Universidad de las Ciencias Informáticas

Facultad 2

Título:

Desarrollo de una aplicación para el control inteligente de un robot manipulador.

Trabajo de Diploma en opción al título de Ingeniero en Ciencias Informáticas

Autores:

Sergio Orlando Boudy González.

Milagro Leonor Leyva Fonseca

Tutor:

Ing. Javier Alexander León Martínez

Ciudad de La Habana

Junio del 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Sergio Orlando Boudy González Milagro Leonor Leyva Fonseca

_____

Javier Alexander León Martínez

PENSAMIENTO

La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica.

Aristóteles

AGRADECIMIENTOS.

Este trabajo de diploma fue posible gracias a la ayuda de algunas personas que contribuyeron y colaboraron de una manera u otra con la realización del mismo, y a los cuales les agradecemos por su apoyo incondicional.

Es imposible dejar de mencionar la valiosa colaboración para la feliz realización de este trabajo, al Dr. Abelardo del Pozo Quintero, al MsC. Guillermo Álvarez Bestard, al Tec. Miguel Ángel Machirán Simón, al Ing. Rafael Acosta, al Dr. Roberto Rodríguez Morales, y demás trabajadores e investigadores del Instituto de Cibernética, Matemática y Física (ICIMAF), que nos acogieron a todos como parte de su equipo.

Consideramos también de inestimable valor la colaboración del Profesor MsC. Eugenio César Sánchez, del Departamento de Automática y Computación del Instituto Superior Politécnico "José Antonio Echeverría" (ISPJAE), así como a los profesores MsC. Yadira Ruiz Constanten, Ing. Yenly Pérez Núñez, Ing. Lourdes Escalona Peral, Ing. Siovel Rodríguez Morales, Ing. Yalice Gámez Batista, Dra. Sayda Coello González y a nuestra Tania Del Sol Viera, sin dejar de mencionar a nuestros compañeros Daniel García González, Eilen Marien Meriño Coronado y Eliecer Peña Reyes, todos de nuestra Universidad de Ciencias Informáticas (UCI). Consideramos necesario agradecer además al Ing. Remberto León Martínez y al Ing. Javier González Orozco.

En especial queremos agradecer a nuestro tutor, Ing. Javier Alexander León Martínez por ser no solo un tutor, sino un amigo y el autor principal de este trabajo, que con paciencia y dedicación nos ha encaminado en nuestra vida profesional.

Sergio y Milagro.

DEDICATORIA:

A Eilen que siempre estuvo a mi lado incondicionalmente y que ayudó a terminar el documento en 4 días, a mis padres, a mi primo Javy, mi hermana, mis abuelos, mis tíos y toda mi familia, a mis futuros suegros, a mis amigos, a todas aquellas personas que de una forma u otra me han apoyado, y a la Revolución.

Sergio

A mis padres, Isabel y Rafael, por su apoyo incondicional en cada momento de mi vida y confiar en mí, por brindarme todo su apoyo y la mejor educación, por tenerme siempre presente en sus corazones y quererme tanto.

A mi hermano Miguel por haber extendido su mano ante cada obstáculo.

A mis abuelos que han depositado toda su confianza en mí.

A mis sobrinos que son mi tesoro más preciado.

A mi tía Ojilma que ha estado en cada momento pendiente de como he ido saliendo en todas las etapas de mi carrera.

A mi novio por su preocupación constante y exigencia, por todo el amor que me ha brindado.

A mis mejores amigos Lester y William que me han ayudado cuando más los he necesitado.

A todos mis familiares que cada día se preocupan por mis estudios y mi salud, y a todos mis amigos por estar presentes en cada momento de mi vida.

A mi compañero de tesis por ayudarme a realizar el trabajo más importante de mi vida.

A mis compañeros de grupo que me han ayudado y apoyado en todo este tiempo en especial a Lisbet, Sheila, Arianna, Gerlin, Pavel, Yuri, Yisi, Yumi, Luis Carlos.

A nuestro querido Fidel por permitirme estudiar en esta universidad en la que tanto he aprendido.

A todas aquellas personas lindas de que una forma u otra han estado presentes para ayudarme a salir adelante, a obtener buenos resultados, a ver la vida de una forma diferente.

Milagro

FORDES

Centro Coordinador para la Formación y el Desarrollo del Capital Humano

La Habana, 3 de junio de 2008

"Año 50 de la Revolución"

La realización de eventos representa otra de las concepciones estratégicas que ha implementado el Centro Coordinador para la Formación y el Desarrollo del Capital Humano, FORDES, en su propósito de gestionar el conocimiento dentro del Sistema MIC. Estos encuentros han funcionado como espacios de reflexión e intercambio en torno a las temáticas que conciernen al desempeño de nuestras entidades y han promovido dialécticamente las relaciones con otros centros y Universidades que se ocupan de las temáticas de interés.

En este sentido, el Comité Organizador del Evento aprovecha la ocasión y se vale de la presente para reconocer al estudiante de la UCI Sergio Orlando Boudy González, por su participación, en categoría de Ponente en el Evento Virtual, en la Séptima Edición del evento Semana Tecnológica en el tema "Desarrollo de una aplicación para la identificación inteligente de objetos."

Lic. Teresita Quintana Gómez

Directora de TI

FORDES

Ave. Independencia y 19 de Mayo, Ciudad de la Habana, Cuba
Telefax: (537) 8638020

Reconocimiento

A "Desarrollo y aplicación práctica de una aplicación inteligente para el control y seguimiento de trayectoria en tiempo real"

Por haber obtenido Relevante

Jornada Científica Estudiantil

se hace camino al andar

Msc. Melchor Gil Morel
Rector UCI

Dr. Alfng Ruiz Jhones
Vice-Rector UCI

Reinier Tan Matamoro
Presidente FEU UCI

Dado en La Habana a los 30 días del mes de Mayo de 2008

Datos del tutor

Javier Alexander León Martínez.

Graduado de Ingeniería automática. Graduado en el Instituto Superior Politécnico José Antonio Echeverría durante el año 2003.Profesor de la asignatura máquinas computadoras de la Universidad de Ciencias Informáticas (UCI) desde el año 2003.Investiga en los temas de robótica, Inteligencia Artificial aplicada al control, sistemas de mediciones de campos magnéticos y desarrollo de hardware para la automatización.

Correo electrónico: jleon@uci.cu

RESUMEN

El presente trabajo muestra el desarrollo de una aplicación para el control inteligente de un robot manipulador, dotado de visión artificial, para la identificación inteligente de objetos mediante reconocimiento de patrones utilizando redes neuronales artificiales.

Se utilizó el método de segmentación de imágenes basado en crecimiento de regiones por agrupación de pixeles para la identificación de los objetos.

El sistema fue probado de forma satisfactoria, identificando objetos con un alto porciento de certeza y determinando su ubicación en el área de trabajo[I].

PALABRAS CLAVES

Visión artificial, reconocimiento de patrones, red neuronal, área de trabajo.

Tabla de contenido

INTRODUCCIÓN

Desde hace varios años el hombre se ha empeñado en automatizar las tareas que requieran un comportamiento inteligente, con lo cual ha desarrollado máquinas que imitan rasgos de la inteligencia humana. Con este fin ha creado técnicas basadas en el análisis formal y estadístico del comportamiento humano ante diferentes problemas, aliviando así áreas tales como el control de sistemas, la planificación automática, la habilidad de responder a diagnósticos y a consultas de los consumidores, el reconocimiento de la escritura, el habla y los patrones.

En nuestra universidad se viene realizando un grupo de trabajos como parte de la línea de desarrollo de hardware, robótica y control inteligente, del Grupo de Investigación de Automática Aplicada, al cual está suscrito el presente trabajo.

Este grupo tiene como misión definir la estrategia investigativa, asesorar metodológicamente las investigaciones y obtener resultados investigativos en los temas de automática vinculados a los proyectos productivos y al desarrollo tecnológico de la universidad en esa rama, con el fin de garantizar la relación investigación-desarrollo a través de la organización de las investigaciones y de la socialización y aplicación de los resultados obtenidos.

Para la realización de la investigación a la que tributa los resultados de este trabajo, se recuperó un pequeño brazo robótico de seis grados de libertad, con la colaboración del Departamento de Control Automático del Instituto de Cibernética, Matemática y Física (ICIMAF[II]), y se diseñó además la interfaz de hardware apropiada para el control del mismo.

El objetivo del presente trabajo es desarrollar una aplicación que permita dotar de cierto grado de inteligencia a un robot manipulador, a partir del reconocimiento de patrones con redes neuronales, en imágenes digitales. La aplicación debe ser capaz de identificar objetos previamente definidos, y determinar su posición.

Puede reconocerse así el siguiente ***problema científico***: ¿Cómo dotar de cierto grado de inteligencia a un robot manipulador?

Ante la necesidad de dar respuesta al problema planteado, se desarrollará una aplicación capaz de dotar de visión artificial a un robot manipulador, planteándose como ***objeto de estudio*** el control de un

robot manipulador mediante el uso de técnicas de visión artificial y/o identificación inteligente de objetos mediante el reconocimiento de patrones, utilizando redes neuronales artificiales (RNA[III]).

Campo de acción: dotar de visión artificial a un robot manipulador, mediante la identificación inteligente de objetos.

En correspondencia con el problema propuesto, se plantea como ***objetivo general*** de este trabajo: desarrollar una aplicación que permita dotar de cierto grado de inteligencia a un robot manipulador.

Su alcance presupone dar respuesta a las siguientes ***preguntas científicas:***

1. ¿Cómo determinar objetos en imágenes digitales?
2. ¿Cómo utilizar redes neuronales para reconocer patrones?

Para responder las preguntas anteriores fue necesario realizar las siguientes ***tareas de investigación***:

- Hacer un levantamiento de trabajos similares realizadas hasta el momento.
- Identificación inteligente de objetos en imágenes digitales mediante RNA.
- Seleccionar la metodología, herramientas y tecnologías que satisfagan las necesidades del sistema.

El desarrollo de este trabajo estará organizado de la siguiente manera:

Capítulo 1: Fundamentación Teórica. Se introducen los principales conceptos relacionados con las redes neuronales, reconocimiento de patrones, tratamiento de imágenes y sistemas inteligentes. Además, se analizan las tendencias actuales y las tecnologías necesarias para el desarrollo del trabajo.

Capítulo 2: Soluciones Técnicas. Se explica el tipo de red neuronal utilizada, así como las cuatro fases por las que atraviesa el proceso para el tratamiento de una imagen[IV]. Se exponen las características que presentará el sistema como solución a los problemas planteados. Se analiza con mayor profundidad el objeto de estudio. Se crea el modelo de dominio, se hace la captura de requisitos y se definen los casos de uso del sistema.

Capítulo 3: Análisis y Diseño del sistema. Se presentan los diagramas de clases de análisis y diseño, los diagramas de interacción, así como la arquitectura y los patrones que se utilizan para el diseño.

Capítulo 4: Implementación. Se presentan los diagramas de componentes y de despliegue de la aplicación.

Capítulo 5: Estudio de Factibilidad. Se hace un Estudio de la Factibilidad del proyecto, dando a conocer los beneficios que puede ofrecer el mismo.

Finalmente se ofrece un glosario de términos para ayudar a la comprensión del lenguaje técnico utilizado en el trabajo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

El objetivo de este capítulo es dar a conocer los principales conceptos que están relacionados con las redes neuronales, reconocimiento de patrones, tratamiento de imágenes y sistemas inteligentes. Además se analizan las tecnologías y tendencias actuales que serán necesario utilizar para darle cumplimiento al objetivo de este trabajo de diploma.

1.2 Redes Neuronales.

1.2.1 Qué es una Red Neuronal Artificial.

Las “redes neuronales artificiales” o simplemente “redes neuronales” constituyen una de las áreas de la inteligencia artificial que ha despertado mayor interés en los últimos años. Este hecho se basa en que son potencialmente capaces de resolver problemas cuya solución por métodos convencionales resulta extremadamente difícil. La idea de las redes neuronales fue concebida originalmente como un intento de modelar la biofisiología del cerebro humano. La meta era crear un modelo capaz de emular el proceso de razonamiento humano. (1)

Las redes neuronales están compuestas de muchos elementos sencillos que operan en paralelo. El diseño de la red está determinado mayormente por las conexiones entre sus elementos, al igual que las conexiones de las neuronas cerebrales. Han sido utilizadas para la realización de funciones complejas en variados campos de aplicación. Hoy en día son empleadas para resolver problemas que son de difícil solución para sistemas computacionales comunes o para el ser humano. Su cualidad más sobresaliente es que ellas son capaces de ¨ aprender ¨. En lugar de programar una red, se les presenta una serie de ejemplos, a partir de los cuales ellas aprenden las relaciones principales que están implícitas en la base de entrenamiento. (1)

1.2.2 Características de las neuronas artificiales.

Las neuronas artificiales constan de una o más entradas llamadas dendritas y una salida o axón, imitando a sus homólogas biológicas, además de una función de activación que caracteriza matemáticamente el comportamiento de la misma. Las dendritas poseen un valor asociado conocido como peso, que caracteriza la importancia de esa entrada. (2) De modo que la función transferencial de una neurona artificial es:

$$y=f\left(\sum_{i=1}^{n} w_i x_i\right)$$

donde:

y: salida o axón

f: función de activación

n: número de entradas o dendritas

x_i: entrada o dendrita

w_i: peso asociado a la entrada x_i

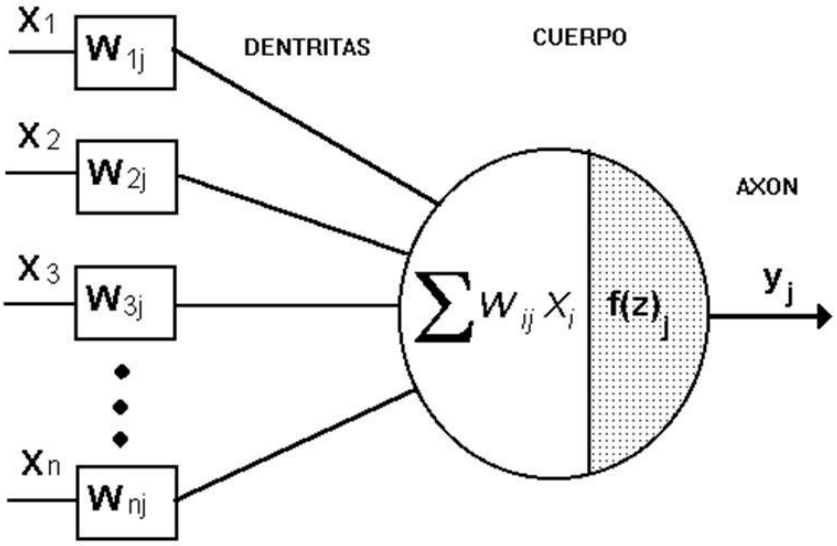


Fig.1 Esquema de una neurona artificial.

1.2.3 Tipos de aprendizaje.

Los tipos de aprendizaje pueden dividirse básicamente en tres, atendiendo ha como esta guiado este aprendizaje:

Aprendizaje supervisado: Se introducen valores de entrada a la red, y los valores de salida generados por esta se comparan con los valores de salida correctos. Si hay diferencias, se ajusta la red en consecuencia.

Aprendizaje de refuerzo: Se introducen valores de entrada, y lo único que se le indica a la red si las salidas que ha generado son correctas o incorrectas.

Aprendizaje no supervisado: No existe ningún tipo de guía. De esta manera lo único que puede hacer la red es reconocer patrones en los datos de entrada y crear categorías a partir de estos patrones. Así cuando se le entre algún dato, después del entrenamiento, la red será capaz de clasificarlo. (2)

1.2.4 Método de aprendizaje por retropropagación.

El procedimiento de retropropagación (Backpropagation) pertenece a la categoría de supervisado, pues requiere conocer las salidas correctas para cada ejemplo de entrada, es duro, ya que no es necesario dar información sobre las salidas de las unidades o capas intermedias.

Este procedimiento de aprendizaje ejecuta dos pasadas a través de la red. Durante la pasada hacia delante se aplica un patrón de entrada a la red con sus pesos actuales, que inicialmente poseen valores pequeños y aleatorios. Las salidas de todas las neuronas en cada nivel se calculan comenzando a partir de la capa de entrada y trabajando adelante en dirección a la capa de salida. La salida real de la red se compara con la salida deseada y se calcula el error. Seguidamente se ejecuta una pasada hacia atrás en la cual la derivada del error se propaga hacia atrás a través de la red, y todos los pesos son ajustados en proporción a su responsabilidad en el error de la salida. Esto se repite para todos los patrones ejemplos disponibles hasta que la red produzca la salida deseada.

La deficiencia más sobresaliente de este algoritmo de aprendizaje es que la superficie de error puede contener un mínimo local de modo que el gradiente descendente no garantiza encontrar un mínimo global. La experiencia con muchos trabajos ha mostrado que una red muy raramente cae en un mínimo local que sea significativamente peor que el mínimo global. Este comportamiento indeseable se ha encontrado en redes que tienen justamente las conexiones suficientes para resolver la tarea, por lo cual añadiendo más conexiones se crean dimensiones extras en el espacio de los pesos que ofrecen caminos para evitar estos mínimos locales indeseables. (2)

1.2.5 Perceptrón multicapa

La topología o arquitectura de una red consiste en la organización y disposición de las neuronas en la misma. Las neuronas se agrupan formando capas, las cuales se unen entre ellas formando redes neuronales.

Un perceptrón multicapa está formado por tres tipos distintos de elementos:

- Unidades sensoras, que constituyen la capa receptora/conversora de los estímulos externos, también conocida como capa de entrada.
- Unidades asociativas, en las cuales se realiza una combinación de las señales de entrada, conocida como capa(s) oculta(s).

- Unidades de respuesta, conocida como capa de salida. (2)

El perceptrón multicapa acepta valores reales. Las conexiones entre las neuronas son de tipo (feedforward) o encadenamiento hacia delante. (3)

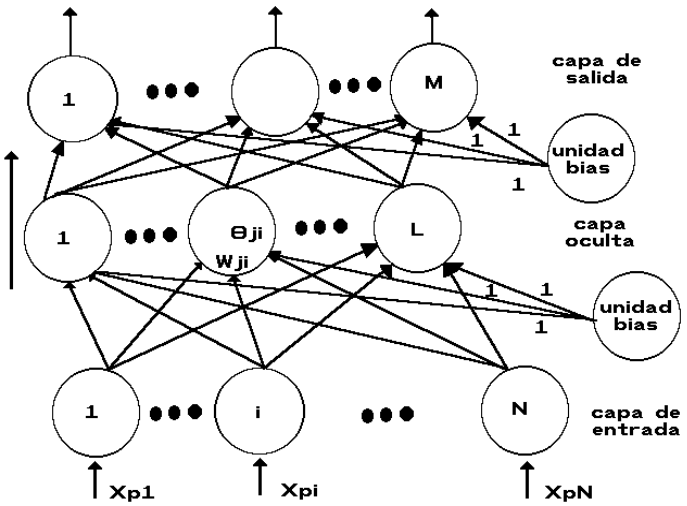


Fig. 2 Perceptrón multicapa (MLP).

En principio todas las unidades asociativas reciben señales desde las unidades sensoras. Las mismas toman las entradas desde el exterior, y sus salidas se dirigen como entradas a las unidades de asociación, en las que se realiza la combinación de estas a través de las respectivas funciones de activación. Las salidas de las unidades de asociación van dirigidas a las unidades de respuesta a través de caminos previamente caracterizados por sus respectivos pesos, y su(s) salida(s) constituyen la(s) salida(s) de la red neuronal. (2)

1.3 Reconocimiento de patrones

1.3.1 ¿Qué es Reconocimiento de Patrones?

El reconocimiento de patrones consiste en un sensor que recoge las observaciones a clasificar, un sistema de extracción de características que transforma la información observada en valores numéricos o simbólicos, y un sistema de clasificación o descripción que, basado en las características extraídas. (4)

1.4 Tratamiento de imágenes

El procesamiento digital de imágenes (PDI) es una de las ramas de la ciencia que más desarrollo ha experimentado en la última década, debido fundamentalmente a dos factores: primero, el avance vertiginoso que se ha originado en el *"hardware"*, particular, los procesadores especializados, los arreglos de procesadores y las computadoras digitales; y segundo, a la importancia que esta técnica ha adquirido en la vida moderna. (5)

El procesamiento de imágenes se divide en cuatro etapas fundamentales:

1. **Captura:** es donde se obtiene la imagen a procesar desde un dispositivo óptico conectado a la computadora.
2. **Filtrado:** en esta etapa se transforma la imagen a un formato más adecuado para facilitar posteriores procedimientos en el análisis de la escena (eliminación de información no relevante, aumento del contraste, aumento o disminución del rango dinámico, entre otros).
3. **Segmentación:** partición de la imagen en un conjunto de regiones no solapadas, homogéneas con respecto a algún criterio. Esta constituye la fase de mayor importancia antes de llevarse a cabo los pasos siguientes del análisis de la imagen. De su calidad depende en gran medida el resultado final del procedimiento.
4. **Reconocimiento:** tiene como objetivo principal detectar a partir de determinadas características conocidas, los patrones visuales presentes en una imagen. Un patrón no es más que una descripción estructural o cuantitativa de un objeto o de alguna otra entidad de interés en la imagen. (5)

En este trabajo, las etapas de filtrado y segmentación se realizan casi de manera simultánea, se va operando la imagen de forma puntual, aplicando primero el filtrado por escala de grises sobre la misma, y luego se segmenta por un método basado en la formación de regiones conocido como crecimiento por agrupación de pixeles. Para el reconocimiento fue utilizada una red neuronal del tipo perceptrón multicapa.

1.5 Visión artificial.

El término visión artificial o visión por computadora se refiere a capacidad de una máquina computadora de reconocer su entorno a partir de dispositivos que le permitan recolectar datos referentes al mismo, y de algoritmos capaces de interpretarlos convenientemente. Entre los más utilizados está el reconocimiento de patrones en imágenes digitales.

En Cuba se viene trabajando este tema en varias universidades y centros de investigación, con propósitos más o menos específicos. Aplicado a la robótica, solamente en la Universidad Central "Marta Abreu" de las Villas se han venido realizando una serie de trabajos para dos y tres dimensiones espaciales. (6) (7)

1.6 Metodologías, herramientas y tecnologías utilizadas.

1.6.1 Metodología de desarrollo de software.

Proceso Unificado de Desarrollo (RUP)

En la actualidad los desarrolladores se encuentran, a la hora de realizar una aplicación, con la necesidad de saber como organizar todo el proceso de desarrollo, como dar a cada desarrollador por separado, y al grupo en general, las tareas que le corresponden y así controlar los productos que se obtienen, por lo que es necesario aplicar una metodología capaz de dirigir estas actividades. (8)
Se escoge RUP por las características que presenta, de ser Orientada a Objetos, iterativo e incremental, o sea que va eliminando los errores cometidos en las iteraciones previas, centrada en la arquitectura y dirigido por casos de uso, logrando la obtención al final de todo su ciclo de vida, de un producto de calidad.

1.6.2 Herramienta Case

Rational Rose Enterprise Edition 2003

Se utilizará el Rational Rose Enterprise Edition 2003 como herramienta CASE (Computer-Aided Software Engineering), para la confección de los diagramas que se ilustran en este documento. Esta herramienta es muy completa y ofrece amplias potencialidades, capacidades de ingeniería inversa (crear modelo a partir código), diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad. (9)

1.6.3 Lenguaje de Modelado.

Lenguaje Unificado de Modelado UML

Utilizamos UML como lenguaje de modelado pues es un lenguaje gráfico para visualizar, especificar, modelar (analizar y diseñar) y documentar los artefactos de sistemas orientados a objetos. (9)

Dicho lenguaje es una notación unificada con la que se permite lograr un entendimiento que propicie el intercambio entre los usuarios y los desarrolladores. Se ha convertido en un estándar de la industria del software, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos, Grady Booch, Ivar Jacobson y Jim Rumbaugh. (10)

1.6.4 Lenguaje de programación.

Visual C#

C# es simple, eficaz y orientado a objetos. Con sus diversas innovaciones, C# permite desarrollar aplicaciones rápidamente y mantiene la expresividad y elegancia de los lenguajes de tipo C. (11)

Se escogió este lenguaje por las disímiles ventajas que presenta, dentro de las que se encuentran la recolección de basura automática. Al empezar a programar, se puede definir una o más clases dentro de un mismo espacio de nombres además, existe un rango más amplio y definido de tipos de datos. Es multithreaded (multihilos), permite muchas actividades simultáneas en un programa.

1.6.5 Herramienta de desarrollo.

Microsoft Visual Studio 2005

Para el desarrollo de la aplicación es necesario escoger la herramienta de desarrollo a utilizar según el lenguaje de programación que se seleccionó, por lo tanto se hará uso del IDE Microsoft Visual Studio 2005, pues permite al desarrollador disfrutar de un entorno de desarrollo altamente productivo con diseñadores visuales variedad de lenguajes de programación y editores de código mejorados.

Framework 2.0

Microsoft ha publicado el Microsoft .NET Micro Framework SDK 2.0 para desarrollar soluciones .NET y C# en dispositivos de funcionalidad reducida.

Para la realización de este trabajo se escogió .NET Framework 2.0 porque ofrece una gran cantidad de novedades y mejoras a todos los niveles que sin duda permitirán aumentar en gran medida la experiencia del usuario y la productividad del desarrollador. Además ofrece continuidad y mejoras en cuanto a seguridad, despliegue, administración y rendimiento. (12)

MATLAB Release 2006a

MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado y orientado para llevar a cabo proyectos o estudios donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. Este integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional.

Se escogió Matlab como otro de los lenguajes de programación para realizar el trabajo con las RNA ya que tiene un Toolbox para redes neuronales que proporciona funciones para el diseño, inicialización, simulación y entrenamiento de los modelos neuronales de uso más extendido de cada uno de los tipos de redes existentes. (13)

1.7 Conclusiones

En este capítulo se ha realizado un estudio del estado del arte del tema en cuestión, así como de los conceptos fundamentales en lo referente a las redes neuronales y al reconocimiento de patrones. Además de argumentar la selección de las metodologías, herramientas y tecnologías utilizadas.

CAPÍTULO 2: SOLUCIONES TÉCNICAS

2.1 Introducción

El objetivo de este capítulo es dar a conocer las soluciones de la aplicación, se describe el tipo de red neuronal empleado así como su topología. Se expone el proceso que se lleva a cabo para el tratamiento de las imágenes digitales. Se muestran además las características del sistema.

2.2 Procesamiento de Imágenes Digitales

Visión artificial o visión de máquina no es más que la adquisición, análisis y preparación automática de imágenes digitales, con el fin de extraer información relevante del entorno. (5)

El procesamiento de las imágenes, en aras de facilitar el trabajo y mejorar la eficiencia de nuestra aplicación, se ha dividido en 4 fases:

1. **Captura**

Se obtiene una imagen desde una cámara digital (webcam).

2. **Filtrado**

Se utiliza un filtro por escala de grises. Al aplicar el algoritmo se obtiene una imagen de 8 bits y 256 colores con el siguiente formato:

$W \; x \; H \; x \; 8$

Donde:

W: ancho de la imagen en pixeles.

H : largo de la imagen en pixeles.

3. **Segmentación**

Para la segmentación fue utilizado un método basado en la formación de regiones conocido como crecimiento de regiones por agrupación de pixeles. El mismo realiza la agrupación de regiones de modo que cada una de ellas se clasifica como objeto o como fondo, en dependencia de un umbral previamente definido. (5)

Se puede contemplar la segmentación como un proceso que divide a *R* en *n* subregiones, R_1, R_2,..., Rn, de tal forma que:

a) $\bigcup_{i=1}^{n} Ri = R$,
b) $R_i$es una región conexa, $i = 1,2, \dots, n$,
c) $R_i \cap R_j = \phi$ para todo i y $j, i \# j$,
d) $P(R_i) =$ VERDADERO para $i = 1,2, \dots, n$,
e) $P(R_i \cup R_j) =$ FALSO para $i \# j$

Donde *R* es la representación completa de una imagen, $P(R_i)$ es una propiedad de los puntos del conjunto R_i y ϕ es el conjunto vacío

Siendo R_i una región de tamaño *N*:

$$m = \frac{1}{N}\sum_{p \in R_i} f(p)$$

entonces se dice que la región es homogénea si,

$\max| f(p) - m| < T$, donde *T* es un umbral y *m* es la media. (5)

4. Reconocimiento

Se utiliza un perceptrón multicapa (MLP). Este tipo de red es el más utilizado para problemas de reconocimiento de patrones. Se definen los patrones “círculo”, “cuadrado” y “no se”, este último definido para el caso en que un objeto no pertenezca a ninguno de los patrones definidos anteriormente.

Al culminar el proceso de segmentación se obtiene un listado con los objetos encontrados, pero como el sistema esta diseñado para operar con objetos de diferentes tamaños, se hace necesario realizar un proceso de extracción de características, que en extraer la información extremadamente relevante de los objetos detectados. Es decir, aquella información que define de forma absoluta al objeto. Para normalizar la región se calcula el factor de relación según la ecuación (I) entre la matriz contenedora del objeto y una matriz estándar de dimensiones *nxm,* donde *n* = *m* = 20. Obteniendo en la ecuación

(IV) el objeto en forma de vector (V_{rna}), con tamaño fijo de (20) elementos y con un formato entendible por Matlab.

$$F_a = M_e / M_a \qquad \text{(I)}$$

$$P = \sum_{i=1,j=1}^{x,y} (M_a * F_a) \qquad \text{(II)}$$

$$V_{rna} = \sum_{i=1,j=1}^{m,n} (2^{P(i,j)}) \qquad \text{(III)}$$

$$V_{rna} = \sum_{i=1}^{n} (\frac{V_{rna}}{10}) \qquad \text{(IV)}$$

Donde:

F_a : factor de relación.

M_a : matriz contenedora del objeto.

M_e: matriz estándar de dimensión nxm (n=m=20).

V_{rna}: Vector que representa el objeto.

P: relación de transferencia entre M_a y M_e.

2.3 Redes Neuronales Artificiales

Para el reconocimiento de patrones se cuenta con una red neuronal implementada en Matlab. Después de varias pruebas se llego a la conclusión de que la topología más óptima es la utilizada ya que se probó con mayor número de capas ocultas donde el resultado fue el mismo pero con un mayor costo de procesamiento.

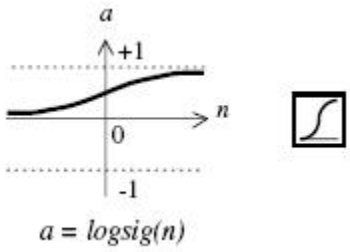


Fig. 3 Función de activación logarítmica.

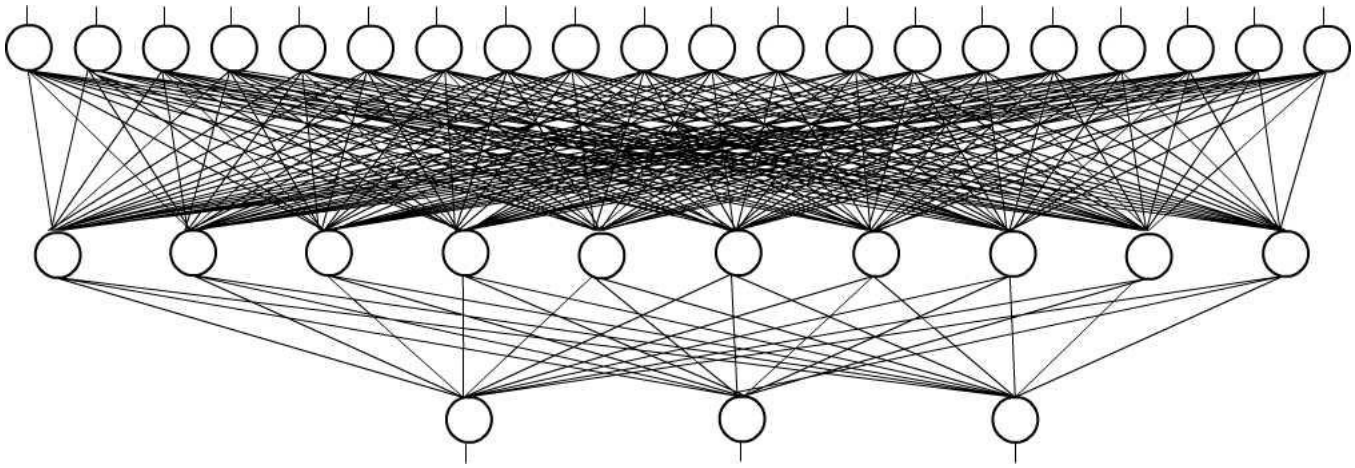


Fig. 4 Topología de la red neuronal empleada.

Se decidió implementar un perceptrón de 3 capas, siguiendo un criterio de selección práctico. El mismo cuenta con 20 neuronas en la capa de entrada, una capa oculta con 10 neuronas con función de activación logarítmica (Figura 3) (13), y una capa de salida con 3 neuronas (Figura 4). El método de entrenamiento utilizado es Reduced Memory Levenberg-Marquardt (trainlm) (13). Para el mismo se contó con 29 muestras entre los 3 patrones, por lo que se fue entrenada la red primeramente con 58 épocas y se obtuvo un error global de 0.0005, y posteriormente se realizo otro entrenamiento con 300 épocas donde se obtuvo un resultado bastante similar, el error fue de 0.0008 (ver anexo 2).

2.4 Sistema de visión del robot

Para dotar de visión al robot manipulador es preciso familiarizar al mismo con su entorno, en términos prácticos, esto no es más que calibrar la cámara. Para este fin se le muestra una hoja de papel con dos puntos del sistema de referencia de la cámara, situados a una distancia conocida. En la Figura 5

se muestra una representación del espacio de visión del robot (o sea el sistema de referencia de la cámara), donde ***C*** es el contorno que encierra el área de visión de la cámara, **R** es el punto de giro de la base del robot, y $\mathbf{P_1}$ y $\mathbf{P_2}$ son dos puntos conocidos alineados dentro del contorno ***C***.

$$d = \frac{(x_2 - x_1)}{2} \quad \text{(V)}$$

$$M = \sqrt{S^2 - d^2} \quad \text{(VI)}$$

$$x_R = x_1 + d \quad \text{(VII)}$$

$$y_R = y_1 - M \quad \text{(VIII)}$$

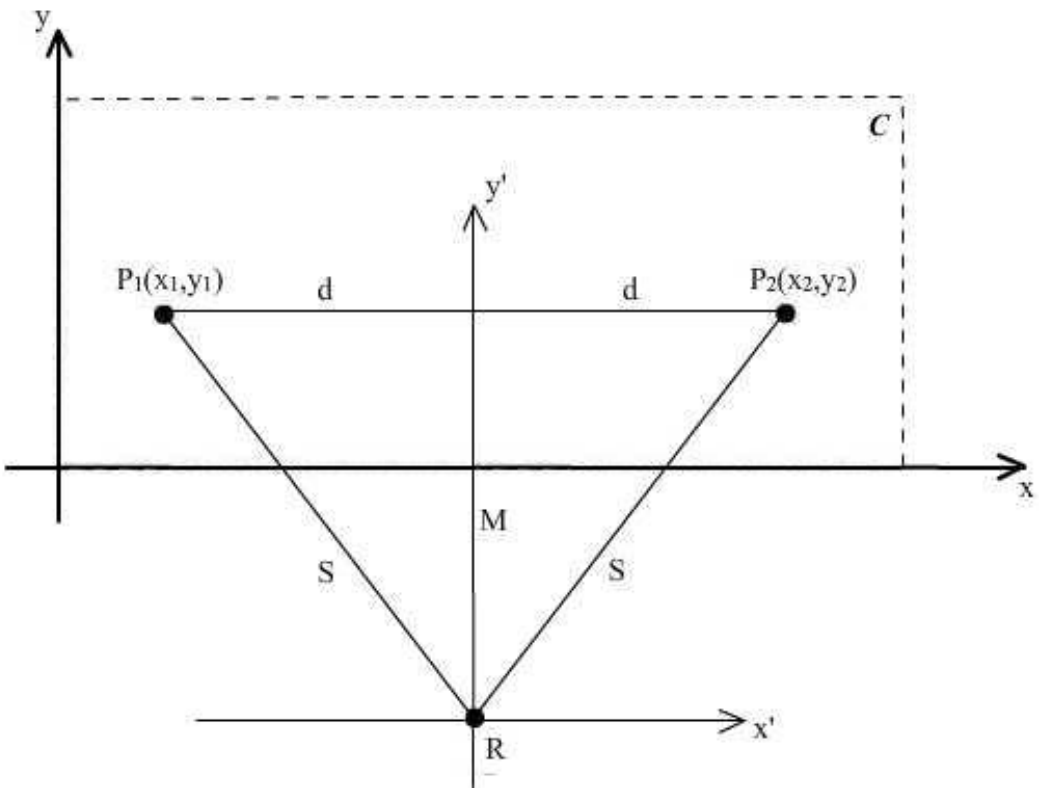


Fig. 5 Representación de los sistemas de referencia de la cámara y el robot respectivamente. $\mathbf{P_1}$ y $\mathbf{P_2}$ son los puntos de calibración de la cámara.

Es necesario sacar la relación de pixeles por centímetro, para convertir la posición del objeto de las unidades de la cámara a unidades físicas reales.

$$D = x_2 - x_1 \qquad \text{(IX)}$$

$$N_{p\,x\,cm} = \frac{D_p(P_n,P_{n+1})}{D_{cm}(P_n,P_{n+1})} \qquad \text{(X)}$$

Donde:

$N_{p\,x\,cm}$: cantidad de pixeles por cm

D : distancia entre dos puntos

D_p: distancia expresada en pixeles

D_{cm}: distancia expresada en cm

2.5 Características del sistema

2.5.1 Reglas del negocio a considerar

1. El formato de pixeles de la imagen debe ser superior a 24 bits.
2. Es necesario un buen funcionamiento de las redes neuronales que se van a utilizar para una correcta identificación.

2.5.2 Modelo de dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los entes que existen o los eventos que suceden en el entorno en el que trabaja el sistema. (9)

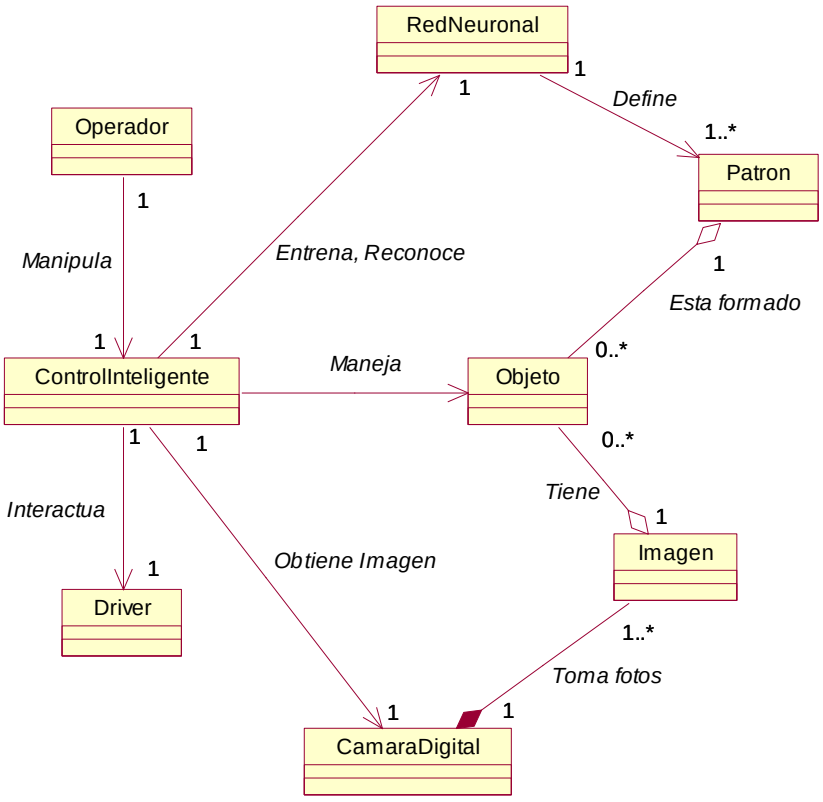


Fig.6 Modelo de dominio

Conceptos

Imagen: Matriz de píxeles que se obtiene de la cámara digital al tomar la fotografía.

Objeto: Son los entes materiales que va a manipular el sistema, representados por una agrupación de pixeles dentro de la imagen y su posición.

Patrón: Es un modelo predeterminado que es identificado por el sistema. En este caso los patrones son: círculo, cuadrado y no sé.

Operador: Persona encargada de la configuración y/o supervisión de la aplicación.

Red Neuronal: Las redes neuronales están compuestas de un gran número de elementos de procesamiento altamente interconectados (Neuronas) trabajando al mismo tiempo para la solución de problemas específicos. Estas se encargan de la identificación de los objetos.

Control Inteligente: Constituye el software, el cual se encarga del control y procesamiento del sistema.

Driver: Es una librería de funciones con la que interactúa el sistema, que se encarga de la comunicación con el robot.

Cámara Digital: Periférico o dispositivo encargado de recolectar la información de entrada del sistema (foto).

2.5.3 Especificación de los requisitos[V] del sistema

2.5.3.1 Requisitos funcionales[VI]

R_1 Accionar robot.

- $R_{1.1}$ Seleccionar patrón.
  - $R_{1.1.1}$ Identificar los objetos que correspondan con el patrón seleccionado.
  - $R_{1.1.2}$ Escoger los objetos identificados.
- $R_{1.2}$ Analiza el estado en que está el robot.
- $R_{1.3}$ Enviar datos que le permita accionar al robot.
- $R_{1.4}$ Esperar mensaje de notificación de fin de operación.

R_2 Tratar imagen.

- $R_{2.1}$ Captura.
- $R_{2.2}$ Filtrado.
- $R_{2.3}$ Segmentación.
- $R_{2.4}$ Reconocimiento.
  - $R_{2.4.1}$ Normalizar el tamaño de las regiones.
  - $R_{2.4.2}$ Identificar patrón.

R_3 Inicializar sistema

- $R_{3.1}$ Calibrar cámara.

$R_{3.1.1}$ Ajustar relación de píxeles por centímetros.

$R_{3.2}$ Posicionar robot con respecto a la imagen.

R_4 Entrenar la red neuronal artificial

R_5 Llenar base de conocimiento de la red neuronal

2.5.3.2 Requisitos no funcionales[VII]

Los requerimientos no funcionales son las propiedades o cualidades que el producto debe tener. Los mismos se identifican a continuación:

Requisitos de Software:

- El sistema operativo a utilizar es Windows XP, se necesita tener instalado Framework 2.0 y Matlab 7 o superior.

Requisitos de Diseño e implementación:

- Se implementa con los lenguaje de programación C# y Matlab, se rige por la filosofía de Programación Orientada a Objetos. El análisis y el diseño del sistema llevarán la metodología RUP, lenguaje de modelación UML apoyándose en la herramienta Rational Rose.

Requisitos de Apariencia:

- La aplicación deberá constar de una interfaz amigable, sencilla y profesional, que permita a los usuarios interactuar con facilidad con la misma.

Requisitos de Rendimiento:

- La herramienta propuesta debe ser eficiente, rápida y el tiempo de respuesta debe ser el mínimo posible.

Requisitos de Hardware:

- Como requisitos de hardware se requiere un disco duro de 10 GB o superior, 256 megabytes (MB) de memoria RAM o más.
- El robot debe contar con al menos una cámara digital (Webcam).

2.5.4 Modelo de Caso de Uso[VIII] del Sistema

Un modelo de casos de uso es un modelo del sistema que contiene actores, casos de uso y sus relaciones. (8)
En esta sección se reconocen los posibles actores y casos de uso a desarrollar.

Definición de los actores.

Actores	Justificación
Operador	Se encarga de entrar datos, realizar acciones o supervisar el sistema.

Diagrama de casos de uso.

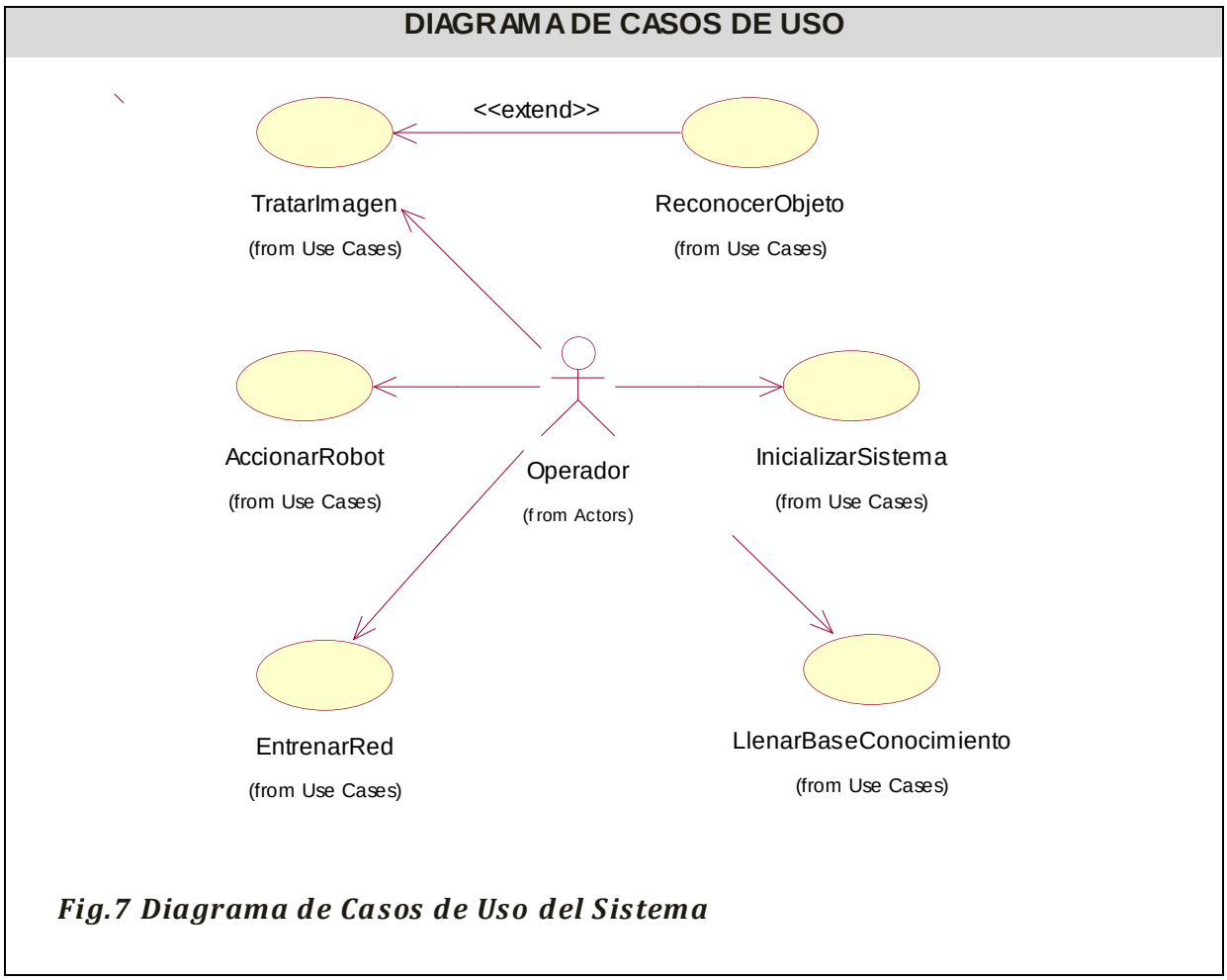


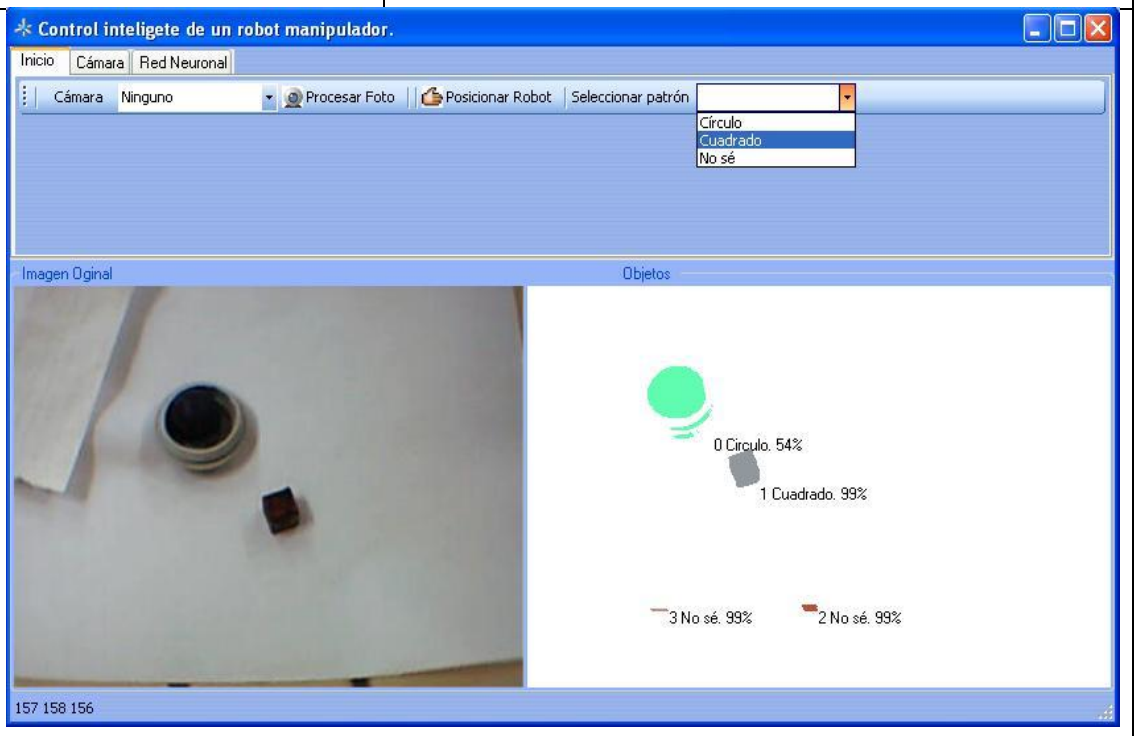
Fig.7 Diagrama de Casos de Uso del Sistema

2.5.5 Descripción de los Casos de Uso del Sistema

Referencia	Caso de uso	Prioridad
CUS1	AccionarRobot	Crítico
CUS2	TratarImagen	Crítico
CUS3	ReconocerObjeto	Crítico
CUS4	InicializarSistema	Crítico
CUS5	EntrenarRed	Crítico
CUS6	LLenarBaseConocimiento	Crítico

2.5.5.1 Descripción del Caso de Uso AccionarRobot

CUS1	AccionarRobot
Actores	Operador
Propósito	Extracción de objetos que cumplan con el patrón especificado por el Operador.
Resumen:	El caso de uso inicia cuando el Operador selecciona el patrón a extraer del área de trabajo. El sistema determina los objetos que cumplan con dicho patrón y ordena al robot su extracción. Termina cuando el robot haya extraído todos los objetos determinados.
Casos de uso asociados	
Referencias	R_1
Precondiciones	La existencia de al menos un objeto en el área de trabajo.
Poscondiciones	No debe quedar ningún objeto en el área de trabajo que cumpla con el patrón.
Flujo Normal de eventos	
Acción del actor	**Respuesta del sistema**

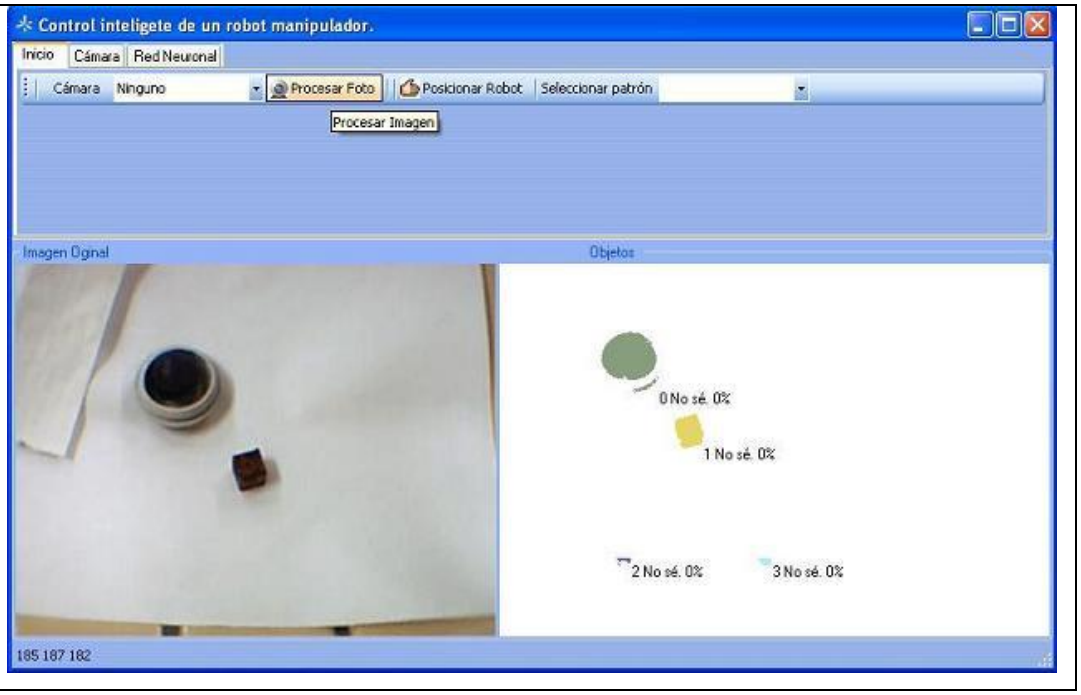
1 El Operador selecciona el patrón.	1.1 Se identifican los objetos que correspondan con el patrón seleccionado. 1.2 Analiza el estado en que está el robot, en caso de que este en estado de Listo realiza la acción 1.3. 1.3 Se escoge cada uno de los objetos identificados. 1.4 Son enviados los datos necesarios para accionar al robot. 1.5 Espera por un mensaje de notificación de fin de operación.
Flujo alternativo "Si el robot no está en estado de Listo"	
Acción del actor	**Respuesta del sistema**
	1.3.1 Si el robot no está listo, el sistema espera a que se encuentre en este estado.
Prototipo	

2.5.5.2 Descripción del Caso de Uso TratarImagen

CUS2	TratarImagen
Actores	Operador
Propósito	La obtención de objetos existentes en el área de trabajo.
Resumen:	El caso de uso se inicia cuando el Operador selecciona la opción Procesar Foto. Luego el sistema realiza los procesos de Captura (obtiene una imagen desde la cámara), Filtrado (se filtra la imagen), Segmentación (se adquieren los objetos existentes en la imagen).
Casos de uso asociados	ReconocerObjeto <<extendido>>
Referencias	R_2
Precondiciones	Debe tener al menos una cámara conectada.
Poscondiciones	Se adquieren los objetos dentro de la imagen.

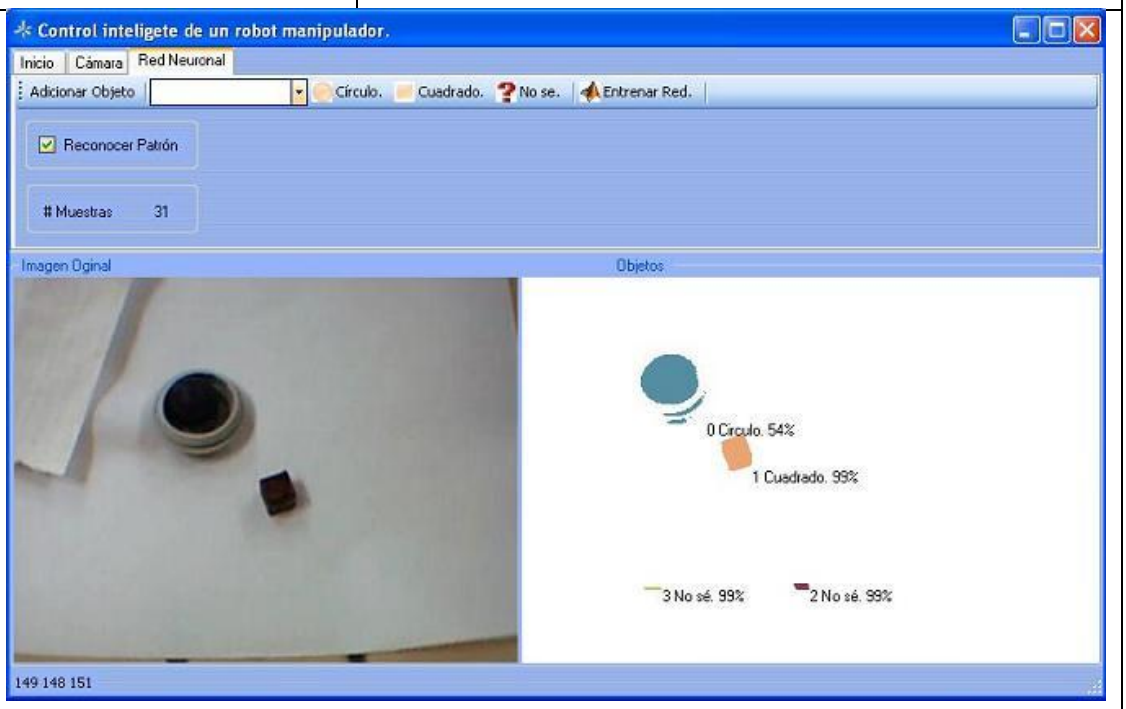
Flujo Normal de eventos	
Acción del actor	**Respuesta del sistema**
1 El Operador selecciona la opción Procesar Foto.	1.1 Se captura la imagen desde la cámara. 1.2 Se realiza el proceso de filtrado de la imagen. 1.3 Se segmenta la imagen filtrada, de esta forma se adquieren los objetos.

Flujo alternativo“”	
Acción del actor	**Respuesta del sistema**

Prototipo	

2.5.5.3 Descripción del Caso de Uso ReconocerObjeto

CUS3	ReconocerObjeto<<*extend*>>
Actores	Operador
Propósito	Identificar los objetos.
Resumen:	Este caso de uso se inicia a través del CUS TratarImagen. Se identifican los objetos obtenidos en el proceso de segmentación.
Casos de uso asociados	
Referencias	$R_{2.4}$
Precondiciones	El Operador debe haber activado la opción Reconocer Patrón.
Poscondiciones	Obtención de los objetos identificados con sus respectivos patrones.
Flujo Normal de eventos	
Acción del actor	**Respuesta del sistema**

1. El Operador activa la opción Reconocer Patrón.	1.1 Se normalizan las regiones a un formato y tamaño fijo para que sea entendible por la red neuronal. 1.2 Se envían los datos a la red neuronal para efectuar la identificación de patrones.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
Prototipo	

2.5.5.4 Descripción del Caso de Uso InicializarSistema

CUS4	InicializarSistema
Actores	Operador
Propósito	Inicializar el sistema

Resumen:	Este caso de uso se inicia cuando el Operador selecciona la opción Posicionar Robot. Se calibra la cámara y se especifican dos distancias desde el robot hasta el área de trabajo.
Casos de uso asociados	
Referencias	R_3
Precondiciones	El sistema debe tener al menos una cámara y el Operador debe mostrar a la cámara un papel calibrador.
Poscondiciones	El sistema quede inicializado.

Flujo Normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El Operador activa la opción Posicionar Robot. 2. Muestra el papel calibrador. 3 Especifica dos distancias del robot al área de trabajo.	1.1 Se muestra una interfaz que le permitirá al Operador calibrar la cámara. 2.1 Calcula la relación de pixeles por centímetro. 3.1 El sistema queda listo para trabajar.

Flujo alternativo	
Acción del actor	**Respuesta del sistema**

Prototipo	

2.5.5.5 Descripción del Caso de Uso EntrenarRed

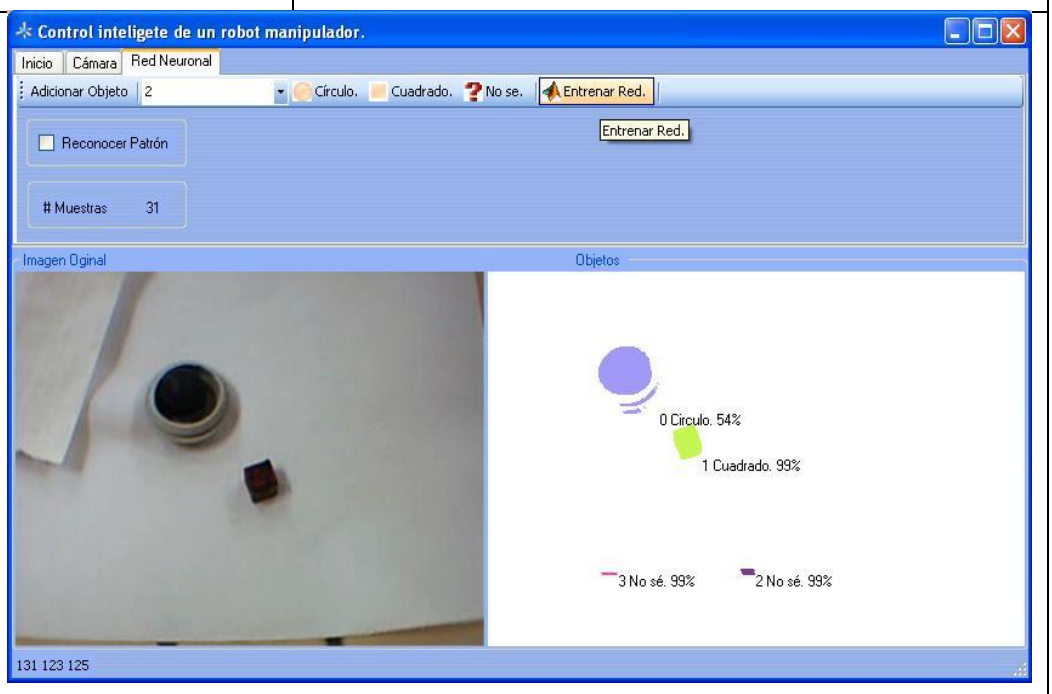
CUS5	EntrenarRed
Actores	Operador
Propósito	Entrenar la Red Neuronal.

Resumen:	El caso de uso inicia cuando el Operador selecciona la opción Entrenar Red. El sistema le ordena a Matlab que entrene la Red Neuronal.
Casos de uso asociados	
Referencias	R_4
Precondiciones	Que se encuentre instalada la versión de Matlab adecuada.
Poscondiciones	Queda entrenada la RNA.

Flujo Normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El Operador activa la opción Entrenar Red.	1.1 Se conecta con Matlab para entrenar la red.

Flujo alternativo	
Acción del actor	**Respuesta del sistema**

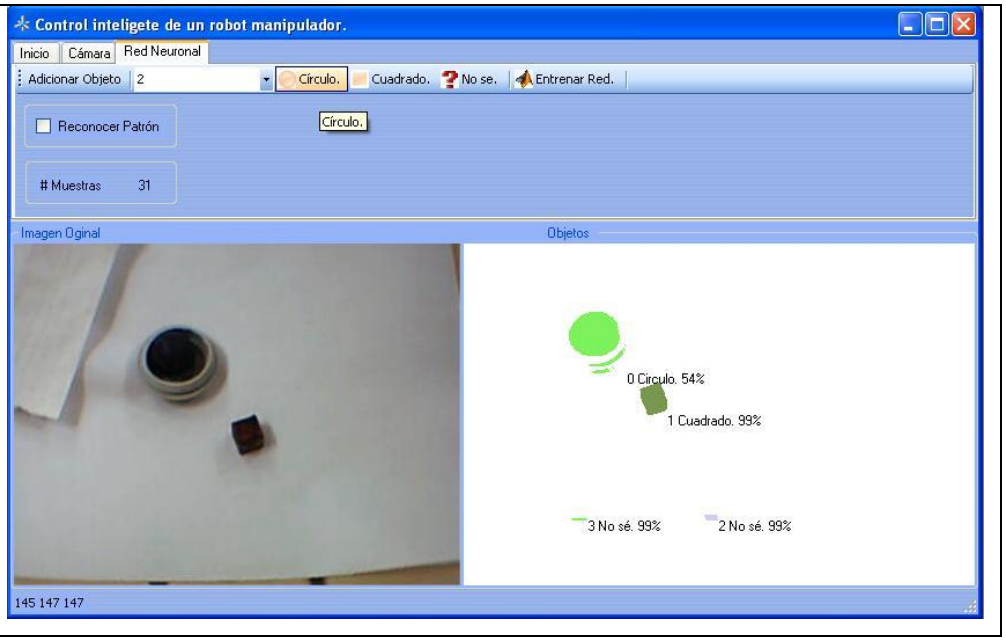
Prototipo



2.5.5.6 Descripción del Caso de Uso LlenarBaseConocimiento

CUS6	LlenarBaseConocimiento
Actores	Operador
Propósito	Llenar la base de conocimiento de la RNA para futuros entrenamientos.
Resumen:	El caso de uso inicia cuando el Operador selecciona la opción Seleccionar Objeto, y especifica a que patrón pertenece el objeto seleccionado. El sistema convierte dicho objeto con su patrón correspondiente a un formato que lo entienda la RNA y se le ordena a Matlab que lo adicione a la base de conocimiento.
Casos de uso asociados	
Referencias	R_5
Precondiciones	Que se encuentre instalada la versión de Matlab adecuada. El usuario debe haber seleccionado un objeto e identificado el patrón al que pertenece.
Poscondiciones	No debe quedar ningún objeto en el área de trabajo que cumpla con el patrón.

Flujo Normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El Operador selecciona el objeto. 2. Se indica el patrón al que pertenece.	1.1Se selecciona el objeto. 2.1 Se convierte la información a un formato entendible por la RNA. 2.2 Se le ordena a Matlab que lo adicione a la base de conocimiento de la RNA.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**

Prototipo	

2.6 Conclusiones

En este capítulo se han expuesto los métodos específicos empleados en la resolución del problema científico planteado. Se han descrito los algoritmos y métodos matemáticos utilizados en las diferentes fases que comprenden el proceso de reconocimiento de patrones, y aquellos empleados para la interacción entre el sistema y el robot. Son descritos además las reglas del negocio, los requisitos tanto funcionales como no funcionales, el modelo de dominio, los casos de uso del sistema, así como sus prototipos de interfaz.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

El objetivo de este capítulo es el desarrollo del flujo de trabajo de análisis y diseño del sistema. La realización de los diagramas de clases[IX] del análisis y diseño, así como los diagramas de interacción. Además se define la arquitectura y los patrones a utilizar en la aplicación para una mejor organización del sistema.

3.2 Análisis del sistema

3.2.1 Diagramas de Clases del Análisis.

Una clase del análisis representa una abstracción de una o varias clases y/o subsistemas del diseño. Esta posee varias características, como por ejemplo que se centra en el tratamiento de los requisitos funcionales y pospone los no funcionales. (8)

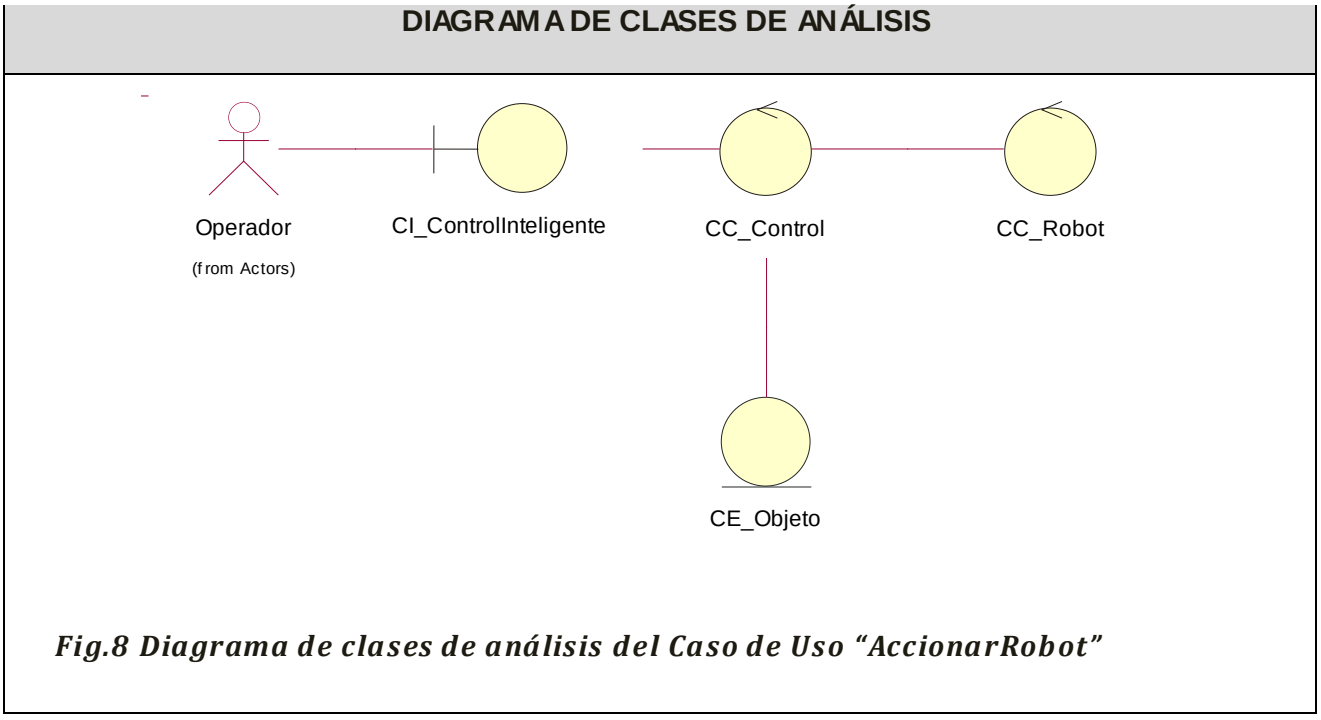


Fig.8 Diagrama de clases de análisis del Caso de Uso "AccionarRobot"

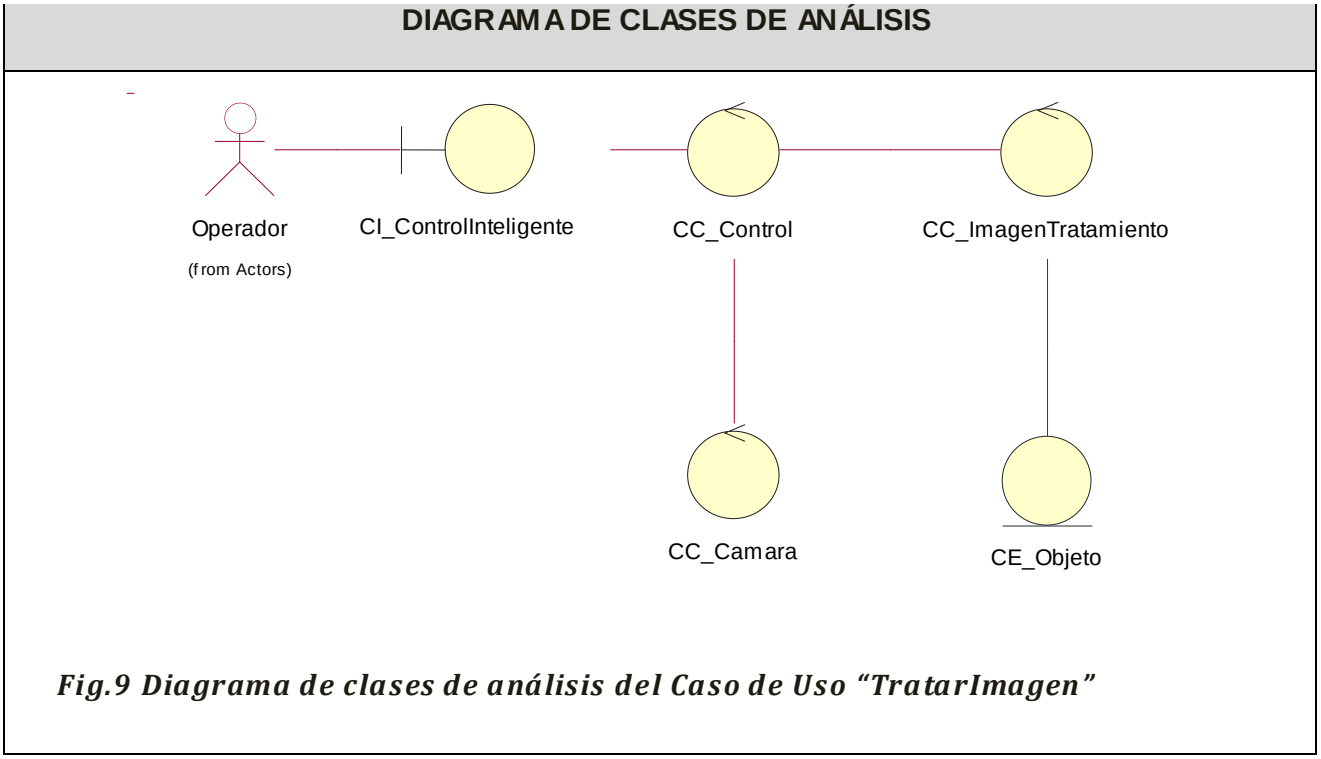


Fig.9 Diagrama de clases de análisis del Caso de Uso "TratarImagen"

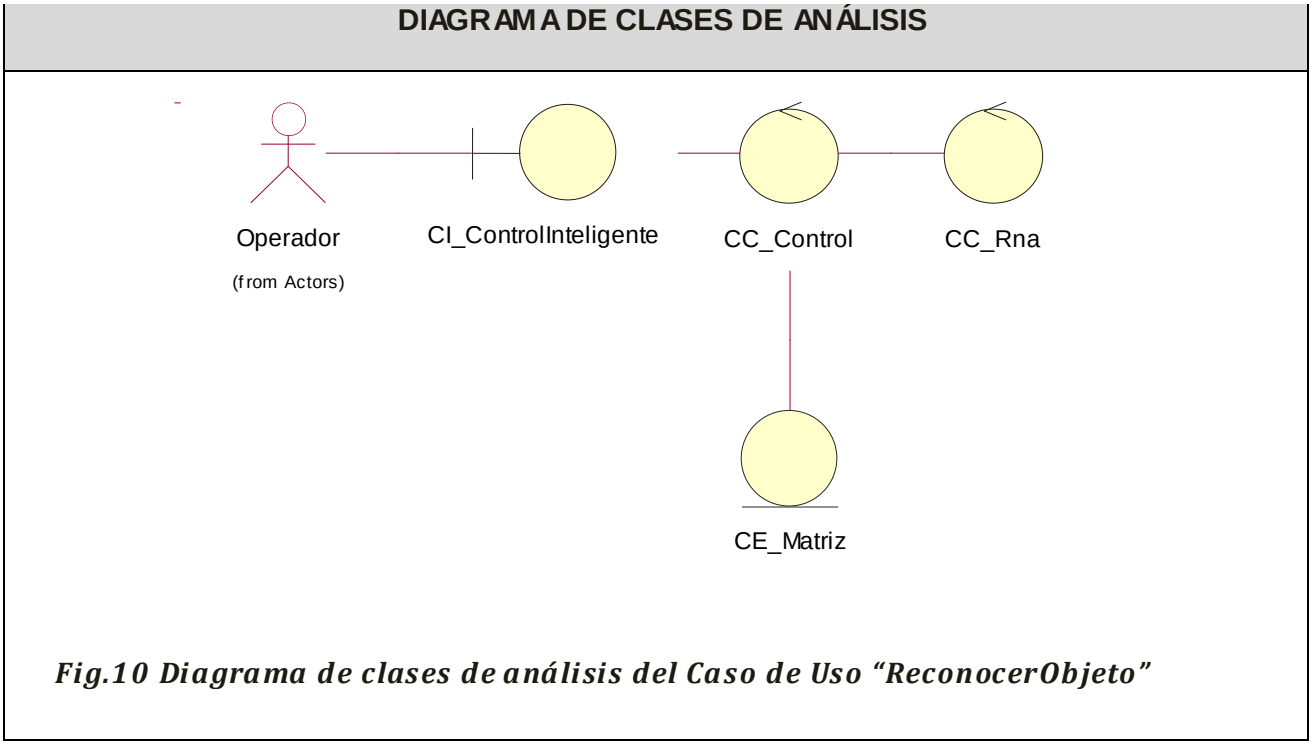


Fig.10 Diagrama de clases de análisis del Caso de Uso "ReconocerObjeto"

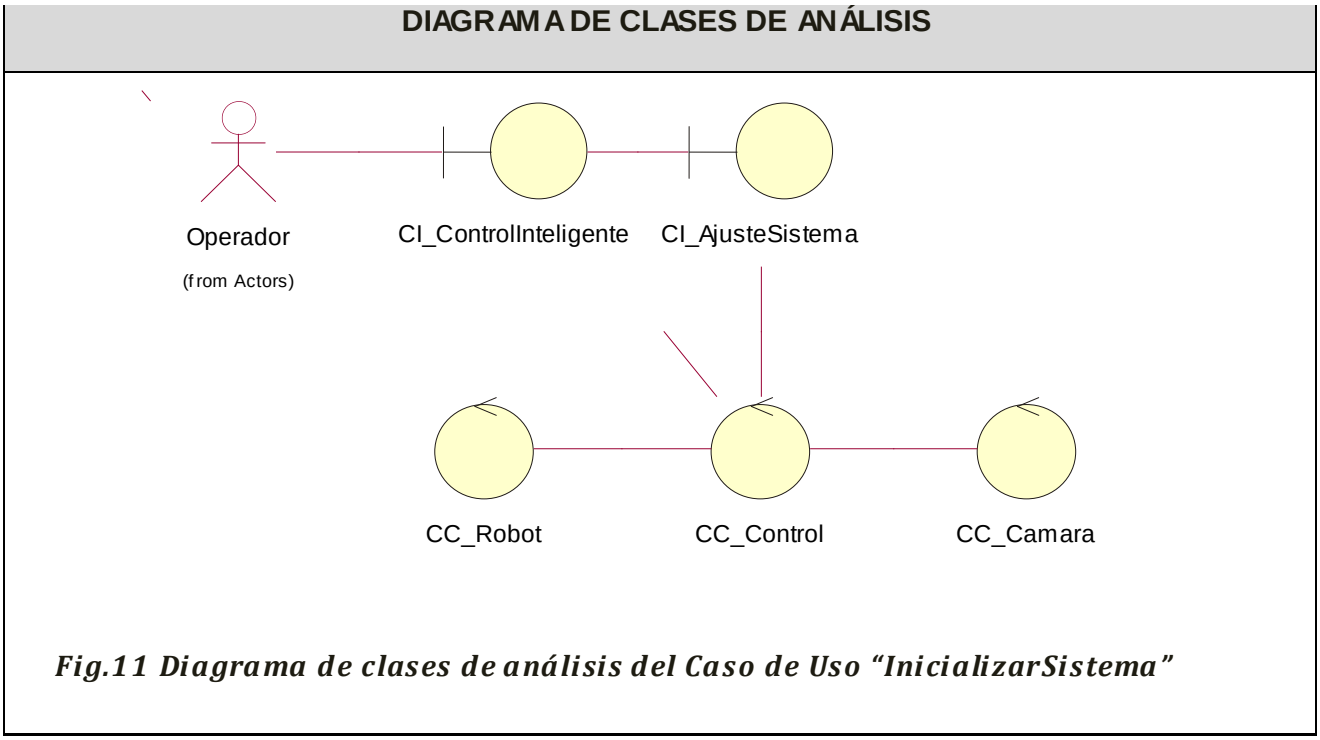


Fig.11 Diagrama de clases de análisis del Caso de Uso "InicializarSistema"

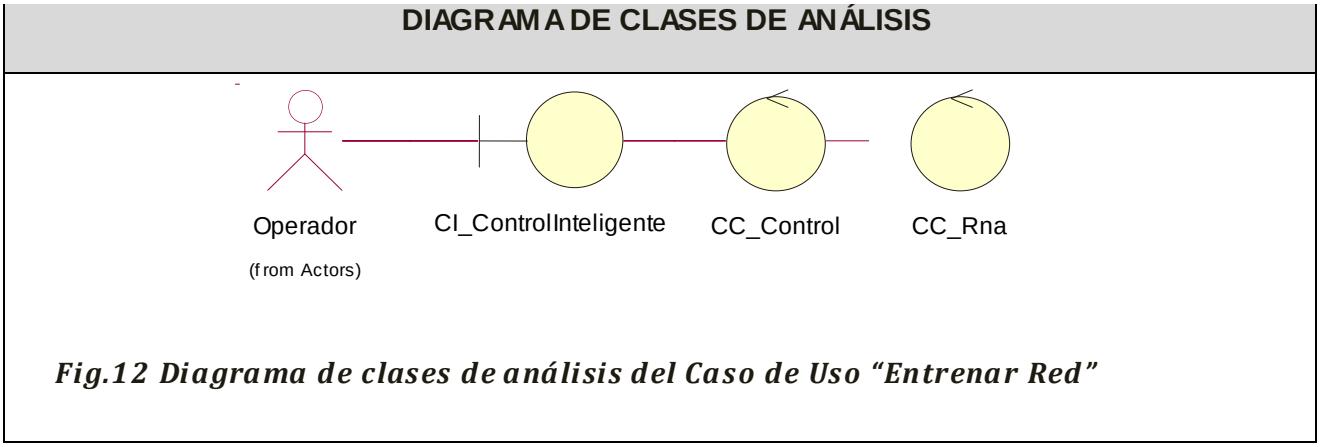


Fig.12 Diagrama de clases de análisis del Caso de Uso "Entrenar Red"

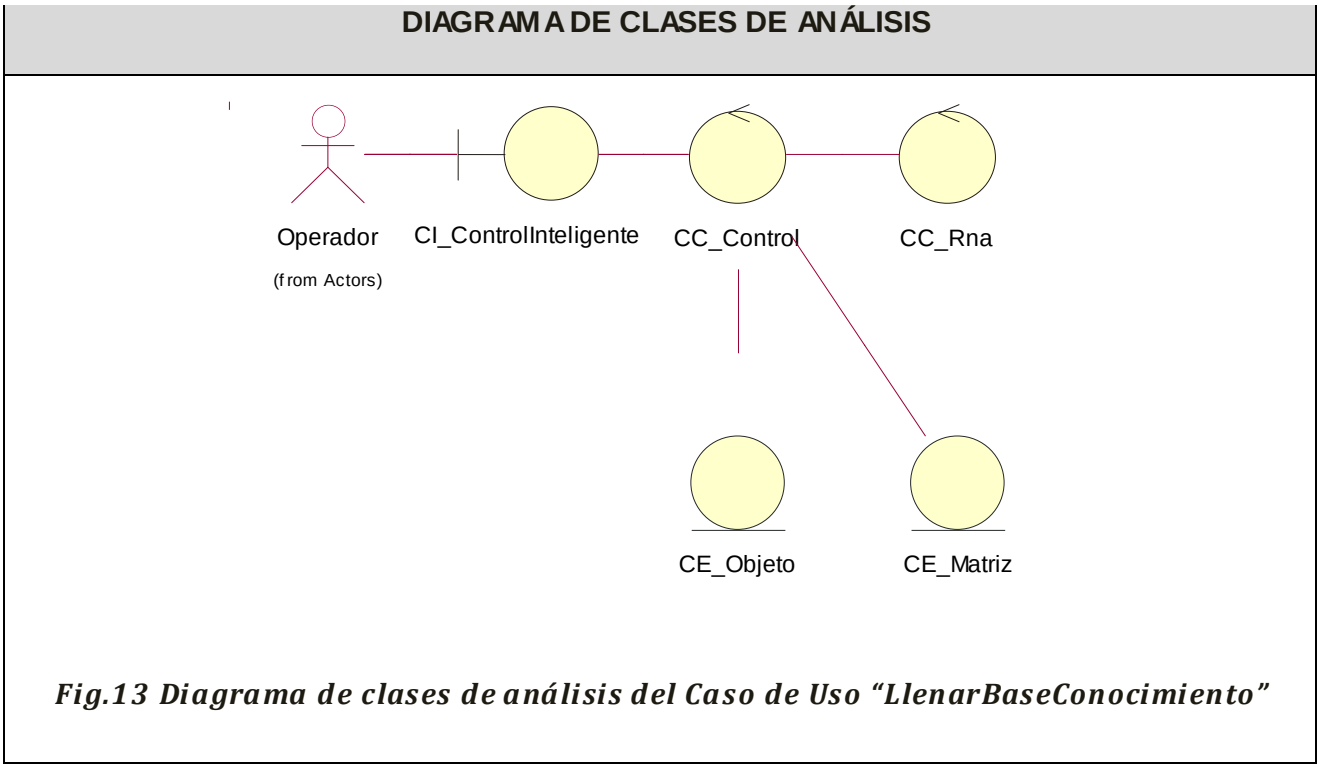


Fig.13 Diagrama de clases de análisis del Caso de Uso "LlenarBaseConocimiento"

3.2.2 Diagramas de interacción del análisis.

3.2.2.1 Diagramas de colaboración

DIAGRAMA DE COLABORACIÓN

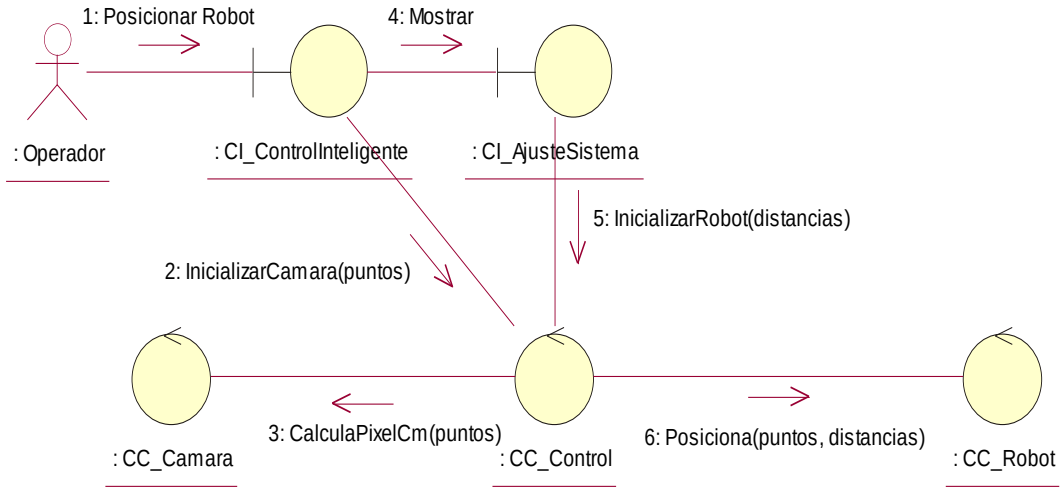


Fig.14 Diagrama de colaboración del Caso de Uso "InicializarSistema"

DIAGRAMA DE COLABORACIÓN

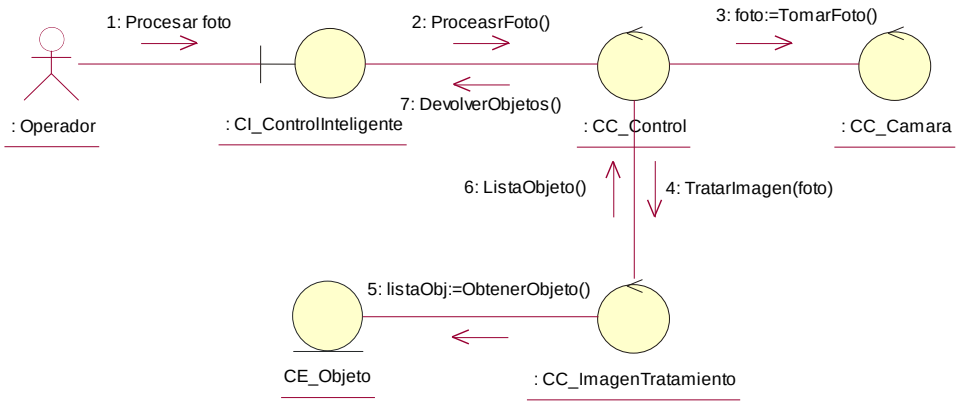


Fig.15 Diagrama de colaboración del Caso de Uso "TratarImagen"

DIAGRAMA DE COLABORACIÓN

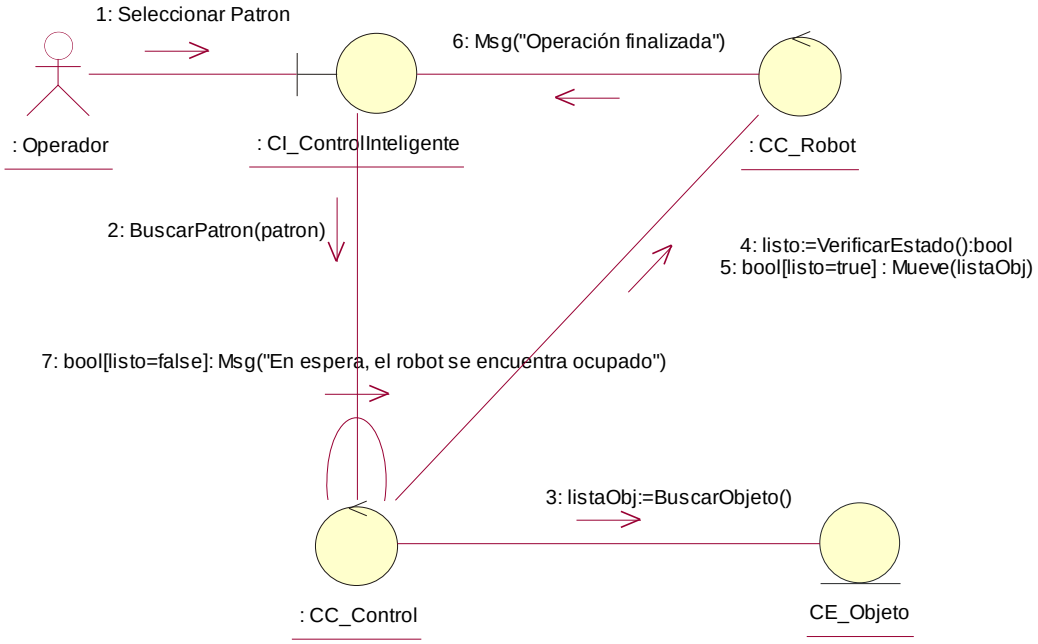


Fig.16 Diagrama de colaboración del Caso de Uso "AccionarRobot"

DIAGRAMA DE COLABORACIÓN

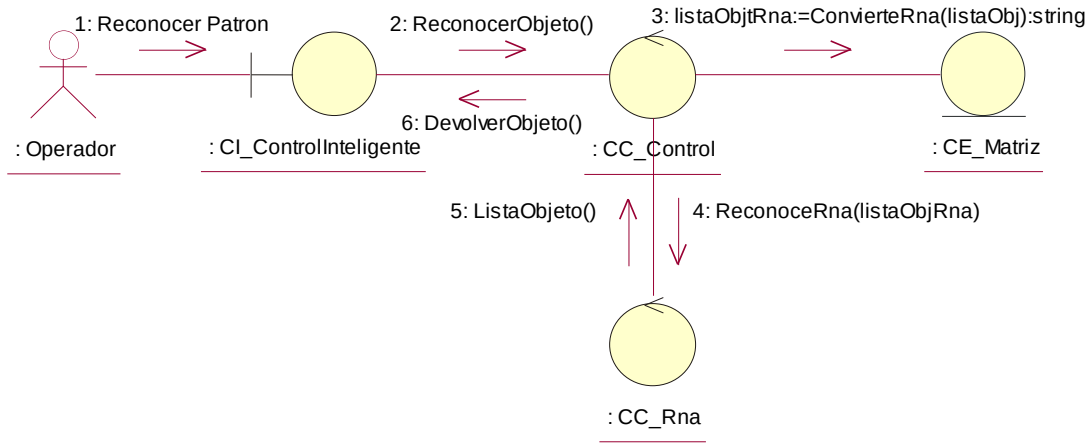


Fig.17 Diagrama de colaboración del Caso de Uso "ReconocerObjeto"

DIAGRAMA DE COLABORACIÓN

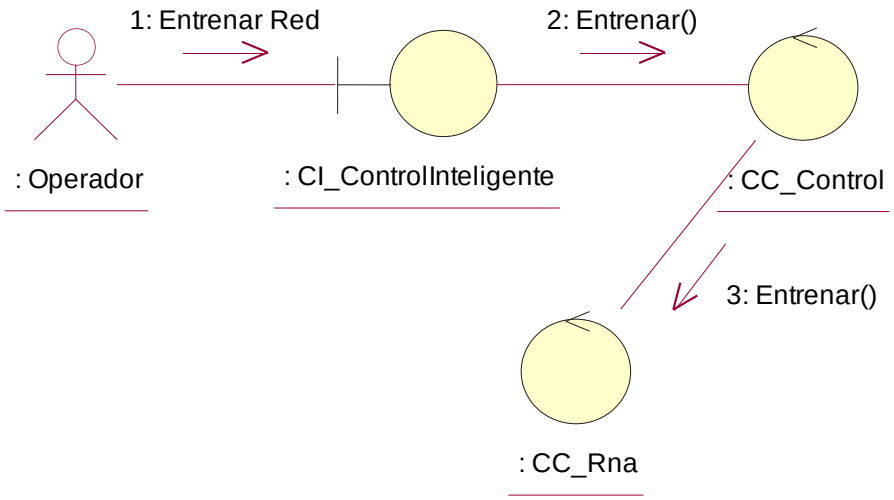


Fig.18 Diagrama de colaboración del Caso de Uso "EntrenarRed"

DIAGRAMA DE COLABORACIÓN

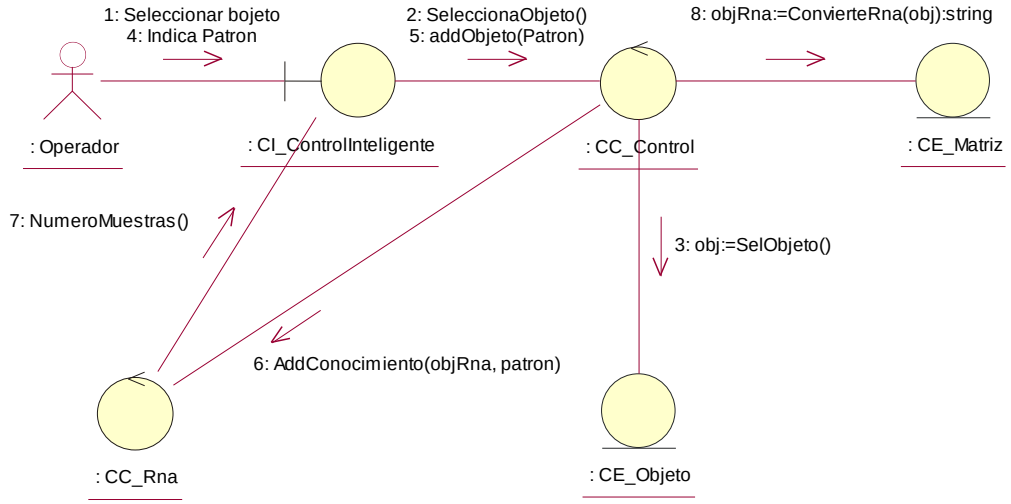


Fig.19 Diagrama de colaboración del Caso de Uso "LlenarBaseConocimiento"

3.2.2.2 Diagramas de secuencia del análisis.

DIAGRAMA DE SECUENCIA

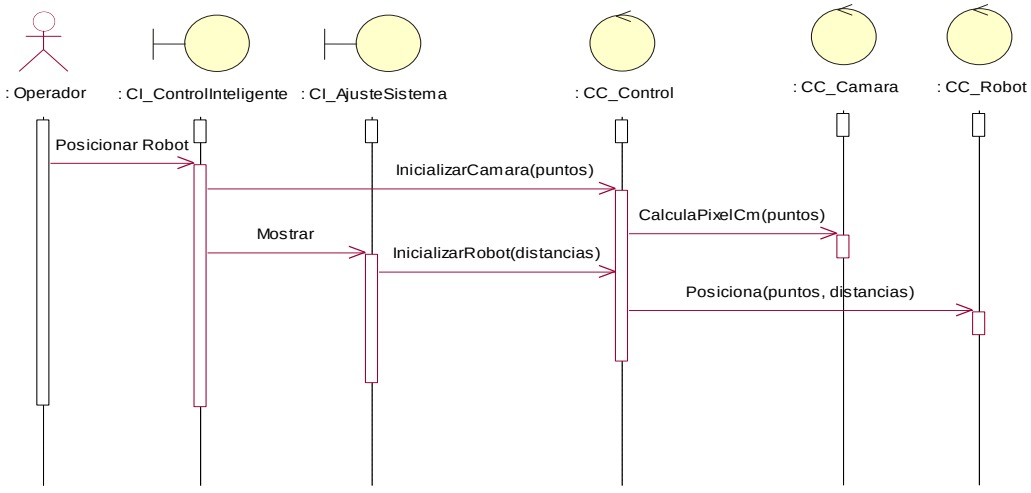


Fig.20 Diagrama de secuencia del Caso de Uso "InicializarSistema"

DIAGRAMA DE SECUENCIA

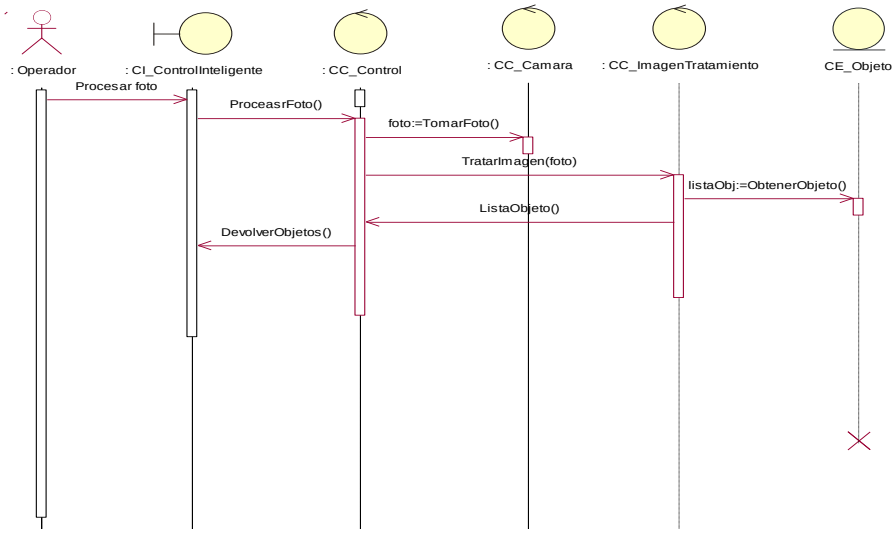


Fig.21 Diagrama de secuencia del Caso de Uso "TratarImagen"

DIAGRAMA DE SECUENCIA

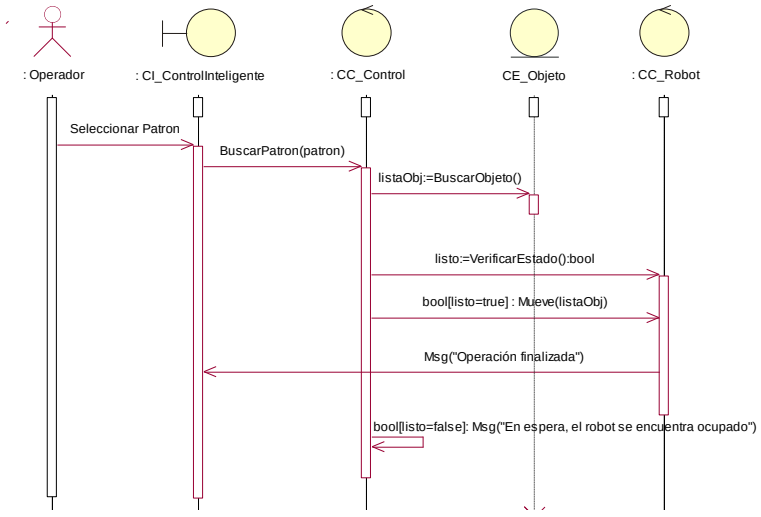


Fig.22 Diagrama de secuencia del Caso de Uso "AccionarRobot"

DIAGRAMA DE SECUENCIA

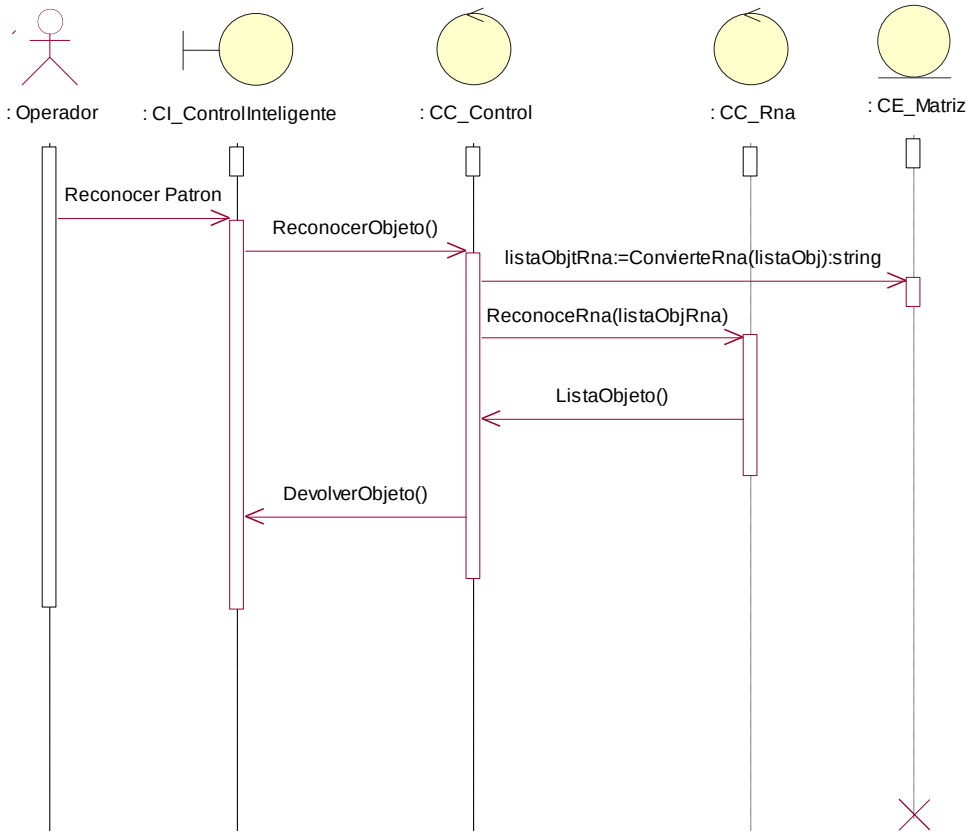


Fig.23 Diagrama de secuencia del Caso de Uso "ReconocerObjeto"

DIAGRAMA DE SECUENCIA

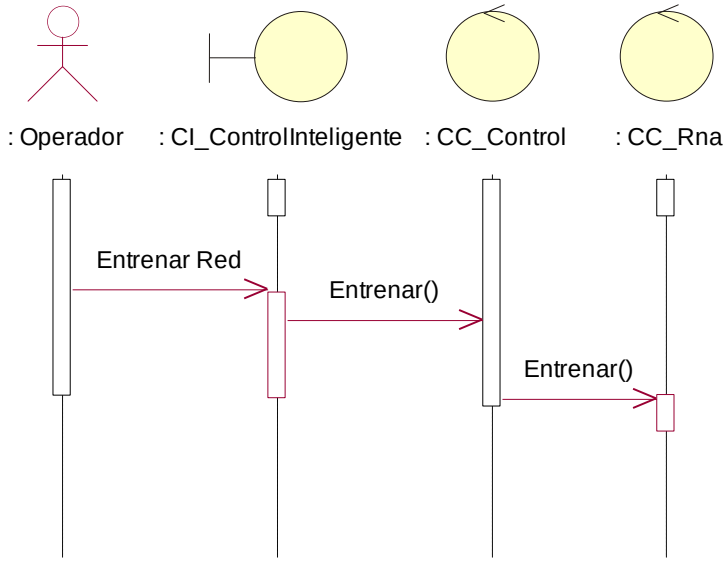


Fig.24 Diagrama de secuencia del Caso de Uso "EntrenarRed"

DIAGRAMA DE SECUENCIA

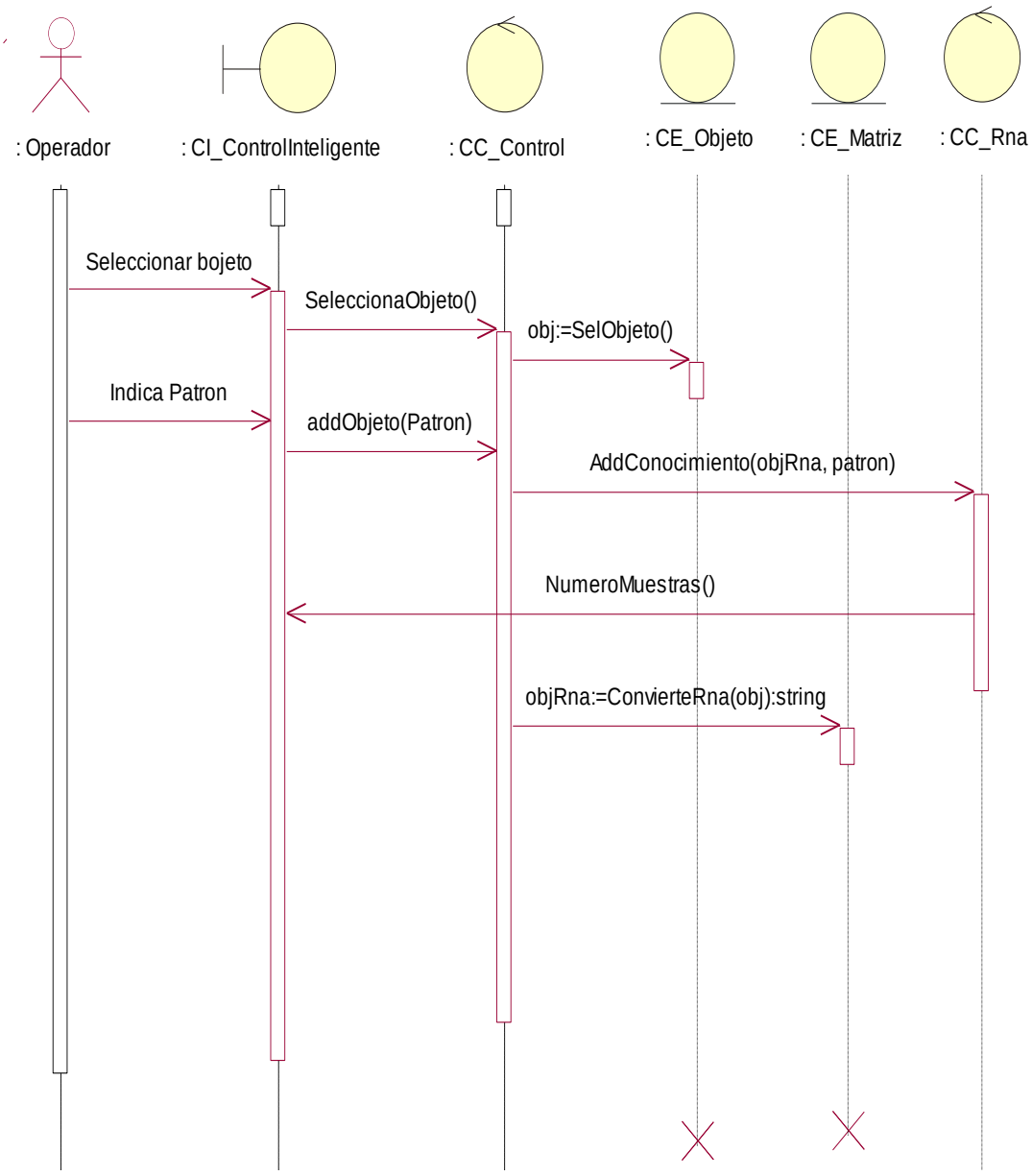


Fig.25 Diagrama de secuencia del Caso de Uso "LLenarBaseConocimiento"

3.3 Diseño del sistema

3.3.1 Diagramas de interacción del diseño.

DIAGRAMA DE COLABORACIÓN

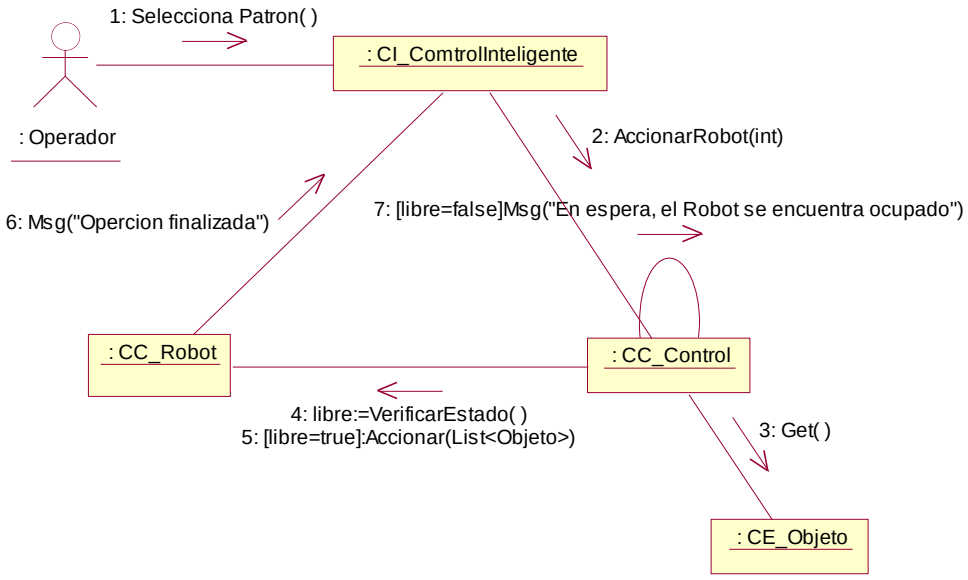


Fig.26 Diagrama de colaboración del Caso de Uso "AccionarRobot"

DIAGRAMA DE COLABORACIÓN

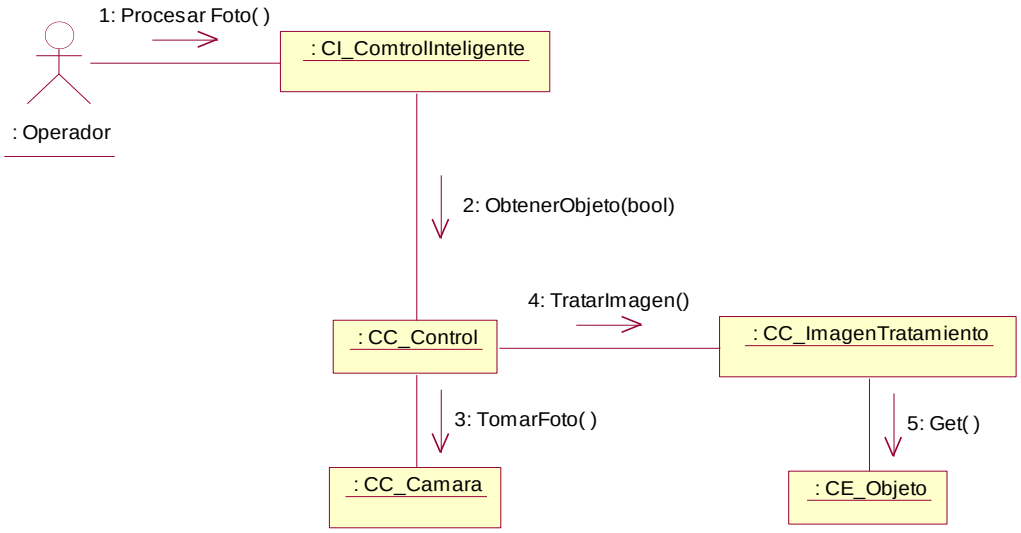


Fig.27 Diagrama de colaboración del Caso de Uso "TratarImagen"

DIAGRAMA DE COLABORACIÓN

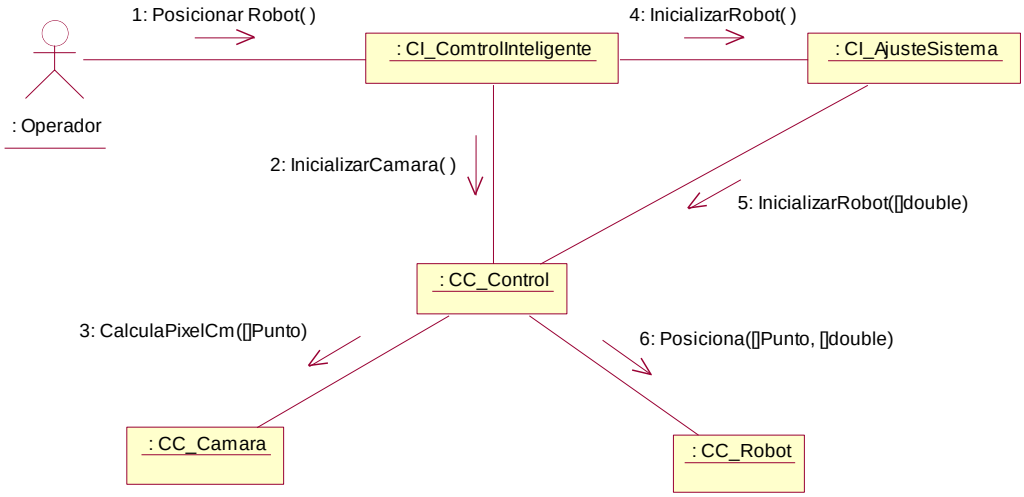


Fig.28 Diagrama de colaboración del Caso de Uso "InicializarSistema"

DIAGRAMA DE COLABORACIÓN

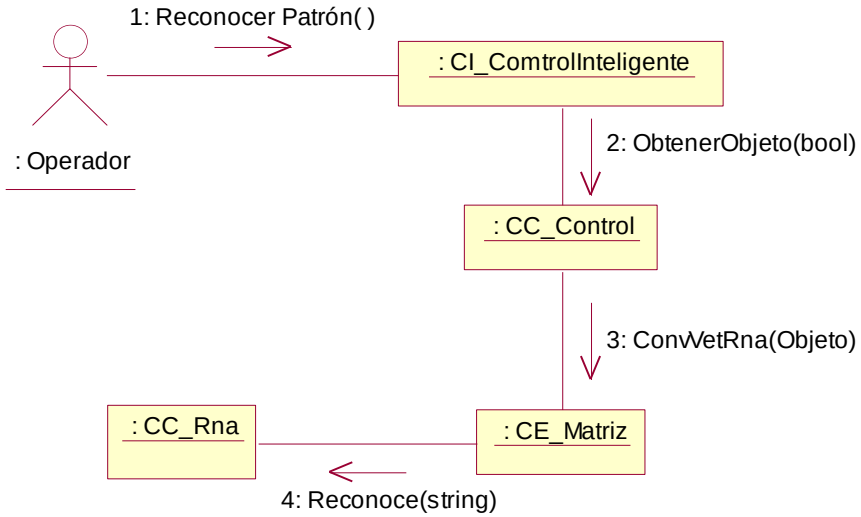


Fig.29 Diagrama de colaboración del Caso de Uso "ReconocerObjeto"

DIAGRAMA DE COLABORACIÓN

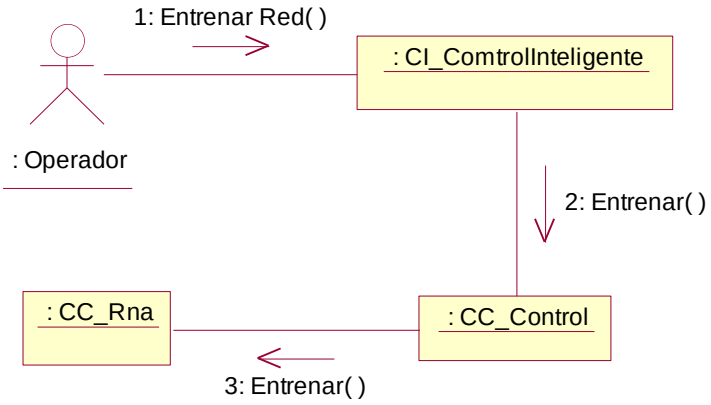


Fig.30 Diagrama de colaboración del Caso de Uso "EntrenarRed"

DIAGRAMA DE COLABORACIÓN

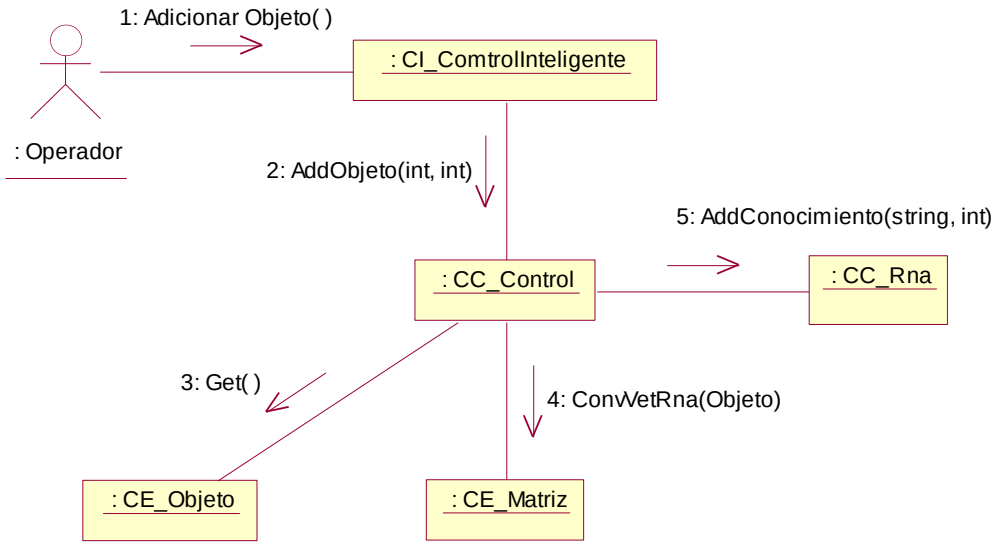


Fig.31 Diagrama de colaboración del Caso de Uso "LlenarBaseConocimiento"

3.3.2 Diagramas de secuencia de diseño.

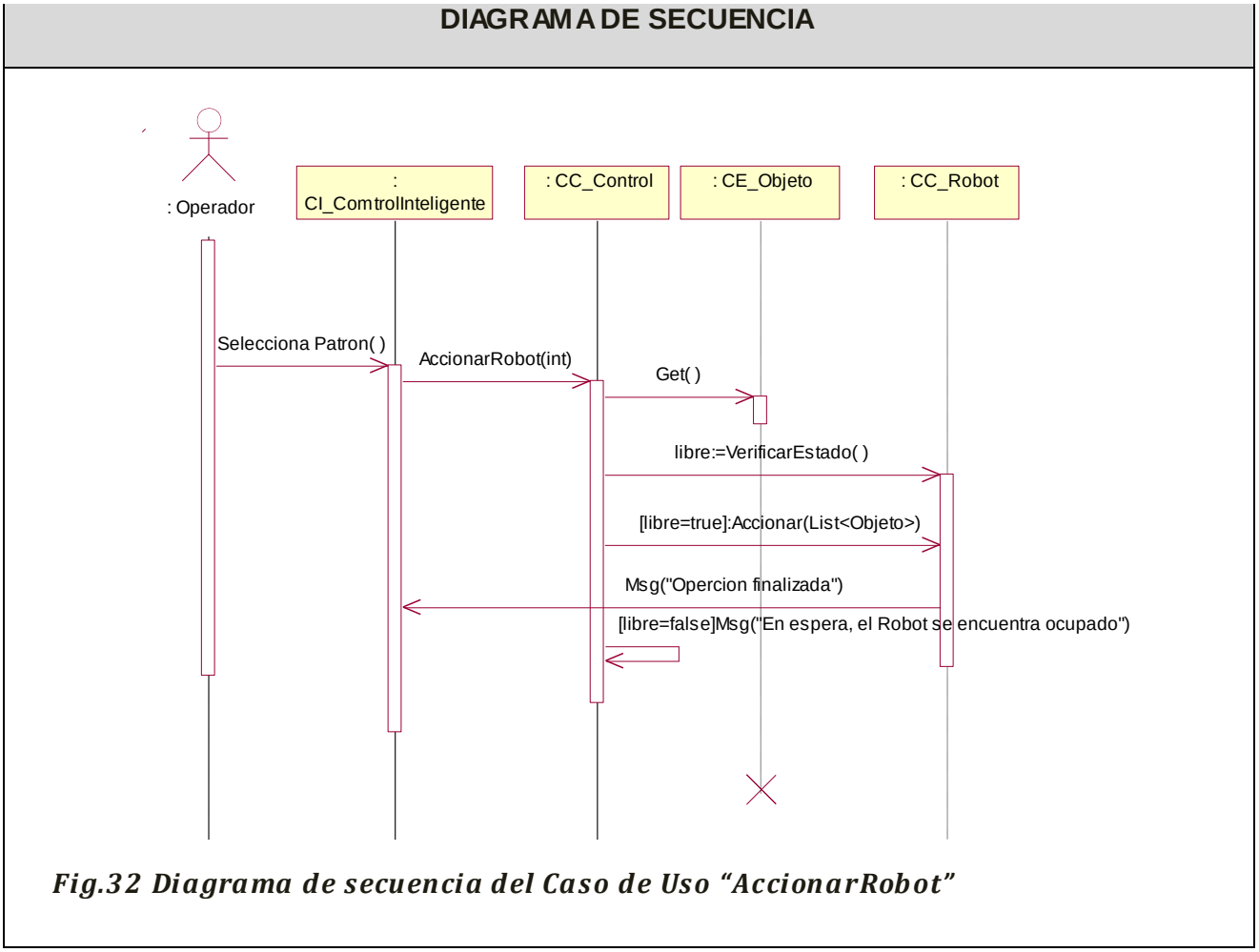


Fig.32 Diagrama de secuencia del Caso de Uso "AccionarRobot"

DIAGRAMA DE SECUENCIA

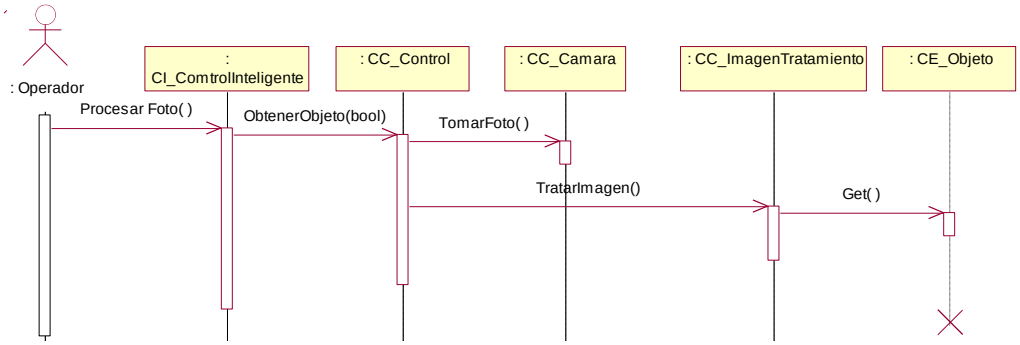


Fig.33 Diagrama de secuencia del Caso de Uso "TratarImagen"

DIAGRAMA DE SECUENCIA

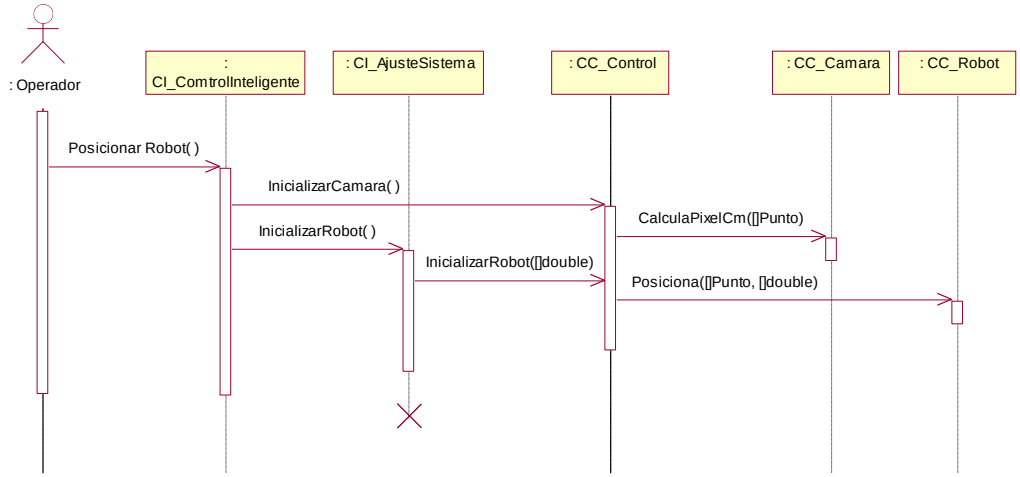


Fig.34 Diagrama de secuencia del Caso de Uso "InicializarSistema"

DIAGRAMA DE SECUENCIA

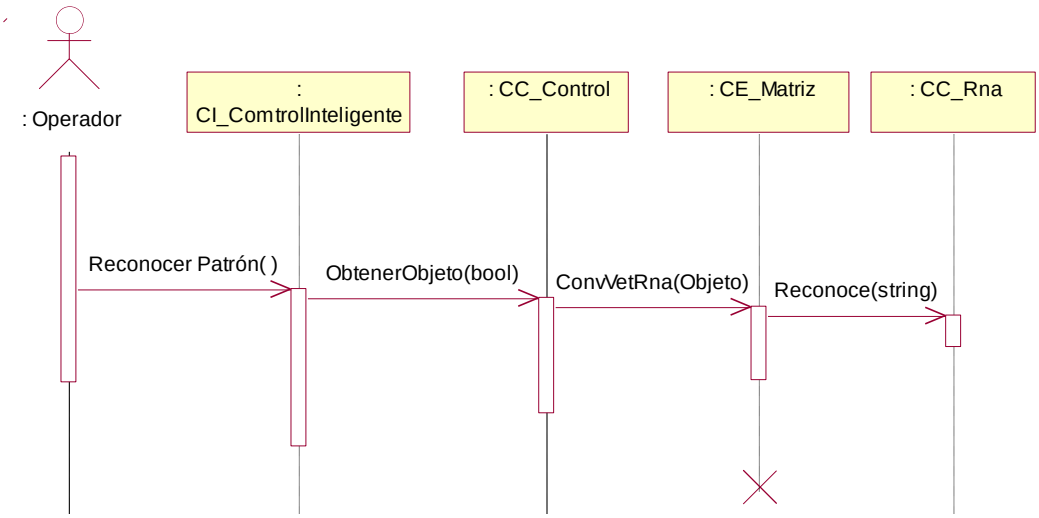


Fig.35 Diagrama de secuencia del Caso de Uso "ReconocerObjeto"

DIAGRAMA DE SECUENCIA

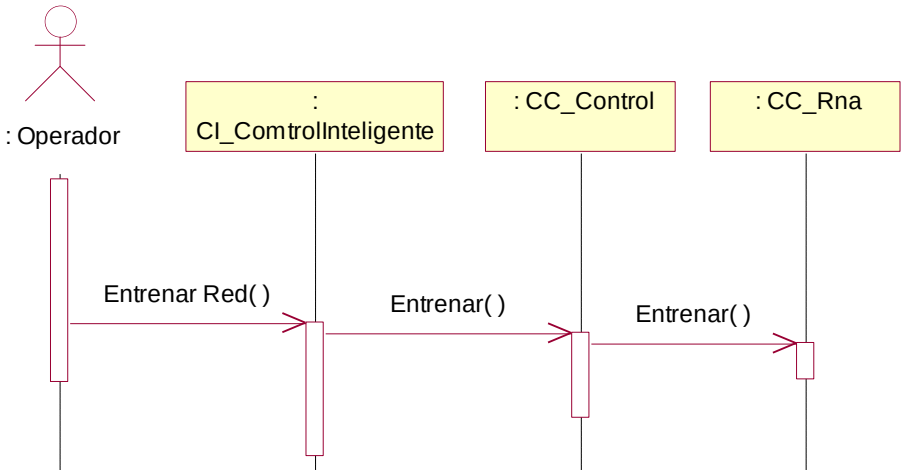


Fig.36 Diagrama de secuencia del Caso de Uso "EntrenarRed"

DIAGRAMA DE SECUENCIA

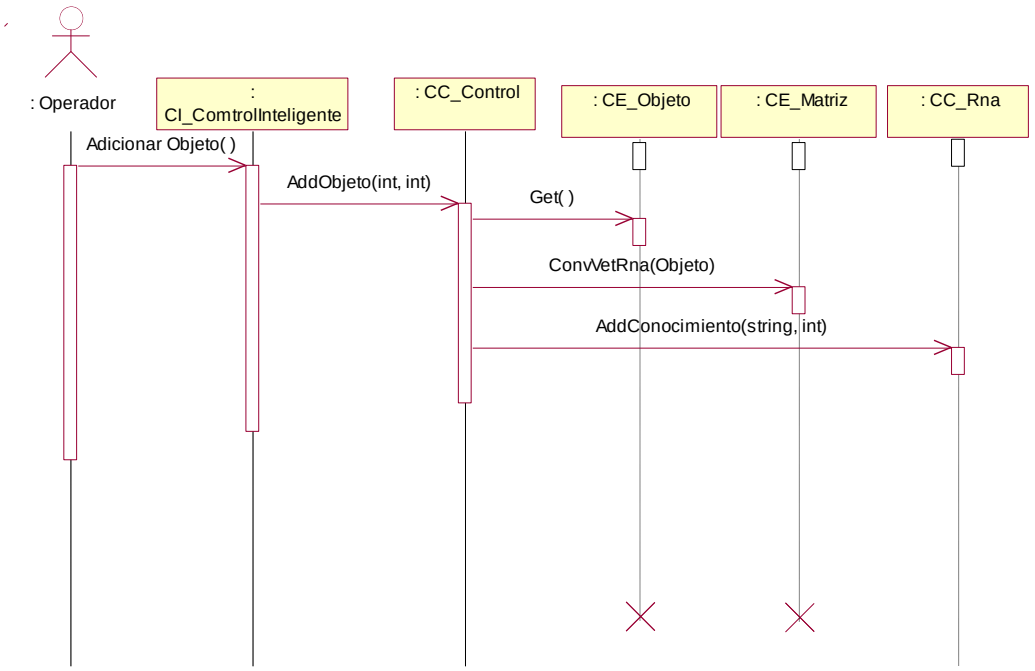


Fig. 37 Diagrama de Secuencia del Caso de Uso "LlenarBaseConocimiento"

3.3.3 Descripción de la Arquitectura[X]

Se utilizó arquitectura orientada a objetos, en la cual los componentes del sistema encapsulan los datos y las operaciones que se deben realizar para manipular los mismos. (10)

3.4 Patrones GRASP[XI] *utilizados.*

Patrón es una pareja problema / solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrón.

3.4.1 Bajo Acoplamiento.

Solución: Asignar una responsabilidad para mantener bajo acoplamiento.

Problema: ¿Cómo dar soporte a una dependencia escasa y a un aumento de reutilización?

El acoplamiento es una medida de la fuerza con que una clase esta conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. (9)

Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas ¿cuánto software podemos extraer de un modo independiente y reutilización en otro proyecto? Para determinar el nivel de acoplamiento de clases, son muy buenos los diagramas de colaboración de UML. Uno de los principales síntomas de un mal diseño y alto acoplamiento es una herencia muy profunda. Siempre hay que considerar las ventajas de la delegación respecto de la herencia. (10)

3.4.2 Experto.

Solución: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Problema: ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?

Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades. Durante el diseño orientado a objetos, cuando se definen las interacciones entre los objetos, tomamos decisiones sobre la asignación de responsabilidades a las clases. Si se hace en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se nos presenta la oportunidad de reutilizar los componentes en futuras aplicaciones. (10)

La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Hay que tener en cuenta que esto es aplicable mientras estemos considerando los mismos aspectos del sistema:

- Lógica de negocio
- Persistencia a la base de datos
- Interfaz de usuario

3.4.3 Controlador.

Solución: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

- El “sistema” global (controlador de fachada).
- La empresa u organización global (controlador de fachada).
- Algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).
- Un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados “Manejador <NombreCasodeUso>”(controlador de casos de uso)

Problema: ¿Quién debería encargarse de atender un evento del sistema?

Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta del sistema.

Un Controlador es un objeto de interfaz no destinada a usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. (10)

Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema

3.5 Diagramas de Clases del Diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contiene la siguiente información: clases,

asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, información sobre los tipos de los atributos, navegabilidad, dependencias. (14)

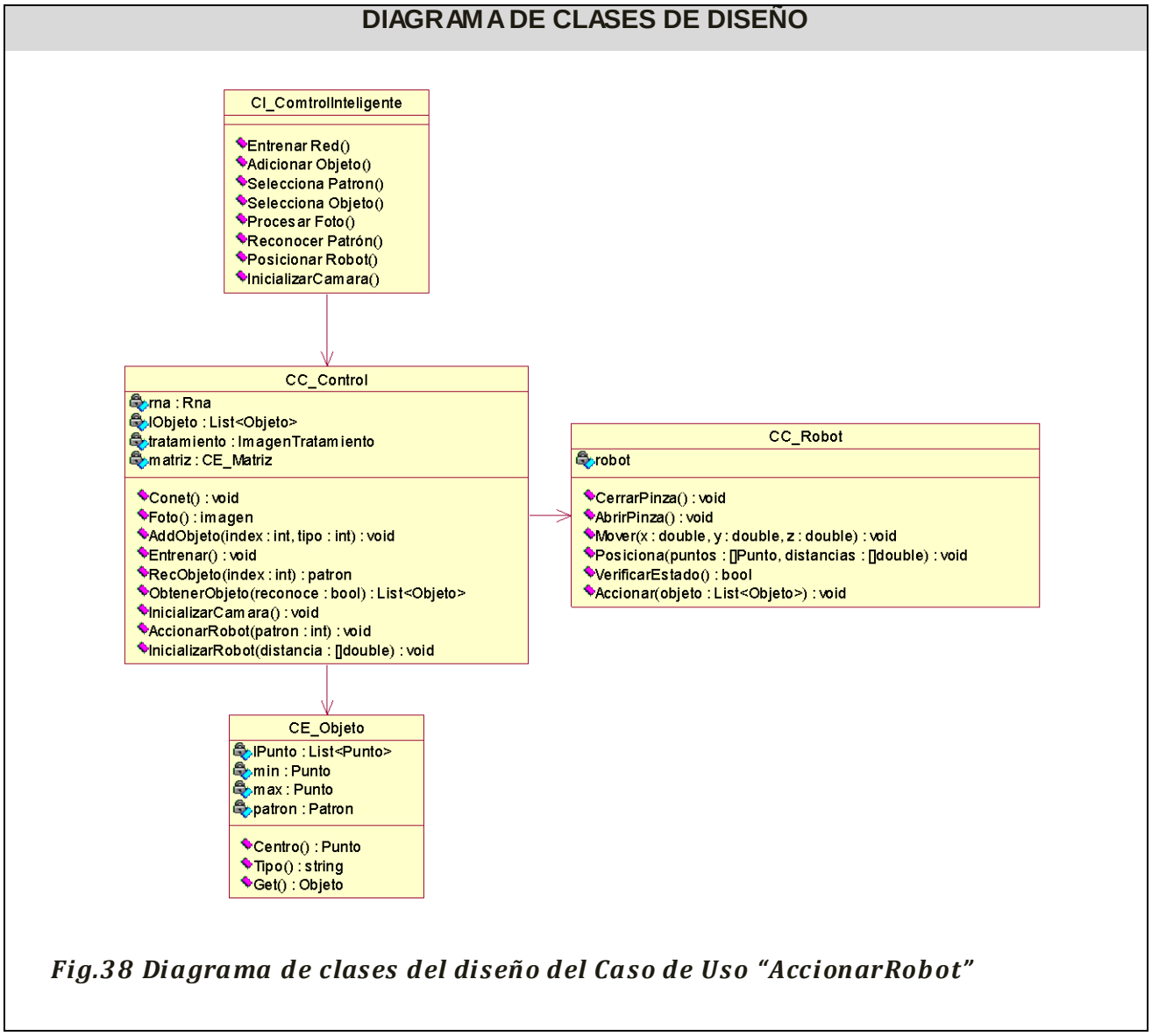


Fig.38 Diagrama de clases del diseño del Caso de Uso "AccionarRobot"

DIAGRAMA DE CLASES DE DISEÑO

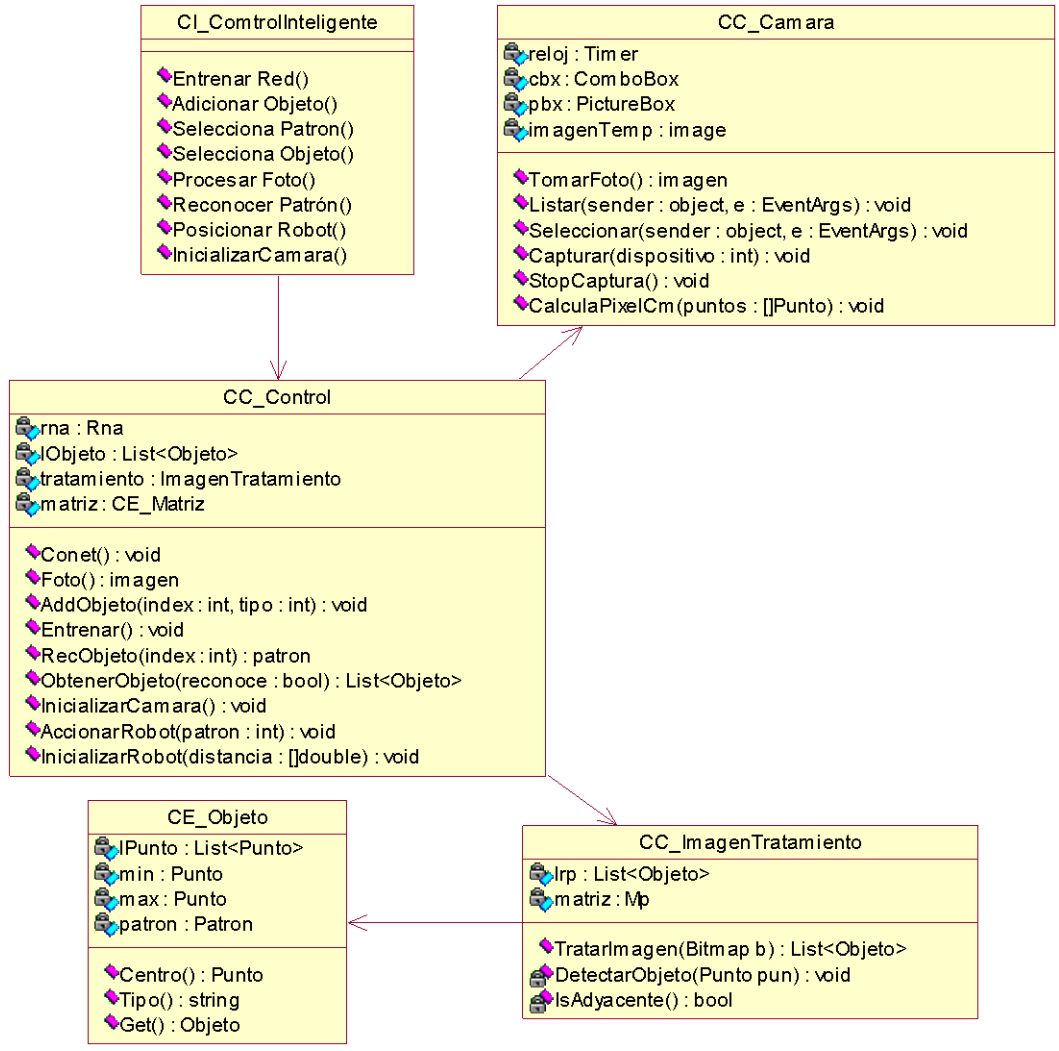


Fig.39 Diagrama de clases del diseño del Caso de Uso "TratarImagen"

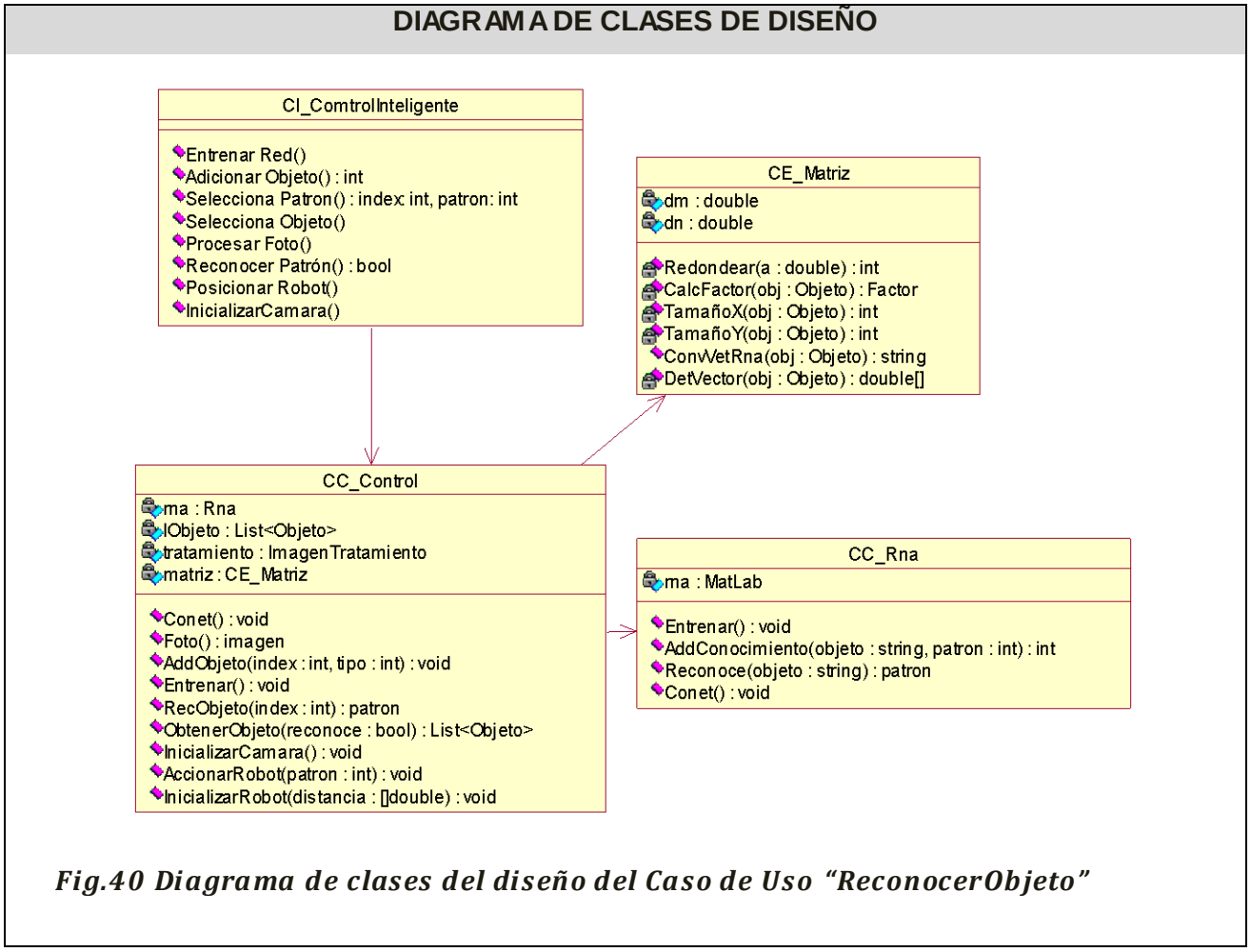


Fig.40 Diagrama de clases del diseño del Caso de Uso "ReconocerObjeto"

DIAGRAMA DE CLASES DE DISEÑO

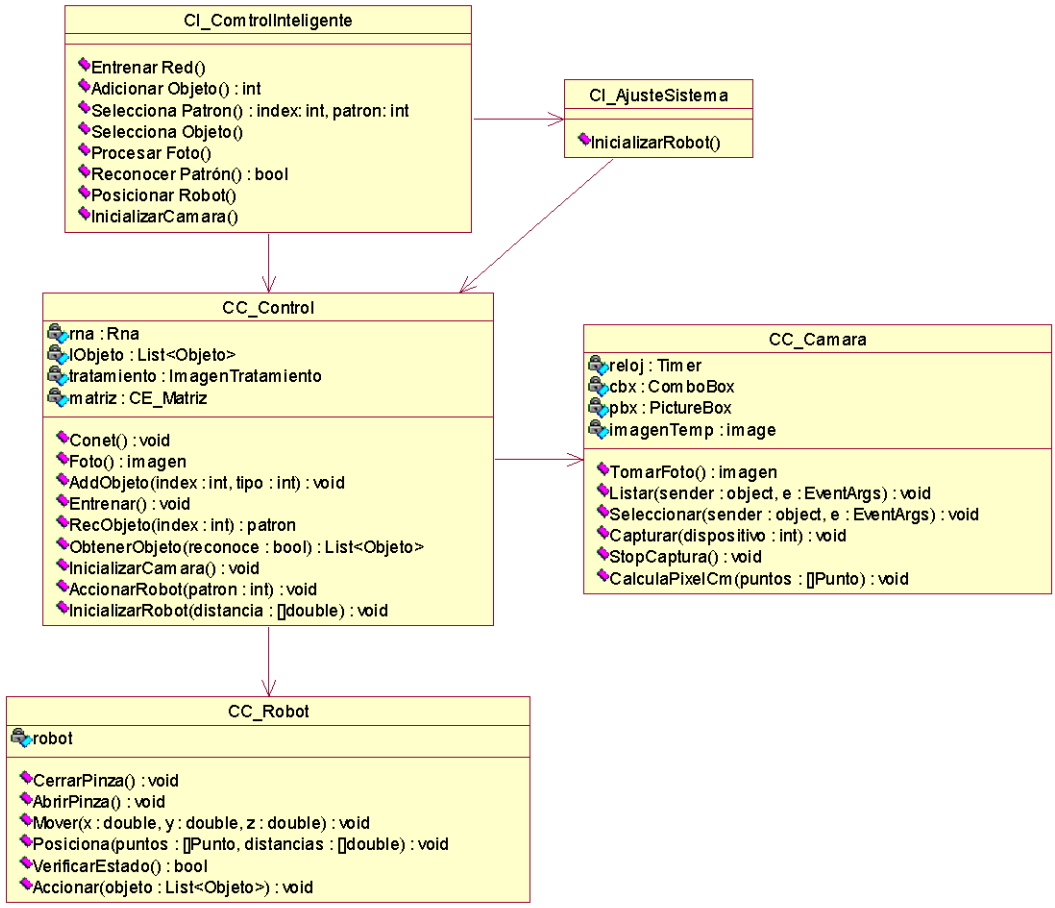


Fig.41 Diagrama de clases del diseño del Caso de Uso "InicializarSistema"

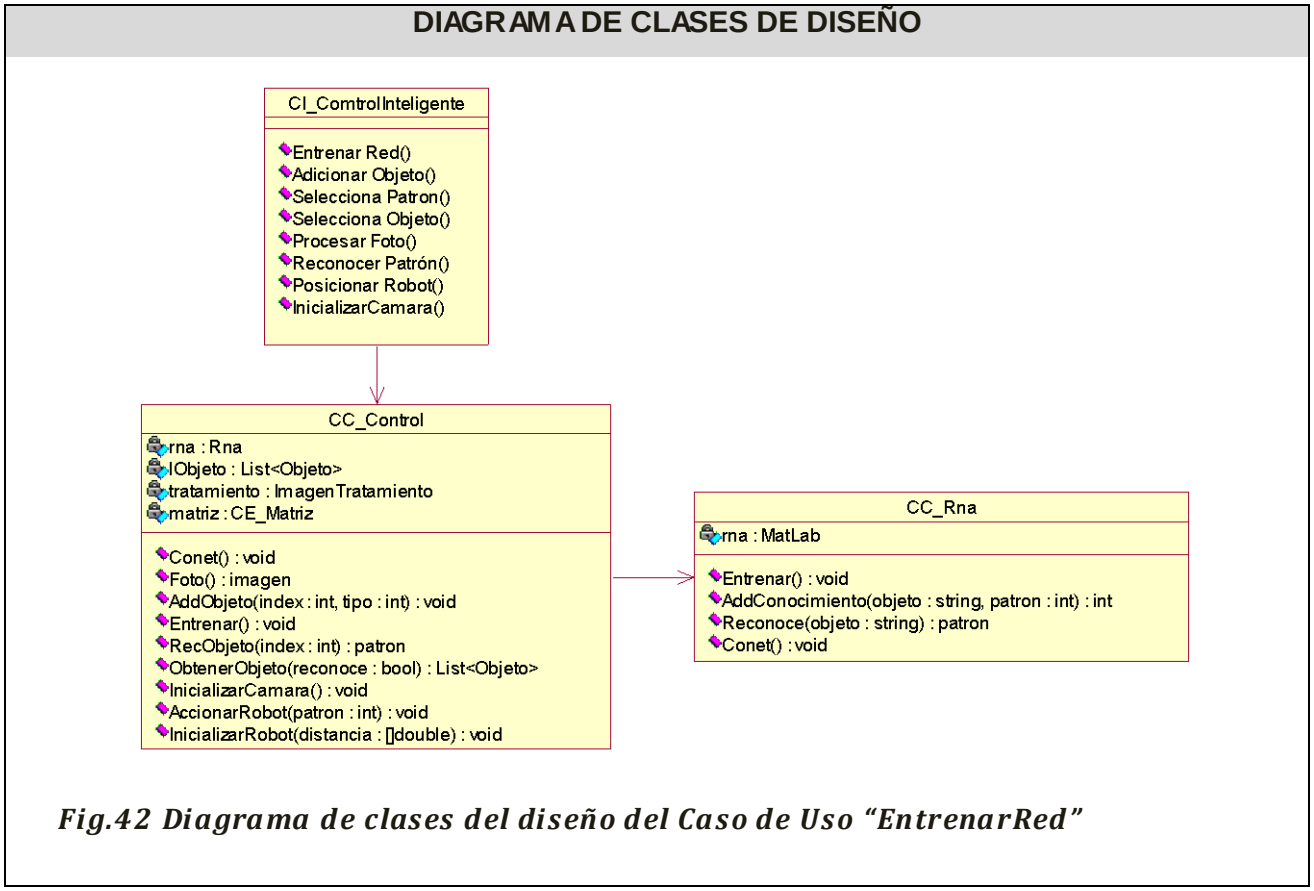


Fig.42 Diagrama de clases del diseño del Caso de Uso "EntrenarRed"

DIAGRAMA DE CLASES DE DISEÑO

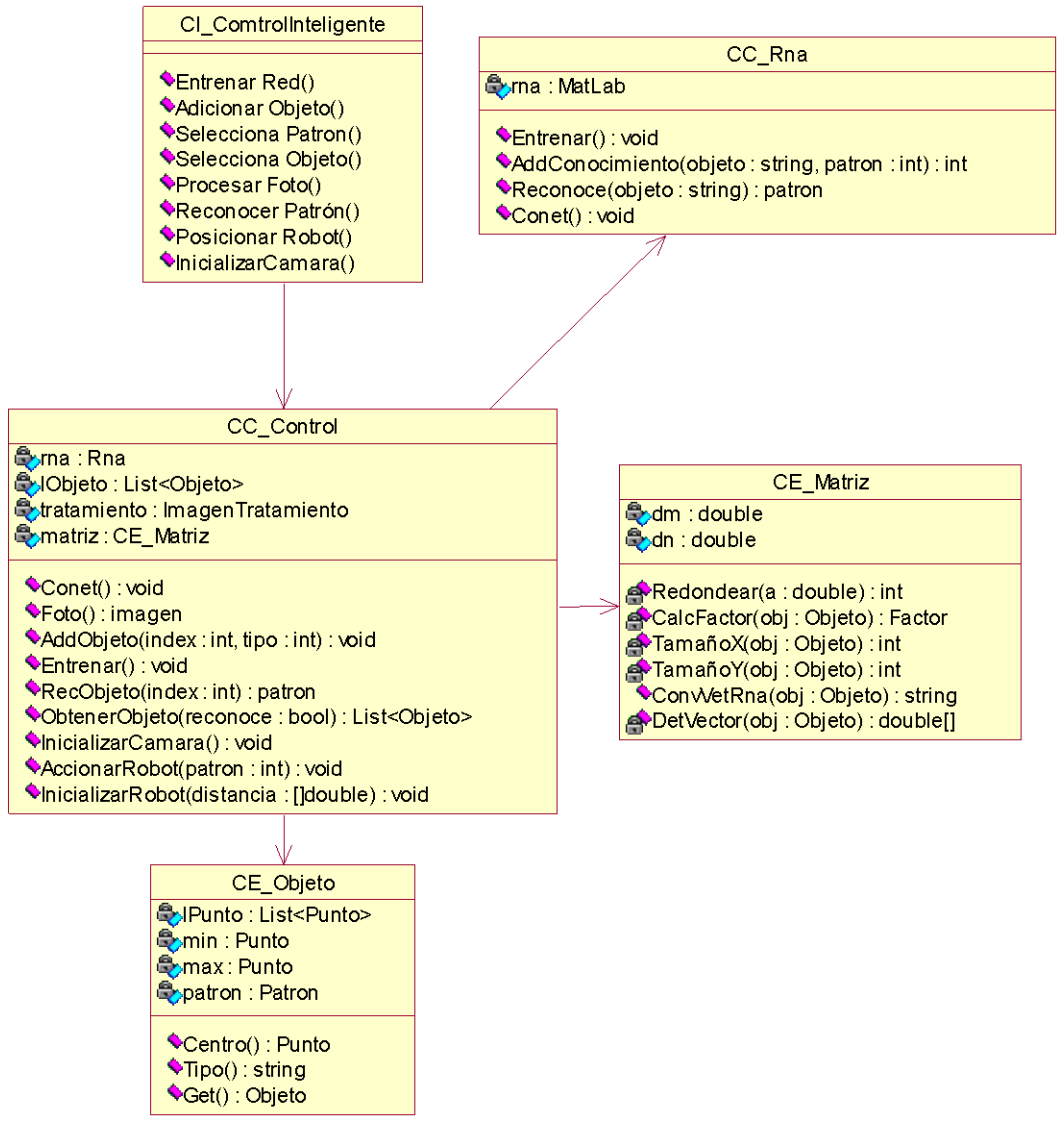


Fig.43 Diagrama de clases del diseño del Caso de Uso "LlenarBaseConocimiento"

3.6 Conclusiones

En este capítulo se presentan los diagramas de clases de análisis y diseño del sistema, así como los diagramas de interacción del mismo. Además se analizó e identificó que tipo de arquitectura se va a utilizar para implementar la aplicación y sus respectivos patrones.

CAPÍTULO 4: IMPLEMENTACIÓN

4.1 Introducción

El objetivo de este capítulo el elaborar los diagramas de implementación, en los que se muestran las dependencias entre las partes del código del sistema (diagrama de componentes) o la estructura del sistema en ejecución (diagrama de despliegue).

4.2 Diagrama de Despliegue

Los diagramas de despliegue, muestra los nodos procesadores, así como la distribución de los procesos y componentes. (10)

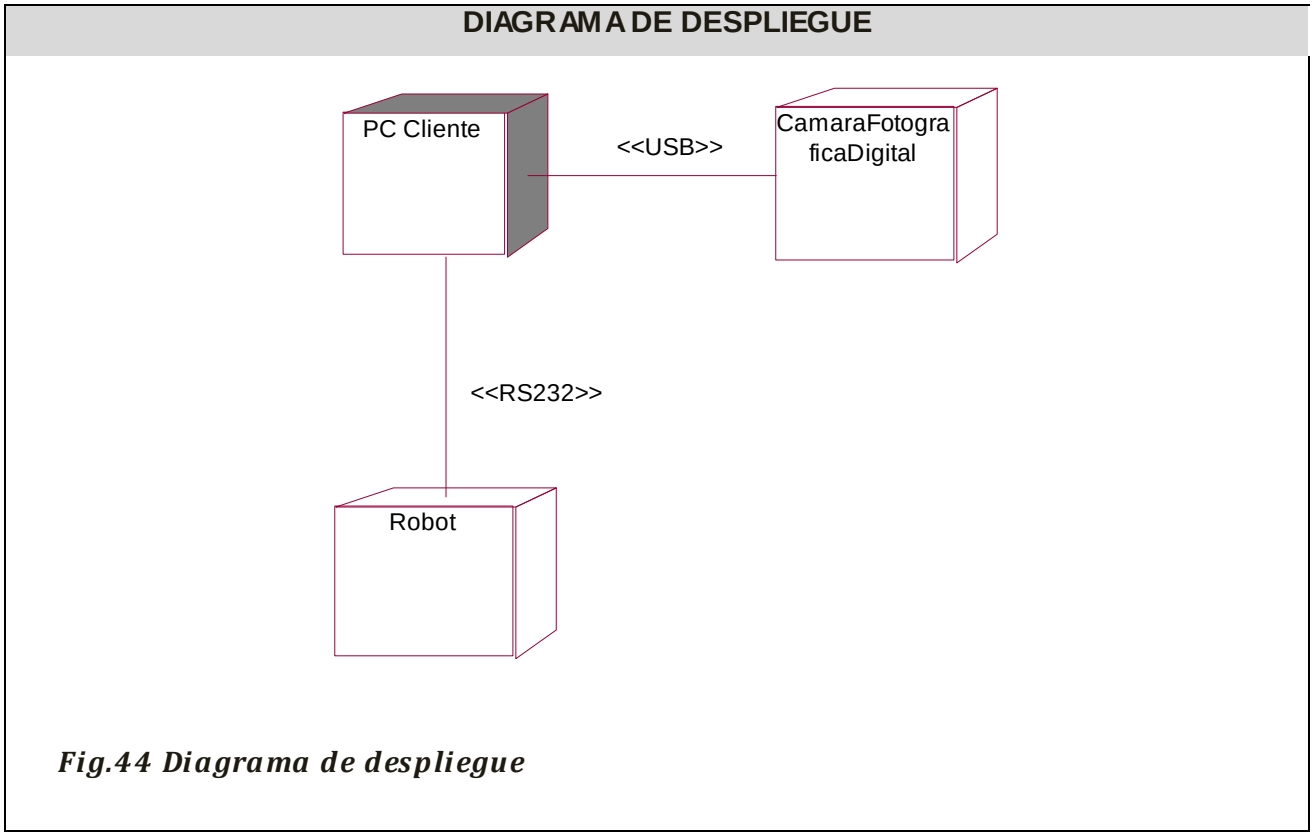


Fig.44 Diagrama de despliegue

4.3 Diagrama de Componentes

Los diagramas de componentes muestran las dependencias del compilador y del "runtime" entre los componentes del software; por ejemplo, los archivos del código fuente y las libreras de funciones. (14)

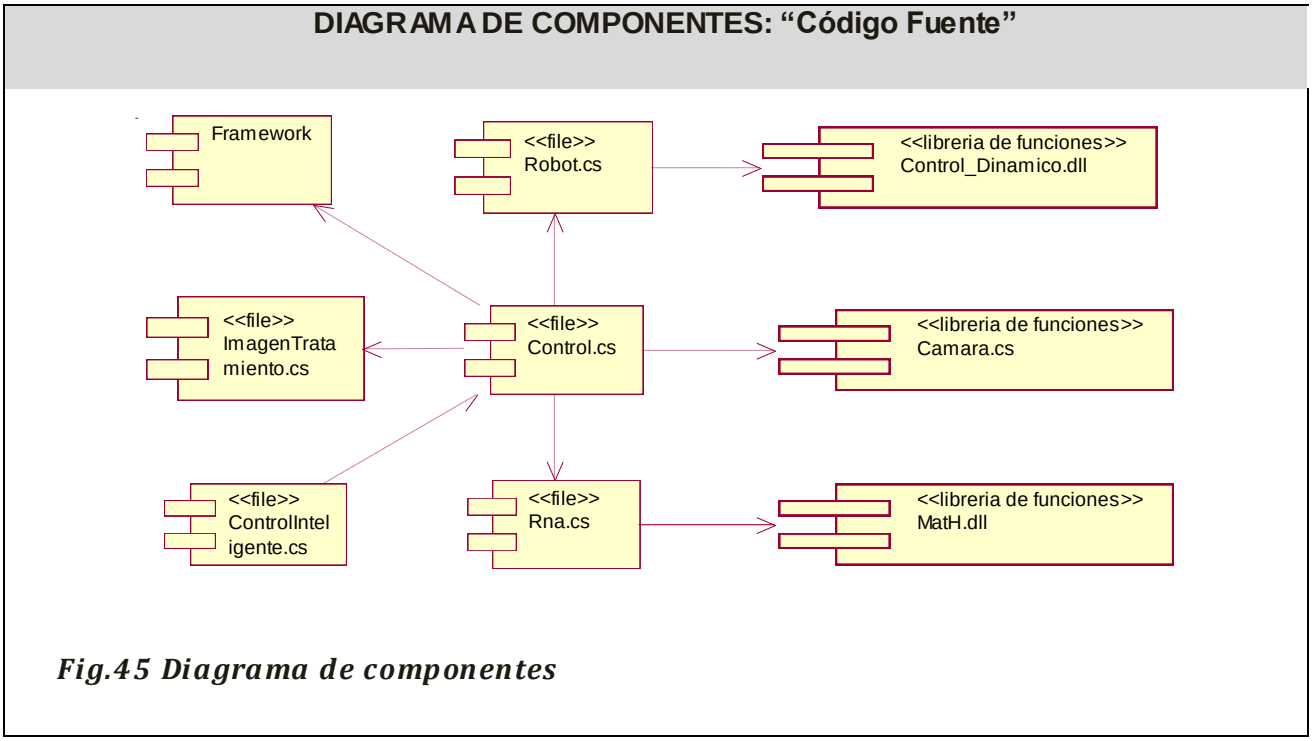


Fig.45 Diagrama de componentes

4.4 Conclusiones

En este capítulo quedaron elaborados los diagramas del flujo de trabajo de implementación (despliegue y componentes).

.

CAPÍTULO 5: ESTUDIO DE FACTIBILIDAD

5.1 Introducción

El objetivo de este capítulo es la realización de un estudio de la factibilidad del software para verificar dada sus entradas, salidas, peticiones, si el producto es o no factible a través del método de COCOMO II.

5.2 Planificación.

Características del proyecto.

1. Obtener los puntos de función. (UFP).

a) Se identifican las características del proyecto.

- EI: Entradas externas.
- EO: Salidas externas.
- ILF: Ficheros lógicos internos.
- EQ: Consultas (peticiones) externas.

b) Se clasifican los puntos de función por tipos.

Cada entidad analizada en el paso anterior la vamos a clasificar y asignarles un peso.

Nombre de la entrada externa (EI)	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación según tabla no 3.
Entrar los datos para accionar el robot	1	4	Bajo
Entrar los datos para entrenar la red	1	1	Bajo
Entrar los datos para adicionar objetos	1	1	Bajo
Entrar los datos para procesar la imagen	1	1	Bajo

Nombre de la salida externa (EO)	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación según tabla no 2.
Nombre de la petición (EQ)	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación según tabla no 2.
Nombre del fichero interno (ILF)	Cantidad de record	Cantidad de elementos de datos	Clasificación según tabla no 1.
Matriz	1	3	Bajo
Rana	1	2	Bajo
Objetos	1	4	Bajo
Robot	1	1	bajo
Cámara	1	6	Bajo

a. Aplicar los pesos. (3).

Aplicando los pesos según la complejidad para obtener los puntos de función desajustados utilizando la tabla 4:

Elementos	Bajo		Media		Alta		Subtotal
	No	Peso	No	Peso	No	Peso	
Entrada Externa	4	3					12
Salida Externa							
Petición							
Fichero lógico Interno	5	7					35
Subtotal (UFP)							47

2. Estimar la cantidad de instrucciones fuente. (SLOC).

En este trabajo de diploma se utilizará el lenguaje C#. Por tanto se trabajará con el ratio del lenguaje C++ que es 53.

SLOC = UFP * ratio.

Luego

SLOC =47 * 53

SLOC ≈ 2491 líneas de código fuente → 2.5 KSLOC

3. Aplicar las fórmulas de Bohem.

a. Obtener esfuerzo (PM) y tiempo de desarrollo (TDEV).

$$PM_{NS} = A \times Size^{E} \times \prod_{i=1}^{n} EM_i \qquad TDEV_{NS} = C \times (PM_{NS})^{F}$$

$$\text{where } E = B + 0.01 \times \sum_{i=1}^{5} SF_j \qquad \text{where } F = D + 0.2 \times 0.01 \times \sum_{j=1}^{5} SF_j$$

$$= D + 0.2 \times (E - B)$$

Donde:

Size: Tamaño estimado (KSLOC).

A = 2.94, B = 0.91, C = 3.67, D = 0.28

Tabla de Multiplicadores de esfuerzo

EM	VALOR	Para el Pos-Arquitectura
PERS	0.83	ACAP PCAP PCON
RCPX	1.00	RELY DATA CPLX DOCU
RUSE	0.95	RUSE
PREX	1.12	APEX PLEX LTEX
FCIL	1.33	TOOL SITE
SCED	1.00	SCED

PDIF	1,00	TIME STOR PVOL

Realizando los cálculos:

CÁLCULO DE	VALOR	JUSTIFICACIÓN
Esfuerzo (Meses-Hombre)	PM =9.25	A=2.94, Size=2.5, B =0.91, ΠEM = 1.17455 SFj PREC = 2.48 FLEX = 3.04 RESL = 2.83 TEAM = 2.19 PMAT = 6.24 ∑SFj 16.78 E= 1.0778
EM	1.17455	Ver tabla anterior.
TEDV	TEDV ≈ 7 meses	B = 0.91, C = 3.67, D = 0.28, E = 1,08 PM = 8.87 F =0, 314
Cantidad de personas	2 personas	PM =9.25 TEDV ≈ 7
Costo	2312.5 pesos	Salario medio de una persona 125. CHM =2*125 =250 C = CHM *PM

5.3 Beneficios tangibles e intangibles.

El sistema no ha sido concebido con fines comerciales, en su primera versión. Sino que por el momento está encaminado a la contribución y desarrollo de la investigación en los temas de la robótica y la visión artificial.

5.3.1 Beneficios Intangibles

- Realizar mediante una aplicación el control inteligente de un robot manipulador
- Garantizar un sistema para la visión artificial a partir de una cámara digital.

5.4 Análisis de costo. Beneficios

El sistema no está concebido en su primera versión con fines comerciales.

5.5 Conclusiones.

En este capítulo se realizó el estudio de la factibilidad del software en cuanto al tiempo de desarrollo, esfuerzo, el costo a desarrollarlo, el análisis de los costos y beneficios y el análisis de los beneficios tangibles e intangibles, concluyendo que ha sido factible la realización del sistema.

CONCLUSIONES GENERALES

Se obtuvo una aplicación capaz de dotar de cierto grado de inteligencia a un robot manipulador, a partir de la implementación de un sistema de visión artificial basado en el reconocimiento de patrones, utilizando redes neuronales. Se aplicaron algoritmos para el procesamiento de imágenes basados en el método de crecimiento de regiones por agrupación de pixeles, sobre imágenes filtradas a tonos de grises. Fueron aplicados además algoritmos para la identificación inteligente de objetos en las mismas. Se realizó el análisis y diseño del sistema propuesto, así como el estudio de factibilidad del mismo.

RECOMENDACIONES

Continuar el desarrollo de la investigación, en aras de mejorar el sistema de visión. Se recomienda mejorar específicamente la etapa de segmentación de la imagen, mediante la aplicación de un método basado en análisis discriminante (método de Otsu) que, además de ser menos ruidoso (la imagen se trabaja de forma local), posibilitaría tener un umbral dinámico (autoajustable) que permitiría al sistema tener un mayor grado de independencia en lo relativo a la iluminación del entorno de trabajo.

BIBLIOGRAFÍA

1. **Behar, Alberto Aguado.** *Temas de Identificación y Control Adaptable.* La Habana : Instituto de Cibernética, Matemática y Física, 2000. 3.

2. **Bello, Rafael.** Curso Introductorio de Redes Neuronales Artificiales. *Curso Introductorio de Redes Neuronales Artificiales.* Madrid : McGrawHill, 1993.

3. Redes Neuronales Artificiales. [En línea] 15 de Octubre de 2000. [Citado el: 11 de Diciembre de 2007.] 5. http://www.electronica.com.mx/neural/informacion/caracteristicas.html.

4. **Duda, Richard O. & Hart, Peter E.** *Pattern classification.* New York, : Wiley, 2001. 0471056693.

5. **Rodríguez Morales, Roberto.** *Monografía del curso de procesamiento de imágenes.* s.l. : Grupo de Procesamiento Digital de Señales. Instituto de Cibernética, Matemática y Física (ICIMAF), 2000.

6. **González Rodríguez, René et al.** Control servovisual cámara en mano. Análisis de estabilidad. Ciudad de la Habana : UCIENCIA, 2007.

7. *Control Servovisual para Robot Manipulador en 3D.* **González Orozco, Javier.** Santa Clara : Universidad Central "Marta Abreu" de Las Villas, 2007.

8. **Pressman, Roger S.** *Ingeniería de software. Un enfoque práctico.* 21.

9. **Larman, Craig.** *UML Y Patrones.* México : Pentice Hall, 1999. 19.

10. **Jacobson, Ivar et al.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Education, 2000. 22.

11. **Seco, José Antonio González.** Qué es C#. [En línea] [Citado el: 10 de Enero de 2008.] http://www.desarrolloweb.com/articulos/561.php. 13.

12. **Ferguson, Jeff, Paterson, Brian & Beres, Jason.** *La biblia de C#.* Madrid : Anaya Multimedia, 2003. 84-415-1484-4.

13. **Demuth, Howart_ Beale, Mark & Hagan, Martin.** *Neural Network Toolbox 5 User's Guide.* Natick : The MathWorks, Inc., 2007. 252.227-7014.

14. **García, Joaquín.** IngenieroSoftware. [En línea] 7 de Mayo de 2005. [Citado el: 15 de Mayo de 2008.] http://www.ingenierosoftware.com/analisisydiseno/uml.php.

15. **Ballesteros, Alfonso.** Redes Neuronales Netfirms. [En línea] 2008. [Citado el: 2 de Febrero de 2008.] http://www.redes-neuronales.netfirms.com/tutorial-redes-neuronales/clasificacion-de-redes-neuronales-por-topologia-arquitectura.htm. 6.

16. **Matthews, James.** Una Introduccion a las Redes Neuronales. [En línea] 9 de Enero de 2006. [Citado el: 2 de Febrero de 2008.] http://vidaartificial.com/index.php?title=Una_Introduccion_a_las_Redes_Neuronales_(Generation5.org). 7.

17. Redes Neuronales. Funcionamiento Básico. [En línea] [Citado el: 5 de Marzo de 2008.] http://perso.wanadoo.es/alimanya/funcion.htm. 8.

18. **Fritz, Walter.** Sistema inteligente. [En línea] 7 de Marzo de 2006. [Citado el: 8 de Enero de 2008.] http://www.intelligent-systems.com.ar/intsyst/intsystSp.htm. 10.

19. **R. Gallart, M. Laucirica y L. Crespo.** [En línea] 2000. [Citado el: 7 de Febrero de 2008.] http://revistas.mes.edu.cu:9900/EDUNIV/03-Revistas-Cientificas/Ingenieria-Electronica-Automatica-y-Comunicaciones/2000/1/10300116.pdf. 11.

20. **Gómez, Julián.** Microsoft.NET Framework 2.0. [En línea] 2008. [Citado el: 8 de Enero de 2008.] http://microsoft-net-framework.softonic.com/. 14.

21. Ediciones Visual Studio 2005 Professional. [En línea] 2008. [Citado el: 19 de Febrero de 2008.] http://www.microsoft.com/spanish/msdn/vs2005/editions/pro/default.mspx. 15.

22. **Felipe.** [En línea] 2008. http://www.monografias.com/trabajos5/matlab/matlab.shtml#intro. 16.

23. **C, Benjamín González.** La plataforma .NET. [En línea] [Citado el: 10 de Enero de 2008.] http://www.desarrolloweb.com/articulos/1704.php. 17.

24. REDES NEURONALES ARTIFICIALES. [En línea] [Citado el: 5 de Marzo de 2008.] http://www.gc.ssr.upm.es/inves/neural/ann2/concepts/app.htm. 9.

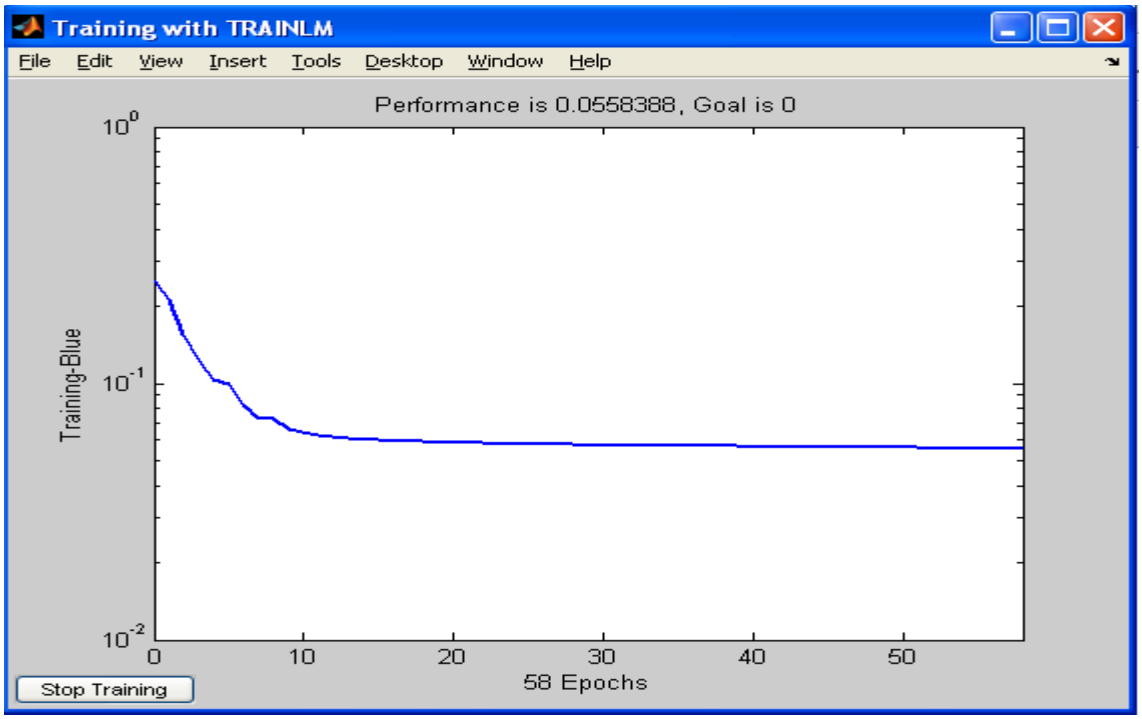
25. **Guide, ULWIS User's.** [En línea] [Citado el: 8 de Marzo de 2008.] http://www.itc.nl/external/unesco-rapca/Casos%20de%20estudios%20SIG/02%20Sensores%20remotos/sensores%20remotos.pdf. 12.

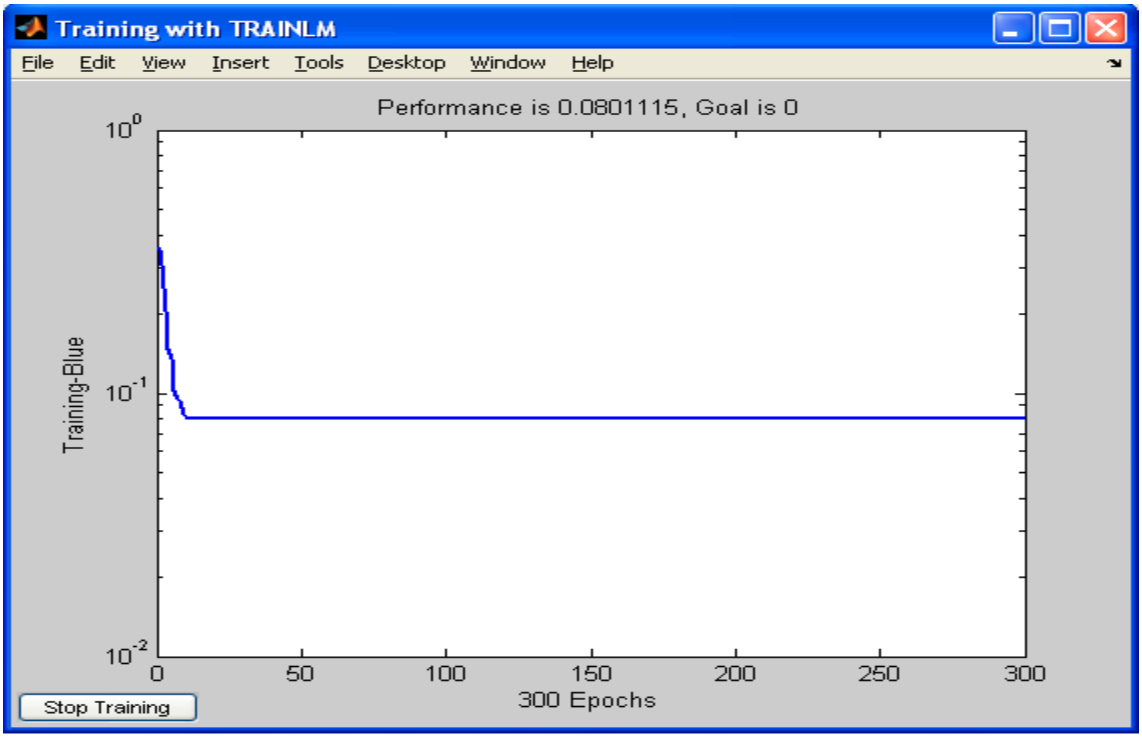
ANEXOS

Anexo 1. Función para el entrenamiento de la red neuronal.

```
function net = rnaTrain
clear()
[P, T] = rnaP;
[F, C] = size(P);
F = F -1;
if C ~= 0
    col = [0 210];
    mm = col;
    for i = 1 : F
      mm = [mm; col];
    end
E = C * 2;
 net = newff(mm,[20, 10, 3],{'tansig' 'logsig' 'logsig' 'hardlim'});

 net.trainParam.show = 50
 net.trainParam.lr = 0.05
 net.trainParam.epochs = E
 net.trainParam.goal = 1e-5

 [net, tr] = train(net, P, T)

try % 2 guardar en fichero
save 'rna\net.mat' net*
catch % 2
    ans = -10
end
end
```

Anexo 2. Gráficas de comportamiento de la red neuronal durante el entrenamiento con 58 y 300 épocas respectivamente.





GLOSARIO

[I] **Área de trabajo:** Espacio donde se encontrarán ubicados los objetos.

[II] **ICIMAF:** Instituto de Cibernética, Matemática y Física

[III] **RNA (Redes neuronales artificiales):** Están compuestas de un gran número de elementos de procesamiento altamente interconectados (Neuronas) trabajando al mismo tiempo para la solución de problemas específicos.

[IV] **Imagen:** Llámese imagen a una representación gráfica plasmada en una superficie y que es copia de otra original localizada a menudo distante de la imagen a una cierta distancia pero sin ser esta copia necesariamente de iguales dimensiones ni idéntica orientación que el original.

[V] **Requisito:** Condición o capacidad, necesidad o deseo que debe cumplir un sistema.

[VI] **Requisito Funcional:** Requisito que especifica una acción que debe ser capaz de realizar el sistema, sin considerar restricciones físicas. Requisito que especifica comportamiento de entrada/salida de un sistema.

[VII] **Requisito no Funcional:** Requisito que especifica propiedades del sistema, como restricciones del entorno o de implementación, rendimiento, dependencias de la plataforma, extensibilidad o fiabilidad. Requisito que especifica restricciones físicas sobre un requisito funcional.

[VIII] **Caso de uso:** es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

[IX] **Clase:** Elemento de software que describe o caracteriza una entidad del mundo real o un entidad del espacio de implementación.

- CC- Clase controladora
- CE- Clase entidad
- CI- Clase interfaz

[X] **Arquitectura:** Conjunto de elementos estructurales significativas acerca de la organización de un sistema software, la selección de los elementos estructurales que representan el sistema, y las interfaces entre ellos, junto con su comportamiento, tal y como se especifica en las colaboraciones entre esos elementos, la composición de estos elementos estructurales y

de comportamiento en subsistemas progresivamente mayores, y al estilo arquitectónico que guía esta organización: estos elementos y sus interfaces, sus colaboraciones y su composición.

[XI] Es un acrónimo que significa General Responsibility Assignment Software Patterns (patrones generales de software para asignar responsabilidades). Desde el punto de vista técnico, deberíamos escribir “Patrones GRAS” en vez de “Patrones GRASP” pero la segunda expresión suena mejor en inglés, idioma en que fue acuñado el término.