Ensamblaje automatizado de partes utilizando técnicas de Aprendizaje por Demostración y visión 3D.

David Arturo Duque Arias

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica.
Bogotá, Colombia
Año 2017

Ensamblaje automatizado de partes utilizando técnicas de Aprendizaje por Demostración y visión 3D.

David Arturo Duque Arias

Tesis o trabajo de grado presentada(o) como requisito parcial para optar al título de:
M.Sc. en Ingeniería - Automatización Industrial

Director:
Ph.D., Flavio Augusto Prieto Ortiz

Co-Director:
Ph.D., José Gabriel Hoyos Gutiérrez

Línea de Investigación: Visión de Máquina
Grupo de Investigación: GAUNAL - Grupo de Automática de la Universidad Nacional

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica.
Bogotá, Colombia
Año 2017

Agradecimientos

A mis padres, Luis Arturo y Maria Nelcy, a mi hermano, Daniel y a mi esposa, Paola. Por ser quienes día a día me han brindado ese apoyo incondicional sin el cual no hubiera culminado esta etapa de mi vida.

A mi director, Flavio Augusto Prieto, y a mi codirector, José Gabriel Hoyos, quienes con su acompañamiento, orientación y aportes, apoyaron en todo momento el desarrollo del proyecto de investigación.

A la Universidad Nacional de Colombia, por brindarme la formación académica desde pregrado y a la cual espero retribuirle todo lo que ha aportado a mi vida.

Resumen

El número de robots industriales y el tipo de aplicaciones en que son utilizados, como las operaciones de ensamble, ha incrementado considerablemente en los últimos años. Sin embargo, la programación de estos sigue siendo una tarea que requiere elevados conocimientos técnicos y un alto consumo de tiempo. La metodología desarrollada en el presente trabajo, utiliza técnicas de Aprendizaje por Demostración (ApD) y visión de máquina, con el fin de ejecutar diferentes tipos de operaciones de ensamble, con manipulares robóticos. Se proponen seis etapas fundamentales: *i)* Demostración de las tareas de ensamble; *ii)* Adquisición de las demostraciones usando un sensor de movimiento Kinect; *iii)* Preprocesamiento y segmentación de las demostraciones; *iv)* Entrenamiento de los modelos probabilísticos TP-GMM (Task Parameterized Gaussian Mixture Model); *v)* Generación de nuevas trayectorias usando modelos entrenados; *vi)* Simulación e implementación en robots reales. Adicionalmente, se propone un sistema para la generación automática de los planes de ensamble usando redes de Petri. Los resultados obtenidos permiten establecer que es posible utilizar técnicas de programación diferentes a la metodología tradicional, obteniendo resultados satisfactorios en operaciones de ensamble de partes, como se demuestra en el trabajo desarrollado.

Palabras clave: Aprendizaje por Demostración, modelo de mezcla de Gaussianas, modelo de mezcla de Gaussianas parametrizadas en la tarea, regresión de mezcla de Gaussianas, redes de Petri.

Abstract

The number of industrial robots and the type of applications in which they are used, such as assembly operations, has increased considerably in recent years. However, programming is still a task that requires high technical knowledge and a high consumption of time. The methodology developed in the present work uses techniques of Learning by Demonstration (LbD) and machine vision, in order to execute different types of assembly operations, with robotic manipulators. Six basic stages are proposed: *i)* Demonstration of assembly tasks; *ii)* Acquisition of demonstrations using a Kinect motion sensor; *iii)* Preprocessing and segmentation of demonstrations; *iv)* Training of probabilistic models TP-GMM (Task Parameterized Gaussian Mixture Model); *v)* Generation of new trajectories using trained models; *vi)* Simulation and implementation in real robots. In addition, it's proposed a system for the automatic generation of assembly plans using Petri nets. The obtained results allow to establish that it is possible to use programming techniques different from the traditional methodology, obtaining satisfactory results in operations of assembly of parts, as demonstrated in the work developed.

Keywords: Learning by Demonstration, Gaussian Mixture Model, Task Parameterized Gaussian Mixture Model, Gaussian Mixture Regression, Petri nets.

Contenido

Lista de Figuras

Lista de Tablas

Lista de símbolos

Abreviaturas

Abreviatura	**Término**
ApD	Aprendizaje por Demostración
PpD	Programación por Demostración
FPM	Filtro Promedio Móvil
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Regression
HSV	Hue, Saturation, Value
LED	Light-Emittin Diode
PCA	Principal Component Analysis
RGB	Red, Green, Blue
$TP-GMM$	Task Parameterized GMM

1 Introducción

El uso de robots ha aumentado considerablemente en la última década, en especial en los entornos industriales, con el fin de ejecutar diferentes tipos de aplicaciones como soldadura, pintura, ensamble de piezas, entre otras. Así mismo, se han desarrolladado herramientas que permiten adquirir información del entorno, como son los sensores de fuerza y los sistemas de visión de máquina. Con relación a esto último, el desarrollo de sistemas robóticos equipados con una o más cámaras, permite observar con mayor precisión el espacio de trabajo, estimando, por ejemplo, la posición y orientación de las piezas disponibles. Sin embargo, a pesar de lo anterior, la metodología de programación tradicional de los robots continúa siendo una etapa crítica para la ejecución de nuevas tareas, debido a varias desventajas que presenta, tales como: ***i)*** Necesidad de personal técnico ampliamente capacitado; ***ii)*** Elevado consumo de tiempo en la etapa de entrenamiento del operario encargado; ***iii)*** Baja flexibilidad para ejecutar satisfactoriamente las operaciones, cuando se presentan cambios en la operación.

Adicionalmente, debido a la baja flexibilidad del sistema específico, la generación del plan de operación es, en la mayoría de los casos, un procedimiento manual, el cual es desarrollado directamente por el operario encargado de la programación del robot. A raiz de lo anterior, se evidencia la necesidad de desarrollar metodologías que permitan programar robots industriales, capaces de adquirir información del entorno y, consecuentemente, adaptar las operaciones en función de los objetos disponibles en la escena y la configuración de los mismos en el espacio de trabajo.

Es por esto que se han desarrollado metodologías tales como ApD (Aprendizaje por Demostración) o PpD (Programación por Demostración) [6, 7, 8], las cuales tienen como objetivo facilitar la etapa de programación de los robots. Como principal característica, el uso de estas técnicas busca replicar el método de aprendizaje de los seres vivos, el cual consiste, a grandes rasgos, en observar la ejecución de una tarea, extraer la información relevante y, finalmente, generalizar las acciones para ser implementadas ante nuevas condiciones. A partir de lo anterior, surge la necesidad de desarrollar un sistema capaz de adquirir, visualmente, la información suficiente para caracterizar una tarea en específico y aprender un modelo que sea capaz de tomar decisiones ante nuevas condiciones del entorno.

Para tal fin, se desarrolló un sistema que permite ejecutar operaciones de ensamble, utilizando técnicas de PpD, con un manipulador industrial ABB IRB 140. El trabajo se desarrolló

en seis fases principales: ***i)*** Adquisición de las demostraciones de la tarea, ejecutadas por una persona; ***ii)*** Entrenamiento del modelo probabilístico que representa la demostraciones; ***iii)*** Generación de nuevas trayectorias, ante diferentes condiciones de la operación; ***iv)*** Generación automática del plan de ensamble, de acuerdo a las piezas disponibles en el espacio de trabajo; ***v)*** Simulación usando el software RobotStudio; y ***vi)*** Implementación en el robot ABB disponible (Ver Figura **1-1**). El uso de herramientas de simulación permitió comprobar si las operaciones de ensamble propuestas eran válidas, sin necesidad de ejecutar directamente la tarea en el brazo robótico. Los resultados obtenidos demuestran que es posible integrar técnicas de ApD para la ejecución de operaciones industriales en robots, tal como lo son operaciones de ensamble. Adicionalmente, se evidencia que el uso de representaciones gráficas, tales como las redes de Petri, permiten generar automáticamente el plan a seguir para la ejecución de una tarea. Es importante mencionar que, previo al desarrollo del sistema para operaciones de ensamble en 3D, se diseñó e implementó una metodología para tareas de manipulación de bloques en 2 dimensiones.

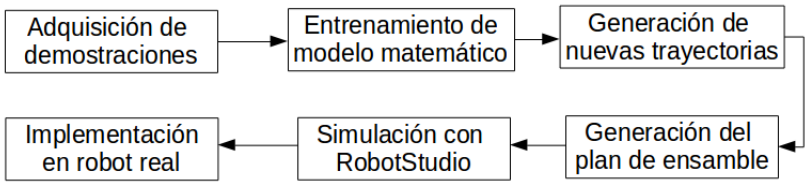

Figura 1-1: Principales etapas de la metodología desarrollada.

Durante el desarrollo del proyecto de investigación, se cumplieron satisfactoriamente el objetivo general y los objetivos específicos propuestos. Adicionalmente, a partir de los resultados obtenidos, se presentó la ponencia titulada "Manipulating blocks using Task Parameterized Gaussian Mixture Models and Machine Vision" en la mesa principal del CIIMA 2016, llevado a cabo en la Universidad Santo Tomas, seccional Bucaramanga, en el mes de Octubre del año 2016. Por otra parte, el artículo "Towards the assembly of parts with robots using Learning by Demonstration and machine vision" fue presentado al Journal of Manufacturing Science and Engineering - Transactions of the ASME, y actualmente se encuentra en etapa de revisión.

La estructura del documento es la siguiente: en el segundo capítulo, se presenta la fundamentación teórica del Aprendizaje por Demostración, de los modelos probabilísticos Gaussian Mixture Model (GMM), Gaussian Mixture Regression (GMR) y Task Parameterized GMM (TP-GMM) y de las técnicas de visión de máquina implementadas; en el tercero, se describe el sistema propuesto para la ejecución de tareas de manipulación de bloques en 2 dimensiones; en el cuarto, se presenta el sistema propuesto para la ejecución de operaciones de ensamble en 3 dimensiones; en el quinto, se presentan los resultados obtenidos de los sistemas

presentados; y en el sexto, se presenta la discusión de los resultados y las conclusiones.

2 Marco teórico

La implementación de técnicas de Aprendizaje por Demostración (ApD) o de Programación por Demostración (PpD), requiere de sensores que permitan adquirir la información relevante de las demostraciones y de los objetos que son usados en la tarea. En el caso general, se tiene un robot aprendiz, el cual, a partir de un conjunto de demostraciones de una tarea en particular, generaliza y aprende cómo ejecutar la acción ante nuevas condiciones. Las demostraciones son ejecutadas por un maestro, el cual tiene las habilidades suficientes para ejecutar correctamente la tarea, ante diferentes condiciones de la operación [9].

De acuerdo al tipo de maestro que se seleccione, ya sea otro robot, una persona o el mismo robot aprendiz, es necesario equipar al sistema con diferentes tipos de sensores para adquirir la información relevante de las demostraciones. Adicionalmente, las demostraciones pueden ser ejecutadas en entornos reales o virtuales, como se presenta en [8], [10], [11] y [12]. En la presente investigación, las demostraciones son ejecutadas por una persona, y adquiridas visualmente utilizando un sensor de movimiento Kinect de primera generación [13]. Se agregaron etapas de filtrado y segmentación, para reducir el ruido e identificar las subtareas de las demostraciones adquiridas visualmente. El robot aprendiz, es un manipulador industrial ABB IRB-140, disponible en el laboratorio de Sistemas Inteligentes Robotizados, de la Universidad Nacional de Colombia, sede Bogotá.

A partir de lo anterior, entonces se presentarán: la fundamentación, características más importantes y algunos trabajos desarrollados, en el área de ApD, utilizando manipuladores robóticos. Luego, las técnicas de visión de máquina que permiten adquirir las demostraciones ejecutadas por el maestro y la posición de las piezas en el espacio de trabajo. A continuación, se presentan las redes de Petri, las cuales son usadas para obtener una representación gráfica de una tarea. Finalmente, la fundamentación en los modelos probabilísticos que permiten representar matemáticamente las demostraciones de la operación, tales como GMM (modelo de mezcla de Gaussianas), GMR (regresión de mezcla de Gaussianas) y TP-GMM (modelo de mezcla de Gaussianas parametrizadas en la tarea), y generar automáticamente nuevas trayectorias en función de la posición de los objetos en el espacio de trabajo.

2.1. Aprendizaje por Demostración

El ApD está compuesto por un conjunto de técnicas desarrolladas con el fin de reducir la complejidad del proceso de programación y el tiempo requerido para implementarlo. Estos dos problemas pueden ser solucionados, en gran parte, si el proceso de programación está vinculado a un proceso de aprendizaje en el cual el algoritmo extrae los objetivos de las tareas, a partir de un conjunto de demostraciones. El uso de técnicas de ApD (o PpD), tiene como objetivo emular el proceso de aprendizaje de los seres vivos, el cual presenta las siguientes etapas [14]:

1. Identificar la acción que será aprendida por el aprendiz.
2. Encontrar al maestro con las habilidades de ejecutar satisfactoriamente la acción.
3. El aprendiz observa cómo el maestro ejecuta satisfactoriamente la tarea.
4. El maestro repite la tarea hasta que el aprendiz identifica las políticas que describen las demostraciones ejecutadas.
5. El aprendiz genera un plan de acciones a partir de lo aprendido y las condiciones actuales de la tarea.
6. El aprendiz ejecuta el plan, utilizando su propia arquitectura.
7. Si la acción ejecutada por el aprendiz no es satisfactoria, se repite nuevamente el proceso desde el paso *3)*.
8. Verificar que el aprendiz cumple los objetivos de la tarea, como fue demostrado por el maestro.

A partir de las etapas anteriores, se evidencia la necesidad de contar con un maestro con la habilidad de demostrar cómo ejecutar la tarea a ser aprendida. En el caso de la implementación de ApD en robots, se cuenta con tres tipos de maestros para que ejecuten las demostraciones [6]: *i)* Otro robot; *ii)* Robot aprendiz; y *iii)* Una persona. Así mismo, de acuerdo al tipo de maestro que se seleccione, se requerirán sensores que permitan adquirir la información relevante de las demostraciones. A continuación se presentan las características, ventajas y desventajas, así como algunos de los resultados obtenidos en diferentes trabajos desarrollados, con respecto al tipo de maestro que ejecuta las demostraciones.

2.1.1. Otro robot

Como se presenta en [6], el uso de otro robot como maestro, requiere que éste haya sido previamente programado para ejecutar la tarea frente a diferentes condiciones o que sea controlado, remota o manualmente, por una persona. En [15], se presentan unos de los primeros

resultados obtenidos usando ApD. En este caso, se usó un carro a control remoto como maestro para demostrarle a otro carro, cómo recorrer un laberinto. El carro aprendiz estaba equipado con una cámara y varios sensores de proximidad, con el objetivo de identificar la posición del maestro, el cual tenía LED's en la parte posterior, y las paredes del laberinto, respectivamente. A partir de la información adquirida, se genera un modelo que permite al carro aprendiz recorrer diferentes laberintos. En la Figura **2-1** se presenta la imagen en escala de grises, adquirida mediante la cámara del robot aprendiz, y la imagen procesada, en la cual es posible identificar los LED's de la parte posterior del vehículo maestro.

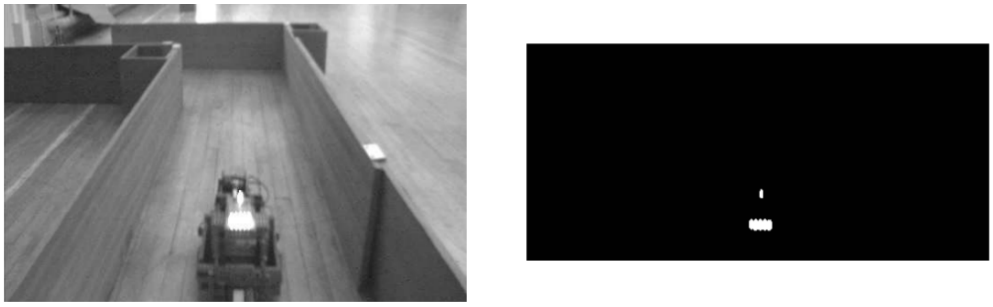

Figura 2-1: a) Vista del maestro y del laberinto, desde aprendiz. b) Imagen procesada. [15]

2.1.2. Robot aprendiz

El uso del robot aprendiz, como robot maestro ha sido implementado en múltiples aplicaciones. Por ejemplo, en [16], se usa un robot equipado con una mano antropomórfica, con el objetivo de agarrar diferentes tipos de piezas como envases plásticos, pocillos, destornilladores, entre otros. En este caso, para la etapa de aprendizaje, las demostraciones son ejecutadas utilizando el robot aprendiz, el cual es movido manualmente por una persona. Es decir, un operario mueve el efector final del robot y agarra diferentes tipos de objetos, en diferentes posiciones y orientaciones. En la Figura **2-2** se presenta el robot George [16] el cual cuenta con un sistema de visión estereo para adquirir la información del entorno.

En [1], se usa un robot antropomórfico para preparar pizza. En este trabajo, el efector final es movido manualmente por un operador, quien ejecuta en múltiples oportunidades la tarea, variando la posición de los ingredientes en el espacio de trabajo, como se presenta en la Figura **2-3**. A diferencia de [16], en este caso, el robot no estaba equipado con cámaras para adquirir información del entorno, sino que únicamente se adquirió la posición de cada articulación, mediante sensores tipo encoders.

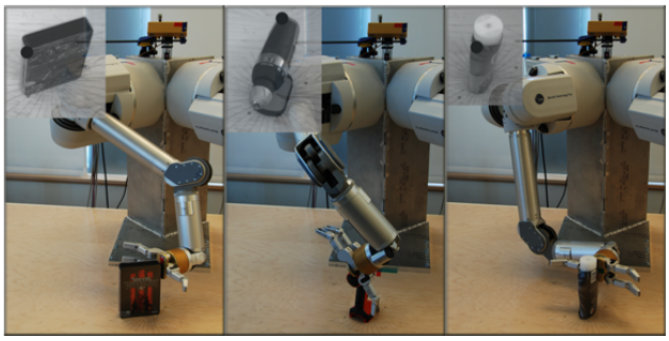

Figura 2-2: Robot George, equipado con mano antropomórfica y visión estereo. [16]

Figura 2-3: Demostración y ejecución de tarea de preparación de pizza [1].

2.1.3. Una persona

Se han desarrollado un amplio número de trabajos de ApD en los que las demostraciones son ejecutadas por una persona. Es importante resaltar que el uso de este tipo de metodología está enfocada al desarrollo de entornos de trabajo seguro en los que interactúen humanos y robots. Así mismo, está ligado al concepto de robots de servicio [17], los cuales tiene como objetivo aumentar el tipo de aplicaciones en que los robots son capaces de ejecutar, y así integrarlos a operaciones de la vida cotidiana de los seres humanos.

En [2] se presenta un sistema que usa ApD, en el que una persona demuestra cómo jugar "Air Hockey" y cómo resolver laberintos, a un robot humanoide DB, el cual tiene 30 grados de libertad. En [3] se ejecutan operaciones de tipo "Pick and Place", con el robot Robo-Erectus Jr, el cual toma una bola y la deposita en un recipiente. En los dos trabajos anteriores, las demostraciones de las operaciones fueron ejecutadas por un persona, y adquiridas mediante un sistema de visión de máquina, compuesto por una o más cámaras; adicionalmente, se evidencia la necesidad de usar algoritmos de visión artificial que permitan identificar la información relevante de la operación. En [2], la información adquirida visualmente fue la posición del disco de hockey en todo instante de tiempo; mientras que en [3], se siguió el movimiento del brazo del maestro, quien movió la bola de la operación y la depositó en el

recipiente. En la Figura **2-4** se presentan los trabajos anteriores, en los cuales es el maestro que demuestra las operaciones es una persona.

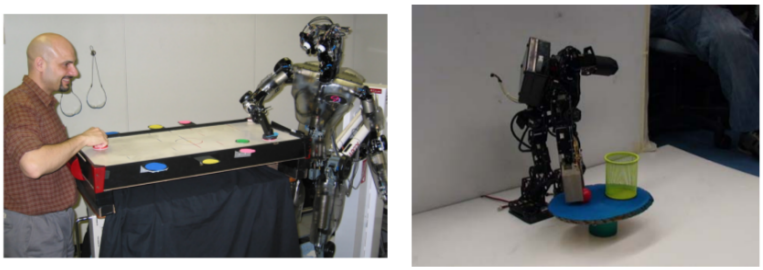

Figura 2-4: Demostración de operaciones por una persona. a)[2], b)[3]

2.2. Técnicas de visión de máquina

El uso de técnicas de visión de máquina permite adquirir información del entorno, utilizando diferentes tipos de cámaras. En particular, en el caso del Aprendizaje por Demostración, si las operaciones son demostradas por una persona o por otro robot, diferente al aprendiz, es necesario equipar al sistema con una o varias cámaras. A partir de lo anterior, y como se presenta también los pasos 3 y 4 de la sección 2.1, se requiere diseñar e implementar un sistema de visión de máquina que permita adquirir información del entorno en el cual se ejecutan las demostraciones. En la Figura **2-5** se presenta un diagrama que describe los componentes más frecuentes de un sistema de visión de máquina, como lo son: ***1)*** Una o varias cámaras; ***2)*** Hardware para el procesamiento de las imágenes; ***3)*** Espacio de trabajo; ***4)*** Sistema de iluminación del espacio de trabajo.

En el caso en particular de las operaciones de ApD, el sistema de visión de máquina que se desarrolle, debe ser capaz de adquirir la información suficiente que le permita al aprendiz caracterizar las demostraciones ejecutadas por el maestro. De acuerdo a lo anterior, es necesario aplicar un conjunto de técnicas que permitan identificar automáticamente la posición del maestro y de los objetos disponibles en el espacio de trabajo.

El color, por ejemplo, es un atributo que permite identificar diferentes tipos de objetos, a partir de las imágenes adquiridas por una o varias cámaras. En las operaciones de ensamble ejecutadas en el proyecto de investigación, se utilizaron objetos de diferentes colores, tales como verde, azul, amarillo y rojo. Por este motivo, es importante presentar las herramientas que permiten identificar los objetos disponibles en la escena.

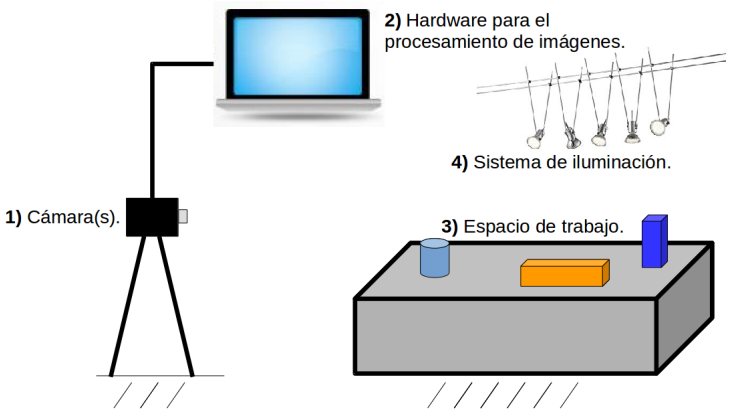

Figura 2-5: Componentes de un sistema de visión de máquina.

2.2.1. Espacio de color RGB

El espacio de color RGB (Red, Green y Blue) permite representar un color, como la combinación de 3 colores primarios, como se presenta en la Figura **2-6**. Por lo general, las cámaras utilizadas en los sistemas de visión artificial adquieren las imágenes en RGB. Sin embargo, a pesar del amplio uso que presenta por diferentes tipos de dispositivos, tales como celulares, computadores, televisores, entre otros, por lo general, no es un espacio de color que permita identificar con precisión los elementos disponibles en una imagen. Lo anterior se debe, principalmente, a la alta incidencia de la luz que afecta la información en cada píxel de la imagen. Por ejemplo, ante condiciones de luminosidad variables, un objeto de color rojo puede ser similar a uno amarillo, si las luz que incide sobre éste es muy alta. Por este motivo, surgieron otros espacios de color, tales como HSV, el cual es ampliamente usados para la identificación de objetos en una escena de acuerdo a sus características de color [18].

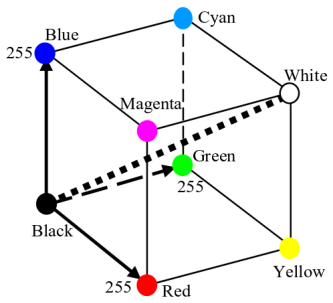

Figura 2-6: Espacio de color RGB [4].

2.2.2. Espacio de color HSV

Una transformación muy usada para la clasificación de imágenes, de acuerdo a características de color, es el espacio HSV (Hue, Saturation, Value). Como se presenta en [19, 20, 21], permite identificar objetos de acuerdo a sus características de color, con mayor eficiencia que en RGB. En la Figura **2-7** se muestra el cono que representa el espacio de color HSV. Cabe resaltar que es posible identificar los colores del espacio RGB, al variar el valor del canal H, el cual tiene un rango entre 0° y 360° .

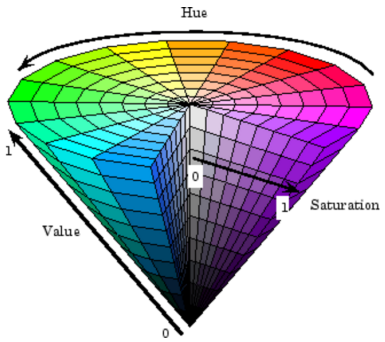

Figura 2-7: Espacio de color HSV [5].

2.2.3. Sensor de movimiento Kinect

Como se presenta en la Figura **2-4**, uno de los componentes de un sistema de visión de máquina, es la cámara, o las cámaras, a través de las cuales se adquieren las imágenes de la escena. Con el objetivo de estimar las coordenadas (x, y, z) de un objeto en el espacio, se han desarrollado dispositivos que permiten estimar información relacionada con la profundidad en una imagen. Este tipo de dispositivos se denominan RGB-D (Red, Green, Blue y Depth) [22], y han sido ampliamente usados para el procesamiento de imágenes y la clasificación de objetos, utilizando información del color (RGB) y la profundidad de una imagen (depth).

Uno de los dispositivos RGB-D más ampliamente usados es el sensor de movimiento Kinect. Así mismo, se han desarrollado sistemas que involucran PpD, en los cuales se utiliza uno o varios Kinect con el objetivo de adquirir la información suficiente de las demostraciones [9, 23, 24]. Como se presenta en [13] y [25], se han desarrollado diferentes trabajos que demuestran la validez de la información adquirida usando el Kinect, así como la precisión en la estimación de la información de profundidad de una imagen. La imagen a color obtenida, tiene resolución VGA (640 x 480 píxeles) y es adquirida utilizando una cámara RGB. Por otra parte, la imagen de profundidad es adquirida mediante un emisor infrarojo y un receptor CMOS (Semiconductor complementario de óxido metálico), que funcionan en conjunto para

estimar la profundidad de cada píxel de la imagen. Es importante resaltar que las imágenes de color y profundidad son adquiridas utilizando dos sensores diferentes, por lo que es necesario alinearlas y así asociar un valor de profundidad, para cada píxel de la imagen RGB.

A continuación, luego de alinear las imágenes RGB y de profundidad, se calculan las coordenadas (P_x, P_y) de cada pixel, de acuerdo a las Ecuaciones 2-1 y 2-2 [26]. Los términos C_x, F_x, C_y y F_y, son los parámetros intrínsecos de la cámara de profundidad, los cuales son obtenidos luego de un proceso de calibración, como el presentado en [27]. El valor P_z es calculado directamente a partir de la imagen de profundidad. Por otra parte, X y Y corresponden a las coordenadas en píxeles, del punto de interés de la imagen.

$$P_x = \frac{(X - C_x) \cdot P_z}{F_x}. \tag{2-1}$$

$$P_y = \frac{(Y - C_y) \cdot P_z}{F_y}. \tag{2-2}$$

2.2.4. Estimación de orientación

El análisis por componentes principales (PCA), es una técnica ampliamente usada para la reducción de la dimensionalidad de un conjunto de datos [28]. Así mismo, puede ser usada para estimar la orientación de un objeto en el plano, como se presenta en [29]. La base del cálculo de PCA consiste en encontrar la descomposición en valores singulares (SVD) de una matriz, o equivalentemente la descomposición en valores propios de la matriz de covarianza.

Debido a que la orientación se estima de acuerdo de imágenes, el cálculo de los componentes principales se realiza a partir de la región de interés, previamente segmentada mediante técnicas de visión de máquina. A continuación, se describe el procedimiento seguido: ***1)*** Cálculo de los vectores μ_x y μ_y (Ver Ecuación 2-3), los cuales corresponden a las coordenadas (x, y) del centro de la región de interés X, compuesta por n píxeles; ***2)*** Cálculo de las desviaciones de los píxeles de la imagen, con respecto al centro (Ver Ecuación 2-4); ***3)*** Cálculo de la matriz de covarianza C a partir de B (Ver Ecuación 2-5); ***4)*** Cálculo de la matriz V de vectores propios, que diagonaliza la matriz de covarianza C (Ver Ecuación 2-6). A partir de esto último, se obtiene la matriz diagonal D, a partir de los valores propios de la matriz C.

$$\mu_x = \frac{1}{n} \cdot \sum_{i=1}^{n} X[i, j]; \mu_y = \frac{1}{n} \cdot \sum_{j=1}^{n} X[i, j] \tag{2-3}$$

$$B = X - h \cdot u^T \tag{2-4}$$

$$C = \frac{1}{n-1} \cdot B^T \cdot B \tag{2-5}$$

$$V^{-1} \cdot C \cdot V = D \tag{2-6}$$

A partir de las etapas anteriores, se obtienen los 2 componentes principales,según la distribución de los píxeles en la región de interés. Es por esto, que el cálculo de la orientación se estima como un rotación alrededor del eje z, perpendicular a la imagen, un ángulo θ. En la Figura **2-8** se representa la estimación de la orientación de un objeto gris, identificado usando técnicas de visión de máquina.

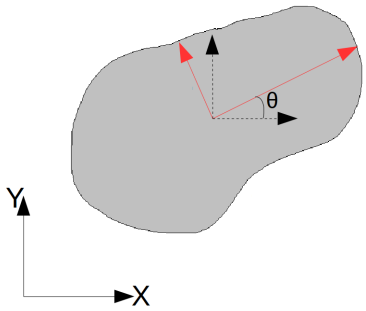

Figura 2-8: Estimación de la orientación usando PCA.

2.3. Redes de Petri

Las redes de Petri [30] son una herramienta gráfica y matemática que permite modelar secuencias de eventos discretos, como por ejemplo, las operaciones industriales [31]. Fueron propuestas por [32], y se han modelado un amplio número de aplicaciones en diferentes campos del conocimiento, como lo son: flujo de datos en computación, protocolos de comunicación, sistemas de control, multiprocesadores, entre otros. Los resultados obtenidos validan el uso de las redes de Petri, como una herramienta para representar procesos industriales, tal como lo son las operaciones de ensamble.

En general, una red de Petri está compuesta por Estados (P), Transiciones (T) y Arcos, con el fin de representar estados, acciones y las relaciones entre estos, respectivamente. Con el objetivo de describir la dinámica de una red, se representan los estados y condiciones del sistema mediante el uso de una marcación representada por tokens, los cuales permiten identificar la posibilidad de ejecutar o no, una o varias acciones [33].

A partir de lo anterior, la operación básica en las redes de Petri, y la cual permite relacionar las transiciones y arcos, se denomina "Firing" (Disparo). Esto último, permite modelar la dinámica del sistema, siguiendo las reglas [34]:

- Una transición T está habilitada, si cada estado de entrada, está marcado con al menos un token.
- Una transición habilitada puede o no dispararse.
- El disparo de una transición habilitada T elimina los tokens de cada uno de los estados de entrada, y suma un token a cada estado de salida.

La notación algebraica de las redes de Petri está compuesta por un conjunto de 5 elementos, como se presenta en la Ecuación 2-7. P es el conjunto de todos los estados p_i, T es el conjunto de todas las transiciones t_i, I es el conjunto de estados de entrada, para la transición t_i, O es el conjunto de estados de salida, para la transición t_i y μ_0 es la marcación inicial de la red. Es importante mencionar que μ_0 es un vector de tamaño k, en donde k es el número de estados de la red, y cada posición tiene uno o más tokens, que representan las condiciones iniciales del sistema.

$$C = \{P, T, I, O, \mu_0\} \tag{2-7}$$

En la Figura **2-9** se presenta una red de Petri que modela la reacción química de la molécula del agua. Se parte *a)*, en donde la red tiene la marcación inicial $\mu_0 = [2, 2]$, y el estado final en *b)*, luego de la activación de la transición t. Como se puede observar, el arco que conecta el estado H_2 y la transición t, tiene un peso de 2 unidades. Es por esto, que luego de activarse la transición t, no hay tokens disponibles en el estado H_2, y no es posible ejecutar nuevamente el disparo en en t.

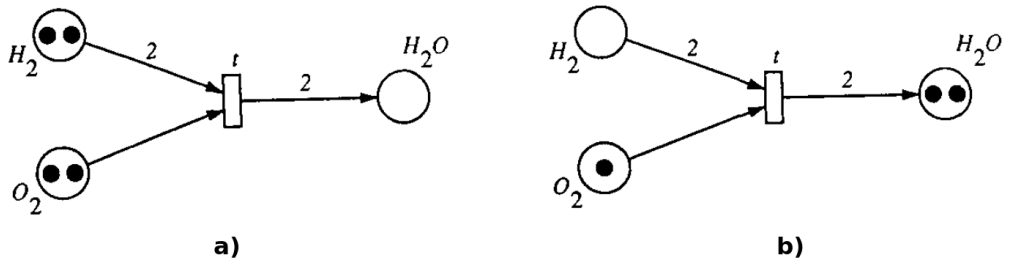

Figura 2-9: *a)* Red de Petri con una marcación inicial; *b*) Red de Petri con transición disparada. [33]

Mediante el uso de Redes de Petri es posible obtener dos funcionalidades muy importantes: ***i)*** Representar gráficamente una tarea; y ***ii)*** Calcular la alcanzabilidad a partir de una marcación inicial [34]. En particular, lo anterior permite determinar el conjunto de acciones que pueden ser ejecutadas a partir de un estado inicial del sistema. Esto último, es conocido como el plan de ensamble de la operación [35].

2.4. Modelos probabilísticos.

Los modelos probabilísticos permiten representar las variaciones y correlaciones entre dos o más variables. En el presente trabajo, los modelos relacionan las variables de posición (x, y, z) en cada instante de tiempo t de las demostraciones. El uso de modelos de mezcla de Gaussianas (GMM) permite caracterizar las diferentes partes que componen un movimiento. A continuación se describen los modelos GMM, GMR y TP-GMM.

2.4.1. Gaussian Mixture Model (GMM).

La representación probabilística de las demostraciones, las cuales son adquiridas mediante el sistema de visión de máquina, permite estimar las variaciones y correlaciones entre las diferentes variables de la información adquirida. La combinación de múltiples distribuciones Gaussianas se denomina GMM y representa una mezcla de K componentes. Se define como una función de probabilidad dada por la Ecuación 2-8, en donde ξ_j es un punto de la trayectoria, π_k es la probabilidad *a priori* y $\mathcal{N}(\xi_j; \mu_k; \Sigma_k)$ es la distribución de probabilidad Gaussiana [36]. Con respecto a los parámetros de la distribución, μ_k y Σ_k son el centro de la Gaussiana y la matriz de covarianza, respectivamente.

$$p(\xi_j) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\xi_j; \mu_k; \Sigma_k). \tag{2-8}$$

A partir de la Ecuación 2-8, se observa que la probabilidad del punto ξ_j depende de π_k y de los parámetros $(\xi_j; \mu_k; \Sigma_k)$, para cada componente Gaussiana k. El cálculo de los parámetros del modelo para cada distribución se realiza usando el metodo iterativo de Expectation Maximization (EM) [37], implementado por [36].

2.4.2. Gaussian Mixture Regression (GMR).

A partir del modelo GMM, generado usando un conjunto de demostraciones, es posible reconstruir las trayectorias utilizando GMR. Al contrario de lo que ocurre al calcular el modelo GMM, en donde se conocen todos los puntos ξ_j de la trayectoria, en cada instante de tiempo t, en el caso de GMR el objetivo es estimar todos los puntos $\hat{\xi_s}$ de la trayectoria.

Por lo tanto, se requiere separar los componentes espaciales y temporales que representan las entradas y salidas, de cada modelo GMM. De esta manera es posible separar μ_k y Σ_k de cada componente Gaussiana k (ver Ecuación 2-9).

$$\mu_k = \{\mu_{t,k}, \mu_{s,k}\} \quad \Sigma_k = \begin{pmatrix} \Sigma_{t,k} & \Sigma_{ts,k} \\ \Sigma_{st,k} & \Sigma_{s,k} \end{pmatrix} \tag{2-9}$$

Para cada componente Gaussiano k, los elementos de la matriz de covarianza Σ_k se calculan como se presenta en la Ecuación 2-10. El componente β_k está dado por le Ecuación 2-11.

$$\hat{\xi}_s = \sum_{i=1}^{K} \beta_k \hat{\xi}_{s,k} \quad \hat{\Sigma_s} = \sum_{i=1}^{K} \beta_k^2 \hat{\Sigma}_{s,k} \tag{2-10}$$

$$\beta_k = \frac{p(\xi_t|k)}{\sum_{i=1}^{K} p(\xi_t|i)}. \tag{2-11}$$

A partir de las ecuaciones 2-10 y 2-11, se evalúan los términos $\left\{\hat{\xi_{s,k}}, \hat{\Sigma_{s,k}}\right\}$, para cada componente Gaussiano k de la trayectoria s, en diferentes instantes de tiempo ξ_t, con el fin de estimar una forma generalizada del movimiento $\hat{\xi} = \{\xi_t, \xi_s\}$ y la matriz de covarianza asociada [36].

2.4.3. TP-GMM.

El modelo de Task Parameterized GMM (TP-GMM por sus siglas en inglés), es una técnica descrita por [1], cuyo objetivo principal es generar trayectorias en función de un conjunto de parámetros que caracterizan la tarea. Los parámetros de la tarea están representados por P sistemas de coordenadas, definidos en cada instante de tiempo t por $\{b_{t,j}, \boldsymbol{A}_{t,j}\}_{j=1}^{P}$, los cuales representan el orígen del observador ($b_{t,j}$) y la matriz de transformación ($\boldsymbol{A}_{t,j}$).

A partir de lo anterior, cada demostración está compuesta por $\xi \in \mathbb{R}^{D \times N}$, en donde D es la dimensionalidad de la trayectoria adquirida y N es el número de demostraciones de la operación, vista desde diferentes puntos de observación. Cada una de las muestras ($\mathbb{X}_t$) puede ser adquirida con sensores ubicados en los diferentes marcos de referencia j, o calculadas mediante la Ecuación 2-12.

$$\boldsymbol{X}_t^{(j)} = \boldsymbol{A}_{t,j}^{(-1)}(\xi_t - \boldsymbol{b}_{t,j}). \tag{2-12}$$

El aprendizaje de los parámetros se hace mediante el algoritmo EM, restringido por el hecho que las demostraciones adquiridas desde diferentes marcos de referencia, corresponden a una

única demostración observada desde puntos distintos. A partir del modelo TP-GMM aprendido, es posible calcular nuevas trayectorias modificando la posición y/u orientación de los diferentes parámetros de la tarea. Un nuevo modelo GMM, con parámetros $\left\{\pi_i, \hat{\xi}_{t,i}, \hat{\Sigma}_{t,i}\right\}_{i=1}^{K}$, se calcula mediante las ecuaciones 2-13 y 2-14, con el fin de generar las trayectorias, dados unos nuevos parámetros de la tarea.

$$\mathcal{N}(\hat{\xi}_{t,i}, \hat{\Sigma}_{t,i}) \propto \prod_{i=1}^{P} \mathcal{N}(\hat{\xi}^{j}_{t,i}, \hat{\Sigma}^{(j)}_{t,i}. \tag{2-13}$$

$$\hat{\xi}^{(j)}_{t,i} = \boldsymbol{A}_{t,j}\mu^{(j)}_{i} + \boldsymbol{b}_{t,j} \quad ; \quad \hat{\Sigma}^{(j)}_{t,i} = \boldsymbol{A}_{t,j}\Sigma i^{(j)}\boldsymbol{A}^{T}_{t,j} \ . \tag{2-14}$$

Resumen del capítulo

En este Capítulo se presenta la fundamentación teórica de los conceptos más importantes usados en el desarrollo del proyecto de investigación, como lo son: Aprendizaje por Demostración, modelos probabilísticos como GMM, GMR y TP-GMM, técnicas de visión de máquina y preprocesamiento de las demostraciones adquiridas. La programación de robots industriales es un proceso dispendioso, por lo que han surgido metodologías de programación novedosas, tal como lo es el la PpD. Estas técnicas buscan imitar el aprendizaje de los seres vivos, los cuales aprenden a ejecutar nuevas tareas, luego de que estas son demostradas por otro individuo. Al aplicar ApD en manipuladores robóticos, se requiere equipar al sistema con una o varias cámaras, para adquirir la información relevante del entorno mediante técnicas de visión de máquina. A partir de lo anterior, se describe el sensor de movimiento Kinect, el cual permite adquirir información RGB-D del entorno, el cual fue usado en el sistema de visión de máquina propuesto. Luego, se describen las redes de Petri, las cuales permiten representar gráficamente una tarea y calcular las acciones que pueden ser ejecutadas, de acuerdo a las condiciones iniciales del sistema. Finalmente, se describen los modelo probabilísticos GMM, GMR y TP-GMM, los cuales son entrenados a partir de las demostraciones adquiridas visualmente para la generación automática de nuevas trayectorias. En el siguiente Capítulo, se presenta el sistema propuesto para la ejecución de tareas de manipulación de bloques en el plano (2 dimensiones), usando técnicas de ApD y visión de máquina, con manipuladores robóticos.

3 Sistema para la ejecución de operaciones de manipulación en el plano (2D).

El desarrollo de un sistema que ejecute automáticamente operaciones de ensamble con manipuladores robóticos industriales es muy útil, en especial si la etapa de programación está asociada a un proceso de aprendizaje automático, como lo es el Aprendizaje por Demostración (ApD) [1, 9, 8, 3, 38, 39, 40, 41, 42]. Para tal fin, como primer acercamiento se desarrolló e implementó un sistema que ejecuta operaciones de manipulación de bloques en en el plano, usando técnicas de Programación por Demostración (PpD) para un robot industrial. Luego, a partir de los resultados obtenidos, se extendió y complementó la metodología propuesta para la ejecución de operaciones de ensamble de un carro y de una botella, en espacios tridimensionales.

La revisión bibliográfica desarrollada permite establecer la importancia de seleccionar un maestro y, adicionalmente, de diseñar un sistema capaz de adquirir la información relevante de las demostraciones. En el presente trabajo, se implementó un sistema de visión de máquina capaz de identificar la posición de la mano del maestro. Cabe resaltar que en el caso de las operaciones en dos dimensiones, la persona que ejecuta las demostraciones está equipada con un guante, con el objetivo de facilitar la identificación visual de la posición de la mano; mientras que en la operación en 3D, el sistema de visión propuesto identifica la mano del maestro sin guante. La información adquirida de las demostraciones es filtrada y normalizada, y a continuación, se entrenan y validan los modelos TP-GMM. Luego, se evalúa el funcionamiento de los modelos entrenados y se generan nuevas operaciones automáticamente, en función de los parámetros de la tarea estimados mediante el sistema de visión de máquina. Finalmente, las trayectorias son implementadas en un robot ABB IRB 140, disponible en el laboratorio, fuera de línea.

A continuación, se presentan las etapas más importantes de la metodología propuesta para las tareas de manipulación de bloques:

1. Ejecución de las demostraciones de la tarea a ser aprendida por parte del maestro.

2. Adquisición de las demostraciones y preprocesamiento de las trayectorias adquiridas,

utilizando el sistema de visión de máquina.

3. Segmentación de las demostraciones adquiridas, entrenamiento y validación de los modelos probabilísticos.

4. Simulación de la operación usando RobotStudio e implementación en un robot ABB.

5. Implementación en los robots reales de las trayectorias simuladas en RobotStudio.

En las secciones que se presentan a continuación, se describe la metodología seguida para la implementación del sistema propuesto para la ejecución de tareas de manipulación con bloques en un plano. Para comenzar, se describen los algoritmos para la adquisición de las demostraciones y de los objetos disponibles en el espacio de trabajo, a través del sistema de visión desarollado. Luego, se presenta el método de segmentación de las demostraciones implementado, el cual usa el concepto de velocidad cero para la separación de tareas [43]. Finalmente, se presentan los modelos probabilísticos entrenados, para la ejecución de operaciones de manipulación de uno y dos bloques.

3.1. Sistema de visión de máquina.

Se implementó un sistema de visión de máquina para adquirir las demostraciones y para estimar la posición y orientación de los objetos en el espacio de trabajo. Se utilizó un sensor de movimiento Kinect de primera generación y un guante equipado con marcas circulares. A continuación se describen las características más importantes del sistema desarrollado.

3.1.1. Adquisición de las demostraciones.

La etapa de adquisición de los movimientos por demostración es importante debido a que el modelo probabilístico es generado a partir de las trayectorias descritas por el maestro. Se utilizaron las imágenes RGB capturadas, y las demostraciones se realizaron con un guante equipado con marcas circulares, como se presenta en la Figura **3-1**. Las demostraciones fueron ejecutas en un ambiente con condiciones de iluminación semicontroladas, bajo la incidencia de luz natural.

En la Figura **3-2** se presenta un diagrama del algoritmo seguido con el fin de calcular las coordenadas (x, y) de las trayectorias demostradas; cabe mencionar que el proceso descrito se realiza con cada fotograma que es capturado en intervalos de 0.2 segundos, con la cámara RGB del Kinect. Se utilizó la librería de código abierto OpenCV para el procesamiento de imágenes.

Cada una de las 4 etapas de la Figura **3-2**, está compuesta por un conjunto de operaciones realizadas con el fin de obtener la trayectoria del movimiento por demostración realizado.

Figura 3-1: Guante equipado con marcas circulares.

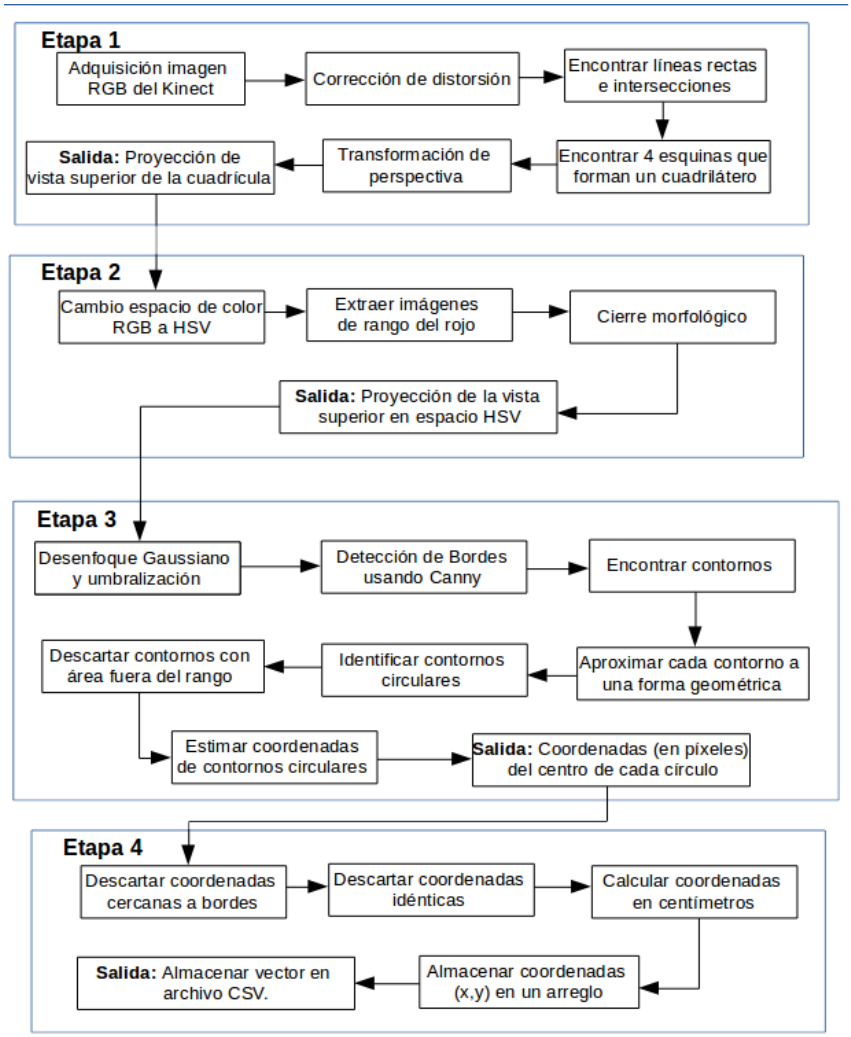

Figura 3-2: Diagrama de bloques del sistema de adquisición de las demostraciones.

Por lo tanto, es un proceso secuencial, que toma como entrada del sistema la imagen RGB adquirida, y produce como salida las coordenadas (x, y) en centímetros del marcador circular,

en cada instante de tiempo durante la trayectoria.

Proyección vista superior de la cuadrícula en HSV.

Inicialmente se adquiere la imagen RGB del Kinect con resolución de 640 × 480 píxeles, usando las librerías de código abierto OpenCV y OpenNI. Se corrigen los parámetros de distorsión intrínsecos de la cámara y posteriormente se encuentra el cuadrilátero que representa el espacio de trabajo sobre el cual se realizarán las demostraciones. Los lados del cuadrilátero son encontrados utilizando la transformación de Hough. En la Ecuación 3-1 se presenta la forma general de una recta en coordenadas polares (ρ, θ). Para un punto (x_a, y_a) en la imagen, la distancia entre el orígen y el punto está dada por los parámetros ρ y θ, representada como una función sinusoidal en coordenadas polares. A partir de las funciones sinusoidales, en el plano polar, que describen cada punto en la imagen, es posible identificar las rectas presentes en la imagen (RGB). El problema de identificación de líneas rectas en la imagen se transforma en la identificación de las curvas sinusoidales concurrentes en el plano polar.

$$\rho = x_a \cdot cos(\theta) + y_a \cdot sin(\theta). \tag{3-1}$$

Posteriormente, se realiza un conjunto de operaciones morfológicas y transformaciones geométricas, con el fin de obtener la proyección de la vista superior de la cuadrícula. Esta operación se denomina transformación de perspectiva. A partir de la imagen anterior, se transforma el espacio de color RGB a HSV, frecuentemente utilizado para segmentar imágenes de acuerdo a características de color [19]. La conversión se realiza mediante la estimación de los valores intermedios que se presentan en las ecuaciones 3-2, 3-3 y 3-4.

$$R' = R/255; G' = G/255; B' = B/255. \tag{3-2}$$

$$C_{max} = max(R', G', B'); C_{min} = min(R', G', B'). \tag{3-3}$$

$$\Delta = C_{max} - C_{min}. \tag{3-4}$$

El cálculo de los valores de Hue, Saturation y Value se realiza mediante las ecuaciones 3-5, 3-6 y 3-7.

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \cdot \frac{G'-B'}{\Delta} mod(6) & C_{max} = R' \\ 60^\circ \cdot \frac{B'-R'}{\Delta} + 2) & C_{max} = G' \\ 60^\circ \cdot \frac{R'-G'}{\Delta} + 4) & C_{max} = B' \end{cases}. \tag{3-5}$$

$$S = \begin{cases} 0 & C_{max} = 0 \\ \frac{\Delta}{C_{max}} & C_{max} \neq 0 \end{cases}. \tag{3-6}$$

$$V = C_{max}. \tag{3-7}$$

A continuación se extrae el plano R del espacio HSV, con el fin de identificar el rango de valores de rojo que representa el color del marcador. La salida del algoritmo es la imagen cuyos píxeles están contenidos en los rangos descritos que representan las diferentes tonalidades del marcador, debido a la incidencia de la luz. En la Figura **3-3** se presenta la imagen adquirida por el Kinect, la proyección de la cuadrícula y la identificación de los dos bloques. Esto último se repite para la identificación de la marca amarilla y del guante.

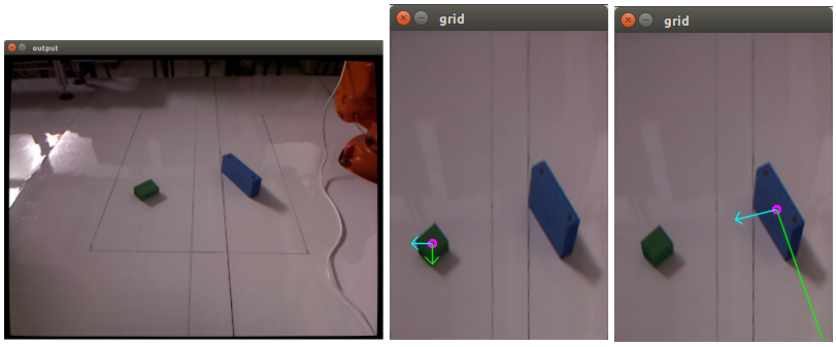

Figura 3-3: Proceso de identificación de los bloques.

Estimación de coordenadas de la demostración.

A partir de la salida obtenida en la Etapa 2, se implementó la Etapa 3, la cual tiene como salida las coordenadas (x, y) del centro de la marca circular del guante. Como primer paso, se implementó un filtro Gaussiano de la imagen (Ecuación 3-8), seguido de la detección de bordes utilizando el algoritmo de Canny. El operador de detección de bordes retorna un valor de la primera derivada en la dirección horizontal G_x y en la dirección vertical G_y. Con esto se calcula el gradiente y la dirección del borde (ecuaciones 3-9 y 3-10).

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{3-8}$$

$$G = \sqrt{G_x^2 + G_y^2}. \tag{3-9}$$

$$\theta = atan(G_y, G_x). \tag{3-10}$$

Posteriormente, se encuentran los contornos candidatos a ser marcas y se seleccionan los que son circulares. Finalmente, se estiman las coordenadas en píxeles del centro de cada marca, las cuales son tomadas como el parámetro de entrada a la última etapa.

Evaluación de las coordenadas estimadas.

De acuerdo a las coordenadas de los píxeles del centro del círculo, en la Etapa 4 se desarrolló el algoritmo para determinar cuales puntos cumplen los criterios establecidos para ser almacenados en la trayectoria, como por ejemplo:

- Cercanía con los bordes de la cuadrícula.
- Mismo valor de coordenadas (x, y) en el instante de tiempo t_i, en comparación con t_{i-1}.

Luego, se realiza la transformación de las coordenadas de píxeles a centímetros, debido a que las dimensiones de la cuadrícula que representa el espacio de trabajo de las demostraciones son conocidas. En la ecuaciones 3-11 y 3-12 se presenta el cálculo realizado para determinar las coordenadas en centímetros a partir del valor en píxeles; O_x y O_y son las coordenadas en píxeles, W_p y H_p es el ancho y el alto en píxeles de la proyección de la vista superior de la cuadrícula, W representa el ancho (730 [mm]) y H la altura de la misma (1030 [mm]). Finalmente, se almacenan las coordenadas que describen cada trayectoria demostrada en un archivo de texto.

$$P_x = \frac{O_x W}{W_p}. \tag{3-11}$$

$$P_y = \frac{O_y H}{H_p}. \tag{3-12}$$

A partir de las etapas descritas, se adquirieron las demostraciones ejecutadas por una persona, y se identificaron las piezas que estaban dispuestas en el espacio de trabajo con la cuadrícula de fondo. Posteriormente, se realizó una etapa de postprocesamiento a las demostraciones adquiridas mediante el sistema de visión, debido a la alta incidencia de ruido y a que en algunos instantes en particular, no se identificaba correctamente la posición de la marca del guante. A continuación se describen los pasos incluidos para mejorar la calidad de las demostraciones adquiridas, tomando como punto de inicio el archivo de texto ya generado:

1. **Oclusiones:** En la Figura **3-1** se presenta el guante equipado con 3 marcas. Se realizó así con el fin de disminuir la posibilidad de pérdida de la referencia en la adquisición visual de la demostración. En caso que se pierda la referencia de las 3 (marcas) en el mismo instante de tiempo, se tienen almacenadas las últimas coordenadas (x, y) válidas. Las coordenadas (x, y) mantienen el mismo valor hasta que se identifique nuevamente la posición de al menos una marca del guante. Las demostraciones fueron ejecutadas a baja velocidad.

2. **Filtro promedio móvil (FPM):** Se utilizó para disminuir la variabilidad de los datos, con $n = 5$, usando la Ecuación 3-13, en donde x es la trayectoria adquirida mendiante el sistema de visión, t es el instante de tiempo y n es el tamaño de la ventana. A partir de lo anterior, se observa que FPM permite calcular dinámicamente el promedio de una señal x en un instante de tiempo t, a partir de los n valores anteriores.

$$x_f[t] = \frac{1}{n}\sum_{k=0}^{n-1} x[t-k]. \tag{3-13}$$

3. **Eliminación datos:** Las coordenadas (x, y) cuyo valor haya variado más del 20 % con respecto al valor anterior, son descartadas. Se implementó con el fin de evitar saltos en los valores de las coordenadas de la trayectoria inducidas por problemas en la segmentación.

4. **Ajuste de tiempos:** Debido a que se eliminaron algunos datos en la trayectoria, se ajusta la secuencia de tiempo. Para este caso en particular, las demostraciones se ejecutaron a velocidad constante. El sensor usado captura imágenes a una frecuencia de 60 fotogramas por segundo.

Operaciones adquiridas visualmente.

Se ejecutaron 2 tipos de tareas de manipulación: con un bloque y con dos bloques. La primera, corresponde a la operación presentada en la Figura **3-4** y consiste en tomar un bloque desde diferentes posiciones iniciales (PI) y transportarlo hasta una posición final establecida (PF). La segunda operación, se presenta en la Figura **3-5** y consiste en mover 2 bloques, siguiendo los movimientos que se presentan a continuación:

1. Mover el robot desde la posición inicial PI del efector final, hasta la posición inicial A del bloque 1.

2. Mover el bloque 1 hasta la posición final B.

3. Mover el efector final vacío hasta la posición inicial C, del bloque 2.

4. Mover el bloque 2 hasta la posición final D.

La estimación de las posiciones iniciales de los objetos en ambas tareas, se calculó utilizando el sistema de visión descrito.

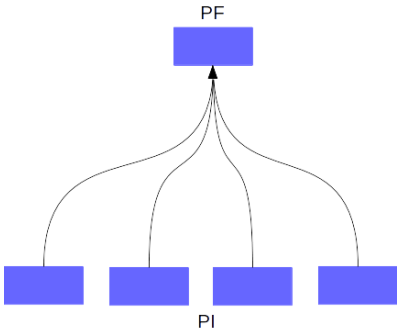

Figura 3-4: Operación con un bloque.

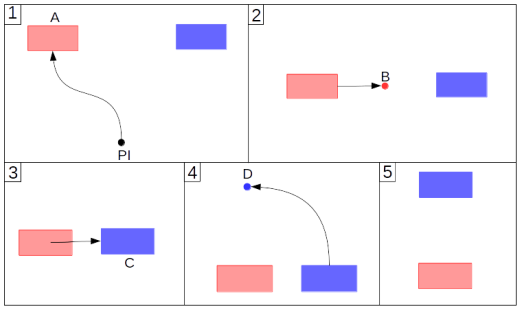

Figura 3-5: Operación con dos bloques.

Las demostraciones fueron ejecutadas en un laboratorio, con condiciones de iluminación semicontroladas. Adicionalmente, los movimientos del maestro fueron ejecutados sobre una superficie blanca demarcado con un tablero con una cuadrícula de 5 cm. de lado.

3.1.2. Estimación de la posición de los objetos.

El método para estimar la posición y orientación de los objetos en el espacio de trabajo, parte de la salida obtenida en la primera etapa del sistema de adquisición de las demostraciones, a la cual se le realizó la transformación al espacio de color HSV y, posteriormente, la proyección de la vista superior de la cuadrícula. A continuación se describen los pasos más importantes en la estimación de la posición y orientación de los objetos:

1. Identificación de colores y preprocesamiento: Debido a que los colores de los bloques a identificar son conocidos, se encontraron los rangos de valores en el espacio HSV correspondientes. A continuación se aplicó desenfoque Gaussiano y umbralización, seguido de la detección de contornos.

2. Clasificación de contornos: Cada contorno se aproximó a un polígono para encontrar los objetos con diferentes geometrías en el espacio de trabajo.

3. Estimación de la posición y la orientación: La posición se calculó a partir del centroide de cada contorno y la orientación mediante PCA (Análisis de Componentes Principales). Los resultados obtenidos mostraron un error entre (0.13 y 1.5) [cm] en posición y de 10° en orientación.

A partir de lo anterior, se estima la posición (x, y) y la orientación (en grados) de cada bloque en el espacio de trabajo. En la Figura **3-6** se presentan los valores estimados (de posición y orientación), para un conjunto de piezas. En el caso anterior, las ruedas (*wheels*, en inglés) son los objetos amarillos y por su forma circular, no se estima la orientación; mientras que para el chasis (*chassis*, en inglés) y la cabina (*cabin*, en inglés), se estimó la posición y orientación. Los resultados obtenidos en esta etapa, son utilizados como los parámetros de la tarea para las diferentes operaciones ejecutadas.

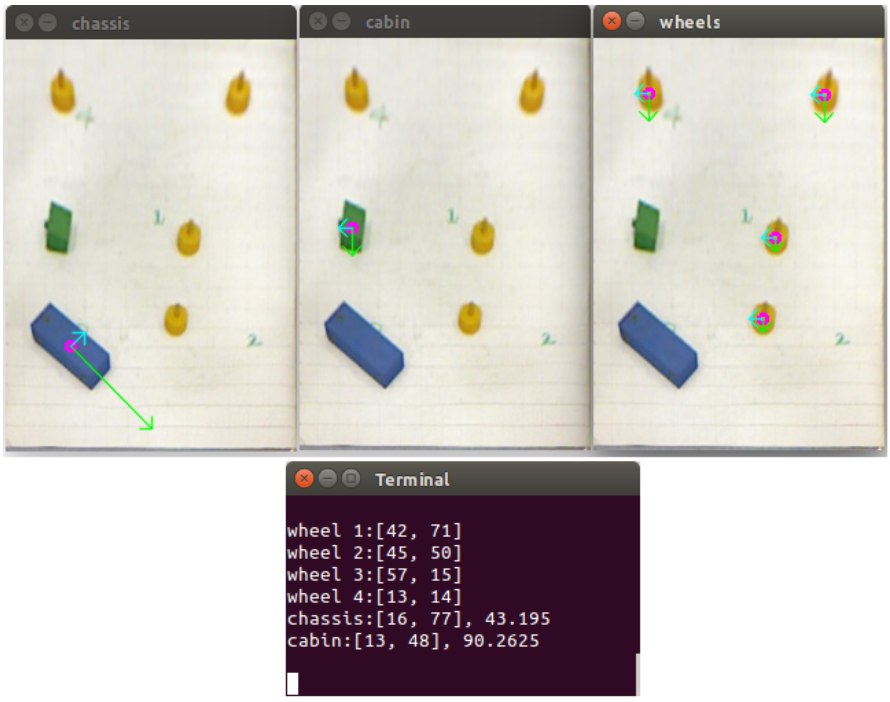

Figura 3-6: Ejemplo de resultados obtenidos al estimar posición y orientación de las piezas.

3.2. Segmentación de las demostraciones.

Se implementó un algoritmo de segmentación para las trayectorias adquiridas, con el fin de identificar las subtareas que componen la operación, tomando como criterio de separación

los instantes de tiempo con velocidades cercanas a cero. Este procedimiento es basado en el trabajo presentado por [43] y [24], el cual está compuesto por las siguientes etapas: ***i)*** Cálculo del modelo GMM de la demostración que mejor describe la operación; ***ii)*** Identificación del valor umbral que permite separar la demostración en subtareas simples; y ***iii)*** Segmentación de cada una de las demostraciones adquiridas, en subtareas. A continuación se presenta la metodología y algunos de los resultados obtenidos en el proceso.

Modelo GMM de cada demostración.

La primera etapa consistió en determinar la demostración que mejor describe las subtareas y la cual sería utilizada como plantilla; se identificaron gráficamente los intervalos de tiempo (t_i, t_f) de cada subtarea (en cada demostración), a partir de las trayectorias adquiridas visualmente. Por lo general, los valores de los intervalos de tiempo de las subtareas son similares en todas las demostraciones. En el caso en particular de la tarea con dos bloques, los intervalos de tiempo de cada subtarea se determinaron a partir de las coordenadas (x, y) en las posiciones A, B, C y D, presentadas en la Figura **3-5**.

En la Figura **3-8** se presentan las gráficas de los componentes (x, y) de posición de una de las demostraciones adquiridas de la operación con dos bloques. Adicionalmente, se calculó la suma del cuadrado de los componentes de la velocidad. En esta segunda gráfica, se observan los instantes de tiempo con velocidad cercana a cero, los cuales serán tomados como criterio de separación de la trayectoria para la identificación de las subtareas.

Posteriormente, se calculan los parámetros π_k, μ_k, Σ_k y el número de Gaussianas K que describen el modelo GMM de cada trayectoria (ver Ecuación 2-8). En este caso en particular, cada demostración contiene 3 variables que describen el movimiento: (x, y, t). En donde (x, y) representan los componentes del movimiento adquirido y t representa el tiempo. Es importante mencionar que es un proceso iterativo, en el cual la trayectoria seleccionada y el número de Gaussianas que describen la operación, varía en función de los resultados que se obtengan en las siguientes etapas. Sin embargo, para la primera iteración, se seleccionó como la demostración ejemplo la que gráficamente representaba todas las subtareas satisfactoriamente.

Identificación del umbral

A partir de la demostración seleccionada en el paso anterior, la segunda etapa consiste en identificar el valor del umbral mínimo que se tomará como criterio de separación de las subtareas (Ver Figura **3-7**). El procedimiento se presenta a continuación:

1. Velocidades al cuadrado: A partir de los componentes (x, y) de la trayectoria seleccionada, se calcularon las velocidades al cuadrado de la demostración.

2. Filtrado de velocidades: Se usó un FPM (ver Ecuación 3-13) con $n = 3$, para suavizar las velocidades al cuadrado. Posteriormente se sumaron los componentes (x, y) filtrados.

3. Normalización de velocidades: Se normalizó la suma de las velocidades para garantizar que todos los valores estuvieran en el rango $\{0, 1\}$.

4. Filtrado de velocidad normalizada: Nuevamente se filtra usando FPM, con $n = 2$.

5. Identificación del valor umbral: A partir de los pasos anteriores, se obtiene la gráfica de suma de velocidades al cuadrado que se presenta en la Figura **3-7**. Se identifica gráficamente el valor del umbral ϑ que permite separar las subtareas.

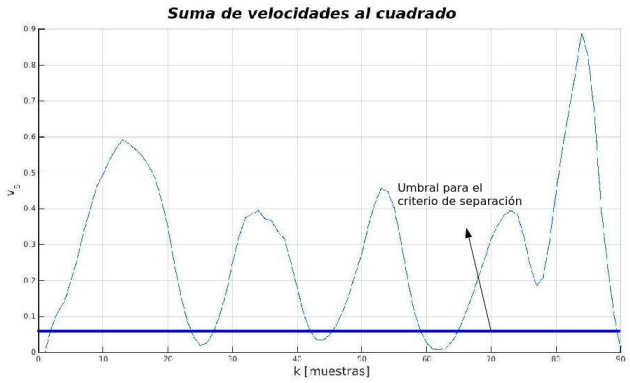

Figura 3-7: Segmentación usando criterio de velocidad cero.

Separación de cada subtarea

A partir del modelo GMM obtenido para cada demostración, se genera nuevamente la trayectoria usando GMR. A continuación se siguió una metodología similar a la desarrollada para la identificación del umbral de velocidad, tomado como criterio de separación:

1. Filtrado de trayectoria: Se usó un FPM, con $n = 4$ para suavizar la trayectoria.

2. Velocidades al cuadrado: Se calculó la velocidad por componentes y la suma al cuadrado.

3. Normalización: Se normalizó la suma de las velocidades y se filtró usando un FPM, con $n = 2$.

4. Umbralización de velocidades: A partir del valor umbral encontrado en la etapa anterior, se asigna $v = 0$ para todos los instantes de tiempo t cuya velocidad sea inferior a ϑ. En caso contrario, se asigna $v = 1$.

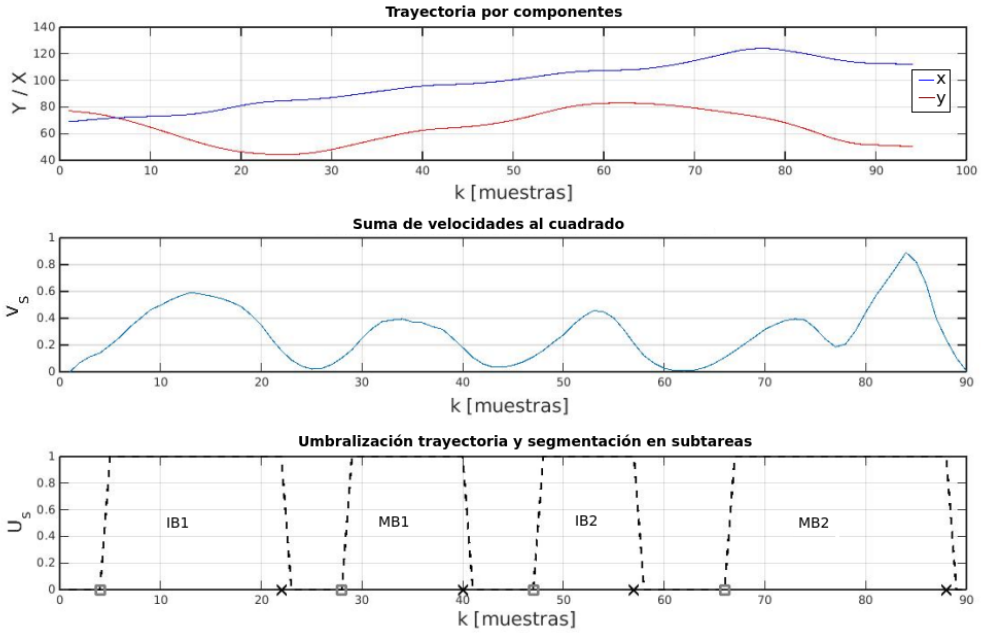

Figura 3-8: Segmentación de la trayectoria en subtareas.

5. Identificación de flancos: Se calcularon los flancos de subida y los flancos de bajada de la trayectoria. Para los primeros, se tomó como criterio que si $v(i) = 0$ y $v(i+1) = 1$, entonces es un flanco de subida. En el caso de los flancos de bajada, se identificaron cuando $v(i) = 1$ y $v(i+1) = 0$.

6. Interpolación de cada subtarea: Se calculó una interpolación cúbica para cada subtarea, con el fin de normalizar a 100 el número de elementos de cada una de las subtareas.

A partir de lo anterior, se encuentran las subtareas en cada demostración según los intervalos contenidos entre un flanco de subida y el siguiente flanco de bajada. De esta manera, se realizó la descomposición de la trayectoria en operaciones más simples. Posteriormente se compara cada una de las subtareas encontradas en cada demostración, con respecto a la demostración seleccionada como ejemplo en el primer paso. La técnica usada se denomina Dynamic Time Warping (DTW, por sus siglas en inglés), la cual permite evaluar la similitud entre dos vectores. Finalmente, cada subtarea en cada demostración se etiqueta con el nombre de la subtarea con la que presenta mayor similitud con respecto a la trayectoria seleccionada como ejemplo [24, 44]. En la tercera gráfica de la Figura **3-8** se presenta la segmentación obtenida para una de las demostraciones de la operación con dos bloques.

3.3. Entrenamiento de modelos TP-GMM.

Se realizaron operaciones con uno y dos bloques. La primera consistió en tomar el bloque desde una posición inicial estimada mediante el sistema de visión y moverlo hasta una posición final fija. La segunda consistió en manipular los bloques con el fin de alinearlos horizontal o verticalmente en un plano. Con relación al espacio de trabajo, las demostraciones fueron ejecutadas utilizando un fondo blanco con una cuadrícula compuesta por cuadrados de 5 $[cm]$ de lado, con el fin de facilitar la medición de las distancias así como de mejorar el posicionamiento de los objetos con respecto a un sistema de coordenadas conocido.

A continuación se describe el proceso seguido para cada operación, a partir de las demostraciones adquiridas, hasta el entrenamiento de los modelos aprendidos de ApD que representan las tarea. Se entrenó el modelo GMM y luego TP-GMM con el fin de generar nuevas trayectorias a partir del modelo aprendido de las demostraciones. Se usó inicialmente la librería desarrollada por [45], disponible en MatLab, y se adoptó para ejecutarla en C++. Esto último, con el fin de integrar la etapa de generación de trayectorias con el sistema de visión artificial.

3.3.1. Operación con un bloque.

Se adquirieron en total 10 demostraciones de la operación, sin embargo, tras realizar la etapa de postprocesamiento de las trayectorias, se seleccionaron las 5 que mejor describían la tarea, como se presenta en la Figura **3-9**. Fue necesario realizar la selección de las demostraciones, debido a que el sistema de visión, en algunos momentos, perdía el seguimiento de la marca del guante por las condiciones de iluminación no controladas, y consecuentemente, la trayectoria generada no representaba correctamente la operación. El punto de inicio está representado por una marca circular (o) y el punto final por una cruz (x).

Debido a que únicamente se manipula un bloque, no fue necesario segmentar la operación en subtareas, ya que los parámetros de la tarea requeridos para generar la trayectoria son la posición inicial del bloque (estimada a través del sistema descrito), y la posición final fija.

A partir de las demostraciones presentadas en la Figura **3-9** se entrena el modelo TP-GMM, variando el número de Gaussianas que describen cada trayectoria. En la Figura **3-10** se presentan los resultados obtenidos al representar cada trayectoria con 2, 3, 4 y 5 Gaussianas, con el fin de determinar el número (de Gaussianas) que describe satisfactoriamente las demostraciones realizadas. Se utilizó GMR para generar nuevamente las demostraciones adquiridas y evaluar el funcionamiento del modelo. En la Figura **3-10**, la trayectoria generada modificando los parámetros de la tarea se presenta en color rojo; las otras trayectorias representan las reproducciones de las demostraciones generadas mediante GMR.

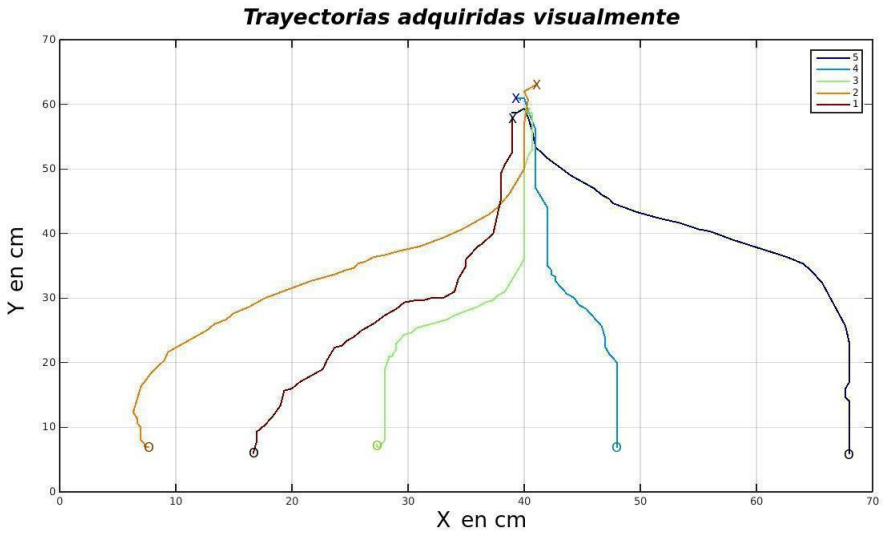

Figura 3-9: Demostraciones adquiridas de la operación movimiento de un bloque.

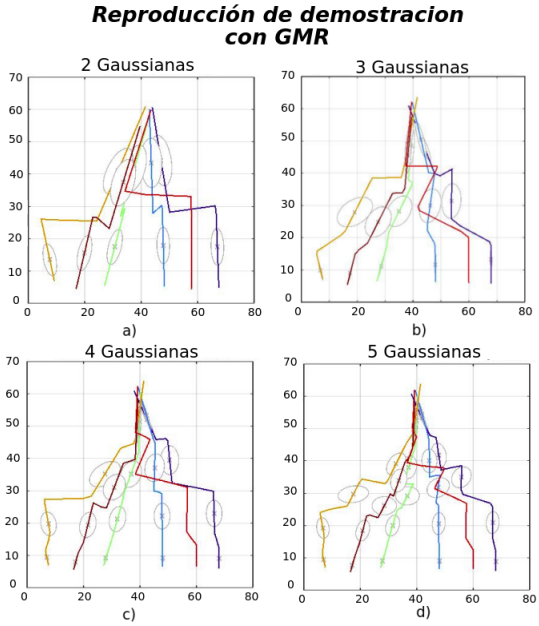

Figura 3-10: Trayectorias generadas modificando número de Gaussianas.

Como se observa en la Figura **3-10**, las trayectorias descritas con 3 Gaussianas permiten obtener un movimiento similar de las demostraciones, por lo cual se considera que usando 3 Gaussianas, la generalización del modelo es satisfactoria. Adicionalmente, se evaluó la exactitud de los movimientos comparando las trayectorias generadas con respecto a las trayectorias adquiridas. En la Figura **3-11** se presenta el error cuadrático medio de posición entre la trayectoria demostrada y la trayectoria se genera nuevamente usando GMR. El error

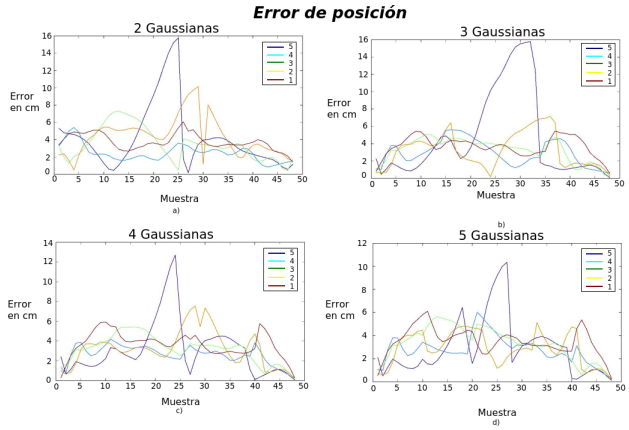

Figura 3-11: Error de las demostraciones generadas por GMR.

se calculó mediante la norma de los componentes (x, y), calculados directamente a partir de los datos.

Como se puede evidenciar, la trayectoria de color azul oscuro (ver Figura **3-11**), es un caso crítico en especial en la región intermedia del movimiento, ya que se alcanzan errores de hasta 16 [cm] al describir el modelo con 2 Gaussianas. El modelo descrito con 5 Gaussianas tiene el menor error de posición, alrededor de 10 [cm]. Es importante resaltar que los errores anteriores, corresponden a la desviación entre la trayectoria generada usando GMR y la demostración adquirida visualmente, en el transcurso de la trayectoria. En todos los casos, el error de posición al inicio y al final de la trayectoria es cercano a 0 [cm].

Parámetros de la tarea.

En la operación con un bloque, los parámetros que describen la tarea son: la posición inicial del bloque, estimada mediante el sistema de visión descrito (ver Figura **3-6**), y la posición final hasta la cual se moverá. Los parámetros están representados de la forma (x_{ini}, y_{ini}) y (x_{fin}, y_{fin}).

3.3.2. Operación con dos bloques.

Se adquirieron 12 demostraciones y nuevamente se seleccionaron las 5 que describían satisfactoriamente la tarea presentada en la Figura **3-5**. La operación consistió en mover 2 bloques desde unas posiciones iniciales, hasta unas posiciones finales en las cuales un bloque se ubica al lado del otro. Se integró la metodología descrita para segmentar cada una de las trayectorias demostradas utilizando el criterio de velocidad cero. En la Figura **3-12** se presentan las demostraciones adquiridas, distribuidas a lo largo del plano de trabajo del robot

con diferentes posiciones (iniciales y finales) y orientaciones.

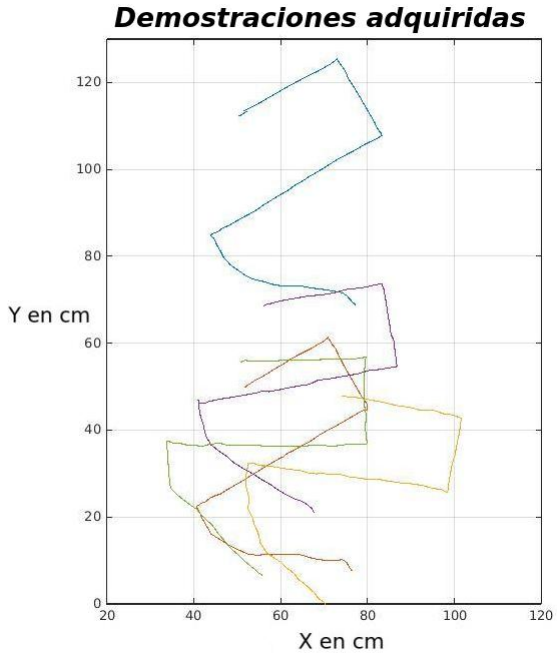

Figura 3-12: Demostraciones de la tarea con 2 bloques.

La operación con dos bloques está compuesta por 4 subtareas, las cuales son identificadas a través de la segmentación implementada. Las subtareas que componen la operación son:

1. **Ir a bloque 1 ("IB1"):** Movimiento desde la posición inicial donde está el gripper hasta la posición donde está el bloque 1.

2. **Mover bloque 1 ("MB1"):** Se toma el bloque 1 y se mueve en la dirección X, hasta que llega a la posición final del bloque donde es soltado.

3. **Ir a bloque 2 ("IB2"):** Se mueve el robot con el gripper vacío hasta la posición donde está el bloque 2.

4. **Mover bloque 2 ("MB2"):** Se toma el bloque 2 y se transporta hasta la posición final donde se suelta el objeto. Finaliza el movimiento.

En la Figura **3-8** se presentan 3 gráficas que representan la trayectoria por componentes adquirida visualmente, la suma de las velocidades al cuadrado y la separación y umbralización de la operación en 4 subtareas, respectivamente. El procedimiento seguido se describió en la sección anterior de segmentación de las demostraciones, y el objetivo de esta etapa fue separar la operación en las 4 subtareas más simples que la componen.

De acuerdo a la representación de la operación (Figura **3-5**), los parámetros de la tarea necesarios para generar el movimiento son las posiciones iniciales de los bloques, cada una representada como (x, y). Se calculó el modelo TP-GMM para cada una de las subtareas, con el fin de generar la operación completa como la concatenación de las 4 trayectorias de cada subtarea. A continuación se presenta el método seguido para calcular los parámetros de la tarea.

Cálculo de los parámetros de la tarea.

Para la generación de trayectorias usando TP-GMM, se requiere identificar los parámetros que describen la tarea. En el caso de la operación con 2 bloques, se estima la posición inicial de cada uno de estos (ver Figura **3-6**) y se calculan las posiciones finales a partir de las coordenadas y la orientación de las diferentes piezas. En la Figura **3-13** se presentan 4 ejemplos de la tarea en donde se calcularon las posiciones finales de los bloques 1 y 2, a partir de la posición y orientación inicial de éstos. Los parámetros de la tarea para cada subtarea corresponden a las posiciones iniciales y finales de los bloques.

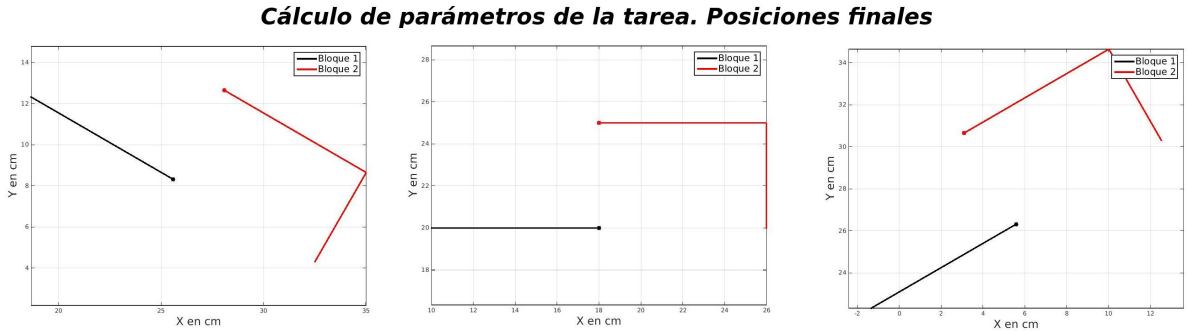

Figura 3-13: Cálculo de los parámetros de la tarea. Operación con 2 bloques.

En la Figura **3-13**, la posición inicial está representada como un punto (.) y la posición final como un asterisco (*). Así mismo, la línea azul indica el punto inicial y el punto final del movimiento del bloque azul, y la línea roja lo hace para el bloque verde (Ver Figura **3-6**).

Modelo TP-GMM de cada subtarea.

Con el fin de modelar la tarea de los 2 bloques, se segmentó cada demostración siguiendo el método anteriormente descrito, en las 4 subtareas mencionadas, y se generó un modelo TP-GMM para cada una de estas. En la Figura **3-14** se presentan las 4 subtareas que componen la trayectoria.

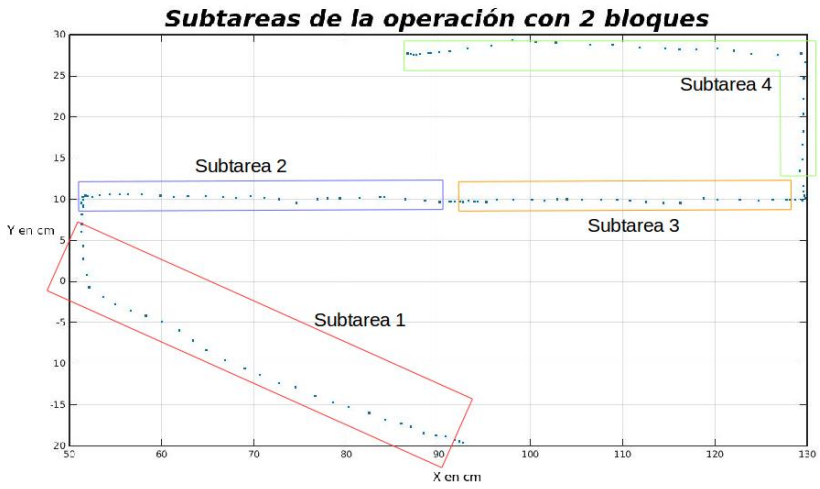

Figura 3-14: Subtareas de la operación con 2 bloques.

Resumen del capítulo

En el presente Capítulo se describió la metodología propuesta para la ejecución de operaciones de manipulación de bloques en un espacio bidimensional, el cual se desarrolló como una etapa previa al desarrollo del sistema de ensamble en 3 dimensiones. Se implementaron técnicas de ApD para la manipulación de uno y dos bloques. La metodología propuesta está compuesta principalmente por las siguientes etapas, y es similar para las tareas de ensamble que se presentará más adelante: ***i)*** Ejecución de las demostraciones por parte del maestro, el cual es una persona equipada con un guante con marcas circulares en los dedos; ***ii)*** Adquisición de las demostraciones mediante un sistema de visión de máquina compuesto por un sensor de movimiento Kinect; ***iii)*** Filtrado y normalización de la información adquirida en la etapa anterior, seguido de la segmentación para identificar las subtareas que componen cada demostración; ***iv)*** Entrenamiento del modelo TP-GMM para cada tarea y validación a través de GMR; y ***v)*** Generación de nuevas trayectorias a partir de los modelos probabilísticos ya entrenados. En el siguiente Capítulo, se presenta el sistema propuesto para la ejecución de operaciones de ensamble usando técnicas de ApD y visión de máquina, con manipuladores robóticos en un espacio tridimensional.

4 Sistema para la ejecución de operaciones de ensamble en 3D.

En el Capítulo 3 se presentó la metodología para la ejecución de operaciones en un espacio bidimensional. En este capítulo se describe la metodología que permite la integración de una técnica de ApD, que utiliza el modelo probabilístico TP-GMM (Task Parameterized Gaussian Mixture Model) y redes de Petri, para la programación de operaciones de ensamble. Lo anterior, debido al elevado número de aplicaciones industriales que requieren ejecutar tareas de ensamble con manipuladores robóticos, y las cuales presentan, por lo general, un elevado nivel de complejidad en la etapa de programación. De manera análoga que en las tareas 2D, el sistema de visión de máquina consta de un sensor de movimiento Kinect y las pruebas fueron implementadas en un robot industrial ABB IRB 140. Se demuestra que el uso de técnicas de ApD permite reducir la complejidad de la programación de robots, en particular, para la ejecución de operaciones industriales como lo son las tareas de ensamble.

4.1. Sistema para la generación de trayectorias.

El sistema propuesto para la generación de trayectorias, está compuesto por tres etapas. La primera, corresponde a la adquisición de las demostraciones ejecutadas por el maestro y tiene como salida la secuencia de puntos coordenados (x, y, z) de la trayectoria; la segunda, corresponde al entrenamiento del modelo probabilístico TP-GMM; y la tercera, corresponde a la generación de nuevas trayectorias, a partir de las posiciones de los objetos en el espacio de trabajo.

4.1.1. Adquisición de demostraciones.

En la Figura **4-1** se presenta el diagrama de bloques de la primera etapa, la cual tiene como objetivo adquirir visualmente las trayectorias, usando un sensor Kinect, y estimar la secuencia de coordenadas (x, y, z) de la mano, durante la ejecución de cada demostración. La salida de esta etapa es un archivo de texto en el cual está almacenado cada demostración adquirida.

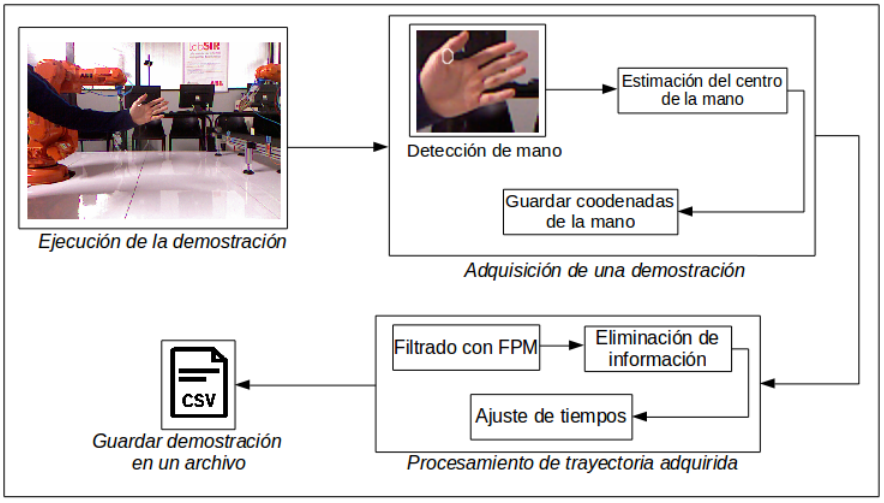

Figura 4-1: Diagrama etapa de adquisición demostraciones.

Ejecución de las demostraciones.

En el trabajo desarrollado, las demostraciones de las operaciones fueron ejecutadas por una persona y adquiridas visualmente utilizando un sensor de movimiento Kinect. La metodología seguida se implementó en dos aplicaciones: la primera, el ensamble de un carro como el presentado en la Figura **4-2**, el cual está compuesto por un chasis (azul), una cabina (verde) y cuatro ruedas (amarillas); y la segunda, el colocado de la tapa de una botella. Las demostraciones fueron ejecutadas por una persona, y adquiridas mediante un algoritmo de reconocimiento de la mano, a través del cual se estiman las coordenadas de posición (x_t, y_t, z_t) del centro de la mano, en todo instante de tiempo t. Se alinearon las imágenes de color y profundidad usando las funcionalidades de las librerías OpenNI y OpenCV.

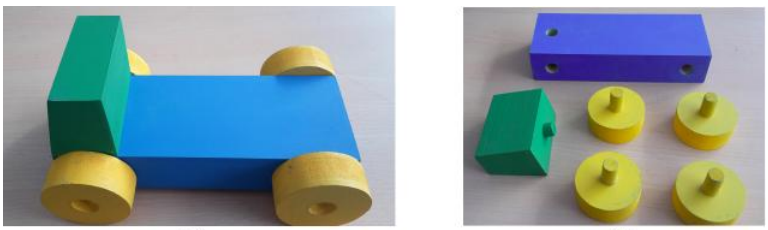

Figura 4-2: a) Carro ensamblado. b) Carro con piezas separadas.

En la Figura **4-3** se presenta una secuencia de imágenes que corresponde a la operación de ensamble de la cabina (verde) al chasis (azul), ejecutada por una persona. Como se puede observar, las demostraciones fueron adquiridas en un laboratorio, en el cual las condiciones de iluminación y de fondo están medianamente controladas. Por otra parte, en la Figura **4-4**

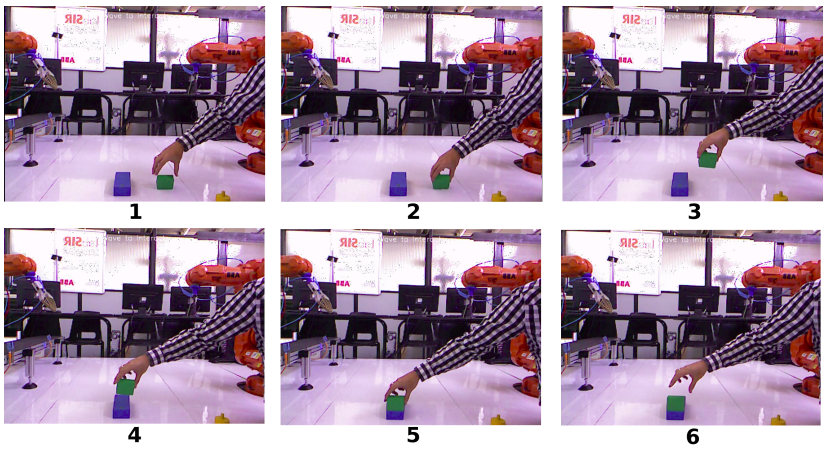

Figura 4-3: Secuencia de demostración del ensamble entre la cabina y el chasis.

se presenta la secuencia del colocado de la tapa de la botella.

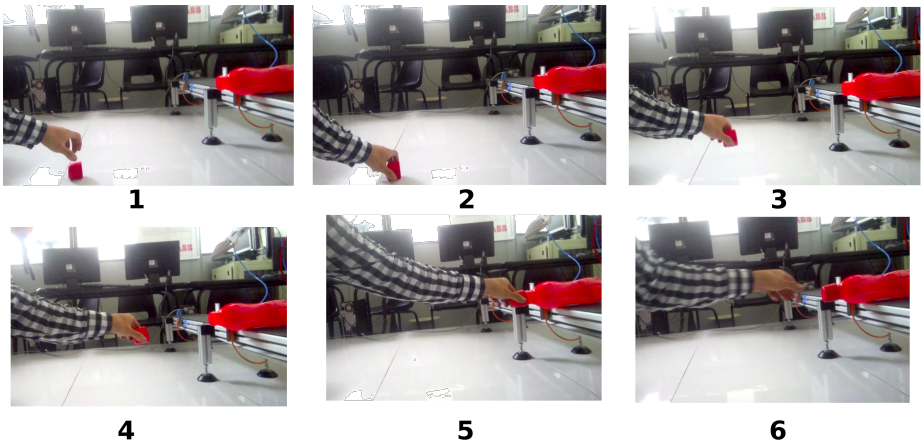

Figura 4-4: Secuencia de demostración de ensamble entre la tapa y la botella.

Reconocimiento de la mano.

Las demostraciones fueron adquiridas utilizando el sensor Kinect, mediante una etapa de procesamiento de imágenes, el cual está compuesto por los 3 bloques (Ver Figura **4-5**).

A partir de la demostración de la operación adquirida, se almacenan las coordenadas (x, y, z) de la mano durante la ejecución de la trayectoria. Debido a factores externos como la incidencia de la luz, las oclusiones y la constante estimación del centro de la mano, se presenta un elevado contenido de ruido en la trayectoria adquirida. Así mismo, cada demostración adquirida puede tener un número de muestras diferente, como consecuencia de la variación

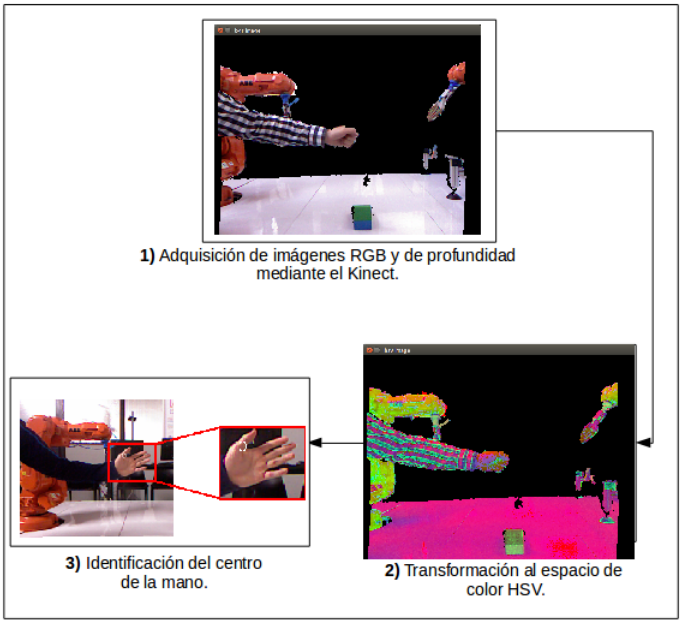

Figura 4-5: Diagrama detección de la mano.

en los tiempos de ejecución de la tarea, por parte del maestro. Con el fin de reducir el ruido, se agregó una etapa de filtrado y normalización de los datos.

Preprocesamiento y normalización de trayectorias.

A cada demostración adquirida se le aplicó una etapa de preprocesamiento y normalización, con el fin de garantizar homogeneidad y suavidad en el movimiento. A continuación se mencionan las etapas del proceso:

1. **Filtro promedio móvil (FPM):** Se utilizó para disminuir la variabilidad de los datos, generada por las oclusiones de la mano debido a los objetos de la operación de ensamble y a la iluminación. A partir de la Ecuación 3-13, en donde $x_f[t]$ corresponde a la señal filtrada en el instante t, n es el número de muestras usadas para calcular el promedio y $x[t]$ es la señal original. Se encontró experimentalmente que con $n = 5$, se obtiene un suavizado satisfactorio de las demostraciones [46]. El procedimiento es similar para el cálculo de y_f y z_f.

2. **Eliminación datos:** Se eliminan los puntos con variaciones mayores al 20 % en el instante $[t]$, con respecto al instante anterior $[t-1]$. Se implementa esta etapa debido a que, en algunos instantes de tiempo, la segmentación no es correcta y se identifica la posición de la mano en una región diferente a la verdadera. Este proceso produjo buenos resultados, debido a que las demostraciones fueron ejecutadas a velocidad constante.

A partir de los resultados obtenidos en los pasos anteriores, se implementó un interpolador cúbico y se normalizó cada demostración a 200 muestras [47]. Esta etapa se agregó con el fin de tener las demostraciones con igual número de datos. En la Figura **4-6** se presenta una de las demostraciones adquiridas y procesadas usando las etapas ya descritas. La gráfica naranja corresponde a la trayectoria adquirida y la azul es la trayectoria luego de las etapas de preprocesamiento y normalización descritas. Finalmente, cada demostración es almacenada en un archivo de texto diferente, los cuales son usados para el entrenamiento del modelo TP-GMM.

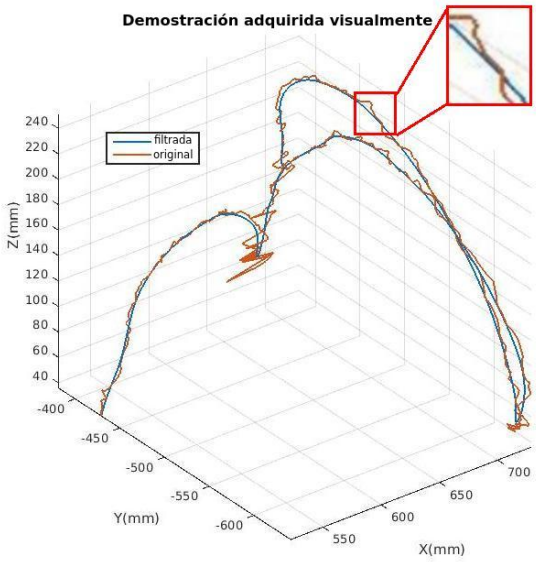

Figura 4-6: Filtrado y normalización de trayectoria.

4.1.2. Obtención de las subtareas y modelos TP-GMM.

La etapa de entrenamiento está compuesta por 2 modulos: ***i)*** Segmentación de las demostraciones, y ***ii)*** Entrenamiento modelo TP-GMM. Luego de finalizar el entrenamiento, se obtuvieron los modelos probabilísticos para la generación de nuevas trayectorias de acuerdo a la posición de los objetos en la escena.

Segmentación de las demostraciones

A continuación se presenta la metodología seguida, así como algunos de los resultados obtenidos en el proceso de segmentación de las demostraciones de la operación con el carro. En el trabajo desarrollado, se realizaron dos operaciones: el ensamble de un carro compuesto por 6 piezas: 1 chasis, 1 cabina y 4 ruedas, como se presenta en la Figura **4-2**, y el colocado de la tapa de una botella. Sin embargo, únicamente se implementó la etapa de segmentación para la operación del carro, la cual está compuesta por 3 subtareas simples: ***1)*** Fijar cabina

al chasis (fijar cabina); ***2)*** Ir a la rueda y tomarla (Ir rueda); ***3)*** Fijar la rueda al chasis (Fijar rueda). Es importante mencionar que no se implementó la etapa de segmentación en subtareas para la operación con la botella, debido a que la tarea está compuesta de una sola subtarea: "Colocar tapa de la botella".

1. **Demostración seleccionada como plantilla:** La primera etapa consiste en determinar la demostración que mejor describe las tarea, la cual es utilizada como plantilla de ejemplo para la segmentación. Para esto, se evalúan las trayectorias filtradas y normalizadas, y se selecciona la que gráficamente presenta instantes de velocidad cercanos a 0. En la Figura **4-7** se presentan los componentes v_x, v_y, v_z, así como su suma al cuadrado normalizada, de una de las demostraciones. Adicionalmente, se presentan con un asterisco el instante de tiempo en el que inicia cada subtarea, y con un triangulo, el tiempo en que finaliza. A partir de los instantes t ya identificados, se asignan los intervalos de tiempo $[t_i, t_f]$ (Tiempo inicial y Tiempo final), para cada una de las subtareas.

2. **Identificación del umbral:** A partir de la demostración seleccionada, la segunda etapa consiste en identificar el valor del umbral que se tomará como criterio de separación de las subtareas. El procedimiento seguido se presenta a continuación: ***i)*** Velocidades al cuadrado: A partir de los componentes (x, y, z) de las demostraciones adquiridas, se calculan los componentes de velocidad al cuadrado (v_x^2, v_y^2, v_z^2); ***ii)*** Filtrado de velocidades: Se usa un FPM con $n = 3$, para suavizar cada componente de la velocidad al cuadrado. A continuación, se suman los componentes filtrados; ***iii)*** Normalización de velocidades: Se normaliza la suma de las velocidades al cuadrado, para garantizar que todos los valores esten en el rango $[0, 1]$, en donde 1 es la velocidad máxima y 0 es el valor mínimo de la demostración; ***iv)*** Filtrado de velocidad normalizada: Nuevamente se filtra usando FPM, con $n = 2$; ***v)*** Identificación del valor umbral: A partir de los pasos anteriores, se obtiene la gráfica de suma de velocidades al cuadrado que se presenta en la Figura **4-7**. Se identifica gráficamente el valor del umbral ϑ que permite separar las subtareas, de modo que cuando la suma de las velocidades esté por debajo del valor ϑ, se asigna $v = 0$, y, en caso contrario, $v = 1$; ***vi)*** Identificación de flancos: Se calculan los flancos de subida y los flancos de bajada de la trayectoria. Para los primeros, se toma como criterio que si $v(i) = 0$ y $v(i + 1) = 1$, entonces es un flanco de subida. En el caso de los flancos de bajada, se identifican cuando $v(i) = 1$ y $v(i + 1) = 0$; ***vii)*** Interpolación de cada subtarea: Se usa una interpolación cúbica para cada subtarea, con el fin de normalizar a 100 el número de elementos de cada una de las subtareas. En la Figura **4-8** se presenta una de las trayectorias adquiridas, luego de interpolar y normalizar a 100 elementos cada subtarea.

Se encuentran las subtareas en cada demostración a partir de los intervalos contenidos entre un flanco de subida y el siguiente flanco de bajada. De esta manera, se realiza la descomposición de la trayectoria en operaciones más simples. En la Figura **4-9** se presentan los

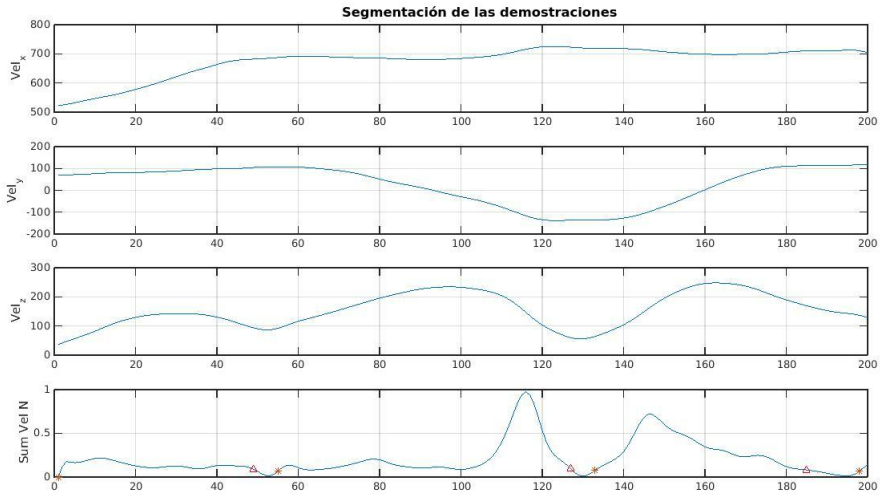

Figura 4-7: Trayectoria plantilla y t_i ,t_f por subtarea.

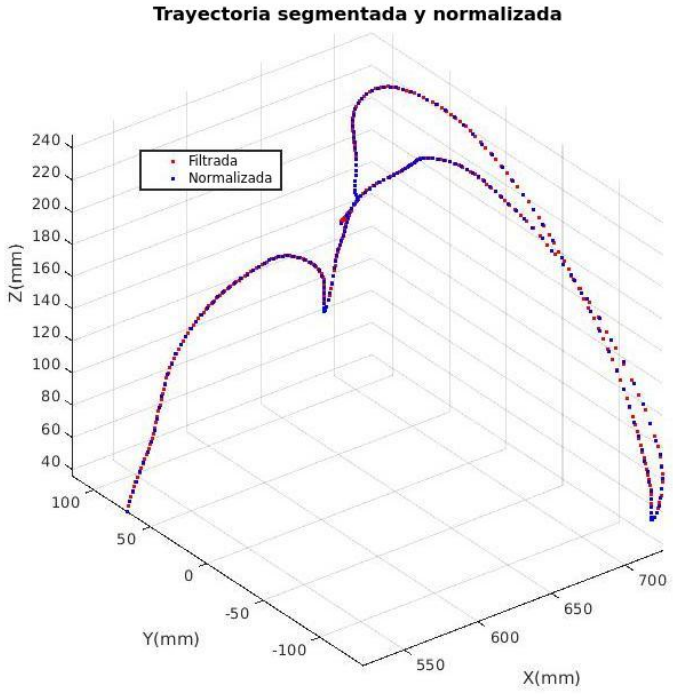

Figura 4-8: Trayectoria segmentada y normalizada a 100 elementos por subtarea.

resultados de segmentación obtenidos para una demostración, así como el punto de inicio t_i (asterisco) y el punto final t_f (triángulo) de cada subtarea (ver Fig. **4-7**).

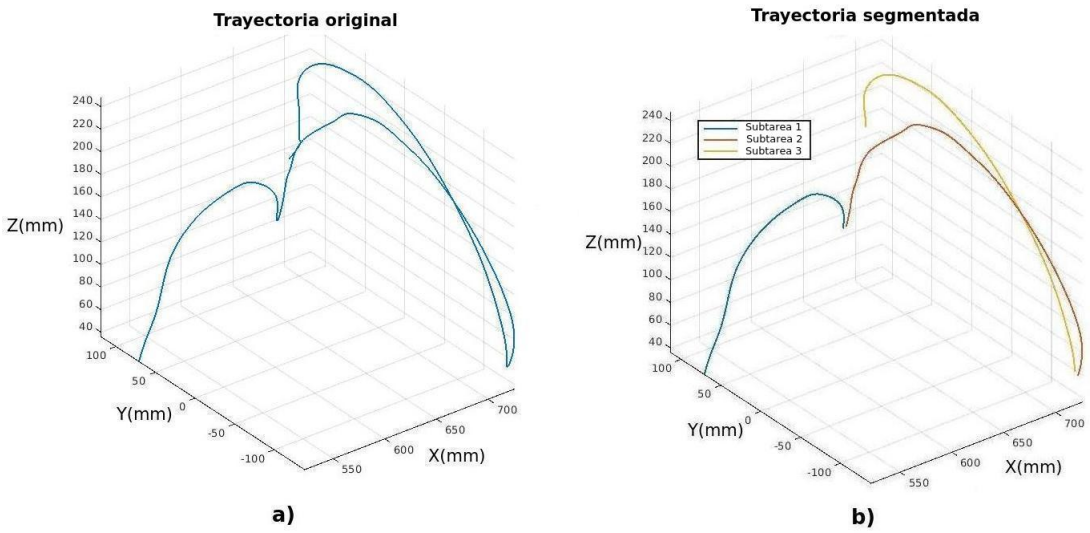

Figura 4-9: Segmentación de una demostración.

Entrenamiento del modelo TP-GMM

Luego de segmentar las demostraciones adquiridas visualmente, la siguiente etapa consiste en entrenar un modelo TP-GMM para cada una de las subtareas. Uno de los puntos que presenta mayor dificultad, debido a que las operaciones de ensamble son ejecutadas en un espacio tridimensional, es encontrar el número de Gaussianas que modelan las demostraciones. En la Figura **4-10** se presenta la proyección en el plano YZ de la reproducción de las demostraciones, usando GMR, y las Gaussianas que componen cada trayectoria. Cada subtarea se modela con 4 Gaussianas, debido a la naturaleza de los movimientos, los cuales no presentan una complejidad mayor que pueda requerir un número mayor.

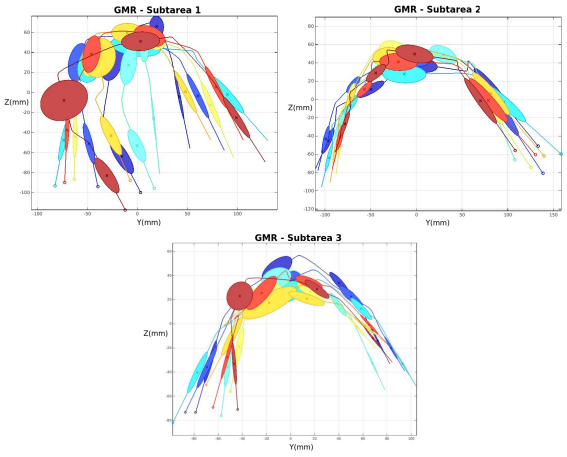

Figura 4-10: Proyección plano YZ de Modelo TP-GMM de cada subtarea.

Reproducción de demostraciones usando GMR.

Para evaluar el número de Gaussianas que mejor describe la tarea, se usa un sistema como el presentado en la Figura **4-11** y se generan los modelos TP-GMM de cada subtarea con 3, 4, 5 y 6 Gaussianas. A continuación, se reprodujeron las demostraciones usadas en el entrenamiento, con el modelo aprendido. Como resultado, se calculó el error cuadrático medio entre la demostración usada en el entrenamiento y la trayectoria generada usando el modelo TP-GMM. Se encontró que al incrementar el número de Gaussianas que describen la trayectoria por encima de 4, no se presenta una mejora representativa al reproducir las demostraciones, mientras que el costo computacional si aumentaba notablemente. En la Tabla **4-1** se presenta el tiempo de entrenamiento en segundos y el error RMS promedio al variar el número de Gaussianas del modelo TP-GMM de la subtarea 1. El procedimiento seguido para la selección del número de Gaussianas para las subtareas 2 y 3 fue análogo al presentado en la Tabla **4-1**. A partir de lo anterior, se determinó usar 4 Gaussianas para representar cada subtarea.

# Gaussianas.	**T. entrenamiento(s.).**	**Error RMS (cm).**
3	14	34,6
4	21	31,2
5	33	30,8
6	49	30,4

Tabla 4-1: Error entre valores reales y valores estimados por visión.

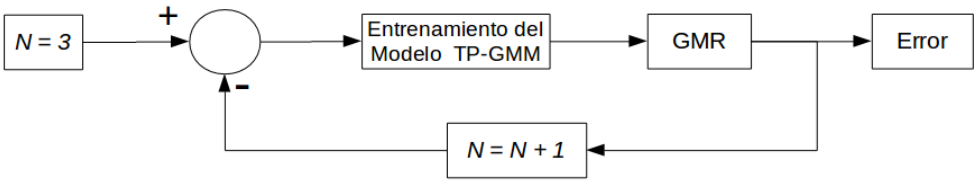

Figura 4-11: Diagrama etapa de entrenamiento modelos TP-GMM.

En la Figura **4-12** se presentan las demostraciones de la subtarea y las reproducciones usando GMR a partir del modelo aprendido. Como se puede observar, las trayectorias generadas usando el modelo probabilístico presentan un comportamiento similar a las ejecutadas por el maestro. Los resultados obtenidos son similares para las subtareas 2 y 3.

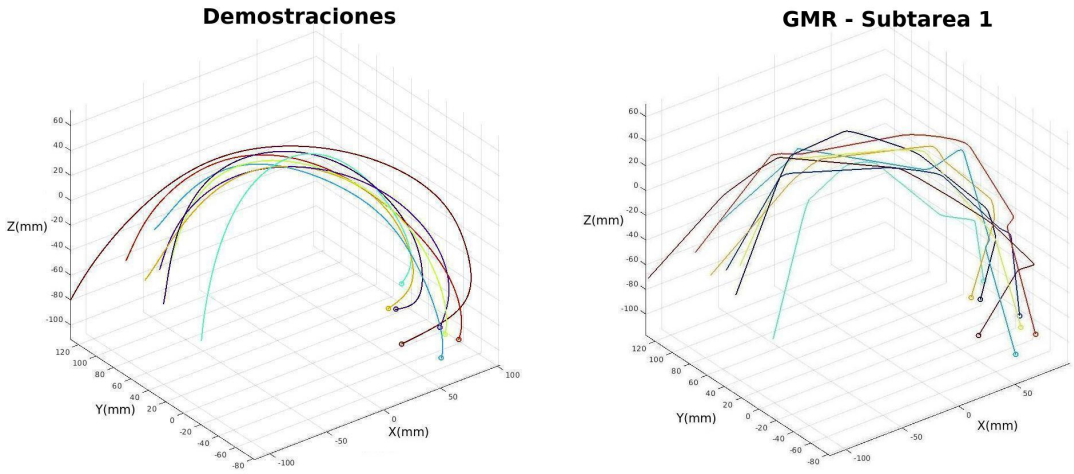

Figura 4-12: Reproducción de demostraciones usando GMR.

4.1.3. Ejecución de nuevas trayectorias.

En la Figura **4-13** se presenta un diagrama de la etapa de ejecución de nuevas trayectorias, a partir de los modelo TP-GMM aprendidos y presentados en secciones anteriores. Como se puede observar, es fundamental identificar los parámetros que describen la tarea, por lo que se describirá con mayor detalle la metodología seguida y los resultados obtenidos para su estimación.

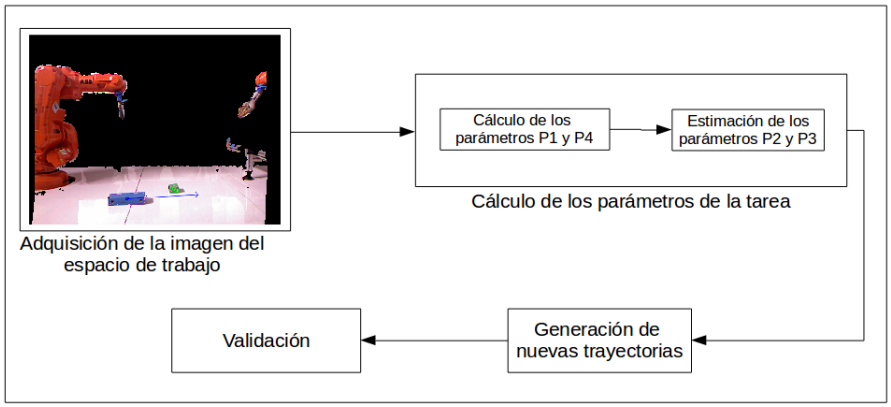

Figura 4-13: Diagrama etapa de ejecución de nuevas trayectorias.

Adquisición de imágenes usando sensor Kinect.

El método para estimar la posición y orientación de los objetos en el espacio de trabajo, parte de la imagen RGB, adquirida a través del Kinect. Dicha imagen se transforma al espacio de

color HSV, y a través de las siguientes etapas, se estima la posición y orientación de cada pieza:

1. Identificación de colores y preprocesamiento: Dado que los colores de las piezas son conocidos, se encuentran los rangos de valores en el espacio HSV correspondientes. A continuación se aplica un filtro Gaussiano (Gaussian blur), seguidamente se umbraliza y, finalmente, se detectan los contornos.

2. Clasificación de contornos: Cada contorno se aproxima a un polígono para encontrar los objetos con diferentes geometrías en el espacio de trabajo.

3. Alineación de imágenes: Se integraron las librerías OpenNI y OpenCV para alinear las imágenes de profundidad y color, adquiridas mediante el Kinect.

4. Cálculo de la posición: Las coordenadas (x, y, z) se calculan usando la imagen de profundidad y las ecuaciones 4-1, 4-3 y 4-2.

5. Cálculo de la orientación: Se estima la orientación de la pieza usando PCA (Principal Component Analysis, en inglés). Los resultados obtenidos mostraron un error máximo de 1.5 $cm.$ en posición y 10° en orientación, los cuales están dentro del rango permitido por las restricciones mecánicas del gripper fijado al robot.

Para el cálculo de las coordenadas (x, y), se parte del valor de z (en metros), el cual es estimado usando la imagen de profundidad. Cabe resaltar que las coordenadas estimadas por visión, están referenciadas con respecto al centro del Kinect. De las ecuaciones 4-2 y 4-3, las variables c_{xd}, c_{yd}, f_{xd} y f_{yd} son los parámetros intrínsecos de la cámara, calculados en la etapa de corrección de la distorsión del lente. Adicionalmente, $rgb[a][b]$ y $depth[a][b]$, corresponden al valor del píxel con coordenadas (a, b), de la imagen a color y de profundidad, respectivamente.

$$z = depth[a][b]. \tag{4-1}$$

$$x = (rgb[a][b] - c_{xd}) \cdot \frac{z}{f_{xd}}. \tag{4-2}$$

$$y = (rgb[a][b] - c_{yd}) \cdot \frac{z}{f_{yd}}. \tag{4-3}$$

Con esto, se estima la posición (x, y, z) y la orientación (en grados) del centro de cada pieza en el espacio de trabajo, con respecto al Kinect. En la Figura **4-14** se presenta la imagen adquirida por visión y la estimación de la posición y orientación de cada pieza disponible en el espacio de trabajo. Los resultados obtenidos en esta etapa, son utilizados como los parámetros de la tarea de la operación de ensamble.

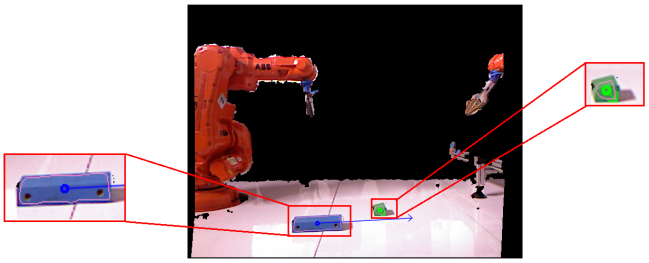

Figura 4-14: Identificación de piezas en espacio de trabajo.

Matriz de Transformación del Kinect al Robot

El cálculo de la matriz de transformación, se hizo en los siguientes etapas: ***i)*** Estimación de las coordenadas (x, y, z) del extremo del efector final del robot, ***ii)*** Movimiento manual del robot a cada una de las 8 esquinas que describen un cubo en el espacio de trabajo, ***iii)*** Almacenar las coordenadas del extremo del efector final, con respecto a la base del robot, ***iv)*** Almacenar las coordenadas del extremo del efector final, estimadas mediante visión, con respecto al Kinect. ***v)*** Calcular la matriz de transformación homogénea.

Para la obtención de ${}^{R}T_{K}$, se parte de la Ecuación 4-4, en donde A (ver Ecuación 4-5) es una matriz con las coordenadas homogéneas de las 8 esquinas del cubo con respecto al Kinect $({}^{K}P_1, {}^{K}P_2, {}^{K}P_3)$, B (ver Ecuación 4-6) es un vector columna con las coordenadas homogéneas del cubo con respecto al robot $({}^{R}P_1)$, M es el vector que contiene los elementos de la matriz de transformación homogénea. El vector fila Z_{14} (ver Ecuación 4-7) es una variable auxiliar, compuesta por 4 elementos con valor igual a 0.

$$M = A^{-1} \cdot B. \tag{4-4}$$

$$A = \begin{bmatrix} {}^{K}P_1 & Z_{14} & Z_{14} \\ Z_{14} & {}^{K}P_1 & Z_{14} \\ Z_{14} & Z_{14} & {}^{K}P_1 \\ {}^{K}P_2 & Z_{14} & Z_{14} \\ Z_{14} & {}^{K}P_2 & Z_{14} \\ Z_{14} & Z_{14} & {}^{K}P_2 \\ . & . & . \\ . & . & . \\ . & . & . \\ {}^{K}P_8 & Z_{14} & Z_{14} \\ Z_{14} & {}^{K}P_8 & Z_{14} \\ Z_{14} & Z_{14} & {}^{K}P_8 \end{bmatrix}. \tag{4-5}$$

$$B = \begin{bmatrix} {}^{R}P_1 & {}^{R}P_2 & {}^{R}P_3 & {}^{R}P_4 & {}^{R}P_5 & {}^{R}P_6 & {}^{R}P_7 & {}^{R}P_8 \end{bmatrix}^T \tag{4-6}$$

$$z_{14} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4-7}$$

A partir de los valores calculados para x, se obtiene la matriz de transformación homogénea ${}^{R}T_K$ (ver Ecuación 4-8). En la ecuación 4-9 se presenta la matriz de transformación obtenida para la posición en la cual se fijó el Kinect en el laboratorio. Adicionalmente, mediante la Ecuación 4-10 es posible calcular la posición ${}^{R}P_{obj}$ de un objeto, con respecto al sistema de referencia del robot, a partir de la posición ${}^{K}P_{obj}$, estimada mediante el Kinect.

$${}^{R}T_K = \begin{bmatrix} x(1) & x(2) & x(3) & x(4) \\ x(5) & x(6) & x(7) & x(8) \\ x(9) & x(10) & x(11) & x(12) \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4-8}$$

$${}^{R}T_K = \begin{bmatrix} 0{,}5525 & -0{,}0353 & 0{,}5528 & -0{,}0102 \\ -0{,}6155 & -0{,}0326 & 0{,}4348 & -990{,}6111 \\ 0{,}0449 & -1{,}0136 & -0{,}0102 & 285{,}6602 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4-9}$$

$${}^{R}P_{obj} = {}^{R}T_K \cdot {}^{K}P_{obj}. \tag{4-10}$$

Identificación de los parámetros de la tarea.

El modelo TP-GMM aprendido permite generar nuevas trayectorias al cambiar los parámetros que describen la tarea [45]. En el caso en particular del trabajo realizado, los parámetros que describen la tarea de ensamble son los objetos detectados mediante el sistema de visión, más específicamente, la posición de cada uno en el espacio de trabajo. Para cada operación generada usando TP-GMM, se requiere de 4 parámetros que describen la tarea, de los cuales 2 son estimados directamente por visión y los otros 2, son calculados indirectamente: ***i)*** Parámetro 1, coordenadas del primer objeto, estimadas directamente por visión; ***ii)*** Parámetro 2, punto cercano al primer objeto, calculado indirectamente; ***iii)*** Parámetro 3, punto cercano al segundo objeto, calculado indirectamente; ***iv)*** Parámetro 4, coordenadas del segundo objeto, estimadas directamente por visión.

Los parámetros 1 y 4, corresponden a las coordenadas (x, y, z) y a la orientación del objeto, estimadas directamente por visión de máquina. Por otra parte, el cálculo de los parámetros 2 y 3, se realiza siguiendo el Algoritmo 1. Como se puede evidenciar, el cálculo de los parámetros 2 y 3 se hace estimando los valores en función del valor absoluto de la variación entre los parámetros 1 y 4. El procedimiento es similar para el cálculo de las coordenadas (y, z) de los parámetros 2 y 3.

Algorithm 1 Estimación de los parámetros 2 y 3

Require: Cálculo de los parámetros 1 y 4.

1: $\Delta x = |P_{1x} - P_{4x}|$
2: Evaluar si la coordenada x del punto 1 es menor o igual a la coordenada x del punto 4. P_{1x}, P_{2x}, P_{3x} y P_{4x} representan la coordenada x de los puntos 1, 2, 3 y 4, respectivamente.
3: **if** $P_{1x} <= P_{4x}$ **then**
4: Asignar la coordenada x de los parámetros 2 y 3, en función de Δx.
5: $P_{2x} = P_{1x} + \Delta x \cdot 5\,\%$
6: $P_{3x} = P_{4x} - \Delta x \cdot 5\,\%$
7: **else**
8: $P_{2x} = P_{1x} - \Delta x \cdot 5\,\%$
9: $P_{3x} = P_{4x} + \Delta x \cdot 5\,\%$
10: **end if**
11: Ejecutar de manera análoga para calcular Δy, Δz y las coordenadas y, z de los puntos 2 y 3.

4.2. Sistema para generación del plan de ensamble.

Luego de obtener los modelos TP-GMM de cada subtarea de la operación de ensamble, la siguiente etapa consiste en obtener automáticamente los planes de ensamble según las piezas disponibles en el espacio de trabajo. La operación fue modelada usando redes de Petri y, a partir de una marcación inicial identificada visualmente, se calcula automáticamente la secuencia de operaciones a seguir para ensamblar las piezas disponibles. A continuación se genera el código RAPID, el cual es exportado manualmente a RobotStudio y a los robots, para ejecutar las operaciones.

4.2.1. Red de Petri.

Representación gráfica de la operación.

Las operaciones industriales, como las tareas de ensamble, pueden ser modeladas, en general, por una representación gráfica, tales como las redes de Petri. Como se presenta en [48], el uso de redes de Petri, permite agregar funcionalidades a las operaciones, como por ejemplo, recuperarse de errores conocidos. En el ejemplo de aplicación del presente trabajo, la operación de ensamble del carro de la Figura **4-2** presenta dos tipos de ensamble: fijar cabina y fijar rueda. A partir de lo anterior, se usó la librería Snakes [49] (disponible para Python), la cual permite trabajar con redes de Petri y ejecutar todas las acciones posibles, a partir de una marcación inicial. En el caso de la operación de ensamble, la red de Petri se diseñó manualmente y se generó el modelo usando la librería de Snakes. A continuación, se calcula la marcación inicial de la red a través del sistema de visión, mediante el cual se identifican las piezas disponibles en el espacio de trabajo. En la Figura **4-15** se presenta un diagrama

simplificado de la red de Petri que modela la operación de ejemplo.

Es importante mencionar que aunque la operación de ensamble de ejemplo es simple, y consecuentemente lo es la red de Petri, es posible extrapolar la metodología seguida para la generación de planes de ensamble de operaciones más complejas. En general, se requiere prestar especial atención al diseño de la red, de modo que al cambiar la marcación inicial, de acuerdo a las piezas disponibles, el plan de ensamble generado sea el esperado.

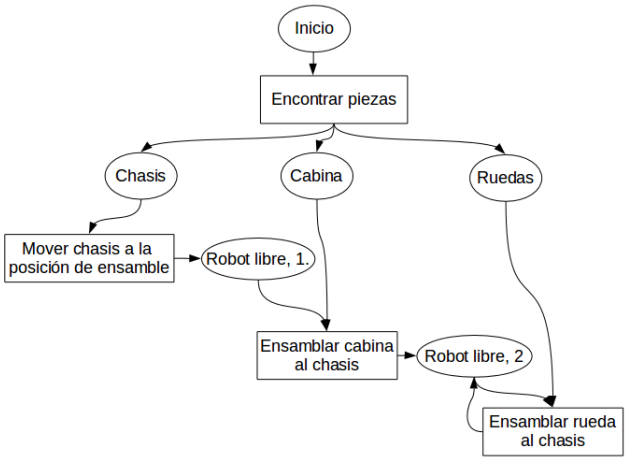

Figura 4-15: Diagrama simplificado de la red de Petri.

Generación automática del plan.

Se diseñó un algoritmo que permite generar automáticamente el plan de ensamble a partir de la marcación inicial, el cual utiliza las funcionalidades de la librería Snakes, y almacena las transiciones que se activan a partir de la marcación inicial estimada por visión. Adicionalmente, se genera un diagrama que permite validar, gráficamente, que el plan de ensamble es correcto. En la Figura **4-16** se presenta uno de los planes de ensamble obtenidos, donde se identificó la disponibilidad del chasis y la cabina, en el espacio de trabajo.

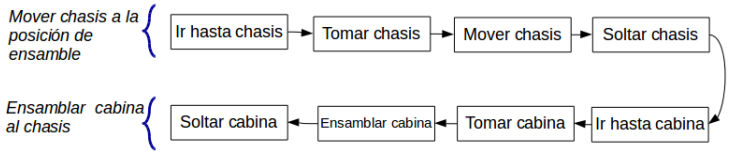

Figura 4-16: Diagrama de estados generado automáticamente.

A partir de la red simplificada (Figura **4-15**) y el diagrama de estados generado (Figura **4-16**), se puede observar que los bloques “Mover chasis a la posición de ensamble”, “Ensamblar

cabina al chasis", están compuestos por operaciones más simples. Lo anterior evidencia que para ejecutar una operación de ensamble, se requieren acciones primitivas de menor complejidad que la tarea completa.

Luego de validar que el plan generado es correcto, se genera automáticamente el código en RAPID, el cual posteriormente es simulado en RobotStudio y, finalmente, exportado al robot ABB. Con relación a esto, cabe resaltar que se usa un archivo diferente para cada acción que se presenta en la Figura **4-16**, para garantizar mayor orden e independencia entre las diferentes tareas. Es importante mencionar que en el lenguaje RAPID, los archivos se manejan bajo la estructura de modulos y procesos, de modo que, de acuerdo al plan generado a partir de la red de Petri, se generan diferente archivos automáticamente.

4.2.2. Generación de nuevas trayectorias.

En esta sección se describe el método seguido para la generación de nuevas trayectorias, a partir del modelo probabilístico ya entrenado. Como se presentó previamente, una de las ventajas de usar TP-GMM, es que permite calcular nuevas trayectorias, en función de los parámetros de la tarea. En el presente trabajo, los parámetros se calculan a partir de la posición y orientación de las piezas disponibles en el espacio de trabajo, los cuales son estimados a través del sistema de visión de máquina. A continuación, se describe la generación de trayectorias para el caso de la subtarea 1, denominada "Fijar cabina al chasis", la cual requiere del modelo previamente entrenado.

i) Posicionar todas las piezas de la operación de ensamble en el espacio de trabajo del robot. En el caso de la subtarea 1, únicamente se requiere de la cabina y el chasis para ejecutar la operación; ***ii)*** Estimar la posición y orientación de cada una de las partes, con respecto al sistema de referencia del Kinect, como se presenta en la Figura **4-14**; ***iii)*** Calcular de la posición de las piezas, con respecto al sistema de referencia del robot, usando la matriz de transformación ${}^{R}T_{K}$, previamente obtenida; ***iv)*** Determinar los parámetros de la tarea, en función de la posición y orientación de las piezas; y ***v)*** Generar nuevas trayectorias, usando el modelo TP-GMM y los parámetros de la tarea calculados en la etapa *iv*.

A partir de la trayectoria obtenida, se validan que todas las coordenadas (x, y, z) de la operación estén definidas. Es decir, se comprueba que se hayan calculado los valores para cada componente de la trayectoria, en todo instante de tiempo t. A continuación, en caso que la nueva trayectoria sea válida, se genera automáticamente el código RAPID, el cual se ejecuta en RobotStudio y, posteriormente, en el robot disponible. En la Figura **4-17** se presenta una trayectoria generada automáticamente. Los pasos anteriores, son ejecutados para cada una de las subtareas, de acuerdo a los objetos disponibles en el espacio de trabajo. Se requiere que los cambios entre las posiciones de los objetos usados en el entrenamiento, y las nuevas

posiciones, no sean considerables. De no cumplirse dicho requerimiento, la trayectoria generada no permitirá ejecutar correctamente la operación de ensamble.

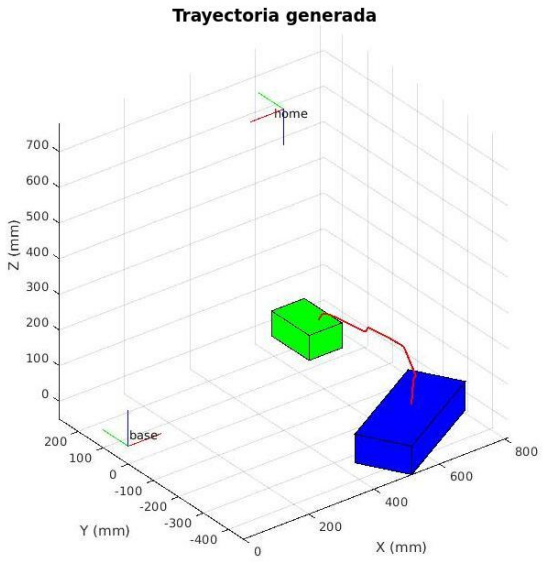

Figura 4-17: Trayectoria generada automáticamente - Subtarea 1.

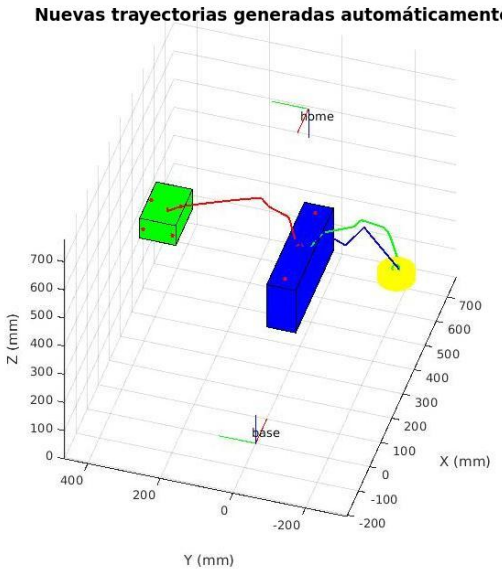

Figura 4-18: Operación con una rueda, la cabina y el chasis.

Adicionalmente a las rutinas de las nuevas trayectorias, se generan otros archivos para la apertura y el cierre del efector final, el cual es controlado mediante una electroválvula, que

a su vez recibe una señal digital proveniente del controlador. A partir del plan de ensamble obtenido automáticamente, se genera la rutina principal, desde la cual se llaman las otras funciones. Luego de obtener los modulos de RAPID, las operaciones fueron simuladas utilizando el software Robot Studio, desarrollado por ABB. Posteriormente, se ejecutaron las rutinas en un robot ABB IRB 140 disponible en el laboratorio.

Es importante mencionar que se encontraron dificultades para definir los parámetros de la tarea en función de la posición de los objetos. Lo anterior debido a que, a pesar de estimar correctamente la posición y orientación de las piezas en la escena, la generación de las nuevas trayectorias en algunas ocasiones no fue correcta. En particular, los problemas ocurrían al calcular los parámetros (de la tarea) 2 y 4, que corresponden a los puntos cercanos al inicio y al final de la trayectoria, respectivamente. Se encontró que al generar una nueva trayectoria usando el modelo TP-GMM aprendido, es necesario que las coordenadas (x_1, y_1, z_1) del parámetro 1 y las coordenadas (x_2, y_2, z_2) del parámetro 2, no tengan ninguna componente con el mismo valor. Lo mismo ocurre para los parámetros 3 y 4. En caso que lo anterior no se cumpla, la trayectoria generada presenta puntos con valores indeterminados y la operación de ensamble no se puede ejecutar. Como ya se mencionó, los parámetros 1 y 4 son estimados directamente mediante le sistema de visión, mientras que los parámetros 2 y 3 son calculados siguiendo el Algoritmo 1.

Resumen del capítulo

En el presente Capítulo se presenta la metodología propuesta para la ejecución de tareas de ensamble en un espacio tridimensional, desarrollada a partir de la descrita en el Capítulo 3 para operaciones en 2D. Dedido a que las tareas de ensamble implementadas se desarrollan en 3D, se agregaron etapas adicionales, como lo son: ***i)*** Desarrollo de un sistema de visión 3D para la identificación de la mano y los objetos en el espacio de trabajo; ***ii)*** Representación gráfica de la operación de ensamble mediante Redes de Petri; ***iii)*** Generación automática del plan de ensamble, usando la red de Petri del proceso y los objetos disponibles en el espacio de trabajo, identificados por el sistema de visión propuesto; ***iv)*** Validación y simulación de las trayectorias generadas automáticamente, con MatLab y RobotStudio. En el siguiente Capítulo, se presentan los resultados obtenidos de las operaciones con los bloques en 2 dimensiones y de las tareas de ensamble en 3 dimensiones.

5 Resultados.

En este capítulo se presentan los resultados obtenidos al implementar las metodologías propuesta para las tareas de manipulación en 2 dimensiones y para las operaciones de ensamble en 3 dimensiones. Es importante resaltar que aunque los sistemas propuestos son similares (2D y 3D), los resultados obtenidos evidencian las dificultades que surgen al extrapolar una técnica en 2D a otra en 3D.

5.1. Operaciones en 2D.

5.1.1. Manipulación de un bloque.

Se generaron diferentes trayectorias, usando los modelos representados por 3 Gaussianas y por 5 Gaussianas, variando los parámetros de la tarea. En la Figura **5-1** se presentan los resultados obtenidos al generar 9 trayectorias (posición inicial en cada marca circular o y posición final con marca x), con diferentes posiciones iniciales del bloque, estimadas mediante el sistema de visión, y la misma posición final. En este caso, se tomó como posición final $(40, 60)$ [cm].

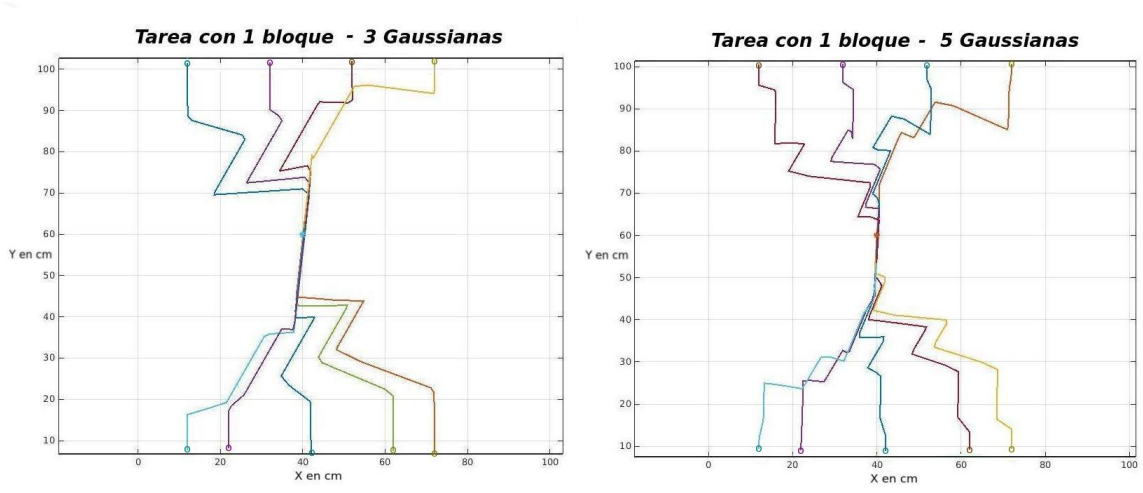

Figura 5-1: Trayectorias generadas con diferentes parámetros de la tarea.

Los parámetros de la tarea de las posiciones iniciales (estimados mediante el sistema de

visión), presentan un error de hasta 1.5 [cm] de posición y de 10° en orientación. Lo anterior puede inducir a que al inicio de la nueva trayectoria generada, no se tome el bloque correctamente con el robot. Sin embargo, los casos en los cuales el error de la estimación de la posición alcanza los valores límites, son en configuraciones muy particulares; como por ejemplo, cuando las piezas están sobre las esquinas más lejanas de la cámara. En la Tabla **5-1** se presentan los resultados obtenidos, al generar 20 trayectorias automáticamente, usando la metodología propuesta. Se determinó que una trayectoria es incorrecta si: ***i)*** El robot no toma la pieza correctamente; ***ii)*** La posición final del bloque no es la esperada.

Operación.	**Correctas.**	**Incorrectas.**	**Total.**	**Error.**
Un bloque	18	2	20	10 %

Tabla 5-1: Generación de trayectoria con un bloque.

5.1.2. Manipulación de dos bloques.

Como se presentó en el Capítulo 3, la operación de manipulación con dos bloques está compuesta por las subtareas "Ir a bloque 1" (IB1), "Mover bloque 1" (MB1), "Ir a bloque 2" (IB2), y "Mover bloque 2" (MB2). En total se entrenaron 4 modelos (uno por cada subtarea), cada uno compuesto por 5 demostraciones. En las Figuras **5-2**, **5-3**, **5-4** y **5-5** se presentan las demostraciones adquiridas y las trayectorias generadas usando GMR para cada subtarea. El número de Gaussianas de cada modelo varía de acuerdo a la complejidad del movimiento. La primera subtarea se compone de tres Gaussianas, la segunda y la tercera de una y la cuarta de tres.

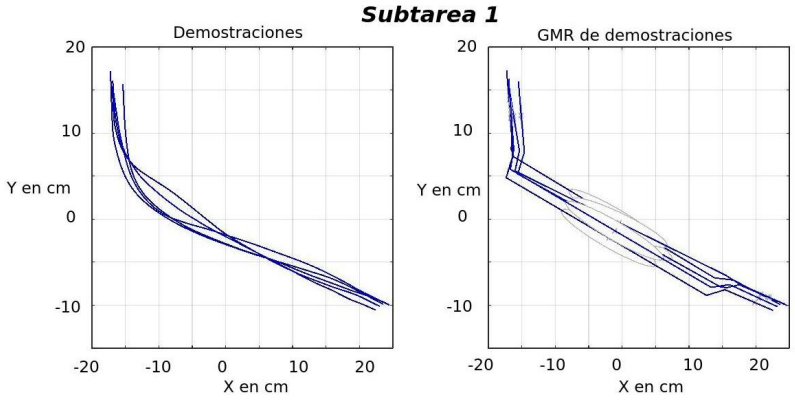

Figura 5-2: Modelo aprendido de la primera subtarea.

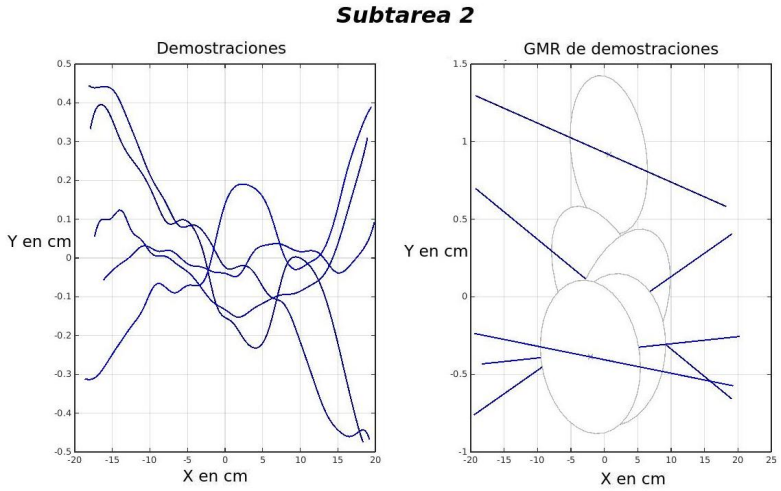

Figura 5-3: Modelo aprendido de la segunda subtarea.

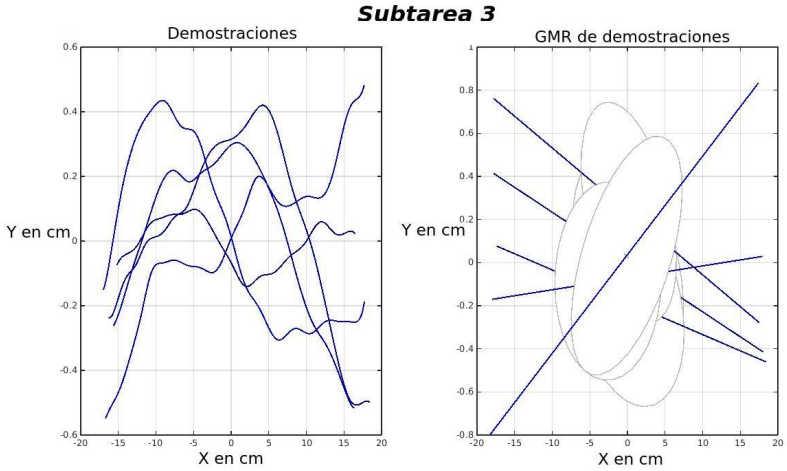

Figura 5-4: Modelo aprendido de la tercera subtarea.

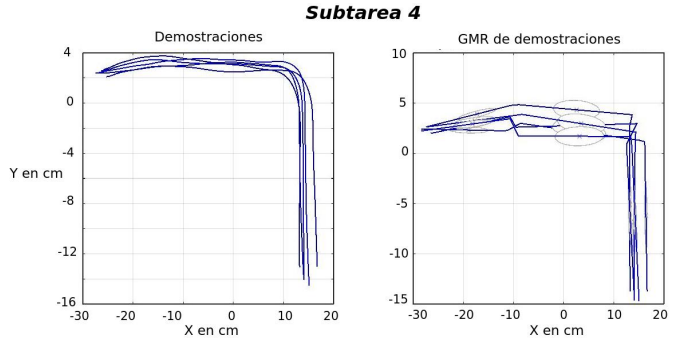

Figura 5-5: Modelo aprendido de la cuarta subtarea.

En las subtareas 2 y 3, se observa que el movimiento ejecutado tiene principalmente componente en x y que la variación en y es cercana a cero. A partir de esto, se encontró que con una Gaussiana el movimiento se modelaba satisfactoriamente. Por otra parte, las subtareas 1 y 4 presentan variaciones tanto en x como en y, por lo que se utilizaron 3 Gaussianas. A partir del cálculo de los parámetros de tarea presentados en la sección anterior, se generaron operaciones con los dos bloques usando los modelos entrenados para cada una de las 4 subtareas.
En la Figura **5-6** se presentan algunas trayectorias generadas, usando los modelos TP-GMM aprendidos de cada tarea. En la Tabla **5-2** se presentan los resultados obtenidos, al generar automáticamente 10 trayectorias de cada subtarea. En la Tabla **5-2** se presentan los resultados obtenidos, al generar automáticamente 15 trayectorias de cada subtarea.

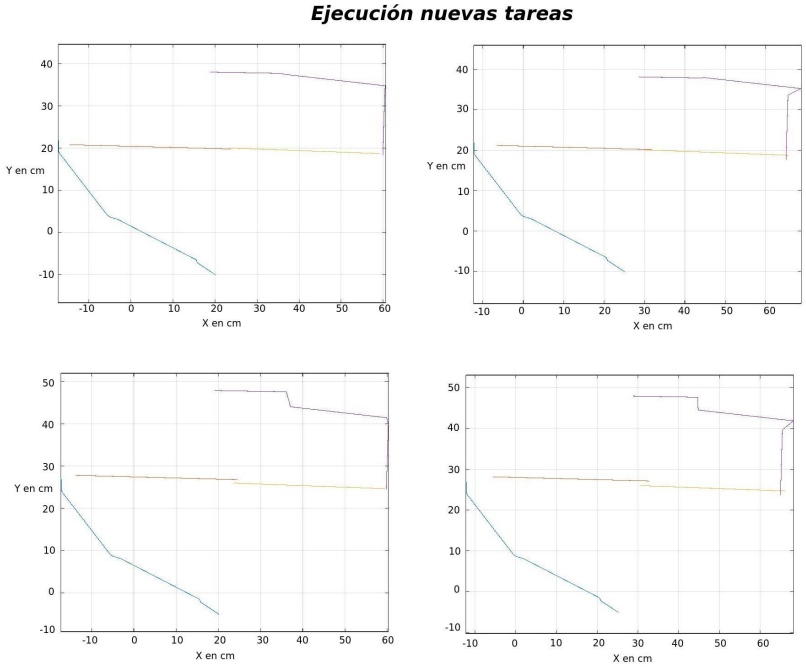

Figura 5-6: Trayectorias de la operación con dos bloques generadas por TP-GMM.

5.2. Operaciones en 3D

En esta sección se presentan los resultados obtenidos, luego de seguir la metodología presentada en secciones anteriores para la adquisición de las demostraciones y el entrenamiento de los modelos probabilísticos. Las nuevas trayectorias generadas, fueron validadas en el software RobotStudio, el cual permite simular todas las tareas con el robot. Luego de comprobar que las operaciones generadas son correctas, se ejecutan las tareas de ensamble en el robot. Se presentan dos operaciones de ensamble diferentes: ensamble del carro y colocación de la

Operación.	**Correctas.**	**Incorrectas.**	**Total.**	**Error.**
IB1	14	1	15	6,66 %
MB1	14	1	15	6,66 %
IB2	14	1	15	6,66 %
MB2	13	2	15	13,33 %
Total	***55***	***5***	***60***	8,33 %

Tabla 5-2: Generación de trayectorias con 2 bloques.

tapa de una botella; esta última, es una operación compuesta por una subtarea, de modo que no se genera automáticamete el plan, utilizando la red de Petri. En ambos casos, se evidencia que el uso de técnicas de ApD para operaciones de ensamble permite alcanzar resultados satisfactorios.

A partir de los modelos TP-GMM aprendidos, se generan nuevas trayectorias modificando los parámetros de la tarea. En ambas operaciones de ensamble, se cambian las posiciones iniciales de los objetos en el espacio de trabajo. Luego, las trayectorias obtenidas son validadas mediante una interfaz desarrolada en MatLab y, posteriormente, ejecutadas en RobotStudio. Finalmente, luego de determinar que la operación de ensamble es correcta, esta es ejecutada en el robot. A continuación, se presentan las etapas de simulación e implementación desarrolladas.

5.2.1. Simulación

A partir de las trayectorias generadas y validadas en la interfaz desarrollada en MatLab, se desarrolló una etapa de simulación utilizando RobotStudio. El uso de este, permite modelar virtualmente el entorno de trabajo, compuesto por el robot y las diferentes piezas, y ejecutar directamente el código RAPID de la trayectoria, generado de manera automática.
Luego, se simularon las operaciones con el fin de evaluar si las operaciones de ensamble son satisfactorias. En la Figura **5-7** se presenta una secuencia de imagenes de una operación de ensamble exitosa, en la cual se fija la cabina al chasis. En este caso, se evidencia que la tarea fue ejecutada satisfactoriamente debido a que la estimación de la posición de las piezas fue precisa, de modo que al generar la trayectoria, se fija la cabina al chasis. Por otra parte, en la Figura **5-8**, se presenta una operación fallida de ensamble, debido a que la estimación de la posición del chasis no fue correcta. Se puede observar que la posición final de la cabina no está en el extremo del chasis, lo cual demuestra que, en caso de ejecutar la operación en el robot real, no se ejecutaría satisfactoriamente.

Con respecto a la operación de colocación de la tapa de la botella, las demostraciones fueron

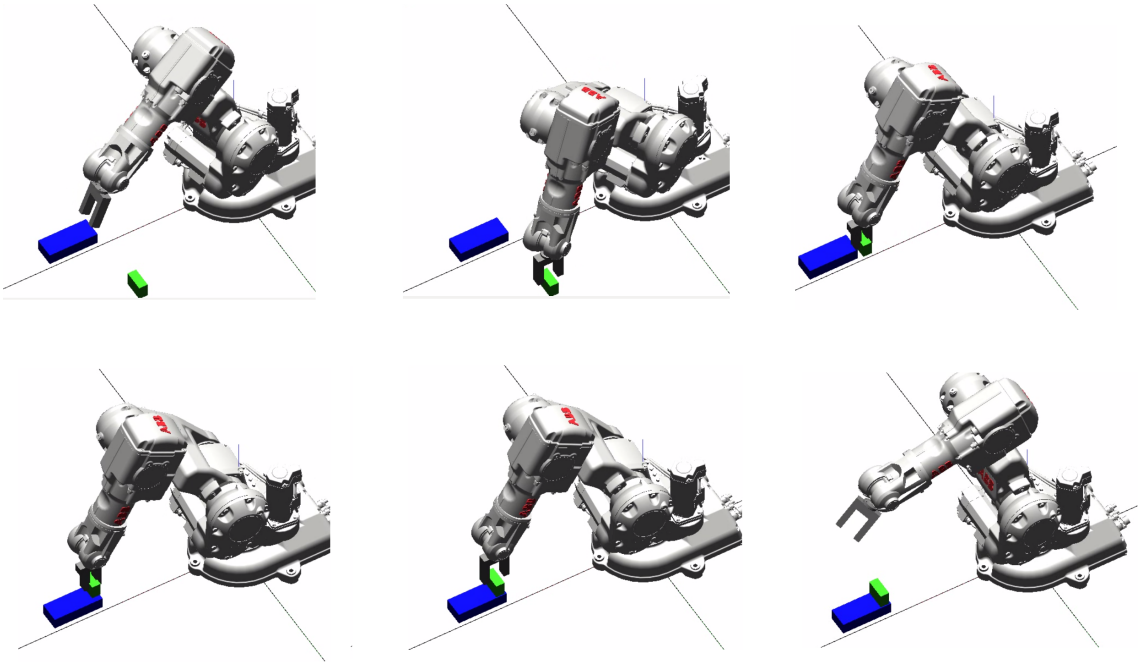

Figura 5-7: Simulación de ensamble exitoso de la cabina.

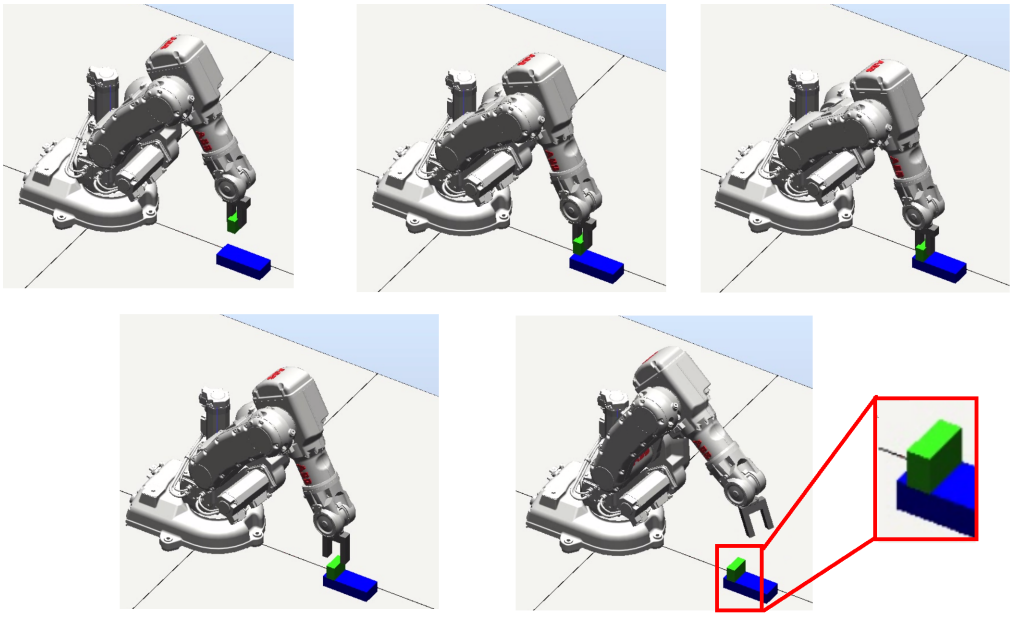

Figura 5-8: Simulación de ensamble fallido de la cabina.

ejecutadas por una persona, como se presenta en la Figura **4-4**. A partir de lo anterior, la

operación consiste en tomar la tapa y ensamblarla la botella, la cual tiene una posición fija. Debido a esto, los parámetros de la tarea están en función de la posición inicial de la tapa, que si varía. De la misma manera a como se hace en la operación del carro, la posición de la tapa es estimada usando el sistema de visión de máquina.

Se siguió la metodología descrita para el preprocesamiento de las demostraciones, el entrenamiento del modelo TP-GMM y la generación de nuevas trayectorias. Debido a que en esta operación únicamente se fija la tapa a la botella, no se implementó la sección de segmentación de las demostraciones, ya que la operación está compuesta solo de una tarea. En la Figura **5-9** se presentan las demostraciones adquiridas y posteriormente generadas usando GMR, con un modelo compuesto por 6 Gaussianas, mayor a los de la operación de ensamble del carro (4).

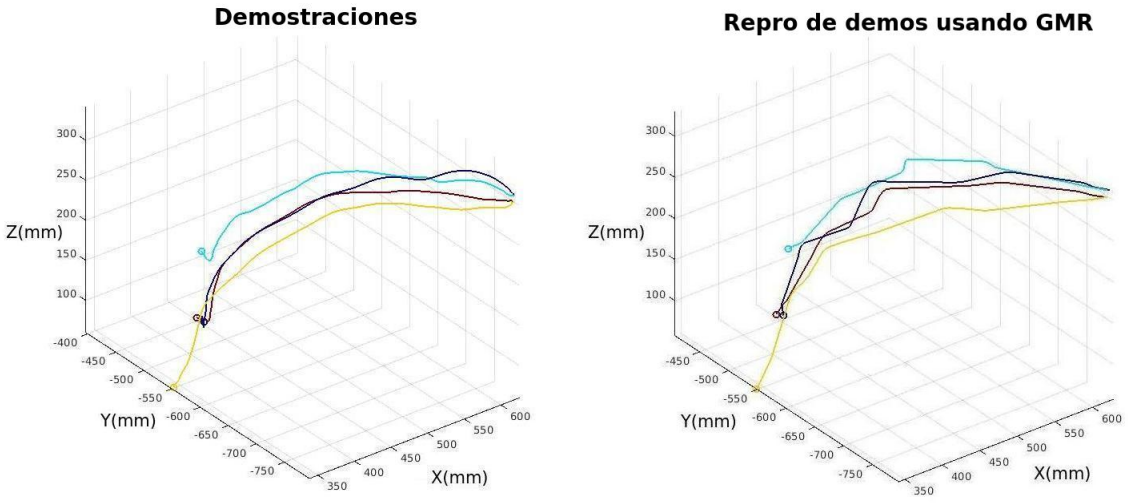

Figura 5-9: a) Demostraciones preprocesadas, b) Reproducción de demostraciones usando GMR.

5.2.2. Implementación

La segunda etapa, luego de simular las trayectorias en RobotStudio, consistió en ejecutar las operaciones de ensamble con el robot. Es importante resaltar que el uso del software permite validar la sintaxis del código, así como determinar si las tareas son ejecutadas satisfactoriamente. Otra ventaja que presenta el uso de RobotStudio, es que permite exportar el código RAPID al robot, siempre que el equipo con el software y el controlador del robot, estén en el mismo segmento de red. El Kinect se fijó usando un trípode, en la dirección del espacio de trabajo del robot ABB.

En la Figura **5-10** se presenta la secuencia de la operación de ensamble de la cabina al chasis. En este caso, la operación fue satisfactoria, ya que la estimación de las posición inicial de las piezas fue precisa y, consecuentemente, la generación automática de la trayectoria. Se encontró que uno de los puntos críticos para la generación automática de las tareas de ensamble está ligado al cálculo de los parámetros de la tarea.

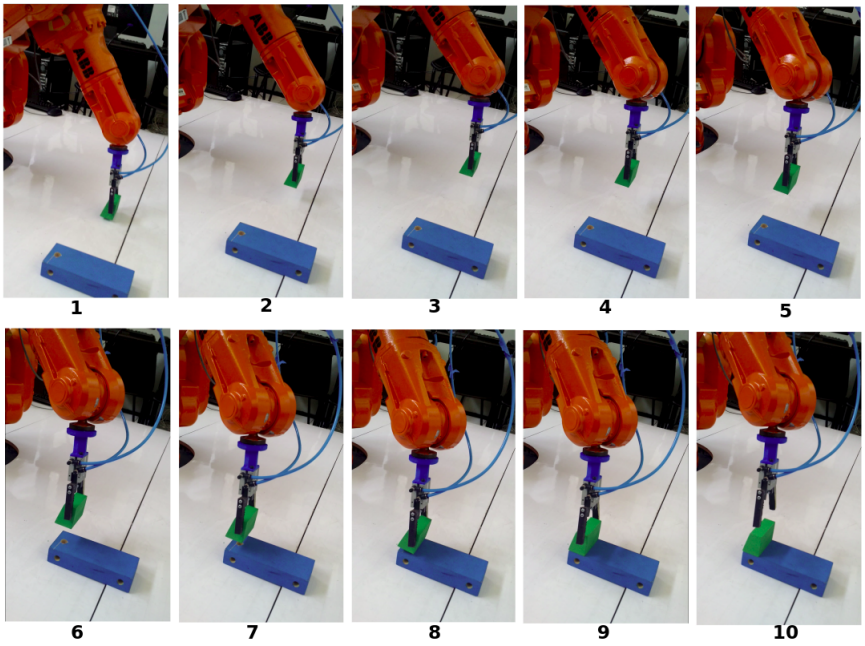

Figura 5-10: Secuencia de ensamble de la cabina al chasis.

Se realizaron 2 experimentos para validar el sistema propuesto: en el primero, se generaron 30 nuevas trayectorias de la operación de ensamble del carro, 10 para cada subtarea, y 20 trayectorias para la operación con la botella. En este caso, la posición de los objetos en la escena cumplió la restricción definida previamente, de garantizar que la posición de cada objeto, este contenida en un círculo de radio 20 *cm*. con respecto a las demostraciones. En el segundo experimento, se generaron 30 trayectorias nuevamente, pero en esta ocasión no se cumplió la restricción en cuanto a la posición de los objetos en la escena. En las tablas **5-4** y **5-3** se presentan los resultados obtenidos en los experimentos anteriores. Las trayectorias incorrectas corresponden a los casos en los cuales la tarea de ensamble no se ejecutó satisfactoriamente.

En la Figura **5-11** se presenta una secuencia de imágenes en la cual la operación de ensamble no fue correcta.

Operación.	Correctas.	Incorrectas.	Total.	Error.
Subtarea 1	6	4	10	40 %
Subtarea 2	7	3	10	30 %
Subtarea 3	6	4	10	40 %
Botella	14	6	20	30 %
Total	***33***	***17***	***50***	34 %

Tabla 5-3: Generación de trayectorias, experimento 1.

Operación.	Correctas.	Incorrectas.	Total.	Error.
Subtarea 1	9	1	10	10 %
Subtarea 2	9	1	10	10 %
Subtarea 3	8	2	10	20 %
Botella	18	2	20	10 %
Total	***44***	***6***	***50***	12 %

Tabla 5-4: Generación de trayectorias, experimento 2.

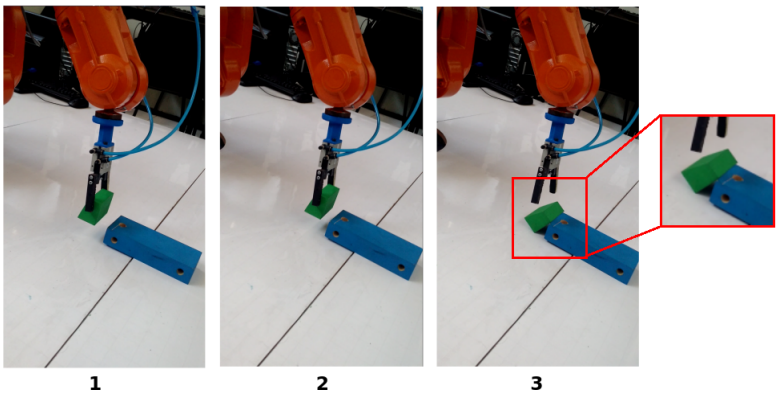

Figura 5-11: Secuencia de ensamble fallido de la cabina al chasis.

Resumen del capítulo

En el presente capítulo se presentan los resultados obtenidos al implementar las metodologías propuestas para la ejecución de operaciones en 2 y 3 dimensiones. Con respecto a las tareas en 2 dimensiones, los resultados obtenidos presentan un error cercano al 13 % para las operaciones con uno o dos bloques. Por otra parte, las operaciones de ensamble, al ser en el espacio, presentan un error considerable, cercano al 40 %, si los objetos no se ubican en posiciones cercanas a las usadas en la etapa de entrenamiento del modelo TP-GMM. De la misma manera, se evaluó el rendimiento del sistema para las operaciones de ensamble

en 3D y se encontró que presenta una eficiencia de alrededor del 88 % para el ensamble del carro presentado en la Figura **4-2**. Adicionalmente, se evaluó la metodología propuesta en la operación de colocación de la tapa de una botella, como se presenta en la Figura **5-9**. A partir de los resultados anteriores, se evidencia que la metodología propuesta permite ejecutar diferentes tipos de tareas en 2 y 3 dimensiones, usando técnicas de ApD en manipuladores robóticos. En el siguiente Capítulo, se presenta la discusión de los resultados obtenidos y las conclusiones.

6 Análisis de resultados y conclusiones.

Los resultados obtenidos, luego de validar la metodología propuesta en diferentes casos de estudio, son satisfactorios. Con respecto a las tareas de manipulación de bloques en 2 dimensiones, como se presenta en la Tabla **5-2**, se obtiene un error promedio de 8.33 %. En particular, se evidencia un error de 6.66 % para las subtareas 1 ("Ir a Bloque 1"), 2 ("Mover Bloque 1") y 3 ("Ir a Bloque 2), mientras que para la subtarea 4 ("Mover Bloque 2") el error es de 13.33 %. Lo anterior se debe a que esta última subtarea, (MB2), corresponde a un movimiento que implica variación tanto en x, como en y. Los valores de error presentan estas magnitudes, debido a que se garantizó que la posición de los objetos en el espacio de trabajo, fuera cercana a las usadas en la etapa de entrenamiento.

Por otra parte, con respecto a la metodología en 3 dimensiones, se encontró que, en general, presenta valores de error más elevados, comparado con las operaciones en 2D. A partir de la Tabla **5-4**, se evidencia que el sistema propuesto genera un 88 % de trayectorias correctas, para las tareas de ensamble ejecutadas. El error en las operaciones de ensamble es bajo, debido a que se cumple la restricción de garantizar que la posición de los objetos, no presente variaciones considerables con respecto a las usadas durante la etapa de entrenamiento. Por otra parte, en la Tabla **5-3**, no se cumple la restricción anterior, el sistema propuesto presenta errores mucho más elevados, cercanos al 40 %. Las diferencias en el error se deben a la posición de los objetos en el espacio de trabajo, ya que en el segundo experimento se ubicaron los objetos en posiciones lejanas a las usadas en la etapa de entrenamiento.

Se encontró que las trayectorias generadas incorrectamente en 2D y 3D ocurren debido a los siguientes factores: ***1)*** La subtarea MB2, en 2D, corresponde a una trayectoria que implica variación en x, y en y, a diferencia de las otras subtareas que únicamente tienen variación en un solo componente; ***2)*** La posición de los objetos en el espacio de trabajo, presenta variaciones considerables, con respecto a las demostraciones; y ***3)*** La estimación de la posición, usando el Kinect, presenta un error de posición superior a 1.5 cm. Con respecto a la operación de ensamble de la botella, únicamente se estima por visión la posición de la tapa, debido a que la posición de la botella es fija. Debido a lo anterior, el cálculo de los parámetros de la tarea, en esta operación, están únicamente en función de la posición de la tapa. Adicionalmente, es importante mencionar que al implementar la metodología en 3D,

aumentó el error, con respecto al valor obtenido en las tareas en 2D, debido a los siguientes dos factores: ***1)*** El entrenamiento de los modelos GMM es más dispendioso, debido a que no corresponden a elipses en un plano, sino a elipsoides en el espacio; ***2)*** La estimación de los parámetros de la tarea es más susceptible al error, debido a que las coordenadas de cada parámetro tiene componentes x, y, z.

Es importante resaltar la importancia del uso del software de simulación, tal como lo es RobotStudio, el cual permitió validar el funcionamiento de la metodología propuesta. Lo anterior, debido a las herramientas con las que cuenta permiten la ejecución en tiempo real del código generado desde otro programa, como por ejemplo MatLab. Así mismo, permite exportar directamente al controlador del robot real, el código generado desde un PC. Adicionalmente, permite la simulación de los objetos detectados en el espacio de trabajo, mediante el sistema de visión de máquina. Sin embargo, si las posiciones estimadas mediante el sistema de visión no son correctas, las simulaciones de las operaciones no serán útiles. Es por esto, que es fundamental contar con un sistema de visión que estime de manera confiable la posición de los objetos en la escena.

Con el objetivo de mejorar los resultados obtenidos, se proponen dos soluciones: ***1)*** Ubicar los objetos en el espacio de trabajo, en posiciones en las que la variación, con respecto a las usadas en las demostraciones, no sea superior a 20 cm; y ***2)*** Usar una cámara con mejor resolución. Por otro lado, aunque los ejemplos implementadas durante el trabajo de investigación, corresponden a operaciones simples, como lo son las tareas de manipulación de bloques en 2 dimensiones, el ensamble de un carro y la fijación de la tapa de una botella, en 3 dimensiones, estas pueden extrapolarse a operaciones más complejas, que requieran un mayor número de pasos.

Bibliografía

[1] S. Calinon, T. Alizadeh, and D. G. Caldwell, "On improving the extrapolation capability of task-parameterized movement models," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 610–616, IEEE, Nov. 2013.

[2] D. C. Bentivegna and C. G. Atkeson, "A Framework for Learning from Observation Using Primitives," *RobotCup 2002: Robot Soccer World Cup VI*, pp. 263–270, 2003.

[3] C. a. Acosta Calderon, R. E. Mohan, and C. Zhou, "Teaching new tricks to a robot learning to solve a task by imitation," in *2010 IEEE Conference on Robotics, Automation and Mechatronics, RAM 2010*, pp. 256–262, IEEE, June 2010.

[4] Y. Wang, Y. Cui, G. Q. Huang, P. Zhang, and S. Chen, "Study on fruit quality inspection based on its surface color in produce logistics," *Proceedings - 2010 International Conference on Manufacturing Automation, ICMA 2010*, pp. 107–111, 2010.

[5] Matlab, "Convert from HSV to RGB Color Space," 2015.

[6] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[7] C. G. Atkeson and S. Schaal, "Learning tasks from a single demonstration," in *International Conference on Robotics and Automation*, vol. 2, pp. 1706–1712, IEEE, 1997.

[8] J. Aleotti and M. Reggiani, "Toward Programming of Assembly," *Proceedings of the 2003 IEEE International Workshop on Robot and Human Interactive Communication Communication*, 2003.

[9] N. Abdo, H. Kretzschmar, L. Spinello, and C. Stachniss, "Learning manipulation actions from a few demonstrations," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1268–1275, 2013.

[10] K. Dixon, S. Iba, J. Dolan, and P. Khosla, "Gesture-based Programming for Robotic Arc Welding Multi-modal interface," *English*, 2002.

[11] P. Cederberg, M. Olsson, and G. Bolmsj, "A semiautomatic task-oriented programming system for sensor-controlled robotised small-batch and one-off manufacturing," *Robotica*, vol. 23, no. 6, pp. 743–754, 2005.

[12] R. Cubek and W. Ertel, "Learning and Application of High-Level Concepts with Conceptual Spaces and PDDL," *3rd Workshop on Planning and Learning*, pp. 76–83, 2011.

[13] K. Khoshelham, "Accuracy Analysis of Kinect Depth Data," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVIII-5/, no. August, pp. 133–138, 2012.

[14] A. Chella, H. Dindo, and I. Infantino, "Learning high-level manipulative tasks through imitation," in *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, pp. 251–256, IEEE, Sept. 2006.

[15] G. Hayes and J. Demiris, "A robot controller using learning by imitation," *University of Edinburgh, Department of Artificial Intelligence*, pp. 198–204, 1994.

[16] B. Balaguer and S. Carpin, "Human-Inspired Grasping of Novel Objects through Imitation Learning," *Robotics.Ucmerced.Edu*, 2010.

[17] M. a. Goodrich and A. C. Schultz, "Human-robot interaction: a survey," *Found. Trends Hum.-Comput. Interact.*, vol. 1, no. 3, pp. 203–275, 2007.

[18] V. a. Oliveira and A. Conci, "Skin Detection using HSV color space," *H. Pedrini, & J. Marques de Carvalho, Workshops of Sibgrapi*, pp. 1–2, 2009.

[19] S. Sural, G. Qian, and S. Pramanik, "Segmentation and histogram generation using the HSV color space for image retrieval," *International Conference on Image Processing*, vol. 2, pp. 589–592, 2002.

[20] M. M. Hasan and P. K. Misra, "Gesture Recognition Using Modified HSV Segmentation," in *2011 International Conference on Communication Systems and Network Technologies*, pp. 328–332, IEEE, June 2011.

[21] O. Ikeda, "Segmentation of faces in video footage using HSV color for face detection and image retrieval," in *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, vol. 2, pp. III–913–6, IEEE, 2003.

[22] M. Hayat, M. Bennamoun, and A. a. El-Sallam, "An RGB-D based image set classification for robust face recognition from Kinect data," *Neurocomputing*, vol. 171, pp. 889–900, July 2016.

[23] N. Dantam, I. Essa, and M. Stilman, "Linguistic transfer of human assembly tasks to robots," *IEEE International Conference on Intelligent Robots and Systems*, no. October, pp. 237–242, 2012.

[24] J. G. Hoyos-Gutiérrez and F. A. Prieto-Ortiz, "Programación por demostración de la secuencia de apretar una tuerca admitiendo variaciones en posición de la llave," *Tecno Lógicas*, vol. 19, no. 36, pp. 77–90, 2016.

[25] M. Bueno, L. Díaz-Vilariño, J. Martínez-Sánchez, H. González-Jorge, H. Lorenzo, and P. Arias, "Metrological evaluation of KinectFusion and its comparison with Microsoft Kinect sensor," *Measurement: Journal of the International Measurement Confederation*, vol. 73, pp. 137–145, May 2015.

[26] N. Burrus, "Kinect Calibration Toolbox." https://www.mathworks.com/matlabcentral/linkexchange/links/2882-kinect-calibration-toolbox, 2011.

[27] R. S. Yang, Y. H. Chan, R. Gong, M. Nguyen, A. G. Strozzi, P. Delmas, G. Gimel'Farb, and R. Ababou, "Multi-Kinect scene reconstruction: Calibration and depth inconsistencies," in *International Conference Image and Vision Computing New Zealand*, pp. 47–52, IEEE, Nov. 2013.

[28] I. T. Jolliffe, *Principal Component Analysis*, vol. 3. Wiley Online library, 2005.

[29] X. Feng and P. Milanfar, "Multiscale principal components analysis for image local orientation estimation," *Signals, {Systems} and {Computers}, 2002. {Conference} {Record} of the {Thirty}-{Sixth} {Asilomar} {Conference} on*, vol. 1, pp. 478–482, 2002.

[30] R. Küffner, T. Petri, L. Windhager, and R. Zimmer, "Petri nets with fuzzy logic (PNFL): Reverse engineering and parametrization," *PLoS ONE*, vol. 5, no. 9, pp. 1–10, 2010.

[31] R. Zurawski and M. C. Zhou, "Petri Nets and Industrial Applications: A Tutorial," *IEEE Transactions on Industrial Electronics*, vol. 41, no. 6, pp. 567–583, 1994.

[32] C. A. Petri, "Communication with Automata," *Applied Data Research*, vol. 15, no. 8, pp. 357–62, 1966.

[33] G. Chang and D. Kulic, "Robot task error recovery using Petri nets learned from demonstration," *2013 16th International Conference on Advanced Robotics, ICAR 2013*, pp. 31–36, 2013.

[34] T. Murata, "Petri Nets : Properties , Analysis and Appl k a t ions," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[35] M. P. n. Cabrera, I. L. Juárez, H. N. Gómez, and R. C. Osorio, "Automatización del Proceso de Ensamble Utilizando Visión Artificial," *séptimo Congreso internacional de Cómputo en optimización y software*, pp. 236–249, 2009.

[36] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, pp. 286–298, Apr. 2007.

[37] a.P. Dempster, N. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society Series B Methodological*, vol. 39, no. 1, pp. 1–38, 1977.

[38] M. Lopes and J. Santos-Victor, "Visual learning by imitation with motor representations," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 35, pp. 438–449, June 2005.

[39] G. C. Manjunath Patel, P. Krishna, and M. B. Parappagoudar, "An intelligent system for squeeze casting process - soft computing based approach," *International Journal of Advanced Manufacturing Technology*, vol. 86, no. 9-12, pp. 3051–3065, 2016.

[40] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5239–5246, 2012.

[41] A. K. Tanwani and S. Calinon, "Learning Robot Manipulation Tasks with Task-Parameterized Semitied Hidden Semi-Markov Model," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 235–242, 2016.

[42] T. Welschehold and C. Dornhege, "Learning manipulation actions from human demonstrations," *Intelligent Robots and*, vol. 1, pp. 3772–3777, 2016.

[43] A. Fod, M. J. Mataric, and O. C. Jenkins, "Automated derivation of primitives for movement classification," *Autonomous Robots*, vol. 12, no. 1, pp. 39–54, 2002.

[44] D. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," *Workshop on Knowledge Knowledge Discovery in Databases*, vol. 398, pp. 359–370, 1994.

[45] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, pp. 1–29, Sept. 2016.

[46] R. W. Gonzalez, R. C., Woods, *Digital Image Processing*, vol. 14. 2002.

[47] C. D. Boor, "B (asic) -Spline Basics," tech. rep., Wisconsin University - Madison Mathematics Research Center, 1986.

[48] G. Chang and D. Kulic, "Robot task error recovery using Petri nets learned from demonstration," *2013 16th International Conference on Advanced Robotics, ICAR 2013*, pp. 31–36, 2013.

[49] F. Pommereau, *SNAKES: A flexible high-level Petri nets library*, vol. 9115, pp. 254–265. Cham: Springer International Publishing, 2015.