. La librería cuenta con una amplia y activa comunidad de usuarios que elaboran tutoriales e intercambian conocimientos en diversos foros [48].

- **C++:** es un lenguaje de programación de alto nivel, creado en la década de 1980 por Bjarne Stroustrup, como una mejora del omnipresente lenguaje C. La principal diferencia con C es que se introdujo el paradigma de la Orientación a Objetos. De hecho, el primer compilador de C++ traducía primero el código a estructuras interpretables en C y luego compilaba todo como si se tratase de C, manteniendo de este modo todas las funcionalidades y optimización de C. El lenguaje ha ido evolucionando desde entonces y ha sufrido cambios y actualizaciones para adaptarse a estándares internacionales ISO [49]. El lenguaje C++ es compilado, fuertemente tipado, aunque soporta que los tipos de las variables sean declarados o inferidos según lo que vayan a almacenar, retrocompatible con C, con una enorme variedad de librerías y plantillas disponibles y multiparadigma, permitiendo no sólo la programación Orientada a Objetos, sino también la Procedimental o la General, fundamentalmente.

Capítulo 5

Análisis del problema y solución adoptada

Según lo visto anteriormente, el problema, ingenierilmente hablando, se reduce a satisfacer las siguientes premisas:

- Detectar el paso de la mano por encima de la barrera divisoria.
- Detectar y contabilizar el número de cubos transferidos.
- Limitar temporalmente la prueba a 60 segundos.
- Llevar un registro de la evolución del paciente.

Mientras que los dos últimos son simples problemas de programación, los dos primeros puntos requieren el uso de técnicas de detección y reconocimiento en tiempo real (o diferido en caso de usar vídeos pregrabados) de objetos pequeños. Con este fin se pueden plantear varias aproximaciones.

La detección de la mano puede hacerse múltiples maneras, pero no conviene que sea demasiado complejo para no ralentizar la detección de los cubos. La precisión de esta medida es, teóricamente, importante para el ensayo, pues, como se indica en las reglas del test, los cubos no serán válidos si la mano no traspasa la pantalla. Y, justo al contrario, si la mano traspasa la pantalla, pero el cubo cae fuera, debe ser contabilizado. Por tanto, los resultados de ambas

detecciones deben ser comparados para obtener una medida más fiable de los resultados del paciente.

Todos los métodos explicados en el capítulo de teoría podrían, de un modo u otro, cumplir con la detección de los cubos. No obstante, hay características que los hacen más o menos adecuados al trabajo. El algoritmo detector utilizado debe ser potente, pero no consumir demasiados recursos y, dado que los objetos a detectar tienen, aproximadamente, la misma forma, no es necesario que sean muy complejos ni que extraigan demasiadas características para poder reconocerlos. Un detector de bordes o de esquinas podría ser suficiente para localizar candidatos. Además, se debe solucionar la problemática de los cubos apilados a varias alturas y la mejor manera es recurrir a la imagen de profundidad de Kinect e incluirla en el detector.

Siguiendo estas premisas se han seleccionado varios algoritmos y técnicas de detección que se han considerado adecuadas para alcanzar los objetivos. Se han implementado dos versiones del sistema que poseen la misma estructura general, pero difieren en el proceso de detección. En los siguientes apartados se analizarán los algoritmos utilizados para facilitar su comprensión y comparativa. Una descripción exhaustiva del sistema, sus clases y métodos puede consultarse en el Apéndice A, al final del presente trabajo.

5.1. Programa principal

El programa principal, en el archivo *ProyectoBBT.cpp*, controla el flujo del proceso y hace las llamadas a las funciones necesarias para el correcto funcionamiento del sistema. El algoritmo implementado, que puede verse en la Figura 5.1 de manera gráfica, es el siguiente:

- Inicializar los sensores de color y profundidad.
- Cargar las plantillas, hacerles el tratamiento y almacenarlas para su uso (Figura 5.2).
- Pedir el nombre del paciente para registrar sus datos.
- Localizar los dos compartimentos y la pantalla divisoria (Figura 5.3).

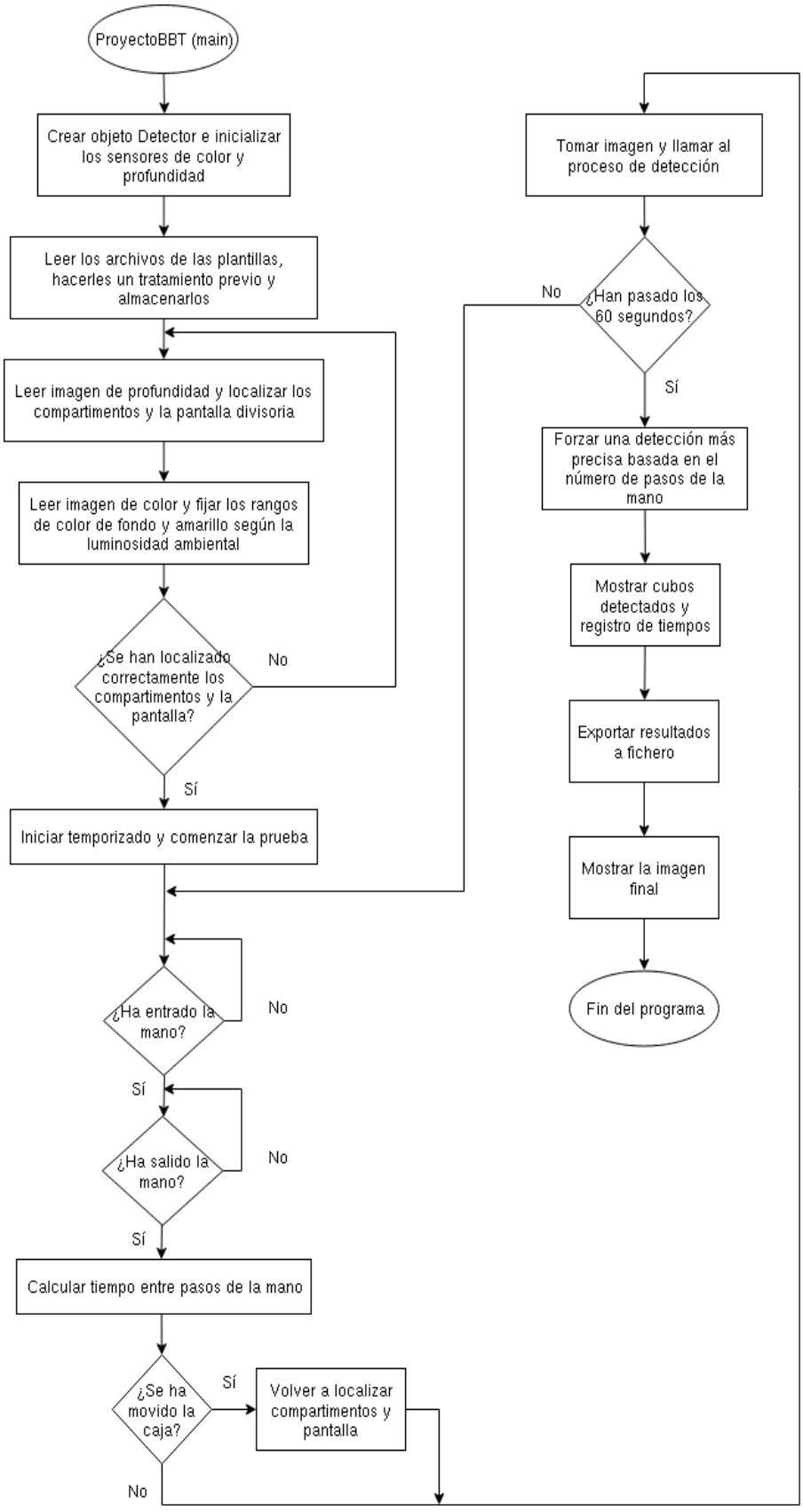


Figura 5.1: Diagrama de flujo del programa principal.

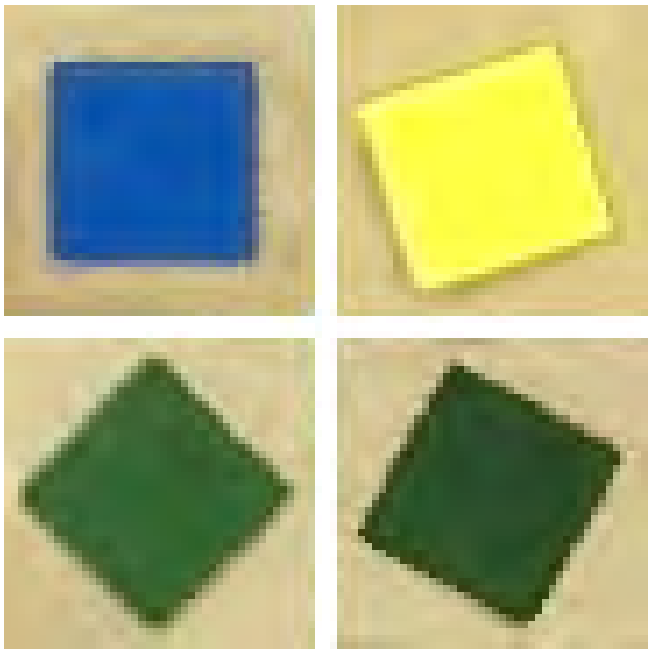


Figura 5.2: Plantillas utilizadas en el *Template matching*.

- Determinar qué compartimento está vacío para saber qué mano se va a poner a prueba (Figura 5.4).
- Ajustar valores HSV del fondo automáticamente para diferenciarlo del amarillo de los bloques.

Figura 5.3: Localización de la pantalla divisoria.

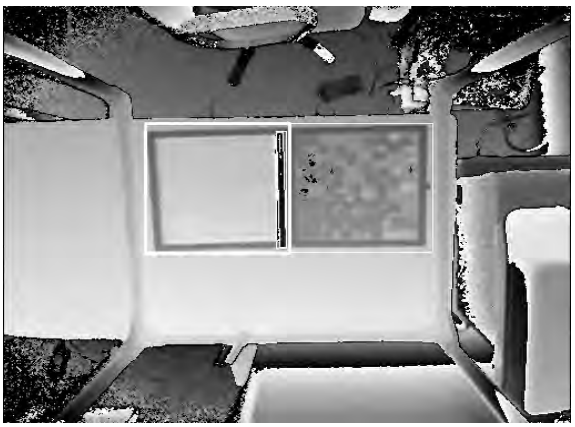


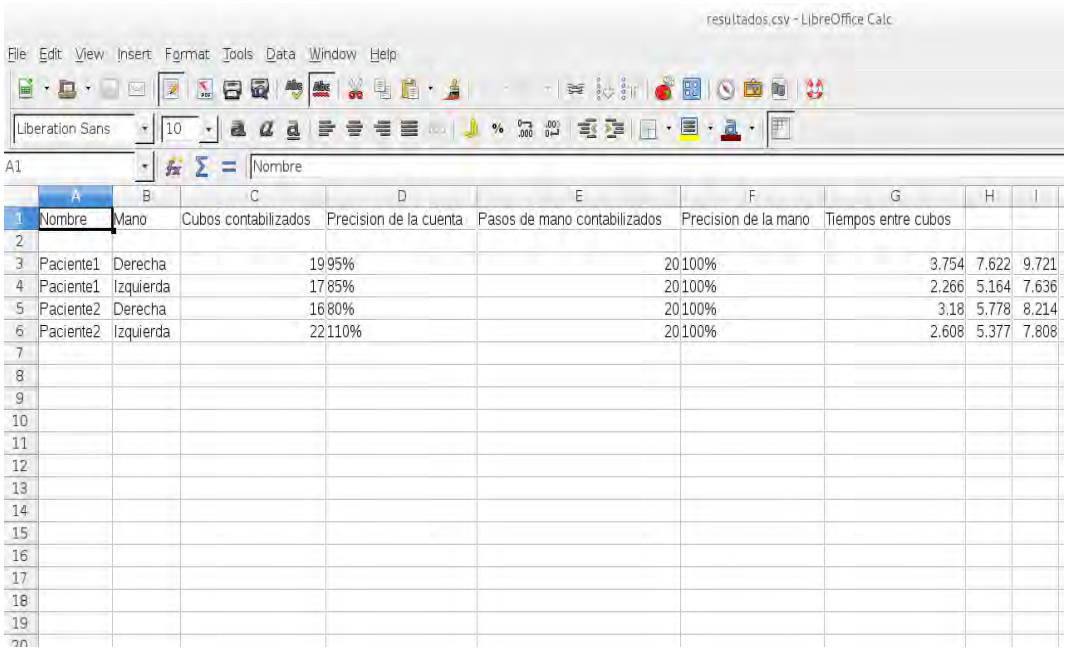
Figura 5.4: Localización de las cajas y el compartimento vacío.

- Pedir confirmación del usuario de que la pantalla y los compartimentos han sido correcta-

mente localizados. En caso contrario, repetir la búsqueda.

- Comenzar la prueba.
- Localizar el paso de la mano por encima de la barrera divisoria. Lo más sencillo es detectar cambios en la altura máxima de la imagen. Cuando la mano vuelva, capturar imagen del compartimento, de este modo, nos aseguramos de que la imagen esté estática. En ese momento, se almacena el tiempo que el paciente ha tardado entre dos movimientos de cubo.
- Usando este tiempo, podemos saber si el sistema ha fallado y se ha descontrolado debido a que el paciente ha desplazado la caja. Si esto sucede, el sistema automáticamente activa la relocalización de los compartimentos y la pantalla divisoria y continúa con el proceso.
- Se llama al proceso de detección correspondiente (difiere según la versión).
- La prueba continúa hasta que acabe el tiempo establecido.
- Si el número de detecciones y el registro de pasos de la mano difieren, se fuerza una nueva localización para asegurarnos de que el resultado final es el mejor posible.
- A continuación se muestra el número final de cubos localizados, el número de pasos de la mano y los tiempos que ha empleado el paciente entre cada movimiento.
- Estos datos, junto con el nombre del paciente y la mano con la que se ha realizado la prueba, se exportan a un fichero de datos (Figura 5.5) para su posterior análisis. Con estos datos pueden llevarse registros pormenorizados de la evolución del paciente en diferentes aspectos, por ejemplo, la cadencia con la que desplaza los bloques (Figura 5.6) o la mejora en sus marcas a lo largo de toda la rehabilitación.
- Finalmente se muestra la imagen con las localizaciones señaladas (Figura 5.7) y el programa termina.

En la versión utilizada para las pruebas, al final del proceso se pide al usuario contar manualmente los cubos desplazados y el sistema calcula y exporta automáticamente el porcentaje de acierto del detector.



	A	B	C	D	E	F	G	H	I
1	Nombre	Mano	Cubos contabilizados	Precision de la cuenta	Pasos de mano contabilizados	Precision de la mano	Tiempos entre cubos		
2									
3	Paciente1	Derecha	19	95%	20	100%	3.754	7.622	9.721
4	Paciente1	Izquierda	17	85%	20	100%	2.266	5.164	7.636
5	Paciente2	Derecha	16	80%	20	100%	3.18	5.778	8.214
6	Paciente2	Izquierda	22	110%	20	100%	2.608	5.377	7.808
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									

Figura 5.5: Captura de los datos exportados a un programa de cálculo.

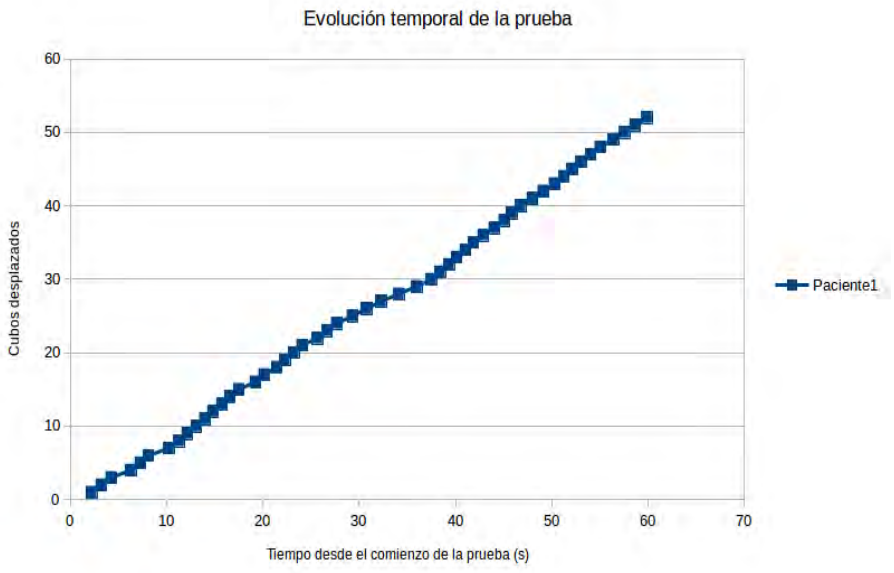


Figura 5.6: Registro de tiempos del paciente durante una prueba.

Además, para corregir problemas derivados de cambios en la iluminación ambiente, se pensó en aplicar un tratamiento previo a las imágenes para conseguir una mayor invarianza. El filtro propuesto es el *White-Patch*, explicado previamente en el Capítulo 3, que, aunque finalmente

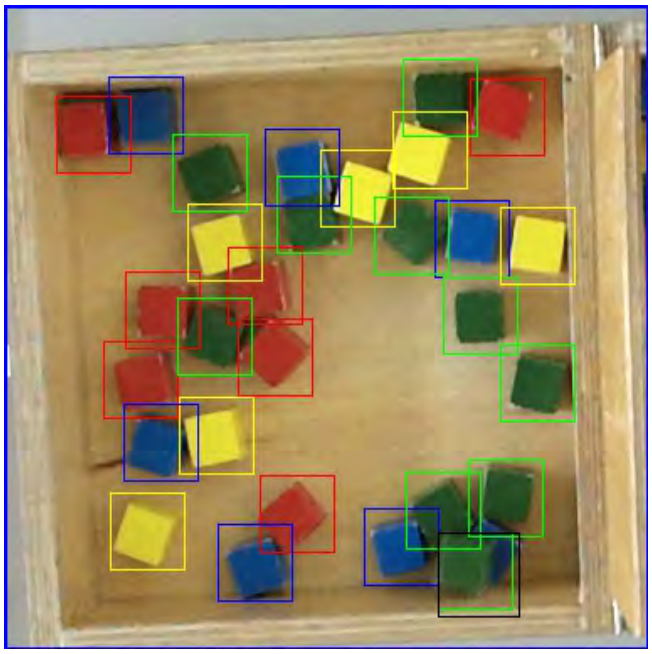


Figura 5.7: Resultado de la detección.

no ha sido utilizado, está implementado en el código y podría ser utilizado si las condiciones ambientales dificultan la detección.

5.2. Detección sin segmentación por color

La primera versión del algoritmo detector realiza el *template matching* directamente sobre la imagen de color obtenida, pasada primero a escala de grises y utiliza, una vez obtenidos los candidatos, los colores del centro de la ventana del candidato y cuatro píxeles adyacentes para por un lado hacer un primer descarte y, por otro, para clasificar los candidatos según su color. Su funcionamiento, explicado en profundidad, puede verse en la Figura 5.8 y se detalla a continuación.

- Redimensionamiento y colocación de los rectángulos que delimitan la región de interés en las dos imágenes(Figuras 5.9 y 5.10). Extraer las regiones de interés de las imágenes recogidas por la cámara (Figuras 5.11 y 5.12).
- La Región de Interés RGB leída es pasada a escala de grises y se le aplica un filtro bilateral

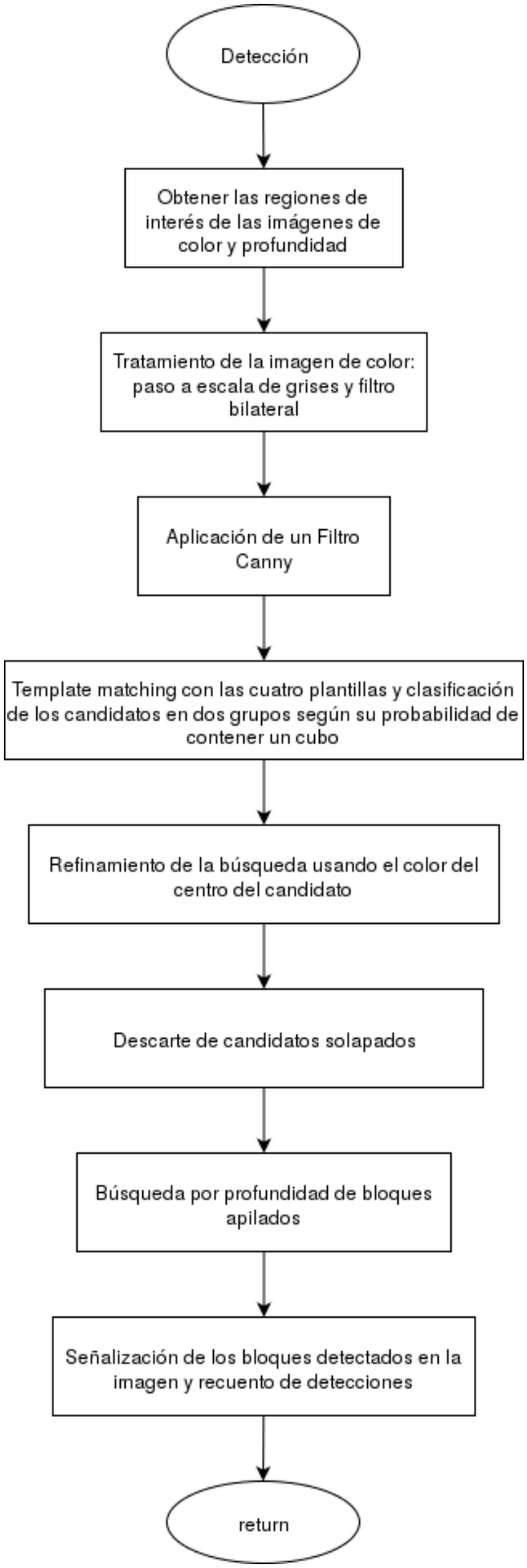


Figura 5.8: Diagrama de flujo del algoritmo de detección sin segmentación por color previa.

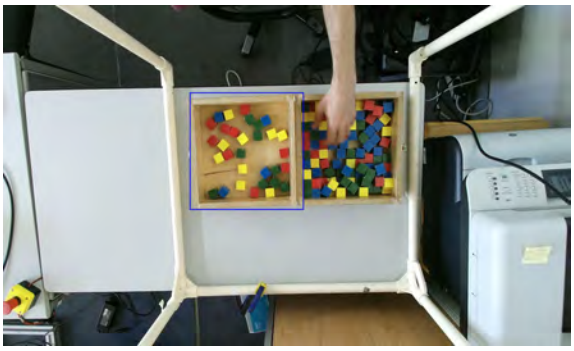


Figura 5.9: Localización del rectángulo contenedor en la imagen de color.

Figura 5.10: Localización del rectángulo contenedor en la imagen de profundidad.

Figura 5.11: Localización de la Región de Interés en la imagen de color.

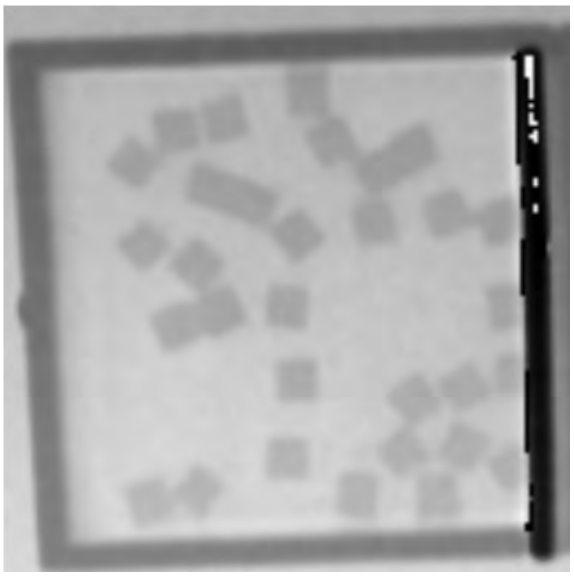


Figura 5.12: Localización de la Región de Interés en la imagen de profundidad.

(Figura 5.13) para difuminar imperfecciones sin perder definición en los bordes.

- La imagen difuminada es transformada entonces por filtro de tipo *Canny Edge Detector* para hallar el descenso del gradiente y así localizar los bordes (Figura 5.14).
- A continuación se realiza *template matching* con unas plantillas de cuadrados en varias posiciones de rotación y se hallan los puntos de diferencia mínima (Figura 5.15). Estos puntos se clasifican según su nivel de confianza en tres grupos: puntos con alta probabilidad

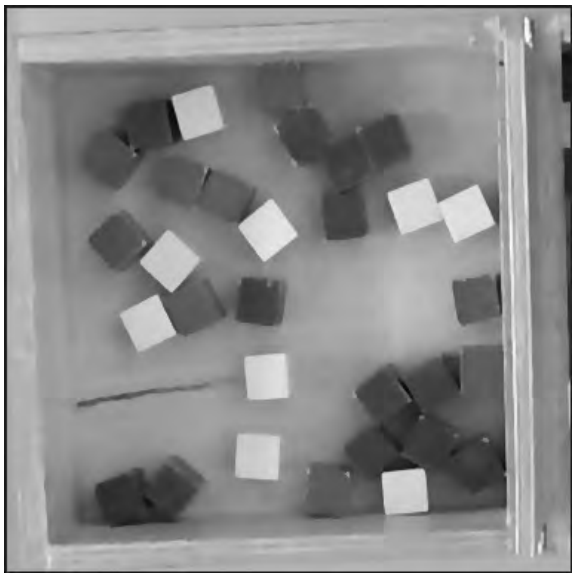


Figura 5.13: Paso a escala de grises y aplicación de filtro bilateral.

de contener cubos (la diferencia entre una plantilla y cierta región de la imagen es mínima), puntos con menor probabilidad y puntos que pueden ser descartados.

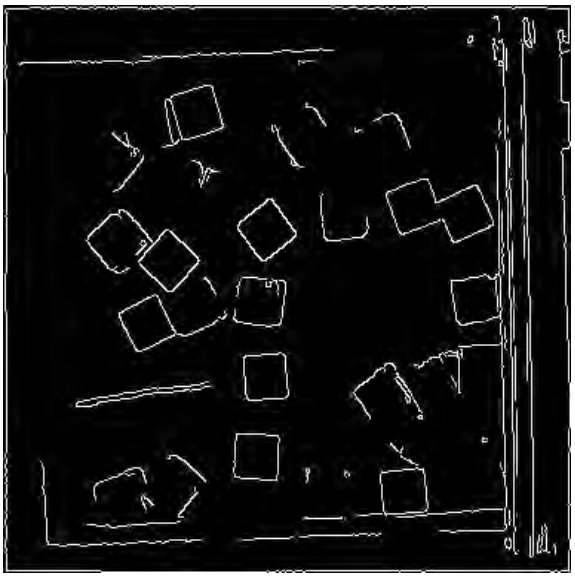


Figura 5.14: Resultado de la aplicación del filtro detector de bordes.

Figura 5.15: Resultados del *Template matching* para la plantilla de 60°.

- El primer paso de la clasificación de candidatos es la discriminación por color. No sólo sirve para clasificar los cubos en función de su color y posicionarlos en la imagen mediante

un rectángulo de su color que los contenga, sino que, además, se hace un descarte que diferencie el color del fondo de la caja de uno de los cuatro colores de los cubos, logrando un menor ratio de falsos positivos.

- Seguidamente, se recorren los dos grupos de puntos, comparándolos entre sí y consigo mismos, para descartar puntos que se encuentren muy próximos entre sí, de este modo, se eliminan localizaciones múltiples producidas por las distintas plantillas y candidatos solapados.

- El último paso, una vez se tiene ya un grupo de puntos con una alta probabilidad de ser bloques, es el algoritmo de búsqueda de bloques apilados. En este paso, se usa la imagen de profundidad para leer la altura a la que se encuentra el centro del hipotético bloque detectado. En función de la altura leída, se determina cuántos bloques se encuentran debajo del bloque estudiado y se añaden al grupo de bloques definitivos, etiquetándolos como bloques de color indeterminado (si se encuentran tapados por el bloque superior, es difícil conocer su color).

- Los bloques localizados se recuadran en la imagen original para su visualización por el usuario.

5.3. Detección con segmentación por color

El segundo algoritmo de detección propuesto tiene una etapa de segmentación por colores antes de hacer los *template matchings*. De esta manera, el filtro de detección de bordes se aplica sobre la imagen binaria resultado de la umbralización, con lo que, por un lado, los bordes tienen un contraste mayor que en una imagen en escala de grises y se detectan mejor, y por otro, al buscar candidatos, la imagen tiene muchas menos imperfecciones relacionadas con el fondo que puedan dar lugar a falsos positivos. El funcionamiento de este algoritmo puede verse en la Figura 5.16 y se explica en detalle a continuación.

- Igual que en el algoritmo sin segmentación, se comienza por el redimensionamiento y colocación de los rectángulos que delimitan la región de interés en las dos imágenes. Se extrae

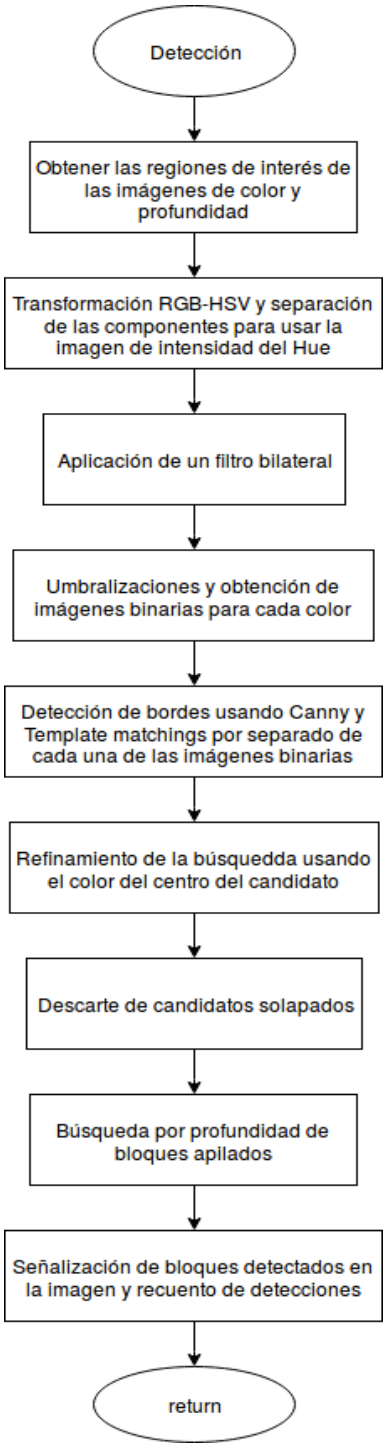


Figura 5.16: Diagrama de flujo del algoritmo de detección con segmentación por color previa.

la región de interés de la imagen recogida por la cámara a partir del rectángulo definido anteriormente.

- La región de interés RGB leída es pasada a HSV (Figura 5.17), se separan sus canales y se almacena el correspondiente al *Hue* (Figura 5.18). A la imagen en escala de grises resultante se le aplica un filtro bilateral para difuminarla.

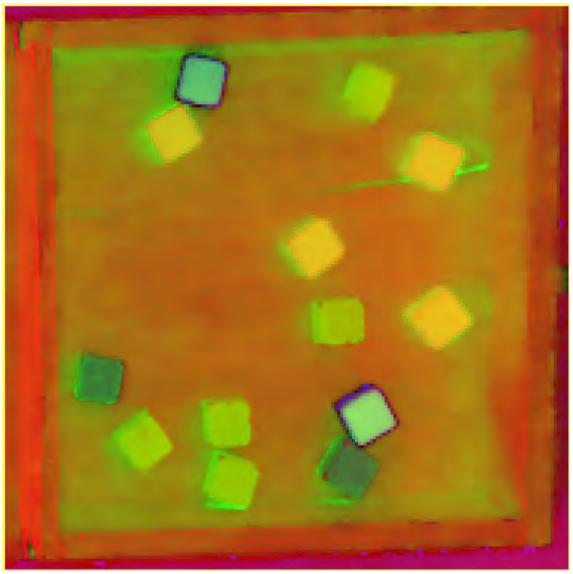


Figura 5.17: Transformación a HSV de la imagen RGB.

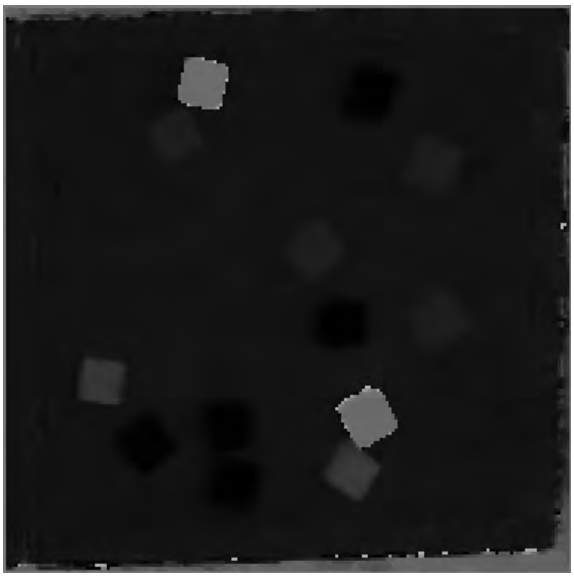


Figura 5.18: Extracción del plano H de la imagen anterior.

- Se crean cuatro imágenes binarias resultado de umbralizar la imagen anterior entre los rangos de *Hue* entre los que se encuentran los colores rojo, azul, verde y amarillo (Figura 5.19).
- Para cada imagen umbralizada se repite el siguiente proceso: extracción de bordes mediante filtro Canny (Figura 5.20), *template matching* con las cuatro plantillas y clasificación y refinación de los candidatos según el color del píxel central de la ventana que lo contiene y cuatro píxeles adyacentes.
- Se eliminan los candidatos solapados de la misma manera que en el primer algoritmo de detección para conseguir localizaciones únicas.
- En último lugar, se comprueba la profundidad que tienen los candidatos detectados y se

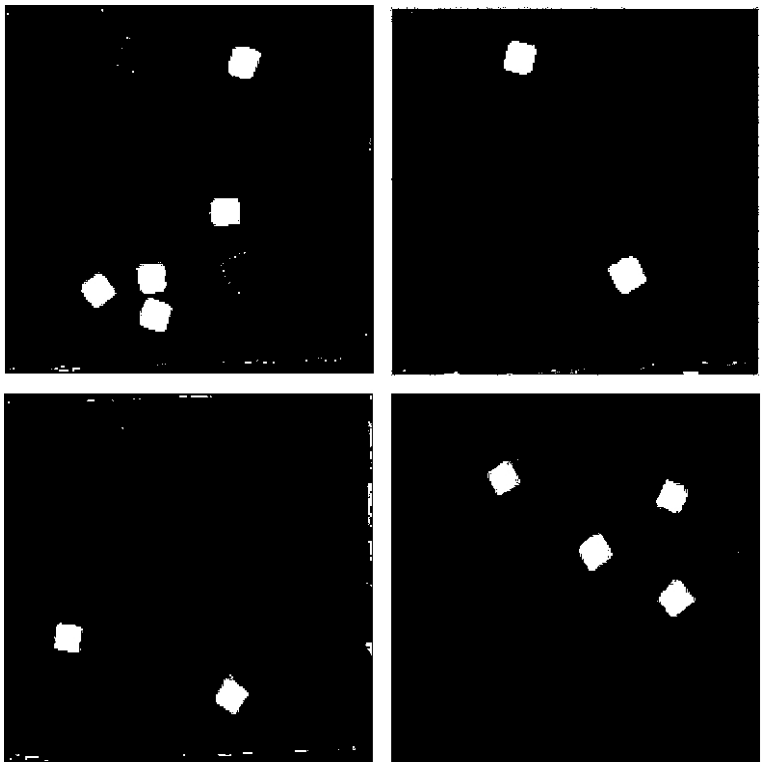


Figura 5.19: De izquierda a derecha y de arriba a abajo: umbrales para los colores rojo, azul, verde y amarillo.

añaden tantos cubos de un color indeterminado como veces la altura leída supere la altura de un cubo.

- Una vez obtenidos los candidatos que ya se consideran localizaciones exactas, se dibujan ventanas a su alrededor en la región de interés original para que el usuario compruebe la correcta localización de los cubos.

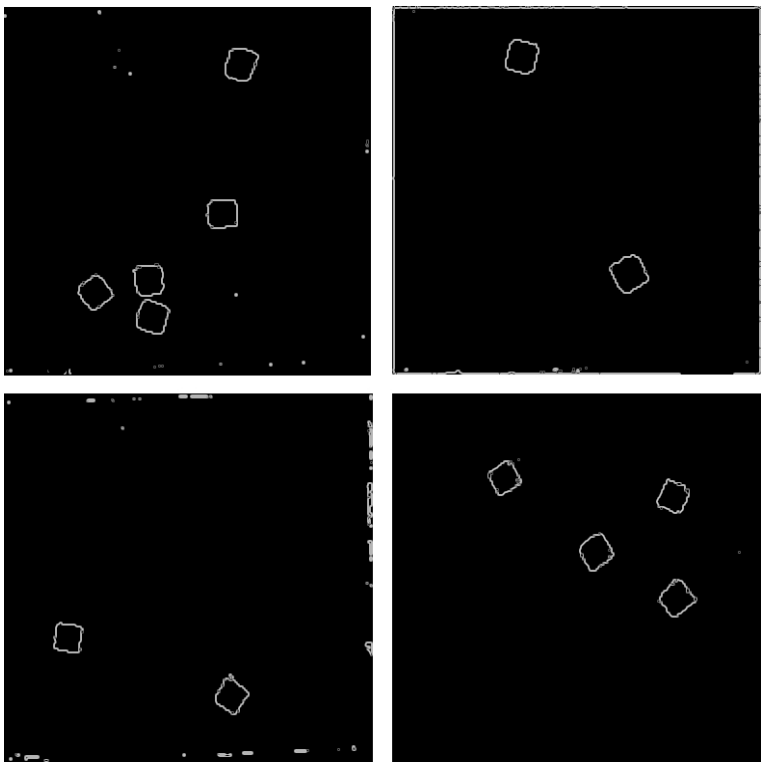


Figura 5.20: Detección de bordes para las imágenes de los umbrales obtenidas anteriormente.

Capítulo 6

Experimentos y resultados

En este capítulo se evaluará de forma estadística el rendimiento del sistema, comparando las dos versiones del algoritmo descritas anteriormente. Además, se expondrán los puntos fuertes y los defectos conocidos de su funcionamiento.

6.1. Comparativa de los resultados de conteo

Como se ha explicado, el sistema lleva la cuenta de dos aspectos: el número de cubos desplazados por el paciente y el número de veces que el paciente pasa la mano sobre la pantalla divisoria, independientemente del número de cubos que desplace cada vez. La razón de que se cuenten las dos cosas es lograr un resultado más fiable por medio de una comparación entre los dos números. A continuación se analizarán las estadísticas de los algoritmos en torno a estos dos aspectos.

6.1.1. Algoritmo sin segmentación por color

Para el primer algoritmo de detección se han realizado un total de 33 pruebas, desplazando distinto número de cubos para evaluar el desempeño del algoritmo en un rango amplio de posibilidades y, a la vez, averiguar cuál es el número máximo de cubos que se logran detectar con un

error aceptable.

En la Tabla 6.1 y la Figura 6.1 se pueden ver los resultados de estas pruebas.

Algoritmo sin segmentación por color			
Cubos desplazados	Acierto en conteo de bloques (%)	Acierto en conteo de pasos de mano (%)	Media de la detección(%)
10	100	100	100
15	91.67	100	95.83
20	92.5	100	96.25
25	78	99	88.5
30	79.17	94.17	86.67
35	77.86	91.43	84.64
40	70	92.5	81.25
45	75.56	89.44	82.5
50	78	90.5	84.25

Tabla 6.1: Resultados de las pruebas en la primera versión del algoritmo.

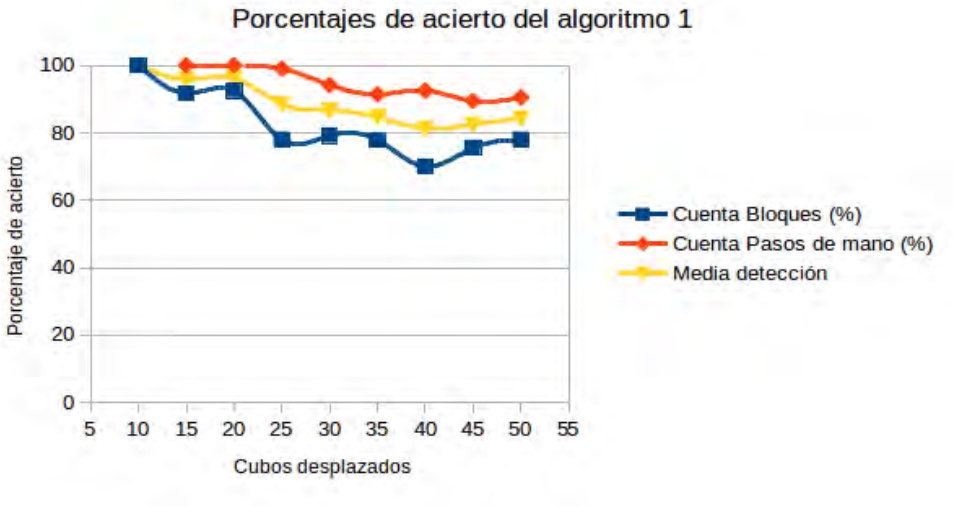


Figura 6.1: Resultados de las pruebas de la primera versión del algoritmo.

No obstante, estos datos sólo prueban la eficacia de la cuenta de cubos, no de la correcta detección de los mismos. Podría darse el caso, por ejemplo, de que se hayan contado exactamente los cubos desplazados, pero sus localizaciones o sus colores sean erróneos, con lo cual, el detector

no estaría funcionando adecuadamente. Por este motivo, es necesario realizar también medidas de evaluación de la detección. Por medio de matrices de confusión se calcula la sensibilidad y la precisión del sistema. La exactitud y la especificidad no pueden ser calculadas, pues en el momento en que se realizaron las pruebas el número de reales negativos (candidatos descartados por no contener bloques) no se midió. No obstante, la última revisión del código ha revelado que los candidatos generados en primera instancia superan en todo caso el millar, de los cuales puede quedar uno, en un caso extremo, tras los descartes sucesivos. La Tabla 6.2 muestra los resultados obtenidos de las matrices de confusión de las pruebas en cada rango de cubos y la Figura 6.2, algunas de las imágenes utilizadas para evaluar el desempeño del primer algoritmo de detección.

Cubos	Precisión	Sensibilidad
10	1	1
20	0.91	1
30	1	0.77
40	0.96	0.66
45	1	0.77
50	0.95	0.75

Tabla 6.2: Evaluación del primer detector

Conclusiones sobre el primer detector

A la vista de los datos expuestos se puede concluir que sistema que utiliza la primera versión del algoritmo detector arroja unos resultados aceptables, en torno al 90 % para 50 bloques, si tenemos en cuenta los datos de cuenta de la mano. No obstante, el rendimiento del detector no es el mejor. Su sensibilidad es pobre, lo que le lleva a dejar parte de los cubos sin detectar cuando se acumulan demasiados. Tiene, además, un límite de bloques acumulados a partir de los cuales su rendimiento cae en picado. Como se ha visto en la Figura 14 del Capítulo 5, la detección de bordes se vuelve confusa y compleja cuando los cubos se encuentran muy cerca entre sí.

Además, la cuenta del paso de la mano no puede ser tomada como un dato fiable para extrapolar el número de cubos desplazados por dos motivos: el paciente podría pasar la mano y mover varios o ningún cubo y si el paciente mueve la caja por error, hasta que el sistema de

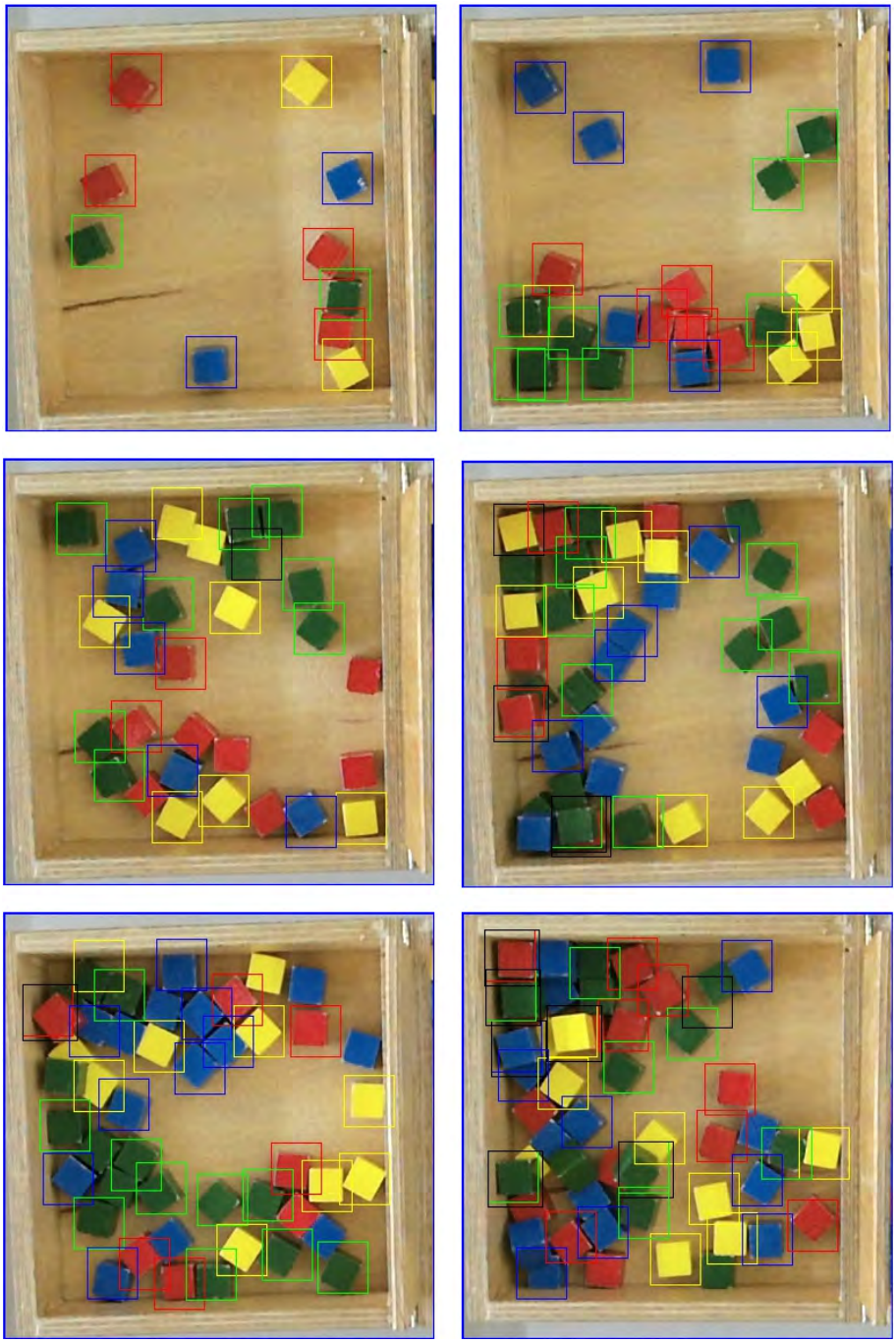


Figura 6.2: Resultados utilizados para la evaluación del detector sin segmentación por color.

reposicionamiento de las ROI actúa, se contabilizarían pasos de la mano erróneos. Por tanto, el valor de la cuenta de pasos de mano debe ser orientativo y no determinante.

6.1.2. Algoritmo con segmentación por color

Para el algoritmo con segmentación por color se han realizado un total de 38 pruebas desplazando diferentes cantidades de cubos para obtener datos estadísticos fiables y calcular el rango de validez del detector.

La Tabla 6.3 y la Figura 6.3 muestran los resultados obtenidos del análisis estadístico de dichas pruebas.

Algoritmo con segmentación por color			
Cubos desplazados	Acierto en conteo de bloques (%)	Acierto en conteo de pasos de mano (%)	Media de la detección (%)
10	103.33	100	101.67
20	91.89	92	91.95
30	100.67	84	92.33
40	101	86	93.5
50	103.2	74.4	88.8
60	98.61	65.28	81.94
65	101.54	60	80.77
70	104.29	48.57	76.43

Tabla 6.3: Resultados de las pruebas en la segunda versión del algoritmo.

Al igual que para el primer algoritmo, también en este caso se han evaluado las características del detector utilizando matrices de confusión. La Tabla 6.4 recoge los resultados medios de las medidas obtenidas a través de dichas matrices. Así mismo, algunos ejemplos de imágenes de resultados del detector con segmentación previa pueden verse en la Figura 6.4.

Conclusiones sobre el segundo detector

Con un simple examen de los datos expuestos se puede comprobar la evidente mejora en el detector que la segmentación por color previa a la búsqueda de candidatos ha provocado. El porcentaje de acierto en la detección ronda en 100 % incluso en cifras de bloques desplazados tan altas como los 70 bloques, una cantidad de bloques muy difícil de lograr, incluso para usuarios

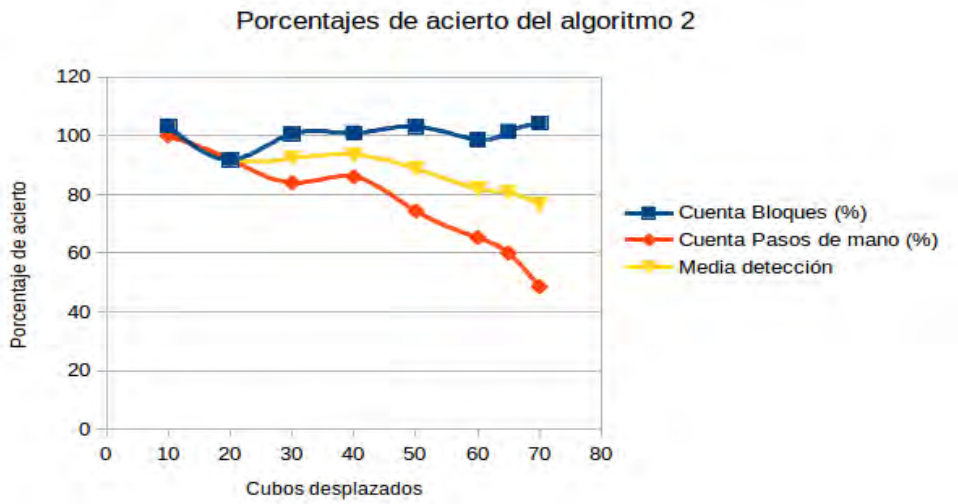


Figura 6.3: Resultados de las pruebas de la segunda versión del algoritmo.

Cubos	Precisión	Sensibilidad
10	1	1
20	1	0.95
30	0.97	0.97
40	1	1
50	0.9	0.98
60	0.93	0.91

Tabla 6.4: Evaluación del segundo detector.

sanos. Es cierto que, en algunos casos, se han detectado bloques de más y esto ha provocado porcentajes de detección superiores al 100 %, lo que parece contradecirse con la, también evidente, mejora en el desempeño del detector, la precisión y la sensibilidad también rondan el 95 %. Sin embargo, esto tiene una explicación sencilla. El aumento del número de detecciones Reales Positivas en detrimento de las Falsas Negativas ha aumentado el porcentaje de acierto a niveles cercanos al 100 % y, a pesar de que la precisión ha aumentado, se siguen detectando algunos bloques de más, lo que provoca que haya más detecciones que cubos y que ese porcentaje, por tanto, supere el 100 %.

Sin duda llamativo es el empeoramiento de la detección del paso de la mano. Esto se explica porque este algoritmo se ha hecho algo más lento para poder mejorar la detección. Debido a

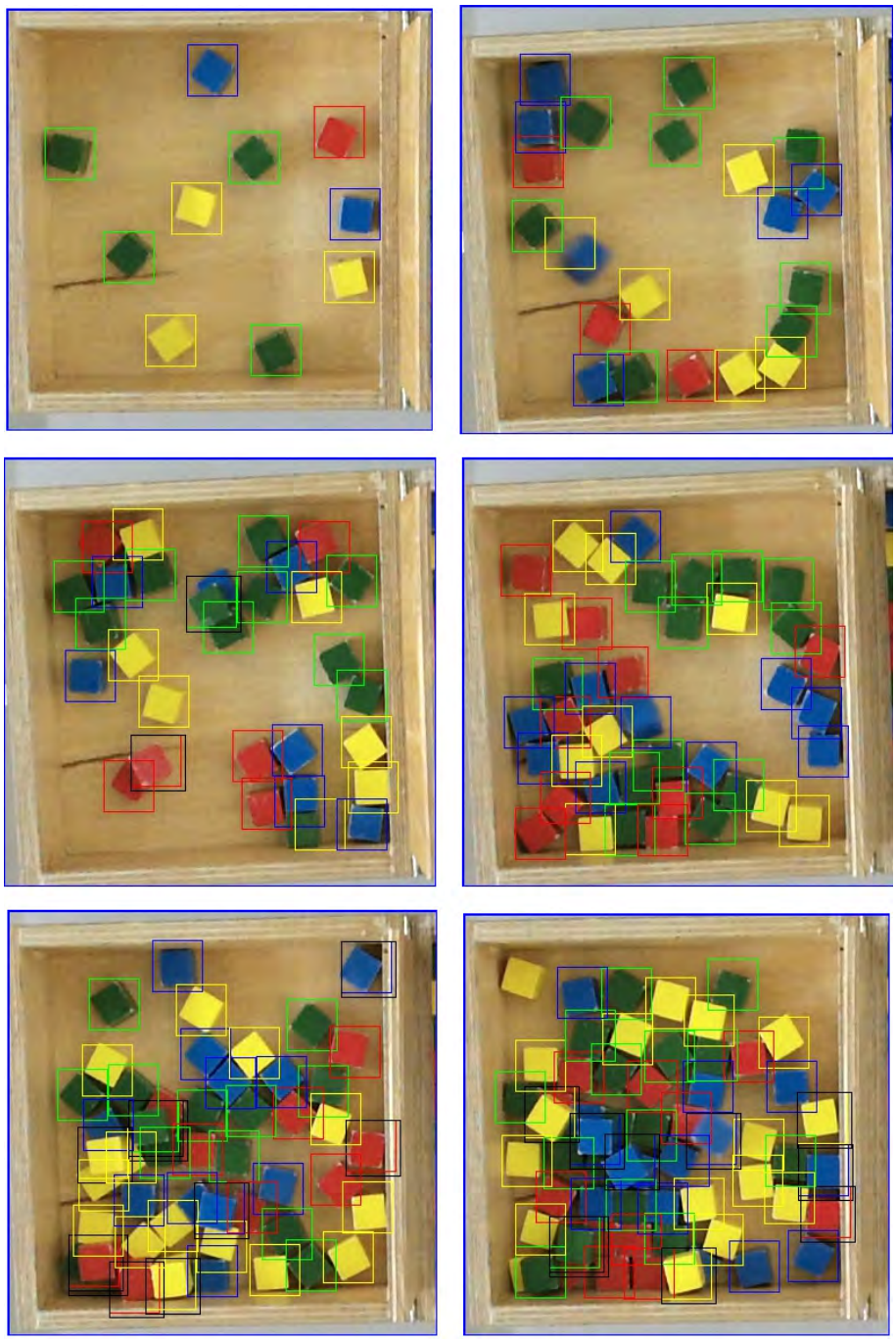


Figura 6.4: Resultados utilizados para la evaluación del detector con segmentación por color.

que tarda algo más en realizar las operaciones, la mano puede traspasar la pantalla mientras el ordenador sigue haciendo cálculos. Por otra parte, algunos de los fallos también podrían deberse a que, para conseguir mover un número alto de cubos desplazados, el usuario tiene que aumentar

la velocidad, con la consiguiente pérdida en precisión de movimientos. Esto podría provocar que la mano del usuario no traspasase completamente la pantalla y el sistema no detectase, por tanto, su paso. Un cubo trasladado de esta manera tampoco debería ser contabilizado según las normas del Ensayo de Caja y Cubos.

6.2. Evaluación del desempeño de los algoritmos en otros aspectos

En esta sección se analizará el funcionamiento de aspectos menos relevantes del sistema que, o bien presentan fallos, o bien han demostrado tener una gran fiabilidad.

6.2.1. Detección del color

La parte del algoritmo detector que clasifica los cubos según su color (el método *colorClassifier*) ha demostrado tener una gran tolerancia a fallos. Si bien es cierto que la luz parecía influir en cierta medida en la clasificación de bloques amarillos, ya que los rangos de valores de *Hue* para el color amarillo y el color marrón de la caja son muy parecidos y en ocasiones llega a haber un intervalo de solapamiento cuando la luz incide directamente sobre la caja, provocando que el fondo de la caja brille, lo que produce falsos positivos y reales negativos (fondo clasificado como bloques amarillos y bloques amarillos no clasificados). La casuística de este problema y las soluciones estudiadas y probadas se analizarán en la siguiente sección.

También se ha producido alguna clasificación errónea, anecdótica, eso sí, entre otros colores. La más usual es clasificar un bloque azul como amarillo, aunque también se ha dado alguna confusión entre el rojo y el verde.

6.2.2. Posicionamiento de Regiones de Interés

Sin duda uno de los puntos fuertes del algoritmo es la localización de las cajas y la pantalla divisoria y, a pesar de los problemas para trasladar coordenadas de la imagen de color a la imagen

de profundidad, solventados de una manera sencilla, pero eficaz.

El sistema es robusto, incluso en situaciones extremas, como puede verse en las Figuras 6.5 y 6.6 y el algoritmo para volver a posicionar las cajas si se desplazan actúa de manera rápida y apenas provoca distorsión en el resultado final.

Figura 6.5: Localización de la caja izquierda girada.

Figura 6.6: Localización de la caja derecha girada.

6.2.3. Velocidad de respuesta

El sistema que usa el algoritmo detector sin segmentación por color cumple con una calidad bastante aceptable con los requerimientos de velocidad de procesado. Sin embargo, si se usa el detector con segmentación previa, las detecciones del paso de la mano caen significativamente, por lo que se puede concluir que ese algoritmo es demasiado lento para cumplir con los dos objetivos: detectar los cubos y detectar la mano. El cuello de botella reside en una pausa de unos 250 milisegundos que se realiza antes de tomar las imágenes e iniciar el proceso de detección. La razón de esta espera es asegurar que la mano ha salido completamente de la caja y que la imagen ha quedado estática. Se ha comprobado empíricamente que, reduciendo esta pausa hasta los 70 milisegundos, el sistema reacciona correctamente a velocidades de transferencia de cubos mayores a 1 cubo por segundo, sin embargo, para pacientes lentos, esta pausa sería insuficiente. En el capítulo 7 se analizarán posibles soluciones para atajar este problema.

6.3. El algoritmo refinado

Tras analizar los datos anteriormente expuestos se pasó a una etapa de depuración de fallos en el código del segundo algoritmo, que ha demostrado ser más preciso en la detección, con el fin de hacer más robustas algunas funcionalidades que aún tenían margen de mejora.

Algunas de las correcciones menores que se han realizado incluyen ajustes de pausas y umbrales, evitar falsos positivos en los bordes de la caja y un pequeño ajuste en la interpolación de la imagen de profundidad para el compartimento derecho.

Como resultado, los tiempos de espera se han reducido a consecuencia de una mejora en la detección del paso de mano y se han eliminado falsos positivos relacionados con la detección del borde de la caja, sobre todo en las esquinas, así como relacionados con bloques apilados.

Sin embargo, los mayores esfuerzos se han concentrado en dotar al clasificador de la capacidad para diferenciar el fondo y los cubos amarillos cuando la luz artificial da de lleno sobre la caja.

Las Figuras 6.7 y 6.8 ejemplifican la diferencia en la calidad de la clasificación de cubos exclusivamente amarillos con y sin luz.

Figura 6.7: Reconocimiento con luz natural.

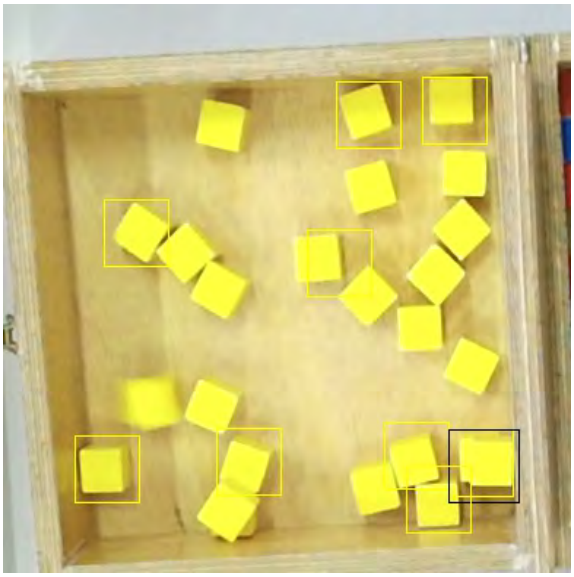


Figura 6.8: Reconocimiento con luz artificial.

Como puede verse, la clasificación con luz artificial es nefasta, mientras que con luz natural, se

logran unos resultados aceptables teniendo en cuenta el poco contraste existente al no haber más que cubos amarillos. Además, es notable como los cubos que sí se han clasificado correctamente se encuentran en su mayoría alejados del centro, en la zona donde las paredes hacen sombra.

El problema parece estar, por tanto, relacionado con la segmentación o la clasificación por colores, en cualquier caso debido al valor de *Hue* asociado al candidato.

Siguiendo este razonamiento se analizaron las imágenes binarias resultado de la segmentación. El problema se hace patente al comparar las imágenes de segmentación del color amarillo y de otro cualquiera (Figuras 6.9 y 6.10).

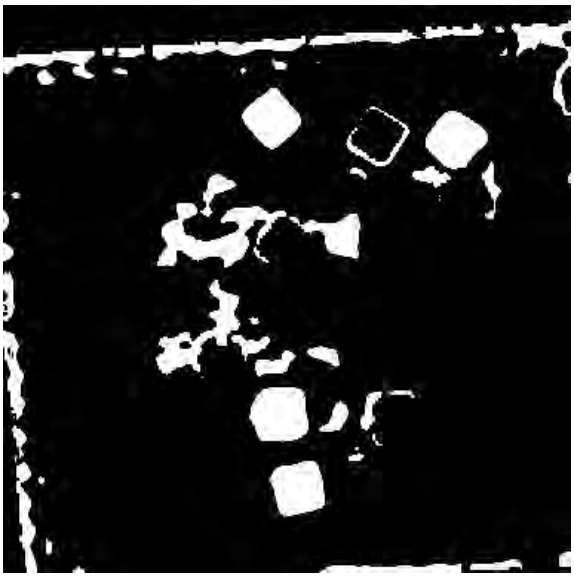


Figura 6.9: Segmentación del amarillo bajo la luz directa.

Figura 6.10: Segmentación del azul bajo la luz directa.

Mientras que los cubos azules aparecen muy bien definidos, con bordes rectos y paralelos, los cubos amarillos están redondeados, y no sólo eso, aparecen manchas alrededor del centro que deberían estar clasificadas como fondo y que pueden ser interpretadas como cubos o dificultar la detección de cubos amarillos.

Cabe, pues, preguntarse a qué se deben estas manchas. Un análisis de los valores de los píxeles del fondo revela que, debido a las condiciones lumínicas, el valor HSV ha cambiado en determinadas zonas y la segmentación falla. Es necesario, por tanto, desplazar la frontera umbral entre el fondo y el amarillo para asegurar la completa separación de los conjuntos.

En un primer momento se pensó en encontrar una frontera de separación válida para todas las condiciones del medio. No obstante, esta idea fue rápidamente rechazada; los rangos válidos se solapan y las fronteras no están lo suficientemente definidas para eso.

A continuación se probó con una frontera intermedia y una segunda variable que permitiera decidir en los casos cercanos a la frontera, de uno y de otro lado. A pesar de que, teóricamente, el valor de *Saturación* del píxel debería ser suficiente para diferenciar, en la práctica, este método tampoco dió los resultados esperados.

La siguiente propuesta consistió en modificar la imagen adquirida para eliminar los efectos del cambio de luz. Esto se llevó a cabo mediante la implementación del filtro *White-Patch*. Se observó cierta mejoría en la detección de amarillos, pero no fue un cambio significativo y se descartó finalmente.

El cuarto método propuesto conllevaba una transformación al espacio de colores CIE L*a*b, que trabaja con el rojo, azul, verde y amarillo como colores base y obtiene el resto a partir de ellos. Las pruebas arrojaron resultados similares a los del HSV, debido a que el marrón del fondo seguía teniendo una fuerte componente de amarillo.

Finalmente, se trató de cambiar la perspectiva de enfoque. En lugar de usar una única frontera, el sistema discerniría primero las condiciones lumínicas y establecería una frontera diferente en función del resultado. La Figura 6.11 ilustra el problema de las fronteras.

Finalmente, el algoritmo depurado y que incorporaba la detección automática de la condición lumínica se probó bajo condiciones de alta luminosidad, obteniéndose una disminución significativa del número de falsos positivos y cierta mejora en la clasificación de cubos amarillos. Si bien es cierto que la totalidad de los falsos negativos se debe a cubos amarillos, como se puede ver en la Figura 6.12, que ilustra algunos de los resultados obtenidos en la experimentación.

La Tabla 6.5 y la Figura 6.13 asociada a ella reflejan los resultados numéricos de las pruebas.

El uso de matrices de confusión para analizar en profundidad el desempeño del algoritmo, no tiene tanto sentido en este caso, dado que existe una diferencia sustancial entre una de las clases clasificadas y las tres restantes, lo que perjudicaría al resultado de la matriz y no sería

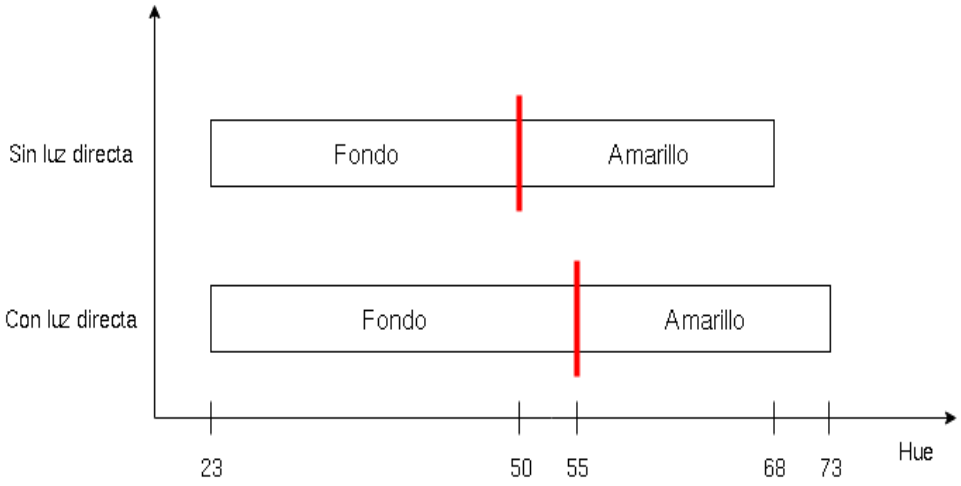


Figura 6.11: Fronteras aproximadas entre el fondo y el amarillo para distintas condiciones de iluminación.

Algoritmo refinado			
Cubos desplazados	Acierto en conteo de bloques (%)	Acierto en conteo de pasos de mano (%)	Media de la detección (%)
10	100	100	100
20	81.7	98.3	90
30	91.7	98.4	95.1
40	77.5	82.5	80
50	87.2	76.6	81.9
60	79.6	60.2	69.9
65	81.5	46.1	63.8

Tabla 6.5: Resultados de las pruebas del algoritmo refinado con luz directa.

del todo realista. En lugar de eso, se han analizado las imágenes resultado de las pruebas y se ha hallado el porcentaje de reales positivos frente al total de la clase. Los cubos rojos, verdes y azules han obtenido porcentajes cercanos todos ellos al 100 %, mientras que los cubos amarillos se encuentran en torno al 36 % de clasificación, lo que, sumado a la reducción de falsos positivos amarillos, supone una mejora respecto a la clasificación de cubos amarillos del segundo algoritmo bajo las mismas condiciones de luz, que se encontraba en torno al 25 %.

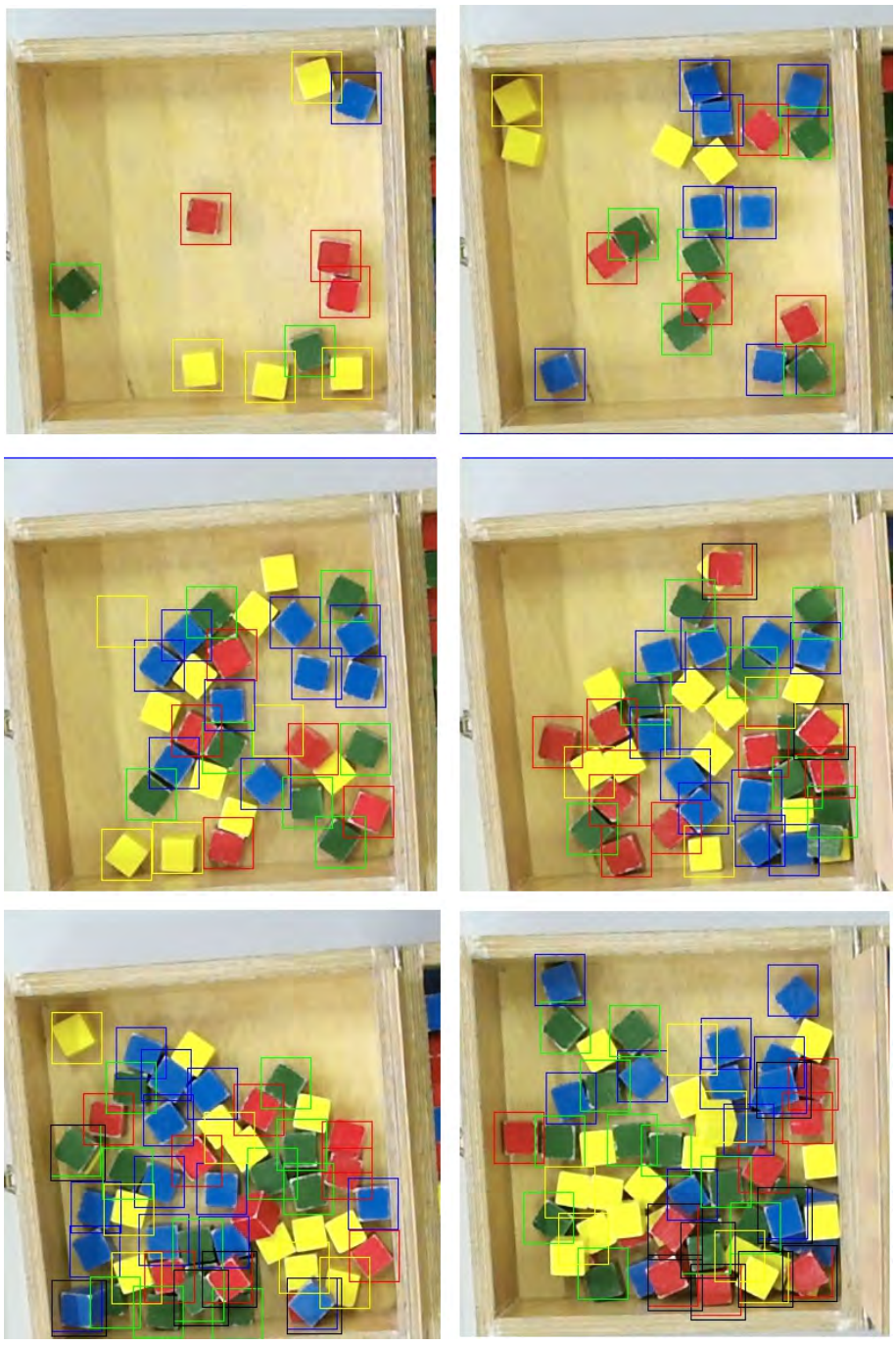


Figura 6.12: Resultados utilizados para la evaluación del detector refinado.

6.3.1. Conclusiones sobre el detector refinado

La versión final del algoritmo ha logrado mejorar los resultados del detector bajo luz directa, eliminando casi por completo los falsos positivos y mejorando sensiblemente el ratio de deteccio-

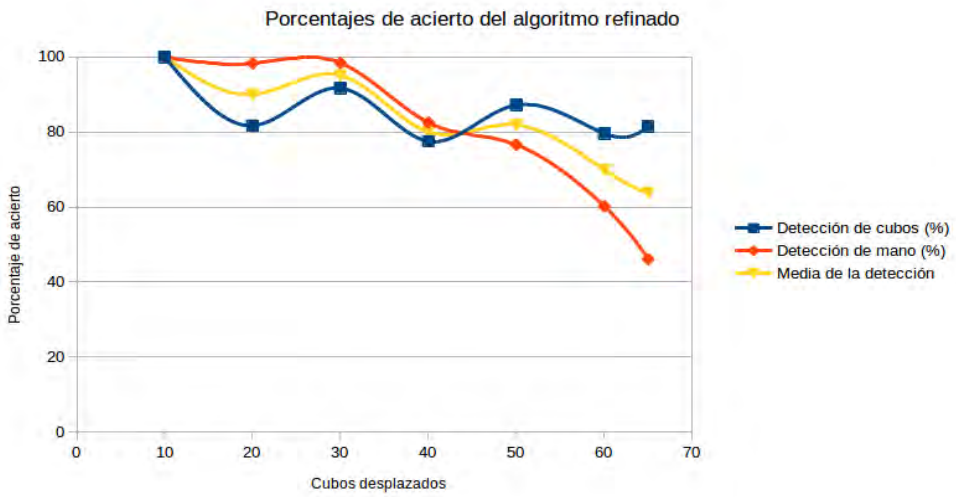


Figura 6.13: Resultados de las pruebas del algoritmo refinado con luz directa.

nes de cubos amarillos en estas condiciones. Todo esto, por supuesto, sin empeorar, de hecho lo mejora en ciertos aspectos, el desempeño del detector bajo luz natural.

La caída del porcentaje de detección de la mano cuando aumenta la velocidad de transferencia de cubos sigue siendo significativa, debido a que el algoritmo refinado basa su funcionamiento en el segundo algoritmo detector, algo más pesado que el primero.

6.4. Sobre las pruebas realizadas

Los datos anteriormente expuestos han sido obtenidos mediante pruebas en laboratorio lo más realistas posibles, pero con un usuario sin impedimentos físicos. A fecha de la escritura de este documento, el sistema no ha sido probado en un entorno completamente realista, pero sí se está estudiando su viabilidad para hacer pruebas en un hospital.

Capítulo 7

Conclusiones y líneas futuras

El objetivo de este capítulo es unificar todo lo expuesto anteriormente en cuanto al desempeño del sistema y proponer mejoras e implementaciones futuras que puedan ampliar la funcionalidad y la utilidad de este proyecto.

El trabajo realizado en este proyecto ha servido para crear una aplicación que cumple con los objetivos propuestos:

- Detectar el paso de la mano: el algoritmo que detecta la mano es muy eficaz y con unas mejoras en la velocidad de detección se podría obtener un mejor rendimiento de él.
- Detección y cuenta de los cubos: el algoritmo detector logra tasas de detección casi perfectas bajo unas condiciones de luz determinadas y es capaz, además, de contabilizar cubos apilados con bastante precisión.
- Limitar temporalmente la prueba: se ha llevado una temporización de la prueba que ha servido no sólo para limitarla, sino también para calcular la velocidad de movimiento del paciente y para activar la función de recolocación de las cajas.
- Registrar la evolución del paciente: los datos resultado de la prueba se han almacenado en un fichero de texto que puede ser exportado a un programa de cálculo para visualizarlos gráficamente. El médico puede acceder a la evolución a lo largo de varias pruebas de un paciente y también a la evolución de sus movimientos en pruebas individuales.

En cuanto a las herramientas utilizadas, se puede concluir que son todas tecnologías muy potentes, de enorme versatilidad y aplicación en el campo de la robótica y la visión artificial y que, gracias a estas características, se ha logrado simplificar enormemente un problema de difícil solución. Lo que para un humano es una tarea no demasiado compleja, pero sí tediosa, se puede hacer en apenas milisegundos mediante este sistema con una fiabilidad bastante alta.

El algoritmo refinado, que basa su funcionamiento en el segundo algoritmo y que ha sido el que mejores resultados ha obtenido en todas las condiciones, tiene aún fallos que corregir y existen funcionalidades que se podrían añadir. Se proponen, pues, algunas posibles mejoras para el sistema, que lo hagan más sencillo, útil y preciso:

- Implementar un sistema que modifique el tiempo de espera antes de adquirir la imagen después de detectar la salida de la mano dinámicamente según la velocidad de movimiento del paciente. O, alternativamente:
- Implementar dos algoritmos detectores. Usar el primero, sin segmentación por color, durante la prueba para dar resultados preliminares y el segundo, más preciso y lento, una vez haya finalizado para dar mejores resultados.
- Encontrar una solución, si no software, al menos hardware, como puede ser pintar la caja de un color mate más oscuro, para mejorar la detección de cubos bajo luz directa.
- Implementar la interpolación entre imágenes de color y profundidad mediante las herramientas proporcionadas por Microsoft.
- Crear una interfaz de usuario gráfica y más intuitiva.
- Ampliar los datos recogidos y exportados al fichero con información adicional acerca de los cubos y de los movimientos del paciente.

APÉNDICES

APÉNDICE A

Diagramas de clases y métodos

En este capítulo se detallarán las clases utilizadas, la forma en que se estructuran y se relacionan entre sí y se explicará el funcionamiento de algunos de los métodos más importantes para el proceso de detección.

A.1. Diagrama de clases del proyecto

Utilizando el paradigma de programación modular y estructurada y un lenguaje orientado a objetos, el programa se compone de un fichero que controla el flujo del sistema y desde el que se instancia un objeto de la clase *Detector*. Este objeto contiene todos los atributos y métodos necesarios y, además, está relacionado mediante agregación y composición con las otras tres clases definidas. En la Figura A.1 puede verse un diagrama de las relaciones entre clases.

En las siguientes secciones se analizarán en profundidad las clases.

A.2. Clase *CColorBasics*

Esta clase funciona como interfaz entre el hardware de lectura de imágenes en color y el sistema detector de cubos. Su código se basa en los ejemplos que el SDK de Microsoft trae para

probar las funcionalidades de la cámara y utiliza, por tanto, las librerías de Microsoft para la inicialización y captura de imágenes. En la Figura A.2 se muestra la composición de esta clase.

Además del constructor y el destructor, que se encargan, respectivamente, de inicializar los atributos del objeto y de liberar la conexión con el sensor, los dos métodos importantes de la clase se explican a continuación. La función de los *getters* es autoexplicativa y carece de relevancia, por lo que no serán detallados.

A.2.1. Método *initializeDefaultSensor*

Este método se llama al inicio del programa y se encarga de inicializar y preparar el sensor de color para que envíe imágenes.

El método comienza buscando un sensor de color por defecto, en caso de que hubiese varias Kinect conectadas, inicializaría sólo una. Cuando lo ha encontrado, lo abre y obtiene de él una «Fuente de fotogramas de color» (*Color Frame Source*). De este objeto se extrae finalmente el lector de imágenes que se utilizará cada vez que ordenemos tomar una fotografía. El *Frame Reader* se almacena en uno de los atributos del objeto y la *Color Frame Source* se destruye.

A.2.2. Método *update*

El método *update* se vale del lector de fotogramas almacenado en la inicialización. Este lector de fotogramas contiene la función que permite leer una imagen del sensor inicializado. Una vez leída, se accede consecutivamente a las características de la imagen para comprobar que se ha

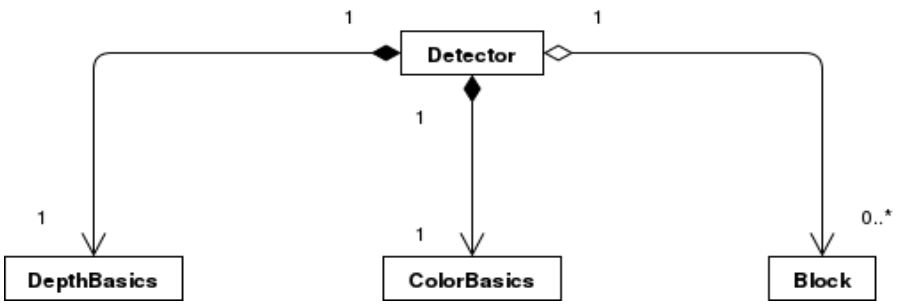


Figura A.1: Diagrama de clases.

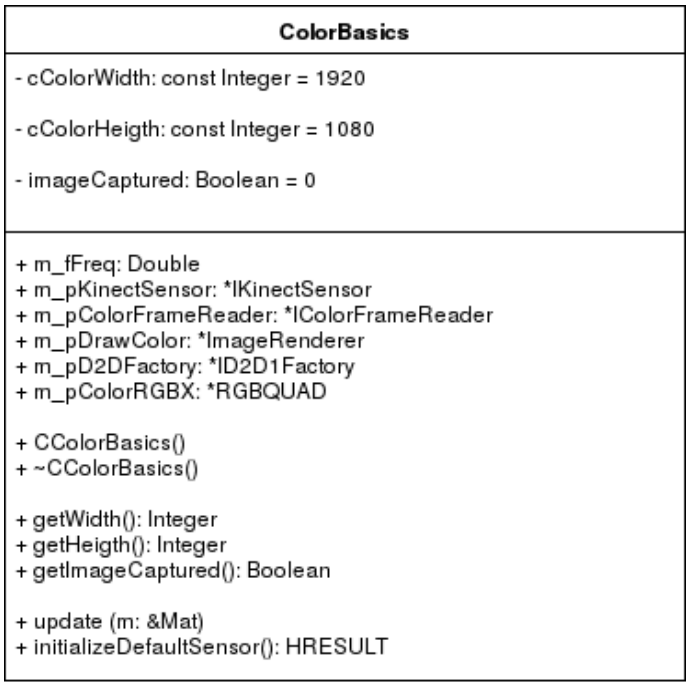


Figura A.2: Clase CColorBasics.

leído correctamente, en cuyo caso la imagen se almacena en una imagen del tipo *Mat* de 8 bits y 4 canales, las utilizadas por la librería OpenCV. Como, además, OpenCV utiliza por defecto la codificación BGRA en lugar de RGBA, la imagen debe ser transformada convenientemente. En ese momento, se activa el flag *imageCaptured* para avisar del éxito de la captura. Esto es necesario porque la cámara está limitada a una frecuencia de muestreo de 30Hz, pero el programa se ejecuta mucho más rápido, por lo que el proceso de captura debe ser repetido hasta que la cámara esté lista para muestrear una imagen.

A.3. Clase *CDepthBasics*

Esta clase es análoga a la descrita en el apartado anterior, con la única diferencia de que funciona como interfaz entre el software detector y el sensor de profundidad. De la misma manera que la clase *CColorBasics*, es una modificación del código de ejemplo proporcionado por Microsoft en el SDK. A continuación, en la Figura A.3, puede consultarse la estructura de la clase.

Además del constructor y el destructor, que se encargan, respectivamente, de inicializar los

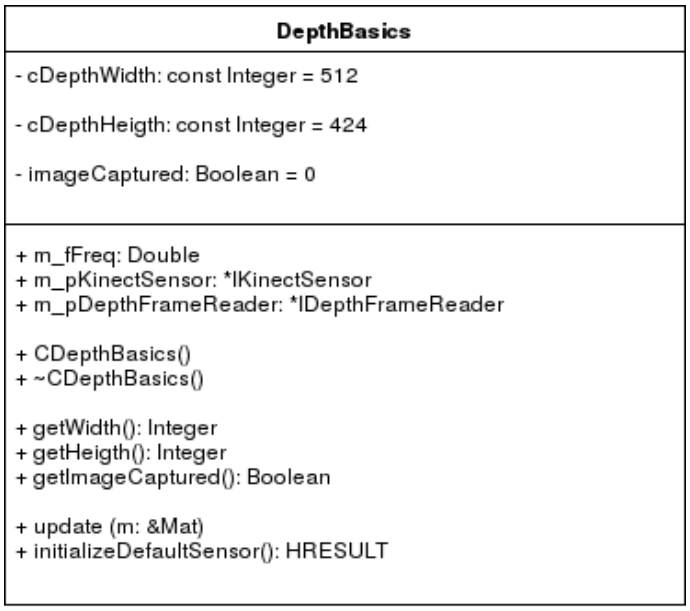


Figura A.3: Clase CDepthBasics.

atributos del objeto y de liberar la conexión con el sensor, los dos métodos importantes de la clase se explican a continuación. La función de los *getters* es autoexplicativa y carece de relevancia, por lo que no serán detallados.

A.3.1. Método *initializeDefaultSensor*

Este método se llama al inicio del programa y se encarga de inicializar y preparar el sensor de profundidad para que envíe imágenes. El funcionamiento de la inicialización es exactamente igual al descrito para el sensor de color y no precisa, por tanto, más explicación.

A.3.2. Método *update*

El método de obtención de imágenes de profundidad es muy similar a su homólogo en imágenes de color; se vale del lector de fotogramas almacenado en la inicialización para leer una imagen del sensor de profundidad. Una vez leída, se accede consecutivamente a las características de la imagen para comprobar que se ha leído correctamente y se hace un filtro para descartar aquellas zonas de la imagen cuya medida de profundidad, por estar demasiado cerca o demasiado lejos,

no es fiable. La imagen obtenida por defecto por el sensor tiene una resolución de 16 bits y las medidas que obtiene están en milímetros. No obstante, al mostrar esta imagen en 16 bits tanta resolución no permite apreciar correctamente los cubos y las cajas por tener una altura similar. Para paliar este efecto y en pos de una mejor comprensión del sistema, la imagen recibida por el sensor es modificada para tener una resolución de 8 bits, descartando los 8 bits más significativos. Esto, por supuesto tiene consecuencias, como que las distancias medidas ya no son reales, pero no afectan al programa. Los píxeles transformados se copian en una imagen de tipo *Mat* de 8 bits y un canal y se activa el flag *imageCaptured* para avisar del éxito de la captura. Esto es necesario porque la cámara está limitada a una frecuencia de muestreo de 30Hz, pero el programa se ejecuta mucho más rápido, por lo que el proceso de captura debe ser repetido hasta que la cámara esté lista para muestrear una imagen.

A.4. Clase *Block*

Los objetos de la clase *Block* representan un cubo de colores. Para cada bloque detectado se crea un objeto de esta clase que contiene en sus atributos toda la información importante relativa al cubo. Cada objeto instanciado contiene un atributo con la posición de la esquina superior izquierda de la ventana que lo contiene, un número que codifica su color y el tamaño de la ventana que lo contiene. Esto último es importante porque las ventanas de los cubos que son añadidos por profundidad son más grandes cuanto más arriba se encuentre el cubo, de esta manera se pueden representar todos los cubos detectados sin sobreescribir las ventanas previamente dibujadas.

El único aspecto reseñable de los métodos de esta clase es el constructor sobrecargado que permite inicializar un cubo con su posición y color de manera directa.

En la Figura A.4 se muestra el esquema de la clase.

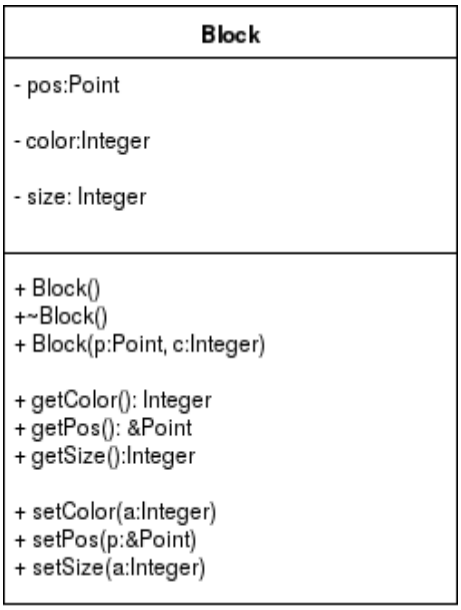


Figura A.4: Clase Block.

A.5. Clase *Detector*

La clase *Detector* controla el proceso de detección y contiene todos los elementos necesarios para llevar a cabo el objetivo. Es, por tanto, la más importante, compleja y extensa de las cuatro clases presentes en el proyecto, como puede verse en la Figura A.5.

A continuación se explicarán los métodos y atributos más relevantes de los que se compone.

En los apartados a continuación se detalla el funcionamiento de los algoritmos que intervienen en la detección. Para facilitar la comprensión del algoritmo, se agruparán estos métodos según la tarea que desempeñan.

A.5.1. Inicialización

- *initializeSensors*: este método se encarga de inicializar los sensores de color y profundidad llamando a los métodos de inicialización correspondientes de los objetos de tipo CColorBasics y CDepthBasics incluidos como atributos en la clase *Detector* (**color** y **depth**). Se comprueba, además, si la inicialización ha sido correcta antes de continuar.

Figura A.5: Clase Detector.

- *setZero*, *setThirty*, *setFortyFive* y *setSixty*: estas funciones como parámetro una plantilla, una imagen cuadrada de 44 píxeles de lado que contiene un bloque en una de las cuatro posiciones fundamentales en las que se pueden encontrar: formando $0, 30, 45$ o 60 grados con la horizontal (Figura A.6). La imagen recibida es pasada a escala de grises, difuminada con filtro bilateral y aplicada un filtro de tipo *Canny Edge Detector* (Figura A.7). El resultado se almacena en la imagen de tipo *Mat* correspondiente contenida en el objeto *detector*: **zero**, **thirty**, **fortyFive** o **sixty**.

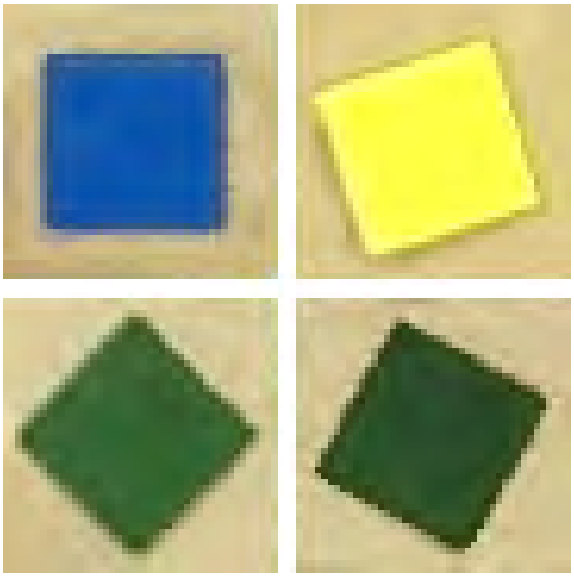


Figura A.6: Plantillas utilizadas en el *Template matching*.

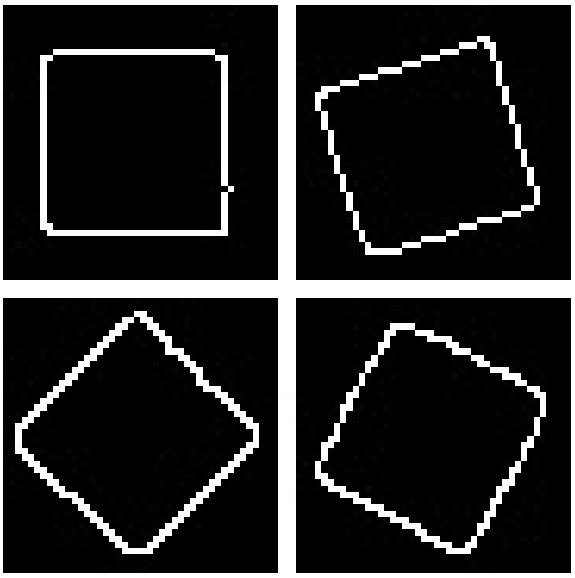


Figura A.7: Plantillas después del tratamiento detección de bordes.

A.5.2. Captura de imágenes

Los métodos *update*, *updateColor* y *updateDepth* se utilizan para tomar una o varias imágenes en un momento determinado. Reciben como parámetro una referencia a una imagen de tipo *Mat* y llaman a los métodos *update* de los objetos **color** y **depth**, que guardan la imagen capturada en la imagen pasada como parámetro. La única diferencia entre los tres métodos es la imagen que capturan; los métodos *updateColor* y *updateDepth* capturan imágenes de color y profundidad respectivamente, mientras que el método *update* captura las dos a la vez.

A.5.3. Localización de Regiones de Interés

- *locateBoxes*: este método se encarga de tres tareas: localizar la pantalla divisoria, localizar las dos cajas y diferenciar cuál de las dos está vacía. La función recibe por referencia una imagen de profundidad.

  En primer lugar se hace una umbralización entre dos valores para aislar la altura en que se encuentra la pantalla divisoria (Figura A.8). A continuación se buscan contornos, usando la función *findContours* de OpenCV, y seleccionamos de entre ellos el más grande que se encuentre aproximadamente en el centro de la imagen. Como los contornos son irregulares, se toma como Región de Interés (ROI por sus siglas en inglés) el rectángulo que lo circunscribe. La ROI y la media de los valores de sus píxeles se almacenan en los atributos **handROI** y **handROImean** para su uso posterior en la localización de la mano.

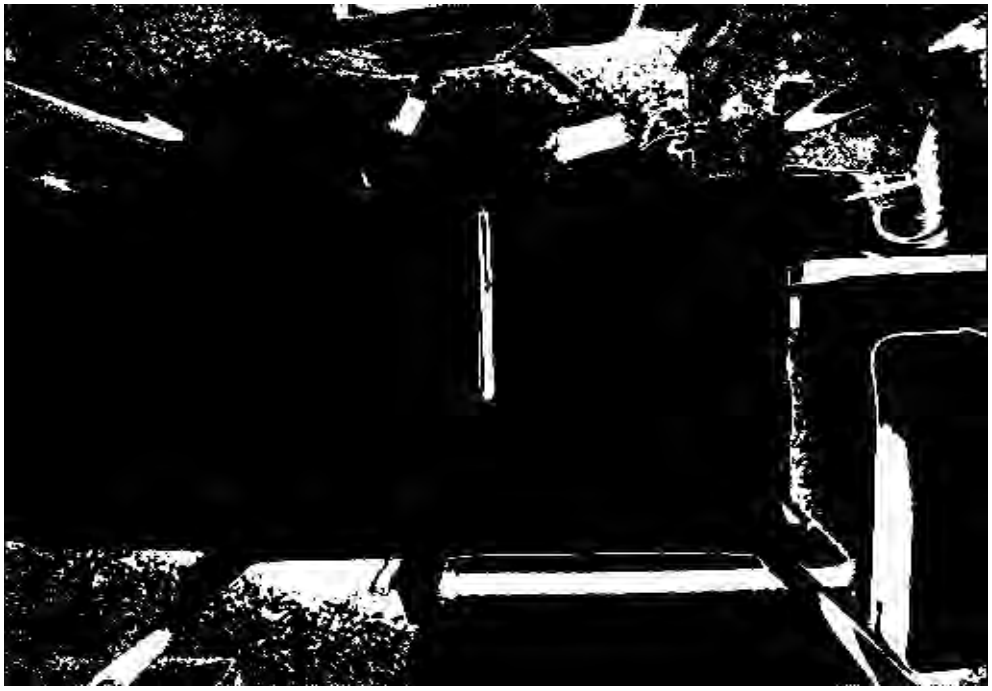


Figura A.8: Umbralización de la altura de la pantalla divisoria.

El siguiente paso es la localización de la caja. Se procede de manera análoga: umbralización entre dos valores de altura, búsqueda de contornos y selección del contorno centrado más grande. El contorno se inscribe en un rectángulo que se guarda en el atributo **boxRect**. La umbralización del borde de las cajas puede verse en la Figura A.9.

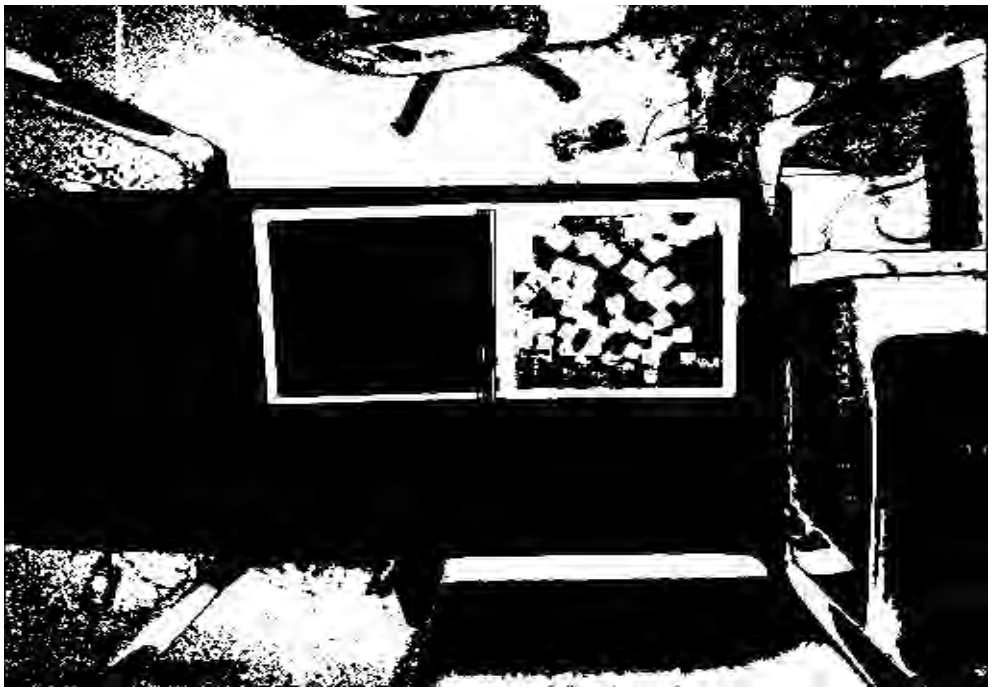


Figura A.9: Umbralización de la altura del borde de la caja.

Tomando como referencia las medidas y la posición de ese rectángulo, se localizan los centroides aproximados de los dos compartimentos de la caja. Se vuelve a hacer una umbralización con la altura de los cubos, como se ve en la Figura A.10, para discernir qué compartimento está vacío y cuál contiene cubos. Comparando la media de la intensidad de dos pequeñas regiones alrededor de los centroides se diferencia entre las dos regiones y se asigna el rectángulo vacío correspondiente al atributo **emptyBoxRect**.

Las Regiones de Interés localizadas se representan en la imagen para que el usuario pueda comprobar que han sido correctamente posicionadas antes de continuar con la prueba.

A continuación se realiza la comprobación de la presencia de luz directa sobre la caja. Para ello, se toma el valor de *Hue* de cinco píxeles de la caja vacía y se decide si es superior o inferior a un cierto umbral, fijado en 55, basándonos en la media de esos píxeles. En función del resultado de esta operación se decidirá la frontera entre el amarillo y el fondo.

Por último, se toma una imagen de profundidad con la caja vacía que se usará más adelante como altura inicial para calcular la altura a la que se encuentra la detección y discernir así el número de cubos apilados. Esto es fundamental, debido a que la distancia al sensor de

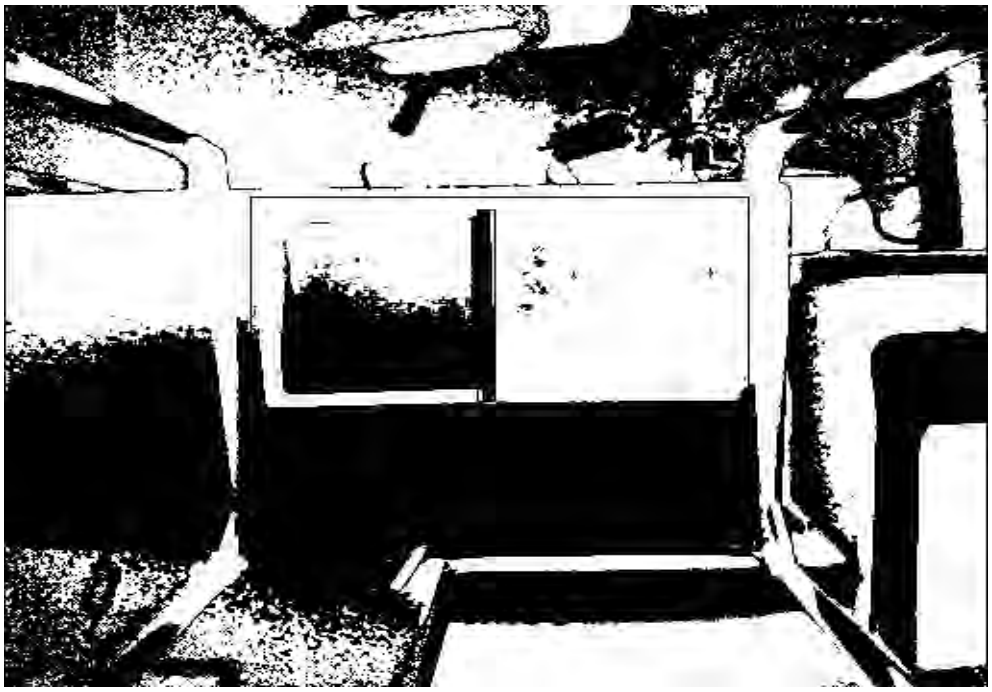


Figura A.10: Umbralización de la altura de los cubos.

dos esquinas opuestas de la cámara difiere significativamente y no se puede usar un valor medio de distancia para calcular correctamente la altura del cubo.

- *correctBoxesPosition*: este método funciona exactamente igual que el anterior, con la diferencia de que no se representan imágenes y no se recalculan la luz ni la altura inicial. Se usa para corregir la posición de las Regiones de Interés si se detecta que el paciente ha desplazado sin querer la caja. Está pensado como una manera rápida de devolver el sistema a la normalidad con la mínima distorsión en los datos posible y sin tener que interrumpir o reiniciar la prueba.

A.5.4. Detección del paso de la mano

La detección del paso de la mano por encima de la pantalla se realiza por medio de la función *detectHand* y utiliza la imagen almacenada en la matriz **handROI** durante el posicionamiento de la pantalla divisoria.

El método recibe una imagen de profundidad desde el bucle principal y, usando el rectángulo

que contiene la pantalla divisoria y fue definido en *locateBoxes*, localiza y almacena una ROI temporal en la zona en la que se sitúa la pantalla divisoria. A continuación, compara la media de la intensidad de la ROI temporal con la media de la intensidad de la ROI inicial. Si la diferencia entre ambas excede un determinado valor, fijado en 50 mediante un proceso experimental iterativo, significa que un objeto, que debería ser la mano, ha atravesado la región sobre la pantalla, alterando la profundidad. Como medida de seguridad se impone que la condición de diferencia en la media se cumpla durante al menos 5 llamadas a la función. Cuando esto ocurre, se realiza una pequeña espera para dar tiempo al paciente a dejar el bloque y se indica al bucle principal que la mano ha sido detectada.

Las Figuras A.11 y A.12 muestran la diferencia en la Región de Interés con y sin mano.

Figura A.11: ROI de la pantalla divisoria en el estado inicial.

Figura A.12: ROI de la pantalla divisoria cuando la mano pasa por encima.

Desde el bucle principal se vuelve a llamar a esta función y se repite el proceso, pero, esta vez, para detectar que la mano ha salido y se puede capturar una imagen estática de la caja.

A.5.5. Filtro White-Patch

- *setCoeffs*: el objetivo de esta función es hallar los coeficientes por los que se multiplican las intensidades de cada píxel en el filtro *White-Patch*. Como se vió en el capítulo teórico, el

filtro *White-Patch* consiste en multiplicar los valores R, G y B de cada píxel de una imagen por el coeficiente resultante de dividir el valor máximo de R, G y B presentes en la imagen entre 255.

Para ello, la función comienza localizando una región de interés en la caja que inicialmente contiene los cubos. Esta subimagen se divide en tres imágenes, una para cada canal, mediante la herramienta *split* de OpenCV. Seguidamente, se calcula el valor máximo de intensidad de cada imagen y se hallan los tres coeficientes (**coefR**, **coefG** y **coefB**), teniendo en cuenta que OpenCV almacena los canales en el orden BGRA y no RGBA, como se ha comentado anteriormente.

- *whitePatchTransform*: este método recibe una imagen en color directamente de la cámara y vuelve a hacer uso de la función *split* para separar sus canales. A continuación se multiplica cada píxel de cada canal por su coeficiente correspondiente. Esta operación es sencilla, pues OpenCV tiene sobrecargado el operador multiplicador para realizar este tipor de operaciones.

  Finalmente, los canales se vuelven a juntar mediante la función *merge* y se devuelve la imagen al bucle principal.

A.5.6. Detección

- *templateMatching*: esta función se encarga de realizar el *templateMatching* y obtener dos listas de candidatos clasificados según su nivel de confianza.

  Para ello, utilizamos la imagen donde se deben buscar candidatos y la plantilla, pasadas ambas como parámetros a la función, para llamar a la función *matchTemplate* de OpenCV. Los resultados de la detección se escriben en una imagen de tamaño igual a la recibida, restándole el ancho y el largo de la plantilla. El resultado es una imagen cuyos píxeles indican de 0 a 255 la confianza de que dicho píxel sea la esquina superior izquierda de una ventana que contiene un bloque (Figura A.13).

  En el siguiente paso se busca el píxel con el valor máximo mediante la función *minMaxLoc*. Usando este valor, umbralizaremos dos veces para hallar los puntos por encima del 70 % de ese valor y entre el 45 y el 70 %. De este modo, se obtienen dos imágenes binarias

Figura A.13: Coincidencias halladas en la imagen para la plantilla de 60°.

(ya no en escala de grises) que contienen puntos con alta probabilidad y puntos con menor probabilidad. En las Figuras A.14 y A.15 puede apreciarse el resultado de la umbralización.

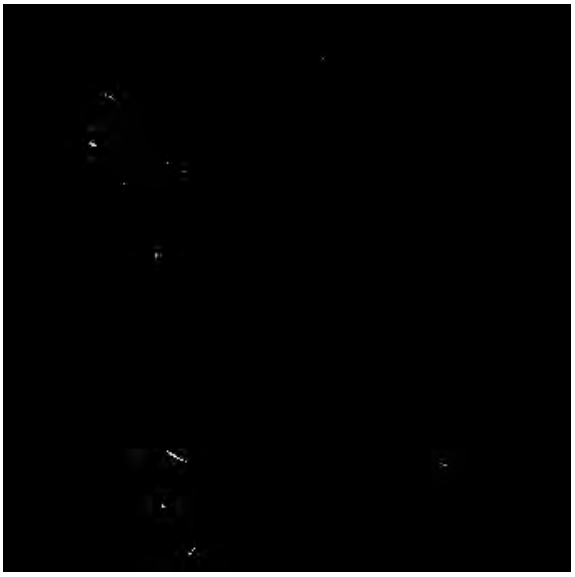


Figura A.14: Localización de los candidatos más probables.

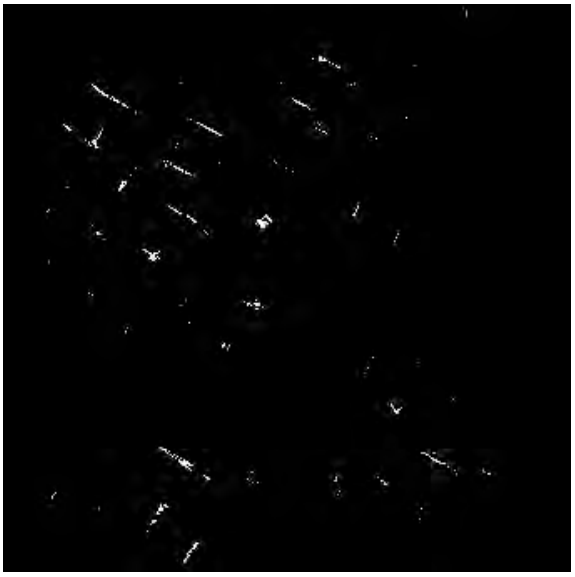


Figura A.15: Localización de los candidatos menos probables.

El último paso consiste en recorrer la imagen, pasando por todos los puntos blancos y añadir sus posiciones a un vector de puntos. Se usa de nuevo la función *minMaxLoc* para

Color	Intervalo de *Hue*
Rojo	340 - 22.8
Verde	78 - 155
Azul	198 - 242
Amarillo sin luz	50 - 73
Fondo sin luz	22 - 49.9
Amarillo con luz	55 - 68
Fondo con luz	22 - 54.9

Tabla A.1: Intervalos de *Hue* para cada color utilizado en el proyecto.

localizar un punto, se almacena su posición y se pinta de negro para no volver a pasar por él. El proceso se repite hasta que todos los puntos han sido añadidos al vector y para las dos imágenes.

El método devuelve, por tanto, dos vectores con posiciones de candidatos que serán posteriormente refinados y clasificados.

- *colorClassifier*: este método cumple varios objetivos. Por un lado sirve para clasificar, como su propio nombre indica, los puntos candidatos según su color en un vector de objetos de la clase **Block**. Por otro lado, además, sirve para descartar falsos positivos y añade una cierta seguridad frente a candidatos que no contienen totalmente al cubo.

  El algoritmo comienza leyendo los colores del centro de la ventana y cuatro píxeles adyacentes distantes 5 píxeles del centro en todas direcciones y realizando las medias de sus canales R, G y B. Con esto se busca una menor variabilidad en el color leído y que si el cubo no está centrado en la ventana, la clasificación de ese candidato falle.

  El color resultante de hacer la media se traslada al espacio HSV, por motivos de sencillez en la clasificación, como se ha explicado anteriormente. Tomando el valor del *Hue*, nos aseguramos primero de que el color no corresponde al fondo, en cuyo caso, lo clasificamos como uno de los cuatro colores de los cubos. La Tabla A.1 muestra los valores límite usados para clasificar los colores según su *Hue*.

- *detection*: el algoritmo de detección controla el proceso de búsqueda de los cubos. Como ya se ha explicado en el Capítulo 5, se han implementado tres versiones distintas con

diferentes resultados. No obstante, todas comparten la mayor parte de las características y sólo cambia el orden en que se ejecutan algunos procesos y llamadas a funciones. Por ser la más compleja, se explicará aquí la última versión del algoritmo, la que utiliza segmentación por color previa a la detección.

El método comienza redimensionando y recolocando el rectángulo que contiene la caja vacía y que fue obtenido en la función *locateBoxes*. Ahora bien, el rectángulo fue posicionado y dimensionado para la imagen de profundidad. Debido a que las imágenes de color y profundidad tienen resoluciones diferentes, se toman por sensores que se encuentran ligeramente distantes entre sí y la distorsión radial y el ángulo barrido por estos es diferente, posicionar el rectángulo en la imagen de color no es una tarea trivial. El SDK de Microsoft proporciona herramientas para hacer la correspondencia entre puntos de las dos imágenes, sin embargo, la complejidad de los cambios necesarios para introducirlas, hizo que finalmente se implementase una solución más sencilla, algo menos precisa, basada en un proceso manual de superposición de las imágenes. La Figura A.16 ilustra el proceso seguido.

Una vez posicionados los dos rectángulos, se obtienen de ellos las dos ROI, que tienen el mismo tamaño y muestran las mismas zonas, con diferencias mínimas y despreciables.

A continuación se convierte la ROI de color a HSV y se separa el canal H, con el que trabajaremos para hacer la segmentación por color. Umbralizando la imagen entre los rangos de *Hue* correspondientes a los colores rojo, azul, verde y amarillo, obtenemos cuatro imágenes binarias que contienen cubos de un único color.

Cada una de las imágenes se somete a un filtro *Canny Edge Detector* y se le hacen cuatro *template matchings*, uno por plantilla. Tenemos, pues, al final de este proceso cuatro vectores de puntos de alta probabilidad y cuatro de puntos menos fiables.

Los ocho vectores son entonces introducidos en la función *colorClassifier* y obtenemos como resultado dos vectores de objetos **Block** que ya han pasado un primer filtro y tienen asignado un color.

A continuación comienza el proceso de descarte de candidatos solapados. El algoritmo de descarte consiste en añadir a un nuevo vector aquellos bloques del vector obtenido en el paso anterior cuya posición diste al menos 20 píxeles en todas direcciones de todos los bloques guardados en el nuevo vector. Es decir, se recorre el vector antiguo bloque a bloque

Figura A.16: Agrandamiento y superposición de las dos imágenes.

comparando si existe en el vector nuevo un bloque en esa misma posición. De no existir, se añadiría al nuevo vector. Esto se realiza tres veces; una entre el vector antiguo de alta probabilidad y el nuevo, una segunda entre el vector antiguo de baja probabilidad y el nuevo y una tercera entre el vector nuevo de alta probabilidad y el nuevo de baja probabilidad. Como resultado se obtiene un solo vector que contiene bloques con una alta probabilidad de ser detecciones correctas.

El último paso de la detección consiste en examinar la profundidad en el centro de cada bloque detectado y compararla con la profundidad que tenía la caja vacía. Si la diferencia es mayor de 50, el bloque detectado se encuentra apilado sobre otro bloque que, con seguridad, no ha sido detectado y es necesario añadir como bloque de un color indeterminado. Si la profundidad fuese no sólo mayor de 50 sino también mayor de 70, se añadiría un segundo bloque de color indeterminado.

El último paso del algoritmo consiste en representar en la ROI de color las ventanas de candidatos en la posición de cada bloque detectado, con su color y tamaño correspondiente (como ya se explicó, las ventanas de candidatos detectados por profundidad son ligeramente más grandes para no tapar a las ventanas de candidatos normales).

APÉNDICE B

Marco legal y entorno socioeconómico

En este capítulo se analizarán las leyes existentes en materia de robótica y su relación con el tratamiento de pacientes, así como en torno a la protección del paciente frente al uso de los datos recabados por el sistema.

B.1. Marco legal

B.1.1. Legislación sobre Robótica

La legislación actual en torno a la robótica se encuentra en un estado poco avanzado de desarrollo, de ahí la creciente inseguridad y diversos miedos hacia los robots que se pueden observar en la ciudadanía. Si bien es cierto que países tecnológicamente más avanzados como Japón o Corea del Sur ya disponen de leyes específicas para regular sus comportamientos, límites y áreas de actuación, en la Unión Europea aún no existe legislación en firme.

En Febrero de 2017, fecha muy reciente, fue aprobado por el Parlamento Europeo un texto que contiene recomendaciones para la Comisión de Normas sobre el Derecho Civil sobre robótica [50]. No contiene, por tanto, leyes, pero trata los puntos más importantes acerca de este campo y será tomado como punto de partida para redactar unas normas flexibles y que puedan regular la actuación en lo que se prevé sea una nueva revolución tecnológica.

Según este texto, se puede definir un robot inteligente como un sistema que cumple los siguientes requisitos:

- Autonomía mediante la adquisición de datos por medio de sensores, intercambio de datos con el exterior y análisis de dichos datos.
- Opcionalmente, puede ser capaz de aprender de esos datos de manera autónoma.
- Tiene un soporte físico mínimo.
- Tiene capacidad de adaptar su comportamiento al entorno en función de los datos que recibe.
- Carece de vida en sentido biológico.

El Parlamento considera importante llevar un sistema global de registro de robots que permita su identificación y, además, es bastante tajante en que las funciones de un robot inteligente deben ser complementarias y no sustitutivas a las de un trabajador para garantizar mayor independencia, física y emocional, entre humano y máquina.

En el plano ético, uno de los que más controversias genera, incide en la importancia de respetar en todo momento los Derechos Humanos y proteger la libertad, la intimidad y la dignidad de las personas, así como sus datos personales. El marco ético en el que se diseñen y produzcan los robots debe satisfacer estas premisas y, a la vez, tener en cuenta la enorme complejidad de la robótica y sus implicaciones en diversos aspectos de la sociedad.

En el documento existe también un apartado dedicado específicamente a robots asistenciales como el que se utiliza en el presente proyecto. Señala que, si bien es cierto que se produce una deshumanización del servicio, la utilidad de estas máquinas en la realización de tareas repetitivas puede redundar en un tratamiento más específico y de mayor calidad a enfermos y personas con discapacidades por parte de los médicos.

B.1.2. Legislación sobre Protección de datos

En España, la legislación actual protege al paciente por medio de la Ley Orgánica de Protección de Datos de Carácter Personal [51]. Esta Ley dispone que, salvo excepciones, los datos personales de un ciudadano no pueden ser almacenados sin su consentimiento explícito. Entre las excepciones se puede encontrar lo siguiente:

No será preciso el consentimiento cuando los datos de carácter personal se recojan para el ejercicio de las funciones propias de las Administraciones públicas en el ámbito de sus competencias;[ ]*cuando el tratamiento de los datos tenga por finalidad proteger un interés vital del interesado en los términos del artículo 7, apartado 6, de la presente Ley.*

El artículo mencionado dispone lo siguiente:

No obstante lo dispuesto en los apartados anteriores, podrán ser objeto de tratamiento los datos de carácter personal a que se refieren los apartados 2 y 3 de este artículo, cuando dicho tratamiento resulte necesario para la prevención o para el diagnóstico médicos, la prestación de asistencia sanitaria o tratamientos médicos o la gestión de servicios sanitarios, siempre que dicho tratamiento de datos se realice por un profesional sanitario sujeto al secreto profesional o por otra persona sujeta asimismo a una obligación equivalente de secreto.

En cuanto al secreto profesional, el artículo 10 impone la confidencialidad independiente del tiempo de los datos:

El responsable del fichero y quienes intervengan en cualquier fase del tratamiento de los datos de carácter personal están obligados al secreto profesional respecto de los mismos y al deber de guardarlos, obligaciones que subsistirán aun después de finalizar sus relaciones con el titular del fichero o, en su caso, con el responsable del mismo.

Por tanto, los datos recabados por el sistema, por tratarse de datos médicos relativos a un paciente, se encuentran protegidos por la Ley frente a divulgaciones y no pueden ser cedidos a

terceras personas que no estén sometidas a secreto profesional. Su adquisición por parte de un médico no requiere consentimiento explícito del paciente por considerarse un hecho en interés de su propia salud.

Esta Ley garantiza los derechos del paciente sobre su imagen personal, al considerarla como un dato personal más. No obstante, la Ley Orgánica sobre Protección Civil del Derecho al Honor, la Intimidad Personal y Familiar y a la Propia Imagen [52], en su Artículo 8, refrenda lo expuesto en la Ley de Protección de Datos de Carácter Personal:

No se reputarán, con carácter general, intromisiones ilegítimas las actuaciones autorizadas o acordadas por la Autoridad competente de acuerdo con la ley, ni cuando predomine un interés histórico, científico o cultural relevante.

B.2. Entorno socioeconómico

Para estudiar el entorno socioeconómico y el impacto que esta investigación puede tener, es necesario entender primero a qué se refiere el término «Socioeconomía».

La socioeconomía es la concepción de la economía que considera que el mercado no se mueve por el interés propio de cada consumidor, sino que, al estar inmerso en una sociedad, hay gran cantidad de factores externos que condicionan a los individuos. Factores como la cultura, los valores, las emociones o los prejuicios son determinantes a la hora de que el consumidor se decante por un producto u otro [53].

Esta definición está también relacionada con el cambio de paradigma de la estrategia empresarial. La ventaja competitiva de una empresa ha dejado de estar centrada exclusivamente en conseguir un producto mejor que el de la competencia, sino que ahora se valoran otros aspectos. Además de, por supuesto, calidad y precio, el cliente valora el diseño, la ética de la empresa, la sostenibilidad de la producción... Estas valoraciones se extienden al resto de *stakeholders* de la empresa de cara, por ejemplo, a conseguir financiación o materia prima.

Por tanto, a la hora de valorar la sostenibilidad de un producto es necesario tener en cuenta

que sea sostenible en tres aspectos: social, medioambiental y económico.

El sistema desarrollado en este proyecto, aunque no es aún apto para su uso comercial, podría ser introducido en el mercado en un futuro. Si bien es cierto que su área de actuación es muy concreta y está dirigido, en principio, a hospitales y clínicas de rehabilitación de pacientes, una simplificación de la interfaz podría acercar el sistema directamente al paciente. De este modo, el paciente podría adquirir un medio para hacer ejercicios de rehabilitación en su propia casa, sin necesidad de desplazamientos, que pueden resultar incómodos para un individuo afectado de parálisis lateral, y enviar telemáticamente los resultados a su médico.

Dado que el producto podría mejorar la calidad de vida de varios miles de personas en España, por medio de la rehabilitación y aumentando también su comodidad, es evidente que la influencia y repercusión social es enorme.

El sistema tiene, por supuesto, un cierto impacto ambiental, en tanto que la fabricación de los componentes es contaminante. No obstante, sin tener en cuenta el posible beneficio por la disminución de los desplazamientos periódicos del paciente, presumiblemente en transporte motorizado, el impacto medioambiental no es especialmente relevante en una sociedad altamente contaminante. Menos aún si tenemos en cuenta que los sistemas no pueden producirse en grandes lotes, sino que se seguiría una producción bajo demanda o, como mucho, en lotes pequeños para no incurrir en grandes costes de almacenaje y obsolescencia. De este modo se garantizaría un impacto ambiental bajo.

En cuanto a la perspectiva económica, tal y como se expuso en el capítulo introductorio, el aumento de la esperanza de vida y hábitos de vida poco recomendables han provocado, y seguirán haciéndolo, un aumento del número de casos de enfermedades cardiovasculares, una de las principales fuentes de pacientes aquejados de pérdida de movilidad. Por tanto, tristemente, se prevé que el número de clientes potenciales para una versión avanzada del sistema sea alto. La estrategia de *marketing* de cara a hospitales y clínicas debería centrarse en la demostración del sistema y, quizá, la venta a través de páginas web especializadas en sistemas de rehabilitación. Para venta a particulares, el hospital debería hacer de enlace entre él y la empresa.

APÉNDICE C

Planificación y presupuesto del proyecto

En este apéndice se presentará un análisis de las tareas que componen el proyecto y su marco temporal aproximado.

Así mismo, se desglosarán los costes derivados de la realización de este Trabajo de Fin de Grado.

C.1. Planificación

La Figura C.1 muestra la planificación mensual de las principales tareas desarrolladas.

Figura C.1: Diagrama de Gantt del proyecto.

C.2. Presupuesto

Los costes asociados a la realización del proyecto se pueden dividir en costes de material y coste de mano de obra.

C.2.1. Coste de material

En la Tabla C.1 se recogen los costes dependientes del hardware y otros materiales utilizados. En estos costes no se incluyen costes de material fungible, por considerarse despreciables, ni costes derivados de la documentación o el software, pues todos estos recursos han sido elegidos gratuitos o, de preferencia, *open-source*.

Material	Coste (€)
Kit BBT	210
Ordenador portátil MSI GT72 2QE-881ES	2099
Kinect V2	138.50
Bastidor	60
TOTAL MATERIAL	**2507.5**

Tabla C.1: Costes de material.

C.2.2. Coste de mano de obra

En la Tabla C.3 se muestran las diferentes fases del proyecto y el tiempo aproximado empleado en cada una de ellas. El total de horas trabajadas se encuentra en torno a las 436, lo que supera con creces los 12 ECTS correspondientes al Trabajo de Fin de Grado.

Los costes horarios de personal se han estimado tomando como referencia los costes de trabajadores adscritos a la Universidad Carlos III de Madrid en función de la tarea desempeñada, como se puede ver en la Tabla C.2.

Teniendo en cuenta que las cargas salariales (Seguridad Social, IRPF, Desempleo...) en España

Puesto	Coste horario
Ingeniero Técnico	30 €/h
Secretaría	20 €/h

Tabla C.2: Coste horario de un trabajador en la UC3M.

de un trabajador medio son de alrededor del 39.5 %, los costes totales derivados de la mano de obra, sin incluir desplazamientos ni dietas, son los que aparecen en la Tabla C.3

Fases del proyecto	Horas empleadas	Coste horario estimado (€/h)	Coste total (€)
Documentación e investigación	20	41.85	837
Desarrollo del software	244	41.85	10211.40
Pruebas	12	41.85	502.2
Elaboración de la memoria	160	27.90	4464
TOTAL MANO DE OBRA			**16014.60**

Tabla C.3: Costes de mano de obra.

C.2.3. Coste total presupuestado

El coste total presupuestado para este proyecto se muestra en la Tabla C.4 y asciende a 18522.10 €, DIECIOCHO MIL QUINIENTOS VEINTIDÓS EUROS CON DIEZ CÉNTIMOS.

Concepto	Coste (€)
Mano de obra	16014.60
Material	2507.50
TOTAL PROYECTO	**18522.1**

Tabla C.4: Coste total del proyecto.

Bibliografía

[1] Datos Macro, "España - esperanza de vida al nacer." Website, 2014. [Último acceso: Junio-2017] http://www.datosmacro.com/demografia/esperanza-vida/espana.

[2] Instituto Nacional de Estadística, "Defunciones según la causa de la muerte." Website. [Último acceso: Junio-2017] http://www.ine.es/jaxiT3/Datos.htm?t=7947.

[3] M. A. Y. Levan Atanelov, Steven A. Stiens, "History of physical medicine and rehabilitation and its ethical dimensions," *AMA Journal of Ethics*, vol. 17, no. 6, pp. 568–574, 2015.

[4] M. B. Brown, *Introduction to Massage Therapy*. Lippincott Williams & Wilkins, 2013.

[5] Claire Voon, "The sophisticated design of a 3,000-year-old wooden toe." Website. [Último acceso: Junio-2017] https://hyperallergic.com/387047/the-sophisticated-design-of-a-3000-year-old-wooden-toe/.

[6] F. G. Romero, *EL DEPORTE GRIEGO Y EL DEPORTE ACTUAL: INFLUENCIA, SEMEJANZAS Y DIFERENCIAS*. Madrid: SERVICIOS DE GESTION Y COMUNICACION LICEUS, 2012.

[7] J. Wirotius, *Enciclopedia Médico-Quirúrgica*, ch. Historia de la rehabilitación, pp. 1–26. París: Editions Scientifiques et Medicales, Elsevier SAS, 1999.

[8] W. Watkins, "Hotes des invalides, facade principale." Website. [Último acceso: Junio-2017] http://www.rareoldprints.com/p/5895.

[9] A. A. Conti, "Western medical rehabilitation through time: A historical and epistemological review," *The Scientific World Journal*, vol. 2014, no. 432506, p. 5, 2014.

[10] Medlineplus, "Accidente cerebrovascular." Website. [Último acceso: Junio-2017] https://medlineplus.gov/spanish/ency/article/000726.htm.

[11] Stroke Rehab, "Everything about stroke rehab, treatment and recovery." Website. [Último acceso: Junio-2017] http://www.stroke-rehab.com/.

[12] Flint Rehab, "37 hand therapy exercises for stroke recovery." Website. [Último acceso: Junio-2017] https://www.flintrehab.com/2015/hand-therapy-exercises-after-stroke/.

[13] Flint Rehab, "Musicglove: Hand therapy with a beat." Website. [Último acceso: Julio-2017] https://www.flintrehab.com/musicglove/?utm_source=Blog&utm_medium=Text&utm_content=More%20MusicGlove%20Info.

[14] BinaryLabs, "Dexteria fine motor/rehab aid." Website. [Último acceso: Julio-2017] https://play.google.com/store/apps/details?id=com.binarylabs.dexteria&hl=es.

[15] Sara Zuboff, "Minnesota dexterity test guide and instructions." Website. [Último acceso: Julio-2017] http://www.prohealthcareproducts.com/blog/minnesota-dexterity-test-guide-and-instructions/.

[16] Pro Healthcare Products, "Purdue pegboard manual dexterity test." Website. [Último acceso: Julio-2017] http://www.prohealthcareproducts.com/blog/purdue-pegboard-manual-dexterity-test/.

[17] Valero-Cuevas, "The strength-dexterity test as a measure of pinch performance in the able and impaired hand." article, 2002.

[18] Canadian Partnership for Stroke Recovery, "Jebsen hand function test (jhft)." Website. [Último acceso: Julio-2017] https://www.strokengine.ca/indepth/jhft_indepth/.

[19] Mobility Smart, "Jebsen-taylor hand function test kit." Website. [Último acceso: Julio-2017] http://www.mobilitysmart.cc/jebsen-taylor-hand-function-test-kit.html.

[20] Ali Dawood, "Rehab measures: Box and block test." Website, 2010. [Último acceso: Diciembre-2016] http://www.rehabmeasures.org/Lists/RehabMeasures/DispForm.aspx?ID=917.

[21] Sabrina Figueredo, "Box and block test." Website, 2011. [Último acceso: Julio-2017] http://www.strokengine.ca/assess/bbt/.

[22] Mathiowetz V, Volland G, Kashman N, Weber K., "Adult norms for the box and block test of manual dexterity." Website, 1985. [Último acceso: Junio-2017] http://www.ncbi.nlm.nih.gov/pubmed/3160243.

[23] Microsoft, "Kinect for windows SDK." Website, 2014. [Último acceso: Febrero-2017] https://msdn.microsoft.com/es-es/library/dn799271.aspx.

[24] OpenCV, "Opencv tutorials." Website, 2017. [Último acceso: Mayo-2017] http://docs.opencv.org/master/d9/df8/tutorial_root.html.

[25] Tsukasa Sigiura, "Kinect v2. introduction and tutorial." Website. [Último acceso: Junio-2017] https://www.slideshare.net/SugiuraTsukasa/kinect-v2-introduction-and-tutorial.

[26] E. Oña, A. Jardón, and C. Balaguer, "Toward an automated assessment method of manual dexterity," in *2016 International Workshop on Assistive and Rehabilitation Technology (IWART)*, pp. 1–2, December 2016.

[27] E. Oña, A. Jardón, and C. Balaguer, "The automated box and blocks test an autonomous assessment method of gross manual dexterity in stroke rehabilitation," in *Conference Towards Autonomous Robotic Systems*, pp. 101–114, July 2017.

[28] C.-P. Hsiao, C. Zhao, and E. Y. L. Do, "The digital box and block test automating traditional post-stroke rehabilitation assessment," in *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 360–363, March 2013.

[29] Microsoft, "Stroke recovery gets a boost from kinect." Website, 2014. [Último acceso: Junio-2017] https://www.microsoft.com/en-us/research/blog/stroke-recovery-gets-a-boost-from-kinect/.

[30] BLINC, "Modified box and blocks test with motion capture." Website. [Último acceso: Junio-2017] https://blinclab.ca/research/outcome-metrics/modified-box-and-blocks-test-with-motion-capture/.

[31] J. S. Hebert and J. Lewicke, "Case report of modified box and blocks test with motion capture to measure prosthetic function," *Journal of Rehabilitation Research & Development (JRRD)*, vol. 49, no. 8, pp. 1163 – 1174, 2012.

[32] C. Chen, C. Dietrich, and C. Miller, "Cubelet detection and localization by template matching," 2012.

[33] vuanhtung2004, "Using opencv matchtemplate for blister pack inspection." Website. [Último acceso: Abril-2017] https://stackoverflow.com/questions/23180630/using-opencv-matchtemplate-for-blister-pack-inspection.

[34] López Peña, Valveny, Vanrell, "Detección de objetos," MOOC, Universidad Autònoma de Barcelona, Julio 2016. [Último acceso: Diciembre-2016] https://www.coursera.org/learn/deteccion-objetos/home/welcome.

[35] A. Das, M. Pukhrambam, and A. Saha, "Real-time robust face detection and tracking using extended haar functions and improved boosting algorithm," in *Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on*, pp. 981–985, 2015. ID: 1.

[36] D. P. Mital, G. W. Leng, and T. E. Khwang, "Colour vision for industrial applications," in *Industrial Electronics Society, 1990. IECON '90., 16th Annual Conference of IEEE*, pp. 548–551 vol.1, 1990. ID: 1.

[37] K. M. Kramer, D. S. Hedin, and D. J. Rolkosky, "Smartphone based face recognition tool for the blind," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 4538–4541, 2010. ID: 1.

[38] Y. Liu, J. Yang, and M. Liu, "Recognition of qr code with mobile phones," in *2008 Chinese Control and Decision Conference*, pp. 203–206, 2008. ID: 1.

[39] V. Tiwari, V. Anand, A. G. Keskar, and V. R. Satpute, "Sign language recognition through kinect based depth images and neural network," in *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, pp. 194–198, 2015. ID: 1.

[40] F. Cordella, F. D. Corato, L. Zollo, B. Siciliano, and P. van der Smagt, "Patient performance evaluation using kinect and monte carlo-based finger tracking," in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pp. 1967–1972, 2012. ID: 1.

[41] X. S. Cuadros, *Human Body Analysis using Depth Data*. PhD thesis, Universitat Politecnica de Catalunya, 2013.

[42] Marcal Monserrat, "Características kinect 2." Website. [Último acceso: Julio-2017] http://www.kinectfordevelopers.com/es/2014/01/28/caracteristicas-kinect-2/.

[43] MSI, "Gt72 2qe dominator pro." Website. [Último acceso: Agosto-2017] https://es.msi.com/Laptop/GT72-2QE-Dominator-Pro.html#hero-specification.

[44] Microsoft, "System requirements." Website. [Último acceso: Junio-2017] https://msdn.microsoft.com/es-es/library/dn782036.aspx.

[45] manu, "Entrevista a Héctor Martín, responsable del hackeo a Kinect en cuestión de horas." Website. [Último acceso: Agosto-2017] https://hipertextual.com/archivo/2010/11/entrevista-hector-martin-kinect/.

[46] OpenKinect, "Main page." Website. [Último acceso: Julio-2017] https://openkinect.org/wiki/Main_Page.

[47] Microsoft, "Bienvenido a visual studio 2015." Website. [Último acceso: Julio-2017] https://msdn.microsoft.com/es-es/library/dd831853.aspx.

[48] OpenCV, "Opencv." Website, 2017. [Último acceso: Agosto-2017] http://opencv.org/.

[49] Albatross, "A brief description." Website, 2017. [Último acceso: Agosto-2017] http://www.cplusplus.com/info/description/.

[50] Parlamento Europeo, "Normas de derecho civil sobre robótica," 2017.

[51] "Ley orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal," Dec. 1999. [Último acceso: Septiembre-2017] https://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750.

[52] "Ley orgánica 1/1982, de 5 de mayo, de protección civil del derecho al honor, a la intimidad personal y familiar y a la propia imagen," June 1982. [Último acceso: Septiembre-2017] http://www.boe.es/buscar/act.php?id=BOE-A-1982-11196&p=20100623&tn=1.

[53] José Pérez Adán, "Socioeconomía." Website. [Último acceso: Septiembre-2017] http://www.uv.es/sasece/socioeconomia.htm.