UNIVERSIDAD MAYOR DE SAN ANDRES

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMATICA

INTELIGENCIA ARTIFICIAL

INF-354

PROYECTO

Lógica difusa aplicada al preproceso y detección de bordes en imágenes digitales

Docente: PhD. Guillermo Choque Aspiazu

Nombre: Álvaro Mamani Quispe

C.I.: 7021324 L.P.

La Paz-Bolivia

2013

CONTENIDO

1. INTRODUCCION

La eficacia de la lógica difusa para manejar la ambigüedad e imprecisión que aparece en numerosos problemas ha motivado en los últimos años una creciente aplicación de dichas técnicas de inferencia al procesamiento de imágenes.

La lógica difusa representa un poderoso enfoque para la toma de decisiones. Dado que el concepto lógica difusa fue formulada en 1965 por Zadeh muchos investigadores se han llevado a cabo sobre su aplicación en las distintas áreas de procesamiento digital de imágenes tales como la calidad de imagen a evaluación, detección de bordes, segmentación de imágenes, etc. Actualmente vivimos en una época en que la tecnología cumple un papel importante en la vida del ser humano. Un campo de estudio muy activo llega a ser la detección de bordes para facilitar el análisis de imagen de más alto nivel, este campo llega a estar continuamente en constante desarrollo.

La detección de bordes es un proceso en el análisis digital de imágenes que detecta los cambios en la intensidad de luz. Estos cambios se pueden usar para determinar la profundidad, tamaño, orientación y propiedades de la superficie dentro de una muestra o pieza de trabajo.

La detección de bordes es una técnica de análisis digital que se puede aplicar a imágenes de muestras. Un borde puede ser llamado como la frontera entre dos regiones diferentes en una imagen.

2. PROBLEMA

En la actualidad se han desarrollado varios algoritmos para la detección de bordes en imágenes digitales, pero ninguno con un mayor porcentaje de exactitud.

3. OBJETIVOS

3.1 OBJETIVO GENERAL

- Utilizar la lógica difusa para la detección de bordes en imágenes digitales

3.2 OBJETIVOS ESPECIFICOS

- Realizar el diseño procedimental de la aplicación de detección de bordes en el laboratorio matricial MatLab.

- Aplicar técnicas de Ingeniería de Software.

4. MARCO CONCEPTUAL

4.1 LOGICA CLASICA

Establece que cualquier enunciado o proposición puede tener un valor lógico verdadero o falso, en definitiva 1 y 0.

De esta forma es posible desarrollar toda una lógica basada en leyes de este tipo.

4.2 LOGICAS MULTIDIMENSIONALES

Según Oostra (2004) y Ojeda (2007), las lógicas multidimensionales admiten y tratan datos imprecisos en lugar de excluirlos, es decir que se admiten y comprenden enunciados que son el producto de premisas imprecisas.

Los aspectos diarios que tienen que ver con la percepción y el comportamiento de los seres humanos como los gustos, el significado de los adjetivos, de los hechos, etc. Pueden ser estudiados con precisión considerando grados de pertenencia complejos, ya que aplicar a estos casos, modelos matemáticos y lógica bivalente puede concluir a aparentes paradojas.

Según Oostra (2004), durante el siglo XX se han propuesto diversas lógicas multidimensionales como ser: la lógica trivalente de Peirce, la lógica intuicionista de Brouswer, las lógicas m-valuadas de Post que tienen una contraparte algebraica en las llamadas algebras de Post; las lógicas multivaluadas o polivalentes de la escuela polaca de lógica y Jan Lukasiewicz, la lógica incontable-valuada que es la base de la lógica borrosa o difusa de Rescher 1969, Rasiowa 1962, Epstein 1993, Wasselkamper 1986 y Zadeh 1965 (Kwang 2005).

4.3 LOGICA DIFUSA

La lógica difusa es una lógica alternativa a la lógica clásica que pretende introducir un grado de vaguedad en las cosas que evalúa. En el mundo en que vivimos existe mucho conocimiento ambiguo e impreciso por naturaleza. El razonamiento humano con frecuencia actúa con este tipo de información. La lógica difusa fue diseñada precisamente para imitar el comportamiento del ser humano.

La lógica difusa se inició en 1965 por Lofti A. Zadeh, profesor de la Universidad de California en Berkeley. Surgió como una herramienta importante para el control de sistemas y procesos industriales complejos, así como también para la electrónica de entretenimiento y hogar, sistemas de diagnóstico y otros sistemas expertos.

La lógica difusa en comparación con la lógica convencional permite trabajar con información que no es exacta para poder definir evaluaciones convencionales, contrario con la lógica tradicional que permite trabajar con información definida y precisa.

4.3.1 Teoría de conjuntos difusos

Los conjuntos difusos surgieron como una nueva forma de representar la imprecisión y la incertidumbre, tienen como herramientas a la probabilidad, estadística, filosofía, psicología, matemáticas (Galindo, 2007).

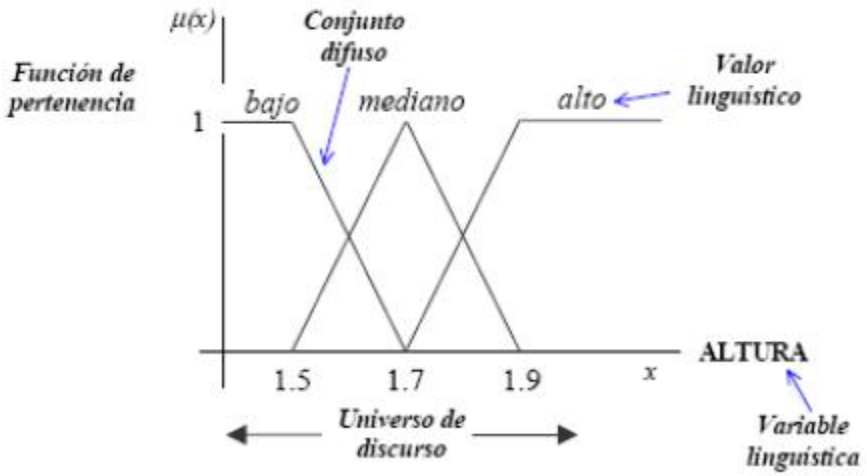
Un conjunto difuso es cualquier conjunto al cual los elementos pueden llegar a pertenecer como miembro, no miembro o miembro parcial en cierto grado, además no tiene una frontera clara para pertenecer o no a él, para definir esta pertenecía utiliza funciones definiendo así la transición de un conjunto a otro.

4.3.2 Conjuntos Difusos

La necesidad de trabajar con conjuntos difusos surge del hecho que existen conceptos que no tienen límites claros. Un conjunto difuso se encuentra asociado por un valor lingüístico que está definido por una palabra, etiqueta lingüística o adjetivo. En los conjuntos difusos la función de pertenencia puede tomar valores del intervalo entre 0 y 1, y la transición del valor entre cero y uno es gradual y no cambia de manera instantánea como pasa con los conjuntos clásicos. Un conjunto difuso en un universo en discurso puede definirse como lo muestra esta ecuación:

$$\bar{A} = \{x, \mu_A(x)\}, x \in X$$

Donde $\mu_A(x)$ es la función de pertenencia de la variable x, y X es el universo del discurso. Cuando más cerca este la pertenencia del conjunto A al valor de 1, mayor será la pertenencia de la variable x al conjunto A, esto se puede ver en la siguiente figura:



Ejemplo de conjuntos difusos

4.3.3 Funciones de Pertenencia

La función característica o también llamada de pertenencia, mide precisamente, el grado de pertenencia de un elemento a un conjunto difuso. La forma de la función de pertenencia que se utiliza depende del criterio para resolver el problema y puede depender de la cultura, geografía, época o punto de vista de usuario.

La función de pertenencia cumple de forma obligada la siguiente condición: "Tomar un valor entre cero y uno de manera continua".

Algunas funciones de pertenencia son más utilizadas de manera frecuente, debido a su simplicidad matemática y manejabilidad estas son: triangular, gaussiana, sigmoidal, gamma, pi, campana, etc. (Pedrycz 1995).

Para determinar la función de pertenencia asociada a un conjunto, se consideran conceptualmente dos aproximaciones (Soneura , 2005):

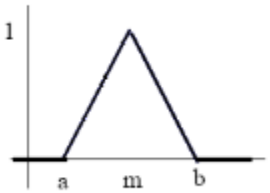
1. Aproximación basada en el conocimiento humano de los expertos

2. Aproximación basada en el empleo de una colección de datos para diseñar la función.

4.3.3.1 Funciones de Pertenencia Típicas

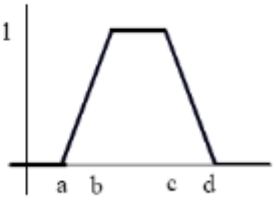
1. Triangular

$$\mu(x)=\begin{cases}0 & \text{para } x\le a\\ \dfrac{x-a}{m-a} & \text{para } a<x\le m\\ \dfrac{b-x}{b-m} & \text{para } m<x\le b\\ 0 & \text{para } x>b\end{cases}$$



2. Trapezoidal

$$\mu(x)=\begin{cases}0 & \text{para } x\le a\\ \dfrac{x-a}{b-a} & \text{para } a<x\le b\\ 1 & \text{para } b<x\le c\\ \dfrac{d-x}{b-c} & \text{para } c<x\le d\\ 0 & \text{para } x>d\end{cases}$$



4.3.4 Operaciones con conjuntos difusos

Las operaciones básicas entre conjuntos difusos son las siguientes:

1. *Unión.* Halla el máximo valor entre las funciones de pertenencia de 2 o más conjuntos difusos.

2. *Intersección*. Halla el mínimo valor entre las funciones de pertenencia de 2 o más conjuntos difusos.

3. *Complemento*. El complemento de un conjunto difuso, está definido por la diferencia que cada grado de pertenencia del de un valor lingüístico tiene con respecto al valor unitario.

Operador	Operación
Unión	$\max(\mu_A(x), \mu_b(y)) \mid x, y \in X$
Intersección	$\min(\mu_A(x), \mu_b(y)) \mid x, y \in X$
Complemento	$1 - (\mu_A(x)) \mid x \in X$

4.4 IMAGEN DIGITAL

Una imagen digital es una representación bidimensional de una imagen a partir de una matriz numérica, frecuentemente en binario. Dependiendo de si la resolución de la imagen es estática o dinámica puede tratarse de una imagen matricial (o mapa de bits) o de un gráfico vectorial.

Los elementos que definen una imagen digital son:

- Tamaño de la imagen (medida en pixeles)
- Resolución de entrada
- Resolución de salida
- Niveles de gris
- Tamaño del fichero
- Tipo de fichero

4.4.1 DETECCION DE BORDES

Los bordes de una imagen digital se pueden definir como transiciones entre dos regiones de noveles de gris significativamente distintos. Suministran una valiosa información sobre las fronteras de los objetos y puede ser utilizada para segmentar la imagen, reconocer objetos, etc.

4.4.2 HISTROGRAMA DE UNA IMAGEN DIGITAL

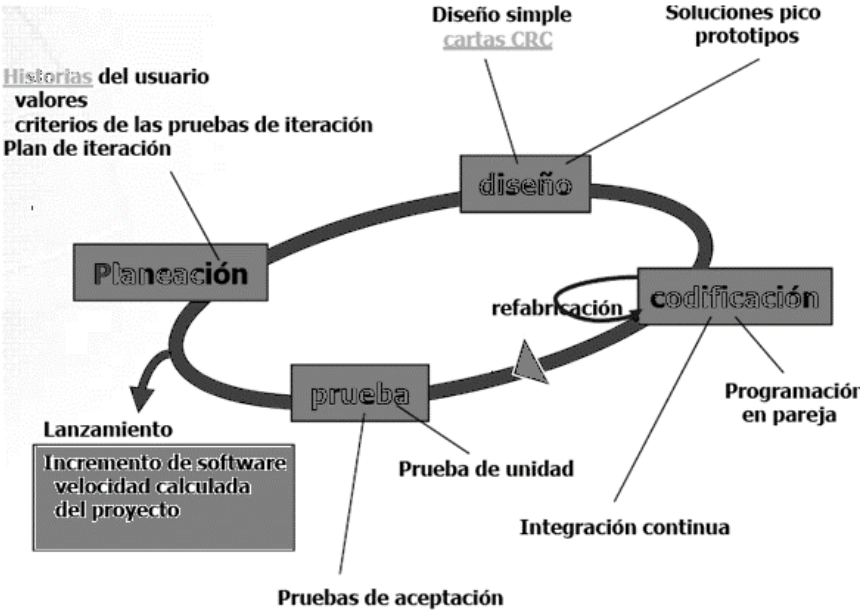
El histograma es una función que muestra, para cada nivel de gris, el número de pixeles de la imagen que tiene ese nivel de gris.

4.5 METODOLOGIA XP

La programación extrema o eXtreme Programming (XP) es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck, autor del primer libro sobre la materia, Extreme Programming Explained*:* Embrace Change (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.
Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

La XP, recibe este calificativo porque defiende un enfoque radical. Reconoce las bondades de las prácticas de las metodologías tradicionales y propone llevarlas hasta su extremo:
"Si diseñar es bueno, diseñemos todo el tiempo"
"Si las pruebas son buenas, probemos todo el tiempo"
La XP utiliza un enfoque OO, como su paradigma de desarrollo preferido. La XP abarca un conjunto de reglas y prácticas que ocurren en el contexto de 4 actividades del marco de trabajo:

- Planeación
- Diseño
- Codificación
- Prueba



4.5.1 Roles XP.- Los roles de acuerdo con la propuesta original de Beck son:

- Programador.- El programador escribe las pruebas unitarias y produce el código del sistema.
- Cliente.- Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.

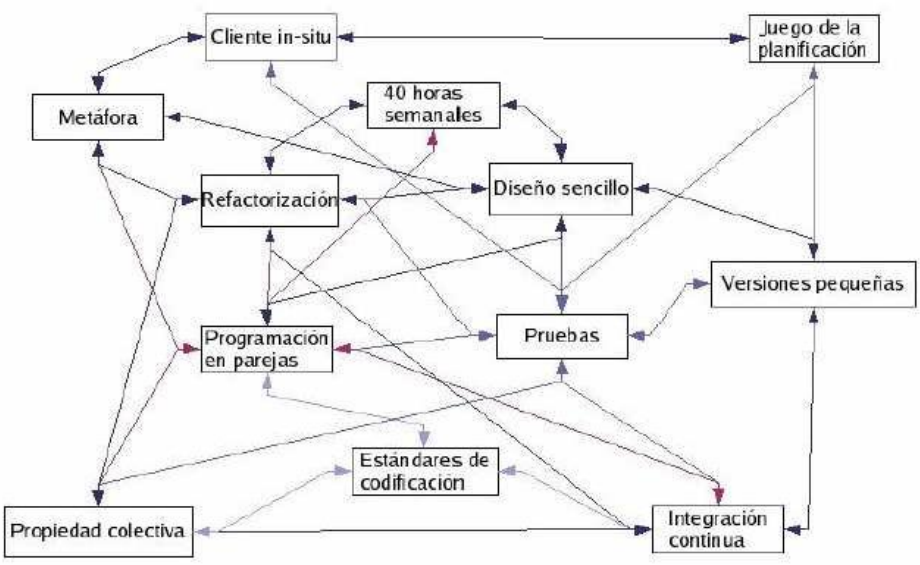
- Encargado de pruebas (Tester).- Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- Encargado de seguimiento (Tracker).- Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- Entrenador (Coach).- Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- Consultor.- Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- Gestor (Big boss).- Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

4.5.2 Prácticas XP.- La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas.

- El juego de la planificación.- Hay una comunicación frecuente el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.
- Entregas pequeñas.- Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más 3 meses.
- Metáfora.- El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).
- Diseño simple.- Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.
- Pruebas.- La producción de código está dirigida por las pruebas unitarias. Éstas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.
- Refactorización (Refactoring).- Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.
- Programación en parejas.- Toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores, etc.).
- Propiedad colectiva del código.- Cualquier programador puede cambiar cualquier parte del código en cualquier momento.
- Integración continúa.- Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.

- 40 horas por semana.- Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo.
- Cliente in-situ.- El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Éste es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita.
- Estándares de programación.- XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

Las prácticas se refuerzan entre sí:



El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras. Esto se ilustra en la Figura, donde una línea entre dos prácticas significa que las dos prácticas se refuerzan entre sí.
Si bien XP es la metodología ágil de más renombre en la actualidad, se diferencia de las demás metodologías que forman este grupo en un aspecto en particular: el alto nivel de disciplina de las personas que participan en el proyecto.

5. CUERPO SOLUCION

5.1 FASE DE PLANIFICACION

SISTEMA DE INFERENCIA DIFUSO

El nuevo sistema de detección de bordes basado en reglas difusas es desarrollado por el diseño de un Sistema de Inferencia Difusa (FIS System Inference Fuzzy) de Tipo Mamdani utilizando MATLAB como caja de herramientas. El algoritmo detecta una imagen de entrada mediante el

uso de una máscara de ventana de tamaño 2x2 que se desliza sobre toda la imagen de pixeles horizontalmente por pixel. El FIS se implementa considerando a cuatro pixeles P1, P2, P3 y P4 de la matriz mascara de 2x2 y una variable de salida.

P1 x(i-1,j-1)	P2 x(i-1,j)
P3 x(i,j-1)	P4 x(i, j)

Figura 1. Mascara 2x2

En la primera fase de la FIS, el proceso de fuzzificacion de entrada se lleva a cabo mediante la definición de dos funciones trapezoidales llamadas blanco y negro. Una vez que el pixel se fuzzifica, en la segunda fase de la FIS, una base de reglas evalúa para obtener la salida. Una función triangular de la salida se define como borde como se muestra en la figura 2.

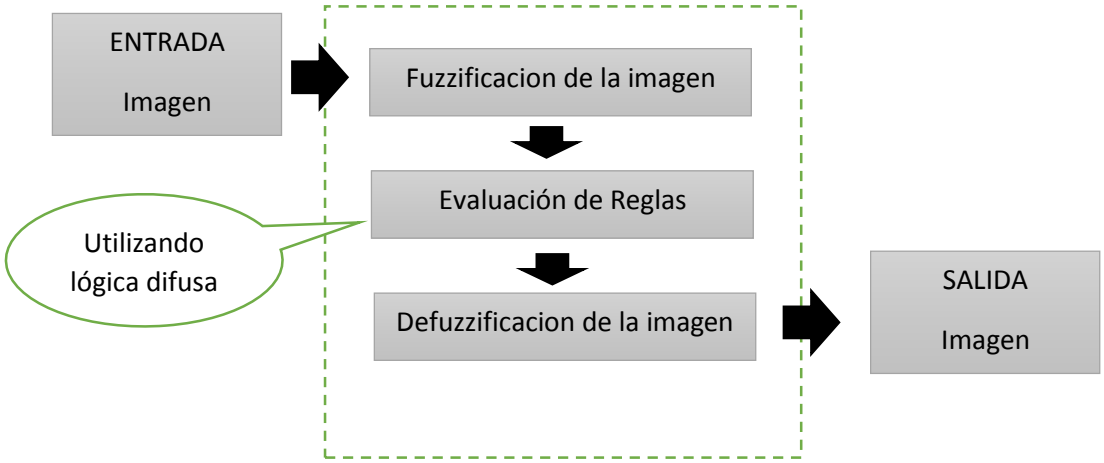


Figura 2 Sistema de Inferencia Difuso

5.2 FASE DE DISEÑO

En cuanto a la evaluación de reglas se han definido 10 reglas para aplicar a las incidencias de las entradas. La inferencia de las reglas dependen de los pesos de 3 vecinos es decir, P1, P2 y P3 y la propia P4, si los pesos son de grado Negro o de grado Blanco. Estos pesos se combinan utilizando el operador AND como se definía en la regla base.

La base de reglas utilizado en el FIS comprende las siguientes 10 reglas difusas para la valoración de los pesos de los tres vecinos P1, P2 y P3 con P4.

1. Si P1 es negro y P2 es negro y P3 es negro y P4 es blanco entonces P4 es borde.
2. Si P1 es negro y P2 es negro y P3 es blanco y P4 es blanco entonces P4 es borde.
3. Si P1 es negro y P2 es blanco y P3 es negro y P4 es blanco entonces P4 es borde.
4. Si P1 es blanco y P2 es negro y P3 es negro y P4 es blanco entonces P4 es borde.
5. Si P1 es blanco y P2 es blanco y P3 es blanco y P4 es negro entonces P4 es borde.
6. Si P1 es blanco y P2 es blanco y P3 es negro y P4 es negro entonces P4 es borde.
7. Si P1 es negro y P2 es negro y P3 es blanco y P4 es blanco entonces P4 es borde.
8. Si P1 es blanco y P2 es negro y P3 es blanco y P4 es negro entonces P4 es borde.
9. Si P1 es negro y P2 es negro y P3 es blanco y P4 es negro entonces P4 es borde.
10. Si P1 es negro y P2 es blanco y P3 es negro y P4 es negro entonces P4 es borde.

Entradas Difusas				**Salida Difusa**
P1	**P2**	**P3**	**P4**	**P4_salida**
N	N	N	B	BORDE
N	N	B	B	BORDE
N	B	N	B	BORDE
B	N	N	B	BORDE
B	B	B	N	BORDE
B	B	N	N	BORDE
N	N	B	B	BORDE
B	N	B	N	BORDE
N	N	B	B	BORDE
N	B	N	N	BORDE

Tabla 1. Matriz de Reglas difusas

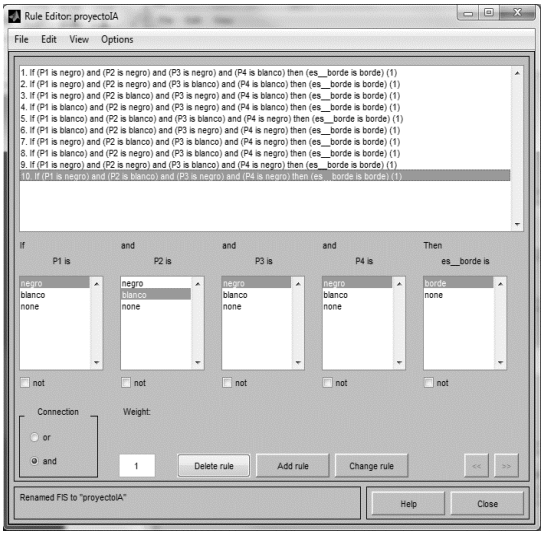
5.3 FASE DE CODIFICACION

En esta fase vamos a desarrollar un algoritmo que evalué las 10 reglas para las entradas. Utilizando el editor de MATLAB se creó la siguiente función denominada det_borde().

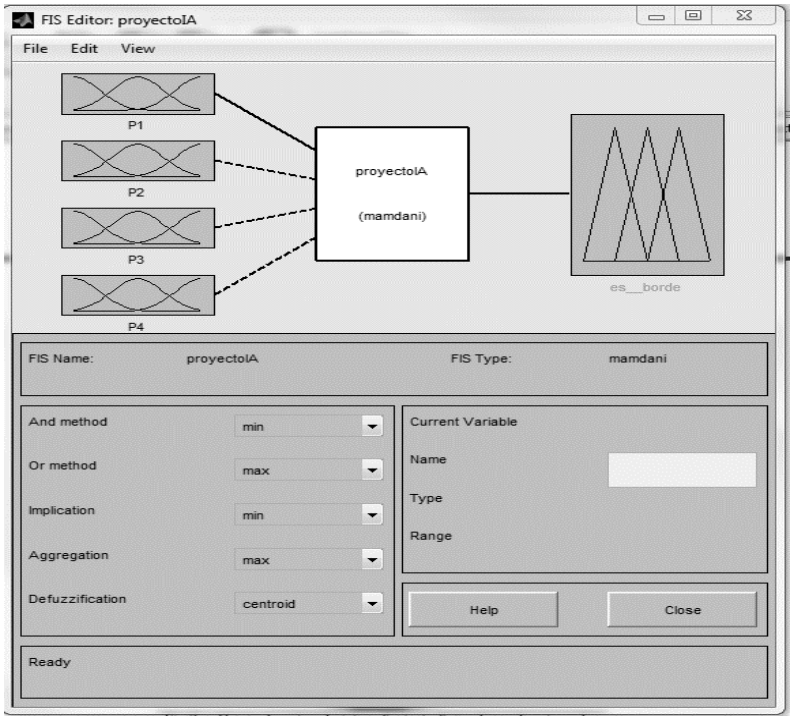
```
function [n]=det_bordes()
   A=imread('c:/3.jpg'); figure, imshow(A);
   g1 = rgb2gray(A);
   g=imresize(g1,[300,300]);
   [m,n]=size(g); figure, imshow(g);
   z=uint8(zeros(m,n));
   for i=2:m-1
      for j=2:n-1
        if g(i-1,j-1)>128 & g(i-1,j)>128 & g(i,j-1)>128 & g(i,j)<=128
            z(i,j)=255;
        else if g(i-1,j-1)>128 & g(i-1,j)>128 & g(i,j-1)<=128 & g(i,j)<=128
              z(i,j)=255;
          else if g(i-1,j-1)>128 & g(i-1,j)<=128 & g(i,j-1)>128 & g(i,j)<=128
                z(i,j)=255;
            else if g(i-1,j-1)<=128 & g(i-1,j)>128 & g(i,j-1)>128 & g(i,j)<=128
                  z(i,j)=255;
              else if g(i-1,j-1)<=128 & g(i-1,j)<=128 & g(i,j-1)<=128 & g(i,j)>128
                    z(i,j)=255;
                else if g(i-1,j-1)<=128 & g(i-1,j)<=128 & g(i,j-1)>128 & g(i,j)>128
                      z(i,j)=255;
                  else if g(i-1,j-1)>128 & g(i-1,j)>128 & g(i,j-1)<=128 & g(i,j)<=128
                        z(i,j)=255;
                    else if g(i-1,j-1)<=128 & g(i-1,j)>128 & g(i,j-1)<=128 & g(i,j)>128
                          z(i,j)=255;
                      else if g(i-1,j-1)>128 & g(i-1,j)>128 & g(i,j-1)<=128 & g(i,j)>128
                            z(i,j)=255;
                        else if g(i-1,j-1)>128 & g(i-1,j)<=128 & g(i,j-1)>128 & g(i,j)>128
                              z(i,j)=255;
                          end
                        end
                      end
                    end
                  end
              end
            end
          end
        end
      end
   end
figure, imshow(z);
end
```

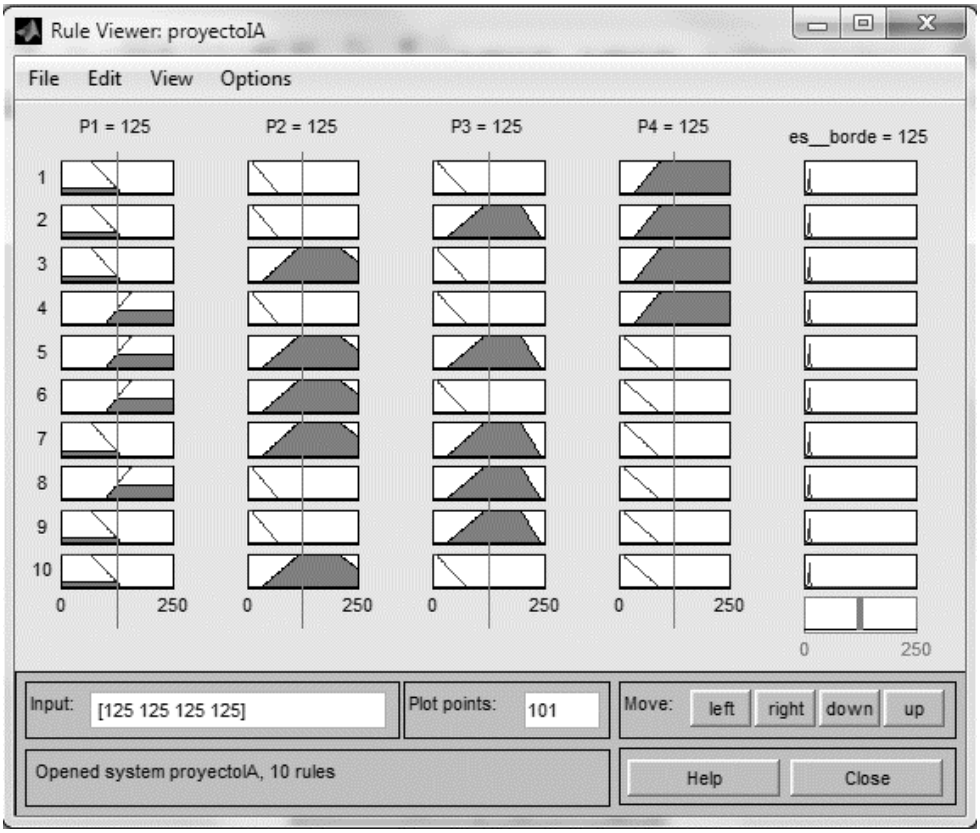
Utilizando el Editor de Sistema de inferencia Difuso FIS y el modelo Mamdani, creamos:

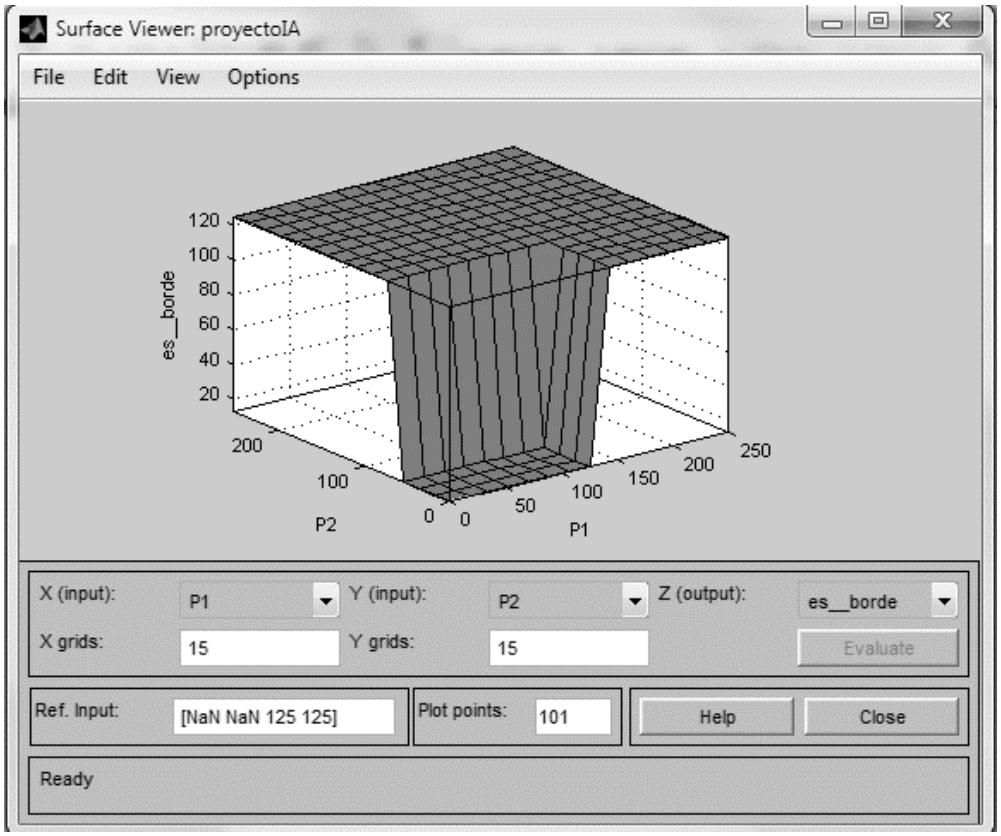
- 4 variables de entrada P1, P2, P3 y P4
- una variable de salida es_borde
- cada variable de entrada con función de pertenencia Blanco y Negro
- Ingresamos las 10 reglas del modelo que definimos anteriormente:



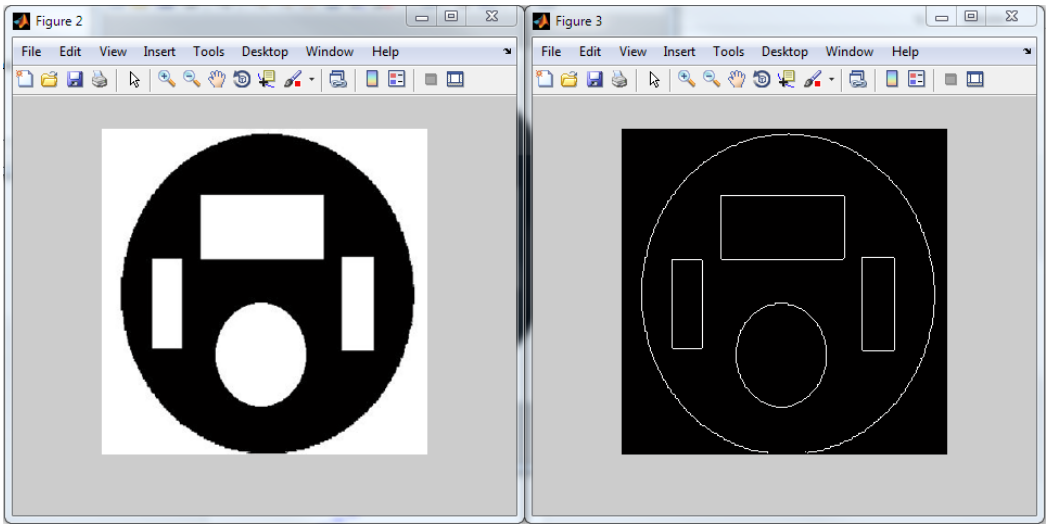
5.4 FASE DE PRUEBAS



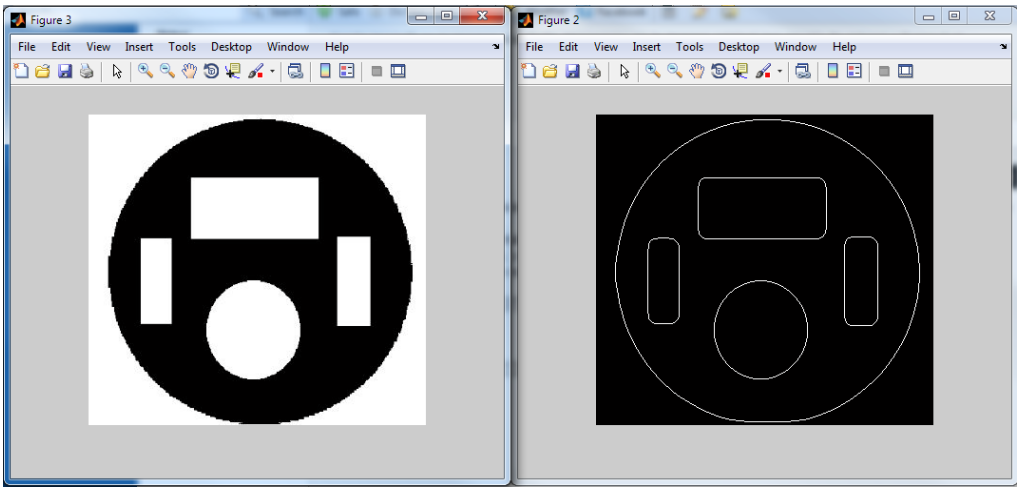

Rule Viewer: proyectoIA
File Edit View Options
P1 = 125
P2 = 125
P3 = 125
P4 = 125
es__borde = 125
1
2
3
4
5
6
7
8
9
10
0
250
Input: [125 125 125 125]
Plot points: 101
Move: left right down up
Opened system proyectoIA, 10 rules
Help
Close

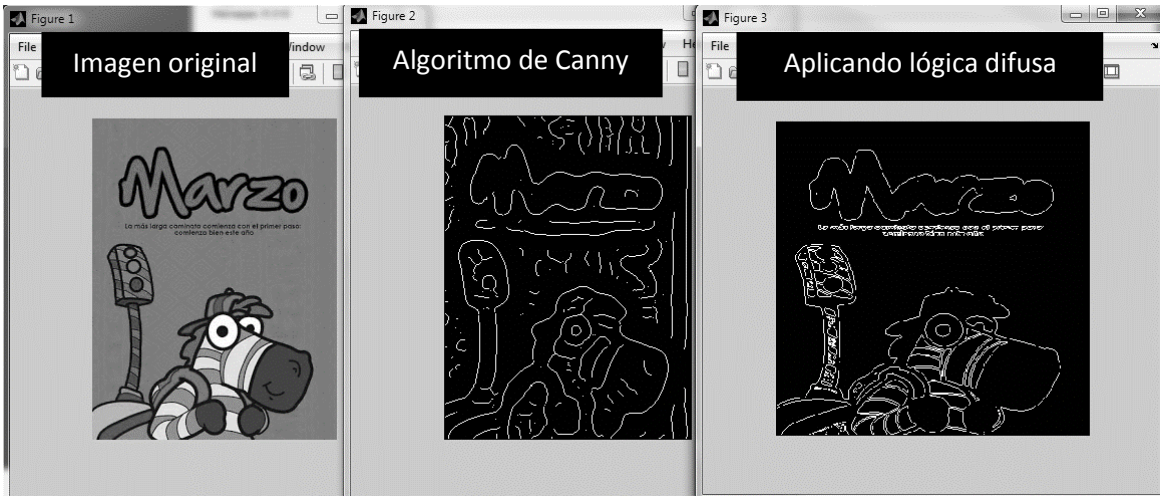

Surface Viewer: proyectoIA
File Edit View Options
es__borde
120
100
80
60
40
20
200
100
0
P2
0
50
100
150
200
250
P1
X (input): P1
Y (input): P2
Z (output): es__borde
X grids: 15
Y grids: 15
Evaluate
Ref. Input: [NaN NaN 125 125]
Plot points: 101
Help
Close
Ready

Una muestra de la imagen original y otra aplicando la lógica difusa para la detección de bordes:



Utilizando el algoritmo de Canny:





El nuevo enfoque desarrollado para la detección de borde es capaz de detectar bordes muy afilados para distinto tipo de imágenes.

6. CONCLUSIONES

En este trabajo, se ha puesto énfasis en el desarrollo de una manera sencilla y pequeña del algoritmo para la detección de bordes utilizando conceptos de inteligencia artificial y procesamiento de imágenes digitales. Las comparaciones se realizaron con distintos algoritmos de detección de bordes que han sido desarrollados. Se ha visto los resultados que demuestran la exactitud de la detección de bordes utilizando el algoritmo basado en reglas difusas en comparación con los otros algoritmos. El algoritmo basado en reglas difusas ha tenido éxito después de su implementación y ejecución con distinto tipo de imágenes.

BIBLIOGRAFIA

http://cdn.intechopen.com/pdfs/36637/InTech-Edge_detection_based_on_fuzzy_logic_and_expert_system.pdf

http://ijircce.com/upload/2013/march/1_Detection%20of%20Edges.pdf

http://www.ijarcsse.com/docs/papers/Volume_3/6_June2013/V3I6-0493.pdf

http://www.ijcaonline.org/journal/number22/pxc387661.pdf

http://www.ijarcsse.com/docs/papers/June2012/Volume_2_issue_6/V2I600257.pdf

http://cdn.intechopen.com/pdfs/36637/InTech-Edge_detection_based_on_fuzzy_logic_and_expert_system.pdf

http://catarina.udlap.mx/u_dl_a/tales/documentos/lmt/ramirez_r_o/capitulo3.pdf

https://www.u-cursos.cl/ingenieria/2007/2/EL42D/1/material_docente/bajar?id_material=143137