INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PROGRAMA DE POSGRADO EN INGENIERÍA DE SISTEMAS

"ALGORITMOS PARA TRAZADO AUTOMÁTICO UTILIZANDO UN ROBOT CARTESIANO"

T E S I S

que para obtener el grado de
DOCTOR EN INGENIERÍA DE SISTEMAS

PRESENTA:
OTONIEL IGNO ROSARIO

DIRECTORES DE TESIS:
DRA. CLAUDIA HERNÁNDEZ AGUILAR
DR. ALFREDO CRUZ OREA

Cd. de México, 2019

SIP-14-BIS

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 12:00 horas del día 11 del mes de Octubre del 2019 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de E.S.I.M.E.-ZAC. para examinar la tesis titulada:

"ALGORITMOS PARA TRAZADO AUTOMÁTICO UTILIZANDO UN ROBOT CARTESIANO"

Presentada por el alumno:

IGNO	**ROSARIO**	**OTONIEL**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: B 1 4 1 1 5 1

aspirante de: **DOCTOR EN INGENIERÍA DE SISTEMAS**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

DRA. CLAUDIA HERNÁNDEZ AGUILAR — **Presidente**

DR. ALFREDO CRUZ OREA — **Segundo Vocal**

DR. LUIS MANUEL HERNÁNDEZ SIMÓN — **Tercer Vocal**

DR. ALFREDO CRUZ OREA — **Secretario**

DR. FLAVIO ARTURO DOMÍNGUEZ PACHECO

DR. JORGE ARMANDO ROJAS RAMÍREZ

PRESIDENTE DEL COLEGIO DE PROFESORES

DR. MIGUEL TOLEDO VELÁZQUEZ

I. P. N.
SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA DE CESIÓN DE DERECHOS

En la Ciudad de México el día 14 de noviembre de 2019, el que suscribe Otoniel Igno Rosario alumno del Programa de Doctorado en Ingeniería de Sistemas con número de registro B141151, adscrito a la Sección de Estudios de Posgrado e Investigación de la ESIME Unidad Zacatenco, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección de la Dra. Claudia Hernández Aguilar y del Dr. Alfredo Cruz Orea y cede los derechos del trabajo intitulado: Algoritmos para trazado automático utilizando un robot cartesiano, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o directores del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección de correo otonieligno@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Otoniel Igno Rosario

Nombre y firma

Agradecimientos

A Dios por el regalo de la vida

Al IPN por permitirme ser parte de su comunidad

A mis directores de tesis por sus valiosas observaciones

A mi familia por su gran apoyo constante

Resumen

Uno de los problemas que enfrenta el sector artesanal de nuestro país en el contexto de la globalización es la falta de desarrollo tecnológico. Con este trabajo se busca sumarse a otros grupos de investigación que trabajan para generar tecnología propia para incidir en este problema de generación de tecnología que repercute en el desarrollo y la calidad de vida de una nación. De esta manera se plantea el diseño de un sistema automatizado basado en un robot cartesiano y en algoritmos computacionales para aplicar pintura estilizada. Por otro lado, también se busca apoyar el desarrollo de nuevas formas de producción artística mediante robot.

Para abordar el problema se planteó el objetivo general: contribuir al desarrollo tecnológico de la producción artesanal mediante el uso de un robot cartesiano y de un modelo computacional que simule el proceso de pintado. Este objetivo se dividió en los objetivos particulares: (1) Desarrollar la interfaz gráfica y el robot cartesiano, (2) Suavizar imagen opcionalmente, (3) Segmentar regiones automáticamente, (4) Realizar segmentación interactiva, (5) Generar campo vectorial automático, (6) Crear campo vectorial interactivo, (7) Producir trazos a partir del campo vectorial, (8) Obtener bordes, (9) Aplicar pintura robótica.

La metodología para resolver el objetivo particular (1) fue: usar una tarjeta electrónica comercial compatible con el hardware de Arduino, fabricar el sistema mecánico basado en motores a paso y por último diseñar el software de la tarjeta Arduino para controlar los motores y para comunicarse con una interfaz gráfica de usuario. Para abordar los objetivos particulares (2-9) se diseñó una aplicación en Matlab con una interfaz gráfica de usuario la cual se basa en las siguientes operaciones: se elige una imagen de entrada a la cual se aplica opcionalmente un suavizado de difusión anisotrópica, luego se trabaja por regiones mediante segmentación automática o interactiva y en dichas regiones se crea un campo vectorial basado en la interpolación local de trayectorias trazadas manualmente o basado en el gradiente, posteriormente se crean trayectorias con curvas Bézier mediante el campo y por último se envía la información de cada pincelada al robot mediante cadenas de 8 bits donde se codifica la longitud, el color y las coordenadas, enviándose únicamente los puntos de control de cada curva Bézier y luego ésta se reconstruye en la tarjeta Arduino, la cual activa los motores trazando cada curva. Finalmente se aplican capas de pintado sucesivas las cuales constan de una serie de pinceladas guiadas por el campo vectorial. Cada pincelada se aproxima mediante curvas Bézier para lograr pinceladas lo más parecido al trazo humano. Los materiales utilizados fueron, pinceles, papel y pintura acrílica.

En cuanto a los resultados obtenidos, para el objetivo particular (1) se logró una resolución de paso de 6.25 micras, mientras que la interfaz electrónica Arduino del robot se comunicó a 9600 bits por segundo a la interfaz gráfica de usuario diseñada en Matlab donde el código diseñado para la tarjeta Arduino constó de 430 líneas y el código en Matlab de 1500 líneas. Para los objetivos particulares (2-8) se logró desarrollar una interfaz gráfica de usuario donde se abre una imagen y se aplican los pasos mencionados anteriormente mediante botones y barras de deslizamiento interactivos lográndose una comunicación en tiempo real con la interfaz electrónica del robot. En cuanto al objetivo particular (9) se logró pintar un cuadro al acrílico con resultados estéticos agradables.

PALABRAS CLAVE: Pintura robótica, procesamiento digital de imagen, aproximación Bézier, interpolación de datos dispersos, difusión anisotrópica, segmentación k-medias, detección de esquinas.

Abstract

One of the problems affecting the artisanal sector in our country in the context of globalization is the lack of technological development. This work seeks to be part of other research groups that work to generate their own technology to address the problem of technology generation that affects the development and quality of life of a nation. For this reason, we propose the design of an automated system based on a Cartesian robot and computational algorithms to produce stylized paint. On the other hand, we also seek to support the development of new forms of artistic production using robots.

In order to address the aforementioned problem, we proposed the general objective: contribute to the technological development of handcrafted production by using a Cartesian robot and a computational model that simulates the painting process. This objective was divided into the following objectives: (1) Develop the graphic interface and the Cartesian robot, (2) Smooth image if necessary, (3) Segment regions automatically, (4) Perform interactive segmentation, (5) Generate automatic vector field, (6) Create interactive vector field, (7) Produce strokes from vector field, (8) Obtain edges, and (9) Apply robotic paint.

The methodology to solve the particular objective (1) was: use a commercial electronic card compatible with the Arduino board, fabricate the mechanical system based on stepper motors and then design the software of the Arduino card to control the motors and communicate with a graphical user interface. To address the particular objectives (2-9) an application was designed in Matlab with a graphical user interface which is based on the following operations: an input image is chosen and optionally an anisotropic diffusion smoothing is applied, then regions are processed by automatic or interactive segmentation and in these regions a vector field is created based on the local interpolation of manually traced trajectories or based on the gradient, then trajectories are created with Bézier curves by means of the field and finally the information of each brushstroke is sent to the robot by means of 8-bit chains where the length, colour and coordinates are coded, and only the control points of each Bézier curve are sent and then this is reconstructed on the Arduino card, which activates the motors by tracing each curve. Finally, successive layers of paint are applied, which consist of a series of brush-strokes guided by the vector field. Each brushstroke is approximated by Bézier curves to achieve brushstrokes as close as possible to the human stroke. The materials used were brushes, paper and acrylic paint.

The results obtained were as follows: for the particular objective (1) a step resolution of 6.25 microns was achieved, while the Arduino interface was connected at 9600 bits per second to the graphical user interface designed in Matlab where the code in the Arduino card consisted of 430 lines and the code in Matlab of 1500 lines. For the particular objectives (2-8) we were able to develop a graphical user interface where an image is opened and the steps mentioned above are applied by means of interactive buttons and sliding bars, achieving real time communication with the robot's electronic interface. As for the particular objective (9), an acrylic painting was achieved with pleasant aesthetic results.

KEY WORDS: Robotic painting, digital image processing, Bezier approximation, scattered data interpolation, anisotropic diffusion, k-media segmentation, corner detection.

Índice general

Resumen I
Abstract II
Lista de figuras XI
Lista de tablas XII
Lista de símbolos XIII
Glosario XIV

Introducción **1**
Antecedentes 1
Presentación del proyecto de tesis 1

1 Contexto de la investigación **3**
1.1 Sistema de estudio 3
1.2 Estado actual de la investigación 5
1.3 Revisión bibliográfica 7
1.3.1 Introducción 7
1.3.2 Materiales y métodos 7
1.3.3 Resultados 8
1.3.4 Discusión 17
1.4 Perspectiva transdisciplinaria 19
1.5 Contexto físico y económico 22
1.6 Contexto cultural y social 27
1.7 Contexto histórico 28
1.8 Justificación 32
1.9 Objetivo principal 32

1.10 Objetivos particulares 32
1.11 Tabla de congruencia 33

2 Marco teórico y metodológico **35**
2.1 Marco teórico 35
2.1.1 Introducción 35
2.1.2 Preparación de la imagen aplicando difusión anisotrópica 36
2.1.3 Segmentación automática usando K-medias 41
2.1.4 Segmentación interactiva 42
2.1.5 Diseño automático del campo de trazos 49
2.1.6 Diseño interactivo del campo de trazos 52
2.2 Marco metodológico 64

3 Aplicación de la metodología **68**
3.1 Actividad de investigación previa 1: interfaz gráfica Matlab 68
3.1.1 Introducción 68
3.1.2 Materiales y métodos 68
3.1.3 Resultados 68
3.2 Actividad de investigación previa 2: tarjeta electrónica del robot cartesiano . 75
3.2.1 Introducción 75
3.2.2 Materiales y métodos 76
3.2.3 Resultados 77
3.3 Actividad de investigación previa 3: programa de la tarjeta arduino 79
3.3.1 Introducción 79
3.3.2 Materiales y métodos 79
3.3.3 Resultados 80
3.4 Actividad de investigación previa 4: elementos mecánicos del robot 81
3.4.1 Introducción 81
3.4.2 Materiales y métodos 82
3.4.3 Resultados 82
3.5 Actividad de investigación 1: Preparación de la imagen utilizando filtrado anisotrópico 85

3.5.1 Introducción 85
3.5.2 Materiales y métodos 85
3.5.3 Resultados 86
3.6 Actividad de investigación 2: Segmentación automática 88
3.6.1 Introducción 88
3.6.2 Materiales y métodos 88
3.6.3 Resultados 90
3.7 Actividad de investigación 3: Segmentación interactiva 93
3.7.1 Introducción 93
3.7.2 Materiales y métodos 94
3.7.3 Resultados 95
3.8 Actividad de investigación 4: Diseño automático de campo vectorial 98
3.8.1 Introducción 98
3.8.2 Materiales y métodos 99
3.8.3 Resultados 99
3.9 Actividad de investigación 5: Diseño interactivo de campo vectorial 101
3.9.1 Introducción 101
3.9.2 Materiales y métodos 101
3.9.3 Resultados 102
3.10 Actividad de investigación 6: Generación de trazos 105
3.10.1 Introducción 105
3.10.2 Materiales y métodos 105
3.10.3 Resultados 108
3.11 Actividad de investigación 7: Detección de bordes 109
3.11.1 Introducción 109
3.11.2 Materiales y métodos 110
3.11.3 Resultados 110
3.12 Actividad de investigación 8: Aplicación de pintura 112
3.12.1 Introducción 112
3.12.2 Materiales y métodos 113
3.12.3 Resultados 114

3.13 Comparación con otros sistemas 116

4 Discusión, conclusiones y trabajos futuros 117

4.1 Discusiones, conclusiones y trabajos futuros de la actividad de investigación 1: Preparación de la imagen utilizando filtrado anisotrópico 117

4.2 Discusiones, conclusiones y trabajos futuros de la actividad de investigación 2: Segmentación automática 118

4.3 Discusiones, conclusiones y trabajos futuros de la actividad de investigación 3: Segmentación interactiva 118

4.4 Discusiones, conclusiones y trabajos futuros de la actividad de investigación 4: Diseño automático de campo vectorial 119

4.5 Discusiones, conclusiones y trabajos futuros de la actividad de investigación 5: Diseño interactivo de campo vectorial 120

4.6 Discusiones, conclusiones y trabajos futuros de la actividad de investigación 6: Generación de trazos 121

4.7 Discusiones, conclusiones y trabajos futuros de la actividad de investigación 7: Detección de bordes 121

4.8 Discusiones, conclusiones y trabajos futuros de la actividad de investigación 8: Aplicación de pintura 122

Bibliografía 132

ANEXOS 133

APORTACIONES CIENTÍFICAS 171

Índice de figuras

No. de Fig.	Descripción	Pág.
Capítulo 1 Contexto de la investigación		
1.1	Sistema de estudio de la tesis (Elaboración propia, 2019)	3
1.2	Límites del sistema de estudio (Elaboración propia, 2019)	5
1.3	Obras generadas a partir del seminario de arte cibernético, 1968-1973 (imágenes disponibles en la página web del museo Reina Sofía, consultado en 2019.) (Elaboración propia, 2019)	6
1.4	Obras presentadas para la competencia de RobotArt 2018, (Elaboración propia, 2019, imágenes disponibles en la página web de RobotArt, consultado en 2019.)	7
1.5	Clasificación de autores basada en el tipo de robot utilizado (Elaboración propia, 2019)	17
1.6	Visión rica de la perspectiva transdisciplinaria para abordar la investigación (Elaboración propia, 2019)	21
1.7	Países exportadores de cerámica ornamental en 2017 (Elaboración propia, 2019, imagen generada del Atlas of Economic Complexity de la Universidad de Harvard)	22
1.8	Total de personas ocupadas en área de artesanías: 1,222,480 (Elaboración propia 2019, a partir de datos del INEGI. Sistema de Cuentas Nacionales de México. Cuenta satélite de la cultura de México, 2016. Año Base 2013)	23
1.9	Proceso de producción de alfarería (Elaboración propia, 2019)	26
1.10	Maneras de solucionar el problema de la pintura robótica en la historia (Elaboración propia, 2019)	29

1.11 Obras representativas de pintura robótica a lo largo de la historia reportadas por: (a)Cohen (1988), (b)Cohen (1995), (c)Stein (2003), (d)Calinon, Epiney, y Billard (2005), (e)Ruchanurucks, Kudoh, Ogawara, Shiratori, y Ikeuchi (2007), (f) Aguilar y Lipson (2008), (g)P. A. Tresset y Leymarie (2012), (h)Deussen, Lindemeier, Pirk, y Tautzenberger (2012), (i)Lindemeier, Pirk, y Deussen (2013), (j)P. Tresset y Deussen (2014), (k)Lindemeier, Metzner, Pollak, y Deussen (2015), (l)Luo, Hong, y Chung (2016), (m)Song, Lee, Kim, Sohn, y Kim (2018), (n)Scalera, Seriani, Gasparetto, y Gallina (2018b), (o) Karimov *et al.* (2019) (Elaboración propia, 2019) 30

1.12 Obras representativas de pintura robótica a lo largo de la historia no reportadas, premiadas por RobotArt en 2016: (a) Obra de TAIDA, (b) Obra de CloudPainter, (c) Obra de RHIT. En 2017: (d) Obra de PIX18/Creative Machines Lab, (e) Obra de CloudPainter, (f) Obra de Christian H. Seidler. En 2018: (g) Obra de CloudPainter, (h) Obra de Custom Autonomous Robotic Painter, (i) Obra de Joanne Hastie (Elaboración propia, 2019) 31

Capítulo 2 Marco teórico y metodológico

2.1 Aplicación de la ecuación de difusión isótropa (Elaboración propia, 2019) . 38

2.2 Aplicación de la ecuación de difusión anisotrópica (Elaboración propia, 2019) 41

2.3 Procedimiento de la segmentación interactiva (Elaboración propia, 2019) . 43

2.4 Polinomios de Bernstein de grado 3 (Elaboración propia, 2019) 45

2.5 Segmentación de la región que encierra el contorno aproximado con curvas Bézier (Elaboración propia, 2019) 48

2.6 Trazo de Bresenham desde (0,1) a (4,3) (Elaboración propia, 2019) 49

2.7 Ejemplo de trazos pictóricos en una región de trabajo (Elaboración propia, 2019) 50

2.8 Representación gráfica de las funciones de ponderación (Elaboración propia, 2019) 51

2.9 Diagrama de flujo del diseño interactivo del campo (Elaboración propia, 2019) 53

2.10 Envolvente convexa de un conjunto de puntos (Elaboración propia, 2019) . 53

2.11 Curva Bézier cúbica y su derivada para 15 puntos (Elaboración propia, 2019) 54

2.12 La nueva celda de Voronoi del punto q que se superpone a las celdas de los puntos $x_1...x_5$ se indica con líneas discontinuas. También se muestran los triángulos de Delaunay correspondientes con líneas punteadas finas (Elaboración propia, 2019) 56
2.13 Interpolación dentro de la envolvente convexa (línea roja) usando vecinos naturales para los 15 puntos (círculos) de la Figura 2.11 (Elaboración propia, 2019) 59
2.14 Geometría de la extrapolación triangular (Elaboración propia, 2019) 60
2.15 Extrapolación lineal fuera de la envolvente convexa (negro = región rectangular y magenta = región triangular) para los datos de la Figura 2.13 (Elaboración propia, 2019) 64
2.16 Metodología de investigación a emplear, Perspectiva Transdisciplinaria (Td) (Elaboración propia, 2019) 65
2.17 Actividades previas a la investigación para obtener resultados (Elaboración propia, 2019) 66
2.18 Actividades específicas de investigación (Elaboración propia, 2019) 67

Capítulo 3 Aplicación de la metodología

3.1 Diseñador de interfaz gráfica de usuario GUIDE en Matlab. La interfaz gráfica de usuario facilita la interacción del usuario con la computadora. (Elaboración propia, 2019) 69
3.2 Pasos para desarrollar la interfaz gráfica de usuario (Elaboración propia, 2019) 70
3.3 Pasos para desarrollar los algoritmos de la interfaz gráfica (Elaboración propia, 2019) 71
3.4 Interfaz gráfica en Matlab (Elaboración propia, 2019) 72
3.5 Bloques principales de la interfaz gráfica (Elaboración propia, 2019) 73
3.6 Flujo de operaciones de procesamiento usando la interfaz gráfica. En círculos se muestra el correspondiente bloque en el que se encuentra la operación en la Figura 3.5 (Elaboración propia, 2019) 74
3.7 Posible orden de uso de la interfaz gráfica (Elaboración propia, 2019) 75
3.8 Visión rica de la tarjeta electrónica (Elaboración propia, 2019) 76

3.9 Metodología para seleccionar la tarjeta electrónica (Elaboración propia, 2019) 77
3.10 Tarjeta electrónica utilizada, (a) montaje de módulos de la tarjeta, (b) conexión de motores (Elaboración propia, 2019) 78
3.11 Editor de programa de la tarjeta Arduino (Elaboración propia, 2019) 79
3.12 Pasos para desarrollar el firmware de la tarjeta electrónica (Elaboración propia, 2019) 80
3.13 Operación entre computadora y microcontrolador para cada curva trazada (Elaboración propia, 2019) 81
3.14 Pasos para desarrollar los elementos mecánicos del robot cartesiano (Elaboración propia, 2019) 82
3.15 Plano de la base del robot cartesiano (Elaboración propia, 2019) 83
3.16 Elementos mecánicos del robot cartesiano (Elaboración propia, 2019) 84
3.17 Difusión anisotrópica en imagen en color (Elaboración propia, 2019) 86
3.18 Entropía de Shannon para la difusión (Elaboración propia, 2019) 87
3.19 Difusión anisotrópica e histograma (Elaboración propia, 2019) 87
3.20 Segmentación *k-medias* en imagen con pocos elementos (Elaboración propia, 2019) 90
3.21 Imagen con varias regiones, donde k indica el número de grupos, y PRI indica el índice aleatorio probabilístico, el cual evalúa la segmentación. (a)–(d) Segmentación *k-medias*, (e)–(h) Segmentación *k-medias*, aplicando previamente 200 iteraciones de difusión anisotrópica, (i)–(l) Segmentación manual de referencia utilizada para evaluar la segmentación *k-medias* (Elaboración propia, 2019) 92
3.22 Selección de puntos consecutivos sobre el contorno (Elaboración propia, 2019) 95
3.23 Segmentación del contorno mediante la detección de esquinas (Elaboración propia, 2019) 96
3.24 Ajuste de curva Bézier de grado n para un segmento del contorno (Elaboración propia, 2019) 97
3.25 Resultado de la segmentación interactiva de región (Elaboración propia, 2019) 98

3.26 Campo vectorial obtenido al seguir los pasos *Abre imagen* → *Campo automático* (Elaboración propia, 2019) 100
3.27 Campo vectorial obtenido al seguir los pasos *Abre imagen* → *Suaviza* →*Campo automático* (Elaboración propia, 2019) 100
3.28 Campo vectorial obtenido al seguir los pasos *Abre imagen* → *Segmentación interactiva* →*Campo automático* (Elaboración propia, 2019) 101
3.29 Diseño interactivo del campo vectorial mediante interpolación. Trazo manual (- - - - -), aproximación Bézier de grado 4 (—) del trazo (Elaboración propia, 2019) 103
3.30 Diseño para el fondo usando varios trazos y aproximaciones (trazo manual: línea punteada, aproximación Bézier: línea continua) (Elaboración propia, 2019) 104
3.31 Influencia de las curvas en la nervadura principal sobre el campo resultante (Elaboración propia, 2019) 105
3.32 Metodología para generar los trazos (Elaboración propia, 2019) 107
3.33 Generación de diversas trayectorias para pinceladas (Elaboración propia, 2019) 108
3.34 Aplicación de trazos en subregión (Elaboración propia, 2019) 109
3.35 Detección de bordes (Elaboración propia, 2019) 111
3.36 Ejemplo de detección de bordes en una región de interés (Elaboración propia, 2019) 112
3.37 Metodología para aplicar la pintura (Elaboración propia, 2019) 114
3.38 Proceso de aplicación de pintura (Elaboración propia, 2019) 115
3.39 Pintura terminada (Elaboración propia, 2019) 115
3.40 Comparación con otros sistemas, (a) P. Tresset y Deussen (2014), (b) Lindemeier *et al.* (2015), (c) Lindemeier, Spicker, y Deussen (2016), (d) Scalera *et al.* (2018b), (e) pintura propuesta, (f) mapa de trazos basado en el gradiente, (g) mapa de trazos propuesto (Elaboración propia, 2019) 116

Índice de tablas

No. de Tabla	Descripción	Pág.

Capítulo 1 Contexto de la investigación

1.1 Revisión bibliográfica sobre pintura artística robótica. (Elaboración propia, 2019) 13

1.2 Producción de cerámica ornamental por estados en México (Elaboración propia 2019, con datos de Gomez (2010)) 23

1.3 Perfil de consumidores de artesanías en México (Elaboración propia 2019, con datos de Dávila (2017)) 28

1.4 Tabla de congruencia, (Elaboración propia, 2019) 34

Capítulo 3 Aplicación de la metodología

3.1 Especificaciones del robot, (Elaboración propia, 2019) 85

Lista de símbolos

∇I	Gradiente de I
$\frac{\partial I}{\partial t}$	Derivada parcial de I con respecto a t
$\|\mathbf{x}_j - \mathbf{v}_i\|^2$	Distancia cuadrática entre el j-ésimo elemento $\mathbf{x}_j$ y el i-ésimo elemento $\mathbf{v}_i$
$B_j^n(t)$	Polinomio de Bernstein de grado n
$\Delta I = I_{xx} + I_{yy}$	Laplaciano de I
I_{xx}	Segunda derivada parcial de I con respecto a x
$\nabla \cdot \mathbf{j}$	Divergencia de j
$n!$	Factorial de n
B^T	Transpuesta de la matriz B
B^{-1}	Inversa de la matriz B
$\int \int_{\Omega_r} \phi(\mathbf{x}, \mathbf{y}) d(\mathbf{y})$	Integral de $\phi(\mathbf{x}, \mathbf{y})$ en la región definida por $\Omega_r(\mathbf{x})$
$\phi(\mathbf{x}, \mathbf{y})$	Ventana cuadrada con elementos **y** y con centro en **x**
$V \bigcap U$	Intersección del conjunto U con el conjunto V
$c'(t)$	Primera derivada de $c(t)$
$c''(t)$	Segunda derivada de $c(t)$

Glosario

Binarización	Es una técnica de segmentación simple que permite separar los pixeles de una imagen en escala de grises en dos categorías a partir de un valor umbral de intensidad (Otsu, 1979)
Diagramas de Voronoi:	Objetos geométricos que particionan el espacio euclídeo, estudiados por el matemático ruso Gueorgui Voronói y por el meteorólogo estadounidense Alfred H. Thiessen (Secord, 2002)
Gradiente de la imagen:	Derivada parcial de la intensidad de la imagen (Gonzalez, Woods, y Eddins, 2009)
Spline:	Curva diferenciable definida en porciones mediante polinomios (Biswas y Lovell, 2007)
Renderizado no fotorrealístico:	Proceso de gráficos por computadora que consiste en interpretar una imagen en otra imagen artística a la que se agregan texturas, materiales, etc. (Kyprianidis, 2013)
Retroalimentación visual:	Consiste en tomar imágenes del avance de la pintura usando una cámara y pintar donde falte (Ruchanurucks *et al.*, 2007)
Robot Nao:	Robot humanoide programable y autónomo, desarrollado por Aldebaran Robotics, una compañía de robótica francesa con sede en París (Singh, Baranwal, y Nandi, 2017)
Robot eDavid:	Brazo robot industrial adaptado para pintar por Lindemeier *et al.* (2016)

Robot Paul:	Pequeño brazo robot para pintar fabricado por P. A. Tresset y Leymarie (2012)
Transdisciplinariedad:	Concierne a lo que está a la vez entre las disciplinas, a través de las diferentes disciplinas y más allá de toda disciplina. Su finalidad es la comprehensión del mundo presente en el cual uno de los imperativos es la unidad del conocimiento, los rasgos fundamentales de la actitud transdisciplinaria son rigor, apertura y tolerancia (Nicolescu, 2006)
Triangulación de Delaunay	Red de triángulos conexa y convexa que cumple la condición de Delaunay. Se le denomina así por el matemático ruso Borís Nikolaevich Delone quien lo ideó en 1934 (Watson, 2001)

Introducción

Antecedentes

La cerámica tradicional en México se encuentra en crisis, debido a la producción masiva, lo que provoca que la gente abandone el oficio. Sin embargo, las artesanías han permitido que miles de familias subsistan en zonas remotas, donde no hay otra actividad económica. Por ello, en esta tesis se propone generar tecnología robótica propia para coadyuvar en la producción de artesanías, buscando que el artesano tenga herramientas tecnológicas de apoyo para desarrollar sus productos, no que el robot sea autónomo.

Por otra parte, el uso reciente de la tecnología en el arte abre la oportunidad de aplicarla en artesanías donde la pintura robótica no ha sido aplicada aún. De esta manera, en esta tesis, encuentran cabida el arte y la tecnología, dos rostros de la creatividad humana, los cuales se encuentran estrechamente relacionados, a pesar de las diferencias que en apariencia tienen entre sí.

Presentación del proyecto de tesis

Este trabajo se plantea como un apoyo a la producción de artesanía cerámica en el cual se propone el uso de un robot cartesiano para automatizar ciertos trazos de motivos a consideración del artesano, es decir utilizándolo como medio que facilite el desarrollo del trabajo artesanal. Se propone como solución a uno de los problemas que enfrenta nuestro país en el contexto de la globalización, a saber la falta de desarrollo tecnológico, siendo el sector artesanal uno de los sectores productivos más rezagados (Díaz-Bautista, 2006). Por otro lado, también se puede emplear como herramienta tecnológica auxiliar en la producción artística.

A la humanidad le ha intrigado la posibilidad de construir criaturas artificiales. Para los antiguos griegos esta posibilidad fue proporcionada por el techné, el procedimiento que Aristóteles concibió para crear lo que la naturaleza encuentra imposible de lograr (Moura, 2016). Por lo tanto, bajo este punto de vista, techné se establece entre la naturaleza y la humanidad como una mediación creativa. Este fue el camino que siguió Norbert Wiener al abrir la perspectiva cibernética, entendida como el estudio unificado de organismos y máquinas (Wiener, 1988).

El arte del futuro, de acuerdo con Moles (1968), es el arte de la sociedad de los Sistemas, y habrá de reposar necesariamente sobre máquinas capaces de manipular la complejidad. La máquina ofrece, por tanto, posibilidades insospechadas para abrir nuevos caminos a la expresión del hombre.

Capítulo 1

Contexto de la investigación

1.1. Sistema de estudio

En la Figura 1.1 se muestra la descripción general del sistema de estudio, el cual se explica a continuación.

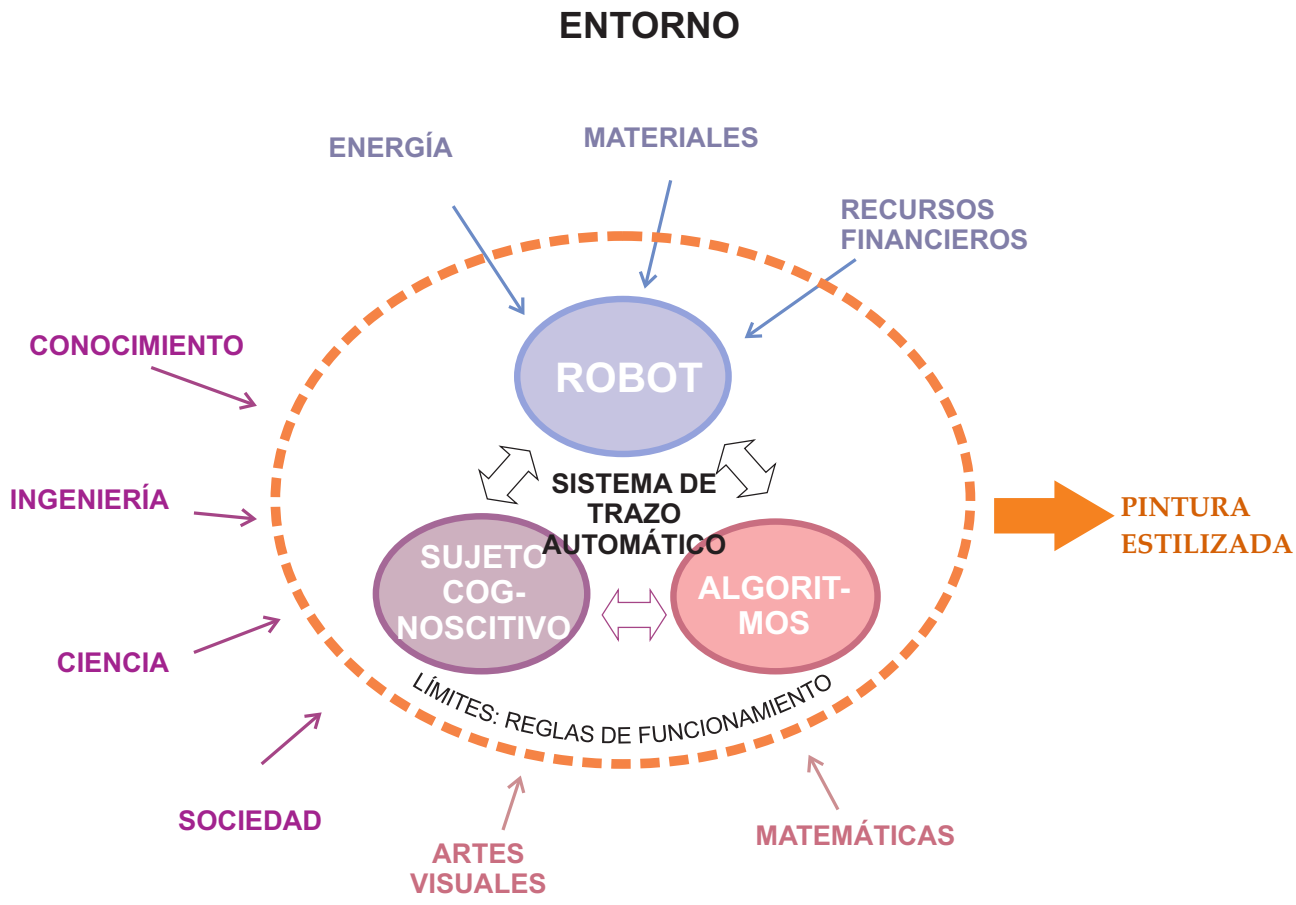


Figura 1.1: Sistema de estudio de la tesis (Elaboración propia, 2019)

De acuerdo con Bunge (2001) un sistema está formado por la composición, el entorno, la estructura y la frontera. En la Figura 1.1, la composición del sistema está formada por los subsistemas: robot, algoritmos y sujeto cognitivo. La estructura son las relaciones entre los subsistemas, así como entre éstos y los elementos del entorno. El entorno es la colección de elementos que actúan sobre los componentes del sistema o a la inversa. El límite es el conjunto de elementos del sistema directamente enlazados con los componentes de su entorno mediante la operación de comunicación utilizando un lenguaje (Luhmann, 1996, p.70).

En la Figura 1.2 se muestran los límites como intersecciones con el entorno en el Sistema de trazado automático, donde los elementos de este se interrelacionan con los elementos de aquel, transformando la energía y la información del exterior para producir la salida "pintura estilizada".

Por tanto, el propósito de este trabajo es describir los procesos y las relaciones para lograr tal fin.

Figura 1.2: Límites del sistema de estudio (Elaboración propia, 2019)

1.2. Estado actual de la investigación

Uno de los precedentes más importantes del uso tecnológico en el arte se sitúa en el seminario Generación Automática de Formas Plásticas del Centro de Cálculo de la Universidad de Madrid durante los años 1968-1973 (Castaños Alés, 2000) . En este seminario se produjeron varios trabajos de arte cibernético, entre los que destacan los del pintor Manuel Barbadillo y los del escultor José Luis Alexanco. En la Figura 1.3(a) se muestra una obra de Barbadillo y en la Figura 1.3(b) una de Alexanco las cuales resultaron del seminario mencionado.

(a) Obra de Barbadillo. *Perena, 1968-1979*

(b) Obra de Alexanco. *Escultura MOUVNT, 1972*

Figura 1.3: Obras generadas a partir del seminario de arte cibernético, 1968-1973 (imágenes disponibles en la página web del museo Reina Sofía, consultado en 2019.) (Elaboración propia, 2019)

Los mecanismos de regulación y control mediante computadora usados para producir los trabajos de arte dieron lugar al uso del término arte cibernético. El cual deriva del uso que Norbert Wiener dio al término cibernética para referirse a las comunicaciones y al gobierno de las máquinas (Wiener, 1988).

Otro de los antecedentes destacados en el uso de arte tecnológico, tiene su origen en la década de los 70 cuando Harold Cohen (1988) creó el programa de cómputo AARON enfocado a crear arte moderno de manera semiautónoma usando un robot cartesiano.

Conforme se desarrollaron las ramas de la ciencia relacionadas con la cibernética, entre otras, la teoría de la información, la robótica, la teoría del control y la computación, aparecieron aplicaciones de dichas ramas al arte como se muestra en la revisión bibliográfica de la siguiente sección.

Los trabajos más recientes relacionados con este estudio se pueden consultar en la página web de RobotArt, el cual organizó una convocatoria de 100 mil dólares durante 2018, en la que 19 equipos de investigación de todo el mundo pintaron obras artísticas usando robots. En la Figura 1.4 se observan dos ejemplos de dichos trabajos.

(a) Obra de CloudPainter

(b) Obra de A Roboto

Figura 1.4: Obras presentadas para la competencia de RobotArt 2018, (Elaboración propia, 2019, imágenes disponibles en la página web de RobotArt, consultado en 2019.)

1.3. Revisión bibliográfica

1.3.1. Introducción

Esta revisión bibliográfica investiga sistemas robots capaces de producir trazos con valor artístico mediante una serie de herramientas de la inteligencia artificial. El objetivo es obtener información aplicable a la automatización de procesos artesanales a voluntad del artesano.

1.3.2. Materiales y métodos

A fin de obtener información sobre el tema de investigación se consultó la base de datos del CONRICYT y de Google Académico. En CONRICYT se activaron las opciones de búsqueda, LÍMITE DE BÚSQUEDA: Académico y revisado por pares y TIPO DE CONTENIDO: Artículo de revista. La fecha de consulta de CONRICYT y de Google Escolar fue en octubre de 2018. Las Palabras clave en CONRICYT y en Google Escolar fueron: Artistic painting robot y Robot drawing art.

En cuanto a criterios de inclusión de artículos, para Conricyt se consultaron los primeros 20 resultados ordenados por relevancia y para Google Escolar se consultaron los primeros 50 resultados de la búsqueda ordenados también por relevancia.

A continuación se seleccionaron los artículos examinando su título y su resumen, cuando fue necesario se examinó el texto completo. No se discriminaron artículos por su nú-

mero de citaciones ya que la cantidad de artículos encontrados fue menor a 100 para esta línea de investigación.

El criterio usado para incluir un artículo fue que dicho artículo reportara el empleo de robot para aplicar pintura, o para realizar trazos con pintura, pluma o tableta. Pero el criterio más importante fue la intención artística del trazo o de la aplicación de la pintura.

Después se consultaron las citaciones a los artículos seleccionados, toda vez que son pocas las citas a cada artículo, menor a 20 en general, debido a que el tema de investigación tiene pocos años de existencia.

Los artículos que resultaron elegidos para formar la base de datos se exponen a continuación.

1.3.3. Resultados

Al realizar la revisión bibliográfica de la pintura artística mediante robot se encontraron los siguientes hallazgos. Los resultados se exponen por fecha.

Scalera, Seriani, Gasparetto, y Gallina (2019) utilizaron dos técnicas para guiar los trazos, usaron splines aleatorios y sombreado de líneas paralelas. Para la manipulación del pincel usaron un brazo robot.

Karimov *et al.* (2019) emplearon un robot cartesiano, donde para la aplicación de la pintura emplearon un dispositivo mezclador conectado al pincel y una bomba de agua. Los algoritmos que usaron fue el gradiente de la imagen para guiar los trazos y la generación de puntos semilla mediante una retícula.

Scalera *et al.* (2018a; 2018b) reportaron un brazo robótico usado para pintar acuarela utilizando diferentes técnicas de renderizado no fotorrealístico a fin de producir trayectorias de pintado.

Dong, Li, Ning, Zhang, y Lu (2018) realizaron el trazo de líneas estilizadas de un rostro utilizando un brazo robot.

Song *et al.* (2018) usaron un brazo robótico industrial para dibujar con una pluma de forma semiautónoma sobre una superficie arbitraria. Los trazos se basan en curvas spline de Bézier. El resultado de su trabajo es agradable visualmente, sin usar retroalimentación visual.

Xue y Liu (2017) extrajeron características faciales para realizar el trazo de líneas de

boceto, para ello usaron una plataforma robótica de brazo articulado y un sistema de visión.

Singh *et al.* (2017) utilizaron un robot humanoide para el trazo de esbozos. Abordaron tres problemas: extracción de contornos significativos, definición de plano de la cámara con respecto al efector y cinemática inversa del robot.

Munoz, Avalos, y Ramos (2017) trazaron bocetos simples de objetos usando un robot humanoide Nao, también analizaron la generación de movimiento del robot.

Ye *et al.* (2017) utilizaron una cámara para adquisición de imagen a fin de realizar un retrato mediante el trazo de líneas del contorno. Para ello usaron un sistema robot con brazos articulados.

Lindemeier *et al.* (2016) segmentaron la imagen de entrada en regiones, las cuales dividieron en capas para representar los detalles gruesos, medios y finos. Luego asignaron un estilo de pintura a cada región, incluyendo una paleta de colores, funciones para orientar los trazos dentro de las capas, y parámetros de trazo previstos.

Huang *et al.* (2016) propusieron el uso de un método que combina la binarización local y la global para extraer características del rostro a fin de dibujar utilizando una trayectoria optimizada mediante un brazo robótico industrial.

Singh y Nandi (2016) utilizaron un robot Nao para calibrar la relación entre los puntos del plano de la imagen y la posición final del efector del robot, a fin de usar dicha calibración para dibujo de bocetos.

Luo *et al.* (2016) usaron un brazo robot para pintar imágenes coloridas con un sistema de control visual usando cinco colores básicos. El proceso que usaron consta de la aplicación de una fina capa de pintura acrílica, la cual se refinó superponiendo capas de pintura.

Rodrigues, Cruz, Dias, y Silva (2016) realizaron trazo de líneas de retrato usando un brazo robot, para ello utilizaron una cámara y procesamiento de imagen para detectar los contornos a trazar.

Por medio de un modelo de control realimentado de torque Lau, Cheng, Baltes, y Anderson (2015) investigaron la presión de la mano de un robot de 4 grados de libertad (gdl) al realizar un bosquejo de dibujo sobre una tableta digitalizadora.

Watanabe, Numakura, Nishide, Gouko, y Kim (2015), generaron movimientos de di-

bujado utilizando un modelo de aprendizaje basado en redes neuronales trazando formas básicas en un robot humanoide Nao.

Lindemeier *et al.* (2015) emplearon pintura acrílica usando el robot eDavid para propósitos de pintura artística además el mecanismo de colocación de trazo se basa en los diagramas de Voronoi.

Jain, Gupta, Kumar, y Sharma (2015) utilizaron un brazo robot equipado con capacidad de detección de fuerza para dibujar retratos en una superficie no calibrada y de forma arbitraria.

Utilizando el robot eDavid, P. Tresset y Deussen (2014) obtuvieron retratos con un estilo más propio del robot. Su técnica consiste la superposición de capas de pintura blanca sobre un fondo oscuro usando tres tamaños de pincel y obteniendo así valores tonales en gris.

En los trabajos relacionados con el trazado de letras chinas empleando pincel y tinta se puede mencionar a Chao *et al.* (2014) que utilizaron reconocimiento de movimientos de un ser humano por medio de una cámara y luego aplicaron cinemática inversa para realizar los trazos vía un pequeño robot de 6 gdl.

Nishide, Mochizuki, Okuno, y Ogata (2014) insertaron una pausa en el movimiento de dibujado usando un robot Nao en el desarrollo de aprendizaje de imitación de trazos sencillos sobre una tableta.

En este trabajo Lindemeier *et al.* (2013) combinaron dos métodos para simular el proceso de pintado, mediante un conjunto predefinido de pinceladas y creando pinceladas dinámicas usando sugerencias semánticas.

P. A. Tresset y Leymarie (2013) presentaron el robot Paul, un robot pequeño, el cual dibuja rostros humanos sobre una hoja de papel usando un bolígrafo con la técnica de dibujo de Tresset y empleando retroalimentación visual.

En este trabajo Mochizuki, Nishide, Okuno, y Ogata (2013) entrenaron un robot NAO mediante redes neuronales para que dibujara figuras básicas sobre una tableta.

En este trabajo Deussen *et al.* (2012) evaluaron técnicas de representación no fotorrealista mediante el robot eDavid. Compararon dos métodos para simular el proceso de pintado: mediante un conjunto predefino de pinceladas y usando pinceladas con la convolución integral de línea.

Utilizando el robot Paul, una instalación robótica que produce bosquejos observacionales de rostros, P. A. Tresset y Leymarie (2012) presentan los resultados de los trazos que produce el robot, así como una descripción general del sistema.

Jean-Pierre y Saïd (2012) acondicionaron un brazo robot industrial para realizar trazos sobre un lienzo. El sistema dibuja el contorno de rostros de personas usando el gradiente de la imagen.

Gurpinar, Alasag, y Kose (2012) utilizaron un robot humanoide Nao para reproducir todo el proceso de pintura mediante un sistema de visión y dedos. La novedad del estudio radicó en el uso de un asistente humano interactuando con el robot.

Mohan *et al.* (2011) utilizaron un robot humanoide iCub para tratar de enseñarle a realizar trazos sencillos mediante la imitación del movimiento humano.

Inspirado en la metodología del renderizado no fotorrealista Y. Lu, Lam, y Yam (2009) automatizaron un dibujo a pluma y tinta basado en la retroalimentación visual, usando un algoritmo para la planificación de trayectorias en un sistema robótico de 5 gdl mediante interpolación del gradiente local.

Kudoh, Ogawara, Ruchanurucks, y Ikeuchi (2009) usaron un robot con múltiples dedos para intentar reproducir todo el procedimiento involucrado en la pintura humana, la obtención de un modelo 3D, composición de un modelo de imagen, y aplicación de pintura, en este caso pintaron contornos inclinando el pincel sobre el lienzo y midiendo la presión del pincel con sensores.

Lin, Chuang, y Mac (2009) desarrollaron un sistema para generar retratos humanos usando un robot humanoide de dos brazos. El sistema de generación de retratos convierte una imagen en segmentos de línea para formar trazos de línea de un retrato.

Lin, Mac, y Chuang (2009) presentan un robot humanoide, que mediante la programación de sus brazos y con la ayuda de una cámara, traza retratos artísticos sencillos utilizando procesamiento de la imagen y planeación de trayectoria.

Aguilar y Lipson (2008) reportó un sistema robótico que producía pinturas con acrílico usando un lienzo. Utilizó un brazo articulado y algoritmos de aprendizaje máquina para trazar pinceladas.

Kudoh, Ogawara, Ruchanurucks, y Ikeuchi (2007) presentaron un procedimiento para manipular un pincel mediante sensores colocados en la mano de un robot humanoide,

además realizaron retroalimentación visual para sujetar el pincel.

Ruchanurucks *et al.* (2007) buscaron nuevas tecnologías de visión para la segmentación de objetos, percepción del color, orientación de las pinceladas y propusieron el uso de la retroalimentación y de un sensor de fuerza para comprobar si la punta del pincel tocaba el lienzo o no. El área de los objetos es señalada por un usuario y emplean algoritmo de K medias y agrupamiento de distancia máxima para segmentar objetos a color. Para generar las pinceladas usan interpolación del gradiente más fuerte en ventanas.

Calinon *et al.* (2005) presentaron un trabajo en el que un robot de 4 grados de libertad dibuja retratos artísticos. La aplicación es puramente para entretenimiento y los movimientos del robot y sus dibujos siguen un estilo característico de los seres humanos. Aquí resalta la forma en que rellenan áreas por medio de un patrón de sombreado en diagonal. Los retratos son sólo imágenes binarias es decir en blanco y negro.

Coristine y Stein (2004) propusieron pintar con brazo robot controlado remotamente por usuarios de todo el mundo a través de una página web con una interfaz gráfica de usuario. El sistema solo realizaba trazos sencillos, debiéndose diseñar la trayectoria y color.

Stein (2003) propuso un sistema robótico manejado online para que los usuarios operaran el sistema desde una posición remota, el robot funcionó mediante una interfaz gráfica para hacer trazos sencillos sobre un lienzo.

Srikaew *et al.* (1998) exploraron la relación entre tecnología y creatividad por medio de un robot de 4 grados de libertad, analizando el problema de seguimiento e imitación de un artista.

Cohen (1988) comenzó la investigación de la pintura robótica, reproduciendo mediante modelos representacionales de diversos objetos sin la entrada directa de instrucciones humanas.

En la Tabla 1.1 se muestra de forma resumida la revisión bibliográfica en orden cronológico. En la columna **Dato analizado** se muestra el tipo de producto artístico realizado y en la columna **Método** se muestra el procedimiento usado, computacional o electromecánico.

Tabla 1.1: Revisión bibliográfica sobre pintura artística robótica. (Elaboración propia, 2019)

Dato analizado	**Tipo de robot**	**Método**
Sombreado con tinta negra	Brazo robot	Splines aleatorios y patrón de sombreado de líneas diagonales (Scalera *et al.*, 2019)
Pintura completa con acrílico	Robot cartesiano	Los algoritmos que usaron fue el gradiente de la imagen para guiar los trazos y la generación de puntos semilla mediante una retícula (Karimov *et al.*, 2019)
Pintura con acuarela	Brazo robot	Detector de bordes, sombreado y pinceladas aleatorias (Scalera, Seriani, Gasparetto, y Gallina, 2018a), (Scalera *et al.*, 2018b)
Trazo de líneas de retrato	Brazo robot	Estilizado exagerado de características faciales (Dong *et al.*, 2018)
Dibujo usando pluma	Brazo robot	Superficie arbitraria. Curvas spline de Bézier (Song *et al.*, 2018)
Líneas de boceto	Brazo articulado	Extracción de características de rostros (Xue y Liu, 2017)
Trazo de bocetos	Robot humanoide	Contornos significativos, cinemática inversa (Singh *et al.*, 2017)
Bocetos simples	Robot humanoide	Generación de movimiento del robot (Munoz *et al.*, 2017)
Líneas de boceto	Robot humanoide	Adquisición y procesamiento de imagen (Ye *et al.*, 2017)
Mezcla y aplicación de pintura	Brazo robot	Uso de técnicas de renderizado no fotorrealístico (Luo *et al.*, 2016)
Pintura completa con acrílico	Brazo robot	Descomposición de la imagen en un conjunto de regiones y capas (Lindemeier *et al.*, 2016)
Dibujo de rostro en blanco y negro	Brazo industrial	Combinación de binarización local y global (Huang *et al.*, 2016)

Continúa en la página siguiente

Continuación de la tabla

Dato analizado	**Tipo de robot**	**Método**
Posición de efector del robot	Robot NAO	Calibración de puntos del plano imagen (Singh y Nandi, 2016)
Esbozo de retrato	Brazo robot	Detección de contornos (Rodrigues *et al.*, 2016)
Dibujo de líneas de retrato	Brazo robot	Detección de bordes y control de fuerza (Jain *et al.*, 2015)
Presión sobre una tableta digitalizadora	Brazo de 4 GDL	Control realimentado de torque (Lau *et al.*, 2015)
Dibujo de formas básicas	Robot NAO	Disminución de error mediante un modelo dinámico (Watanabe *et al.*, 2015)
Pintura completa con acrílico	Brazo, eDavid	Diagramas de Voronoi (Lindemeier *et al.*, 2015)
Retratos con calidad artística	Brazo, eDavid	Superposición de capas de pintura (P. Tresset y Deussen, 2014)
Trazo de letras chinas	Brazo de 6 GDL	Cinemática inversa para realizar los trazos (Chao *et al.*, 2014)
Trazos sencillos sobre una tableta	Robot NAO	Aprendizaje por imitación (Nishide *et al.*, 2014)
Pintura completa	Brazo, eDavid	Usa sugerencias semánticas para generar pinceladas (Lindemeier *et al.*, 2013)
Dibujo de rostros	Pequeño robot Paul	Trazo de bolígrafo sobre hoja de papel con retroalimentación (P. A. Tresset y Leymarie, 2013)
Dibujo de figuras básicas sobre tableta	Robot NAO	Entrenamiento mediante redes neuronales (Mochizuki *et al.*, 2013)
Prueba de pintado	Robot humanoide	Interacción humano robot (Gurpinar *et al.*, 2012)

Continúa en la página siguiente

Continuación de la tabla

Dato analizado	**Tipo de robot**	**Método**
Pintura completa	Robot eDavid	Las pinceladas se basan en la técnica de representación no fotorrealista (Deussen *et al.*, 2012)
Bosquejo observacional de rostros	Pequeño robot Paul	Mediante líneas de saliencia se realiza el trazo (P. A. Tresset y Leymarie, 2012)
Dibujo del contorno de rostros	Brazo robótico	Contorno basado en el gradiente de la imagen (Jean-Pierre y Saïd, 2012)
Dibujo de trazos sencillos	Robot iCub	Imitación del movimiento humano (Mohan *et al.*, 2011)
Dibujo a pluma	Cartesiano	Retroalimentación visual (Y. Lu *et al.*, 2009)
Trazo de líneas curvas	Brazo c/dedos	Detección de contacto del pincel con sensores (Kudoh *et al.*, 2009)
Líneas de retratos sencillos	Brazo robot	Procesamiento de imagen y planeación de trayectoria (Lin, Chuang, y Mac, 2009)
Trazo de retratos sencillos	Humanoide	Procesamiento de imagen y planeación de trayectoria (Lin, Mac, y Chuang, 2009),
Pintura al acrílico	Brazo robot	Algoritmos genéticos (Aguilar y Lipson, 2008)
Manipulación de pincel	Humanoide	Uso de sensores en la mano de robot (Kudoh *et al.*, 2007)
Simulación del proceso de trazo	Humanoide	Uso de algoritmos para percepción visual y sensor de fuerza (Ruchanurucks *et al.*, 2007)
Dibujo en blanco y negro	Robot de 4 GDL	Rellenado de área por patrón de sombreado diagonal (Calinon *et al.*, 2005)
Trazo diseñado remotamente	Brazo robot	Se indica trayectoria mediante una interfaz gráfica (Coristine y Stein, 2004)

Continúa en la página siguiente

Continuación de la tabla

Dato analizado	**Tipo de robot**	**Método**
Trazos sencillos sobre lienzo	Brazo robótico	Manejo online mediante interfaz (Stein, 2003)
Trazos sencillos	Humanoide de 4GDL	Seguimiento a trazos humanos (Srikaew *et al.*, 1998)
Pinturas diversas	Cartesiano	Mediante modelos se obtiene cierta independencia del robot (Cohen, 1988)

De la revisión bibliográfica sobre pintura artística robótica se puede realizar la clasificación mostrada en la Figura 1.5 basada en el tipo de robot.

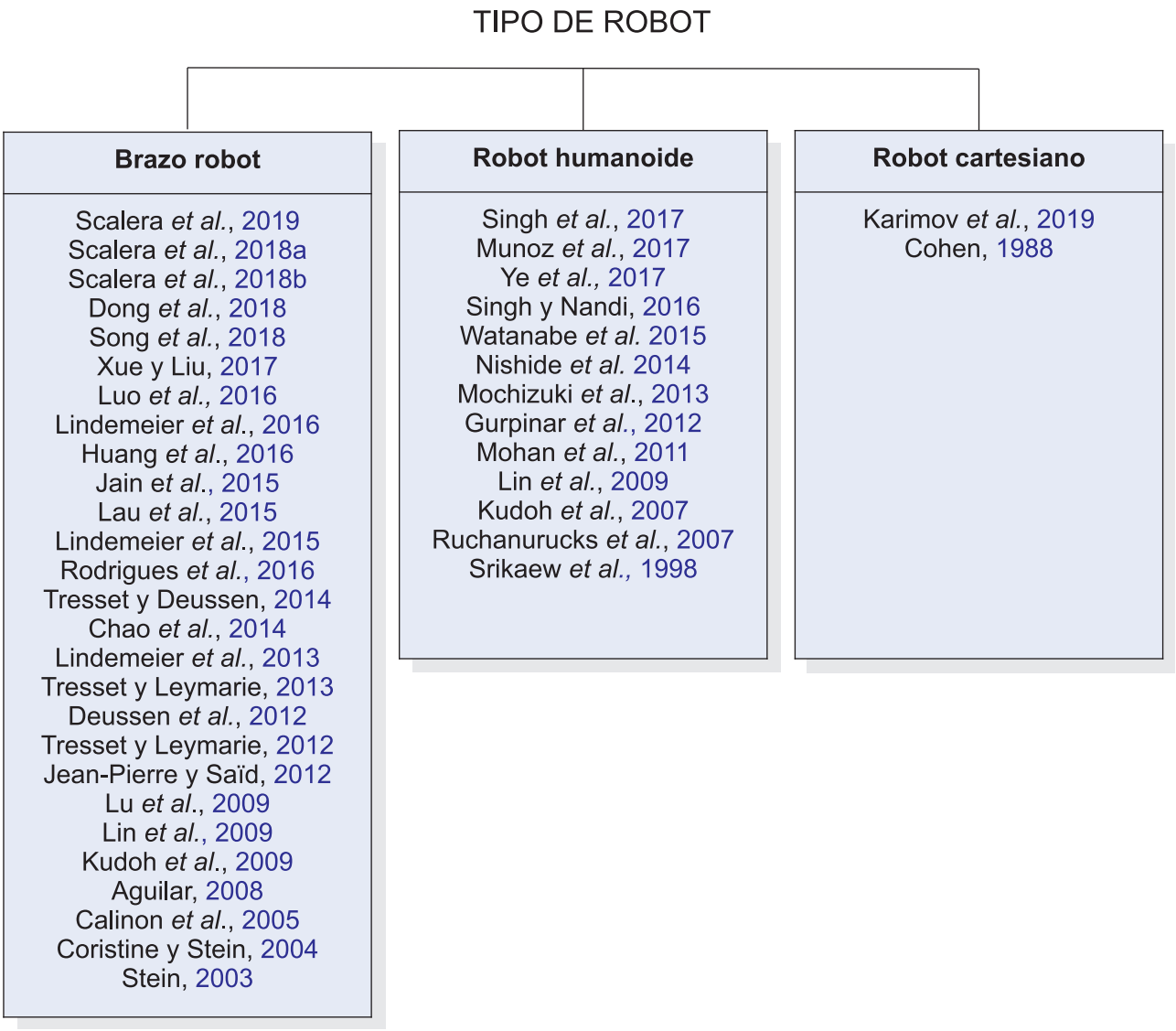


Figura 1.5: Clasificación de autores basada en el tipo de robot utilizado (Elaboración propia, 2019)

1.3.4. Discusión

Mediante la revisión bibliográfica se obtuvo información sobre el trabajo actual de los robots con intención artística. Sobre todo en lo relacionado a las técnicas y métodos de procesamiento de imagen y algoritmos para resolver el problema estético en la aplicación de trazos. Por ejemplo, en cuanto al procesamiento de la imagen, cómo orientar el trazo de la pintura en una región, cómo dividir regiones significativas. Lo que faltaría reportar es cómo realimentar el sistema con una cámara de video. En cuanto al movimiento mecánico la información no se pudo aplicar en esta tesis debido a que es un robot cartesiano y lo

investigado es sobre robots humanoides o brazos articulados.

Un robot cuyo significado es 'trabajo', agiliza y aumenta la eficiencia de procesos, aumentando la calidad de los productos y disminuyendo costos. De este modo, si se aplica en el sector artesanal podría resolver el problema de los precios elevados de los objetos decorados. Del mismo modo disminuir el tiempo de producción al pintar una figura con motivos repetitivos o al producir de forma masiva.

Esta revisión le puede servir al sector de manufactura que produce muebles de madera, losetas de cerámica, alfarería, y otros productos donde se aplique decoración ornamental. Especialmente en decorados donde los diseños sean repetitivos o permitan automatizarse disminuyendo costos de producción. También puede ser útil en la generación no autónoma de trabajo artístico para quienes investigan la incorporación de tecnología en el arte.

1.4. Perspectiva transdisciplinaria

Para Nicolescu (2006) la disciplinariedad es la organización del conocimiento científico a partir de campos especializados del saber.

La pluridisciplinariedad consiste en el estudio de un objeto de una sola y misma disciplina, por varias disciplinas a la vez. Por ejemplo, una pintura de Giotto puede ser estudiada por la historia del arte, por la física, la química, la historia de las religiones, la historia de Europa y la geometría.

La interdisciplinariedad se refiere a la transferencia de métodos de una disciplina a otra.

La transdisciplinariedad, indica, lo que está a la vez entre, a través y más allá de toda disciplina. Su objetivo o finalidad es la comprensión del mundo actual, donde uno de sus imperativos es la unidad del conocimiento. De acuerdo con Nicolescu, existen tres axiomas de la metodología de la transdisciplinariedad:

El axioma ontológico: Existen en la naturaleza y en nuestro conocimiento de la naturaleza, diferentes niveles de realidad y correspondientemente, diferentes niveles de percepción.

El axioma lógico: El paso de un nivel de realidad a otro es asegurado por la lógica del tercero incluido.

El axioma epistemológico: La estructura de la totalidad de niveles de realidad y de percepción, es una estructura compleja: cada nivel es lo que es porque todos los niveles existen al mismo tiempo.

En esta tesis se aborda la disciplina del arte desde la perspectiva de las ciencias exactas. Es pues su tratamiento, según Nicolescu, interdisciplinario, ya que se transfieren los métodos de ingeniería al arte de la pintura. Pero es transdicisplinario porque se aborda una disciplina considerada como espiritual (Schopenhauer, 1987). El arte no es racional, ya que su percepción se realiza desde otro registro lingüístico. A diferencia de la lógica aristotélica, de que entre un ente A y un ente B no se da ningún tipo de relación, en esta tesis se sigue el principio lógico del tercero incluido: el término T está al mismo tiempo en A y en no A. Por ejemplo, el principio de la dualidad onda - partícula se encuentra en T.

Aunque el eje fundamental de la investigación trasdisciplinaria es el diálogo entre ciencia y espiritualidad, se requiere articular los conocimientos fragmentados en disciplinas

o campos del saber mediante un pensamiento complejo como menciona Morín (Morín y Le Moigne, 2006). Para el desarrollo de esta tesis se utilizan como base, la teoría cibernética y de la información.

En la Figura 1.6 se muestra una visión rica de la perspectiva transdisciplinaria para abordar esta investigación.

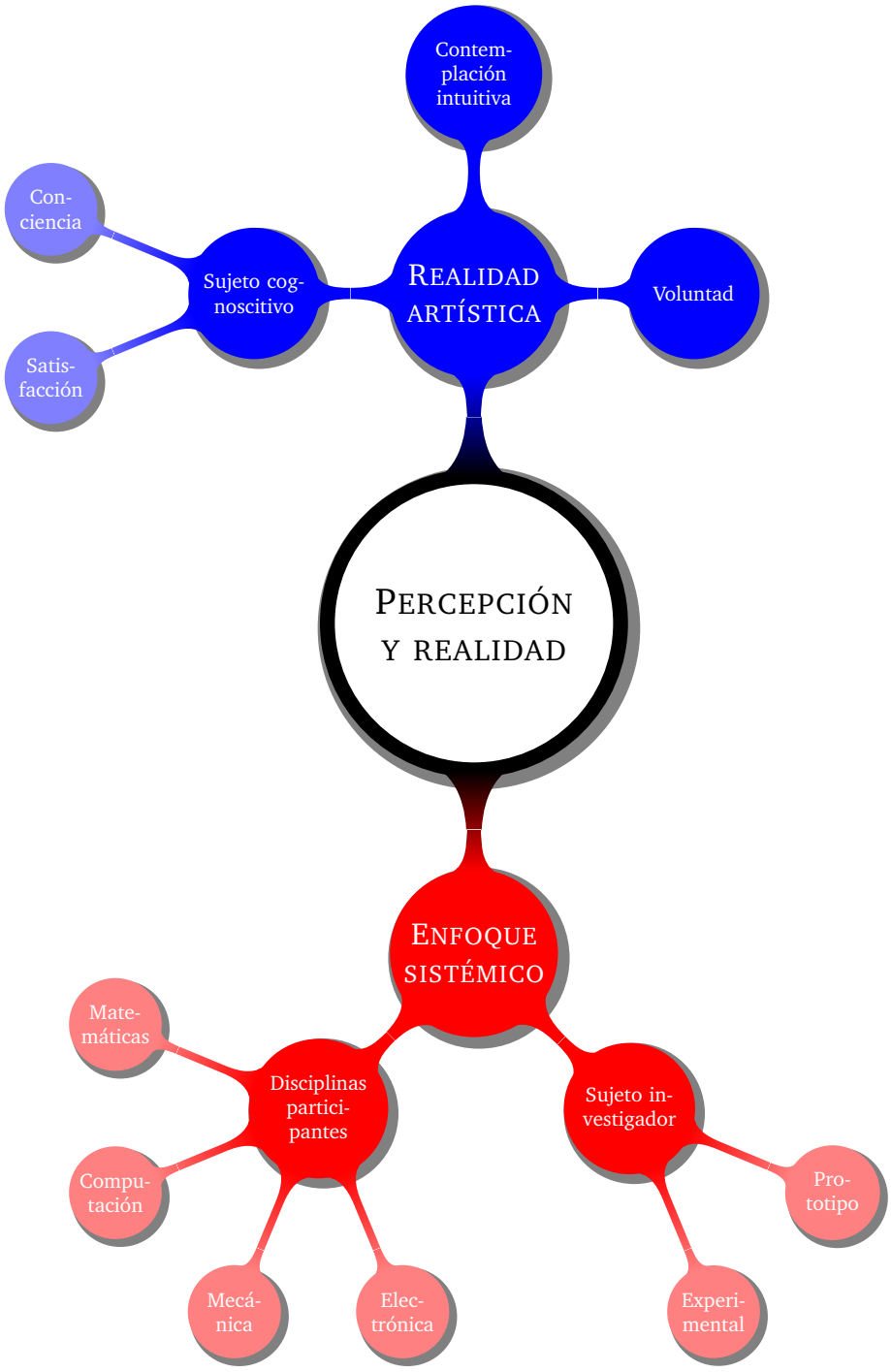


Figura 1.6: Visión rica de la perspectiva transdisciplinaria para abordar la investigación (Elaboración propia, 2019)

1.5. Contexto físico y económico

Las ubicaciones donde se fabrican artesanías de cerámica y alfarería, se pueden describir comenzando a nivel mundial y luego a nivel regional.

A nivel mundial las ubicaciones se obtuvieron a partir de los datos del Atlas de la Complejidad Económica de la Universidad de Harvard, los cuales se generaron ingresando a la página web: http://atlas.cid.harvard.edu/. y accediendo a la opción: EXPLORE, y escribiendo en el explorador de dicha opción la palabra clave: "Who exported ornamental ceramics (6913 HS4) in 2017". Dichos datos se observan en la Figura 1.7 donde se muestran los principales países exportadores de artículos cerámicos ornamentales durante 2017. De la figura mencionada se nota el predominio de China con un 62.3 % de 1,890 millones de dólares exportados durante 2017.

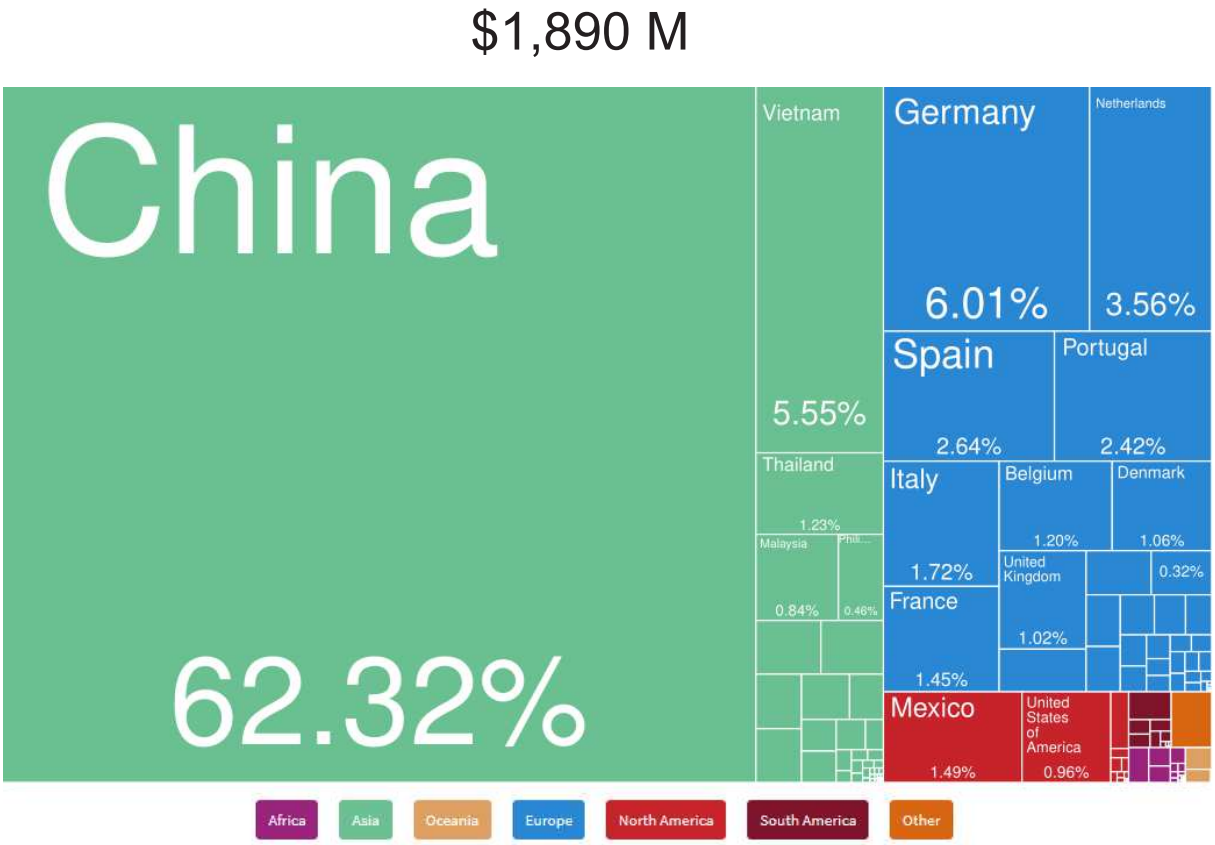


Figura 1.7: Países exportadores de cerámica ornamental en 2017 (Elaboración propia, 2019, imagen generada del Atlas of Economic Complexity de la Universidad de Harvard)

A nivel nacional, en México se realizan diversas actividades artesanales, las cuales se

muestran en la Figura 1.8, donde se observa un total de 37,406 personas ocupadas en el área de cerámica y alfarería durante 2016 según el INEGI.

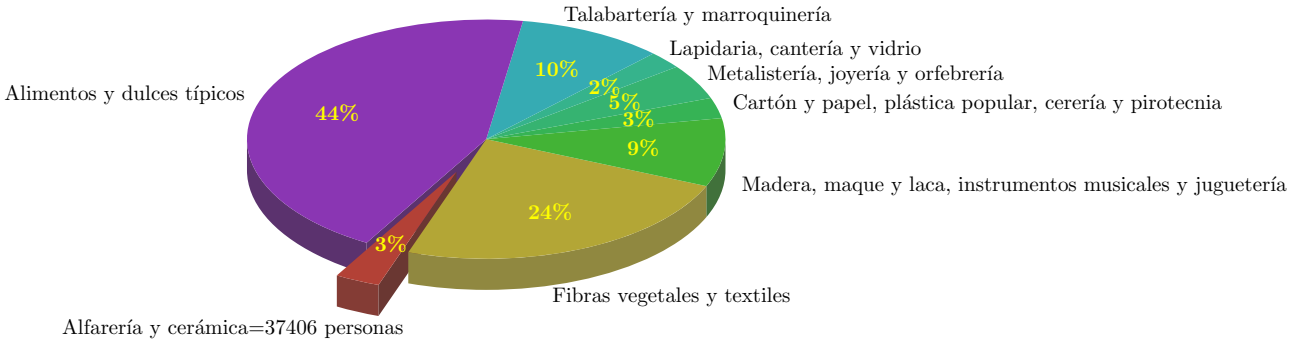


Figura 1.8: Total de personas ocupadas en área de artesanías: 1,222,480 (Elaboración propia 2019, a partir de datos del INEGI. Sistema de Cuentas Nacionales de México. Cuenta satélite de la cultura de México, 2016. Año Base 2013)

A nivel regional, la Tabla 1.2 resume las principales ubicaciones donde se practican actividades artesanales en México. También se muestran los productos que se elaboran en esos lugares (Gomez, 2010), (Ayala, 2015).

Tabla 1.2: Producción de cerámica ornamental por estados en México (Elaboración propia 2019, con datos de Gomez (2010))

Estado	**Comunidad**	**Producto**
Michoacán	Huacinto, Ichán, Patambán, San José de Gracia, Capula, Puruándiro, Cocucho, Ocumicho, Tzintzuntzan, Santa Fe de la Laguna	Cántaros bruñidos, ollas y cazuelas vidriadas, loza vidriada, bruñida y de alta temperatura, piñas vidriadas, vajillas de alta y baja temperatura, cántaros decorados con engove, figuras moldeadas a mano, loza decorada con peces, aves y soles, piezas para usos ceremoniales como sahumerios, candelabros.

Continúa en la página siguiente

Continuación de la tabla

Estado	**Comunidad**	**Producto**
Jalisco	Tonalá, Tlaquepaque	Alfarería vidriada y vajillas, decoración bruñida, de bandera: blanco sobre rojo, bruñida policroma con oro de hoja y de petatillo.
Estado de México	Metepec, Santa María Canchesdá	Alfarería de uso doméstico, árboles de la vida, cerámica tradicional y de alta temperatura.
Guanajuato	Dolores Hidalgo, Guanajuato	Cerámica de talavera, jarrones, aguamaniles, macetones, vajillas, candelabros, platones, fruteros.
Puebla	Amozoc de Mota, Izúcar de Matamoros, San Marcos, Acteopan, San Bartolo, Atlixco.	Loza colorada, talavera, policromado.
Oaxaca	Juchitán, Oaxaca, Jamiltepec, Santa María Atzompa, Coyotepec, San Marcos Tlapazola	Barro negro, loza verde, cántaros, tazones, jarrones.
Guerrero	Ameyaltepec, Tuliman, San Agustín, Chilapa, Acatlán , Ometepec	Ollas, cántaros, tinajas, cajetes, jarros, candeleros y esculturas humanas y de animales, personajes fantásticos y nacimientos de Navidad.

Continúa en la página siguiente

Continuación de la tabla

Estado	**Comunidad**	**Producto**
Chiapas	Amatenango del Valle, San Cristóbal de las Casas	Ollas, cántaros, tinajas, macetas, palomas, lámparas de palomas, platones de alcatraz, soles, tortugas, jaguares, incensarios, palomas, gallos, conejos, floreros.

Por último se considera la ubicación del trabajo de pintura en el proceso de producción de alfarería para el caso de una pequeña empresa de acuerdo con el Instituto Nacional de Economía Social como se observa en el diagrama de la Figura 1.9.

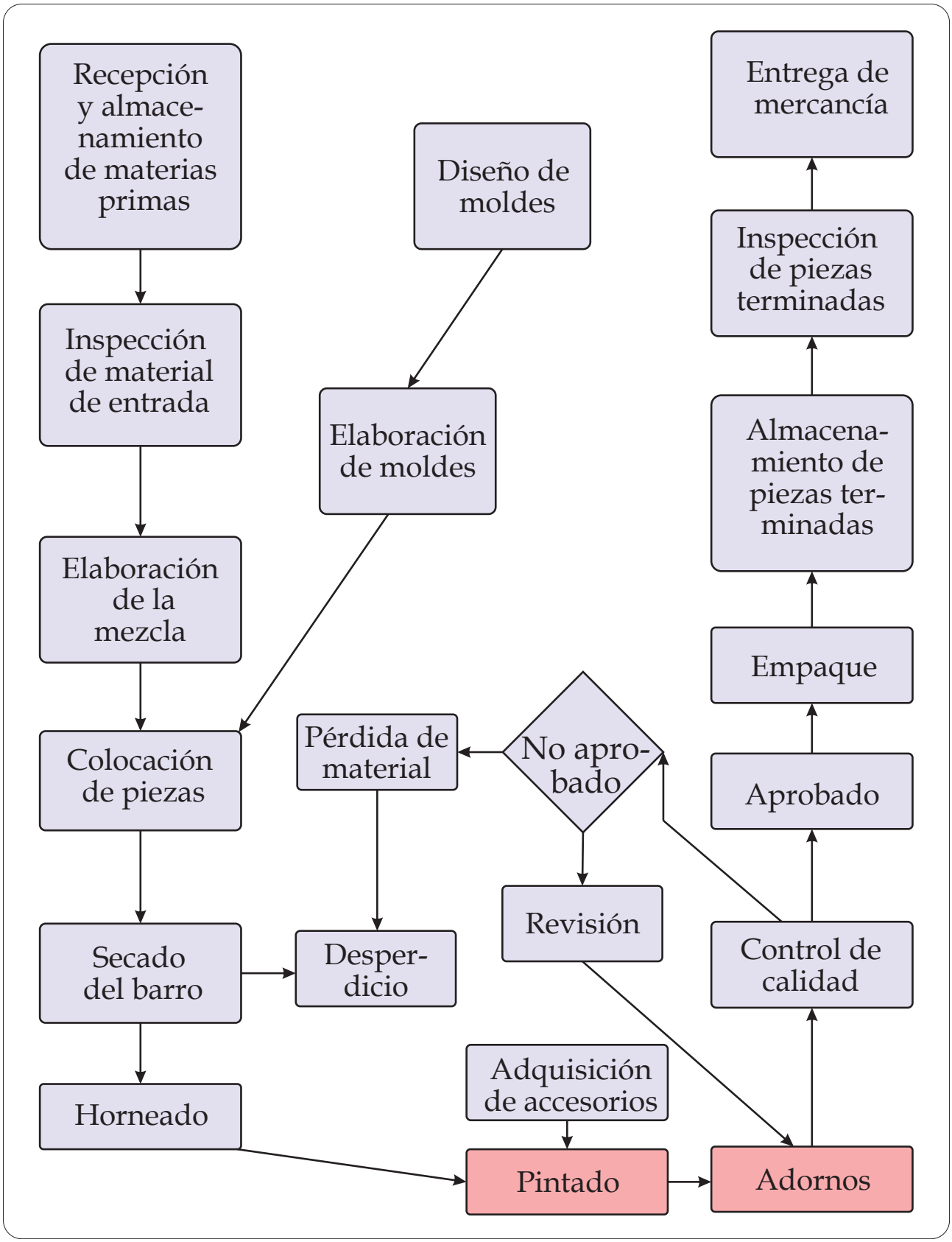


Figura 1.9: Proceso de producción de alfarería, (Elaboración propia, 2019. Consultado en 2019 en INAES, http://www.inaes.gob.mx/doctos/pdf/guia_empresarial/alfareria_y_ceramica.pdf)

1.6. Contexto cultural y social

Según el Fondo Nacional para el Fomento de las Artesanías (FONART) la artesanía es un objeto o producto de identidad cultural comunitaria, hecho por procesos manuales continuos, auxiliados por implementos rudimentarios y algunos de función mecánica que aligeran ciertas tareas.

La producción de artículos artesanales en México se remonta a la época prehispánica, sin embargo en la mayoría de las artesanías se mezcla la cultura prehispánica mexicana y la española, orientándose principalmente en la producción de artículos de uso doméstico como platos, jarros, ollas, cazuelas, entre otros artículos (Ayala, 2015).

En cuanto a la producción de cerámica, las normas aplicables en México son:NOM-199-SSA1-2000, NOM-004-SSA1-1993 y NOM-231-SSA1-2002 los cuales regulan la producción alfarera respecto a la utilización de materiales peligrosos o contaminantes y están disponibles en la página oficial de la COFEPRIS.

En México el consumo de artesanías se realiza principalmente para decoración, para el uso personal y para regalar, como se observa en la Tabla 1.3, la cual se basa en el estudio llevado a cabo por (Dávila, 2017).

Tabla 1.3: Perfil de consumidores de artesanías en México (Elaboración propia 2019, con datos de Dávila (2017))

Factores geográficos	**Factores demográficos**	**Factores psicográficos**	**Factores conductuales**
Mexicanos o extranjeros que viajan o residen en alguna parte de México	Hombres y mujeres de entre 26 y 55 años de edad, casados, con hijos. De nivel socioeconómico C+ o superior, con ingresos mensuales superiores a los $6,800 pesos, con escolaridad licenciatura o superior.	Gustan de los viajes y de las compras por distracción. Afines a la emotividad, le asignan un valor tangible a los recuerdos, gustan de adquirir nuevos conocimientos para compartirlos en sus grupos sociales.	Utilizan las artesanías como decoración, para su uso personal como textiles y joyería y para regalar. Identifican como puntos de venta los estados de la república y las zonas turísticas. Identifican pocos puntos de venta especializados. Eligen artesanías principalmente por cuestiones emotivas, prefiriendo comprar directamente con artesanos. Invierten un promedio de $400 a $1,199 pesos por compra, mismas que realizan entre 2 y 5 veces al año.

1.7. Contexto histórico

Hay muchos ejemplos de sistemas computarizados intentando plasmar artísticamente la realidad. Esto es el objetivo principal de los gráficos por computadora en el subcampo conocido como renderizado no fotorrealístico (NPR), el cual apareció en los años 90 (P. A. Tresset y Leymarie, 2013). De los trabajos del NPR nació el interés por transferir los métodos de procesamiento de imagen a medios artísticos mediante robots. El problema que se busca resolver en esta línea de investigación es entender mejor y tratar de emular la actividad artística humana.

La manera en que se ha solucionado este problema a lo largo de la historia se describe

en la columna Método de la Tabla 1.1, y se resume en la Figura 1.10.

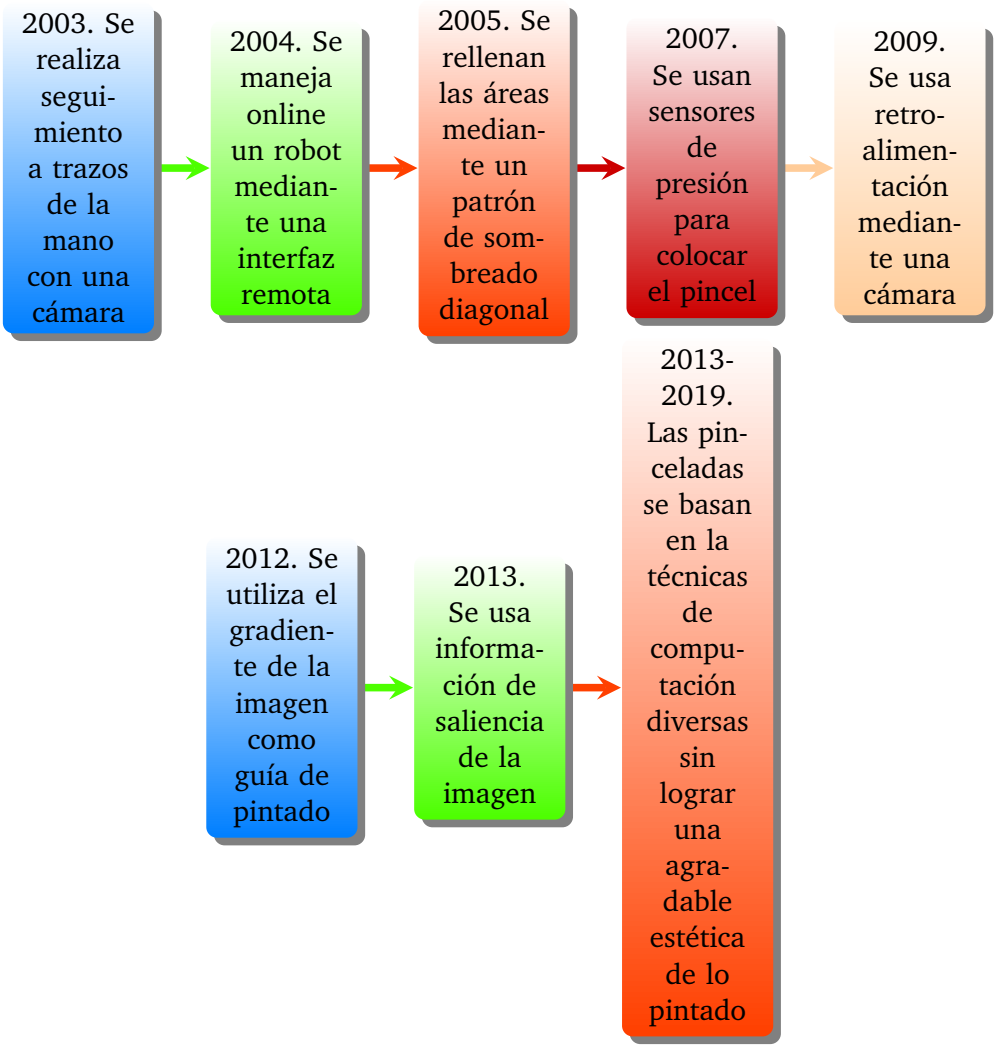


Figura 1.10: Maneras de solucionar el problema de la pintura robótica en la historia (Elaboración propia, 2019)

El problema a lo largo de la historia también se puede abordar con imágenes representativas como se observa en la Figura 1.11, donde se muestran los trabajos reportados en la literatura.

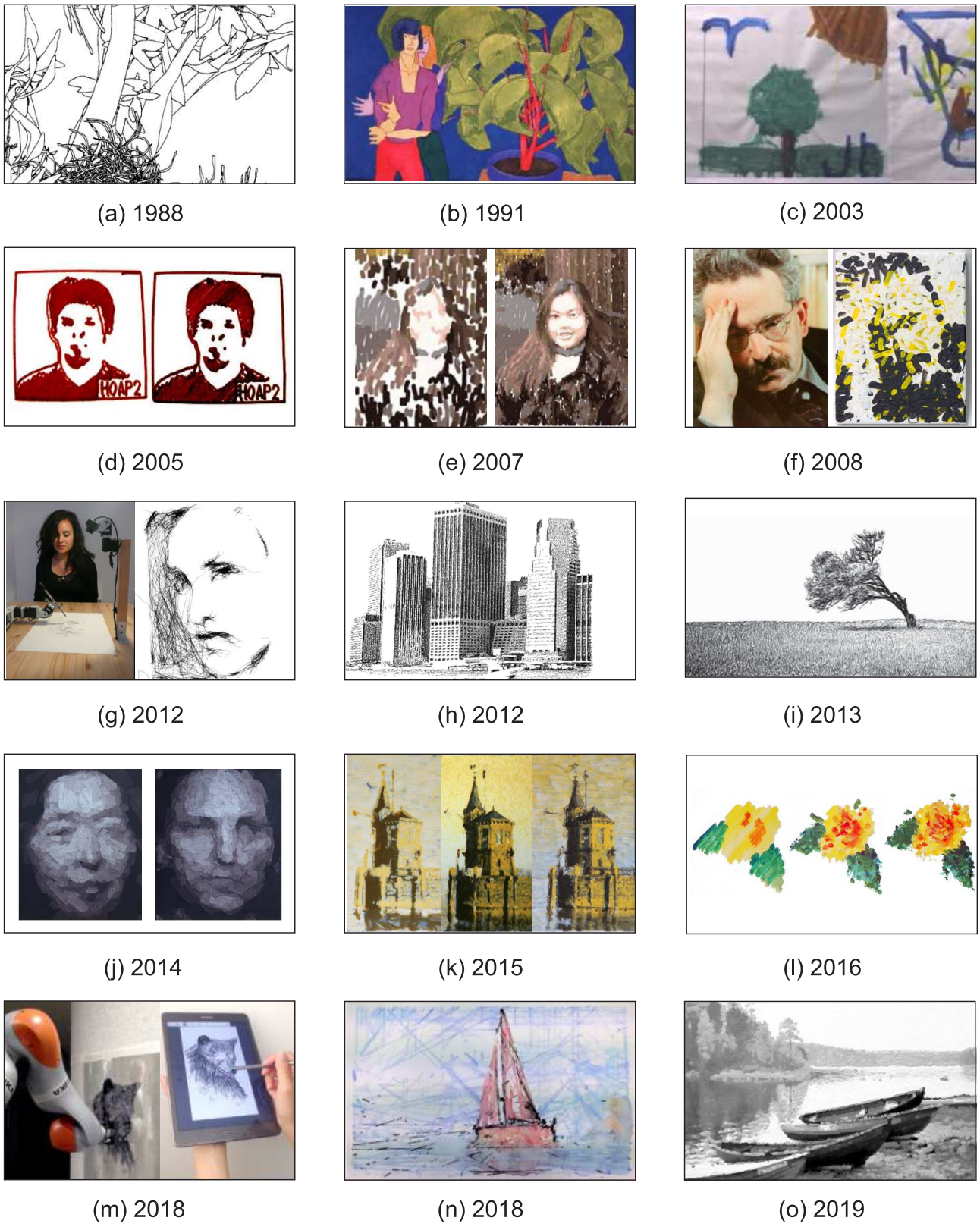


Figura 1.11: Obras representativas de pintura robótica a lo largo de la historia reportadas por: (a)Cohen (1988), (b)Cohen (1995), (c)Stein (2003), (d)Calinon *et al.* (2005), (e)Ruchanurucks *et al.* (2007), (f) Aguilar y Lipson (2008), (g)P. A. Tresset y Leymarie (2012), (h)Deussen *et al.* (2012), (i)Lindemeier *et al.* (2013), (j)P. Tresset y Deussen (2014), (k)Lindemeier *et al.* (2015), (l)Luo *et al.* (2016), (m)Song *et al.* (2018), (n)Scalera *et al.* (2018b), (o) Karimov *et al.* (2019) (Elaboración propia, 2019)

También hay trabajos que no están reportados en la literatura pero que fueron premiados en la competencia de arte robot organizado por RobotArt en 2016, 2017 y 2018, los cuales se pueden consultar en la página web de RobotArt y se muestran algunos representativos en la Figura 1.12.

Figura 1.12: Obras representativas de pintura robótica a lo largo de la historia no reportadas, premiadas por RobotArt en 2016: (a) Obra de TAIDA, (b) Obra de CloudPainter, (c) Obra de RHIT. En 2017: (d) Obra de PIX18/Creative Machines Lab, (e) Obra de CloudPainter, (f) Obra de Christian H. Seidler. En 2018: (g) Obra de CloudPainter, (h) Obra de Custom Autonomous Robotic Painter, (i) Obra de Joanne Hastie (Elaboración propia, 2019)

1.8. Justificación

La actividad artesanal en su rama de cerámica o alfarería fue durante 2016 fuente de ingreso para 37,406 mexicanos que desarrollaron este trabajo como una actividad económica, la cual participó en el Valor Agregado Bruto con 1240 millones de pesos, de acuerdo con la Cuenta satélite de la cultura de México, 2016, año base 2013, del INEGI.

Sin embargo la competencia con productos realizados de forma masiva con alta tecnología ha llevado a un rezago de la práctica artesanal (Díaz-Bautista, 2006), (Hernández-Ramírez, Pineda-Domínguez, y Andrade-Vallejo, 2011). Este trabajo busca ayudar a la solución de este problema, proponiendo un robot que apoye la producción artesanal en sus procesos repetitivos, buscando innovar en los procesos de producción sin perder la esencia artesanal.

Por otro lado, la aplicación de pintura para propósitos artísticos ha sido investigado según lo revisado en la bibliografía, sin embargo no se ha aplicado en artesanía. Además, actualmente los trazos artísticos en dichas investigaciones aún parecen mecanizados, por lo que en esta tesis se pretende que el objeto artístico producido tenga mayor calidad artística que dichos trabajos.

1.9. Objetivo principal

Contribuir al desarrollo tecnológico de la producción artesanal mediante el uso de un robot cartesiano y de un modelo computacional que simule el proceso de pintado

1.10. Objetivos particulares

1. Desarrollar la interfaz gráfica y el robot cartesiano
2. Suavizar imagen opcionalmente
3. Segmentar regiones automáticamente
4. Realizar segmentación interactiva
5. Generar campo vectorial automático
6. Crear campo vectorial interactivo
7. Producir trazos a partir del campo vectorial

8. Obtener bordes
9. Aplicar pintura robótica

1.11. Tabla de congruencia

En la Tabla 1.4 se resumen el problema de investigación, la justificación y los objetivos de este proyecto.

Tabla 1.4: Tabla de congruencia, (Elaboración propia, 2019)

PROBLEMA DE INVESTIGACIÓN Falta de desarrollo tecnológico en la producción de artesanías cerámicas la cual se busca disminuir usando un robot cartesiano y algoritmos computacionales		
Justificación La competencia con productos realizados de forma masiva con alta tecnología ha llevado a un rezago de la práctica artesanal en México (Díaz-Bautista, 2006)		
Objetivo general Contribuir al desarrollo tecnológico de la producción artesanal mediante el uso de un robot cartesiano y de un modelo computacional que simule el proceso de pintado		
Objetivo particular 1 Desarrollar la interfaz gráfica y el robot cartesiano *Preguntas de investigación* ¿Cuáles son sus características? ¿su costo? ¿su rendimiento?	**Objetivo particular 2** Suavizar imagen opcionalmente *Preguntas de investigación* ¿Qué técnicas de procesamiento de imagen se usarán?	**Objetivo particular 3** Segmentar regiones automáticamente *Preguntas de investigación* ¿Qué parámetros de la imagen serán la entrada del algoritmo?
Objetivo particular 4 Realizar segmentación interactiva *Preguntas de investigación* ¿Cuántos puntos se seleccionarán en el contorno? ¿Cuál será el umbral para detección de esquina?	**Objetivo particular 5** Generar campo vectorial automático *Preguntas de investigación* ¿Cuántas iteraciones se realizarán? ¿Cuál será el tamaño de la ventana?	**Objetivo particular 6** Crear campo vectorial interactivo *Preguntas de investigación* ¿Qué método de interpolación se utilizará?
Objetivo particular 7 Producir trazos a partir del campo vectorial *Preguntas de investigación* ¿Qué método se usará para generar los trazos?	**Objetivo particular 8** Obtener bordes *Preguntas de investigación* ¿Qué técnica de detección de bordes se utilizará?	**Objetivo particular 9** Aplicar pintura robótica *Preguntas de investigación* ¿Cómo se aplicará la pintura?¿Qué pasos se seguirán?

Capítulo 2

Marco teórico y metodológico

2.1. Marco teórico

2.1.1. Introducción

A continuación se exponen los fundamentos teóricos necesarios para efectuar las operaciones mostradas en el diagrama de flujo de la Figura 3.6.

Una imagen se puede definir como una función bidimensional $f(x, y)$, donde x y y son las coordenadas espaciales y el valor de f en un punto se llama intensidad (Gonzalez *et al.*, 2009). Cuando la imagen $f(x, y)$ se muestrea de tal forma que tenga M filas y N columnas, se dice que la imagen es de tamaño M x N y se representa en forma de matriz como sigue.

$$f(x) = \begin{pmatrix} f(0,0) & f(0,1) & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,N-1) \\ \vdots & \vdots & \vdots \\ f(M-1,0) & f(M-1,1) & f(M-1,N-1) \end{pmatrix}$$

De esta manera, el tamaño de la imagen define su resolución espacial y el número de posibles valores de intensidad de un pixel define su cuantización o resolución en bits. En este trabajo se utilizaron imágenes de diferentes tamaños con una resolución de 8 bits para las imágenes en escala de gris y 24 bits para las imágenes en color.

2.1.2. Preparación de la imagen aplicando difusión anisotrópica

La preparación de la imagen es un paso opcional previo a la segmentación automática, en esta sección se describe el método de difusión anisotrópica el cual es una técnica iterativa de suavizado direccional que preserva los bordes y se puede describir de la siguiente manera.

Considérese un recipiente lleno de agua en equilibrio hidrostático donde cada molécula de agua está en movimiento aleatorio debido a continuas colisiones con las moléculas vecinas. Si se deposita una gota de tinta, la cual dependiendo de la sustancia y de la temperatura, invadirá toda la región que inicialmente contenía agua pura, hasta extenderse por todo el volumen de manera homogénea.

Este proceso microscópico es lo que se llama difusión molecular. Si en un punto dado se considera un elemento de superficie ΔS con normal n, en un instante dado habrá moléculas de tinta que atraviesen ese elemento de área en dos sentidos. Si la concentración de tinta es distinta a ambos lados del elemento de superficie, habrá un flujo neto de moléculas de tinta a través de ΔS desde el lado en donde la concentración es mayor hacia el lado en que es menor. La relación cuantitativa entre la cantidad de flujo neto y las variaciones de concentración de un punto a otro viene expresada por la ley de Fick (Alonso y Finn, 1967):

$$\mathbf{j} = -c\nabla I \tag{2.1}$$

donde $I(\mathbf{r}, t)$ representa la concentración de moléculas, $\mathbf{j}(\mathbf{r}, t)$ el flujo neto en ese punto, y c el coeficiente de difusión que depende del tipo de moléculas, $\mathbf{r}$ es la coordenada espacial, t es el tiempo, y ∇I es el gradiente de la concentración.

La distribución en el espacio y en el tiempo de las moléculas está dada por la ecuación 2.2 y por la restricción de que el número de moléculas se mantenga constante, lo cual se expresa mediante la ecuación de continuidad en forma local como,

$$\frac{\partial I}{\partial t} + \nabla \cdot \mathbf{j} = 0 \tag{2.2}$$

donde $\nabla \cdot \mathbf{j}$ es la divergencia del flujo. Si se combina la ecuación de continuidad con la

ley de Fick, se obtiene la relación

$$\frac{\partial I}{\partial t} = \nabla \cdot (c \nabla I) \tag{2.3}$$

la cual es la ecuación de difusión.

Koenderink (1984) utilizó el modelo de la difusión lineal en imágenes, considerando el nivel de gris I como la concentración de la sustancia. Siguiendo dicha analogía, el problema queda planteado de la siguiente manera,

$$\begin{cases} \frac{\partial I}{\partial t} = \Delta I \\ I(x, y, 0) = I_0(x, y), \end{cases} \tag{2.4}$$

donde $\Delta I = I_{xx} + I_{yy}$ es el Laplaciano de I. Siendo la condición inicial la imagen I_0 original.

La ecuación 2.4 se puede resolver numéricamente usando diferencias finitas como sigue,

$$\frac{I_{i,j}^{n+1} - I_{i,j}^{n}}{\Delta t} \approx \frac{I_{i+1,j}^{n} - 2I_{i,j}^{n} + I_{i-1,j}^{n}}{(\Delta x)^2} + \frac{I_{i,j+1}^{n} - 2I_{i,j}^{n} + I_{i,j-1}^{n}}{(\Delta y)^2} \tag{2.5}$$

donde se toma por lo general $\Delta x = \Delta y = 1$ pixel, por tanto la solución en el tiempo t es la iteración n multiplicada por Δt:

$$I_{i,j}^{n+1} \approx I_{i,j}^{n} + \Delta t * (-4I_{i,j}^{n} + I_{i+1,j}^{n} + I_{i-1,j}^{n} + I_{i,j+1}^{n} + I_{i,j-1}^{n}) \tag{2.6}$$

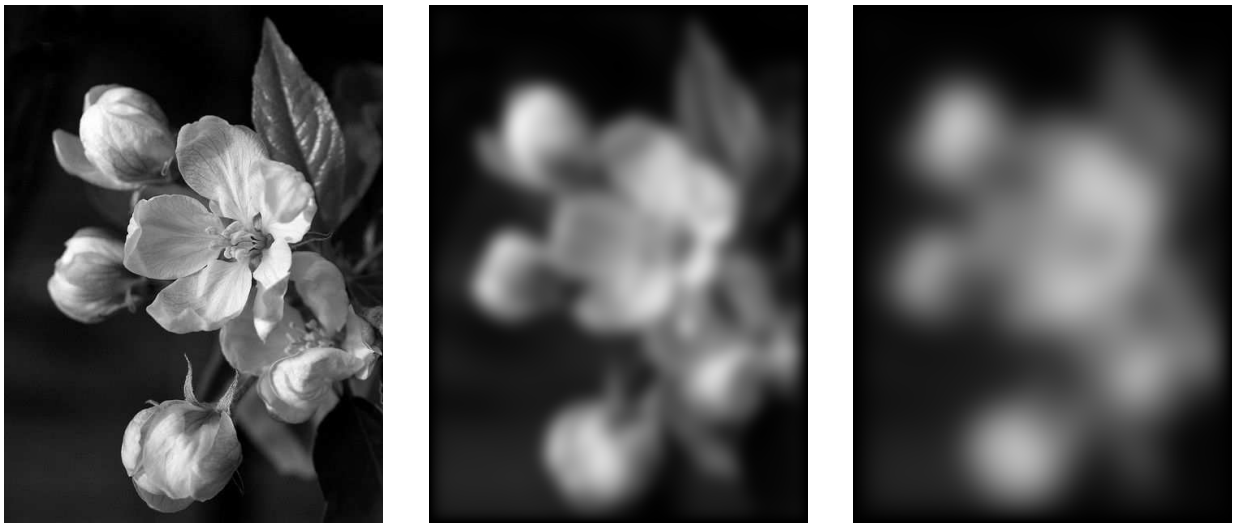
Con la condición de estabilidad que $\Delta t \leq 1/4$.

Así, el Código Matlab siguiente simula la difusión isótropa mediante la ecuación 2.6 usando un $\Delta t = 0.25$ para un $t = 50$:

```
I0 = double(rgb2gray(imread('imagen.jpg')));
[p,q]=size(I0);
I=zeros(p,q);
dt=0.25;
T=50;
for t=0:dt:T
    for i=2:p-1
        for j=2:q-1
         Ixx=I0(i+1,j)-2*I0(i,j)+I0(i-1,j);
         Iyy=I0(i,j+1)-2*I0(i,j)+I0(i,j-1);
         I(i,j)=I0(i,j)+dt*(Ixx+Iyy);
        end
```

```
    end
    I0=I;
end
imshow(I0,[]);
```

En la Figura 2.1(b) se muestra la difusión isótropa de la imagen de la Figura 2.1(a) para $t = 50$. Adicionalmente, se muestra la difusión isótropa para $t = 200$ en la Figura 2.1(c).



(a) Imagen I_0 (b) Imagen I para t=50 (c) Imagen I para t=200

Figura 2.1: Aplicación de la ecuación de difusión isótropa (Elaboración propia, 2019)

Otra forma de resolver la ecuación de difusión es aplicando una serie de filtros gaussianos con varianza creciente (Witkin, 1983), lo cual es equivalente a encontrar $I(x, y, t), t > 0$, convolucionando I_0 con un filtro gaussiano cuya desviación estándar $\sigma = \sqrt{2t}$:

$$\begin{aligned} I(x,y,t) &= I_0(x,y) * \frac{1}{\sqrt{4\pi t}} e^{-(\frac{x^2}{4t}+\frac{y^2}{4t})} \\ &= \frac{1}{\sqrt{4\pi t}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} exp(-(\frac{(x-\xi)^2}{4t} + \frac{(y-\eta)^2}{4t}))I_0(\xi,\eta)\, d\xi\, d\eta \end{aligned} \tag{2.7}$$

donde " $*$ " denota la convolución con respecto al pixel (x, y).

El código Matlab siguiente simula la ecuación 2.7 para un tiempo $t = 50$, es decir, $\sigma = \sqrt{2t} = 10$, obteniéndose la misma imagen de difusión que la Figura 2.1(b):

```
sigma=10;
tamano=2*ceil(2*sigma)+1;
I = rgb2gray(imread('imagen.jpg'));
h = fspecial('gaussian',[tamano tamano], sigma);
I2 = imfilter(I,h);
imshow(I2);
```

Aunque el resultado es semejante, el tiempo para simular la difusión isótropa para el caso iterativo cuando $\Delta t = 0.25$ es mayor casi n veces que para el filtrado gaussiano.

La difusión isótropa produce por tanto, un efecto de desenfoque o borrosidad en la imagen. Este método que se usa para disminuir ruido, no se usará en esta tesis, sin embargo se usará la modificación descrita a continuación.

El enfoque novedoso del modelo de la difusión se dio con el trabajo de Perona y Malik (1990) quienes modificaron el filtrado isótropo logrando acentuar el proceso de difusión en las zonas homogéneas mientras se preservaban los bordes, usando para esto un coeficiente de conducción variable $c(x, y, t)$, como se plantea en el sistema siguiente:

$$\begin{cases} \frac{\partial I}{\partial t} = \nabla \cdot (c(x, y, t)\nabla I) \\ I(x, y, 0) = I_0(x, y), \end{cases} \tag{2.8}$$

donde $\nabla\cdot$ representa el operador de divergencia y ∇I el gradiente de la imagen.

La ecuación 2.8 se puede discretizar como sigue:

$$I_{i,j}^{n+1} = I_{i,j}^{n} + \frac{1}{4}[c_N \nabla_N I + c_S \nabla_S I + c_E \nabla_E I + c_W \nabla_W I]_{i,j}^{n} \tag{2.9}$$

donde

$$\begin{aligned} \nabla_N I_{i,j} &\equiv I_{i-1,j} - I_{i,j} \\ \nabla_S I_{i,j} &\equiv I_{i+1,j} - I_{i,j} \\ \nabla_E I_{i,j} &\equiv I_{i,j+1} - I_{i,j} \\ \nabla_W I_{i,j} &\equiv I_{i,j-1} - I_{i,j} \end{aligned} \tag{2.10}$$

representan los gradientes de los cuatro pixeles vecinos al norte, sur, este y oeste, respectivamente, siendo c_N, c_S, c_E, c_W los coeficientes de difusión evaluados en el gradiente

correspondiente mediante la función siguiente,

$$g(\nabla I) = \frac{1}{1 + \left(\frac{\|\nabla I\|}{K}\right)^2} \tag{2.11}$$

donde K controla la sensibilidad a los bordes. La ecuación 2.12 muestra los coeficientes aproximando la norma euclidiana del gradiente por su valor absoluto (Perona y Malik, 1990):

$$\begin{aligned} c_N &= \frac{1}{1 + \left(\frac{|\nabla_N I_{i,j}|}{K}\right)^2} \\ c_S &= \frac{1}{1 + \left(\frac{|\nabla_S I_{i,j}|}{K}\right)^2} \\ c_E &= \frac{1}{1 + \left(\frac{|\nabla_E I_{i,j}|}{K}\right)^2} \\ c_W &= \frac{1}{1 + \left(\frac{|\nabla_W I_{i,j}|}{K}\right)^2} \end{aligned} \tag{2.12}$$

Las líneas 1286-1308 del Código 1 Matlab en el Anexo 1, simulan la ecuación 2.9 utilizando un valor de K=3. En la Figura 2.2 se muestra el resultado de 100 y 500 iteraciones donde se observa el suavizado en las zonas más homogéneas mientras se preservan los bordes. Este método de suavizado se usará para facilitar la segmentación de regiones.

(a) Imagen I_0 (b) Imagen I para t=100 (c) Imagen I para t=500

Figura 2.2: Aplicación de la ecuación de difusión anisotrópica (Elaboración propia, 2019)

2.1.3. Segmentación automática usando K-medias

Para obtener regiones de trabajo se propone el uso de dos técnicas; una automática para imágenes simples y otra semiautomática para imágenes complejas. En el caso de la segmentación automática se propone el uso de *k-medias* (Macqueen, 1967), uno de los algoritmos de agrupamiento de datos más simple y más usado. Para el caso de la segmentación semiautomática o interactiva se propone el uso de curvas de Bézier para aproximar el contorno de las regiones a segmentar. A continuación se describen los fundamentos teóricos utilizados en ambos casos.

El algoritmo *k-medias* usa la distancia euclidiana para comparar objetos y utiliza promedios para estimar los centroides de las clases. Consiste básicamente en los siguientes pasos:

1. Selecciona aleatoriamente k centroides iniciales.

2. Asigna cada objeto x_i a su centroide más cercano.

3. Recalcula los nuevos centroides, regresa al paso 2, hasta que el algoritmo converge.

El objetivo del algoritmo es minimizar la función objetivo:

$$V = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in X} \|\mathbf{x}_j - \mathbf{v}_i\|^2 \tag{2.13}$$

donde $X = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbb{R}^p, i = 1, 2, ..., n\}$ es el conjunto de datos. $\|\mathbf{x}_j - \mathbf{v}_i\|^2$ es la distancia cuadrática entre el j-ésimo elemento $\mathbf{x}_j$ y el i-ésimo centroide $\mathbf{v}_i$ y $k \leq n$ es el número de grupos.

La ecuación 2.13 se puede simular en Matlab 2018b utilizando la función *imsegkmeans* del toolbox de procesamiento de imagen. De este modo, la instrucción: *etiquetas = imsegkmeans (I, 4, 'NumAttempts', 3);* del código Matlab siguiente, segmenta la imagen *I* en 4 grupos usando el algoritmo *k-medias* y devuelve la imagen etiquetada *etiquetas*:

```
% Lee la imagen
I=(imread('Imagen.jpg'));
% K-medias
grupos = 4;
% Repite agrupamiento 3 veces para evitar minimo local
etiquetas = imsegkmeans(I,grupos,'NumAttempts',3);
etiquetasRGB = label2rgb(etiquetas);
imshow(etiquetasRGB);
```

2.1.4. Segmentación interactiva

Para abordar los aspectos teóricos de la segmentación interactiva se incluye el diagrama de flujo mostrado en la Figura 2.3 el cual se detalla en Aplicación de la metodología. En dicho diagrama se adaptó la metodología que Sarfraz (2008), Capítulo 13, utilizó para representar el contorno de caracteres.

Los temas necesarios para realizar la segmentación interactiva siguiendo la Figura 2.3 son los siguientes. Para realizar el paso 2 se requiere el algoritmo de detección de esquinas, para el paso 3 las curvas Bézier usadas como aproximación, y para el paso 4, el algoritmo de Bresenham que segmenta la región de interés. Estos tres tópicos se describen a continuación.

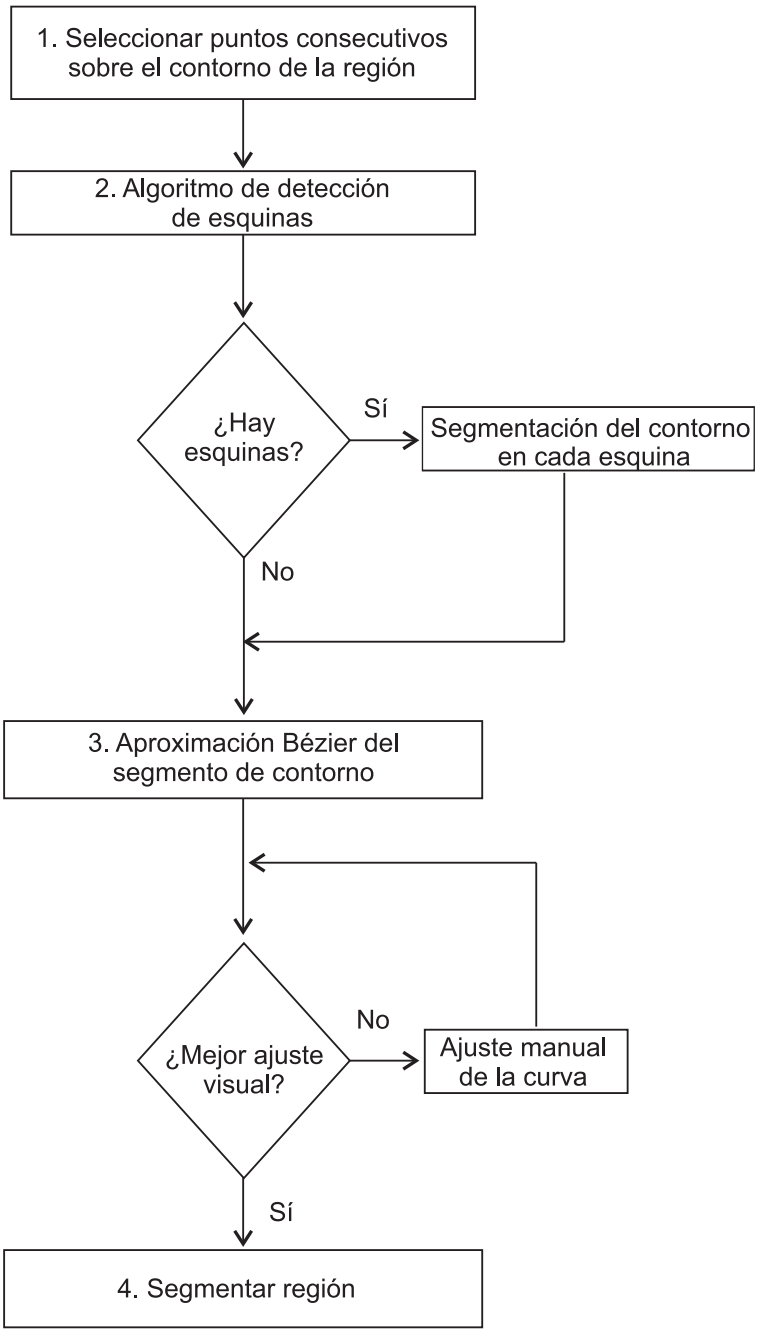


Figura 2.3: Procedimiento de la segmentación interactiva (Elaboración propia, 2019)

I. Detección de esquinas para la segmentación del contorno

Las esquinas son características importantes en imágenes digitales ya que con ellas se pueden representar y analizar las formas y se conservan bajo traslación, rotación y cambio de escala. La detección de esquinas se usa en varias aplicaciones, por ejemplo, en el entendimiento de la percepción humana de objetos (Asada y Brady, 1986), (Attneave, 1954), en

la representación de imágenes binarias (Cabrelli y Molter, 1990), en visión estéreo (Deriche y Faugeras, 1990), en seguimiento del movimiento (H. Wang y Brady, 1995), en cotejo de imágenes (Vincent y Laganière, 2005), entre otras.

Los algoritmos de detección de esquinas se dividen en dos grandes grupos (J. Wang y Zhang, 2018): métodos basados en la intensidad y métodos basados en el contorno. En esta tesis se utiliza el método de Rosenfeld y Weszka (1975), uno de los métodos más simples basado en el contorno. En la revisión bibliográfica de Abe, Morii, Nishida, y Kadonaga (1993) se pueden consultar otras técnicas de detección de esquinas para contorno.

El método de Rosenfeld aplicado a esta tesis es como sigue.

Sea $P = p_1, ..., p_l$ el conjunto de puntos muestras del contorno sobre los que se da clic secuencialmente en el objeto a segmentar, donde $p_i = (x_i, y_i)$ es el punto para $i = 1, ..., l$. Siendo l el número de puntos del contorno elegidos a criterio del usuario, donde dicho número muestree significativamente el contorno, y x_i, y_i las coordenadas de dicho punto en la imagen. Lo significativo quiere decir que se debe dar clic sobre las esquinas en la imagen de entrada para que el algoritmo las incluya.

Para la detección del ángulo se calculan los valores de la curvatura K para cada i mediante la fórmula $K = c_{ik}(p_i)$, para la longitud k de la cuerda, donde

$$c_{ik} = \frac{a_{ik}.b_{ik}}{|a_{ik}||b_{ik}|} \tag{2.14}$$

siendo c_{ik} el coseno del ángulo entre los vectores a_{ik} y b_{ik} en p_i y $a_{ik} = (x_i - x_{i+k}, y_i - y_{i+k})$, $b_{ik} = (x_i - x_{i-k}, y_i - y_{i-k})$. De este modo c_{ik} es mayor cuando la curva tiene un ángulo muy pequeño y menor cuando el ángulo es grande, es decir $-1 \leq c_{ik} \leq 1$, y la cuerda $k > 1$ es la distancia desde p_i en pixeles siguiendo el contorno P muestreado mediante clics. Los valores de c_{it} se promedian para cada punto i, siendo $t = 1, .., k$.

II. Aproximación Bézier de cada segmento del contorno

Una vez detectadas las esquinas en el contorno muestreado, se procede a dividir el contorno en segmentos desde cada esquina. Cada segmento se aproxima manualmente usando curvas Bézier de grado $1 < n < 20$ hasta lograr el efecto estético deseado del contorno.

Las curvas Bézier forman parte de la familia de $splines$, la cual incluye entre otros,

los splines cúbicos, B-splines, Beta-splines, splines de Hermite, y splines Bézier (Biswas y Lovell, 2007), (Farin, 1996).

En este trabajo se decidió utilizar las curvas Bézier debido al control global de su forma, lo cual genera trazos estéticos más libres al momento de pintar.

Las curvas de Bézier de grado n son curvas paramétricas y se pueden definir como

$$c(t) = \sum_{j=0}^{n} B_j^n(t) p_j \tag{2.15}$$

donde $c(t)$ es un punto sobre la curva para un valor particular del parámetro $t \in [0, 1]$, p_j son los puntos de control que controlan la forma de la curva Bézier y $B_j^n(t)$ son los polinomios de Bernstein de grado n. Se puede afirmar que un punto $c(t)$ es la suma ponderada de los puntos de control, donde los pesos son los polinomios de Bernstein evaluados en t. Los polinomios de Bernstein de grado n están dados por

$$B_j^n(t) := \binom{n}{j} t^j (1-t)^{n-j} \tag{2.16}$$

siendo $\binom{n}{j} = \frac{n!}{j!(n-j)!}$, donde por convención, $0! = 1$. Como ejemplo en la Figura 2.4 se muestran los polinomios de Bernstein de grado 3.

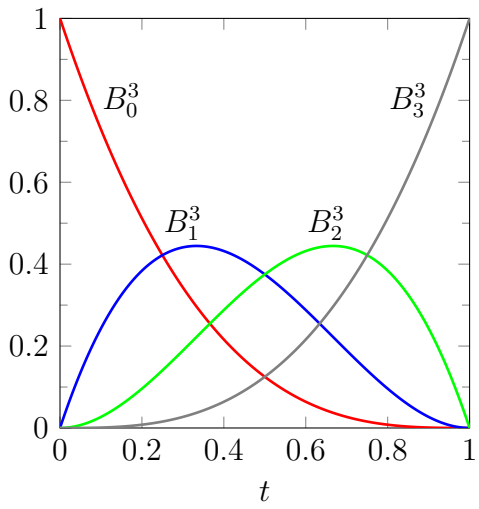


Figura 2.4: Polinomios de Bernstein de grado 3 (Elaboración propia, 2019)

Supóngase que ahora se tienen el conjunto de puntos $q_i = (x_i, y_i),\ i = 0, ..., m$ obtenidos al muestrear manualmente el segmento de contorno y que se desea obtener la curva Bézier de grado n dado que mejor se aproxime al conjunto de puntos en el sentido de los mínimos

cuadrados. Para lograrlo se tendrán que calcular los puntos de control $p_j,\ j = 0, ..., n$ de la curva, minimizando la suma de los cuadrados de los residuos:

$$E = \sum_{i=0}^{m} \left(\sum_{j=0}^{n} B_j^n(t_i) p_j - q_i \right)^2 \tag{2.17}$$

donde la elección de los nodos $\{t_0, ..., t_m\}$, los cuales establecen el comportamiento de la curva resultante al aproximar $\{q_0, ..., q_m\}$, se realiza considerando una aceleración centrípeta suave (Lee, 1989), es decir basándose en la longitud de la cuerda $\delta t_i = k\sqrt{\|\delta q_i\|}$, donde $\|\delta q_i\|$ es la distancia euclidiana entre dos puntos consecutivos suponiendo que la curva está parametrizada aproximadamente por su longitud de arco, siendo k una constante de escala. Esta consideración de aceleración se realiza para evitar variaciones bruscas de las fuerzas de inercia en la curva resultante y estimando que este modelo simple satisface los requisitos de esta tesis.

Así, los coeficientes p_j que minimizan la Ecuación 2.17 se pueden encontrar derivando E respecto a p_j. De este modo, la solución en forma de matriz es (Stephen Boyd, 2018) pp.228:

$$\nabla E = 2B^T(Bp - q) \tag{2.18}$$

la cual si se iguala a 0 como condición para el mínimo de E y se reescribe quedando como

$$B^T B p = B^T q \tag{2.19}$$

llamada las *ecuaciones normales*, cuyos elementos son:

$$\begin{bmatrix} B_0^n(t_0) & B_0^n(t_1) & \cdots & B_0^n(t_m) \\ B_1^n(t_0) & B_1^n(t_1) & \cdots & B_1^n(t_m) \\ \vdots & \vdots & \ddots & \vdots \\ B_n^n(t_0) & B_n^n(t_1) & \cdots & B_n^n(t_m) \end{bmatrix} \begin{bmatrix} B_0^n(t_0) & B_1^n(t_0) & \cdots & B_n^n(t_0) \\ B_0^n(t_1) & B_1^n(t_1) & \cdots & B_n^n(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ B_0^n(t_m) & B_1^n(t_m) & \cdots & B_n^n(t_m) \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \tag{2.20}$$

$$= \begin{bmatrix} B_0^n(t_0) & B_0^n(t_1) & \cdots & B_0^n(t_m) \\ B_1^n(t_0) & B_1^n(t_1) & \cdots & B_1^n(t_m) \\ \vdots & \vdots & \ddots & \vdots \\ B_n^n(t_0) & B_n^n(t_1) & \cdots & B_n^n(t_m) \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_m \end{bmatrix}$$

Si las columnas de B son linealmente independientes entonces la matriz B^TB es invertible y los puntos de control p_j se pueden encontrar con la ecuación

$$p = (B^TB)^{-1}B^Tq \tag{2.21}$$

Estos n puntos de control encontrados se sustituyen en la ecuación 2.15 utilizando un nuevo muestreo de t y de este modo se aproximan los puntos manuales del contorno a la curva Bézier de grado n deseado.

III. Segmentación de la región

Teniendo los mejores ajustes del contorno se procede a segmentar la región que encierra dicho contorno como se muestra en la Figura 2.5.

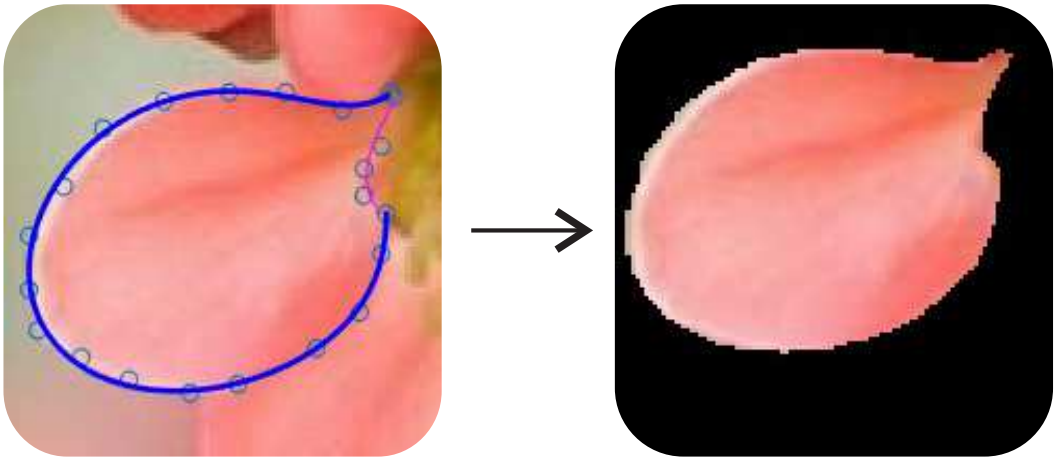


Figura 2.5: Segmentación de la región que encierra el contorno aproximado con curvas Bézier (Elaboración propia, 2019)

Para ello se utiliza el algoritmo de Bresenham (1965) para línea, con el fin de trazar pixeles a lo largo de la línea entre los puntos $c(t_i)$ y $c(t_{i+1})$ de la curva Bézier del contorno. En la Figura 2.6 se muestra como ejemplo el trazo del Algoritmo 2.1 de Bresenham, cuando la pendiente $m < 1$, desde (0,1) a (4,3).

Algoritmo 2.1: Método de Bresenham para línea

```
1  graficaLinea(x0,y0, x1,y1)
2  dx = x1 - x0
3  dy = y1 - y0
4  D = 2*dy - dx
5  y = y0
6  for x desde x0 hasta x1
7      grafica(x,y)
8      if D >0
9          y = y + 1
10         D = D - 2*dx
11     end
12     D = D + 2*dy
```

(xi, yi)
(0,1)
(1,1)
(2,2)
(3,2)
(4,3)

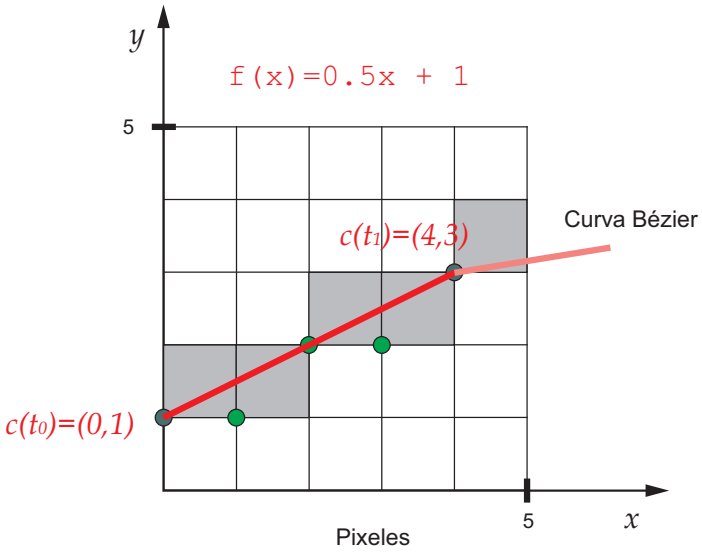


Figura 2.6: Trazo de Bresenham desde (0,1) a (4,3) (Elaboración propia, 2019)

Una vez que se traza cada pixel del contorno sobre una imagen binaria, se procede a llenar la región encerrada por el contorno usando la función *imfill* de Matlab y de esta forma se segmenta cada región de forma interactiva.

2.1.5. Diseño automático del campo de trazos

Para generar las pinceladas a partir de una región de trabajo como la de la Figura 2.7(b), se requiere un campo vectorial que guíe la trayectoria de cada trazo. Esto se ha abordado en la bibliografía utilizando técnicas basadas en el gradiente de la imagen.

Sin embargo, esto causa problemas con la iluminación. Por ejemplo, el brillo de la fruta de la Figura 2.7(a) produciría algunas pinceladas horizontales si se usaran dichas técnicas, cuando por lo general se requieren trazos verticales para esa región, como se observa en la Figura 2.7(c), donde se trazaron pinceladas digitales en Corel Draw y el resultado es visualmente agradable.

Dado lo anterior, en esta tesis se propone el diseño interactivo de un campo vectorial que guíe los trazos, a partir de la información de textura de la región. El diseño interactivo permite el control del trazo de las pinceladas y una interpretación personalizada de la imagen.

En la Figura 2.7(b) se muestra la región de trabajo elegida para explicar esta sección, la cual corresponde a la región principal de la fruta.

(a) Imagen de entrada

(b) Región de trabajo

(c) Ejemplo de trazos pictóricos con cinco niveles de intensidad rojo diseñados digitalmente en Corel Draw

Figura 2.7: Ejemplo de trazos pictóricos en una región de trabajo (Elaboración propia, 2019)

Para diseñar el campo vectorial con un estilo artístico, como se muestra en la Figura 2.7(c), se propone la interpolación mediante curvas Bézier las cuales son suaves en su trayectoria. La interpolación se propone para que los trazos sean uniformes y de esta forma evitar el problema de la iluminación.

Sin embargo, a fin de comparar las técnicas basadas en el gradiente las cuales son automáticas, se expone primero una técnica representativa automática y después la propuesta en esta tesis la cual es interactiva.

En este caso se empleó la construcción de Kang, Seungyong, y Chui (2009) el cual es básicamente un tipo de filtrado bilateral (Tomasi, 1998). En este tipo de filtro el valor en un pixel se obtiene promediando los pixeles vecinos y con valores similares, es decir se toma en cuenta un criterio de distancia espacial y otro de medida de similitud que puede ser de intensidad. En el caso de la construcción mencionada se utilizan cuatro funciones de ponderación como se observa en la Figura 2.8.

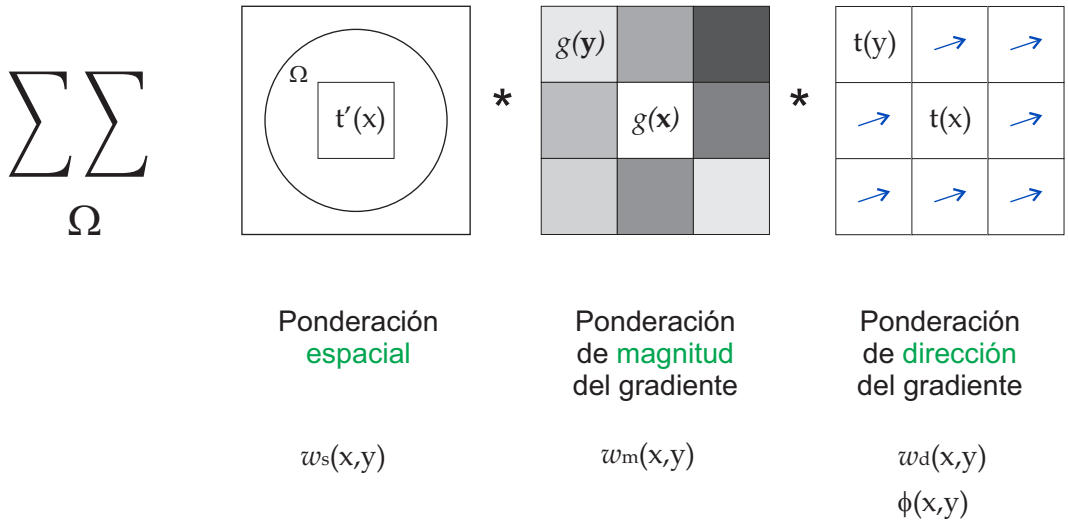


Figura 2.8: Representación gráfica de las funciones de ponderación (Elaboración propia, 2019)

De este modo el filtro queda definido como

$$\mathbf{t'}(\mathbf{x}) = \frac{1}{k}\int\int_{\Omega_r} \phi(\mathbf{x},\mathbf{y})\mathbf{t}(\mathbf{y})w_s(\mathbf{x},\mathbf{y})w_m(\mathbf{x},\mathbf{y})w_d(\mathbf{x},\mathbf{y})d\mathbf{y} \tag{2.22}$$

donde $\Omega_r(\mathbf{x})$ denota la región definida por la función de ponderación espacial dada por

$$w_s(\mathbf{x},\mathbf{y}) = \begin{cases} 1 & \text{si } \|\mathbf{x}\text{ - }\mathbf{y}\| < r \\ 0 & \text{en otro caso} \end{cases} \tag{2.23}$$

donde **x** es el pixel central de una ventana y **y** un pixel de la vecindad, siendo k una constante de normalización. Del mismo modo la función de ponderación de la magnitud del gradiente está dado por

$$w_m(\mathbf{x},\mathbf{y}) = [\hat{g}(\mathbf{y}) - \hat{g}(\mathbf{x}) + 1]/2 \tag{2.24}$$

donde $\hat{g}(\mathbf{x})$ denota la magnitud normalizada del gradiente en **x**. También la función de ponderación de la dirección del gradiente está dada por

$$w_d(\mathbf{x},\mathbf{y}) = |\mathbf{t}(\mathbf{x}) \cdot \mathbf{t}(\mathbf{y})| \tag{2.25}$$

donde $\mathbf{t}(\mathbf{x})$ denota el vector tangente normalizado en **x**. Además se añade la función de

reversión de dirección del vector **y**, si el ángulo entre los vectores es mayor a $\pi/2$.

$$\phi(\mathbf{x},\mathbf{y}) = \begin{cases} 1 & \text{si } \mathbf{t}(\mathbf{x}) \cdot \mathbf{t}(\mathbf{y}) > 0, \\ -1 & \text{en otro caso} \end{cases} \tag{2.26}$$

El campo vectorial inicial $\mathbf{t}^0(\mathbf{x})$ se obtiene tomando los vectores perpendiculares al gradiente normalizado $\mathbf{g}^0(\mathbf{x})$ de la imagen de entrada I. El gradiente de I está dado por

$$\nabla I(x,y) = \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{bmatrix} \approx \begin{bmatrix} I(x,y) - I(x-1,y) \\ I(x,y) - I(x,y-1) \end{bmatrix} \tag{2.27}$$

y la magnitud del gradiente

$$|\nabla I(x,y)| = \sqrt{\left(\frac{\partial I(x,y)}{\partial x}\right)^2 + \left(\frac{\partial I(x,y)}{\partial y}\right)^2} \tag{2.28}$$

El filtro se puede aplicar iterativamente de modo que $\mathbf{t}^i(\mathbf{x}) \rightarrow \mathbf{t}^{i+1}(\mathbf{x})$, para suavizar el campo vectorial.

2.1.6. Diseño interactivo del campo de trazos

Para diseñar el campo interactivo se propone que el usuario introduzca unas pocas líneas manualmente, siguiendo la textura de la región, o a voluntad del usuario cuando no haya textura. Después el programa interpolará automáticamente la(s) curva(s) trazadas. En la Figura 2.9 se muestra el diagrama de flujo del diseño interactivo del campo.

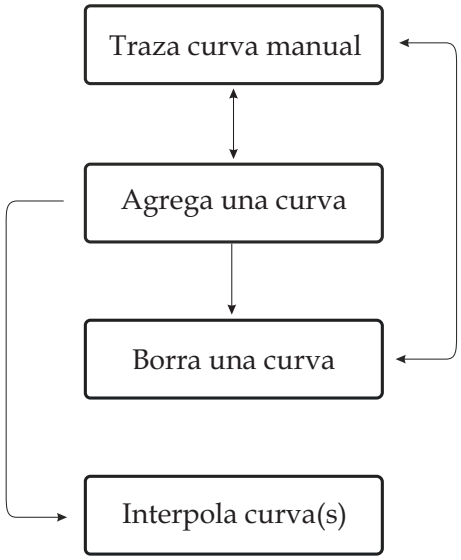


Figura 2.9: Diagrama de flujo del diseño interactivo del campo (Elaboración propia, 2019)

Aunque en esta tesis se utilizó la función de Matlab 2018b: ***F = scatteredInterpolant (x,y,v,'natural','linear')*** para interpolar las curvas, a continuación se describe cómo funciona dicha función.

La interpolación de las curvas requiere el abordaje de dos temas teóricos: 1) la interpolación, en el plano, de valores conocidos solo en algunos puntos dispersos dentro de la envolvente convexa de dichos puntos y, 2) la extrapolación fuera de la envolvente convexa.

La envolvente convexa se puede describir intuitivamente si dado un conjunto de puntos en el plano, en cada punto se coloca un clavo y luego se toma una goma elástica que envuelva todos los puntos como se muestra en la Figura 2.10 .

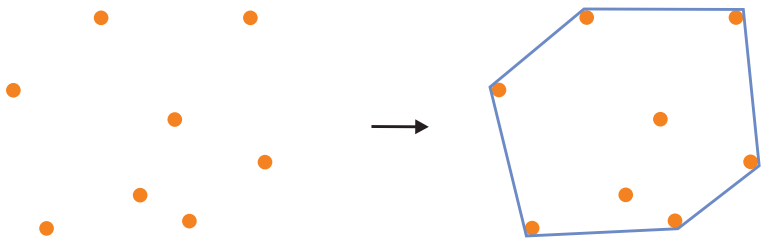


Figura 2.10: Envolvente convexa de un conjunto de puntos (Elaboración propia, 2019)

Los valores a interpolar en el plano son las primeras derivadas en los puntos de la curva. Esto se logra garabateando manualmente la curva y luego aproximándola usando curvas de Bézier como se establece en la ecuación 2.21. De este modo, se pueden derivar los polinomios de Bernstein, y así se tendrían las derivadas en cada punto de la curva.

Como ejemplo supóngase que se garabateó una curva, se guardaron los puntos q_i de dicha curva manual y se aplicó la ecuación 2.21 estableciéndose el grado n en 3 y obteniéndose los 4 puntos de control siguientes: $pt_1 = (5, 10), pt_2 = (18, 8), pt_3 = (38, 25), pt_4 = (45, 15)$. Al desarrollar la ecuación 2.15 para $n = 3$ se obtiene

$$c(t) = (1-t)^3 * pt_1 + 3t(1-t)^2 * pt_2 + 3t^2(1-t) * pt_3 + t^3 * pt_4 \tag{2.29}$$

y derivando con respecto a t se tiene

$$c'(t) = (-3t^2 + 6t - 3) * pt_1 + (9 * t^2 - 12 * t + 3) * pt_2 + (-9 * t^2 + 6 * t) * pt_3 + 3t^2 * pt_4 \tag{2.30}$$

Luego, sustituyendo los 4 puntos de control en ambas ecuaciones se obtiene la gráfica mostrada en la Figura 2.11 usando 15 valores de t entre 0 y 1.

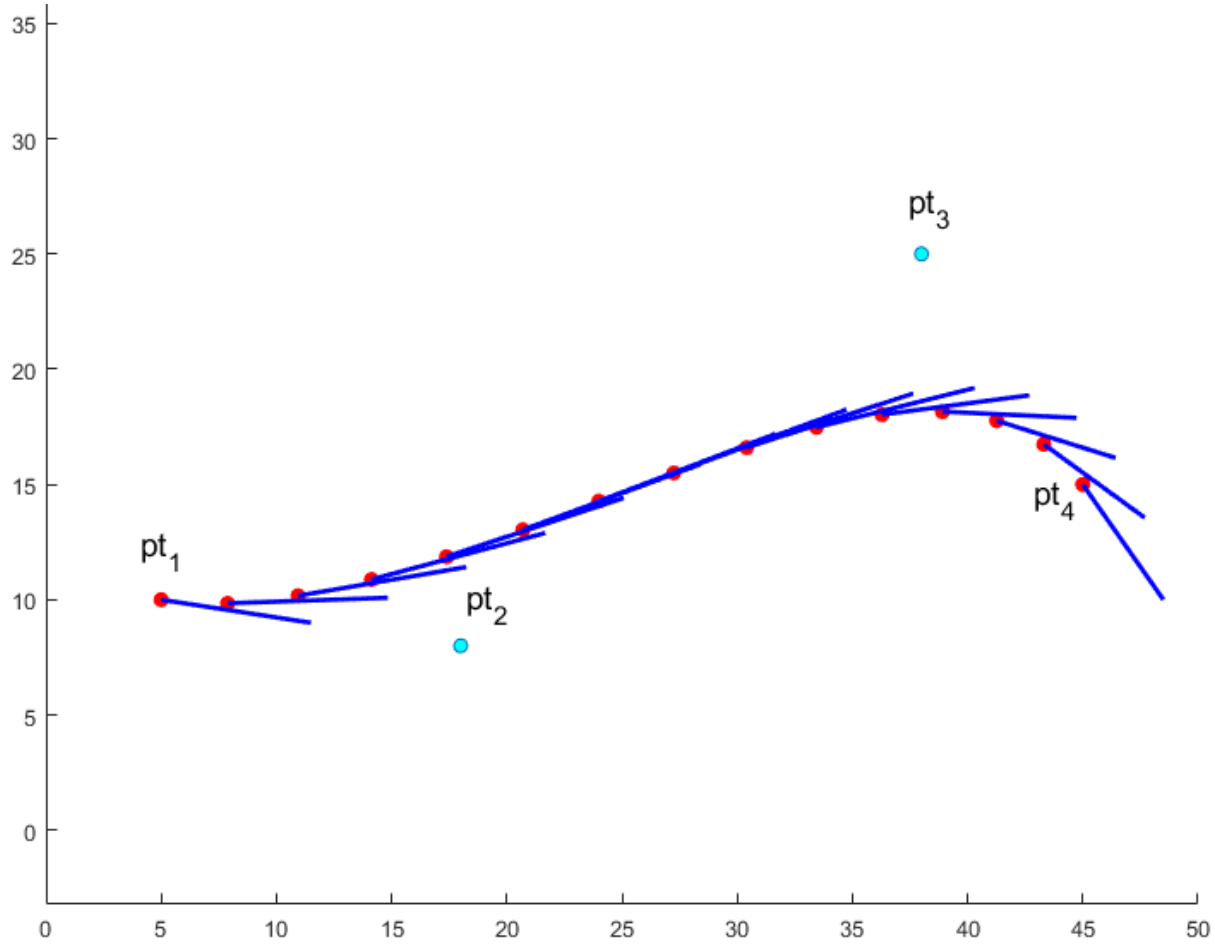


Figura 2.11: Curva Bézier cúbica y su derivada para 15 puntos (Elaboración propia, 2019)

Lo que se busca por tanto, es interpolar estos 15 valores de la derivada en el plano

bidimensional, lo cual se explica a continuación.

I. Interpolación dentro de la envolvente convexa

Los más importantes métodos de interpolación de datos dispersos se pueden clasificar en cuatro categorías (Amidror, 2002): métodos basados en triangulación, métodos de distancia inversa ponderada, métodos de funciones de base radial y métodos de vecino natural.

En esta tesis se utilizó el método de interpolación por vecinos naturales usando las coordenadas de Sibson (Watson, 2001), ya que es un método robusto aunque computacionalmente más lento lo cual no es problema para este trabajo.

El método por vecinos naturales es local y se basa en el diagrama de Voronoi y su estructura geométrica dual, la triangulación de Delaunay del conjunto de datos. Así, los vecinos naturales de un punto son aquellos que están asociados con los polígonos de Voronoi adyacentes.

El diagrama de Voronoi particiona el plano en un conjunto de celdas disjuntas. Cada celda V_i encierra un punto x_i del conjunto de puntos. La celda V_i se define como el área más cercana al punto x_i que a cualquier otro punto, es decir,

$$V_i = \{x \in \mathbb{R}^2 | d(x, x_i) \leq d(x, x_j) \forall j = 1, ..., n\} \tag{2.31}$$

donde $d(a, b)$ denota la distancia euclidiana entre a y b.

Inicialmente se construye un diagrama de Voronoi de todos los puntos dados $\{x_i\}$, después para evaluar el valor interpolado en un nuevo punto q, éste se inserta al conjunto de puntos, permitiendo que dicho punto genere su propia celda robando espacio de las celdas correspondientes a los puntos alrededor (véase la Figura 2.12).

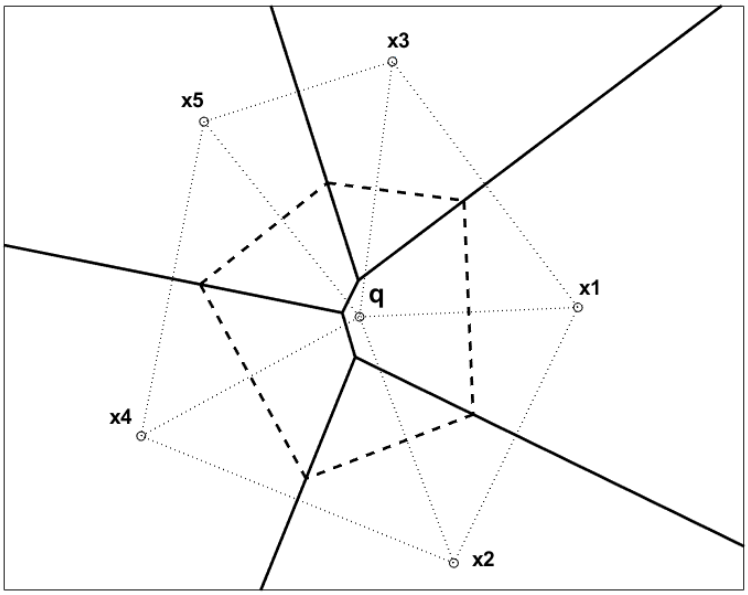


Figura 2.12: La nueva celda de Voronoi del punto q que se superpone a las celdas de los puntos $x_1...x_5$ se indica con líneas discontinuas. También se muestran los triángulos de Delaunay correspondientes con líneas punteadas finas (Elaboración propia, 2019)

La nueva celda de Voronoi creada por el punto de interrogación q se define como

$$V(q) = \{z \in \mathbb{R}^2 | d(z,q) \leq d(z,x_i) \forall i = 1,...,n\} \tag{2.32}$$

donde x_i son los vecinos naturales de q y n es el número de vecinos naturales de q. Las intersecciones de la nueva celda con las celdas anteriores son

$$V_i(q) = V(q) \bigcap V_i \tag{2.33}$$

El interpolante del vecino natural en el punto q se define como (Amidror, 2002)

$$f(q) = \sum_i w_i(q) z_i \tag{2.34}$$

donde z_i es el valor de la función subyacente en el vecino natural x_i, en este trabajo dicha función es la derivada de la curva en x_i y w_i es el peso que pondera la función subyacente y está dado por

$$w_i(q) = \frac{Area[V_i(q)]}{Area[V(q)]}, \; 0 \leq w_i(q) \leq 1, \; \sum_i w_i(q) = 1. \tag{2.35}$$

siendo $Area[Vi(q)]$ el área de cada intersección y $Area[V(q)]$ el área total de la celda.

Este método resulta en una superficie que es continuamente diferenciable excepto en los puntos x_i.

Los pesos o coordenadas w_i en la ecuación 2.35 se pueden obtener de forma más fácil mediante triangulación de Delaunay, para ello se puede utilizar el algoritmo de Watson para calcular las coordenadas de Sibson (Watson, 2001) el cual se resume a continuación.

Algoritmo 2.2: Método de Watson para calcular w_i

1 Obtener la triangulación de Delaunay del conjunto de puntos originales x_i usando el algoritmo de Bowler-Watson (Sloan y Houlsby, 1984);
2 **for** *cada triángulo de Delaunay* t *cuyo circuncírculo contiene a* q **do**
3 sea t_c el circuncentro y t_s el signo de t;
4 **for** *cada arista* i *de* t *con vértices* j *y* k **do**
5 Construir c_i:= circuncentro (q, j, k)
6 **end**
7 **for** *cada vértice* i *de* t*, calcular* **do**
8 $w_i := w_i + t_s$ x determinante$(c_j, c_k, t_c)/2$
9 **end**
10 **end**
11 Normalizar todas las coordenadas w_i;

El código Matlab siguiente simula el Algoritmo 2.2 en forma de función:

```
function [Rx,Ry]=coordenadasWi(x,y,dx,dy,tamx,tamy)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Coordenadas o pesos wi del vecino natural
% (x,y) son las coordenadas de entrada
% (dx,dy) son los valores a interpolar
% (tamx,tamy) es el tamaño de la imagen
% [Rx,Ry] es el resultado
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Se triangula Delaunay
coordXY=[x',y'];
DT = delaunayTriangulation(coordXY);
CH = convexHull(DT);
coordVert = DT.Points;
conectividad = DT.ConnectivityList;% vértices
[C,r] = circumcenter(DT);
CCs=zeros(3,2);
Rx=zeros(tamx,tamy);
Ry=zeros(tamx,tamy);
```

```
for qx=1:tamx
 for qy=1:tamy
  % si el punto a interrogar se encuentra en la
  % envolvente convexa de los puntos a interpolar
  if( inpolygon(qx,qy,DT.Points(CH,1),DT.Points(CH,2)))
  wi=zeros(length(coordXY),1);

  % Se calculan las áreas relacionadas con los vértices
  for i=1:length(conectividad) % n° de triángulos
  xxx=coordVert(conectividad(i,:),1);
  yyy=coordVert(conectividad(i,:),2);
  % Determinante para ver si X está dentro
  % de la circunferencia circunscrita de cada triángulo Delaunay
  D=det([xxx(1), yyy(1), xxx(1)^2+yyy(1)^2, 1;
         xxx(2), yyy(2), xxx(2)^2+yyy(2)^2, 1;
         xxx(3), yyy(3), xxx(3)^2+yyy(3)^2, 1;
         qx    , qy    , qx^2+qy^2        , 1]);
  % Triángulos vecinos naturales de X
   if(D>0)
      % se construyen los circumcentros(X,j,k) de un triángulo
      % tomando los dos vértices de cada arista y el punto X
        for j=1:3
          p1=j;
          p2=mod(j,3)+1;
          P=[xxx(p1),yyy(p1);xxx(p2),yyy(p2);qx,qy];
          % lista de conectividad
          TT=[1 2 3];
          % triangulacion
          TR = triangulation(TT,P);
          % 3 circumcentros
          [CC,rr]=circumcenter(TR);
          CCs(j,:)=CC;
        end

        % para cada vértice de un triángulo de Delaunay
        % se calcula el área signada
        % vértice 1
        areaSignada1=0.5*det([C(i,:),1; CCs(3,:),1; CCs(1,:),1]);
        wi(DT(i,1))=wi(DT(i,1))+areaSignada1;
        % vértice 2
        areaSignada2=0.5*det([C(i,:),1; CCs(1,:),1; CCs(2,:),1]);
        wi(DT(i,2))=wi(DT(i,2))+areaSignada2;
        % vértice 3
        areaSignada3=0.5*det([C(i,:),1; CCs(2,:),1; CCs(3,:),1]);
        wi(DT(i,3))=wi(DT(i,3))+areaSignada3;
   end
  end
  % Se normaliza áreas
  summ=sum(wi);
  wiN=wi/summ;
  % se interpola la derivada en el plano
  ddx=dx'.*wiN;
  ddy=dy'.*wiN;
  Rx(qy,qx)=sum(ddx);
  Ry(qy,qx)=sum(ddy);
```

```
  end

 end
end
```

Mediante este código al interpolar los valores de la derivada de la Figura 2.11 se obtuvo la interpolación dentro de la envolvente convexa como se muestra en la Figura 2.13.

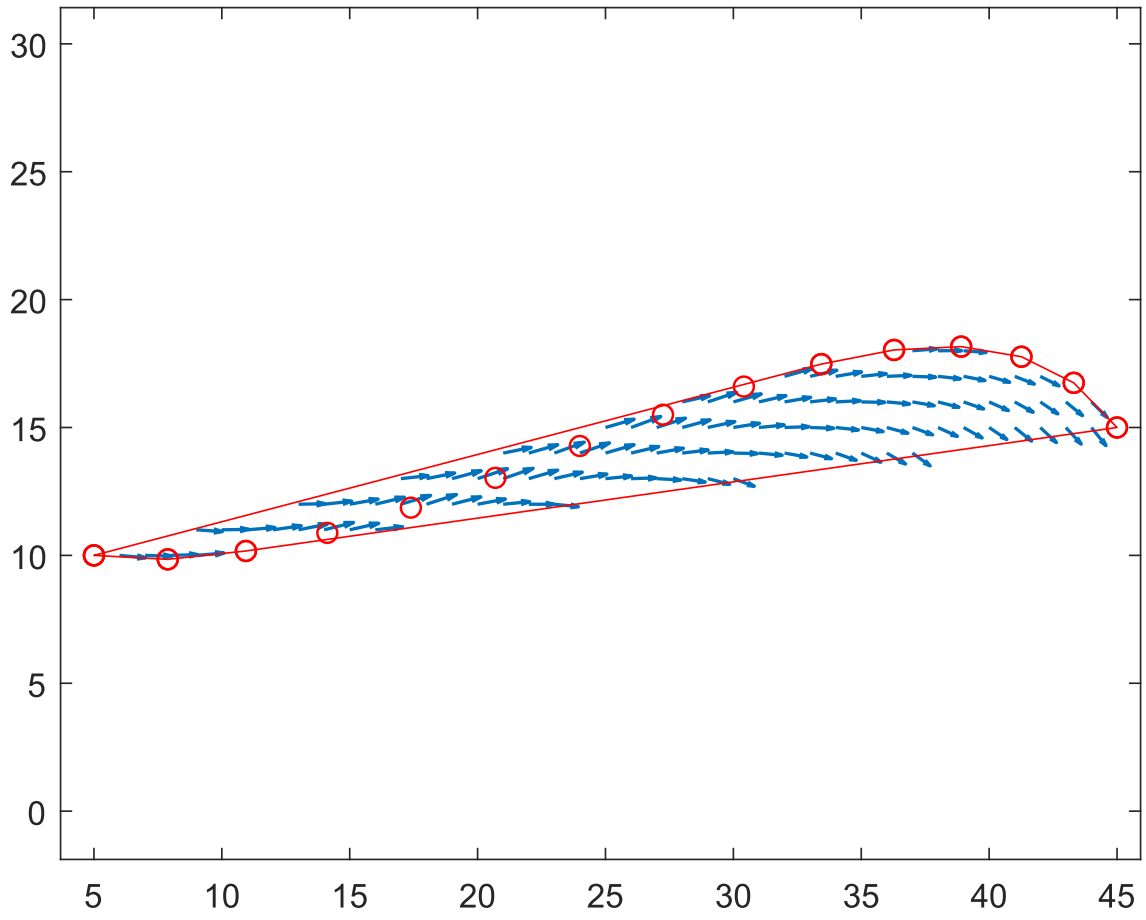


Figura 2.13: Interpolación dentro de la envolvente convexa (línea roja) usando vecinos naturales para los 15 puntos (círculos) de la Figura 2.11 (Elaboración propia, 2019)

II. Extrapolación fuera de la envolvente convexa

Para extrapolar fuera de la envolvente convexa se requiere construir una función que interpole suavemente en los límites de la envolvente convexa. Esto se puede lograr utilizando el método de la extensión de la envolvente convexa (Franke, 1979) e introduciendo el concepto de las curvas de Bézier como sigue.

El exterior de la envolvente convexa se divide en rectángulos y triángulos semiinfinitos,

como se muestra en la Figura 2.14, trazando perpendiculares a las aristas exteriores de la envolvente convexa desde cada vértice.

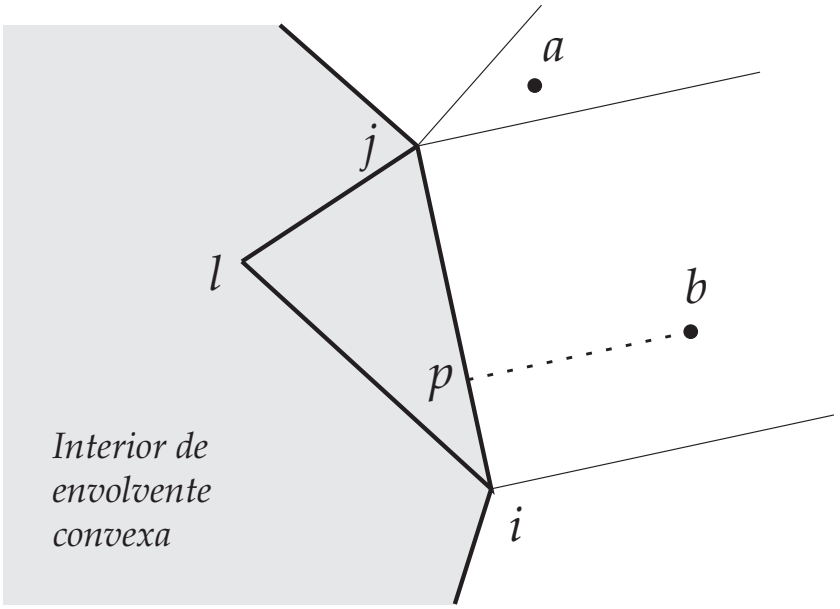


Figura 2.14: Geometría de la extrapolación triangular (Elaboración propia, 2019)

Para un punto en un triángulo exterior como a, el valor a extrapolar es el que produce la función subyacente en el vértice j al que se agrega la derivada de dicha función ponderada por la distancia es decir

$$F(x_a, y_a) = c'(t_j) + c''(t_j)\frac{\|p_a - p_j\|}{k} \tag{2.36}$$

donde k es una constante de normalización, $\|p_a - p_j\| = \sqrt{(x_a - x_j)^2 + (y_a - y_j)^2}$ es la distancia entre los puntos p_a y p_j, $c'(t)$ es la derivada de la curva Bézier en los puntos i, j y está dada por la ecuación 2.30 y $c''(t)$ es la segunda derivada de la curva Bézier de la ecuación 2.29 y está dada por

$$c''(t) = (-6t + 6) * pt_1 + (18t - 12) * pt_2 + (-18t + 6) * pt_3 + 6t * pt_4 \tag{2.37}$$

siendo pt_i los puntos de control de la ecuación 2.29.

Para un punto en un rectángulo exterior como b, sea p la proyección de (x_b, y_b) en el lado ij, y sean $(b_i, b_j, 0)$ las coordenadas baricéntricas de p en el triángulo T_{ijl}. Entonces la

extrapolación se obtiene de la combinación lineal de la función subyacente en i y en j:

$$F(x_b, y_b) = b_i[c'(t_i) + c''(t_i)\frac{\|p_b - p\|}{k}] + b_j[c'(t_j) + c''(t_j)\frac{\|p_b - p\|}{k}] \tag{2.38}$$

Siguiendo la metodología anterior se simuló la extrapolación lineal en un rectángulo y en un triángulo mediante el código Matlab siguiente, donde los puntos pt1, pt2, pt3 y pt4 se muestran en la Figura 2.11.

```
% Puntos de control hipotéticos, suponiendo aproximación Bézier cúbica
pt1 = [ 5;10];
pt2 = [18; 8];
pt3 = [38; 25];
pt4 = [45; 15];
% muestreo del parámetro t
t = linspace(0,1,15);
% ecuación Bézier cúbica
pts = kron((1-t).^3,pt1) + kron(3*(1-t).^2.*t,pt2) +...
      kron(3*(1-t).*t.^2,pt3) + kron(t.^3,pt4);
% primera derivada de los polinomios de Bérnstein
a = -3*t.^2 +  6*t - 3;
b =  9*t.^2 - 12*t + 3;
c = -9*t.^2 +  6*t;
d =  3*t.^2;
der = kron(a,pt1) + kron(b,pt2) + kron(c,pt3) + kron(d,pt4);
% segunda derivada de los polinomios de Bérnstein
aprima =-6*t + 6;
bprima = 18*t - 12;
cprima = -18*t +6;
dprima = 6*t;
der2=kron(aprima,pt1) + kron(bprima,pt2) + ...
     kron(cprima,pt3) + kron(dprima,pt4);

%% envolvente convexa
DT = delaunayTriangulation(pts');
CH = convexHull(DT);
plot(DT.Points(CH,1),DT.Points(CH,2),'r-o');
axis equal;hold on;
%% extrapolación  en el rectángulo
n1=3;n2=4;
p1=DT.Points(CH(n1),:);
p2=DT.Points(CH(n2),:);
plot(p1(1),p1(2),'bx'); % primer punto
plot(p2(1),p2(2),'gx'); % segundo punto
% ecuación de la recta
m=(p2(2)-p1(2))/(p2(1)-p1(1));
b=-m*p1(1) + p1(2);
denominador=sqrt(m*m+1);

for yyy=1:1:40
 for xxx=1:1:64
  Pxy=[xxx,yyy]; %punto de prueba
  alfa=dot(Pxy-p1,p2-p1)/norm(p2-p1)^2; % coordenadas baricéntricas
  %return ((b.X - a.X)*(c.Y - a.Y) - (b.Y - a.Y)*(c.X - a.X)) > 0;
```

```
  izder=(p2(1)-p1(1))*(yyy-p1(2))-(p2(2)-p1(2))*(xxx-p1(1));
   if(alfa>=0 && alfa<=1 && izder<=0)
   beta=1-alfa;
   distancia=abs(m*xxx -yyy + b)/denominador;
   cteNormal=20;
   % segunda derivada
   dxx=distancia*(beta*(der2(1,CH(n1))) +alfa*(der2(1,CH(n2))))/cteNormal;
   dyy=distancia*(beta*(der2(2,CH(n1))) +alfa*(der2(2,CH(n2))))/cteNormal;
   % primera derivada
   Fdx=beta*(der(1,CH(n1))) +alfa*(der(1,CH(n2)));
   Fdy=beta*(der(2,CH(n1))) +alfa*(der(2,CH(n2)));
   % suma la segunda derivada
   Fdx=Fdx+dxx;
   Fdy=Fdy+dyy;
   % normaliza
   Fdxn=Fdx/(sqrt(Fdx*Fdx + Fdy*Fdy));
   Fdyn=Fdy/(sqrt(Fdx*Fdx + Fdy*Fdy));
   l2=line([xxx,xxx+Fdxn],[yyy,yyy+Fdyn]); % visualiza
   l2.Color='black';
  end
 end
end
%% extrapolación en el triángulo
% producto punto p1 y p2 de la envolvente convexa:
% pi=p2-p1, y el punto desconocido px en el exterior
n1=3;n2=4; % puntos de la envolvente convexa elegidos
p1=DT.Points(CH(n1),:);
p2=DT.Points(CH(n2),:);
pi1=p2-p1;
pxy1=0; % coordenada conocida
pxx1=(pi1(1)*p2(1)-pi1(2)*pxy1+pi1(2)*p2(2))/pi1(1);%coord. desconocida
p1_ext=[pxx1,pxy1]; % punto 1 exterior
% se busca la coordenada desconocida del segundo punto
n3=5;% punto de la envolvente convexa elegido
p3=DT.Points(CH(n3),:);
pi2=p3-p2;
pxx2=64; % coordenada conocida
pxy2=(pi2(1)*p2(1)-pi2(1)*pxx2 + pi2(2)*p2(2))/pi2(2);%coord. desconocida
p2_ext=[pxx2,pxy2]; % punto 2 exterior
% Puntos del triángulo
T1=[p2(1)         p2(2)];
T2=[p1_ext(1) p1_ext(2)];
T3=[p2_ext(1) p2_ext(2)];
cteNormal=20; %constante de normalización

for yyy=1:1:40
 for xxx=1:1:64
    dist=norm(p2-[xxx,yyy]);
    % coordenadas baricéntricas, para trabajar sólo dentro del triángulo
    alfa = f_ab(xxx, yyy, T2, T3) / f_ab(T1(1), T1(2), T2, T3);
    beta = f_ab(xxx, yyy, T3, T1) / f_ab(T2(1), T2(2), T3, T1);
    gamma= f_ab(xxx, yyy, T1, T2) / f_ab(T3(1), T3(2), T1, T2);
      if(alfa >= 0 && alfa <= 1 && beta >= 0 && beta <= 1 && gamma >= 0 && gamma <= 1)
       Fdx=der(1,CH(n2))+dist*der2(1,CH(n2))/cteNormal;
       Fdy=der(2,CH(n2))+dist*der2(2,CH(n2))/cteNormal;
       %normaliza
```

```
       Fdxn=Fdx/(sqrt(Fdx*Fdx + Fdy*Fdy));
       Fdyn=Fdy/(sqrt(Fdx*Fdx + Fdy*Fdy));
       l3=line([xxx,xxx+Fdxn],[yyy,yyy+Fdyn]); % visualiza
       l3.Color='magenta';
       end
 end
end

%% interpolación dentro de la envolvente convexa
[Rx,Ry]=coordenadasWi(pts(1,:),pts(2,:),der(1,:),der(2,:),60,45);
[xx,yy]=meshgrid(1:45,1:60)
quiver(xx,yy,Rx,Ry,'LineWidth',1);axis equal;hold on;plot(pts(1,:),pts(2,:),'ro');
%% Función para calcular las coordenadas baricéntricas
function f=f_ab(x, y, pa, pb)
f = (pa(2) - pb(2)) * x + (pb(1) - pa(1)) * y + pa(1)*pb(2) - pb(1)*pa(2);
end
```

En la Figura 2.15 se muestra la extrapolación lineal fuera de la envolvente convexa para los datos de la Figura 2.13 utilizando el código Matlab anterior.

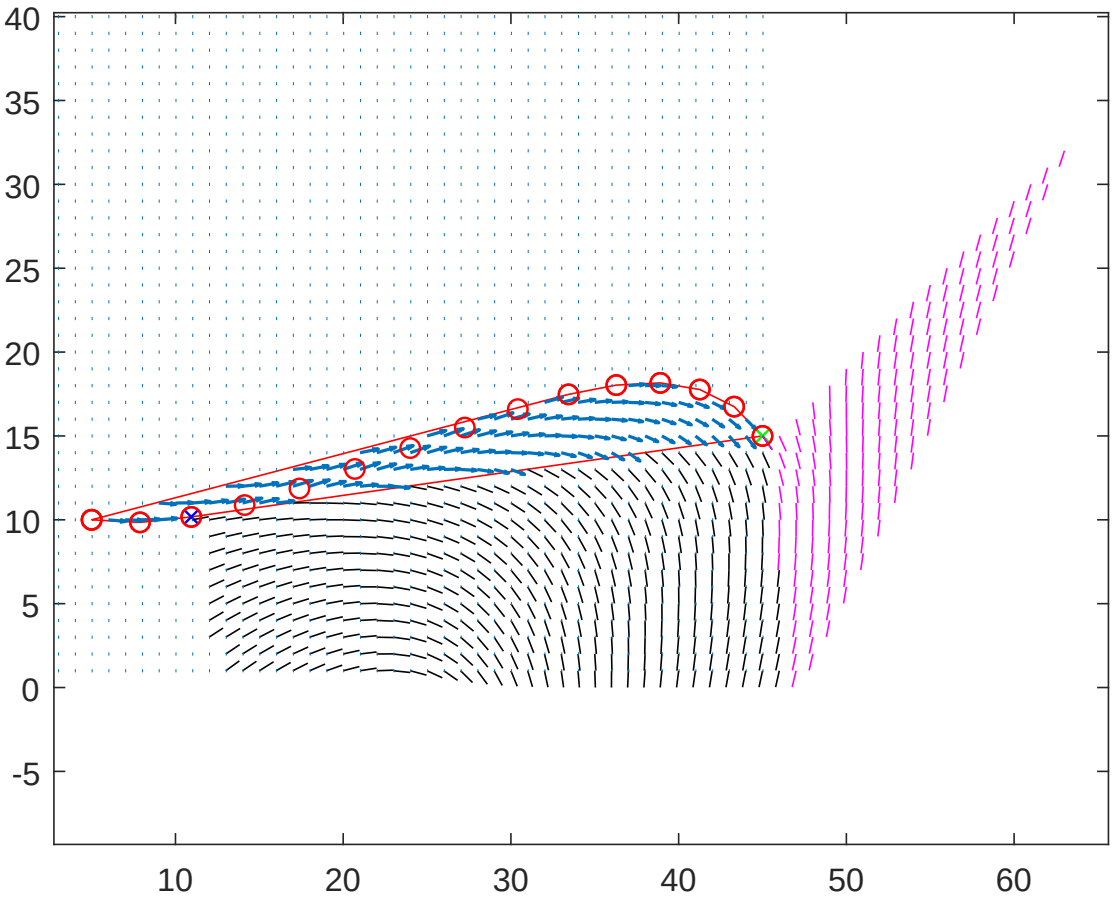


Figura 2.15: Extrapolación lineal fuera de la envolvente convexa (negro = región rectangular y magenta = región triangular) para los datos de la Figura 2.13 (Elaboración propia, 2019)

Como se observa, se simularon dos regiones exteriores significativas, una rectangular y una triangular, las cuales se pueden extender según sea necesario.

2.2. Marco metodológico

A fin de resolver el problema de investigación, la falta de desarrollo tecnológico en el sector de la producción artesanal, se utilizó la metodología general para un proceso de investigación transdisciplinario (Hernández-Aguilar, 2018) la cual consta de cuatro fases. Para este capítulo se utilizó la fase (III) Investigación experimental, estableciéndose la planeación y el establecimiento de las actividades de investigación experimental, como se

muestra en la Figura 2.16.

Figura 2.16: Metodología de investigación a emplear, Perspectiva Transdisciplinaria (Td) (Elaboración propia, 2019)

El marco metodológico se dividió en dos grupos principales: actividades previas a la investigación y actividades específicas de investigación, las cuales se describen gráficamente en las Figuras 2.17 y 2.18.

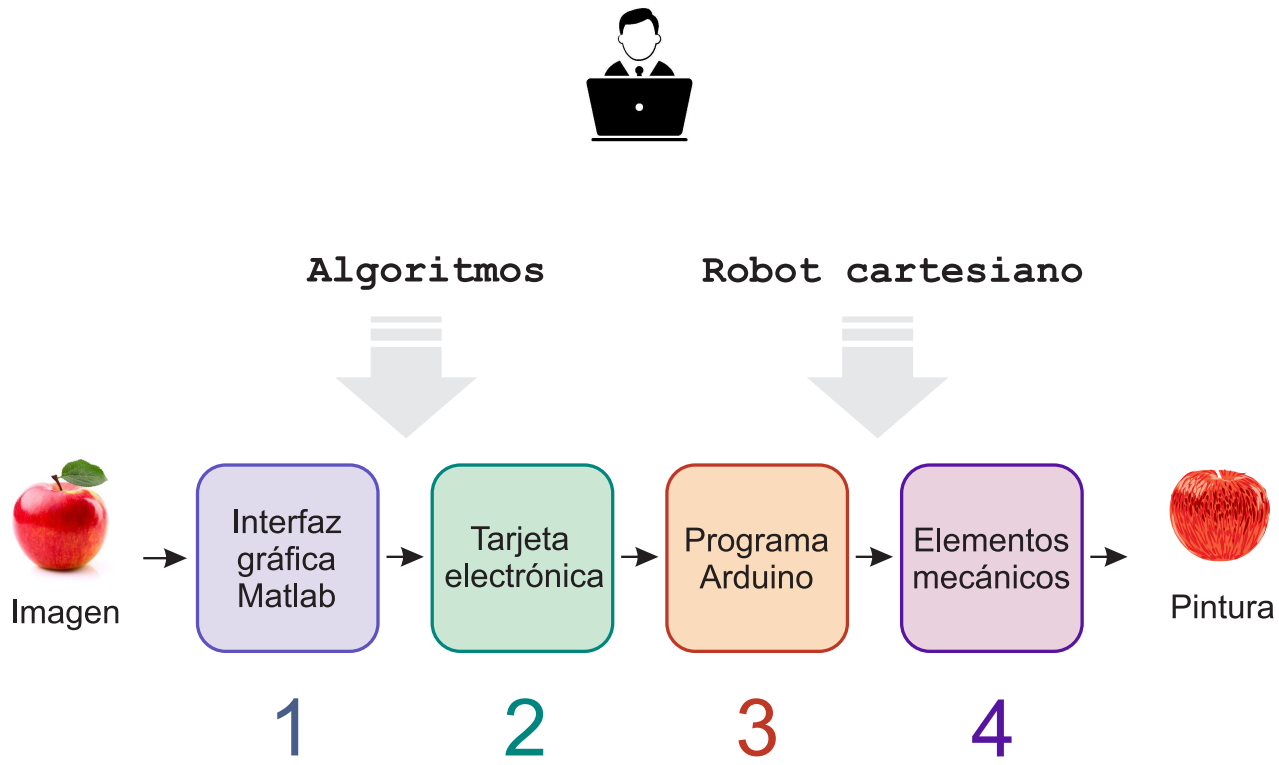


Figura 2.17: Actividades previas a la investigación para obtener resultados (Elaboración propia, 2019)

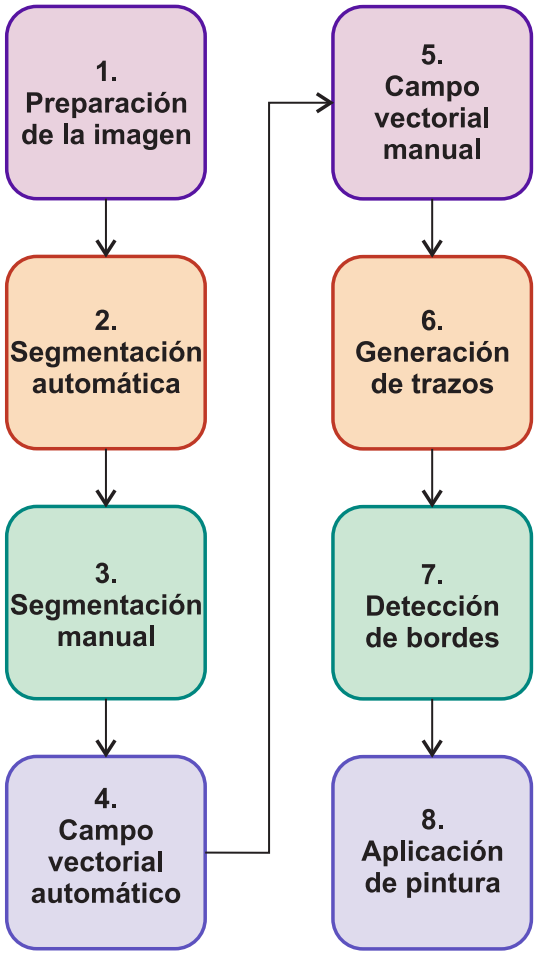


Figura 2.18: Actividades específicas de investigación (Elaboración propia, 2019)

Como se observa en la Figura 2.17, se realizaron cuatro actividades previas a la investigación. Así mismo, para resolver el problema de investigación se propusieron 8 actividades de investigación a evaluar las cuales se muestran en la Figura 2.18. Las 4 actividades previas a la investigación y las 8 actividades específicas de investigación se abordan a detalle en el Capítulo 3: Aplicación de la metodología.

Es importante mencionar que las 8 actividades de investigación sólo se centran en el aspecto computacional ya que no se abordan los aspectos mecánicos ni electrónicos del robot.

Capítulo 3

Aplicación de la metodología

3.1. Actividad de investigación previa 1: interfaz gráfica Matlab

3.1.1. Introducción

La interfaz gráfica permite al usuario usar el programa sin preocuparse por los comandos que se ejecutan en el programa.

En la Figura 3.1 se muestra el diseñador de interfaz gráfica de usuario GUIDE de Matlab, donde se observan los elementos disponibles.

3.1.2. Materiales y métodos

La interfaz gráfica de usuario permite que el manejo de los procesos sea más rápido y adaptable. Para lograr esto se proponen los pasos descritos en la Figura 3.2.

La interfaz gráfica administrará los algoritmos subyacentes, los cuales se describen en el marco teórico. Sin embargo para desarrollar los algoritmos se requieren realizar al menos los pasos descritos en la Figura 3.3.

3.1.3. Resultados

En la Figura 3.4 se muestra una imagen de la interfaz gráfica RegionesTrazos.fig que se diseñó en Matlab, junto con su código correspondiente RegionesTrazos.m que se encuentra

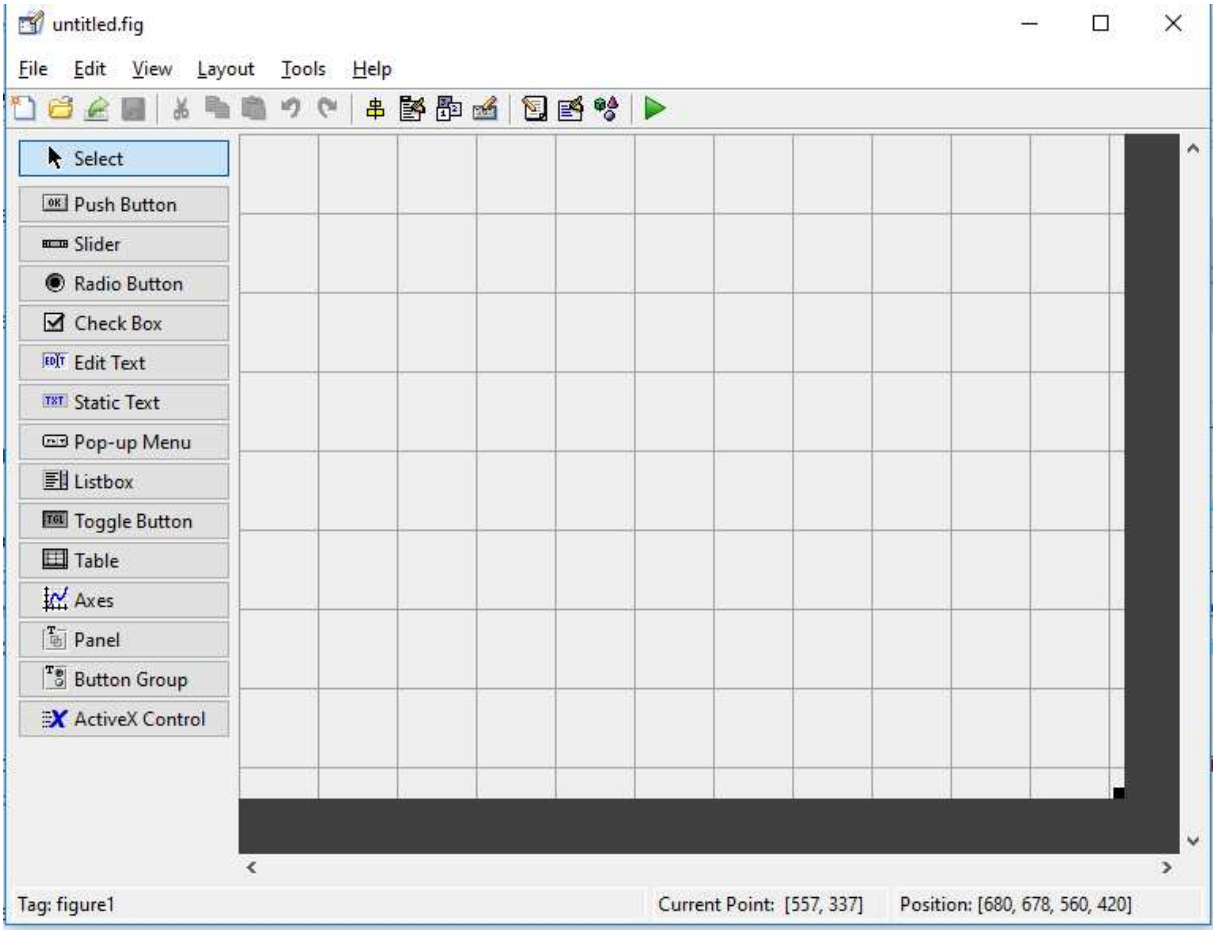


Figura 3.1: Diseñador de interfaz gráfica de usuario GUIDE en Matlab. La interfaz gráfica de usuario facilita la interacción del usuario con la computadora. (Elaboración propia, 2019)

en el Código 1 de Matlab del Anexo 1. En total se escribieron 1500 líneas de código en Matlab para desarrollar la interfaz gráfica.

Se describen a continuación los componentes de la interfaz en orden numérico, indicándose las líneas del código Matlab correspondiente en RegionesTrazos.m. (1) Botón de zoom. (2) Botón de panorama. (3) Botón para abrir imagen. Líneas 54-111. (4) Botón para dar clic en puntos consecutivos, espaciados, sobre el contorno de la región a segmentar. Los puntos deberán cerrar el contorno finalizando en el punto inicial. El algoritmo tomará los puntos muestreados y detectará esquinas en el contorno marcado, por lo que los puntos seleccionados deberán muestrear lo suficiente el contorno y sus esquinas. Líneas 112-265. (5) Control deslizante que especifica el grado de ajuste de la curva Bézier seleccionada. Líneas 308-420. (6) Menú desplegable que muestra la curva seleccionada. El número de curvas lo determina el algoritmo de detección de esquina. Líneas 268-306. (7) Botón para

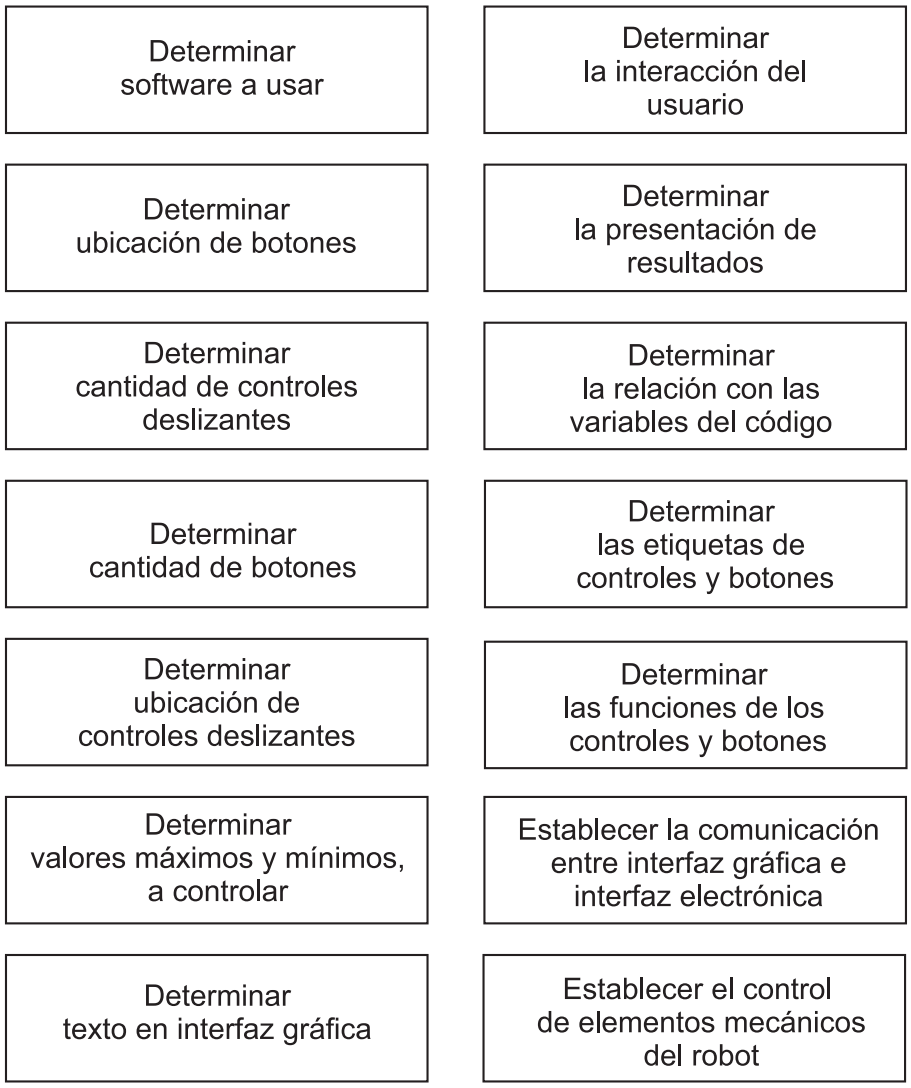


Figura 3.2: Pasos para desarrollar la interfaz gráfica de usuario (Elaboración propia, 2019)

agregar región. Líneas 422-448. (8) Botón para segmentar regiones de forma automática. Líneas 603-628. (9) Control deslizante para especificar el número de regiones a segmentar de forma automática. Líneas 1118-1125. (10) Control deslizante para especificar el número de iteraciones de difusión anisotrópica. Líneas 1128-1135. (11) Botón para realizar difusión anisotrópica. Líneas 577-601. (12) Casilla de verificación, si está activada la operación de suavizado usando difusión anisotrópica se realiza sobre la imagen original, en caso contrario se realiza sobre la última imagen de forma iterativa. (13) Botón para realizar detección de bordes usando el método de Canny. Líneas 915-1036. (14) Información de un pixel en la imagen. (15) Botón para obtener el campo vectorial basado en el gradiente de forma automática. Líneas 799-824. (16) Control deslizante que especifica el número de iteraciones para suavizar el campo vectorial. Líneas 1147-1155. (17) Control

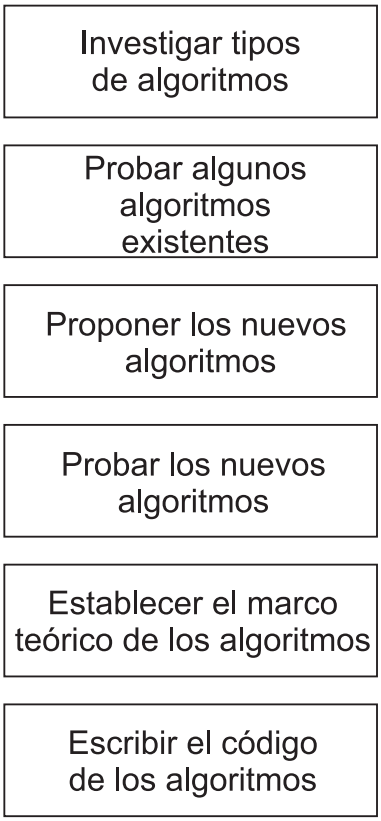


Figura 3.3: Pasos para desarrollar los algoritmos de la interfaz gráfica (Elaboración propia, 2019)

deslizante que especifica el tamaño de la ventana para calcular el campo vectorial. Líneas 1137-1145. (18) Botón para agregar curva que servirá para interpolar el campo vectorial. Líneas 468-492. (19) Botón para trazar curva que servirá para interpolar el campo vectorial. Líneas 450-466. (20) Botón para interpolar las curvas que generarán el campo vectorial. Líneas 520-575. (21) Borra una curva. Líneas 494-518. (22) Ventana donde se visualiza la imagen y los procesamientos efectuados en ella. (23) Control deslizante que especifica el número de niveles de color a pintar. Líneas 1157-1165. (24) Botón para crear trazos tomando como guía el campo vectorial diseñado. Líneas 630-772. (25) Menú desplegable que define la región de trabajo seleccionada. Líneas 1083-1096. (26) Control deslizante que define el ancho de la pincelada. Líneas 1167-1175. (27) Control deslizante que define el largo de la pincelada. Líneas 1177-1185. (28) Control deslizante que define el área de trabajo del robot. Líneas 1187-1201. (29) Casilla de verificación, si está activada las pinceladas son de un sólo color. (30) Control deslizante que define el número de color a pintar a partir de una paleta de pintura. Líneas 1204-1211. (31) Botón para abrir el puerto serie. Líneas 773-798. (32) Botón para pintar la región. Líneas 826-868. (33) Botón para

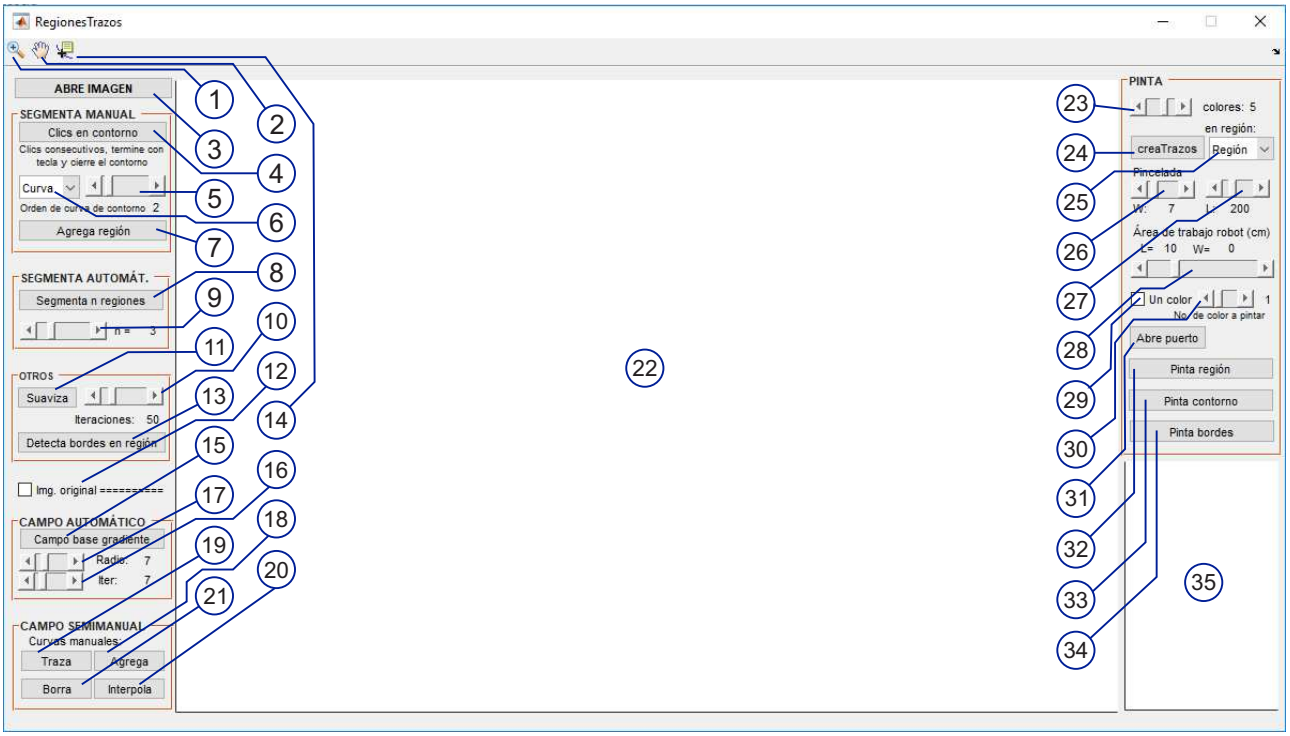


Figura 3.4: Interfaz gráfica en Matlab (Elaboración propia, 2019)

pintar el contorno. Líneas 870-912. (34) Botón para pintar bordes. Líneas 1038-1081. (35) Ventana para visualizar imagen.

Todos los elementos de la interfaz gráfica se agruparon en 6 bloques principales los cuales se muestran en la Figura 3.5.

Figura 3.5: Bloques principales de la interfaz gráfica (Elaboración propia, 2019)

De esta manera, el orden en que se empleen los distintos elementos de la interfaz gráfica dependerá de la imagen de entrada y de las intenciones del usuario. En el diagrama de la Figura 3.6 se muestra el flujo de las operaciones de procesamiento de imagen que se pueden realizar con la interfaz gráfica diseñada, mostrándose en los números el correspondiente bloque en el que se encuentra la operación.

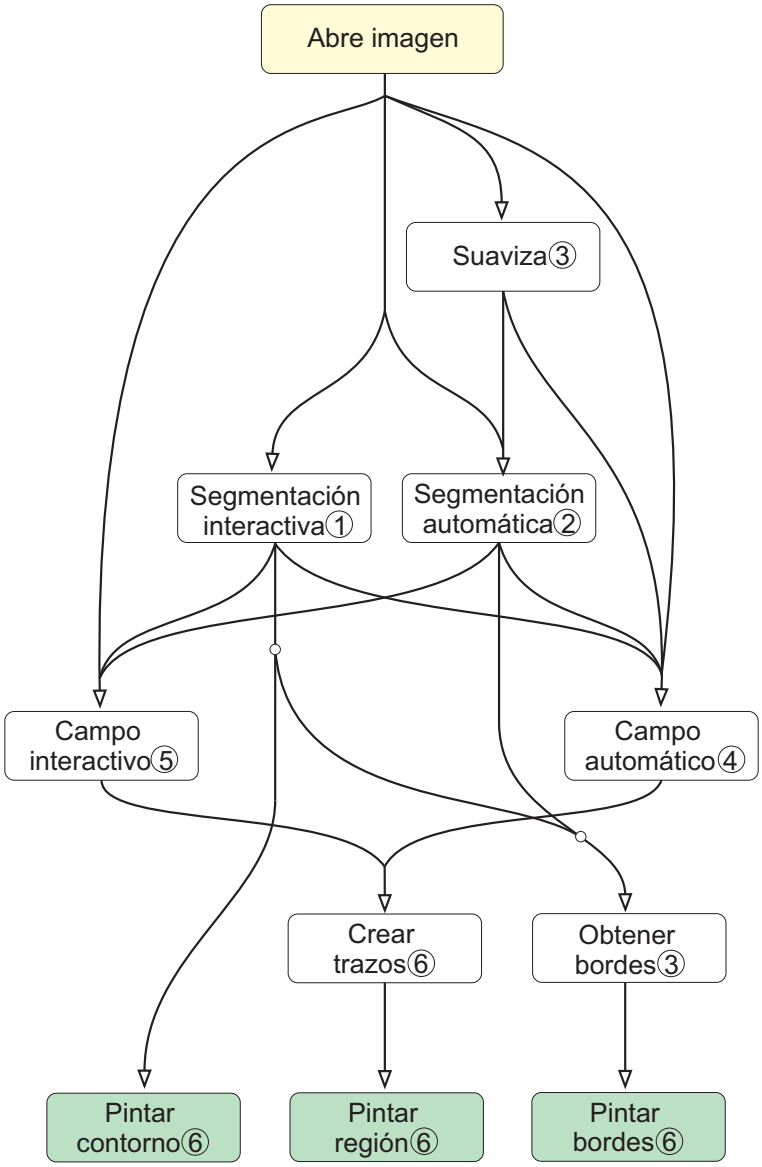


Figura 3.6: Flujo de operaciones de procesamiento usando la interfaz gráfica. En círculos se muestra el correspondiente bloque en el que se encuentra la operación en la Figura 3.5 (Elaboración propia, 2019)

A modo de ejemplo, el diagrama de la Figura 3.7 muestra un posible orden de uso siguiendo un enfoque interactivo.

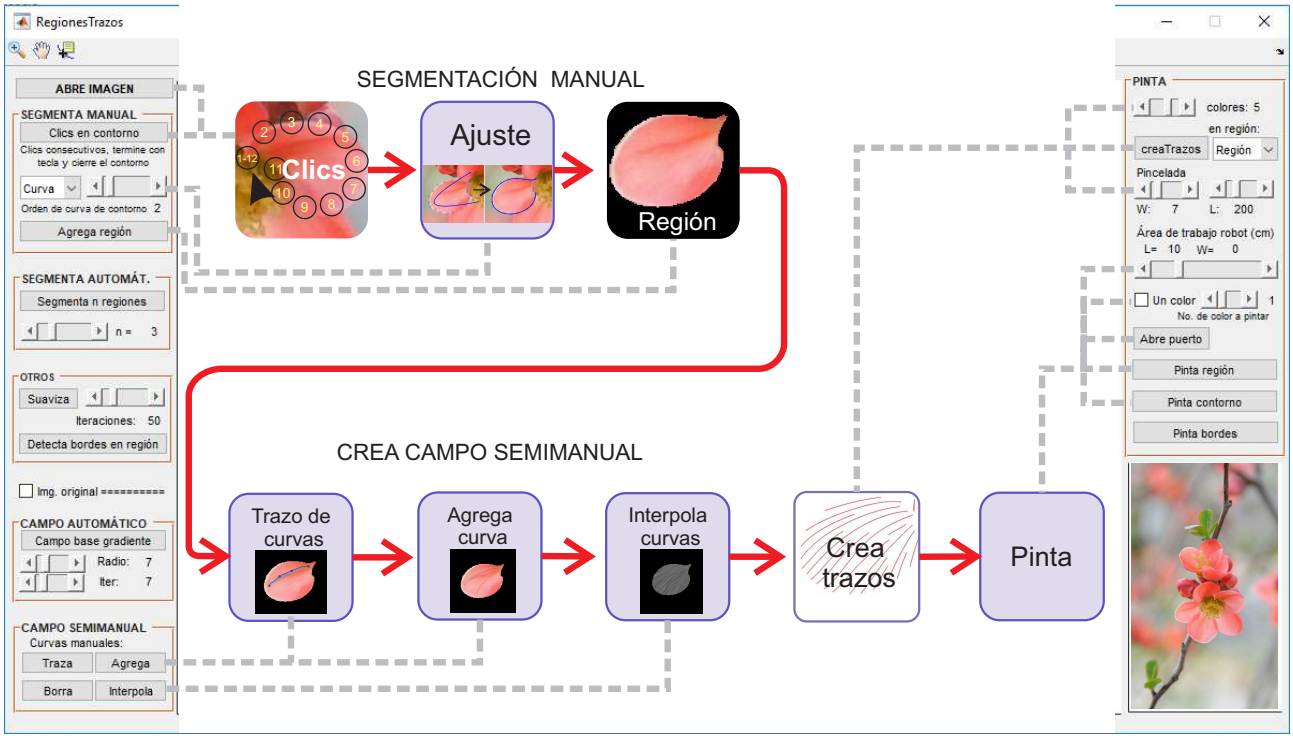


Figura 3.7: Posible orden de uso de la interfaz gráfica (Elaboración propia, 2019)

3.2. Actividad de investigación previa 2: tarjeta electrónica del robot cartesiano

3.2.1. Introducción

La tarjeta electrónica sirve de interfaz entre los elementos mecánicos del robot y la computadora, recibiendo instrucciones de ésta y activando los motores para lograr la posición deseada. En la Figura 3.8 se muestra una visión rica de la tarjeta electrónica.

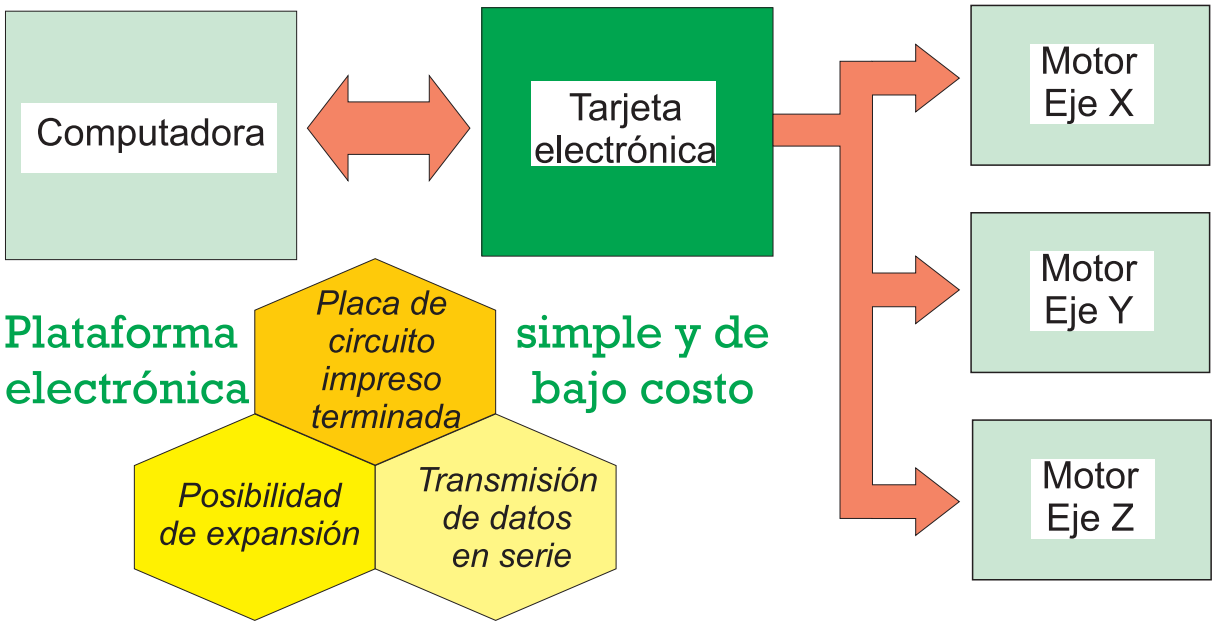


Figura 3.8: Visión rica de la tarjeta electrónica (Elaboración propia, 2019)

3.2.2. Materiales y métodos

Siendo el objetivo establecer comunicación entre la computadora y el robot, se siguió la metodología mostrada en la Figura 3.9 para seleccionar la tarjeta electrónica.

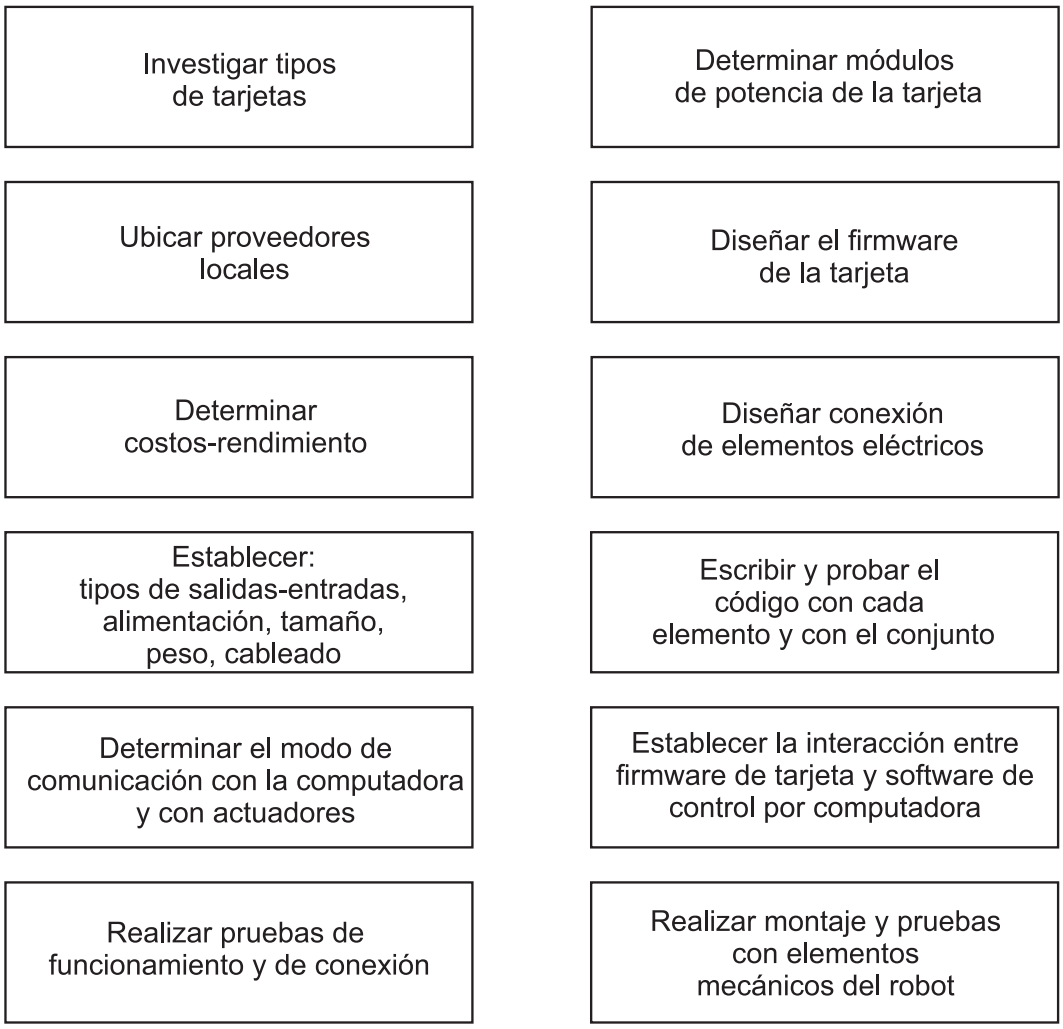
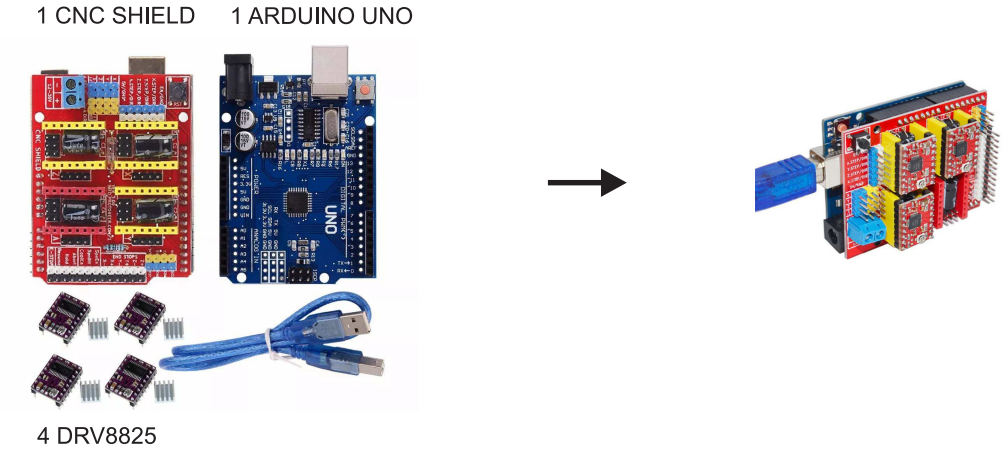


Figura 3.9: Metodología para seleccionar la tarjeta electrónica (Elaboración propia, 2019)

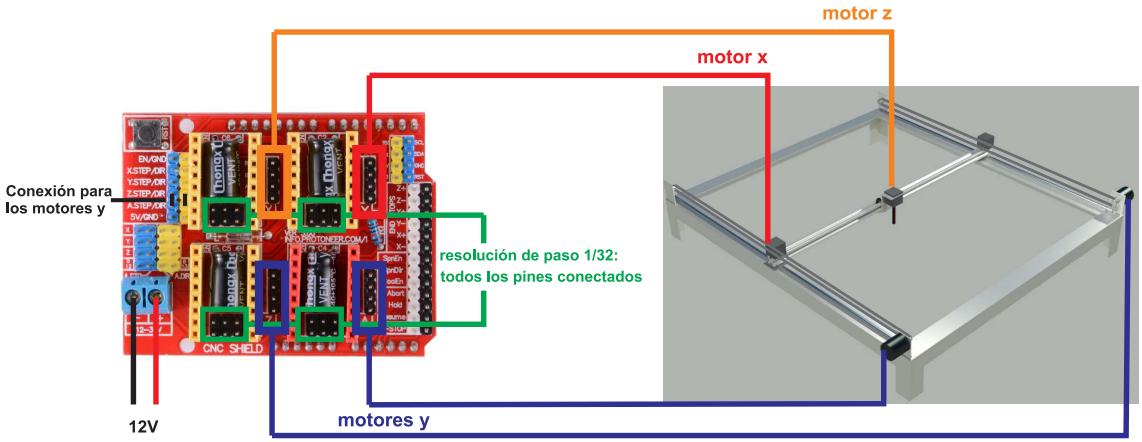
3.2.3. Resultados

Para manejar el robot cartesiano se utilizó la tarjeta CNC Shield, una pequeña placa comercial que permite controlar hasta 4 motores paso a paso gracias a su formato insertable adaptado para la tarjeta Arduino Uno. La tarjeta CNC Shield soporta 4 controladores de potencia Pololu DRV8825 y dispone de conexiones para conectar interruptores de final de carrera y sensores. Es compatible con el firmware de control GRBL, aunque en este trabajo se escribió el software de la tarjeta Arduino de forma personalizada, esto con el fin de tener mayor control sobre el trazo de la pinceladas las cuales se aproximan mediante curvas Bézier. En la Figura 3.10 (a) se muestran los módulos de la tarjeta que se utilizó: 1 tarjeta CNC Shield, 1 tarjeta Arduino UNO, 4 tarjetas DRV8825, los cuales se insertan como muestra la Figura 3.10 (a) a la derecha. Del mismo modo se muestran las conexiones

de la tarjeta CNC Shield con los motores a paso y la configuración de pines para que la resolución de paso para cada motor sea de 1/32. El motor que se utilizó fue un motor de 200 pasos por vuelta, por lo que la resolución lograda con la tarjeta fue de 6400 pasos por vuelta.



(a)



(b)

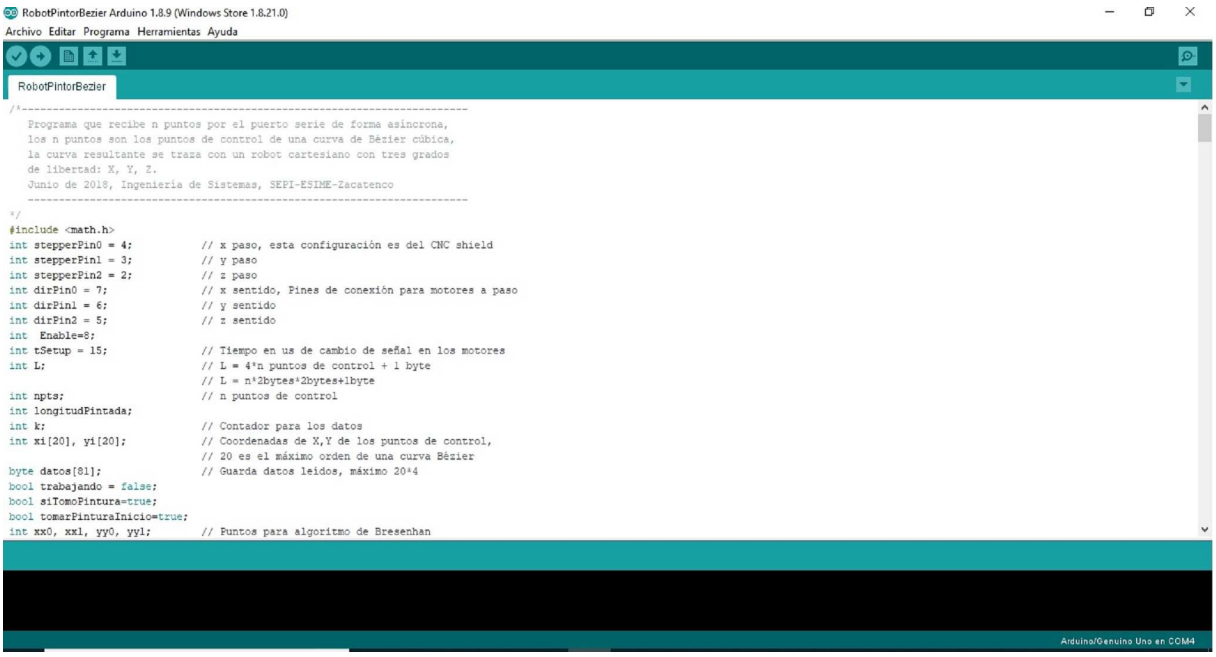
Figura 3.10: Tarjeta electrónica utilizada, (a) montaje de módulos de la tarjeta, (b) conexión de motores (Elaboración propia, 2019)

3.3. Actividad de investigación previa 3: programa de la tarjeta arduino

3.3.1. Introducción

El programa que se escribe en la tarjeta electrónica es el firmware del microcontrolador el cual se graba en la memoria de la tarjeta.

En este trabajo debido a su bajo costo y sencillez, se utilizó la tarjeta Arduino Uno, la cual se programa en el editor que se muestra en la Figura 3.11.




```
/*-------------------------------------------------------------------------
   Programa que recibe n puntos por el puerto serie de forma asincrona,
   los n puntos son los puntos de control de una curva de Bézier cúbica,
   la curva resultante se traza con un robot cartesiano con tres grados
   de libertad: X, Y, Z.
   Junio de 2018, Ingeniería de Sistemas, SEPI-ESIME-Zacatenco
   -------------------------------------------------------------------------
*/
#include <math.h>
int stepperPin0 = 4;          // x paso, esta configuración es del CNC shield
int stepperPin1 = 3;          // y paso
int stepperPin2 = 2;          // z paso
int dirPin0 = 7;              // x sentido, Pines de conexión para motores a paso
int dirPin1 = 6;              // y sentido
int dirPin2 = 5;              // z sentido
int  Enable=8;
int tSetup = 15;              // Tiempo en us de cambio de señal en los motores
int L;                        // L = 4*n puntos de control + 1 byte
                              // L = n*2bytes*2bytes+1byte
int npts;                     // n puntos de control
int longitudPintada;
int k;                        // Contador para los datos
int xi[20], yi[20];           // Coordenadas de X,Y de los puntos de control,
                              // 20 es el máximo orden de una curva Bézier
byte datos[81];               // Guarda datos leidos, máximo 20*4
bool trabajando = false;
bool siTomoPintura=true;
bool tomarPinturaInicio=true;
int xx0, xx1, yy0, yy1;       // Puntos para algoritmo de Bresenhan
```

Figura 3.11: Editor de programa de la tarjeta Arduino (Elaboración propia, 2019)

3.3.2. Materiales y métodos

Para desarrollar el firmware de la tarjeta se requieren seguir los pasos de la Figura 3.12.

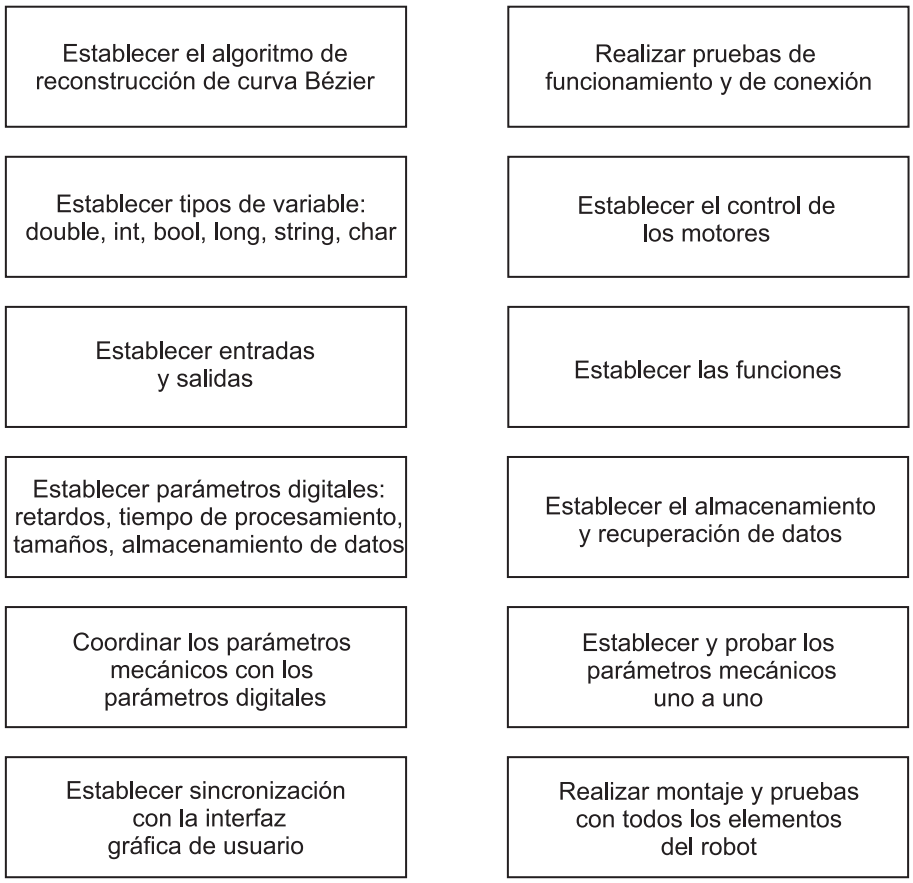


Figura 3.12: Pasos para desarrollar el firmware de la tarjeta electrónica (Elaboración propia, 2019)

3.3.3. Resultados

El programa que se grabó en la tarjeta Arduino UNO se encuentra en el Código del Anexo 2. El algoritmo que se ejecuta en dicho programa se muestra en el Algoritmo 3.1.

Algoritmo 3.1: Programa en Arduino

Entrada: N puntos de control de una curva Bézier

1 Lee los n puntos de control;

2 Reconstruye la curva Bézier de grado $n < 20$;

3 Traza líneas rectas en el robot usando el algoritmo de Bresenham;

Salida : Líneas de la t-ésima parte del parámetro t

Se puede tener una apreciación general del Código del Anexo 2 si se observa el diagrama de la Figura 3.13 donde se muestra la operación entre la computadora y el microcontrolador en Arduino para cada curva trazada.

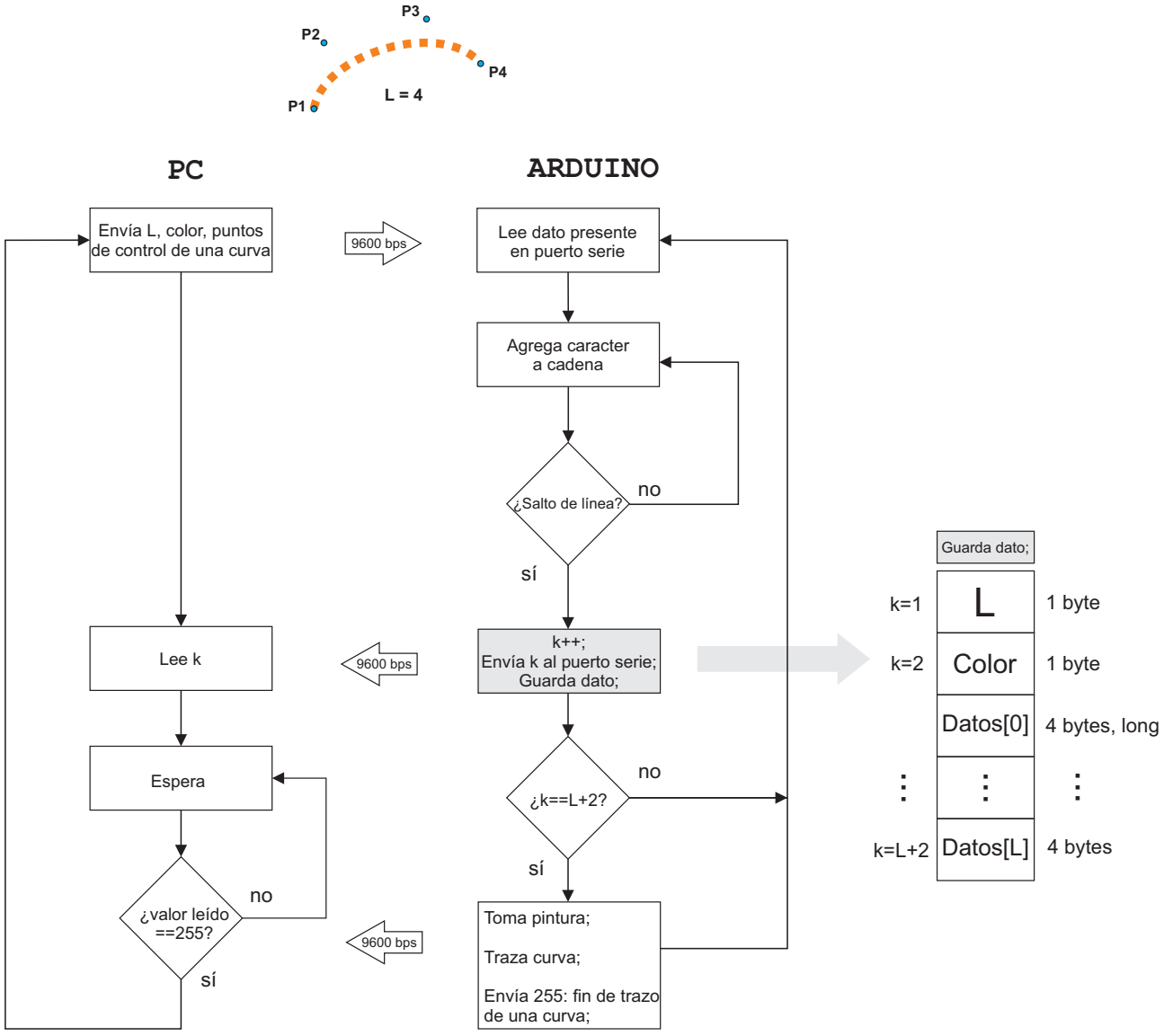


Figura 3.13: Operación entre computadora y microcontrolador para cada curva trazada (Elaboración propia, 2019)

El total de líneas de código que se programaron en Arduino fue de 430 líneas.

3.4. Actividad de investigación previa 4: elementos mecánicos del robot

3.4.1. Introducción

Para probar los algoritmos de aplicación de pintura se fabricó un robot cartesiano de tres grados de libertad debido a las siguientes dos razones.

1. Aunque el brazo robot es el sistema más usado de acuerdo con la revisión bibliográfica, es muy alto su costo de adquisición.

2. Un robot cartesiano comercial, está limitado, por hardware y por software, al trazo de líneas y curvas preestablecidas. Por lo que el trazo personalizado de curvas de Bézier no es posible.

3.4.2. Materiales y métodos

El diseño de los elementos mecánicos del robot cartesiano se propuso siguiendo los pasos mostrados en la Figura 3.14.

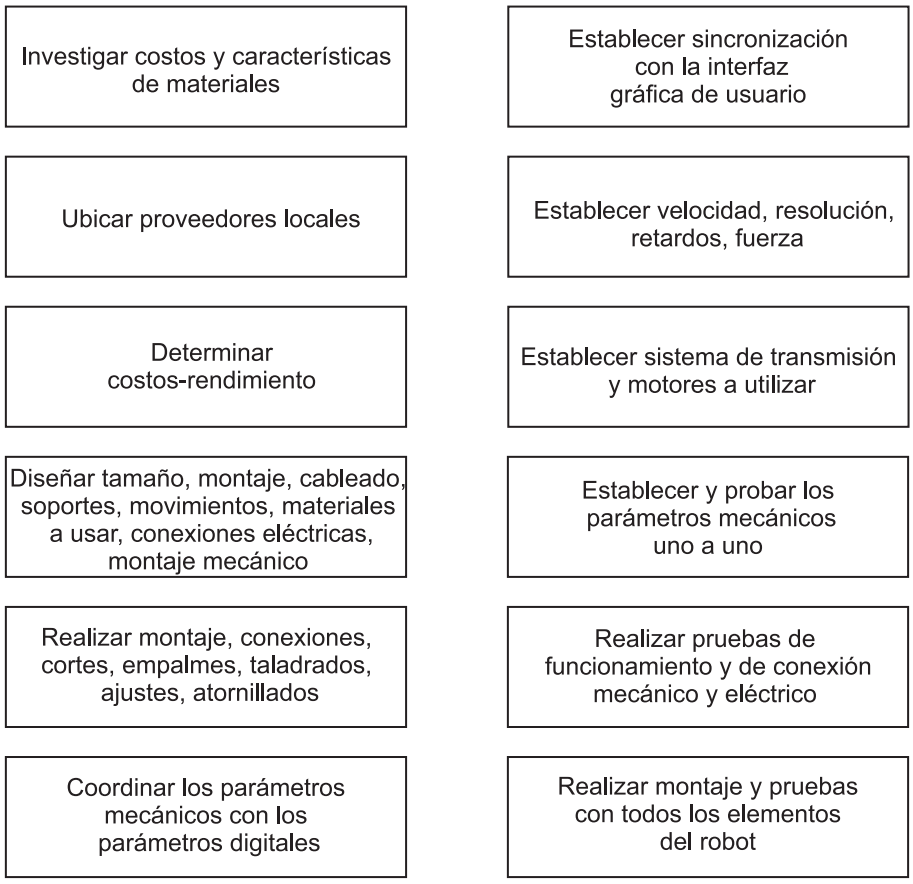


Figura 3.14: Pasos para desarrollar los elementos mecánicos del robot cartesiano (Elaboración propia, 2019)

3.4.3. Resultados

De este modo, se elaboró el sistema usando materiales disponibles en tiendas locales con las dimensiones descritas en la Figura 3.15.

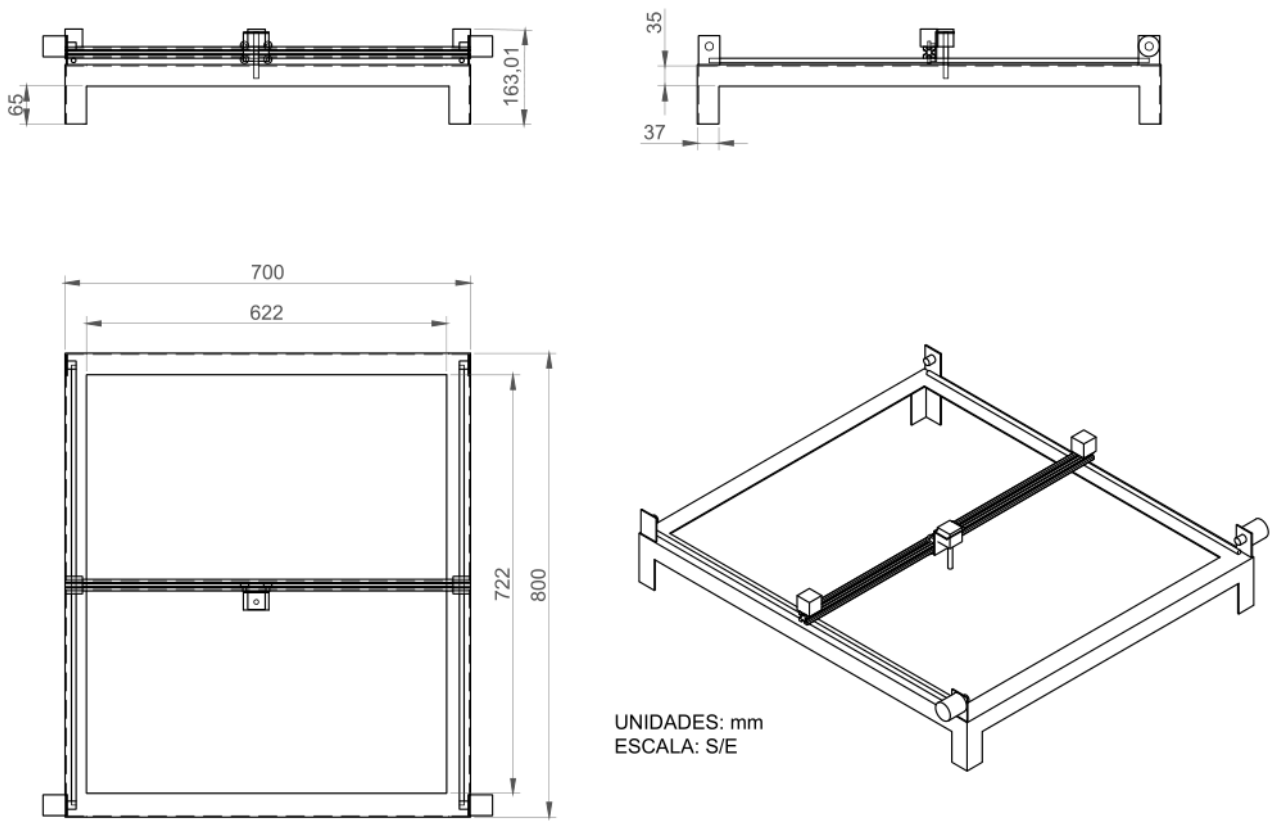
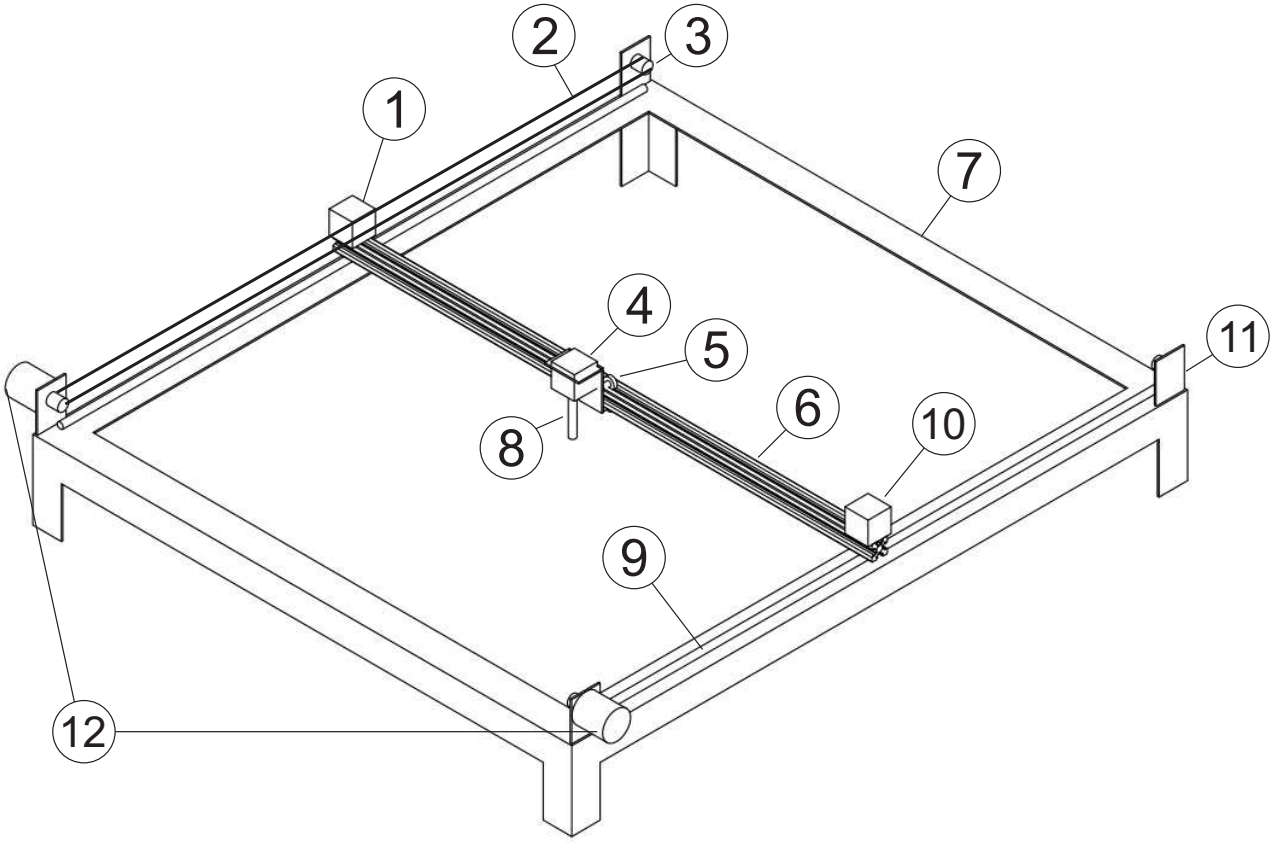


Figura 3.15: Plano de la base del robot cartesiano (Elaboración propia, 2019)

Los elementos mecánicos que lo integran se muestran en la Figura 3.16



DESCRIPCIÓN DE ELEMENTOS

① Motor a pasos NEMA 17, 200 pasos por vuelta, eje x
② Banda dentada de 6mm de ancho y 2mm de paso, de poliuretano
③ Polea dentada, 5 mm diámetro interior, diente redondo, distancia entre dientes 2mm, n° de dientes 20, diámetro exterior 16mm, alto 16mm, de aleación de aluminio
④ Motor a pasos NEMA 17, 200 pasos por vuelta, eje z
⑤ Set de 4 ruedas sujetadas con rodamiento de bolas de 5mm de diámetro interior, de 5mm de espesor y 25 mm de diámetro, de caucho, separadas 30mm una de la otra, aseguradas con tornillo y tuerca de 3mm, forma un carro para transportar el motor④
⑥ Eslabón de perfil de aluminio estructural de 20x20mm
⑦ Base de aluminio de ángulo de 1 1/2" con las medidas especificadas en el plano de la base.
⑧ Tornillo sin fin de paso por vuelta de 6.4mm
⑨ Varilla de acero inoxidable de 8mm de diámetro, sobre la que se montan dos rodamientos lineales de 8mm que se sujetan al eslabón⑥
⑩ Guía ciega para polea③
⑪ Soporte para polea③
⑫ 2 Motores a pasos NEMA 17, 200 pasos por vuelta, eje y

NOTA: las dimensiones de la base se especifican en el plano de la base.

Figura 3.16: Elementos mecánicos del robot cartesiano (Elaboración propia, 2019)

En la Tabla 3.1 se muestran las especificaciones del robot que se diseñó.

Tabla 3.1: Especificaciones del robot, (Elaboración propia, 2019)

Característica	Especificación
Área de trabajo	60x50cm
Resolución	6.25μm
Velocidad	20mm/s

3.5. Actividad de investigación 1: Preparación de la imagen utilizando filtrado anisotrópico

3.5.1. Introducción

El objetivo principal de preparar la imagen es facilitar su segmentación automática. Lo ideal sería que la imagen preparada tuviera regiones homogéneas en color, en intensidad y en textura; además de preservar los bordes. Esto con el fin de que el algoritmo de segmentación, la etapa de procesamiento siguiente, logre mejores resultados. La preparación puede resolverse para imágenes sencillas, con dos o tres regiones simples, utilizando un filtro de difusión anisotrópica tal como describen las ecuaciones 2.9 – 2.12. También se puede utilizar el filtrado bilateral (Tomasi, 1998), otro filtro que es más rápido que el filtrado anisotrópico pero que ofrece menores resultados. En este trabajo no se considera crítico el tiempo, por lo que el filtro iterativo de difusión anisotrópica se utiliza para imágenes simples.

3.5.2. Materiales y métodos

El filtrado anisotrópico se implementó en Matlab 2018b en una computadora con dos procesadores Intel de 1.7Ghz, 8GB de memoria RAM, utilizando el sistema operativo Windows 10.

Para realizar el filtrado anisotrópico se utilizó la ecuación 2.9, usando un valor de k=2 en la ecuación 2.11. Como imagen de entrada se utilizó una imagen en color de 256 niveles de gris en cada componente de color R,G,B.

El flujo de operaciones realizadas respecto del procesamiento general de la Figura 3.6 es *Abre imagen* $\rightarrow$ *Suaviza*. El orden de elementos activados en la interfaz gráfica de la Figura 3.4 fue: Botón 3 $\rightarrow$ Botón 11, los cuales llamaron las líneas de código 54–111 y

578–601 respectivamente de RegionesTrazos.m, existiendo la llamada a la función *difusionanisotropica* en las líneas 587, 588 y 589 donde se procesan los componentes R,G,B de la imagen. La función mencionada, la cual realiza la difusión anisotrópica, se encuentra en las líneas 1286-1308 del código Matlab de Anexo 1.

3.5.3. Resultados

La imagen de la Figura 3.17 muestra la aplicación de difusión anisotrópica en una imagen en color de 550x580 pixeles a la cual se aplicó 600 iteraciones.

(a) Imagen I_0 (b) Imagen I para t=600

Figura 3.17: Difusión anisotrópica en imagen en color (Elaboración propia, 2019)

Para medir el desempeño de la imagen de difusión se puede emplear el criterio de la entropía de Shannon. En teoría de la información, la entropía de Shannon es una medida de la incertidumbre asociada a una variable aleatoria. Una entropía grande indica un alto grado de incertidumbre y una alta complejidad sobre un evento.

La entropía de Shannon para una imagen cuya intensidad tiene valores entre 0 y 255 está dada por

$$E = -\sum_{i=0}^{255} p(z_i)\, log_2\, p(z_i) \tag{3.1}$$

donde $p(z_i)$ es el histograma de la imagen y z_i es una variable aleatoria indicando la intensidad.

A fin de calcular la entropía de Shannon, la imagen en color se convirtió a escala de gris y se graficó la entropía hasta 2000 iteraciones como se muestra en la Figura 3.18.

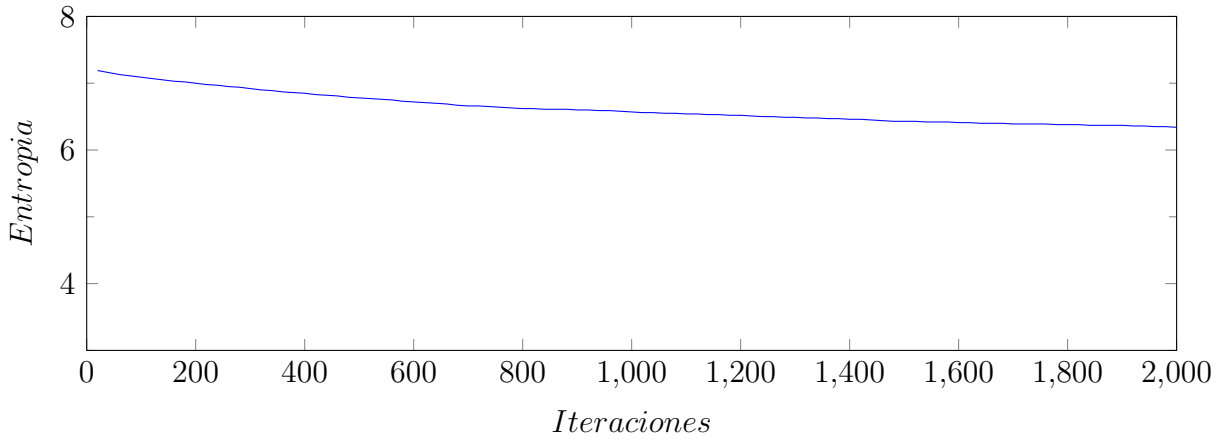
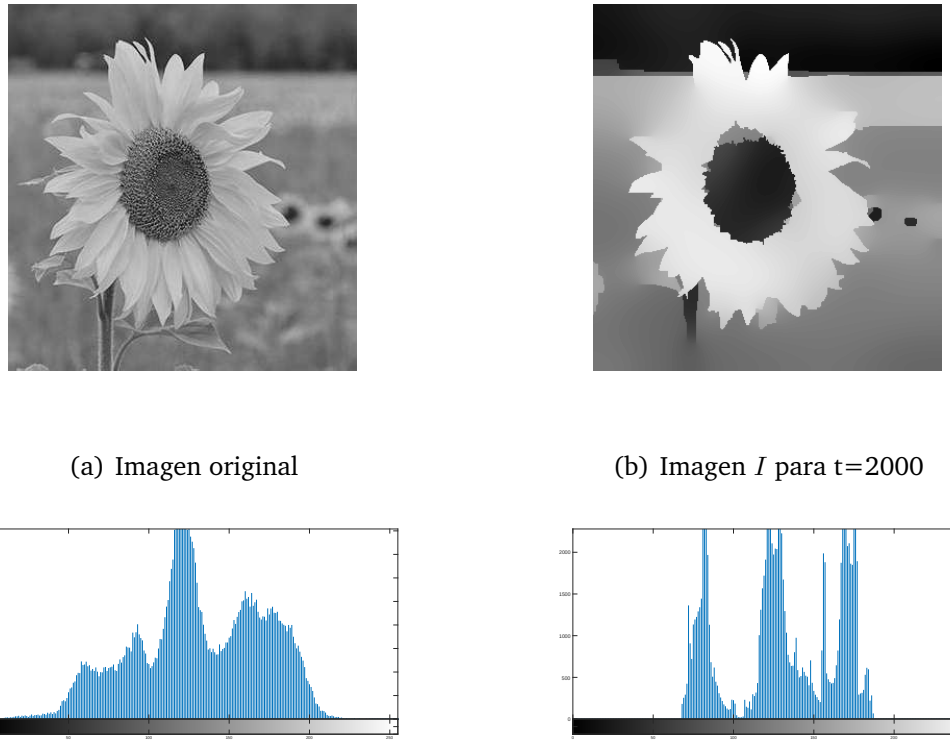


Figura 3.18: Entropía de Shannon para la difusión (Elaboración propia, 2019)

La imagen de evolución de la difusión de I_0 (Figura 3.19a) para $t = 2000$ se muestra en la Figura 3.19 (b), también se muestran los respectivos histogramas.



(a) Imagen original

(b) Imagen I para t=2000

(c) Histograma de imagen original

(d) Histograma de I para t=2000

Figura 3.19: Difusión anisotrópica e histograma (Elaboración propia, 2019)

3.6. Actividad de investigación 2: Segmentación automática

3.6.1. Introducción

La segmentación es la división de una imagen en regiones significativas. Aunque existen muchas técnicas de segmentación de imagen (Vantaram, 2012), éstas se pueden dividir en cuatros grupos:

1. Por bordes
2. Por umbralización
3. Por regiones
4. Por agrupamiento

Los métodos que se basan en la segmentación por bordes consideran que la mayor cantidad de información se encuentra en los bordes. Un borde es la frontera entre dos pixeles con valores de intensidad significativamente diferentes.

Las técnicas de umbralización se basan en el uso del histograma, estas técnicas son simples y se pueden clasificar en dos tipos: de un umbral o de umbralización múltiple.

En el caso de los métodos basados en regiones, los pixeles se agrupan de acuerdo con criterios de similitud y de proximidad espacial. Se pueden clasificar en dos grupos: de crecimiento y de división y fusión.

Las técnicas de agrupamiento agrupan los datos en clases con atributos comunes de forma autónoma. El algoritmo de *k-medias* es el método de agrupamiento más usado para segmentación, y se usa en este trabajo para realizar la segmentación automática.

3.6.2. Materiales y métodos

El orden de elementos activados en la interfaz gráfica de la Figura 3.4 para el caso de segmentación sin difusión anisotrópica fue: Botón 3 $\rightarrow$ Botón 8 $\rightarrow$ Control deslizante 9, los cuales llamaron las líneas de código 54–111, 603-628 y 1118-1125 respectivamente de

RegionesTrazos.m. El flujo de operaciones realizadas respecto del procesamiento general de la Figura 3.6 es *Abre imagen* $\rightarrow$ *Segmentación automática*.

Para el caso de segmentación con suavizado previo, el orden de activación de elementos fue: Botón 3 $\rightarrow$ Botón 11 $\rightarrow$ Botón 8 $\rightarrow$ Control deslizante 9. El flujo de operaciones realizadas respecto del procesamiento general de la Figura 3.6 es *Abre imagen* $\rightarrow$ *Suaviza* $\rightarrow$ *Segmentación automática*.

La ecuación 2.13 se aplicó utilizando la función *imsegkmeans* del toolbox de procesamiento de imagen de Matlab 2018b la cual aparece en la línea 614 de RegionesTrazos.m siendo el parámetro *Ngrupos* el número de grupos k a segmentar el cual se estableció usando el Control deslizante 9. Se repitió el agrupamiento tres veces para evitar el mínimo local con el parámetro *NumAttempts=3*. Las imágenes de entrada fueron RGB.

El algoritmo *k-medias* se aplicó a dos imágenes, una con pocos elementos y otra más compleja, se utilizaron valores para k de 2, 3 y 4, esto debido a que valores mayores no producen mejoras significativas.

A fin de evaluar la segmentación se usó la métrica del Índice Aleatorio Probabilístico (PRI) (Mújica-Vargas, 2013), un método subjetivo, que compara la segmentación automática con la segmentación manual y se describe a continuación.

Sea $G = I_1, .., I_m$ el conjunto de segmentos manuales y S la segmentación *k-medias*. El índice aleatorio probabilístico se calcula con

$$P(S, G) = \frac{2}{n(n-1)} \sum_{i,j;i<j} p_{i,j}^{c_{i,j}} (1 - p_{i,j})^{1-c_{i,j}} \tag{3.2}$$

donde n es el número de pixeles, $c_{i,j}$ está dada por

$$c_{i,j} = \begin{cases} 1 & \text{si } L_i^S = L_j^S \\ 0 & \text{si} L_i^S \neq L_j^S \end{cases} \tag{3.3}$$

siendo $p_{i,j}$ es el valor esperado de la distribución de Bernoulli para el par de pixeles dado por

$$p_{i,j} = \frac{1}{m} \sum_{k=1}^{m} T(i, j, k) \tag{3.4}$$

donde $I_k \in G$ y

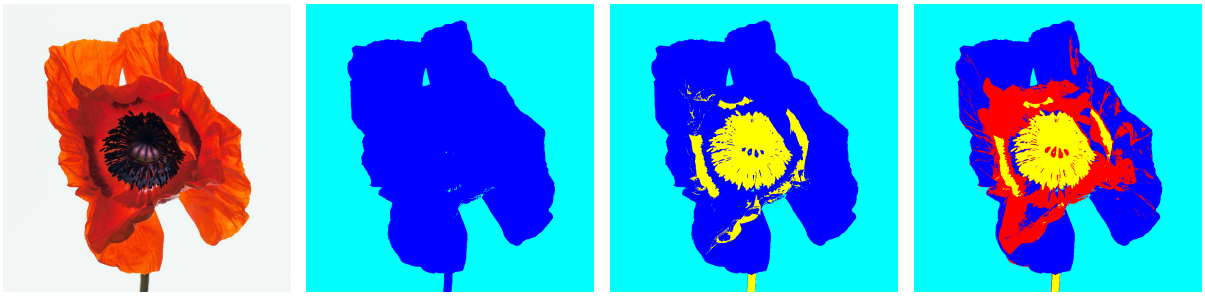
$$Ti,j,k = \begin{cases} 1 & \text{si } L_i^{I_k} = L_j^{I_k} \\ 0 & \text{si } L_i^{I_k} \neq L_j^{I_k} \end{cases} \tag{3.5}$$

El índice aleatorio está en el rango de [0,1], donde valores altos indican mayor similitud entre la imagen segmentada automáticamente y la imagen segmentada manualmente.

El código Matlab utilizado para calcular el PRI se encuentra en la página Web1 (2019).

3.6.3. Resultados

Usando el algoritmo *k-medias* se segmentó la imagen de la Figura 3.20(a), la cual tiene pocos elementos, obteniéndose dos regiones cuando se utilizó *k=2*, como se muestra en la Figura 3.20(b). Del mismo modo se aplicó $k = 3$ y $k = 4$, obteniéndose tres y cuatro regiones, como se observa en las Figuras 3.20(c) y 3.20(d), respectivamente.



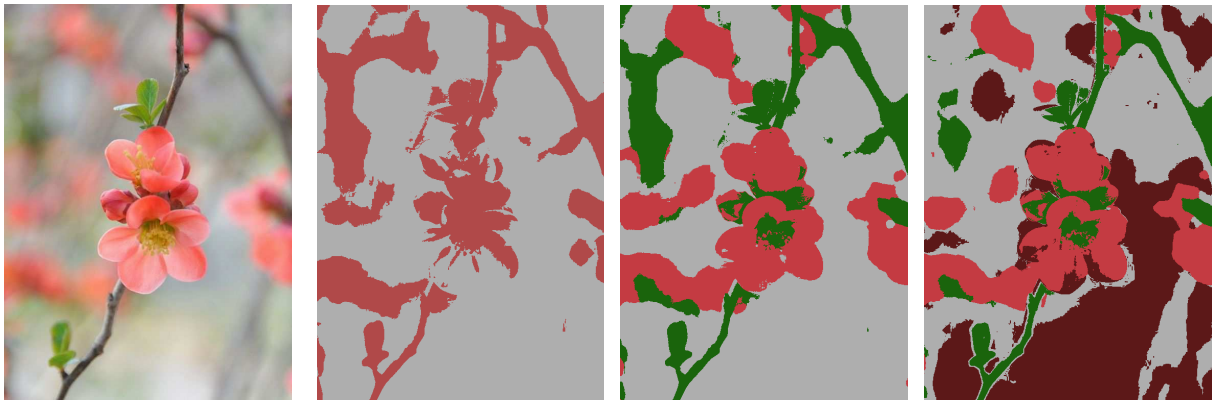
(a) Imagen original (b) k=2 (c) k=3 (d) k=4

Figura 3.20: Segmentación *k-medias* en imagen con pocos elementos (Elaboración propia, 2019)

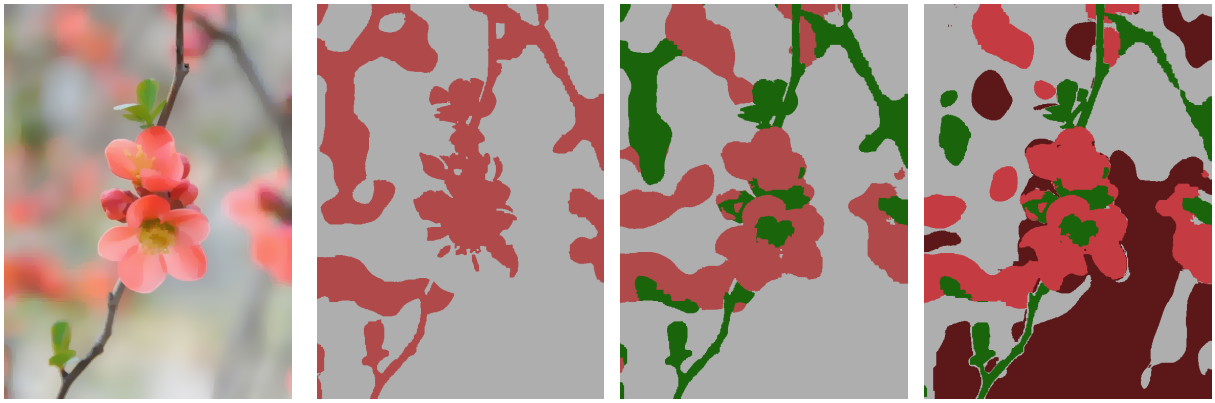
El algoritmo *k-medias* también se aplicó a la imagen de la Figura 3.21, la cual tiene más elementos que la imagen de la Figura 3.20. Al aplicar el algoritmo *k-medias* se utilizaron valores para k de 2, 3 y 4, y se obtuvieron los segmentos que se muestran en las imágenes de la Figura 3.21(b), (c) y (d), respectivamente

Para analizar el efecto de la difusión anisotrópica sobre la segmentación se aplicaron 200 iteraciones a la imagen de la Figura3.21(a), el resultado se muestra en la Figura 3.21(e), luego se aplicó el algoritmo *k-medias* utilizándose valores para k de 2, 3 y 4; los segmentos resultantes se muestran en las imágenes de la Figura 3.21(f), (g) y (h), respectivamente.

La segmentación se evaluó utilizando el Índice Aleatorio Probabilístico (PRI) dada por la ecuación 3.2, donde S son los segmentos resultantes de *k-medias* y G son los segmentos manuales mostrados en la Figura 3.21 (j),(k),(l). Al aplicar dicha ecuación se obtuvieron los valores PRI mostrados en la Figura 3.21 (b),(c),(d) y (f)(g)(h) para la segmentación *k-medias* de la imagen original y de la imagen suavizada, respectivamente.



(a) Imagen original (b) k=2, PRI=0.6241 (c) k=3, PRI=0.6238 (d) k=4, PRI=0.4967



(e) Imagen con difusión anisotrópica (f) k=2, PRI=0.6201 (g) k=3, PRI=0.6277 (h) k=4, PRI=0.5003

(i) Imagen original (j) k=2 (k) k=3 (l) k=4

Figura 3.21: Imagen con varias regiones, donde k indica el número de grupos, y PRI indica el índice aleatorio probabilístico, el cual evalúa la segmentación. (a)–(d) Segmentación *k-medias*, (e)–(h) Segmentación *k-medias*, aplicando previamente 200 iteraciones de difusión anisotrópica, (i)–(l) Segmentación manual de referencia utilizada para evaluar la segmentación *k-medias* (Elaboración propia, 2019)

3.7. Actividad de investigación 3: Segmentación interactiva

3.7.1. Introducción

La segmentación interactiva o semiautomática es diferente a la segmentación automática en que requiere de la ayuda de un operador humano para proveer información de alto nivel a fin de detectar y extraer con precisión objetos semánticos. Generalmente los operadores marcan puntos o áreas en la imagen que los algoritmos usan para realizar la segmentación de forma rápida y precisa. Los métodos interactivos, no son mejores unos de otros sino que se adaptan para resolver un problema particular de segmentación.

Algunos de los métodos interactivos destacados en la literatura son los siguientes: trazo poco preciso del contorno deseado (Terzopoulos, 1988), (Blake, Rother, Brown, Perez, y Torr, 2004), (J. Wang, Agrawala, y Cohen, 2007), marcado de partes del objeto o fondo (Bai y Sapiro, 2007), (Boykov y Funka-Lea, 2006), (Grady, 2006), uso de cuadro delimitador (Lempitsky, Kohli, Rother, y Sharp, 2009), (Rother, Kolmogorov, y Blake, 2004).

En esta tesis se utilizó una segmentación interactiva basada en el trazado poco preciso del contorno y refinando la precisión del contorno deseado.

Dos de los programas de edición de imagen, Photoshop y Corel Draw, utilizan para este propósito técnicas similares. En Corel se puede segmentar una imagen con la herramienta Forma, manipulando los nodos que envuelven un objeto y ajustando la curva entre los nodos mediante funciones Bézier cúbicas. En Photoshop se puede utilizar la herramienta Magnetic Lasso para segmentar un objeto seleccionando puntos sobre el contorno mediante clics, donde dicha herramienta busca la ruta global óptima desde un pixel de inicio hasta un pixel objetivo, los detalles del algoritmo usado se encuentran en Mortensen y Barrett (1995).

De este modo, buscando obtener resultados estéticos se decidió usar curvas Bézier para aproximar los contornos de los objetos a segmentar, para ello se seleccionaron puntos significativos sobre el contorno mediante clics.

3.7.2. Materiales y métodos

Para realizar la segmentación interactiva se utilizó la imagen de la manzana (Figura 2.7), la cual se usará en lo sucesivo para las demás actividades de investigación. El tamaño de la imagen fue de 564x549 pixeles.

El flujo de operaciones realizadas respecto del procesamiento general de la Figura 3.6 es *Abre imagen* $\rightarrow$ *Segmentación interactiva*. Es decir, una vez abierta la imagen se puede aplicar la segmentación interactiva.

La segmentación interactiva se realizó siguiendo el diagrama de la Figura 2.3, el cual se implementó en la interfaz gráfica con los elementos que contiene el Bloque 2 de la Figura 3.5. A continuación se transcriben los 4 pasos principales y luego se explica su metodología:

1. Seleccionar puntos consecutivos sobre el contorno de la región
2. Aplicar algoritmo de detección esquinas
3. Aproximar cada segmento del contorno usando curvas Bézier
4. Segmentar región

La selección de puntos, se efectuó activando el Botón 4 de la interfaz gráfica de la Figura 3.4 el cual llama las líneas de código 112-138 del código RegionesTrazos. Al terminar la selección se oprime cualquier tecla para salirse del ciclo.

Así, el paso 1 se concretó seleccionando puntos consecutivos sobre el contorno de la región de interés. Los puntos se seleccionaron de forma secuencial y el punto final coincidió con el primer punto para cerrar la región de interés. Los puntos que captaron más detalles se realizaron más cercanos unos de otros. También se dio clics sobre los puntos correspondientes a las esquinas del contorno.

El paso 2 se realizó automáticamente aplicando la ecuación 2.14 mediante las líneas 140 a 175 del Código 1 Matlab, estableciéndose una longitud de cuerda $k = 1$, y un umbral de prominencia de pico de 0.35 para la detección de esquinas.

Para realizar el paso 3, a cada segmento se asignó inicialmente una aproximación automática de curva Bézier de grado 2 y luego se refinó el ajuste manual de cada curva mediante los elementos 5 y 6 de la interfaz de usuario. Las líneas de código 175-420 del

Código 1 Matlab llevan a cabo la aproximación Bézier. Especialmente en las líneas de código mencionadas existen llamadas a la función Bezierm (línea 1216) la cual simula la ecuación 2.21.

En el paso 4 se utilizó el algoritmo de Bresenham 2.1 el cual se implementó en la línea 1250 del Código 1 Matlab, y se llama mediante el elemento 7 de la interfaz de usuario.

3.7.3. Resultados

Empleando la imagen de la manzana de la Figura 2.7 se marcaron 44 puntos consecutivos sobre el contorno, los cuales se muestran en la Figura 3.22 y el punto final se marcó lo más cerca posible del primer punto, ya que el código acepta una tolerancia de 4 pixeles de radio. Como se observa los puntos que captaron más detalles se realizaron más cercanos unos de otros.

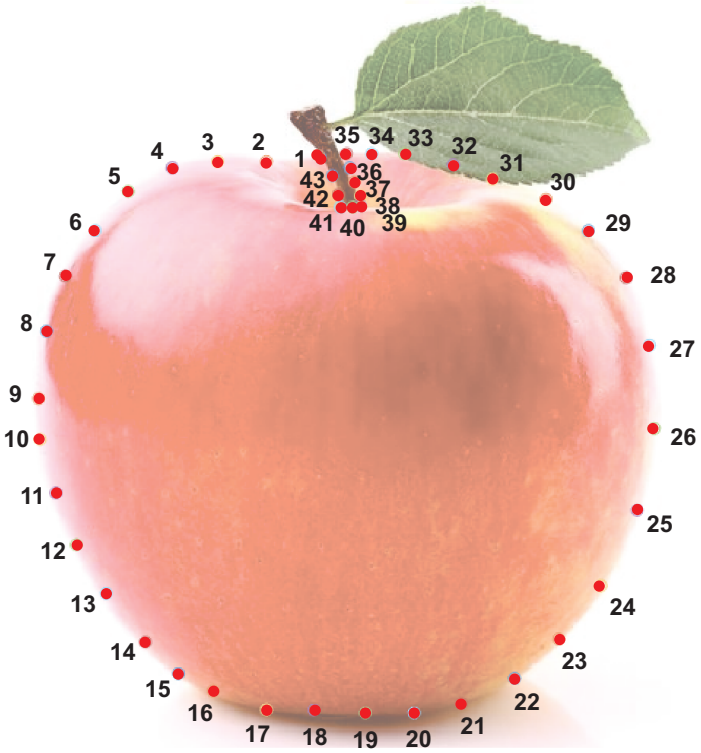
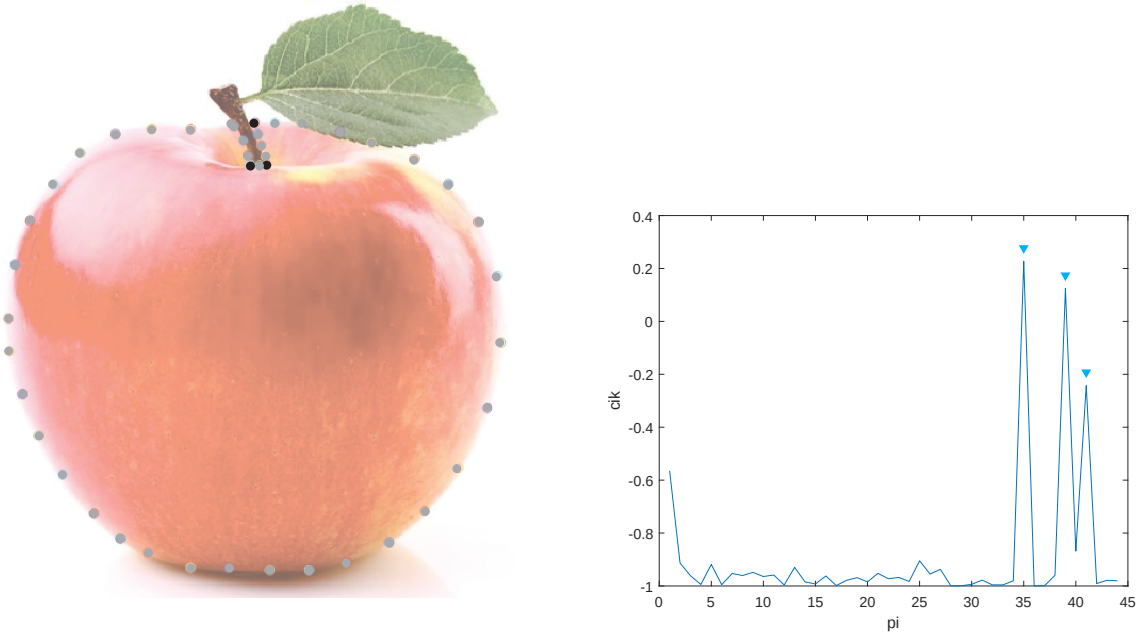


Figura 3.22: Selección de puntos consecutivos sobre el contorno (Elaboración propia, 2019)

Al aplicar el algoritmo de detección de esquinas al contorno muestreado se detectaron 3 esquinas con umbrales mayor a 0.35 y por tanto se obtuvieron 4 segmentos como se

muestra en la Figura 3.23.

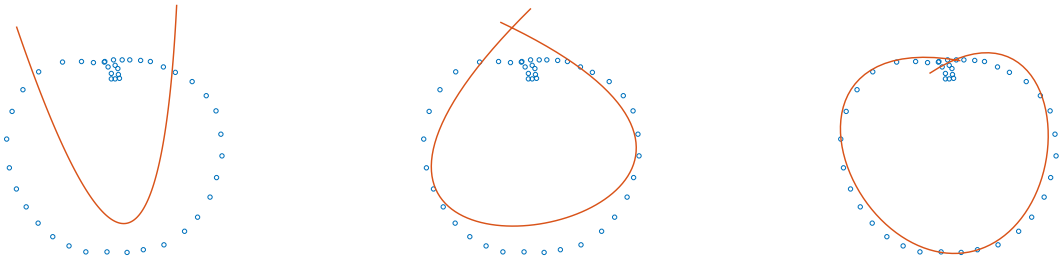


(a) Puntos con curvatura mayor a 0.35 donde se rompe el contorno dando lugar a 4 segmentos

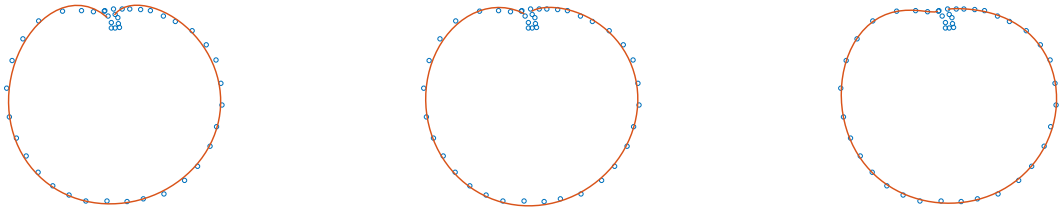
(b) Curvatura del contorno con $k = 1$

Figura 3.23: Segmentación del contorno mediante la detección de esquinas (Elaboración propia, 2019)

De los 4 segmentos detectados en la Figura 3.23, el grado de la curva Bézier se refinó manualmente sólo para el primer segmento como se observa en la Figura 3.24 ya que en los otros segmentos se dejó en $n = 2$.



(a) Grado 2 (b) Grado 3 (c) Grado 4



(d) Grado 5 (e) Grado 6 (f) Grado 9

Figura 3.24: Ajuste de curva Bézier de grado n para un segmento del contorno (Elaboración propia, 2019)

De este modo, la región encerrada por los 4 segmentos usando el algoritmo de Bresenham y luego el llenado de la región usando la función *imfill* de Matlab, se muestra en la Figura 3.25.

Figura 3.25: Resultado de la segmentación interactiva de región (Elaboración propia, 2019)

3.8. Actividad de investigación 4: Diseño automático de campo vectorial

3.8.1. Introducción

La idea de usar un campo vectorial que guíe la generación de las pinceladas viene del NPR (renderizado no fotorrealista) (Kyprianidis, 2013), una técnica computacional que busca crear versiones pictóricas de una imagen fuente sin pintar físicamente. Los métodos de renderizado basados en pinceladas tratan de lograr su objetivo mediante las propiedades de las pinceladas como son: el color, la textura, la posición, la densidad, el ancho y longitud, la orientación y el ordenamiento.

Algunas formas de abordar la orientación automática de los trazos en NPR basado en pinceladas son las siguientes. Haeberli (1990), Hertzmann (1998), Santella y DeCarlo (2002), Collomosse y Hall (2002), Orzan, Bousseau, Barla, y Thollot (2007), J. Lu, Sander, y Finkelstein (2010) usaron el campo vectorial perpendicular al gradiente de la imagen.

Shiraishi y Yamaguchi (2000) utilizaron, en cada pixel, los momentos de rectángulos vecinos para orientar las pinceladas. Por otro lado Litwinowicz (1997) y Hays y Essa (2004) interpolaron los gradientes más fuertes usando funciones de base radial.

En cuanto a los trabajos donde se aplica físicamente la pintura se puede mencionar a los siguientes. Lindemeier *et al.* (2015) implementaron campos tensoriales e interpolación del gradiente con métodos de difusión para guiar las pinceladas físicas, Luo *et al.* (2016) utilizaron trazos guiados por el gradiente, Scalera *et al.* (2018a) realizaron trazos aleatorios y Scalera *et al.* (2018b) pintaron acuarelas con trazos aleatorios basados en el gradiente.

Por tanto, en este trabajo se utilizó el modelo de Kang *et al.* (2009), el cual es una modificación del campo vectorial tangente al gradiente aplicando un filtrado bilateral que favorece los gradientes más fuertes.

3.8.2. Materiales y métodos

Para generar el campo vectorial utilizando la ecuación 2.22. se utilizó la imagen de la manzana de la Figura 2.7. El campo automático se implementó en la interfaz gráfica con los elementos que contiene el Bloque 4 de la Figura 3.5. Así, los elementos que forman dicho bloque son 15, 16 y 17 de la interfaz gráfica de la Figura 3.4. Las líneas del código Matlab del Anexo 1 activadas son 800-825.

Con el propósito de comparar el campo automático con el campo interactivo de esta tesis se aplicaron varias cantidades de iteraciones y tamaños de r (ecuación 2.23).

El campo vectorial resultante se visualizó convolucionando una imagen de ruido con cada línea del campo usando la técnica descrita por Cabral y Leedom (1993). El código Matlab correspondiente se encuentra en las líneas 1408-1514 del Anexo 1.

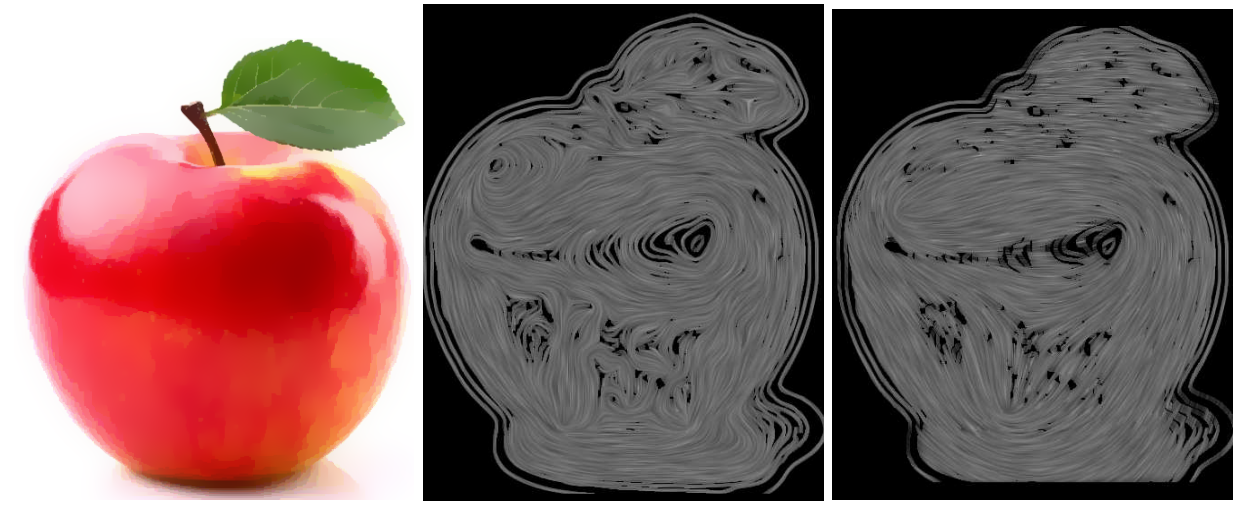
3.8.3. Resultados

Los pasos *Abre imagen* $\rightarrow$ *Campo automático* de la Figura 3.6 producen el campo mostrado en la Figura 3.26 para 7 iteraciones y $r = 7$. En este caso no se realizó ninguna operación previa a la imagen.

(a) Imagen de entrada (b) Iteración=7, r=7 (c) Iteración=30, r=20

Figura 3.26: Campo vectorial obtenido al seguir los pasos *Abre imagen* → *Campo automático* (Elaboración propia, 2019)

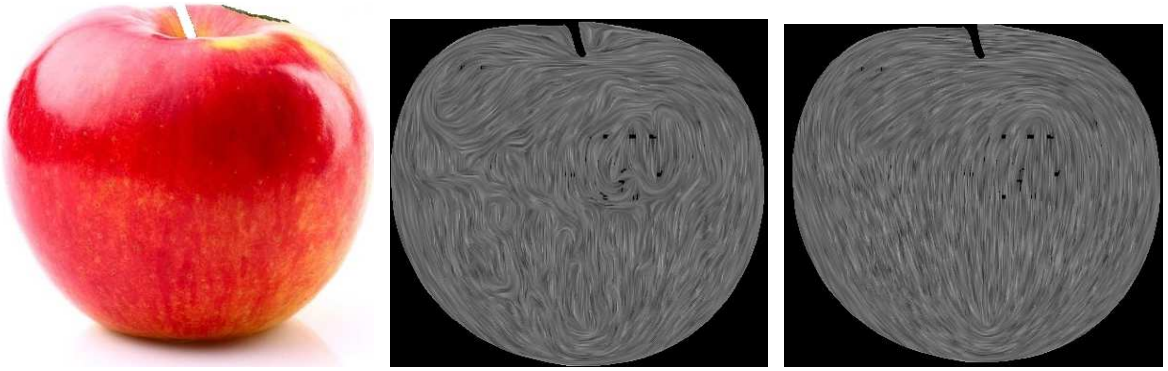
También se aplicaron 300 iteraciones de suavizado anisotrópico previo siguiendo los pasos *Abre imagen* → *Suaviza* → *Campo automático* de la Figura 3.6, produciéndose el campo mostrado en la Figura 3.27.



(a) Imagen de entrada (b) Iteración=7, r=7 (c) Iteración=30,r=20

Figura 3.27: Campo vectorial obtenido al seguir los pasos *Abre imagen* → *Suaviza* →*Campo automático* (Elaboración propia, 2019)

De forma similar se siguieron los pasos *Abre imagen* → *Segmentación interactiva* → *Campo automático* de la Figura 3.6, produciéndose el campo mostrado en la Figura 3.28.



(a) Imagen de entrada (b) Iteración=7, r=7 (c) Iteración=30, r=20

Figura 3.28: Campo vectorial obtenido al seguir los pasos *Abre imagen* → *Segmentación interactiva* →*Campo automático* (Elaboración propia, 2019)

La segmentación automática previa se utilizó sólo para crear los trazos ya que las regiones producidas por la segmentación pueden usar el mismo campo.

3.9. Actividad de investigación 5: Diseño interactivo de campo vectorial

3.9.1. Introducción

De los trabajos del NPR donde se emplea interacción del usuario para determinar la orientación de las pinceladas se pueden mencionar los siguientes ejemplos.

Haeberli (1990) permitió al usuario determinar la orientación de cada pincelada mediante la dirección del movimiento. Olsen, Maxwell, y Gooch (2005) diseñaron una interfaz gráfica de usuario para asignar mediante regiones el campo vectorial deseado, sea usando campos de fluido o interpolando el gradiente más fuerte mediante funciones de base radial.

En este trabajo se asignó mediante regiones el campo vectorial, interpolando en cada región el trazo manual de una o más curvas.

3.9.2. Materiales y métodos

Se introdujeron manualmente una o pocas líneas, siguiendo la textura de la región. Para realizar el diseño interactivo del campo se utilizó el bloque principal 5 de la Figura

3.5 que consta de los elementos 18, 19, 20 y 21 de la Figura 3.4, los cuales llaman a las líneas de código 468-493, 450-467, 520-575, 494-519 respectivamente del código Matlab del Anexo 1.

El procedimiento para llevar a cabo el diseño interactivo del campo se muestra en el diagrama de la Figura 2.9 que consta de los procesos: *Traza curva manual*, *Agrega curva*, *Borra curva* e *Interpola curva* los cuales se procesan con los elementos mencionados del bloque 5 de la Figura 3.5.

El proceso *Traza curva manual* traza y almacena cada curva mediante la función *drawfreehand(gca,'Closed',0,'Multiclick',true)*.

El proceso *Agrega curva* convierte la curva manual a curva Bézier de grado 4.

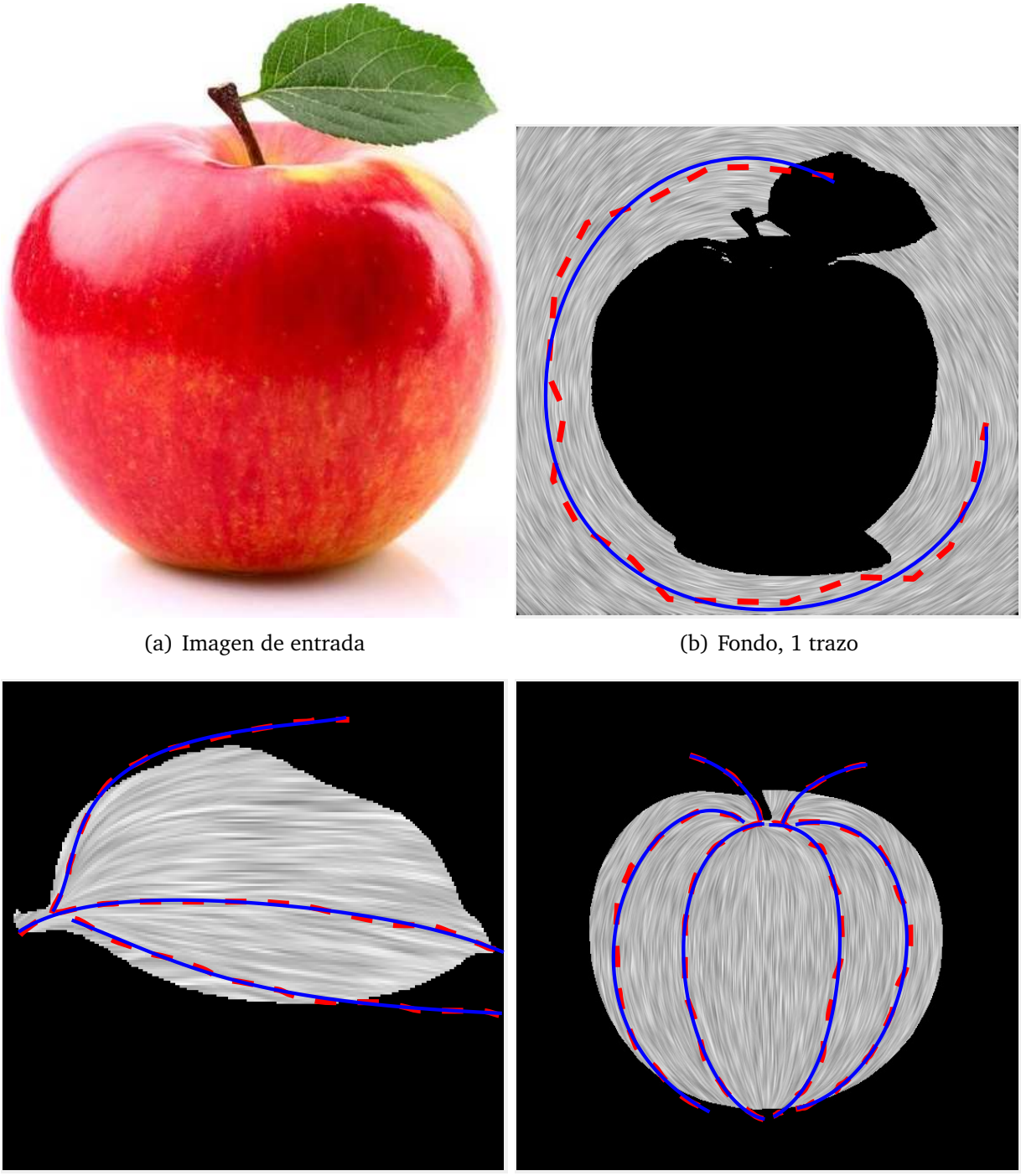
El proceso *Borra curva* borra la última curva trazada.

El proceso *Interpola curva* interpola la derivada de cada punto de la curva Bézier mediante la función interna Matlab *Fx = scatteredInterpolant (x, y, dx, 'natural', 'linear')*, donde *x,y* son las coordenadas de cada punto y *dx* es la derivada en ese punto. La opción *natural* es para la interpolación dentro de la envolvente convexa de los puntos y la opción *linear* es para extrapolar fuera de la envolvente convexa.

El campo vectorial resultante se visualizó convolucionando una imagen de ruido con cada línea del campo usando la técnica descrita por Cabral y Leedom (1993). El código Matlab correspondiente se encuentra en las líneas 1408-1514 del Anexo 1.

3.9.3. Resultados

En la Figura 3.29 se muestra el trazo manual de líneas sueltas las cuales se aproximaron a curvas Bézier de grado 4 y luego se interpolaron en las regiones de interés. La imagen de entrada se dividió en tres regiones. El fondo se segmentó de forma automática usando *k-medias*, mientras la hoja y la fruta se segmentaron de forma interactiva.



(a) Imagen de entrada

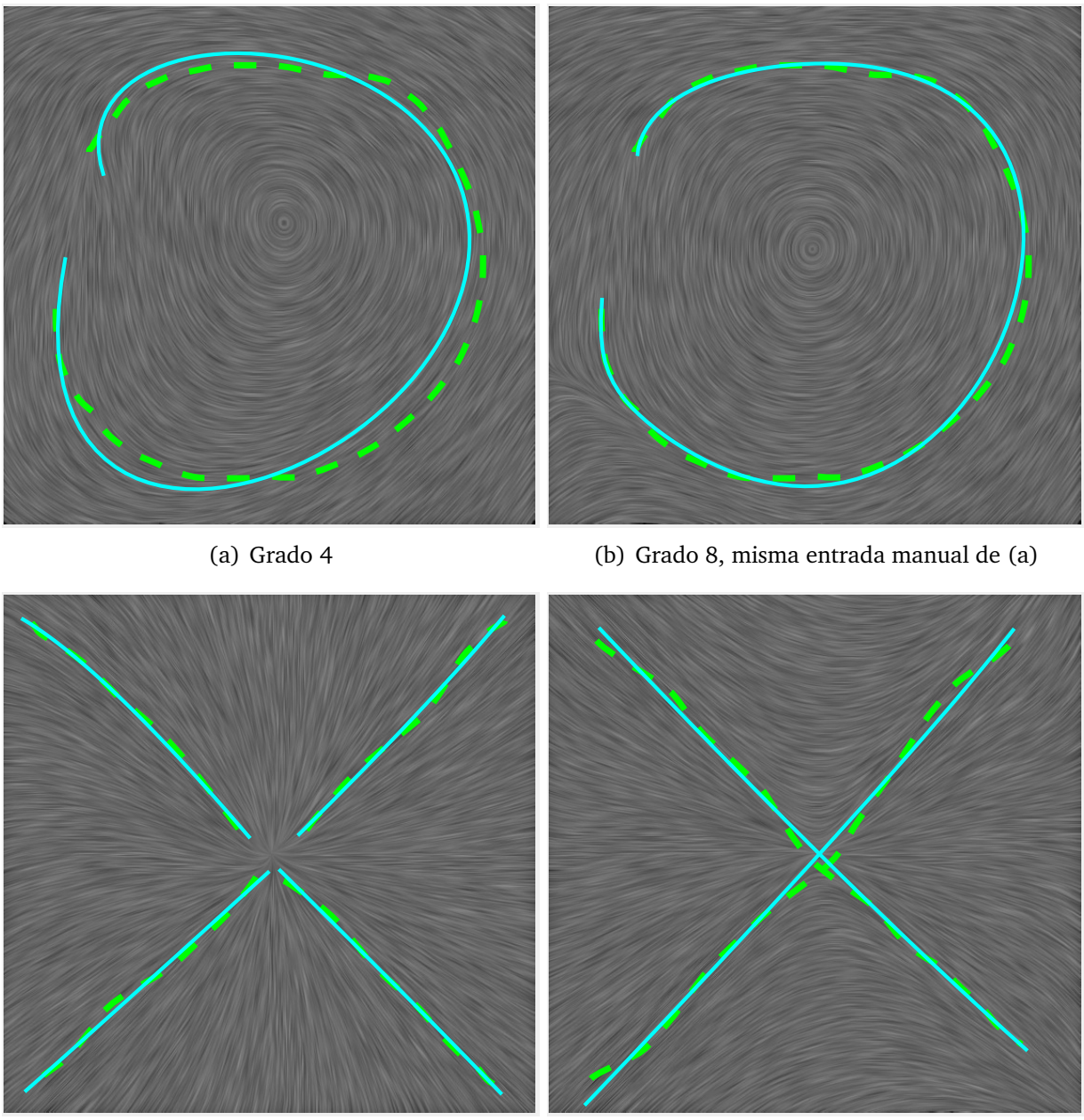
(b) Fondo, 1 trazo

(c) Hoja, 3 trazos

(d) Manzana, 6 trazos

Figura 3.29: Diseño interactivo del campo vectorial mediante interpolación. Trazo manual (- - - - -), aproximación Bézier de grado 4 (—) del trazo (Elaboración propia, 2019)

También se realizaron posibles diseños de campo para el fondo usando varios trazos y aproximaciones los cuales se muestran en la Figura 3.30.



(a) Grado 4

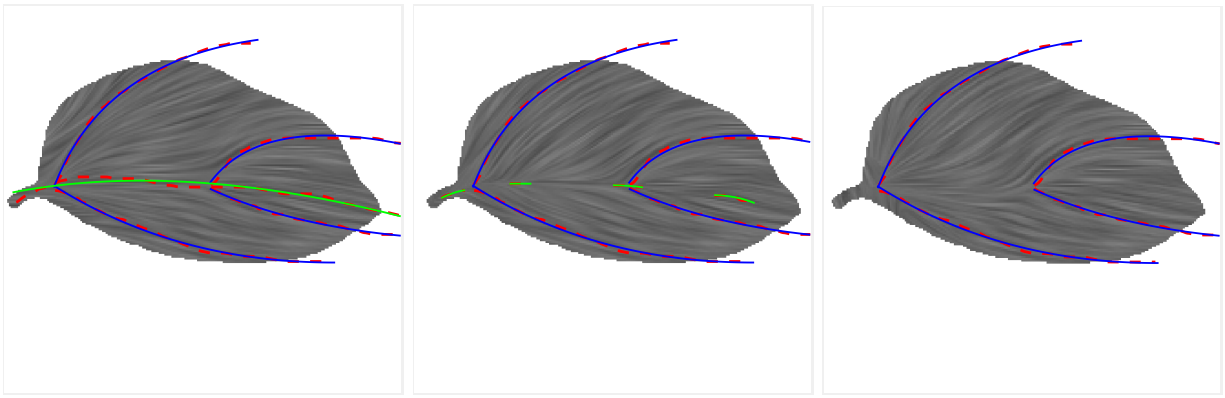
(b) Grado 8, misma entrada manual de (a)

(c) Grado 2

(d) Grado 2

Figura 3.30: Diseño para el fondo usando varios trazos y aproximaciones (trazo manual: línea punteada, aproximación Bézier: línea continua) (Elaboración propia, 2019)

Para investigar el comportamiento de las líneas de campo se trazaron las curvas que se muestran en la Figura 3.31. En especial se analizó la influencia sobre el campo causado por los trazos en la nervadura primaria de la hoja.



(a) Curva larga (b) Curvas cortas (c) Sin curva

Figura 3.31: Influencia de las curvas en la nervadura principal sobre el campo resultante (Elaboración propia, 2019)

3.10. Actividad de investigación 6: Generación de trazos

3.10.1. Introducción

Las estrategias para muestrear pinceladas semilla empleadas en la literatura son las siguientes.

Lindemeier *et al.* (2016) utilizó muestreo aleatorio, muestreo de cuadrícula (Hertzmann, 1998), muestreo basado en la densidad (Zhao y Zhu, 2011) y el método de relajación de Lloyd (Secord, 2002).

Scalera *et al.* (2018b) utilizaron patrones de líneas paralelas y muestreo aleatorio basado en el gradiente.

En esta tesis se utilizó un muestreo aleatorio donde para cada pincelada se actualiza el área restante hasta que se termina el área a pintar.

3.10.2. Materiales y métodos

Los elementos de la interfaz gráfica (Figura 3.4) necesarios para generar los trazos son: 24, 25, 26 y 27, los cuales emplean las líneas 631-774 del código Matlab del Anexo 1.

La generación de trazos se realizó por regiones aunque también se podría usar el campo creado por el gradiente. Así, del diagrama de flujo de la Figura 3.6 se pueden realizar las siguientes combinaciones de operaciones después de abrir la imagen de trabajo y antes de

crear los trazos:

- *Campo automático*
- *Suaviza* → *Campo automático*
- *Suaviza* → *Segmentación automática*→ *Campo automático*
- *Suaviza* → *Segmentación automática*→ *Campo interactivo*
- *Segmentación automática*→ *Campo automático*
- *Segmentación automática*→ *Campo interactivo*
- *Segmentación interactiva*→ *Campo automático*
- *Segmentación interactiva*→ *Campo interactivo*

La metodología para generar los trazos se muestra en el diagrama de flujo de la Figura 3.32, cuyos procesos se explican a continuación.

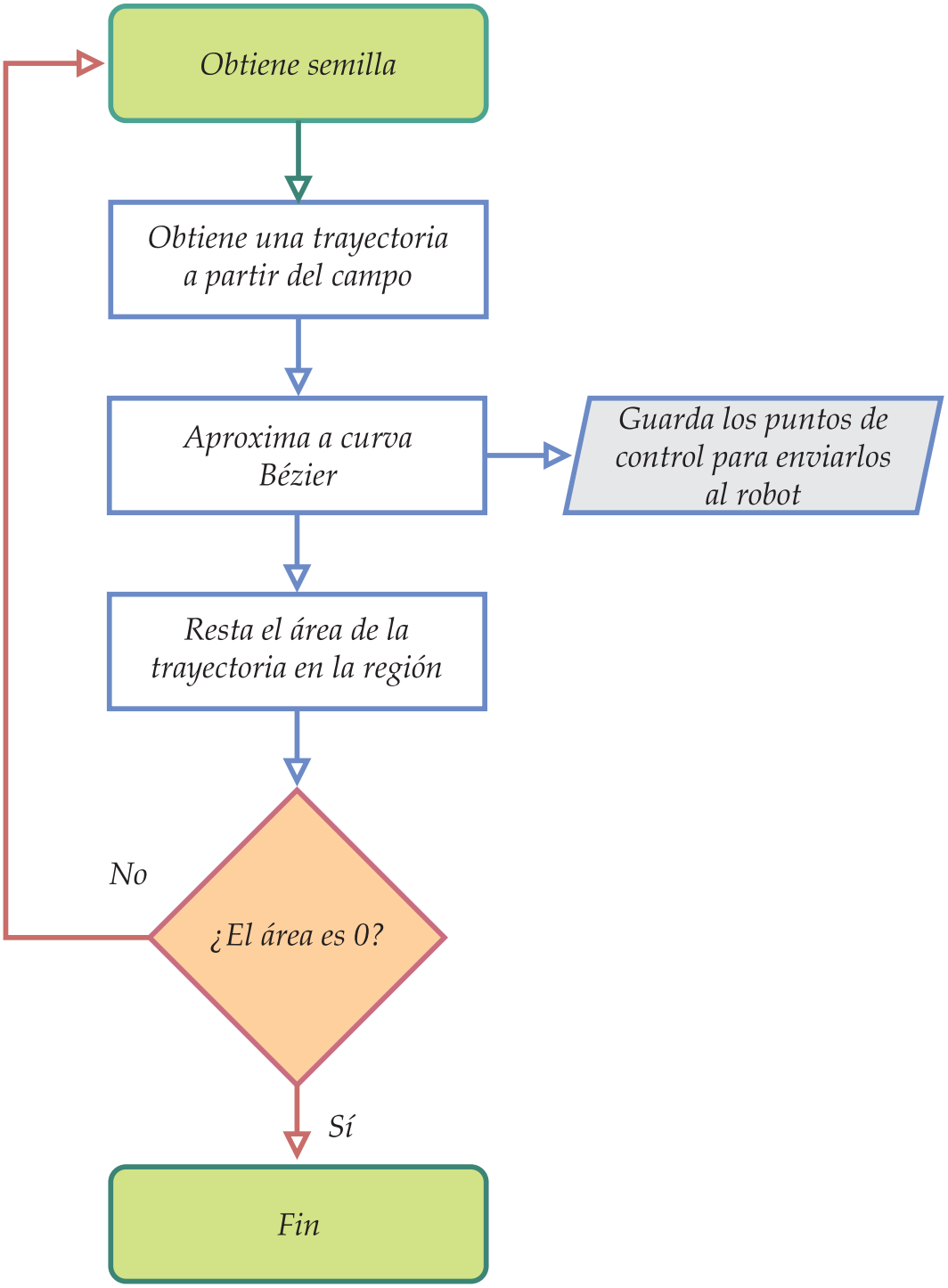


Figura 3.32: Metodología para generar los trazos (Elaboración propia, 2019)

La obtención de cada curva a partir del campo se realizó mediante el método LIC (Cabral y Leedom, 1993) (líneas 692-729 del código Matlab del Anexo 1).

Se utilizó un grado = 4 para la aproximación Bézier, por lo tanto se almacenaron 5 puntos de control para cada curva, los cuales se enviarán al robot, donde la tarjeta Arduino

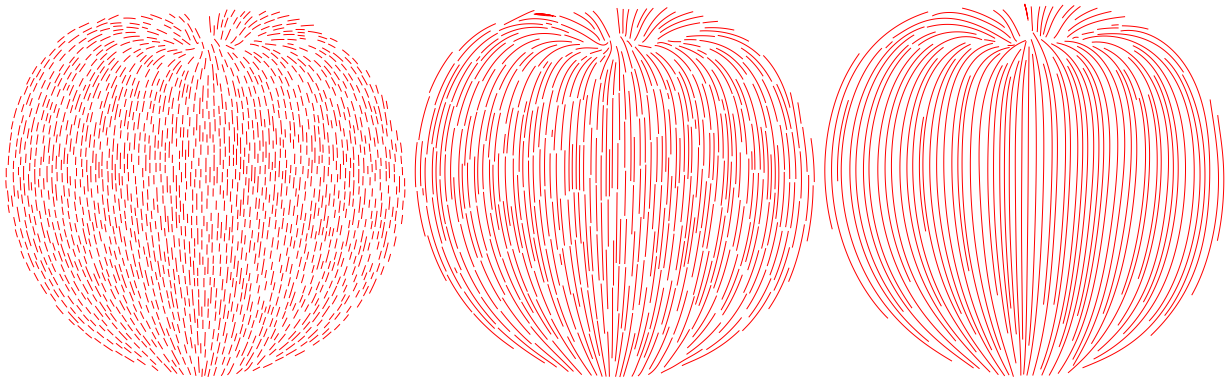
reconstruye la curva Bézier.

El ancho de cada trazo y su longitud se establecieron manualmente mediante los controles 26 y 27 de la interfaz gráfica (Figura 3.4).

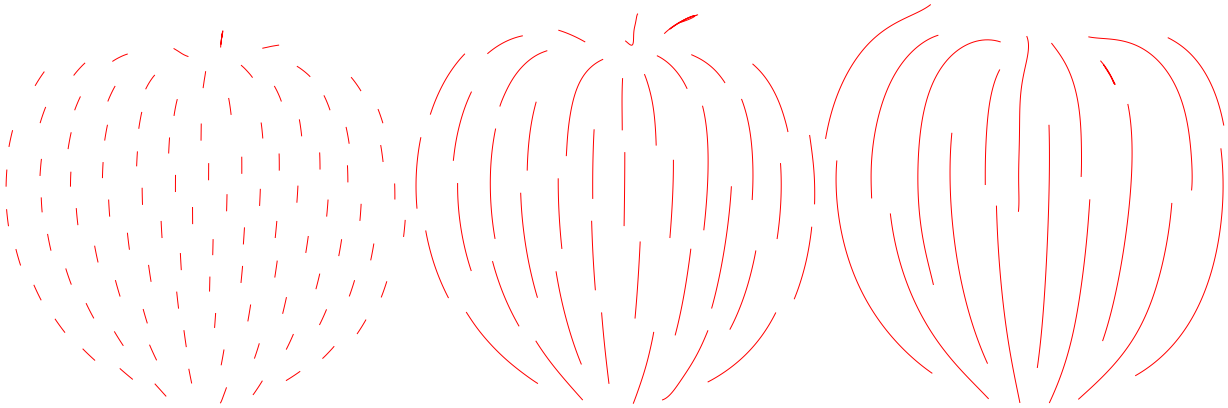
Para restar el área de la curva en la región de trabajo, se dilató la imagen binaria de la curva usando un estructurante circular del ancho del trazo establecido.

3.10.3. Resultados

Al utilizar las operaciones *Segmentación interactiva*→ *Campo interactivo*→ *Crear trazos* en la manzana (Figura2.7), se obtuvieron diferentes trayectorias para diferentes anchos de pincel como se muestra en la Figura 3.33. Una vez seleccionado el ancho del pincel también se generaron trayectorias de diversas longitudes como se observa en la misma figura.



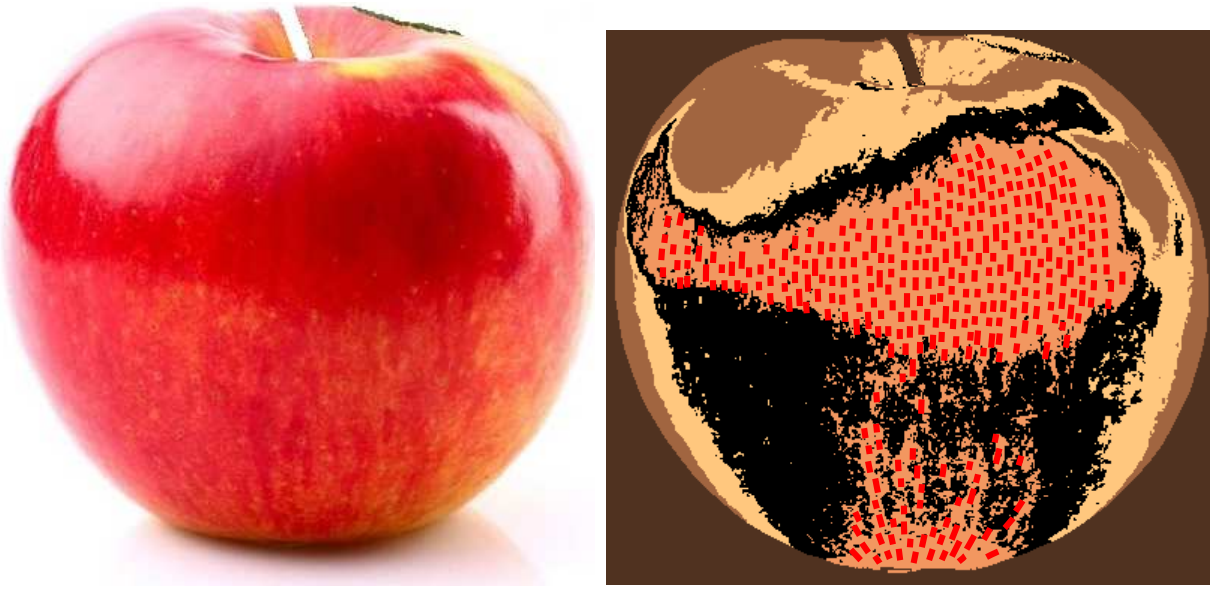
(a) Trazos cortos, pincel delgado (b) Trazos medianos, pincel delgado (c) Trazos largos, pincel delgado



(d) Trazos cortos, pincel grueso (e) Trazos medianos, pincel grueso (f) Trazos largos, pincel grueso

Figura 3.33: Generación de diversas trayectorias para pinceladas (Elaboración propia, 2019)

Los trazos también se aplicaron por subregiones como se muestra en la Figura 3.34 donde la región de entrada la cual se segmentó manualmente, se dividió en 5 regiones utilizando *k-medias* y se asignaron pinceladas sólo a una subregión.



(a) Región segmentada de forma interactiva

(b) 5 subregiones segmentadas con *k-medias*, con asignación de trazos a una subregión

Figura 3.34: Aplicación de trazos en subregión (Elaboración propia, 2019)

3.11. Actividad de investigación 7: Detección de bordes

3.11.1. Introducción

Esta actividad de investigación es independiente de las demás y puede o no aplicarse dependiendo del objeto a pintar.

El propósito de esta actividad es reafirmar ciertos detalles de un objeto, para dar un toque final a la pintura. También sirve para realizar un boceto, en el cual los contornos y los detalles no están definidos, sino insinuados de forma esquemática.

Para la detección de bordes se utiliza el operador de Canny (Canny, 1986) el cual usa un algoritmo óptimo de detección.

3.11.2. Materiales y métodos

El elemento de la interfaz gráfica (Figura 3.4) necesario para la detección de bordes es el 13 el cual emplea las líneas 916-1038 del código Matlab del Anexo 1.

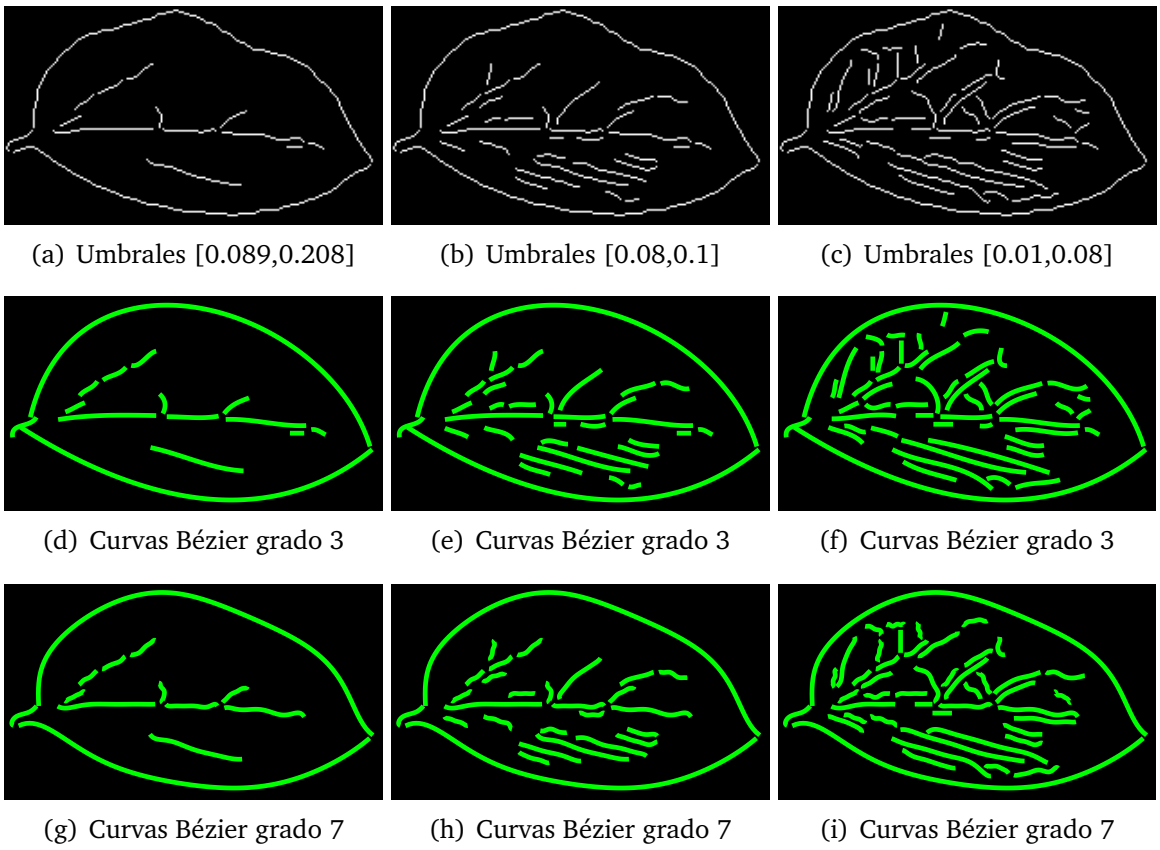
El detector de bordes de Canny utiliza el algoritmo siguiente: Aplica un filtrado gaussiano para disminuir el ruido. Obtiene la magnitud y la dirección del gradiente de la intensidad de la imagen. Aplica supresión de no máximos adelgazando el ancho de los bordes hasta obtener bordes de un pixel de ancho. Realiza umbralización con histéresis.

El detector de Canny se implementó con la función del Toolbox de Matlab: ***R=edge(I,'Canny', [UmbralMenor UmbralMayor], sigma);***, siendo el valor de la desviación estándar para el filtro gaussiano igual a 2.

Después de calcular los bordes mediante el algoritmo de Canny se eliminan intersecciones de ramas en el esqueleto binario, después se obtienen las esquinas de los bordes y se obtienen segmentos a partir de dichas esquinas y luego cada segmento se aproxima con curvas Bézier de grado n.

3.11.3. Resultados

Al utilizar las operaciones *Segmentación interactiva*$\rightarrow$ *Detección de bordes* en la hoja de la manzana (Figura 2.7), se obtuvieron los bordes mostrados en la Figura 3.35 para los umbrales indicados



(a) Umbrales [0.089,0.208] (b) Umbrales [0.08,0.1] (c) Umbrales [0.01,0.08]

(d) Curvas Bézier grado 3 (e) Curvas Bézier grado 3 (f) Curvas Bézier grado 3

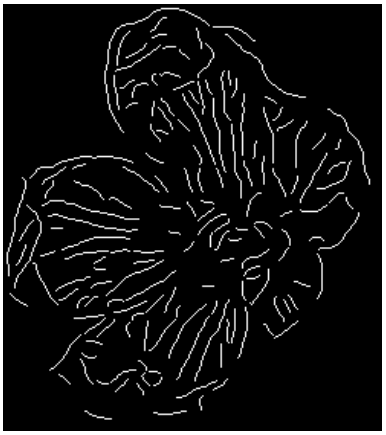
(g) Curvas Bézier grado 7 (h) Curvas Bézier grado 7 (i) Curvas Bézier grado 7

Figura 3.35: Detección de bordes (Elaboración propia, 2019)

También se aplicó la detección de bordes y aproximación Bézier en una región de interés segmentada interactivamente como se muestra en la Figura 3.36.

(a) Imagen de entrada

(b) Región de interés



(c) Detección de bordes

(d) Curvas Bézier de grado 5

Figura 3.36: Ejemplo de detección de bordes en una región de interés (Elaboración propia, 2019)

3.12. Actividad de investigación 8: Aplicación de pintura

3.12.1. Introducción

Las formas recientes de aplicar la pintura artística robótica según la revisión de literatura son las siguientes.

P. Tresset y Deussen (2014) aplicaron capas muy delgadas de pintura al acrílico, de forma que se transparenta la capa inferior, Lindemeier *et al.* (2015) pintaron capas de pintura al acrílico donde la orientación de las pinceladas se generaron mediante el método de relajación de Lloyd, Scalera *et al.* (2018b) realizaron pintura de acuarela usando tra-

zos aleatorios basados en el gradiente, Karimov *et al.* (2019) aplicaron pintura al acrílico mediante el gradiente de la imagen.

En este trabajo se aplicó pintura al acrílico guiada por trazos generados de forma interactiva. Donde un trazo manual se interpola en una región de trabajo para que todos los demás trazos estén alineados al trazo manual. Cada pincelada final se aproxima a una curva Bézier de grado $n < 20$ para obtener trazos con estilo artístico.

3.12.2. Materiales y métodos

Los parámetros establecidos para el trazo de cada pincelada fueron: ángulo recto del pincel respecto a la superficie de aplicación, longitud del trazo largo y corto, velocidad de 5cm/s, mezclas de pintura preparadas previamente.

En cuanto a los materiales, se utilizó pintura acrílica, pincel de pelo de marta y papel ilustración y se utilizaron cinco niveles de pintura en tono rojo.

Tomando como base el flujo de operaciones de la Figura 3.6 se realizaron las operaciones *Abre imagen* $\rightarrow$ *Segmentación interactiva* $\rightarrow$ *Campo interactivo* $\rightarrow$ *Crear trazos* $\rightarrow$ *Pintar región*.

De modo general, la Figura 3.37 muestra los pasos tomados para realizar la pintura completa de la manzana de la Figura 2.7.

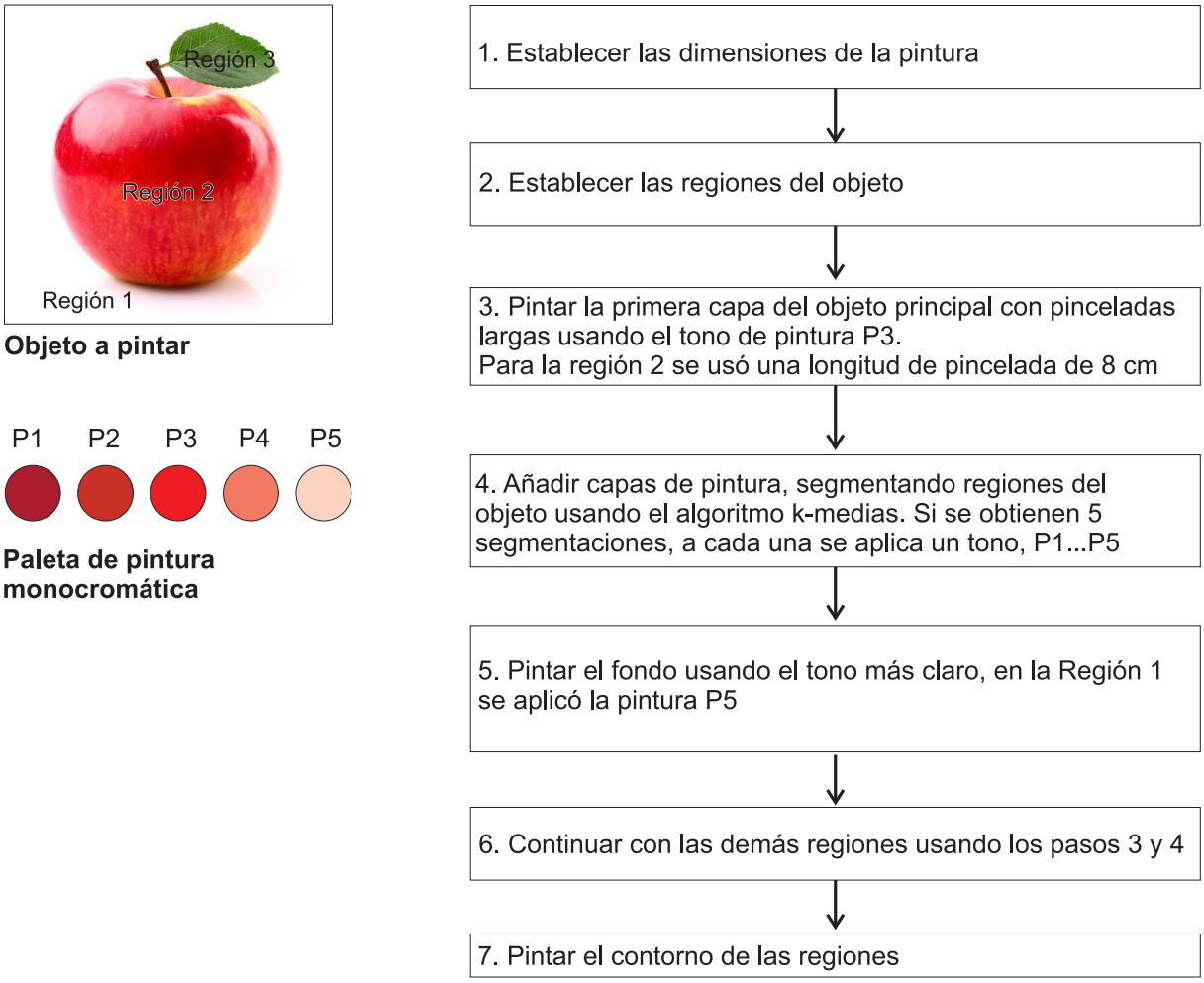
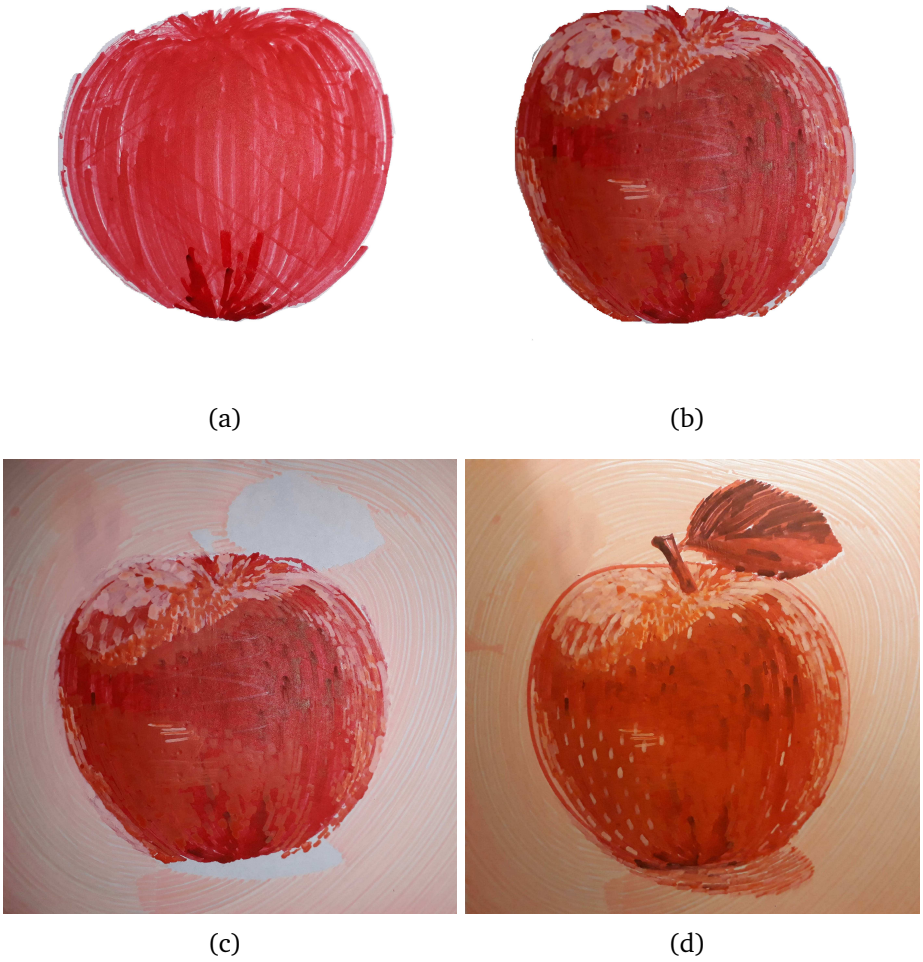


Figura 3.37: Metodología para aplicar la pintura (Elaboración propia, 2019)

3.12.3. Resultados

En la Figura 3.38 se muestra la aplicación de pintura acrílica siguiendo la metodología de la Figura 3.37, utilizando una paleta de pintura monocromática para las tres regiones de dicha figura. Se observa el la Figura 3.38 (a) la aplicación de la primera capa de pintura y en la Figura3.38 (b) la aplicación de capas de pintura por regiones usando los cinco tonos de pintura preestablecidas. En la Figura 3.38 (c) se muestra la aplicación de pintura en el fondo y en la Figura 3.38 (d) se muestra la pintura terminada.

La paleta de pintura se muestra en la Figura 3.39 (a) y la pintura terminada sobre el papel se observa en la Figura 3.39 (b).



(a) (b)

(c) (d)

Figura 3.38: Proceso de aplicación de pintura (Elaboración propia, 2019)

(a) Paleta de colores utilizados (b) Pintura robótica terminada

Figura 3.39: Pintura terminada (Elaboración propia, 2019)

3.13. Comparación con otros sistemas

En la Figura 3.40 se muestra la comparación con otros sistemas reportados en la bibliografía, donde se observa que el mapa de trazos propuesto en este trabajo (Figura 3.40 (g)) es más uniforme que los mapas de trazos propuestos por los autores (Figura 3.40 (f)).

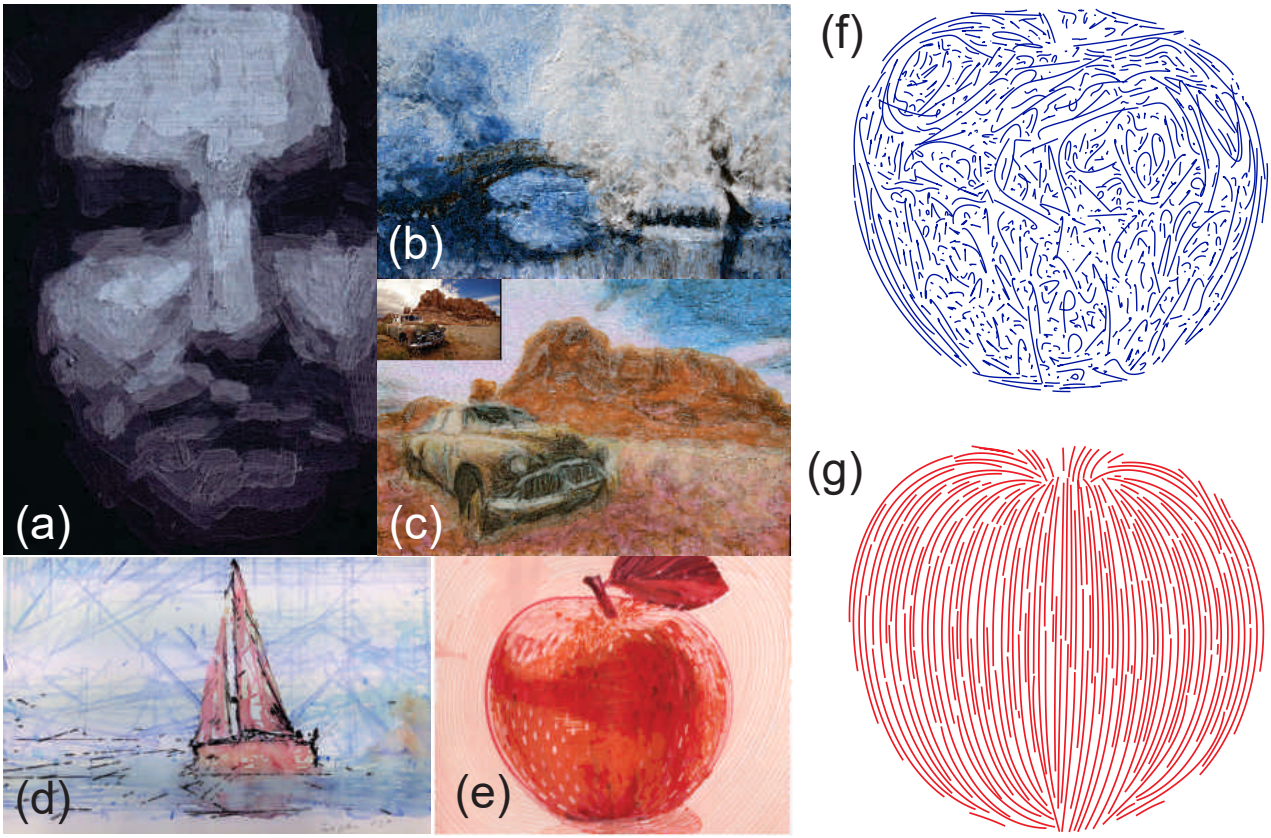


Figura 3.40: Comparación con otros sistemas, (a) P. Tresset y Deussen (2014), (b) Lindemeier *et al.* (2015), (c) Lindemeier *et al.* (2016), (d) Scalera *et al.* (2018b), (e) pintura propuesta, (f) mapa de trazos basado en el gradiente, (g) mapa de trazos propuesto (Elaboración propia, 2019)

Capítulo 4

Discusión, conclusiones y trabajos futuros

4.1. Discusiones, conclusiones y trabajos futuros de la actividad de investigación 1: Preparación de la imagen utilizando filtrado anisotrópico

Como se observa en la Figura 3.18 la entropía de Shannon disminuye poco cuando avanza t (iteraciones), esto es porque se suavizan las regiones pero tienden a permanecer los bordes.

Debido a que el objetivo de suavizar la imagen mediante la difusión es preparar la imagen para segmentarla, se espera que las regiones sean más definidas al aumentar la difusión. En el histograma del ejemplo de la Figura 3.19(d) se observan tres picos definidos, lo cual sugiere que en la imagen de la Figura 3.19(b) existen al menos tres posibles regiones.

Sin embargo queda a criterio del usuario si utiliza o no la preparación de la imagen mediante la difusión como paso previo a la segmentación automática como se muestra en la Figura 3.21.

4.2. Discusiones, conclusiones y trabajos futuros de la actividad de investigación 2: Segmentación automática

Cuando la imagen tiene pocos objetos la segmentación *k-medias* produce buenos resultados, por ejemplo en la Figura 3.20(b), al establecer k=2 dio como resultado dos regiones bien definidas: un fondo y un objeto principal. Este es un resultado satisfactorio, sin embargo también se habría obtenido si se hubiera aplicado umbralización, el cual es un método de segmentación más rápido que *k-medias* para imágenes en gris. Para imágenes en color *k-medias* funciona mejor. Del mismo modo, los resultados en la Figura 3.20(c) y (d) son aceptables, ya que las regiones concuerdan con el nivel de intensidad en la imagen original y se podrían utilizar para pintar.

Sin embargo, *k-medias* no produce resultados aceptables para imágenes con varios objetos, como se observa en las regiones obtenidas en la Figura 3.21. Al tomar como modelo de referencia la segmentación manual, Figura 3.21 (j),(k),(l), y al compararla con las segmentaciones producidas por la segmentación *k-medias*, Figura 3.21 (b),(c),(d), se observa la disminución del índice PRI al aumentar el número de clases k, indicando un alejamiento del modelo patrón. Y esto ocurre también para el caso de la difusión anisotrópica, Figura 3.21 (f),(g),(h).

Por tanto, considerando estos resultados y los revisados en la literatura para otras técnicas de segmentación, se comprueba que ningún algoritmo automático de segmentación produce la separación definida de regiones que pudieran utilizarse en la aplicación estética de pintura considerada en esta tesis, por lo que se propone el uso de la segmentación interactiva a fin de definir áreas específicas de pintado. Con todo, es posible que en alguna imagen sea de utilidad la segmentación automática y por esta razón se incluye en este trabajo.

4.3. Discusiones, conclusiones y trabajos futuros de la actividad de investigación 3: Segmentación interactiva

En la Figura 3.22, del punto 1 al 35 los espacios entre puntos son más abiertos que del punto 36 al 44. Esto permitió captar la curvatura y las esquinas, sin embargo, lo más

importante para que el algoritmo de detección de esquinas funcione correctamente es dar clic sobre las esquinas que el usuario desee captar. En cuanto a la longitud de la cuerda k esta deberá ser de 1 debido a que no se toma en cuenta la distancia entre cada punto muestreado.

Otro aspecto importante a considerar es que el número y la ubicación de los puntos seleccionados determinarán la forma del ajuste de la curva Bézier, por ejemplo si en lugar de seleccionar 44 puntos del contorno se decidiera marcar 20 o 10 puntos. O si dada la resolución de la imagen el usuario marcara los puntos de una forma más errática. Lo que sí es crucial es que los puntos, dada la restricción de las curvas de Bézier, no pueden marcarse hacia atrás (Figura 3.22), es decir, el orden de marcado 1,3,2,5,4 daría lugar a curvas indeseables.

Para valores altos del grado de la curva no existen cambios significativos, como se observa en la Figura 3.24(e-f) donde un grado 6 es aceptable visualmente aunque el grado 9 se ajusta más a los puntos de entrada. La pérdida de detalles de una curva, al final podría producir un trazo estético al momento de pintar.

En cuanto al tamaño de la imagen, este influirá en la segmentación ya que se toma dicho tamaño al momento de aplicar el algoritmo de Bresenham.

Por último considerar que la segmentación final puede contener objetos ajenos a la región. En la región de Figura 3.25 se observa un pequeño fragmento de hoja, sin embargo al momento de pintar todo se pintará como si fuera parte de la región principal.

4.4. Discusiones, conclusiones y trabajos futuros de la actividad de investigación 4: Diseño automático de campo vectorial

Como se muestra en la Figura 3.26 (b) y (c) el campo vectorial se alinea a los gradientes más fuertes al aumentar el número de iteraciones y al aumentar el tamaño de la ventana de integración. También se observan áreas negras donde el gradiente es nulo y por lo tanto no se producirían trazos.

Cuando se aplica suavizado de difusión anisotrópica se produce un campo vectorial

más suavizado pero no se producen mejoras en cuanto al sentido de los trazos.

También se puede utilizar el campo vectorial sólo en regiones de interés como se muestra en la Figura 3.28.

En términos generales el campo producido automáticamente es sensible a la iluminación y por tanto los trazos que produciría no serían estéticos al pintar, aunque esta metodología con sus variantes es la que se usa actualmente. Esto motivó el diseño interactivo del campo expuesto a continuación.

4.5. Discusiones, conclusiones y trabajos futuros de la actividad de investigación 5: Diseño interactivo de campo vectorial

El campo resultante en una región, como las mostradas en la Figura 3.29 depende de la cantidad de curvas, del grado de aproximación Bézier, de la longitud de las curvas y de su ubicación.

Así, la curva Bézier de la Figura 3.30(b) logró un ajuste mejor al trazo manual que el de la Figura 3.30(a).

En cuanto al número de curvas en la Figura 3.29(d) se trazaron más curvas, lo que permite un mejor ajuste a la forma de la textura de la manzana.

En la Figura 3.30(c) y (d) se buscaba obtener el mismo resultado, pero debido a la interpolación triangular sólo en (c) se obtuvo la forma de singularidad tipo nodo.

Al investigar la interpolación triangular en la Figura 3.31 se observó que el efecto del trazo largo central produce una mayor alineación no deseada (Figura 3.31(a)), mientras que el efecto de los trazos cortos centrales (Figura 3.31(b)) logran alinear el campo con la textura de la nervadura.

Por lo tanto, el diseño interactivo del campo vectorial permite configurar varios parámetros de entrada los cuales producen resultados más personalizables que el diseño automático, por ejemplo si se comparan con los producidos en la Figura 3.26.

4.6. Discusiones, conclusiones y trabajos futuros de la actividad de investigación 6: Generación de trazos

La combinación de la segmentación interactiva con la segmentación automática permite completar el proceso de pintado de un objeto completo. Para ello se utilizan diversas longitudes de pincelada y varias regiones de un mismo objeto obtenidas de forma automática o interactiva.

En este caso la generación de trazos no tiene realimentación, lo cual permitiría que las pinceladas fueran dinámicas. Es decir, si alguna pincelada no cubrió el área deseada, se volvería a trazar o se generarían los trazos conforme se avanza el trabajo de pintura. Esto requiere un sistema de video que capte y procese la aplicación de pintura.

Otros aspectos interesantes de investigar son, otras formas de segmentación automática que produzcan subregiones para aplicación de pintura y otras estrategias para muestrear las pinceladas semilla.

4.7. Discusiones, conclusiones y trabajos futuros de la actividad de investigación 7: Detección de bordes

La detección de bordes permite obtener curvas simples de un objeto las cuales se pueden pintar sobre una hoja de papel o sobre un objeto de cerámica. Mediante las herramientas diseñadas en la interfaz gráfica Matlab se pueden realizar operaciones en regiones de interés de una imagen de entrada, logrando de este modo, el objetivo planteado en esta tesis.

Esta actividad de investigación se registró ante el Instituto Nacional del Derecho de Autor, el certificado se presenta en el Anexo 3.

4.8. Discusiones, conclusiones y trabajos futuros de la actividad de investigación 8: Aplicación de pintura

La pintura acrílica, la cual se diluye con agua, tiene un tiempo de secado más rápido que la pintura de aceite, lo que permite agregar capas de pintura en períodos relativamente cortos. Sin embargo se observó que el cartón empleado se deformó por la humedad, lo cual ocasionó que el pincel no alcanzara algunas zonas o se sobrepintara, al momento de aplicar las capas secundarias.

La aplicación de pintura sobre cerámica, no presentaría este problema debido a su dureza. Tampoco se presentaría este problema en un lienzo de tela.

Una de las ventajas de la aplicación interactiva de pintura es que el usuario decide cuándo se detiene la aplicación de pintura, ya que se puede continuar refinando la pintura mediante el empleo de más capas de pintado.

En trabajos a futuro, se considera, la aplicación de más capas de pintura, el empleo de una cámara de video para que el sistema realice realimentación y la transferencia de las técnicas de este trabajo a artesanía cerámica.

En el Anexo 4 se muestra un artículo publicado con los resultados de las actividades de investigación.

Bibliografía

Abe, K., Morii, R., Nishida, K., y Kadonaga, T. (1993). Comparison of methods for detecting corner points from digital curves-a preliminary report. En *Proceedings of 2nd international conference on document analysis and recognition (icdar '93)* (p. 854-857).

Aguilar, C., y Lipson, H. (2008). A robotic system for interpreting images into painted artwork. En *International conference on generative art* (Vol. 11).

Alonso, M., y Finn, E. J. (1967). *Fundamental university physics* (Vol. 2). Addison-Wesley.

Amidror, I. (2002). Scattered data interpolation methods for electronic imaging systems: a survey. *J. Electronic Imaging*, *11*(2), 157–176.

Asada, H., y Brady, M. (1986). The curvature primal sketch. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *8*, 2–14.

Attneave, F. (1954). Some informational aspects of visual perception. *Psychol. Rev*, 183–193.

Ayala, A. (2015). *La producción artesanal alfarera en metepec, estado de méxico: una visión sustentable* (Tesis de maestría). Universidad Autónoma del Estado de México.

Bai, X., y Sapiro, G. (2007). A geodesic framework for fast interactive image and video segmentation and matting. En *2007 ieee 11th international conference on computer vision* (pp. 1–8).

Biswas, S., y Lovell, B. C. (2007). *Bezier and splines in image processing and machine vision* (1.ª ed.). Springer Publishing Company, Incorporated.

Blake, A., Rother, C., Brown, M., Perez, P., y Torr, P. (2004). Interactive image segmentation using an adaptive gmmrf model. En *Computer vision - eccv 2004* (Vol. 10.1007/b97865, pp. 428–441). Springer Berlin Heidelberg.

Boykov, Y., y Funka-Lea, G. (2006). Graph cuts and efficient n-d image segmentation.

International Journal of Computer Vision, *70*(2), 109–131.

Bresenham, J. E. (1965). Algorithm for computer control of a digital plotter. *IBM Systems Journal*, *4*, 25–30.

Bunge, M. (2001). *Diccionario de filosofía*. Siglo XXI, México.

Cabral, B., y Leedom, L. C. (1993). Imaging vector fields using line integral convolution. En *Proceedings of the 20th annual conference on computer graphics and interactive techniques* (pp. 263–270). New York, NY, USA: ACM.

Cabrelli, C. A., y Molter, U. M. (1990). Automatic representation of binary images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(12), 1190-1196.

Calinon, S., Epiney, J., y Billard, A. (2005). A humanoid robot drawing human portraits. En *5th ieee-ras international conference on humanoid robots, 2005.* (pp. 161–166).

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*, 679–698.

Castaños Alés, E. (2000). *Los orígenes del arte cibernético en españa : el seminario de generación automática de formas plásticas del centro de cálculo de la universidad de madrid : (1968-1973)* (Tesis Doctoral no publicada). Alicante : Biblioteca Virtual Miguel de Cervantes, 2000.

Chao, F., Chen, F., Shen, Y., He, W., Sun, Y., Wang, Z., ... Jiang, M. (2014). Robotic free writing of chinese characters via human–robot interactions. *International Journal of Humanoid Robotics*, *11*(01), 1450007.

Cohen, H. (1988). How to draw three people in a botanical garden. En *Aaai* (Vol. 89, pp. 846–855).

Cohen, H. (1995). The further exploits of aaron, painter. *Stanford Hum. Rev.*, *4*(2), 141–158.

Collomosse, J., y Hall, P. (2002). Painterly rendering using image salience. En *Proceedings 20th eurographics UK conference.* IEEE Comput. Soc.

Coristine, M., y Stein, M. R. (2004). Design of a new pumapaint interface and its use in one year of operation. En *Robotics and automation, 2004. proceedings. icra'04. 2004 ieee international conference on* (Vol. 1, pp. 511–516).

Díaz-Bautista, J. A. (2006). *Programas del fonart y desarrollo tecnológico en artesanías* (Tesis de maestría). Instituto Politécnico Nacional.

Deriche, R., y Faugeras, O. (1990). 2-d curve matching using high curvature points: application to stereo vision. En *[1990] proceedings. 10th international conference on pattern recognition* (Vol. 1, p. 240-242).

Deussen, O., Lindemeier, T., Pirk, S., y Tautzenberger, M. (2012). Feedback-guided stroke placement for a painting machine. En *Proceedings of the eighth annual symposium on computational aesthetics in graphics, visualization, and imaging* (pp. 25–33).

Dong, X.-l., Li, W.-j., Ning, X., Zhang, L.-p., y Lu, Y.-x. (2018). Stylized portrait generation and intelligent drawing of portrait rendering robot. *DEStech Transactions on Engineering and Technology Research*(icmeit).

Dávila, I. N. (2017). *Identificación del perfil del consumidor de artesanías mexicanas para la generación de bases que contribuyan a la creación de estrategias de marketing* (Tesis de maestría). Instituto Politécnico Nacional.

Farin, G. E. (1996). *Curves and surfaces for computer-aided geometric design: A practical code* (4th ed.). Orlando, FL, USA: Academic Press, Inc.

Franke, R. (1979). *A critical comparison of some methods for interpolation of scattered data.* Calhoun. Descargado de `https://calhoun.nps.edu/handle/10945/35052`

Gomez, M. (2010). *Difusión de la artesanía michoacana a través del diseño editorial* (Tesis de licenciatura). Universidad Don Vasco A.C.

Gonzalez, R. C., Woods, R. E., y Eddins, S. L. (2009). *Digital image processing using matlab*. Gatesmark Publishing.

Grady, L. (2006). Random walks for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, *28*(11), 1768–1783.

Gurpinar, C., Alasag, T., y Kose, H. (2012). Humanoid robot painter assisted by a human. En *2012 ieee workshop on advanced robotics and its social impacts (arso)* (pp. 79–82).

Haeberli, P. (1990). Paint by numbers: abstract image representations. En *Proceedings of the 17th annual conference on computer graphics and interactive techniques - SIGGRAPH '90*. ACM Press.

Hays, J., y Essa, I. (2004). Image and video based painterly animation. En *Proceedings of the 3rd international symposium on non-photorealistic animation and rendering* (p. 113-120).

Hernández-Aguilar, C. (2018). Transdisciplinary methodological option for initial research

process: Training of researchers. *Transdisciplinary Journal of Engineering and Science*, *9*, 157-181.

Hernández-Ramírez, V., Pineda-Domínguez, G., y Andrade-Vallejo, M. A. (2011). Las mipymes artesanales como un medio de desarrollo para los grupos rurales en méxico. *Universidad y empresa*, *13*(21), 65–92.

Hertzmann, A. (1998). Painterly rendering with curved brush strokes of multiple sizes. En *Proceedings of the 25th annual conference on computer graphics and interactive techniques - SIGGRAPH '98*. ACM Press.

Huang, X., Bi, S., Dong, M., Chen, H., Fang, S., y Xi, N. (2016). Automatic feature extraction and optimal path planning for robotic drawing. En *Cyber technology in automation, control, and intelligent systems (cyber), 2016 ieee international conference on* (pp. 19–24).

Jain, S., Gupta, P., Kumar, V., y Sharma, K. (2015). A force-controlled portrait drawing robot. En *2015 ieee international conference on industrial technology (icit)* (pp. 3160–3165).

Jean-Pierre, G., y Saïd, Z. (2012). The artist robot: a robot drawing like a human artist. En *Industrial technology (icit), 2012 ieee international conference on* (pp. 486–491).

Kang, H., Seungyong, L., y Chui, C. K. (2009). Flow-based image abstraction. *IEEE Transactions on Visualization and Computer Graphics*, *15*, 0–76.

Karimov, A. I., Kopets, E. E., Rybin, V. G., Leonov, S. V., Voroshilova, A. I., y Butusov, D. N. (2019). Advanced tone rendition technique for a painting robot. *Robotics and Autonomous Systems*, *115*, 17 - 27.

Koenderink, J. (1984). The structure of images. *Biological Cybernetics*, *50*, 363–370.

Kudoh, S., Ogawara, K., Ruchanurucks, M., y Ikeuchi, K. (2007). Painter robot: Manipulation of paintbrush by force and visual feedback. En *Ieee/rsj intl. conf. on intelligent robots and systems (iros) workshop.ªrt and robots* (pp. 63–68).

Kudoh, S., Ogawara, K., Ruchanurucks, M., y Ikeuchi, K. (2009). Painting robot with multi-fingered hands and stereo vision. *Robotics and Autonomous Systems*, *57*(3), 279–288.

Kyprianidis, J. W. T. I.-T., Jan Eric; Collomosse. (2013). State of the 'art': A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization*

and Computer Graphics, *19*, 866-885.

Lau, M. C., Cheng, C.-T., Baltes, J., y Anderson, J. (2015). Drawing pressure estimation using torque feedback control model of a 4-dof robotic arm. En *Robot intelligence technology and applications 3* (pp. 401–410). Springer.

Lee, E. T. Y. (1989). Choosing nodes in parametric curve interpolation. *Comput. Aided Des.*, *21*(6), 363–370.

Lempitsky, V., Kohli, P., Rother, C., y Sharp, T. (2009). Image segmentation with a bounding box prior. En *2009 ieee 12th international conference on computer vision* (pp. 277–284).

Lin, C.-Y., Chuang, L.-W., y Mac, T. T. (2009). Human portrait generation system for robot arm drawing. En *2009 ieee/asme international conference on advanced intelligent mechatronics* (pp. 1757–1762).

Lin, C.-Y., Mac, T. T., y Chuang, L.-W. (2009). Real-time artistic human face portrait by humanoid robot. En *2009 ieee control applications,(cca) & intelligent control,(isic)* (pp. 205–210).

Lindemeier, T., Metzner, J., Pollak, L., y Deussen, O. (2015). Hardware-based non-photorealistic rendering using a painting robot. En *Computer graphics forum* (Vol. 34, pp. 311–323).

Lindemeier, T., Pirk, S., y Deussen, O. (2013). Image stylization with a painting machine using semantic hints. *Computers & Graphics*, *37*(5), 293–301.

Lindemeier, T., Spicker, M., y Deussen, O. (2016). Artistic composition for painterly rendering. En *Proceedings of the conference on vision, modeling and visualization* (pp. 119–126). Eurographics Association.

Litwinowicz, P. (1997). Processing images and video for an impressionist effect. En *Proceedings of the 24th annual conference on computer graphics and interactive techniques - SIGGRAPH '97*. ACM Press.

Lu, J., Sander, P. V., y Finkelstein, A. (2010). Interactive painterly stylization of images, videos and 3d animations. En *Proceedings of the ACM SIGGRAPH symposium on interactive 3d graphics and games - i3d 10*. ACM Press.

Lu, Y., Lam, J. H., y Yam, Y. (2009). Preliminary study on vision-based pen-and-ink drawing by a robotic manipulator. En *2009 ieee/asme international conference on*

advanced intelligent mechatronics (pp. 578–583).

Luhmann, N. (1996). *Introducción a la teoría de sistemas*. Universidad Iberoamericana.

Luo, R. C., Hong, M.-J., y Chung, P.-C. (2016). Robot artist for colorful picture painting with visual control system. En *2016 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 2998–3003).

Macqueen, J. (1967). Some methods for classification and analysis of multivariate observations. En *In 5-th berkeley symposium on mathematical statistics and probability* (pp. 281–297).

Mújica-Vargas, F. J. R.-S. A. J., Dante; Gallegos-Funes. (2013). A fuzzy clustering algorithm with spatial robust estimation constraint for noisy color image segmentation. *Pattern Recognition Letters*.

Mochizuki, K., Nishide, S., Okuno, H. G., y Ogata, T. (2013). Developmental human-robot imitation learning of drawing with a neuro dynamical system. En *2013 ieee international conference on systems, man, and cybernetics* (pp. 2336–2341).

Mohan, V., Morasso, P., Zenzeri, J., Metta, G., Chakravarthy, V. S., y Sandini, G. (2011). Teaching a humanoid robot to draw 'shapes'. *Autonomous Robots*, *31*(1), 21–53.

Moles, A. (1968). *Teoría de la información y percepción estética*. Júcar.

Morín, E., y Le Moigne, J.-L. (2006). *Inteligencia de la complejidad*. Ediciones dee l'aube.

Mortensen, E. N., y Barrett, W. A. (1995). Intelligent scissors for image composition. En *Proceedings of the 22nd annual conference on computer graphics and interactive techniques* (pp. 191–198). New York, NY, USA: ACM.

Moura, L. (2016). Machines that make art en: Herath d., kroos c., stelarc (eds), robots and art. Springer.

Munoz, J.-M., Avalos, J., y Ramos, O. E. (2017). Image-driven drawing system by a nao robot. En *Electronic congress (e-con uni), 2017* (pp. 1–4).

Nicolescu, B. (2006). *La transdisciplinariedad. manifiesto*. Mónaco: Editions du Rocher.

Nishide, S., Mochizuki, K., Okuno, H. G., y Ogata, T. (2014). Insertion of pause in drawing from babbling for robot's developmental imitation learning. En *2014 ieee international conference on robotics and automation (icra)* (pp. 4785–4791).

Olsen, S. C., Maxwell, B. A., y Gooch, B. (2005). Interactive vector fields for painterly rendering. En *Proceedings of graphics interface 2005* (pp. 241–247). Canadian Human-

Computer Communications Society.

Orzan, A., Bousseau, A., Barla, P., y Thollot, J. (2007). Structure-preserving manipulation of photographs. En *Proceedings of the 5th international symposium on non-photorealistic animation and rendering - NPAR '07*. ACM Press.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(1), 62–66.

Perona, P., y Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(7), 629-639.

Rodrigues, B., Cruz, E., Dias, A., y Silva, M. F. (2016). Lsa portraiture robot. En *Robot 2015: Second iberian robotics conference* (pp. 341–352).

Rosenfeld, A., y Weszka, J. S. (1975). An improved method of angle detection on digital curves. *IEEE Transactions on Computers*, *C-24*(9), 940–941.

Rother, C., Kolmogorov, V., y Blake, A. (2004). "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, *23*(3), 309–314.

Ruchanurucks, M., Kudoh, S., Ogawara, K., Shiratori, T., y Ikeuchi, K. (2007). Humanoid robot painter: Visual perception and high-level planning. En *Proceedings 2007 ieee international conference on robotics and automation* (pp. 3028–3033).

Santella, A., y DeCarlo, D. (2002). Abstracted painterly renderings using eye-tracking data. En *Proceedings of the second international symposium on non-photorealistic animation and rendering - NPAR '02*. ACM Press.

Sarfraz, M. (2008). *Interactive curve modeling: With applications to computer graphics, vision and image processing*.

Scalera, L., Seriani, S., Gasparetto, A., y Gallina, P. (2018a). Busker robot: A robotic painting system for rendering images into watercolour artworks. En *Iftomm symposium on mechanism design for robotics* (pp. 1–8).

Scalera, L., Seriani, S., Gasparetto, A., y Gallina, P. (2018b). Watercolour robotic painting: a novel automatic system for artistic rendering. *Journal of Intelligent & Robotic Systems*, 1–16.

Scalera, L., Seriani, S., Gasparetto, A., y Gallina, P. (2019). Non-photorealistic rendering techniques for artistic robotic painting. *Robotics*, *8*(1).

Schopenhauer, A. (1987). *El mundo como voluntad y representación*. Porrúa.

Secord, A. (2002). Weighted voronoi stippling. En *Proceedings of the second international symposium on non-photorealistic animation and rendering - npar 2002*. ACM Press.

Shiraishi, M., y Yamaguchi, Y. (2000). An algorithm for automatic painterly rendering based on local source image approximation. En *Proceedings of the first international symposium on non-photorealistic animation and rendering - NPAR '00*. ACM Press.

Singh, A. K., Baranwal, N., y Nandi, G. C. (2017). Development of a self reliant humanoid robot for sketch drawing. *Multimedia Tools and Applications*, *76*(18), 18847–18870.

Singh, A. K., y Nandi, G. (2016). Nao humanoid robot: Analysis of calibration techniques for robot sketch drawing. *Robotics and Autonomous Systems*, *79*, 108–121.

Sloan, S., y Houlsby, G. (1984). An implementation of watson's algorithm for computing 2-dimensional delaunay triangulations. *Advances in Engineering Software (1978)*, *6*(4), 192 – 197.

Song, D., Lee, T., Kim, Y. J., Sohn, S., y Kim, Y. J. (2018). Artistic pen drawing on an arbitrary surface using an impedance-controlled robot. En *Ieee international conference on robotics and automation (icra)*.

Srikaew, A., Cambron, M., Northrup, S., Peters II, R., Wilkes, M., y Kawamura, K. (1998). Humanoid drawing robot. En *Iasted international conference on robotics and manufacturing*.

Stein, M. R. (2003). The pumapaint project. *Autonomous Robots*, *15*(3), 255–265.

Stephen Boyd, L. V. (2018). *Introduction to applied linear algebra: Vectors, matrices, and least squares*. Cambridge University Press.

Terzopoulos, M. K. A. W. D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, *1*, 321–331.

Tomasi, R., C.; Manduchi. (1998). Bilateral filtering for gray and color images. En *Sixth international conference on computer vision (ieee cat. no.98ch36271)* (pp. 839–846).

Tresset, P., y Deussen, O. (2014). Artistically skilled embodied agents. En U. o. L. Goldsmiths (Ed.), *Proceedings of aisb2014*.

Tresset, P. A., y Leymarie, F. (2012). Sketches by paul the robot. En *Proceedings of the eighth annual symposium on computational aesthetics in graphics, visualization, and imaging* (pp. 17–24).

Tresset, P. A., y Leymarie, F. F. (2013). Portrait drawing by paul the robot. *Computers and*

Graphics, *37*(5), 348 - 363.

Vantaram, E., Sreenath Rao; Saber. (2012). Survey of contemporary trends in color image segmentation. *Journal of Electronic Imaging*, *21*, 21–28.

Vincent, E., y Laganière, R. (2005). Detecting and matching feature points. *Journal of Visual Communication and Image Representation*, *16*(1), 38 - 54.

Wang, H., y Brady, M. (1995). Real-time corner detection algorithm for motion estimation. *Image and Vision Computing*, *13*(9), 695 - 703.

Wang, J., Agrawala, M., y Cohen, M. F. (2007, julio). Soft scissors: An interactive tool for realtime high quality matting. *ACM Trans. Graph.*, *26*(3).

Wang, J., y Zhang, W. (2018). A survey of corner detection methods. En *2018 2nd international conference on electrical engineering and automation (iceea 2018)*. Atlantis Press.

Watanabe, K., Numakura, A., Nishide, S., Gouko, M., y Kim, C. H. (2015). Efficient body babbling for robot's drawing motion. En *2015 ieee international conference on mechatronics and automation (icma)* (pp. 1162–1167).

Watson, D. (2001). *Compound signed decomposition, the core of natural neighbor interpolation in n-dimensional space.* Descargado de `https://www.iamg.org/images/File/documents/oldftp/Watson/core.ps`

Web1. (2019). Descargado de `https://people.eecs.berkeley.edu/~yang/software/lossy_segmentation/`

Wiener, N. (1988). *Cibernética y sociedad* (3ra. ed.). Editorial Sudamericana.

Witkin, A. P. (1983). Scale-space filtering. En *Proceedings of the eighth international joint conference on artificial intelligence - volume 2* (pp. 1019–1022). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Xue, T., y Liu, Y. (2017). Robot portrait rendering based on multi-features fusion method inspired by human painting. En *Robotics and biomimetics (robio), 2017 ieee international conference on* (pp. 2413–2418).

Ye, X., Gu, Y., Sheng, W., Wang, F., Chen, H., y Chen, H. (2017). Deep learning-based human head detection and extraction for robotic portrait drawing. En *Robotics and biomimetics (robio), 2017 ieee international conference on* (pp. 1282–1287).

Zhao, M., y Zhu, S.-C. (2011). Customizing painterly rendering styles using stroke proces-

ses. En *Proceedings of the acm siggraph/eurographics symposium on non-photorealistic animation and rendering* (pp. 137–146). New York, NY, USA: ACM.

ANEXOS

Tabla A. Productos para el sistema robot

No. anexo	Tipo de producto	Título	Líneas de código
1	Interfaz gráfica	Código Matlab	1500
2	Firmware	Código Arduino	430
3	Instrucciones	Guía de uso	

Anexo 1. Código Matlab

Código 1 Programa RegionesTrazo para la interfaz gráfica de usuario

```
function varargout = RegionesTrazos(varargin)
% REGIONESTRAZOS MATLAB código para RegionesTrazos.fig
% Última modificación por GUIDE v2.5 30-Jan-2019 11:33:41
% Inicia código de inicialización - NO EDITE
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @RegionesTrazos_OpeningFcn, ...
                   'gui_OutputFcn', @RegionesTrazos_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback', []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% Fin código de inicialización - NO EDITE

% --- Se ejecuta justo antes de que RegionesTrazos sea visible.
function RegionesTrazos_OpeningFcn(hObject, eventdata, handles, varargin)
% Elige salida de línea de comando default para RegionesTrazos
handles.output = hObject;
% Actualiza estructura handles
guidata(hObject, handles);
evalin( 'base', 'clear␣variables' ); % Limpia variables del workspace
axes(handles.axes2);% ejes sin números
cla(handles.axes2);
set(gca, 'xtick', []);
set(gca, 'ytick', []);
axes(handles.axes1);% ejes sin números
cla(handles.axes1);
set(gca, 'xtick', []);
set(gca, 'ytick', []);

try
% texto en menú3: PUERTOS SERIE conectados
lista=seriallist;
nPuertos=length(lista);
set(handles.popupmenu3,'Value',nPuertos);
set(handles.popupmenu3,'String',lista);
catch
   errordlg('No␣hay␣puertos␣conectados','Atención');
end

% --- Outputs from this function are returned to the command line.
function varargout = RegionesTrazos_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
53
54 %% -- ABRE IMAGEN
55 function togglebutton1_Callback(hObject, eventdata, handles)
56 clc;
57 evalin( 'base', 'clear␣variables' ); % Limpia variables del workspace
58 nRegion=0;
59 assignin('base','nRegion',nRegion);
60 xx=[];
61 yy=[];
62 xCurva=[];
63 yCurva=[];
64 CX=[];
65 CY=[];
66 assignin('base','xCurva',xCurva);
67 assignin('base','yCurva',yCurva);
68 assignin('base','CX',CX);
69 assignin('base','CY',CY);
70 assignin('base','xx',xx);
71 assignin('base','yy',yy);
72 p=path;
73 [f pa] = uigetfile({'*.jpg';'*.bmp';'*.gif';'*.*'},'Selecciona␣imagen');
74 path(p,pa);
75 imagen=imread(f);
76 path(p);
77 imagen_gris=rgb2gray(imagen);
78 ROIcolor=imagen;
79 assignin('base','ROIcolor',ROIcolor);
80 assignin('base','imagen_original',imagen);
81 assignin('base','imagen_trabajo',imagen);
82 assignin('base','imagen_trabajo_gris',imagen_gris);
83 [fil,col,c]=size(imagen);
84 escala = 1600 * 10 / col;
85 assignin('base','escala',escala);
86 imagen_regiones=zeros(fil,col);
87 imagen_regiones_binaria=true(fil,col);
88 rgb_bin_regiones = true(fil,col,3);
89 assignin('base','imagen_regiones',imagen_regiones);
90 assignin('base','imagen_regiones_binaria',imagen_regiones_binaria);
91 assignin('base','rgb_bin_regiones',rgb_bin_regiones);
92 cla(handles.axes2);
93 axes(handles.axes2);% ejes sin números
94 set(gca, 'xtick', []);
95 set(gca, 'ytick', []);
96 imshow(imagen);% muestra imagen original en axes2
97 cla(handles.axes1);
98 axes(handles.axes1);% ejes sin números
99 set(gca, 'xtick', []);
100 set(gca, 'ytick', []);
101 imshow(imagen);% muestra imagen original en axes1
102 % quita lista de menú
103 set(handles.popupmenu1,'Value',1);
104 set(handles.popupmenu1,'String','curva');
105 % área de trabajo robot
106 L= round((get(handles.slider9,'Value')));
107 set(handles.text39,'String',num2str(L));
108 escala = 1600 * L / col;
```

```
assignin('base','escala',escala);
ancho=round(escala*fil/1600);
set(handles.text41,'String',num2str(ancho));
%% -- SELECCIONA PUNTOS DEL CONTORNO
function togglebutton2_Callback(hObject, eventdata, handles)
axes(handles.axes1);
cla(handles.axes1);
imagen=evalin('base','imagen_original');
imshow(imagen);
xx=evalin('base','xx');
yy=evalin('base','yy');
xx=[];
yy=[];
hold on;

% 1. SELECCIONAR PUNTOS CONSECUTIVOS SOBRE EL CONTORNO DE LA REGION
while true
[x y]=ginput(1);
x=round(x);
y=round(y);
xx=[xx; x];
yy=[yy;y];
plot(x,y,'o');
w = waitforbuttonpress;
if(w==1)  % presiona tecla para salirse del ciclo
  break
end
end
assignin('base','xx',xx);
assignin('base','yy',yy);

% 2. DETECCIÓN DE ESQUINAS
Lcontorno = length(xx);  % Longitud del contorno
ci = zeros(Lcontorno,1);  % Guarda el coseno para k
% Obtiene la curvatura para k=1
cuerda = 1;
for k=1:Lcontorno
   ax=0; ay=0; bx=0; by=0;
   % Calcula los vectores a_ik y b_ik, el centro del vector es cada punto
  for j=1:cuerda
     if( (Lcontorno -(k+j))>=0 )  % Compensa al final del contorno
        ay = ay + (yy(k)- yy(k+j));
        ax = ax - (xx(k)- xx(k+j));
     else
        ay = ay + (yy(k)-yy(k+j-Lcontorno));
        ax = ax - (xx(k)-xx(k+j-Lcontorno));
     end

     if( (k-j)>0 )  % Compensa al inicio del contorno
        by = by + (yy(k)-yy(k-j));
        bx = bx - (xx(k)-xx(k-j));
     else
        by = by + (yy(k)-yy(Lcontorno-abs(k-j)));
        bx = bx - (xx(k)-xx(Lcontorno-abs(k-j)));
     end

```

```
   end
    % Obtiene el coseno promedio
   num = ax*bx + ay*by;
   den= sqrt(ax*ax + ay*ay) * sqrt(bx*bx +by*by);
   ci(k)=num/den;
end

% Encuentra los máximos de la curvatura
xxx=linspace(1,Lcontorno,Lcontorno);
[pk,lc,w1,p1] = findpeaks(ci,xxx,'MinPeakProminence',.35);

% 3a. En cada esquina se empieza un nuevo segmento de curva
% y se reconstruye con curvas de Bézier
nEsq=length(lc);
[fil,col,c]=size(imagen);
evalin( 'base', 'clear␣Contorno' ); % Limpia del workspace

if(nEsq>0) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
assignin('base','hayEsquinas',true);
lc=[1 lc]; % añade primer punto trayectoria
L_ci=length(ci);
lc=[lc L_ci]; % añade ultimo punto trayectoria
plot(xx,yy,'o');
ncurvas=length(lc);
assignin('base','lc',lc);
orden= round((get(handles.slider1,'Value')))+1;
vectorOrden=zeros(ncurvas-1,1);
imagen_bresenham=false(fil,col);

% esquinas
for i=1:ncurvas-1
    xxi=xx(lc(i):lc(i+1));
    yyi=yy(lc(i):lc(i+1));
    if(orden<=length(xxi))
        [xB,yB,xc,yc]=Bezierm(xxi,yyi,orden);
    else
        [xB,yB,xc,yc]=Bezierm(xxi,yyi,length(xxi));
    end
    Contorno{i}=[xc',yc'];
    plot(xB,yB);
    vectorOrden(i)=orden;

        for j=1:length(xB)-1
        if(xB(j)<col && yB(j)<fil && xB(j)>0 && yB(j)>0)
          % convierte linea Bezier en imagen
         [xbm,ybm]=bresenham(xB(j),yB(j),xB(j+1),yB(j+1));
         assignin('base','xbm',xbm);
         assignin('base','ybm',ybm);
            for k=1:length(xbm)
                if(xbm(k)<col &&ybm(k)<fil && xbm(k) >0 && ybm(k)>0)
                  imagen_bresenham(ybm(k),xbm(k))=true;
                end
            end

        end
        end
```

```

end
assignin('base','vectorOrden',vectorOrden);
assignin('base','imagen_bresenham',imagen_bresenham);

% texto en menú1
num = ncurvas-2;
set(handles.popupmenu1,'Value',num);
string_nombre='1';
for k = 2 : num+1
   string_nombre = strcat(string_nombre,'|',num2str(k));
end
set(handles.popupmenu1,'String',string_nombre);

else %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 3b. Regiones sin esquinas, REGIÓN CIRCULAR
assignin('base','hayEsquinas',false);

imagen_bresenham=false(fil,col);

if(Lcontorno>=9)
   [xB,yB,xc,yc]=Bezierm(xx,yy,9);
else
   [xB,yB,xc,yc]=Bezierm(xx,yy,Lcontorno);
end
Contorno{1}=[xc',yc'];

hold on;
plot(xx,yy,'o',xB,yB);

for j=1:length(xB)-1
    if(xB(j)<col && yB(j)<fil && xB(j)>0 && yB(j)>0)
        % convierte linea Bezier en imagen
        [xbm,ybm]=bresenham(xB(j),yB(j),xB(j+1),yB(j+1));
           for k=1:length(xbm)
              if(xbm(k)<col &&ybm(k)<fil && xbm(k) >0 && ybm(k)>0)
               imagen_bresenham(ybm(k),xbm(k))=true;
              end
           end

     end
end
assignin('base','imagen_bresenham',imagen_bresenham);
end
assignin('base','Contorno',Contorno);


%% --- MODIFICA UNA CURVA BEZIER DEL CONTORNO
function popupmenu1_Callback(hObject, eventdata, handles)
   cla(handles.axes1);% borra imagen del área de visualización
   axes(handles.axes1);
   imagen=evalin('base','imagen_original');
   imshow(imagen);% muestra imagen original en axes1
   [fil,col,c]=size(imagen);
   hold on;
   xx=evalin('base','xx');
```

```
277    yy=evalin('base','yy');
278    lc=evalin('base','lc');
279    ncurvas=length(lc);
280    plot(xx,yy,'o'); % puntos muestra
281    Ncurva=get(hObject,'Value');
282    vectorOrden=evalin('base','vectorOrden');
283
284   % esquinas
285   for i=1:ncurvas-1
286     xxi=xx(lc(i):lc(i+1));
287     yyi=yy(lc(i):lc(i+1));
288     if(vectorOrden(i)<=length(xxi))
289        [xB,yB]=Bezierm(xxi,yyi,vectorOrden(i));
290     else
291        [xB,yB]=Bezierm(xxi,yyi,length(xxi));
292     end
293
294     if(i==Ncurva)
295        plot(xB,yB,'LineWidth',2,'Color','b');
296     else
297        plot(xB,yB,'m');
298     end
299
300   end
301 % --- Executes during object creation, after setting all properties.
302 function popupmenu1_CreateFcn(hObject, eventdata, handles)
303 if ispc && isequal(get(hObject,'BackgroundColor'),...
304        get(0,'defaultUicontrolBackgroundColor'))
305     set(hObject,'BackgroundColor','white');
306 end
307
308 %% --- SLIDER CAMBIA EL ORDEN DE UNA CURVA BEZIER DEL CONTORNO %%%%%%%%%%%%
309 function slider1_Callback(hObject, eventdata, handles)
310 v=round(get(hObject,'Value'));
311 set(handles.text4,'String',num2str(v));
312 hayEsquinas=evalin('base','hayEsquinas');
313 evalin( 'base', 'clear Contorno' ); % Limpia del workspace
314
315 if(hayEsquinas)
316 % selecciona una curva
317 Ncurva = get(handles.popupmenu1,'Value');
318 assignin('base','Ncurva',Ncurva);
319 cla(handles.axes1);% borra las curvas que se salen del área de imagen
320 axes(handles.axes1);
321 imagen=evalin('base','imagen_original');
322 imshow(imagen);% muestra imagen original en axes1
323 [fil,col,c]=size(imagen);
324 imagen_bresenham=false(fil,col);
325 hold on;
326
327 xx=evalin('base','xx');
328 yy=evalin('base','yy');
329 lc=evalin('base','lc');
330 ncurvas=length(lc);
331 plot(xx,yy,'o');
332
```

```
orden= v+1;
vectorOrden=evalin('base','vectorOrden');

% esquinas
 for i=1:ncurvas-1
  xxi=xx(lc(i):lc(i+1));
  yyi=yy(lc(i):lc(i+1));
  if(i==Ncurva)
     vectorOrden(i)=orden;
     if(vectorOrden(i)<=length(xxi))
       [xB,yB,xc,yc]=Bezierm(xxi,yyi,vectorOrden(i));
     else
       [xB,yB,xc,yc]=Bezierm(xxi,yyi,length(xxi));
     end

     plot(xB,yB,'LineWidth',2,'Color','b');
  else
     if(vectorOrden(i)<=length(xxi))
       [xB,yB,xc,yc]=Bezierm(xxi,yyi,vectorOrden(i));
     else
       [xB,yB,xc,yc]=Bezierm(xxi,yyi,length(xxi));
     end
     plot(xB,yB,'m');
  end

  Contorno{i}=[xc',yc'];


  for j=1:length(xB)-1
     if(xB(j)<col && yB(j)<fil && xB(j)>0 && yB(j)>0)
      % convierte linea Bezier en imagen
      [xbm,ybm]=bresenham(xB(j),yB(j),xB(j+1),yB(j+1));
        for k=1:length(xbm)
           if(xbm(k)<col &&ybm(k)<fil && xbm(k) >0 && ybm(k)>0)
             imagen_bresenham(ybm(k),xbm(k))=true;
           end
        end

     end
  end

 end
assignin('base','vectorOrden',vectorOrden);
assignin('base','imagen_bresenham',imagen_bresenham);

else % si no hay esquinas, es una curva redonda
    orden= v+1;
    % regiones sin esquinas, REGIÓN CIRCULAR
    cla(handles.axes1);% borra las curvas que se salen del área de imagen
    axes(handles.axes1);
    imagen=evalin('base','imagen_original');
    imshow(imagen);% muestra imagen original en axes1
    [fil,col,c]=size(imagen);
    imagen_bresenham=false(fil,col);
    xx=evalin('base','xx');
    yy=evalin('base','yy');
```

```
    if(orden<=length(xx))
      [xB,yB,xc,yc]=Bezierm(xx,yy,orden);
    else
      [xB,yB,xc,yc]=Bezierm(xx,yy,length(xx));
    end
    Contorno{1}=[xc',yc'];

    hold on;
    plot(xx,yy,'o',xB,yB);

     for j=1:length(xB)-1
      if(xB(j)<col && yB(j)<fil && xB(j)>0 && yB(j)>0)
       % convierte linea Bezier en imagen
       [xbm,ybm]=bresenham(xB(j),yB(j),xB(j+1),yB(j+1));
         for k=1:length(xbm)
            if(xbm(k)<col &&ybm(k)<fil && xbm(k) >0 && ybm(k)>0)
              imagen_bresenham(ybm(k),xbm(k))=true;
            end
         end

      end
     end
     assignin('base','imagen_bresenham',imagen_bresenham);

end
assignin('base','Contorno',Contorno);
% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),...
      get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end

%% -- AGREGA REGIÓN
function togglebutton3_Callback(hObject, eventdata, handles)
axes(handles.axes1);
imagen_bresenham=evalin('base','imagen_bresenham');
se = strel('disk',2);
ii=imdilate(imagen_bresenham,se);
iii = imfill(ii,'holes');
iiii=imerode(iii,se);
if (get(handles.checkbox1,'value')==1)
  imagen=evalin('base','imagen_original');% Lee imagen original
else
   imagen=evalin('base','imagen_trabajo');% Lee imagen original
end
[fil,col,c]=size(imagen);
imagen_regiones_binaria=false(fil,col);
imagen_regiones_binaria(iiii)=true;
rgb_bin_regiones = repmat(imagen_regiones_binaria,[1 1 3]);
ROIcolor = imagen;
ROIcolor(~rgb_bin_regiones) = 0;
assignin('base','imagen_regiones_binaria',imagen_regiones_binaria);
assignin('base','rgb_bin_regiones',rgb_bin_regiones);
assignin('base','ROIcolor',ROIcolor);
imshow(ROIcolor);
```

```
% Una región
Ngrupos=1;
grupos_etiqueta = imsegkmeans(ROIcolor,Ngrupos,'NumAttempts',3);
assignin('base','grupos_etiqueta',grupos_etiqueta);

%% -- DISEÑA CURVA PARA CAMPO
function togglebutton4_Callback(hObject, eventdata, handles)
axes(handles.axes1);
cla(handles.axes1);
ROI_color=evalin('base','ROIcolor');
imshow(ROI_color);
xCurva=evalin('base','xCurva');% Lee coordenadas x de curva
yCurva=evalin('base','yCurva');% Lee coordenadas y de curva
[filas cols]=size(yCurva);
hold on;
for i=1:filas
   plot(xCurva(i,:),yCurva(i,:));
end
hold off;
hDibujo = drawfreehand(gca,'Closed',0,'Multiclick',true);
xy = (hDibujo.Position);
assignin('base','xy',xy);

%% --- AGREGA CURVA PARA CAMPO
function togglebutton5_Callback(hObject, eventdata, handles)
ROIcolor=evalin('base','ROIcolor');% Lee imagen segmentada
axes(handles.axes1);
cla(handles.axes1);
imshow(ROIcolor);
xy=evalin('base','xy');% Lee dibujo a mano
[xB,yB,controlX,controlY]=Bezierm(xy(:,1),xy(:,2),5);  %orden 4
xCurva=evalin('base','xCurva');% Lee cordenadas x de curva
yCurva=evalin('base','yCurva');% Lee cordenadas y de curva
CX=evalin('base','CX');% Lee puntos de control en x
CY=evalin('base','CY');% Lee puntos de control en y
xCurva=[xCurva; xB'];
yCurva=[yCurva; yB'];
CX=[CX; controlX'];
CY=[CY; controlY'];
assignin('base','xCurva',xCurva);
assignin('base','yCurva',yCurva);
assignin('base','CX',CX);
assignin('base','CY',CY);
[filas cols]=size(yCurva);
hold on;
for i=1:filas
   plot(xCurva(i,:),yCurva(i,:));
end

%% --- BORRA UNA CURVA PARA CAMPO
function togglebutton6_Callback(hObject, eventdata, handles)
xCurva=evalin('base','xCurva');% Lee coordenadas x de curva
yCurva=evalin('base','yCurva');% Lee coordenadas y de curva
[filas cols]=size(yCurva);
xCurva(filas,:)=[];
yCurva(filas,:)=[];
```

```
assignin('base','xCurva',xCurva);
assignin('base','yCurva',yCurva);
CX=evalin('base','CX');% Lee puntos de control en x
CY=evalin('base','CY');% Lee puntos de control en y
[filasCX colsCX]=size(CX);
[filas cols]=size(yCurva);
CX(filasCX,:)=[];
CY(filasCX,:)=[];
assignin('base','CX',CX);
assignin('base','CY',CY);
ROIcolor=evalin('base','ROIcolor');% Lee imagen segmentada
axes(handles.axes1);
cla(handles.axes1);
imshow(ROIcolor);
hold on;
for i=1:filas
   plot(xCurva(i,:),yCurva(i,:));
end

%% --- INTERPOLA CURVAS DE CAMPO
function togglebutton7_Callback(hObject, eventdata, handles)
% falta . no tomar en cuenta NaN
xCurva=evalin('base','xCurva');% Lee coordenadas x de curva
yCurva=evalin('base','yCurva');% Lee coordenadas y de curva
CX=evalin('base','CX');% Lee puntos de control en x
CY=evalin('base','CY');% Lee puntos de control en y
[filasCX colsCX]=size(CX);
t = linspace(0,1,100);

for i=1:filasCX
% derivada de la curva de Bézier de orden 4
derivada = kron(4.*t.*t.*t-12.*t.*t+12.*t-4,[CX(i,1);CY(i,1)]) +...
        kron(-16.*t.*t.*t+36.*t.*t-24.*t+4,[CX(i,2);CY(i,2)]) +...
        kron(24.*t.*t.*t-36.*t.*t+12.*t,[CX(i,3);CY(i,3)]) +...
        kron(12.*t.*t-16.*t.*t.*t,[CX(i,4);CY(i,4)]) +...
        kron(4.*t.*t.*t,[CX(i,5);CY(i,5)]);
dx(i,:)=derivada(1,:);
dy(i,:)=derivada(2,:);
end

[filas cols]=size(yCurva);
nc1=linspace(1,filas*cols,filas*cols);
xcorr=[]; dxcorr=[]; ycorr=[]; dycorr=[];% concatena las filas de XX1

% región
for i=1:filas
  xcorr=[xcorr,xCurva(i,:)];
  dxcorr=[dxcorr,dx(i,:)];
  ycorr=[ycorr,yCurva(i,:)];
  dycorr=[dycorr,dy(i,:)];
end
% plot(xcorr,ycorr,'o');
barraProgreso = waitbar(0,'Espere␣por␣favor...');
% interpola campo region
Fx1 = scatteredInterpolant(xcorr',ycorr',dxcorr','natural','linear');
Fy1 = scatteredInterpolant(xcorr',ycorr',dycorr','natural','linear');
```

```
assignin('base','Fx1',Fx1);
assignin('base','xcorr',xcorr);
assignin('base','ycorr',ycorr);
assignin('base','dxcorr',dxcorr);
assignin('base','dycorr',dycorr);
% visualiza
imagen=evalin('base','imagen_original');% Lee imagen
[Fim Cim can]=size(imagen);
[xq,yq] = meshgrid(1:1:Cim, 1:1:Fim);
dvx1=Fx1(xq,yq);% region
dvy1=Fy1(xq,yq);
[LICImage, intensity,normvx,normvy] = grayLIC(dvy1,dvx1,3);
assignin('base','normvx',normvx);% guarda en workspace
assignin('base','normvy',normvy);
imagen_regiones_binaria=evalin('base','imagen_regiones_binaria');
LICImage(~imagen_regiones_binaria) = 0;
close(barraProgreso);
assignin('base','LICImage',LICImage);
figure;imshow(LICImage);% hold on; plot(xCurva,yCurva,'b.');

%% --- DIFUSIÓN ANISÓTROPA
function togglebutton8_Callback(hObject, eventdata, handles)
if (get(handles.checkbox1,'value')==1)
  J=evalin('base','imagen_original');% Lee imagen original
else
   J=evalin('base','imagen_trabajo');% Lee imagen original
end
jR=J(:,:,1);jG=J(:,:,2);jB=J(:,:,3);% R,G,B
% Aplica difusión anisótropa
iteraciones= round(get(handles.slider3,'Value'));
id1 = difusionanisotropica(jR,iteraciones);
id2 = difusionanisotropica(jG,iteraciones);
id3 = difusionanisotropica(jB,iteraciones);
I=zeros(size(J));
I(:,:,1)=id1(:,:);I(:,:,2)=id2(:,:);I(:,:,3)=id3(:,:);  % RGB
I=uint8(I);
axes(handles.axes1);
cla(handles.axes1);
II=rgb2gray(I);
assignin('base','imagen_trabajo_gris',II);
assignin('base','imagen_trabajo',I);
colorSegmento = I;
rgb_bin_regiones=evalin('base','rgb_bin_regiones');% Lee región manual
colorSegmento(~rgb_bin_regiones) = 0;
imshow(colorSegmento);

%% --- SEGMENTACIÓN K MEDIAS
function togglebutton9_Callback(hObject, eventdata, handles)
if (get(handles.checkbox1,'value')==1)
  J=evalin('base','imagen_original');% Lee imagen original
else
  J=evalin('base','imagen_trabajo');% Lee imagen original
end
Ngrupos= round(get(handles.slider2,'Value'));
rgb_bin_regiones=evalin('base','rgb_bin_regiones');% Lee región manual
% Repite agrupamiento 3 veces para evitar mínimo local
```

```
J(~rgb_bin_regiones)=0;
grupos_etiqueta = imsegkmeans(J,Ngrupos,'NumAttempts',3);
subSegmentos = label2rgb(grupos_etiqueta);
assignin('base','grupos_etiqueta',grupos_etiqueta);
subSegmentos(~rgb_bin_regiones) = 0;
axes(handles.axes1);
cla(handles.axes1);
imshow(subSegmentos);
% texto en menú2
num = Ngrupos-1;
set(handles.popupmenu2,'Value',num);
string_nombre='1';
for k = 2 : num+1
   string_nombre = strcat(string_nombre,'|',num2str(k));
end
set(handles.popupmenu2,'String',string_nombre);

%% --- CREA TRAZOS
function togglebutton10_Callback(hObject, eventdata, handles)
try
% selecciona región
I=evalin('base','imagen_original');
imgGris=rgb2gray(I);
Nregion = get(handles.popupmenu2,'Value')
if(Nregion==1)
   imagen_regiones_binaria=evalin('base','imagen_regiones_binaria');
   regionPintar =imagen_regiones_binaria;
else
J = evalin('base','grupos_etiqueta');% Lee imagen etiquetada
regionPintar = (J==Nregion);
 try
 rgb_bin_regiones=evalin('base','rgb_bin_regiones');% Lee región manual
 region=rgb_bin_regiones(:,:,1);
 regionPintar(~region) = false;
 catch
 end
end

regionPintar2=regionPintar;
imGris=imgGris;
imGris(~regionPintar)=0;
% realiza kmedias
Ncolores= round(get(handles.slider6,'Value'));
label = imsegkmeans(imgGris,Ncolores,'NumAttempts',3);
imgLabelKmedias=kmediasOrdenada(label,imgGris);
assignin('base','imgLabelKmedias',imgLabelKmedias);
Lienzo=255*ones(size(I));
[fils cols]=size(imgGris);
xx=0;
yy=0;
TXX = evalin('base','normvx');% Lee campo vectorial
TYY = evalin('base','normvy');
Longitud=round(get(handles.slider8,'Value'));
anchoPincel=round(get(handles.slider7,'Value'));
se = strel('disk',anchoPincel);
se2 = strel('disk',round(anchoPincel/2));
```

```
669 borde=2;
670 xbb=[];
671 ybb=[];
672 CXrobot=[];
673 CYrobot=[];
674 color=[];
675 dilatada=false(size(imgGris));
676 unaCurvaBinaria=false(size(imgGris));
677 barraProgreso = waitbar(0,'Espere por favor...');
678
679 while true
680     % 1. Obtiene semilla
681    unaCurvaBinaria=true(size(imgGris)); % prueba si termina
682    for i=borde:fils-borde % orilla para evitar puntos solos
683      for j=borde:cols-borde
684         if(regionPintar2(i,j))
685            xx=j; yy=i; colorTemp=[]; xPast = 0; yPast = 0;
686            cooxD=[]; cooxI=[]; cooyD=[]; cooyI=[];
687            unaCurvaBinaria=false(size(imgGris));ccc=0;
688            break;
689         end
690      end
691    end
692    % 2. Obtiene una curva del campo vectorial
693       % Derecha
694       x = xx; y = yy;
695       for k = 1:Longitud % integración hacia adelante, LIC
696             %evita repetición en un punto
697          if (((round(x) ~= round(xPast)) || (round(y) ~= round(yPast))))
698              Lienzo(round(y),round(x),:)=255;ccc=ccc+1;
699              colork=imgLabelKmedias(round(y),round(x));
700              unaCurvaBinaria(round(y),round(x))=true;
701              % agrega las coordenadas a la curva
702              cooxD=[cooxD;x]; cooyD=[cooyD;y];
703              colorTemp=[colorTemp;colork];
704          end
705          % actualiza coordenadas
706          xPast = x; yPast = y;
707          x = x - TYY(round(y),round(x));if x<1 || x>cols break; end
708          y = y - TXX(round(y),round(x));if y<1 || y>fils break; end
709          if(~regionPintar(round(y),round(x))) break; end
710       end % fin de Longitud
711
712       % izquierda
713       x = xx; y = yy;
714       for k = 1:Longitud % integración hacia adelante, LIC
715             % evita coordenadas en un punto
716          if (((round(x) ~= round(xPast)) || (round(y) ~= round(yPast))))
717              Lienzo(round(y),round(x),:)=255;ccc=ccc+1;
718              colork=imgLabelKmedias(round(y),round(x));
719              unaCurvaBinaria(round(y),round(x))=true;
720              % agrega las coordenadas a la curva
721              cooxI=[cooxI;x]; cooyI=[cooyI;y];
722              colorTemp=[colorTemp;colork];
723          end
724          % actualiza coordenadas
```

```
725          xPast = x; yPast = y;
726          x = x + TYY(round(y),round(x));if x<1 || x>cols break; end
727          y = y + TXX(round(y),round(x));if y<1 || y>fils break; end
728          if(~regionPintar(round(y),round(x))) break; end
729       end % fin de Longitud
730
731     % 3. Procesa una curva
732       cooxD=flip(cooxD);cooyD=flip(cooyD);
733       cooxD=[cooxD;cooxI];cooyD=[cooyD;cooyI];
734       colorCurva=mode(colorTemp);
735       if(length(cooxD)>5)
736       [xB,yB,controX,controY]=Bezierm(cooxD,cooyD,5);  %orden 4
737       CXrobot=[CXrobot; controX'];
738       CYrobot=[CYrobot; controY'];
739       xbb=[xbb; xB'];
740       ybb=[ybb; yB'];
741       color=[color;colorCurva];
742       end
743
744       % interroga tamaño de región
745       dilatada = ~(imdilate(unaCurvaBinaria,se));
746       regionPintar=regionPintar&dilatada;
747       regionPintar2=imerode(regionPintar,se2);
748       if(~any(regionPintar,'all'))  % si es cero la región
749          break;  % termina while
750       end
751
752 end  % fin while
753
754 close(barraProgreso);
755  % ordena por color
756 [colorOrdenado,indice]= sort(color);
757 CXordenado=CXrobot(indice,:);
758 CYordenado=CYrobot(indice,:);
759 assignin('base','CXrobot',CXordenado);
760 assignin('base','CYrobot',CYordenado);
761 assignin('base','color',colorOrdenado);
762 axes(handles.axes1);
763 cla(handles.axes1);
764 imshow(uint8(Lienzo));
765 [numCurvas cls]=size(ybb);
766 hold on;
767 for m=1:numCurvas
768    plot(xbb(m,:),ybb(m,:),'r');
769 end
770
771 catch
772   errordlg('Falta␣crear␣campo','Atención');
773 end
774 %% --- ABRE PUERTO SERIE
775 function togglebutton11_Callback(hObject, eventdata, handles)
776 try
777 f=get(hObject,'value');
778 fin = findobj(gcf,'tag','togglebutton26');
779 contents = cellstr(get(handles.popupmenu3,'String'))
780 nombrePuerto = contents{get(handles.popupmenu3,'Value')};  % selecciona item
```

```
781 if f==1
782  set(fin,'string','Cerrar puerto');
783  s=serial(nombrePuerto);
784  s.BaudRate=9600;
785  %s.InputBufferSize = 1;
786  %s.OutputBufferSize = 1;
787  assignin('base','s',s);
788  fopen(s);
789 elseif f==0
790  set(fin,'string','Abrir puerto');
791  s = evalin('base','s');
792  fclose(s); % cierra puerto
793  delete(s);
794  %evalin( 'base', 'clear variables' ); % Limpia variables del workspace
795 end
796
797 catch
798   errordlg('Falta conectar puerto','Atención');
799 end
800 %% --- CAMPO AUTOMÁTICO CON ETF
801 function togglebutton12_Callback(hObject, eventdata, handles)
802 if (get(handles.checkbox1,'value')==1)
803   J=evalin('base','imagen_original');% Lee imagen original
804 else
805    J=evalin('base','imagen_trabajo_gris');% Lee imagen de trabajo
806 end
807 % convierte a gris
808 [a b c]=size(J);
809 if(c==3)
810 II=double(rgb2gray(J));
811 else
812    II=double(J);
813 end
814 radio= round(get(handles.slider4,'Value'));
815 iteraciones=round(get(handles.slider5,'Value'));
816 [TXX TYY]= EdgeTangentFlow(II,radio,iteraciones); % Ajuste de gradiente
817 % Visualiza campo vectorial
818 [LICImage, intensity,normvx,normvy] = grayLIC(TYY,TXX,3);
819 assignin('base','normvx',normvx);% guarda en workspace
820 assignin('base','normvy',normvy);
821 assignin('base','LICImage',LICImage);
822 % visualiza
823 imagen_regiones_binaria=evalin('base','imagen_regiones_binaria');
824 LICImage(~imagen_regiones_binaria) = 0;
825 figure;imshow(LICImage);title('Gradiente ajustado');
826
827 %% --- PINTA REGIÓN
828 function togglebutton13_Callback(hObject, eventdata, handles)
829 CXrobot = evalin('base','CXrobot');% Lee puntos de control
830 CYrobot = evalin('base','CYrobot');% Lee puntos de control
831 escala=evalin('base','escala');% Lee escala
832 CXrobot=escala*CXrobot;
833 CYrobot=escala*CYrobot;
834 color=uint8(evalin('base','color'));
835 colorAsignado= 249+round(get(handles.slider10,'Value'));
836 color2=color;color2(color2<250)=250;
```

```
837 if (get(handles.checkbox2,'value')==1)
838    % un solo color
839    color2(:)=colorAsignado;
840 end
841 A=[CXrobot,CYrobot];
842 [L,K]=size(A);
843 s = evalin('base','s');
844 barraProgreso = waitbar(0,'Pintando...');
845
846 % envía al puerto serie s
847 for i=1:L % L es el no. de filas
848    fprintf(s,int2str(K));% envía longitud de la fila
849    entra=fread(s,1);
850    fprintf(s,int2str(color2(i)));% envía color de la fila
851    entra=fread(s,1);
852    % envía los puntos de control
853   for j=1:K
854      if(entra==j+1)
855        fprintf(s,int2str(A(i,j)));
856        entra=fread(s,1);
857      end
858   end
859   waitbar(i/L,barraProgreso,sprintf(' %d de %i',i,L));
860   while true % espera hasta que Arduino finalice trazo de una curva
861    entra=fread(s,1);
862    if entra==255
863     break;
864    end
865   end
866 end
867 %fin de trabajo
868  fprintf(s,'255');
869  close(barraProgreso);
870
871 %% -- PINTA CONTORNO
872 function togglebutton14_Callback(hObject, eventdata, handles)
873 % pendiente negativo
874 Contorno=evalin('base','Contorno');
875 escala=evalin('base','escala');% Lee escala
876 L=length(Contorno);
877 colorContorno=zeros(L,1);
878 colorAsignado= 249+round(get(handles.slider10,'Value'));
879 if (get(handles.checkbox2,'value')==1)
880    colorContorno(:)=colorAsignado;
881 else
882 colorContorno(:)=250;
883 end
884 s = evalin('base','s');
885 barraProgreso = waitbar(0,'Pintando...');
886
887 % envía al puerto serie s
888 for i=1:L % L es el no. de filas
889    L2=length(Contorno{1,i});
890    fprintf(s,int2str(L2));% envía longitud de la fila
891    entra=fread(s,1);
892    fprintf(s,int2str(colorContorno(i)));% envía color de la fila
```

```
893    entra=fread(s,1);
894    %envía puntos de control
895   for j=1:L2
896      if(entra==j+1)
897        fprintf(s,int2str(int32(escala*Contorno{i}(j))));
898        entra=fread(s,1);
899      end
900   end
901
902   while true % espera hasta que Arduino finalice trazo de una curva
903     entra=fread(s,1);
904     if entra==255
905       break;
906     end
907   end
908
909 waitbar(i/L,barraProgreso,sprintf('%d de %i',i,L));
910 end
911 %fin de trabajo
912  fprintf(s,'255');
913  close(barraProgreso);
914
915
916 %% --- DETECCIÓN DE BORDES USANDO CANNY
917 function togglebutton15_Callback(hObject, eventdata, handles)
918 % región de interés
919 Nregion = get(handles.popupmenu2,'Value');
920 if(Nregion==0)
921    imagen_regiones_binaria=evalin('base','imagen_regiones_binaria');
922    regionPintar =imagen_regiones_binaria;
923 else
924 eti = evalin('base','grupos_etiqueta');% Lee imagen etiquetada
925 regionPintar = (eti==Nregion);
926  try
927  rgb_bin_regiones=evalin('base','rgb_bin_regiones');% Lee región manual
928  region=rgb_bin_regiones(:,:,1);
929  regionPintar(~region) = false;
930  catch
931  end
932 end
933
934 % imagen
935 if (get(handles.checkbox1,'value')==1)
936   J=evalin('base','imagen_original');% Lee imagen original
937 else
938    J=evalin('base','imagen_trabajo_gris');% Lee imagen de trabajo
939 end
940 % convierte a gris
941 [a b c]=size(J);
942 if(c==3)
943 II=double(rgb2gray(J));
944 else
945    II=double(J);
946 end
947 assignin('base','II',II);
948 % detecta bordes usado Canny
```

```
II=II/max(max(II));
BW=edge(II,'Canny',[.02 .1]);
BW(~regionPintar)=0;  % en región de interés
BW2=bwmorph(BW,'skel',Inf);% adelgaza esqueleto
% elimina intersecciones
BW3=bwmorph(BW2,'branchpoints');% intersección de ramas
BW4=imdilate(BW3,strel('square',2));
BW5=~BW4&BW2;
% elimina ramas de tam 8
BW6 = bwareaopen(BW5,8,8);% elimina puntos sueltos tam 8, conexión 8
cc1= bwconncomp(BW6);
L = labelmatrix(cc1);
% detecta esquinas
n=max(max(L));
BB=false(size(BW6));
for i=1:n
   Li=L==i;
   [B lab] = bwboundaries(Li,'noholes');  % Obtiene coordenadas del borde
   ubicacion_contorno = B{1};
   xx=ubicacion_contorno(:,2);
   yy=ubicacion_contorno(:,1);
   Lcontorno = length(xx);  % Longitud del contorno
   ci = zeros(Lcontorno,1);  % Guarda el coseno para k
   % Obtiene la curvatura para k=1
   cuerda = 3;
   for k=1:Lcontorno
    ax=0; ay=0; bx=0; by=0;
    % Calcula los vectores a_ik y b_ik, el centro del vector es cada punto
    for j=1:cuerda
      if( (Lcontorno -(k+j))>=0 )  % Compensa al final del contorno
        ay = ay + (yy(k)- yy(k+j));
        ax = ax - (xx(k)- xx(k+j));
      else
        ay = ay + (yy(k)-yy(k+j-Lcontorno));
        ax = ax - (xx(k)-xx(k+j-Lcontorno));
      end

      if( (k-j)>0 )  % Compensa al inicio del contorno
        by = by + (yy(k)-yy(k-j));
        bx = bx - (xx(k)-xx(k-j));
      else
        by = by + (yy(k)-yy(Lcontorno-abs(k-j)));
        bx = bx - (xx(k)-xx(Lcontorno-abs(k-j)));
      end

    end
    % Obtiene el coseno promedio
    num = ax*bx + ay*by;
    den= sqrt(ax*ax + ay*ay) * sqrt(bx*bx +by*by);
    ci(k)=num/den;
   end
% Encuentra los máximos de la curvatura
xxx=linspace(1,Lcontorno,Lcontorno);
[pk,lc,w1,p1] = findpeaks(ci,xxx,'MinPeakProminence',.45);
BB(yy(lc(:)), xx(lc(:)))=true;
end
```

```
BB=~imdilate(BB,strel('square',2)); %elimina esquinas
BW7=BB&BW6;
BW8 = bwareaopen(BW7,8,8);% elimina puntos sueltos tam 8, conexión 8
% reconstruye las curvas
cc1= bwconncomp(BW8);
L = labelmatrix(cc1);
n=max(max(L));
CXrobotBordes=[];
CYrobotBordes=[];
xbb=[];
ybb=[];
for i=1:n
   Li=L==i;
   EX=bwmorph(Li,'endpoints');[y, x] = find(EX == 1); % primer punto
   B = bwtraceboundary(Li,[y(1) x(1)],'W');% coordenadas del borde
   xx=B(1:ceil(length(B)/2),2);
   yy=B(1:ceil(length(B)/2),1);
   if(length(xx)>8)
     [xB,yB,controX,controY]=Bezierm(xx,yy,8); % orden 7
     CXrobotBordes=[CXrobotBordes; controX'];
     CYrobotBordes=[CYrobotBordes; controY'];
     xbb=[xbb; xB'];
     ybb=[ybb; yB'];
   end
end
[numCurvas cls]=size(ybb);
assignin('base','CXrobotBordes',CXrobotBordes);
assignin('base','CYrobotBordes',CYrobotBordes);
imshow(L);
hold on;
for m=1:numCurvas
   plot(xbb(m,:),ybb(m,:),'r');
end

%% --- PINTA BORDES
function togglebutton16_Callback(hObject, eventdata, handles)
CXrobot = evalin('base','CXrobotBordes');% Lee puntos de control
CYrobot = evalin('base','CYrobotBordes');% Lee puntos de control
escala=evalin('base','escala');% Lee escala
CXrobot=escala*CXrobot;
CYrobot=escala*CYrobot;
A=[CXrobot,CYrobot];
[L,K]=size(A);
colorBordes=zeros(L,1);
colorAsignado= 249+round(get(handles.slider10,'Value'));
if (get(handles.checkbox2,'value')==1)
   colorBordes(:)=colorAsignado;
else
colorBordes(:)=250;
end

s = evalin('base','s');
barraProgreso = waitbar(0,'Pintando...');
% envía al puerto serie s
for i=1:L % L es el no. de filas
   fprintf(s,int2str(K));% envía longitud de la fila
```

```
    entra=fread(s,1);
    fprintf(s,int2str(colorBordes(i)));% envía color de la fila
    entra=fread(s,1);
     %envía los puntos de control
   for j=1:K
      if(entra==j+1)
        fprintf(s,int2str(A(i,j)));
        entra=fread(s,1);
      end
   end
   waitbar(i/L,barraProgreso,sprintf('%d de %i',i,L));
   while true  % espera hasta que Arduino finalice trazo de una curva
     entra=fread(s,1);
     if entra==255
       break;
     end
   end

end
% fin de trabajo
 fprintf(s,'255');
 close(barraProgreso);

 %% --- VISUALIZA REGIONES POR MENÚ
function popupmenu2_Callback(hObject, eventdata, handles)
item = get(hObject,'Value');
J = evalin('base','grupos_etiqueta');% Lee imagen etiquetada
Segmento = (J==item);
axes(handles.axes1);
cla(handles.axes1);
try
rgb_bin_regiones=evalin('base','rgb_bin_regiones');% Lee región manual
region=rgb_bin_regiones(:,:,1);
Segmento(~region) = 0;
catch
end
imshow(Segmento);
% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
      get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
function popupmenu3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
      get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

 % --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
 % --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
```

```
1117
1118 % --- SLIDER N REGIONES K-MEDIAS
1119 function slider2_Callback(hObject, eventdata, handles)
1120 v=round(get(hObject,'Value'));
1121 set(handles.text9,'String',num2str(v));
1122 function slider2_CreateFcn(hObject, eventdata, handles)
1123 if isequal(get(hObject,'BackgroundColor'), ...
1124       get(0,'defaultUicontrolBackgroundColor'))
1125    set(hObject,'BackgroundColor',[.9 .9 .9]);
1126 end
1127
1128  % --- SLIDER ITERACIONES DIFUSIÓN
1129 function slider3_Callback(hObject, eventdata, handles)
1130 v=round(get(hObject,'Value'));
1131 set(handles.text8,'String',num2str(v));
1132 function slider3_CreateFcn(hObject, eventdata, handles)
1133 if isequal(get(hObject,'BackgroundColor'), ...
1134       get(0,'defaultUicontrolBackgroundColor'))
1135    set(hObject,'BackgroundColor',[.9 .9 .9]);
1136 end
1137
1138  % --- SLIDER RADIO ETF
1139 function slider4_Callback(hObject, eventdata, handles)
1140 v=round(get(hObject,'Value'));
1141 set(handles.text11,'String',num2str(v));
1142 function slider4_CreateFcn(hObject, eventdata, handles)
1143 if isequal(get(hObject,'BackgroundColor'), ...
1144       get(0,'defaultUicontrolBackgroundColor'))
1145    set(hObject,'BackgroundColor',[.9 .9 .9]);
1146 end
1147
1148 % --- SLIDER ITERACIONES ETF
1149 function slider5_Callback(hObject, eventdata, handles)
1150 v=round(get(hObject,'Value'));
1151 set(handles.text15,'String',num2str(v));
1152 function slider5_CreateFcn(hObject, eventdata, handles)
1153 if isequal(get(hObject,'BackgroundColor'), ...
1154       get(0,'defaultUicontrolBackgroundColor'))
1155    set(hObject,'BackgroundColor',[.9 .9 .9]);
1156 end
1157
1158 % --- SLIDER N NIVELES DE COLORES A PINTAR
1159 function slider6_Callback(hObject, eventdata, handles)
1160 v=round(get(hObject,'Value'));
1161 set(handles.text44,'String',num2str(v));
1162 function slider6_CreateFcn(hObject, eventdata, handles)
1163 if isequal(get(hObject,'BackgroundColor'), ...
1164       get(0,'defaultUicontrolBackgroundColor'))
1165    set(hObject,'BackgroundColor',[.9 .9 .9]);
1166 end
1167
1168 % --- SLIDER ANCHO DE PINCELADA
1169 function slider7_Callback(hObject, eventdata, handles)
1170 v=round(get(hObject,'Value'));
1171 set(handles.text30,'String',num2str(v));
1172 function slider7_CreateFcn(hObject, eventdata, handles)
```

```
if isequal(get(hObject,'BackgroundColor'), ...
      get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- SLIDER LARGO DE PINCELADA
function slider8_Callback(hObject, eventdata, handles)
v=round(get(hObject,'Value'));
set(handles.text35,'String',num2str(v));
function slider8_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), ...
      get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end

 % ---SLIDER ESCALA ÁREA DE TRABAJO ROBOT
function slider9_Callback(hObject, eventdata, handles)
L=round(get(hObject,'Value'));
set(handles.text39,'String',num2str(L));
J=evalin('base','imagen_original');% Lee imagen original
[filas,columnas,canales]=size(J);
escala = 1600 * L / columnas;
assignin('base','escala',escala);
ancho=round(escala*filas/1600);
set(handles.text41,'String',num2str(ancho));
function slider9_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), ...
      get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end

 % --- SLIDER COLOR A PINTAR
function slider10_Callback(hObject, eventdata, handles)
v=round(get(hObject,'Value'));
set(handles.text43,'String',num2str(v));
function slider10_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), ...
      get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Aproximación por curva de Bézier
function [x,y,cx,cy]=Bezierm(xi,yi,grado)
% Aproxima los puntos xi,yi sin pasar por ellos
% u calculada como longitud de cuerda
for i=1:length(xi)-1
  d(i)=sqrt(sqrt((xi(i+1)-xi(i))^2+(yi(i+1)-yi(i))^2));
end
scum=cumsum(d);
uu=zeros(length(xi),1);
uu(1)=0;uu(length(xi))=1;
for i=1:length(xi)-2
  uu(i+1)=scum(i)/scum(length(xi)-1);
end
```

```
n=length(uu);beta=zeros(n,grado);
for i=1:n
    for j=1:grado
      beta(i,j)= Bi(j-1,uu(i),grado-1);
    end
end
cx=(beta'*beta)\(beta'*xi); cy=(beta'*beta)\(beta'*yi);
% Reconstrucción de la curva
u=linspace(0,1,100); x=zeros(100,1); y=zeros(100,1);
for i=1:100
    for j=1:grado
        x(i)=x(i)+ cx(j)*Bi(j-1,u(i),grado-1);
        y(i)=y(i)+ cy(j)*Bi(j-1,u(i),grado-1);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function B=Bi(i,u,n)
B=(factorial(n)/(factorial(i)*factorial(n-i)))*(u^i)*((1-u)^(n-i));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,y]=bresenham(x1,y1,x2,y2)
%Versión optimizada del algoritmo de línea de Bresenham
%Formato:
% [x y]=bham(x1,y1,x2,y2)
%Entrada:
% (x1,y1): posición inicial
% (x2,y2): posición final
%Salida:
% x y: coordenadas desde (x1,y1) a (x2,y2)
%Ejemplo:
% [x y]=bham(1,1, 10,-5);
% plot(x,y,'or');
x1=round(x1); x2=round(x2);
y1=round(y1); y2=round(y2);
dx=abs(x2-x1);
dy=abs(y2-y1);
steep=abs(dy)>abs(dx);
if steep t=dx;dx=dy;dy=t; end

%Algoritmo principal
if dy==0
    q=zeros(dx+1,1);
else
    q=[0;diff(mod([floor(dx/2):-dy:-dy*dx+floor(dx/2)]',dx))>=0];
end
%-------------------
if steep
    if y1<=y2 y=[y1:y2]'; else y=[y1:-1:y2]'; end
    if x1<=x2 x=x1+cumsum(q);else x=x1-cumsum(q); end
else
    if x1<=x2 x=[x1:x2]'; else x=[x1:-1:x2]'; end
    if y1<=y2 y=y1+cumsum(q);else y=y1-cumsum(q); end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Difusión anisótropa
function J = difusionanisotropica(I,iteraciones)
I=double(I);
K=2;
dt=0.25;
[Ny,Nx]=size(I);
barraProgreso = waitbar(0,'Espere por favor...');
for i=1:iteraciones
  % calcula el gradiente en las direcciones (N,S,E,W)
  In=[I(1,:); I(1:Ny-1,:)]-I;
  Is=[I(2:Ny,:); I(Ny,:)]-I;
  Ie=[I(:,2:Nx) I(:,Nx)]-I;
  Iw=[I(:,1) I(:,1:Nx-1)]-I;
  % calcula los coeficientes de difusión en las direcciones (N,S,E,W)
  Cn=1./(1+(abs(In)/K).^2);
  Cs=1./(1+(abs(Is)/K).^2);
  Ce=1./(1+(abs(Ie)/K).^2);
  Cw=1./(1+(abs(Iw)/K).^2);
  % Próxima imagen I
  I = I + dt*(Cn.*In + Cs.*Is + Ce.*Ie + Cw.*Iw);
  waitbar(i/iteraciones,barraProgreso,sprintf('Procesando %d',i));
end
close(barraProgreso);
J=uint8(I);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Flujo tangente al borde
function [TXX TYY]= EdgeTangentFlow(I,ventana,iteraciones)
%recibe una imagen en color
%obtiene gradiente
[a b c]=size(I);
if(c==3)
II=double(rgb2gray(I));
else
   II=double(I);
end
% op_dif_x=[-1 0 1];
% op_dif_y=[-1; 0 ;1];
op_dif_x=[-1 -2 0 2 1;
        -4 -8 0 8 4;
        -6 -12 0 12 6;
        -4 -8 0 8 4;
        -1 -2 0 2 1;];
op_dif_y=[-1 -4 -6 -4 -1;
        -2 -8 -12 -8 -2;
         0 0 0 0 0;
         2 8 12 8 2;
         1 4 6 4 1;];

gradx=double(imfilter(II,op_dif_x));
grady=double(imfilter(II,op_dif_y));
[tamy,tamx]=size(II);
mag=double(sqrt(gradx.^2+grady.^2)); % magnitud del gradiente

% Campo vectorial perpendicular al gradiente sin normalizar
tx=-grady;  % tangente al gradiente
```

```
ty= gradx;
%tx=gradx;ty=grady;

% calcula Edge Tangent Flow
r=ventana; % radio de omega, ventana de integración
kk=1;
eta=1;  %ajustar bordes afilados
TXX=zeros(tamy,tamx); TYY=zeros(tamy,tamx);magETF=zeros(tamy,tamx);
barraProgreso = waitbar(0,'Espere␣por␣favor...');
% iteraciones
for it=1:iteraciones

for k=r+1:tamy-r
    for l=r+1:tamx-r
          sumx=0;
          sumy=0;
          for i=k-r:k+r %en una ventana tamaño rxr
            for j=l-r:l+r
            if( sqrt((j-l).^2+(i-k).^2)<=r )
                wm=0.5*(1+tanh(eta*(mag(i,j)-mag(k,l))));
                pp=tx(k,l)*tx(i,j)+ty(k,l)*ty(i,j); %producto punto tx.ty
                wd=abs(pp);
                TX= tx(i,j)*wm*wd;
                TY= ty(i,j)*wm*wd;
                if(pp<=0) TX=-1*TX; TY=-1*TY; end %voltea el vector
                sumx=sumx+TX;
                sumy=sumy+TY;
            end
            end
          end
          magETF(k,l)=sqrt(sumx * sumx + sumy * sumy);
          if(magETF(k,l)>0)
          TXX(k,l)=sumx/magETF(k,l); %componente x del nuevo ETF,normalizado
          TYY(k,l)=sumy/magETF(k,l); %componente y del nuevo ETF.
          end
    end
end

% reasigna valores para iterar
maxx=max(max(magETF));
magETF=255*magETF/maxx;  % [0 255]
tx=TXX;
ty=TYY;
mag=magETF;
waitbar(it/iteraciones,barraProgreso,sprintf('Procesando␣%d',it));
end
close(barraProgreso);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ordena kmedias de una imagen
function etiqOrdenada=kmediasOrdenada(label,gris)
m=max(max(label));
color=254-m;
vectorMedia=zeros(m,1);
for i=1:m
vectorMedia(i)=mean(gris(label==i));
```

```
end
[B,Idx] = sort(vectorMedia);
idxOrd=Idx;
etiqOrdenada=label;
for i=1:m
  idxOrd(Idx(i))=i;
end
for i=1:m
   etiqOrdenada(label==i)=idxOrd(i)+color;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Visualiza campo vectorial usando la convolución
% de línea de una imagen de ruido

% GRAYLIC is an internal command of the toolbox. It uses Regular LIC method
% implemented in internal Matlab commands
% to generate an intensity image.
% Usage:
% [LICIMAGE, INTENSITY,NORMVX,NORMVY] = GRAYLIC(VX, VY, ITERATIONS);
% VX and VY should contain X and Y components of the vector field. They
% should be M x N floating point arrays with equal sizes.
%
% ITERATIONS is an integer number for the number of iterations used in
% Iterative LIC method. use number 2 or 3 to get a more coherent output
% image.
%
% LICIMAGE returns an M x N floating point array containing LIC image
% INTENSITY returns an M x N floating point array containing magnitude
% of vector in the field
% NORMVX and NORMVY contain normalized (each vector is normalized to have
% the length of 1.0) components of the vector field

function [LICImage, intensity,normvx,normvy] = grayLIC(vx,vy, iterations)
[width,height] = size(vx);
LIClength = 7;
LICImage = zeros(width, height);
intensity = ones(width, height);
% Ruido blanco
noiseImage = zeros(width, height);
rand('state',0) % reset generador aleatorio
for i = 1:width
   for j = 1:height
      noiseImage(i,j)= rand;
   end
end

% Normaliza campo
normvx = zeros(width, height);
normvy = zeros(width, height);
for i = 1:width
   for j = 1:height
      l = sqrt( vx(i,j)^2 + vy(i,j)^2);
      intensity(i,j) = l;
      if l > 0
         normvx(i,j) = vx(i,j) / l;
```

```
1453          normvy(i,j) = vy(i,j) / l;
1454       end;
1455    end;
1456 end;
1457
1458 % Imagen LIC
1459 for m = 1:iterations
1460 for i = 1:width
1461    for j = 1:height
1462       stepCount = 1;
1463       sum = 0;
1464       x = i; y = j;
1465
1466       for k = 1:LIClength * 1 % integración hacia atrás
1467          xPast = x;
1468          yPast = y;
1469
1470          x = x + normvx(round(x),round(y));
1471          if x < 1 break;end;
1472          if x > width break; end;
1473
1474          y = y + normvy(round(x),round(y));
1475          if y < 1 break;end;
1476          if y > height break; end;
1477
1478          if (round(x) ~= round(xPast)) || (round(y) ~= round(yPast))
1479             stepCount = stepCount + 1;
1480             sum = sum + noiseImage(round(x),round(y));
1481          end;
1482        if stepCount > LIClength
1483           break;
1484        end;
1485
1486       end;
1487
1488       x = i; y = j;
1489
1490       for k = 1:LIClength * 1 % integración adelante
1491          xPast = x;
1492          yPast = y;
1493          x = x - normvx(round(x),round(y));
1494          if x < 1 break;end;
1495          if x > width break; end;
1496
1497          y = y - normvy(round(x),round(y));
1498          if y < 1 break;end;
1499          if y > height break; end;
1500          if (round(x) ~= round(xPast)) || (round(y) ~= round(yPast))
1501             stepCount = stepCount + 1;
1502             sum = sum + noiseImage(round(x),round(y));
1503          end;
1504        if stepCount > LIClength * 2
1505           break;
1506        end;
1507       end;
1508       LICImage(i,j) = sum / stepCount;
```

```
1509    end;
1510 end;
1511
1512 %LICImage = imadjust(LICImage);  % Adjusta rango
1513 noiseImage = LICImage;
1514 end;
```

Anexo 2. Código Arduino

Código 2 Programa que se graba en la tarjeta Arduino

```
1  /*-------------------------------------------------------------------------
2    Programa que recibe n puntos por el puerto serie de forma asíncrona,
3    los n puntos son los puntos de control de una curva de Bézier de
4    grado n-1. Este programa reconstruye la curva Bézier y la curva
5    resultante se traza con un robot cartesiano de tres grados
6    de libertad: X, Y, Z.
7    Diciembre de 2018, Ingeniería de Sistemas, SEPI-ESIME-Zacatenco
8    -------------------------------------------------------------------------
9  */
10 #include <math.h>
11 int stepperPin0 = 4; // x paso, esta configuración es del CNC shield
12 int stepperPin1 = 3; // y paso
13 int stepperPin2 = 2; // z paso
14 int dirPin0 = 7; // x sentido, Pines de conexión para motores a paso
15 int dirPin1 = 6; // y sentido
16 int dirPin2 = 5; // z sentido
17 int Enable=8;
18 int tSetup = 15; // Tiempo en us de cambio de señal en los motores
19 long L; // L = 2*n puntos de control
20                         // L = n*(x+y)
21 int color; // posición de pintura
22 int npts; // no. puntos de control
23 long longitudPintada;
24 int k=0; // Contador de datos
25 double xi[20], yi[20]; // Coordenadas X,Y, de los puntos de control,
26                         // 20 es el máximo orden de una curva Bézier
27 long datos[42]; // Guarda datos leídos, máximo 20*2 números +
28                         // 1 byte de tamaño + 1 byte de color
29 bool trabajando = false;
30 bool siTomoPintura=true;
31 bool tomarPinturaInicio=true;
32 double xx0, xx1, yy0, yy1; // Puntos para algoritmo de Bresenhan
33 double xPint, yPint; // Coord. de la pintura
34 double xPintAnterior, yPintAnterior; // Coord. anteriores de la pintura
35 double xAgua = 5600, yAgua = -7520; // Coordenadas del agua
36 double xAnterior, yAnterior; // último punto trazado de una curva
37 double tt = 10; // Parámetro t= no. de subdivisiones de la curva B
38 int m0 = 9; // Pines para configuración de
39 int m1 = 10; // de velocidad del DRV8825
40 int m2 = 11; // por hardware
41 double P[20];
42 String vectorCaracteres;
43 char vectorChar[10]; // 9 digitos mas el signo de un numero entero
```

```
boolean TransmisionCompleta = false;
long numero = 0;
int v=10;

static const double TablaFactorial[20] = {
      1.0,
      1.0,
      2.0,
      6.0,
      24.0,
      120.0,
      720.0,
      5040.0,
      40320.0,
      362880.0,
      3628800.0,
      39916800.0,
      479001600.0,
      6227020800.0,
      87178291200.0,
      1307674368000.0,
      20922789888000.0,
      355687428096000.0,
      6402373705728000.0,
      121645100408832000.0,
};


void setup() {
  /* ---------------------------------------------------------------
                    Se ejecuta una vez
     ---------------------------------------------------------------
  */
  pinMode(dirPin0, OUTPUT); // Configuración de pines para motores
  pinMode(stepperPin0, OUTPUT);
  pinMode(dirPin1, OUTPUT);
  pinMode(stepperPin1, OUTPUT);
  pinMode(dirPin2, OUTPUT);
  pinMode(stepperPin2, OUTPUT);
  pinMode(Enable, OUTPUT);
  digitalWrite(Enable, false);
  Serial.begin(9600); // Inicializa serial
  vectorCaracteres.reserve(20);
}

void loop() {
  /* ---------------------------------------------------------------------
                    Se ejecuta repetidamente
     ---------------------------------------------------------------------
  */
 if (TransmisionCompleta)
 {
   vectorCaracteres.toCharArray(vectorChar,10);
   numero = atol(vectorChar);
   vectorCaracteres = ""; //Limpia el String
   TransmisionCompleta = false; //Limpia la bandera
```

```
100    k++; // k máximo es 42
101    if(k==1){L=numero; npts = L / 2;}
102    if(k==2){color=numero;}
103    if(k>2){datos[k-3]=numero;}
104    if(L!=255){Serial.write(k);} // Responde a la PC
105  }
106
107
108  // TERMINA TODO, se deja a condiciones iniciales
109  if (L == 255) // Termina todo, L=255 es un valor particular
110  { sube(9500);
111    if(siTomoPintura==false)
112    { // va a coordenada de origen de la imagen
113      siTomoPintura = true;
114      tomarPinturaInicio = true;
115      /*Enjuague de pincel*/
116      lineaDDA(xx1, yy1, xAgua, yAgua, 80,false);
117      baja(9500); delay(100);
118      sube(9500); delay(100);
119      baja(9500); delay(100);
120      // 3 círculos de radio = 400*6.25um=2.5mm
121      for (int j = 0; j < 4; j++) {
122        for (int i = 0; i < 100; i++)
123        { float tt0 = (float)i * 2 * PI / 100;
124          float tt1 = (float)(i + 1) * 2 * PI / 100;
125          float radio = 400;
126          float xt0 = radio * cos(tt0);
127          float xt1 = radio * cos(tt1);
128          float yt0 = radio * sin(tt0);
129          float yt1 = radio * sin(tt1);
130          lineaDDA(xt0, yt0, xt1, yt1, 88,false);
131        }
132      }
133      sube(9500);
134      lineaDDA(xAgua, yAgua, 0, 0, 80,false); // punto inicial
135    }
136    k = 0;
137    L=0;
138    longitudPintada = 0;
139  } // Fin de Termina todo
140
141  // POSICIONES DE LA PINTURA Godete
142  // Unidades de medida: pasos
143  // 1600 pasos = 1 cm
144  if(color==250) // color 1
145  { xPint=0; yPint=-4640;} //-2.9cm
146  if(color==251) // color 2
147  { xPint=-5600; yPint=-7520;}
148  if(color==252) // color 3
149  { xPint=-5600; yPint=-14080;}
150  if(color==253) // color 4
151  { xPint=0; yPint=-17280;}
152  if(color==254) // color 5
153  { xPint=5600; yPint=-14080;}
154
155  // TRANSMISIÓN DE CURVA FINALIZADA
```

```
156 //Si k==L+2 se procede a realizar el trazo
157 if ((k == (L + 2)) && (L > 0) && (L<43))
158 // 0 para almacenar el segmento siguiente en datos[k]
159 { k = 0;
160  // No se reciben datos mientras "trabajando" sea verdadero
161   trabajando = true;
162   /* Puntos de control */
163   for (int i = 0; i < npts ; i++)
164   { xi[i] = (double)(datos[i]);
165     yi[i] = (double)(datos[i + npts]);
166   }
167
168  // TOMA PINTURA INICIAL UNA VEZ ====
169  if(tomarPinturaInicio){
170  // va a la coordenada de pintura desde el origen
171   lineaDDA(0,0,xPint, yPint,10,false);
172   baja(9500); delay(100);
173   sube(9500); delay(100);
174   baja(9500); delay(100); // se queda abajo
175   tomarPinturaInicio = false;
176   xPintAnterior = xPint;
177   yPintAnterior = yPint;}
178
179  // VA POR + PINTURA?: 1)longitud y 2) colores desiguales o o o o o
180  // si longitud es mayor a 10cm: 0.1m / 6.25e-6m = 16,000
181  if(longitudPintada>16000 || xPintAnterior!=xPint || yPintAnterior!=yPint )
182  {longitudPintada=0;
183   siTomoPintura=true;
184   sube(9500);// 9.5mm
185   // toma pintura, se queda en xPint,yPint, y abajo
186   lineaDDA(xAnterior, yAnterior, xPint, yPint, 10,false);
187   baja(9500);}
188
189  // POSICIÓN INICIAL DE UNA CURVA - - - - - - - - - - - -
190  if(siTomoPintura) {
191   sube(9500); // desde coordenada de la pintura
192   lineaDDA(xPint, yPint, xi[0], yi[0],10,false);
193   baja(9500);}
194  else { // desde el final del trazo anterior
195   sube(5000);
196   lineaDDA(xAnterior, yAnterior, xi[0], yi[0], 10,false);
197   baja(5000);}
198
199  // TRAZA UNA CURVA ------------------------------------------------
200  // Los puntos de control xi,yi se procesan en Bezier(T,bool XoY)
201  double T, T1; if(npts>6){tt=100;v=300;}else{tt=10;v=10;}
202  for (int t = 0; t < tt; t++)
203  { T = (double)t / (double)tt; T1 = (double)(t + 1) / (double)tt;
204    xx0 = Bezier(T, true);
205    xx1 = Bezier(T1, true);
206    yy0 = Bezier(T, false);
207    yy1 = Bezier(T1, false);
208    // Imprime la línea de la t-ésima parte del parámetro t
209    lineaDDA(xx0, yy0, xx1, yy1, v,true);
210  }
211
```

```
212  // datos que servirán para la futura curva
213  siTomoPintura=false;
214  xAnterior=xi[npts-1]; // último punto de cada segmento
215  yAnterior=yi[npts-1]; // último punto de cada segmento
216  xPintAnterior = xPint; // coord. x de pintura anterior
217  yPintAnterior = yPint; // coord. y de pintura anterior
218  trabajando = false;
219  Serial.write(255);// señala fin de trazado
220 } // Termina segmento
221
222 } // Fin de loop ============================================================
223
224
225 /*-------------------------------------------------------------------
226  * Guarda un string ej. '-458'
227  -------------------------------------------------------------------*/
228 void serialEvent()
229 {
230  while (Serial.available()&& trabajando == false)
231  {
232   char CharEntrada = Serial.read(); //Lee un byte del puerto serial
233   vectorCaracteres += CharEntrada; //Agrega el char anterior al string
234   if (CharEntrada == '\n') { //Si se detecta un fin de linea
235    TransmisionCompleta = true; //Fin de número
236   }
237  }
238 }
239
240
241 /*-------------------------------------------------------------------
242   Trazo de línea usando el algoritmo de Bresenhan
243   Cada paso, estando configurado a 1/32 el DRV8825,
244   equivale a ~6.25 micras.
245   Se activan los motores para X,Y
246 -------------------------------------------------------------------*/
247 void lineaDDA(double x0, double y0, double x1, double y1, int v, bool pintando
      )
248 { x0=round(x0); x1=round(x1); // (-) para que coincida c máquina.
249  y0=-round(y0); y1=-round(y1); // Se quitan los decimales.
250
251  long X0, Y0, X1, Y1;
252  double dx, dy, longitud_linea;
253  double x_inc, y_inc, x = x0, y = y0;
254  dx = (x1 - x0);
255  dy = (y1 - y0);
256
257  if (abs(dx) > abs(dy))
258  {
259   longitud_linea = abs(dx);
260  }
261  else
262  {
263   longitud_linea = abs(dy);
264  }
265
266  x_inc = dx / longitud_linea;
```

```
267  y_inc = dy / longitud_linea;
268
269  for (long j = 0; j < longitud_linea; j++)
270  {
271    X0 = round(x);
272    Y0 = round(y);
273
274    x += x_inc;
275    y += y_inc;
276
277    X1 = round(x);
278    Y1 = round(y);
279
280    if(pintando){ longitudPintada++;}
281
282    // Activa motores a paso con X0,X1,Y0,Y1
283    if (X0 == X1 && Y0 < Y1) { //1
284      digitalWrite(dirPin0, true);
285      delayMicroseconds(tSetup);//setupTime
286      digitalWrite(stepperPin0, HIGH);
287      delayMicroseconds(v);
288      digitalWrite(stepperPin0, LOW);
289      delayMicroseconds(v);
290    }
291    if (X0 > X1 && Y0 < Y1) { //2
292      digitalWrite(dirPin0, true);
293      digitalWrite(dirPin1, false);
294      delayMicroseconds(tSetup);//setupTime
295      digitalWrite(stepperPin0, HIGH);
296      digitalWrite(stepperPin1, HIGH);
297      delayMicroseconds(v);
298      digitalWrite(stepperPin0, LOW);
299      digitalWrite(stepperPin1, LOW);
300      delayMicroseconds(v);
301    }
302    if (X0 > X1 && Y0 == Y1) { //3
303      digitalWrite(dirPin1, false);
304      delayMicroseconds(tSetup);//setupTime
305      digitalWrite(stepperPin1, HIGH);
306      delayMicroseconds(v);
307      digitalWrite(stepperPin1, LOW);
308      delayMicroseconds(v);
309    }
310    if (X0 > X1 && Y0 > Y1) { //4
311      digitalWrite(dirPin0, false);
312      digitalWrite(dirPin1, false);
313      delayMicroseconds(tSetup);//setupTime
314      digitalWrite(stepperPin0, HIGH);
315      digitalWrite(stepperPin1, HIGH);
316      delayMicroseconds(v);
317      digitalWrite(stepperPin0, LOW);
318      digitalWrite(stepperPin1, LOW);
319      delayMicroseconds(v);
320    }
321    if (X0 == X1 && Y0 > Y1) { //5
322      digitalWrite(dirPin0, false);
```

```
323     delayMicroseconds(tSetup);//setupTime
324     digitalWrite(stepperPin0, HIGH);
325     delayMicroseconds(v);
326     digitalWrite(stepperPin0, LOW);
327     delayMicroseconds(v);
328    }
329    if (X0 < X1 && Y0 > Y1) { //6
330     digitalWrite(dirPin0, false);
331     digitalWrite(dirPin1, true);
332     delayMicroseconds(tSetup);//setupTime
333     digitalWrite(stepperPin0, HIGH);
334     digitalWrite(stepperPin1, HIGH);
335     delayMicroseconds(v);
336     digitalWrite(stepperPin0, LOW);
337     digitalWrite(stepperPin1, LOW);
338     delayMicroseconds(v);
339    }
340    if (X0 < X1 && Y0 == Y1) { //7
341     digitalWrite(dirPin1, true);
342     delayMicroseconds(tSetup);//setupTime
343     digitalWrite(stepperPin1, HIGH);
344     delayMicroseconds(v);
345     digitalWrite(stepperPin1, LOW);
346     delayMicroseconds(v);
347    }
348    if (X0 < X1 && Y0 < Y1) { //8
349     digitalWrite(dirPin0, true);
350     digitalWrite(dirPin1, true);
351     delayMicroseconds(tSetup);//setupTime
352     digitalWrite(stepperPin0, HIGH);
353     digitalWrite(stepperPin1, HIGH);
354     delayMicroseconds(v);
355     digitalWrite(stepperPin0, LOW);
356     digitalWrite(stepperPin1, LOW);
357     delayMicroseconds(v);
358    }
359
360  }
361
362 }
363
364 /*-----------------------------------------------
365  Activa el motor Z, resolución 1um
366 -----------------------------------------------*/
367 void sube(int altura)
368 {
369  for (int i = 0; i < altura; i++) {
370   digitalWrite(dirPin2, false);
371   delayMicroseconds(tSetup);// t de estabilización
372   digitalWrite(stepperPin2, HIGH);
373   delayMicroseconds(20); // cuando m00= H H H,
374   digitalWrite(stepperPin2, LOW);
375   delayMicroseconds(20); //depende del potenciómetro
376  }
377 }
378
```

```
379
380 /*-----------------------------------------------
381  Activa el motor Z, resolución 1um
382 -----------------------------------------------*/
383 void baja(int altura)
384 {
385  for (int i = 0; i < altura; i++) {
386   digitalWrite(dirPin2, true);
387   delayMicroseconds(tSetup);//setupTime
388   digitalWrite(stepperPin2, HIGH);
389   delayMicroseconds(20); // cuando m00= H H H
390   digitalWrite(stepperPin2, LOW);
391   delayMicroseconds(20);
392  }
393 }
394
395 /*-----------------------------------------------
396  Función Bézier, procesa los puntos de control
397 -----------------------------------------------*/
398 double Bezier(double t, bool xoy) {
399
400  double a = 1.0 - t;
401  double B=0.0;
402  int nn;
403
404  for(int i=0;i<npts;i++)
405  {
406   nn=npts-i-1;
407   P[i] = factorial(npts-1)/(factorial(i)*factorial(nn));
408   if (xoy){ B += double((double)(xi[i])*P[i]*power(t,i)*power(a,nn)); }
409   else{ B += double((double)(yi[i])*P[i]*power(t,i)*power(a,nn)); }
410  }
411
412  return B;
413 }
414
415
416 /*-----------------------------------------------
417  Potencia
418 -----------------------------------------------*/
419 double power(double base, int e) {
420  double res = 1.0;
421  for(int i=0; i<e; i++)
422   res *= base;
423  return res;
424 }
425
426 /*-----------------------------------------------
427  Factorial hasta 20
428 -----------------------------------------------*/
429 double factorial(int n) {
430   return TablaFactorial[n];}
```

Anexo 3. Guía de uso

Esta guía de uso tiene como finalidad dar a conocer el funcionamiento básico de la interfaz gráfica de usuario que se detalló en la Actividad de investigación previa 1, en el capítulo 3.

Requerimientos técnicos para el uso del programa

- Tener instalado el software Matlab en su versión R2018b o superior.
- Se puede ejecutar en máquinas con sistema operativo Windows

Ejecución del programa

Para lanzar el programa se abre Matlab y se ejecuta el código RegionesTrazos.m que se adjunta en el Anexo 1 de esta tesis mediante el botón Run del software Matlab como se muestra en la Figura A1 circulado en rojo.

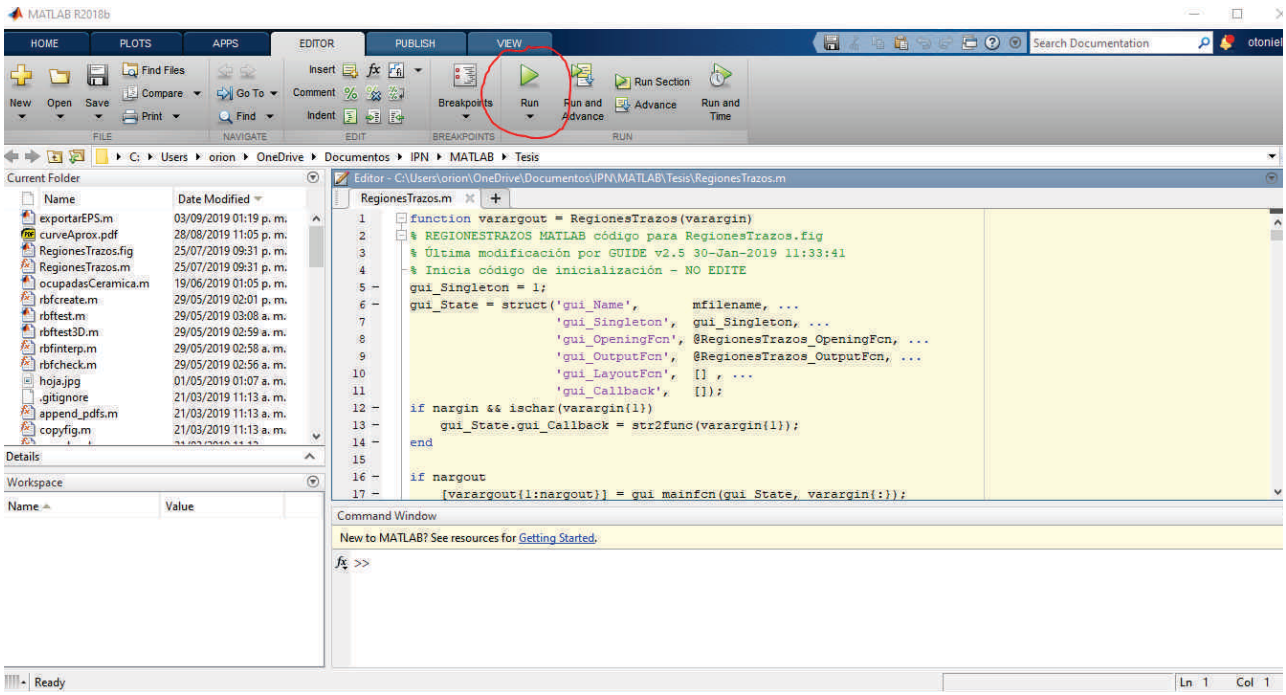


Figura A1. Lanzamiento del programa en Matlab

Una vez abierto el programa el usuario puede utilizarlo a su criterio. Un posible ejemplo de uso se muestra en la Figura 3.7, donde se muestran los botones y controles que el usuario puede seleccionar para realizar una pintura robótica. En la Actividad de investigación previa 1 del capítulo 3 se detallan los elementos de la interfaz gráfica y en la Figura 3.6 se muestran las operaciones que se pueden realizar con la interfaz gráfica de usuario.

Para conocer ejemplos específicos de uso de la interfaz gráfica se pueden consultar las actividades de investigación siguientes. En la Actividad de investigación 4 se muestra un ejemplo de diseño de campo vectorial automático, en la Actividad de investigación 5 se

observa un ejemplo de diseño de campo vectorial interactivo, en la Actividad de investigación 6 se muestra un ejemplo de generación de trazos y en la Actividad de investigación 7 se presenta un ejemplo de detección de bordes, finalmente en la Actividad de investigación 8 se muestra un ejemplo de aplicación de pintura. Todas las actividades mencionadas muestran las operaciones realizadas usando la interfaz gráfica de usuario diseñada.

APORTACIONES CIENTÍFICAS

Tabla B. Aportaciones científicas

No.	Tipo de producto	Título	Autores	Journal y/o Instituto
1	Derechos de autor	Algoritmos para robot: trazos estilizados	Dr. Alfredo Cruz Orea, Dr. Arturo Domínguez Pacheco, Dra. Claudia Hernández Aguilar, M.C. Otoniel Igno Rosario	Instituto Nacional del Derecho de Autor
2	Artículo	Interactive system for painting artworks by regions using a robot	M.C. Otoniel Igno Rosario, Dra. Claudia Hernández Aguilar, Dr. Arturo Domínguez Pacheco, Dr. Alfredo Cruz Orea	Robotics and Autonomous Systems

Aportación científica 1. Certificado de Derechos de Autor

2573
LVII

CERTIFICADO

Registro Público del Derecho de Autor

Para los efectos de los artículos 13, 162, 163 fracción I, 164 fracción I, 168, 169, 209 fracción III y demás relativos de la Ley Federal del Derecho de Autor, se hace constar que la **OBRA** cuyas especificaciones aparecen a continuación, ha quedado inscrita en el Registro Público del Derecho de Autor, con los siguientes datos:

AUTORES:	CRUZ OREA ALFREDO DOMINGUEZ PACHECO FLAVIO ARTURO HERNANDEZ AGUILAR CLAUDIA IGNO ROSARIO OTONIEL
TITULO:	ALGORITMOS PARA ROBOT: TRAZOS ESTILIZADOS
RAMA:	PROGRAMAS DE COMPUTACION
TITULAR:	INSTITUTO POLITECNICO NACIONAL (CON FUNDAMENTO EN EL ARTICULO 83 DE LA L.F.D.A.)

Con fundamento en lo establecido por el artículo 168 de la Ley Federal del Derecho de Autor, las inscripciones en el registro establecen la presunción de ser ciertos los hechos y actos que en ellas consten, salvo prueba en contrario. Toda inscripción deja a salvo los derechos de terceros. Si surge controversia, los efectos de la inscripción quedarán suspendidos en tanto se pronuncie resolución firme por autoridad competente.

Con fundamento en los artículos 2, 208, 209 fracción III y 211 de la Ley Federal del Derecho de Autor; artículos 64, 103 fracción IV y 104 del Reglamento de la Ley Federal del Derecho de Autor; artículos 1, 3 fracción I, 4, 8 fracción I y 9 del Reglamento Interior del Instituto Nacional del Derecho de Autor, se expide el presente certificado.

Número de Registro: **03-2018-082111170300-01**

México D.F., a 24 de agosto de 2018

EL DIRECTOR DEL REGISTRO PÚBLICO DEL DERECHO DE AUTOR

JESUS PARETS GOMEZ

INSTITUTO NACIONAL DEL DERECHO DE AUTOR
DIRECCIÓN DEL REGISTRO PÚBLICO DEL DERECHO DE AUTOR

CULTURA
SECRETARÍA DE CULTURA

INDAUTOR

Aportación científica 2. Artículo

Robotics and Autonomous Systems 121 (2019) 103263

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Interactive system for painting artworks by regions using a robot

Otoniel Igno-Rosario [a,*], Claudia Hernandez-Aguilar [a], Alfredo Cruz-Orea [b], Arturo Dominguez-Pacheco [a]

[a] National Polytechnic Institute of Mexico, Sepi-Esime, Zacatenco, Unidad Profesional "Adolfo López Mateos". Col. Lindavista. Mexico City, C.P. 07738, Mexico

[b] CINVESTAV – IPN, Physics Department, P.O. Box 14-740, Mexico City, C.P. 07360, Mexico

ARTICLE INFO

Article history:
Received 17 June 2019
Accepted 15 August 2019
Available online 19 August 2019

Keywords:
Bézier curve
Robotic artwork
Scattered interpolation
Interactive painting

ABSTRACT

In this work we present an interactive system capable of producing realistic artworks with acrylic paint based on a cartesian robot. This system focuses on painting artworks by regions, and can be applied for example in the painting of objects such as fruits, flowers, leaves and other individual objects.

We have divided the development of the proposed system in three interactive stages: (1) interactive segmentation of work regions, (2) interactive designing of the field for orienting the brush strokes by tracing curves manually in each region and interpolating the curves to generate the vector field, and (3) painting by regions with the field.

With our system it is possible to reproduce interactively an image producing pleasant results. The experimental results are presented by painting an apple with three main regions, for this we have utilized a realistic pictorial style and a monochromatic palette of 5 colors.

1. Introduction

Humans have been intrigued by the possibility of constructing mechanical devices that mimic their behavior. One of the topics currently being investigated is trying to imitate the painting process by using robots. Several robotic systems have been designed for this purpose, such as AARON [1], e-David [2], Cloudpainter [3], etc.

In this paper we report a system capable of producing realistic artworks with acrylic paint and is based on a pen plotter, where we investigate painting activity by means of computational models. This system focuses on the painting of objects by regions, and can be applied for example in the painting of objects such as fruits, flowers, leaves and other individual objects.

Various studies related to robots that artists use for producing artworks can be found in the present literature. Harold Cohen [1] with the AARON system is one of the pioneers in the use of technology in art. The AARON system focuses on creating art almost autonomously using a robot. Aguilar and Lipson [4] reported a robotic system that produced acrylic paintings on canvas using machine learning algorithms. Tresset and Leymarie [5] created the robotic installation Paul, to draw sketches of people by extracting salience lines from image. Deussen et al. [6] developed e-David for create strokes using the line integral convolution and the image gradient to produce stroke paths that guide the brush strokes. Luo et al. [7] used a robot arm to make paintings by superimposing layers of paint. Song et al. [8] used a robotic arm to draw with a semi-autonomous pen on an arbitrary surfaces. Scalera et al. [9] reported a robotic arm used to paint watercolors using different non photorealistic rendering techniques in order to produce painting trajectories. Karimov et al. [10] employed a cartesian robot with a mixing device connected to the brush for the paint application. The techniques they used were the gradient of the image to guide the strokes and the generation of seed points with a grid.

We can also mention the RobotArt competition, an on-line contest in which the best paintings were selected from more than 100 artworks created by robots during 2018 [11].

Many of the algorithms used to produce robotic painting were obtained from the non-photorealistic rendering (NPR) technique. In the bibliographic review of Kiprianidis et al. [12] several methods can be found on this field of computational research. One of these approaches, the brush strokes method, produces paintings on a virtual canvas and can be used for physical painting with robots. Some research works on NPR related to robotic painting are the following. Cabral and Leedom [13] proposed a framework that can be used to generate strokes paths through the line integral convolution. Kang et al. [14] presented a technique that both smooths the gradient and preserves the edge flow of a field, in this case the image abstraction technique can be used to generate smoother brush strokes when using the gradient. Hertzmann [15] proposed a general scheme for painterly rendering that approximates an image by refining it in subsequent steps of smaller brush

* Corresponding author.
E-mail address: otonieligno@gmail.com (O. Igno-Rosario).

https://doi.org/10.1016/j.robot.2019.103263
0921-8890/

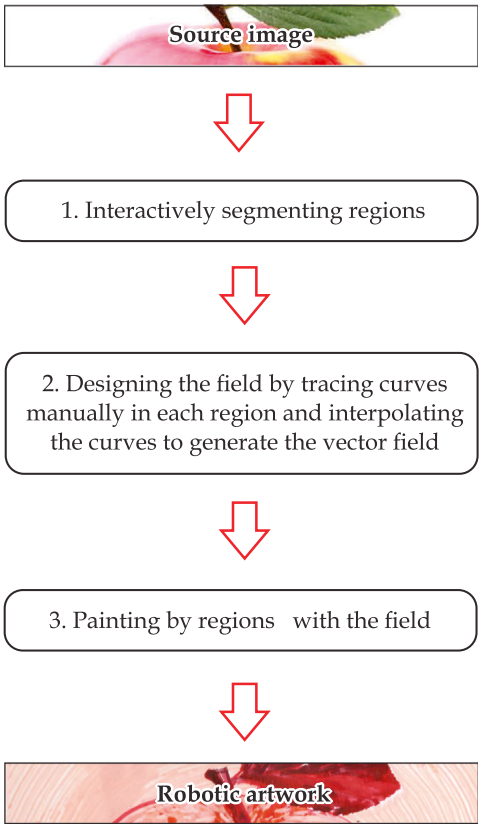


Fig. 1. Overview of the proposed framework.

strokes, this idea can be applied in physical painting by using brushes of various sizes. Hays and Essa [16] globally interpolated the strongest gradients by using radial basis functions, obtaining strokes simulations with styles like Van Gogh, this idea was used by Lindemeier et al. [2] where they interpolate the missing gradient information.

In the case of works where the user interaction is used to determine the orientation of the brush strokes, the following examples can be mentioned. Haeberli [17] allowed the user to determine the orientation of each brush stroke using the direction of movement. Olsen et al. [18] designed a graphical user interface for regionally assigning the desired vector field, either using fluid fields or interpolating the strongest gradient using radial base functions.

In this paper we present a new framework for addressing the painting activity that differs from those previously reported in the bibliography in the following aspects. We developed a novel interactive system for producing robotic artworks by regions where interactively we assigning the vector field for orientation of the brush strokes. In such a case, the approach for generating the vector field is based on the interpolation of the tangent vector of Bézier curves. Also we present an interactive segmentation method, based on imprecise contour drawing where the refining of the desired contour is made by approximating Bézier curves in order to obtain aesthetic results.

Therefore, all the work is divided into three stages as shown in Fig. 1, and it is organized in this paper as follows. We present the interactive segmentation of regions method in Section 2. Section 3 illustrates how to generate the field for orienting the brush strokes through the interactive field design. In Section 4 we describe the painting machine. Experimental results are depicted in Section 5 and the Section 6 provides the conclusion.

2. Interactive segmentation of regions

The image segmentation is a commonly used technique in digital image processing to partition an image into regions of pixels with similar characteristics, such as color or texture. There is a great amount of research on image segmentation. This technique seeks to simplify the representation of an image to make it more suitable for analysis. The work of Vantaram [19] shows several methods of this technique.

In this work we choose a heuristic approach for controlling both the region of painting and the boundaries of the region of interest. The idea is to interactively segment an object located in the foreground of the scene and design the field for orienting the brush strokes in a customized way, while the background can be processed with the classical techniques based on the gradient.

Some of the notable interactive methods in the literature are the following: imprecise tracing of desired contour [20–22], marking-up of parts of object or background [23–25], use of bounding box [26,27]. Two of the image editing programs, Photoshop and Corel Draw, use similar techniques for this purpose. In Corel Draw one can segment an image with the Shape tool, manipulating the nodes surrounding an object and adjusting the curve between the nodes using cubic Bézier functions. In Photoshop it is possible to use the tool Magnetic Lasso to segment an object selecting points on the contour, where this tool looks for the optimal global route from a starting pixel to a target pixel, the details of the algorithm used in this method are in [28].

We present an interactive segmentation, based on imprecise contour drawing where the refining of the desired contour is made by Bézier curves in order to obtain aesthetic results and to approximate the contours of the objects to be segmented. Fig. 2 shows our proposal, which consists of 4 steps: (1) Consecutive clicks on the contour of the region of interest, to define the painting area, (2) Corner detection for curve segmentation, the corner points are break points that allow decomposing the contour into segments, then, these segments are fitted with Bézier curves, (3) Bézier curve fitting with $n < 20$ degree, this fitting is chosen by the user, and (4) Region segmentation. All of these topics are discussed below.

In step 1, significant points on the contour are selected with mouse clicks. The procedure used in this case is similar to that of Polygon Lasso tool in Photoshop where the polygon vertices are selected and these vertices are joined by straight lines. For the purpose of using these vertices in the Bézier approximation, it is necessary that the points are selected sequentially forming simply connected regions. In the example of Fig. 2(1), 44 points were selected consecutively in order to segment the apple.

In step 2, the corner detection algorithm proposed by Rosenfeld and Weszka [29] is used. This is one of the simplest methods based on the contour [30]. Other contour corner detection techniques can be found in the bibliographic review of Abe et al. [31]. The Rosenfeld's method is as follows.

Let $P = p_1, \ldots, p_l$ be the set of contour points selected sequentially. Being l the number of contour points chosen at the user's discretion, where that number significantly samples the contour, and x_i, y_i are the coordinates of the point p_i in the image.

For detection of the angle, the values of the curvature K are calculated for each i by means of the formula $K = c_{ik}(p_i)$, for the chord length $k \geq 1$, where

$$c_{ik} = \frac{a_{ik}.b_{ik}}{|a_{ik}||b_{ik}|} \tag{1}$$

being c_{ik} the cosine of the angle between the vectors a_{ik} and b_{ik} in p_i, and $a_{ik} = (x_i - x_{i+k}, y_i - y_{i+k})$, $b_{ik} = (x_i - x_{i-k}, y_i - y_{i-k})$. The chord k is the distance from p_i in pixels following the P contour. The values of c_{it} are averaged for each point i, where $t = 1, \ldots, k$.

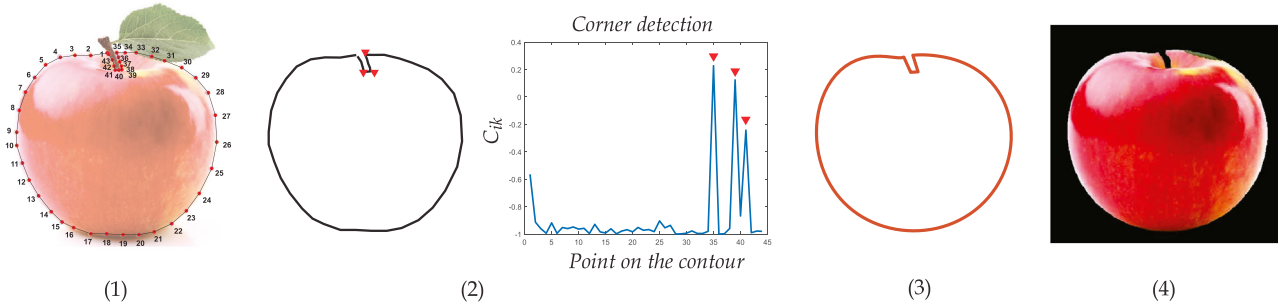


Fig. 2. Interactive segmentation requires (1) Consecutive clicks on the contour of the region of interest, (2) Corner detection with a threshold of the c_{ik} peaks prominence, in this example threshold = 0.5 was established, (3) Fitting of Bezier curve $c(t)$ of degree n, ($n = 9$ in this case) (4) Region segmentation.

In the example of Fig. 2(2) corner detection with a threshold = 0.5 of the c_{ik} peaks prominence was performed, being $k = 1$. For such a case, three corners were detected as shown in Fig. 2(2) and in the corresponding curvature graph.

In step 3, the fitting is performed manually, although it could also be done automatically by computing the squared distance between each of the points on the boundary and its corresponding points on the parametric curve and setting a threshold to the obtained error. The boundary points of each object are automatically divided into segments, the division is based on the corner points. So, each of the segment is manually fitted to Bezier curve of grade $1 < n < 20$ until the desired aesthetic result of the curve is obtained. Generally a value of n less than 10 achieves a pleasant result. The Bézier approximation is as follows.

Grade n Bezier's curves are parametric curves and can be defined as [32]

$$c(t) = \sum_{j=0}^{n} B_j^n(t) p_j \tag{2}$$

where $t \in [0, 1]$, p_j are the points that control the shape of the Bézier curve and $B_j^n(t)$ are the Bernstein polynomials of degree n which are given by

$$B_j^n(t) = \binom{n}{j} t^j (1-t)^{n-j} \tag{3}$$

and $\binom{n}{j} = \frac{n!}{j!(n-j)!}$, where by convention, $0! = 1$.

Now let us suppose that we have the set of points $q_i = (x_i, y_i)$, $i = 0, \ldots, m$ and that we want to obtain the Bézier curve of degree n that better approximates the set of points in the sense of the least squares:

$$E = \sum_{i=0}^{m} \left(\sum_{j=0}^{n} B_j^n(t_i) p_j - q_i \right)^2 \tag{4}$$

where the choice of nodes $\{t_0, \ldots, t_m\}$, which establish the behavior of the resulting curve by approximating $\{q_0, \ldots, q_m\}$, is done considering a smooth centripetal acceleration [33], i.e. based on the length of the chord $\delta t_i = k\sqrt{\|\delta q_i\|}$, where $\|\delta q_i\|$ is the Euclidean distance between two consecutive points assuming that the curve is parameterized approximately by its arc length, being k a scale constant.

The pseudo-inverse matrix of B that minimizes Eq. (4) is $B^+ = (B^T B)^{-1} B^T$, where B^T denotes transposition, and A^{-1} means the inverse of A. Hence, unknown control points can be found with

$$p_j = B^+ q_i \tag{5}$$

These p_j values are replaced in Eq. (2) to obtain the Bézier curve $c(t)$ of degree n that better approximates the set of points q_i.

In the example of Fig. 2(3), 4 segments were approximated with grade 9 Bézier curves.

In step 4, after having the best contour adjustment then we proceed to segment the region that encloses that contour using the line Bresenham algorithm [34] in order to convert the corresponding parametric curve between points $c(t_i)$ and $c(t_{i+1})$ of the curve into a binary contour. Then we proceed to fill the region enclosed by the binary contour and segment the region. An example of this method of segmentation is shown in Fig. 2(4) for an image size of 550 × 550 pixels.

The approach we propose produces two useful results for applying robotic painting: the working region where the field will be designed to orient the brushstrokes, and the stylized contour by means of the Bézier curves; the contour in this case will be used to give a final touch to the painting by reaffirming its contour, that is, the boundary is painted at the end, after painting the regions.

3. Interactive field design for orienting the brush strokes

To design the interactive field we propose that the user enter a few lines manually, following the texture of the region, or according to the user's preference when there is no texture. This approach would allow the working region to overcome the problem of brightness and shadows due to the illumination on the object. Afterwards, the traced curves will be automatically interpolated to generate the vector field.

The interpolation of the curves was performed using the scattered data interpolation technique. This method consists of constructing a continuous function that interpolates a set of unorganized samples. An example of the use of scattered data interpolation occurs in meteorology where the weather measurements are available from irregularly located observation stations.

Before carrying out interpolation, it is necessary to make an adjustment to the manual curve in order to improve the smoothness as follows.

3.1. Regularization of hand traces

The manually scribbled curve can be approximated to a Bezier curve of degree n for smoothing the curve and subsequently performing the interpolation, i.e. generating the vector field.

The Bézier curves are part of the splines family, which includes among others, the cubic splines, B-splines, Beta-splines, Hermite splines, and Bézier splines [35].

In this work we used Bézier curves due to the overall control of their shape, which generates more free strokes.

Manual strokes have irregular edges, which if interpolated with the method of natural neighbors would produce an irregular field. For this reason, the strokes are approximated with Bézier

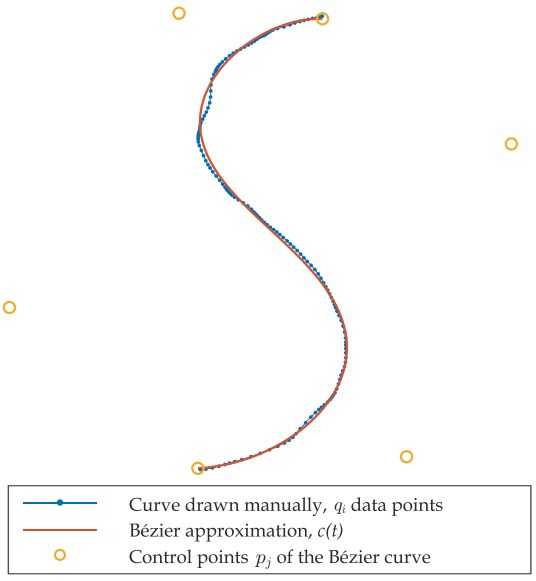


Fig. 3. Manually scribbled curve approximated to a Bézier curve $c(t)$ of degree $n = 5$.

curves which are smooth and in this way the field generated via the natural neighbor interpolation will produce a smoother field.

Considering the curve drawn manually as the set of points $q_i = (x_i, y_i)$, $i = 0, \ldots, m$, the Bézier curve $c(t)$ of degree n that better approximates the set of points in the sense of the least squares can be obtained with Eqs. (5) and (2).

The fitting can automatically be refined via squared distance between each of the points on the boundary and its corresponding points on the parametric curve, but we performed it manually for better personalization. Fig. 3 shows an example of a manually scribbled curve, approximated to a Bézier curve $c(t)$ of degree $n = 5$. This curve $c(t)$ will be used to create the field for orienting the brush strokes.

3.2. Field generation

The field that guides the brush strokes is obtained by interpolating the first derivatives at the points of the Bézier curve. The field designed in this way follows the slope of Bézier curve. Due to the fact that the painting is performed by regions, one or more curves can be drawn manually and regularized using Bézier curves in each region.

The different approaches for interpolating scattered data can be classified into global methods, and local methods. Global methods are limited to small data, on the other hand, local methods can process larger data [36].

In this work we have used the natural neighbor interpolation which is a local method that gets the field to be aligned to the curves in the neighborhood of such curves [37]. This method operates within the convex hull of the data points. Thus, based on Fig. 4, the data points are the Bézier curve $c(t)$ points for $t_1, t_2, \ldots, t_n$. At these points, the slope $c'(t)$ of the Bézier curve is known. To get the slope $f(X)$ at any point X within the convex hull, we followed the procedure of Watson [37], where the natural neighbor's interpolant in the point X is defined as

$$f(X) = \sum_i w_i(X)\, c'(t_i) \tag{6}$$

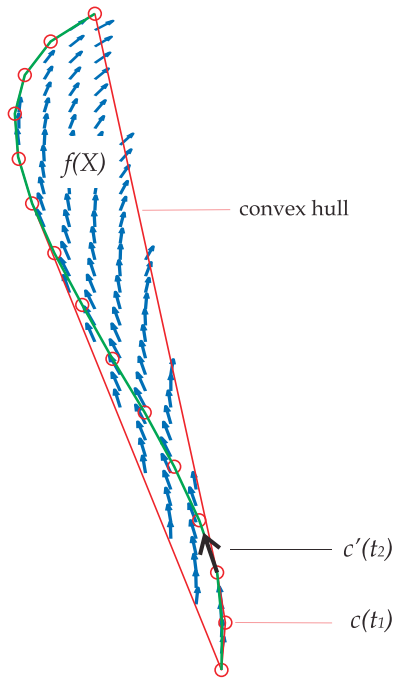


Fig. 4. Example of natural neighbor interpolation based on the Bézier curve.

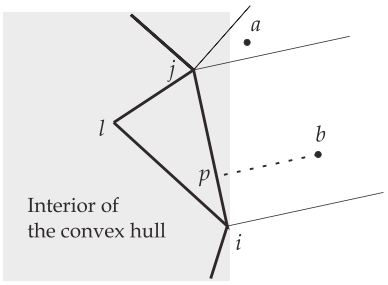


Fig. 5. Geometry of triangular extrapolation.

being $c'(t_i)$ the derivative of Eq. (2) in the natural neighbor x_i of X, i.e.

$$c'(t_i) = n \sum_{j=0}^{n-1} B_j^{n-1}(t_i)(p_{j+1} - p_j) \tag{7}$$

Fig. 4 shows an example of natural neighbor interpolation which uses the vectors tangent at the t_i points of the Bézier curve.

w_i is the weight associated with the natural neighbor x_i and can be determined by

$$w_i(X) = \frac{Area[V_i(X)]}{Area[V(X)]},\ 0 \le w_i(X) \le 1,\ \sum_i w_i(X) = 1 \tag{8}$$

where $Area[Vi(X)]$ is the area of each intersection and $Area[V(X)]$ is the total area of the Voronoi cell.

The weights w_i in Eq. (8) can be obtained in an easier way by means of Delaunay triangulation, for this purpose we used the algorithm proposed by Watson [37].

To evaluate the function at a point x_i outside the convex hull for the scattered points we used the Franke's method [38], based on the second derivative of the Bézier curve. This allows the smooth transition between the interior and exterior of the convex hull. The Fig. 5 shows the geometry of the extrapolation, which is described below.

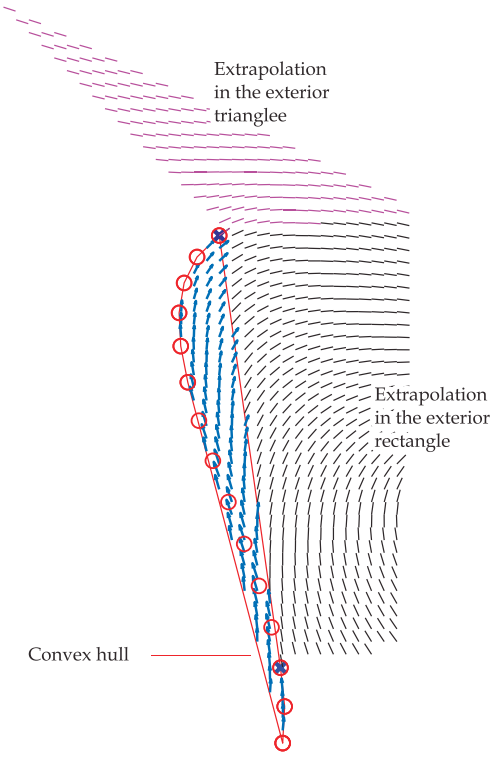


Fig. 6. Example of extrapolation based on the Bézier curve between two points of the convex hull.

For the point a in the exterior triangle, the derivative of the Bezier curve $c'(t_j)$ in the vertex j is extrapolated, adding the distance-weighted Bezier curve second derivative $c''(t_j)$ in order to achieve an acceptable behavior outside the convex hull as follows:

$$F(x_a, y_a) = c'(t_j) + c''(t_j)\frac{\|p_a - p_j\|}{\kappa} \tag{9}$$

where κ is a normalization constant, and

$$\|p_a - p_j\| = \sqrt{(x_a - x_j)^2 + (y_a - y_j)^2} \tag{10}$$

is the distance between the points p_a and p_j.

For the point b in the exterior rectangle, let p be the projection of (x_b, y_b) on the ij side, and let $(b_i, b_j, 0)$ be the barycentric coordinates of p on the T_{ijl} triangle. Then the extrapolation is obtained from the linear combination of the $c'(t)$ function in i and j, adding the distance-weighted second derivative $c''(t)$ as follows:

$$F(x_b, y_b) = b_i\left[c'(t_i) + c''(t_i)\frac{\|p_b - p\|}{\kappa}\right] + b_j\left[c'(t_j) + c''(t_j)\frac{\|p_b - p\|}{\kappa}\right] \tag{11}$$

The Fig. 6 shows an example of extrapolation between two points of the convex hull for both the exterior rectangle and the exterior triangle.

For visualizing the vector field it is better to use the Cabral and Leedom line integral convolution method [13], as shown in Fig. 7 where the tracing of loose lines were approximated to Bézier curves of degree 4 and then interpolated in the regions of interest. In this case, the input image was divided into three regions.

In order to investigate the behavior of the field caused by the manually traced curves, different configurations of curves were traced, as shown in Fig. 8. In this particular situation, the influence on the field caused by traces in the primary vein of the leaf was analyzed.

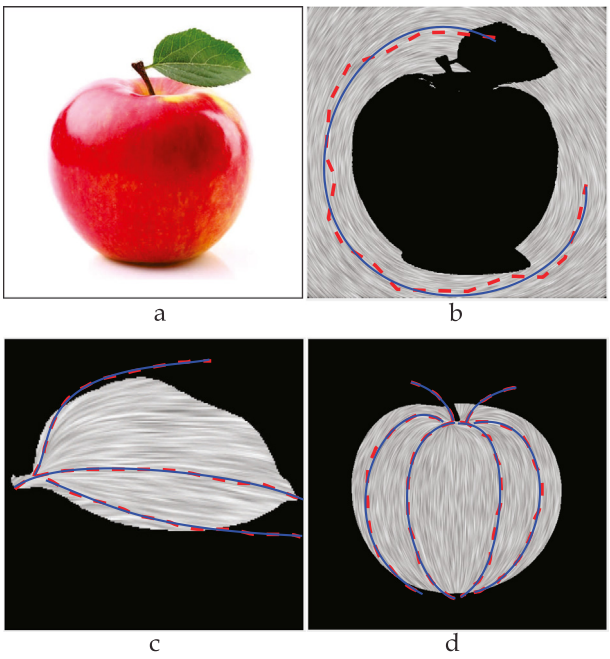


Fig. 7. Interactive design of the vector field. Manual drawing (dashed lines), Bézier approximation (continuous lines), degree $n = 4$ in this case. (a) Input image, (b) background region, 1 freehand line (c) leaf region, 3 freehand lines (d) apple region, 6 freehand lines.

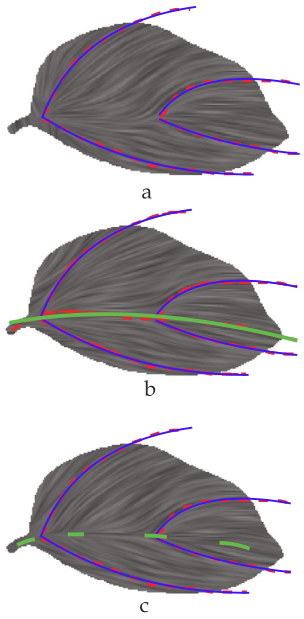


Fig. 8. Influence of the central curves on the resulting field. (a) without curves in the main vein, (b) long curve in the main vein (c) short curves on the main vein.

3.3. Generation of brush stroke paths

The generation of brush stroke paths was randomly made by obtaining the trajectory from the field and approximating each trajectory to Bézier curves. Each curve was obtained from the field using the method from Cabral and Leedom [13]. A degree

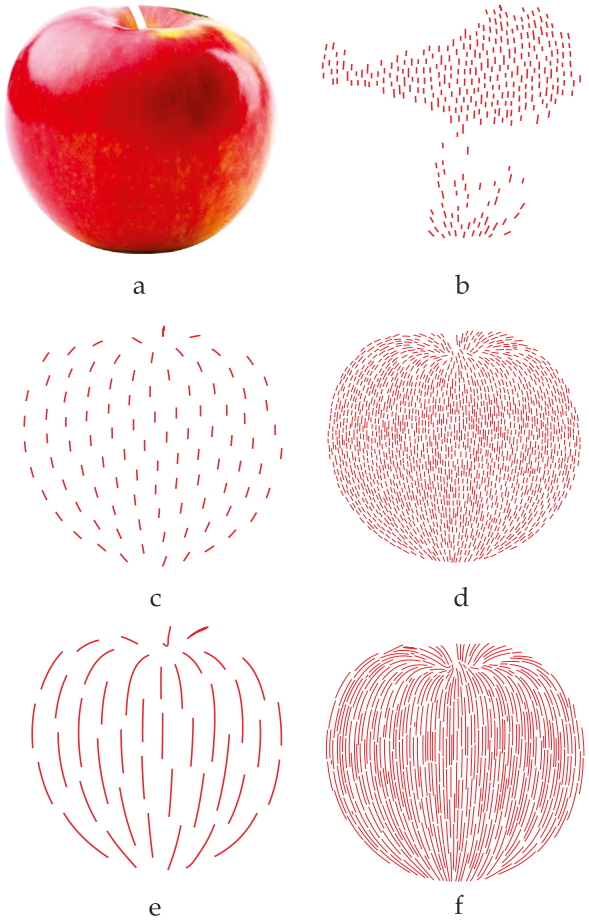


Fig. 9. Different trajectory maps, (a) reference region, (b) assignment of paths to a sub-region segmented with k-means, (c) short trajectories, thick brush, (d) short trajectories, thin brush, (e) medium trajectories, thick brush, (f) medium trajectories, thin brush.

$n = 4$ was used for the Bézier approximation, therefore 5 control points were stored for each curve. The coordinates of these control points will be sent for tracing with the robot, where the Arduino circuit reconstructs the Bézier curve. The width of each stroke was established depending on the width of the chosen brush. For each new curve, the area of that curve in the region is subtracted, this is done by dilating the binary image of the curve using a circular structurant of the established stroke width and then subtracting the area successively until the region is completed. As example, different trajectory maps are shown in Fig. 9 for the apple region.

In order to apply the paint tones in the region of Fig. 9(a) a sub-segmentation is necessary. In this case, the *k-means* algorithm [39] was used to perform the sub-segmentation. K-means is one of the simplest data grouping algorithms. The Fig. 9(b) shows the assignment of brush stroke trajectories to the darkest sub-region segmented with k-means.

4. Painting machine

The machine has the following specifications:

1. working area of 50 × 60 cm
2. resolution of 0.00625 mm/step
3. speed 20 mm/s
4. brush width 3 mm
5. acrylic paint

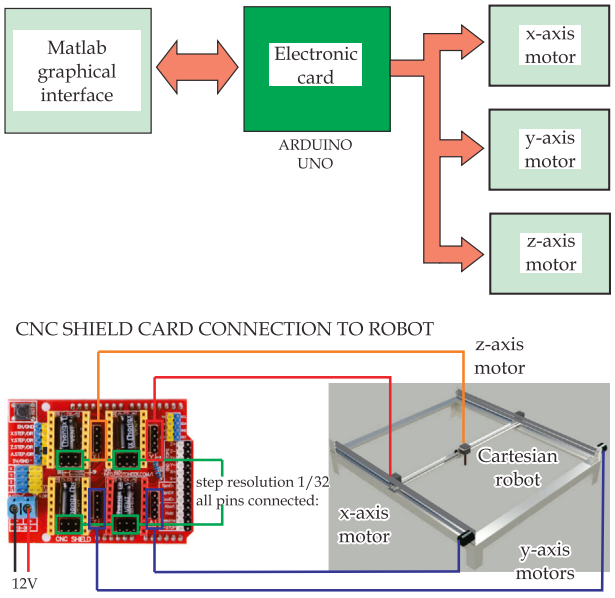


Fig. 10. Connection of the robot to the electronic card.

6. round brush
7. 5 levels of paint tones
8. NEMA 17 stepper motors

In order to control the Cartesian robot, a CNC Shield card was used, which allows the control of up to 4 stepper motors and is adapted for the Arduino Uno card. Fig. 10 shows the connection of the robot to the electronic card.

In this work the Arduino UNO card software was written in order to process the Bézier curves sent by the computer. The algorithm that runs on the Arduino UNO card is listed in the Algorithm 1. A general appreciation of this algorithm can be obtained by looking at the diagram in Fig. 11 that shows the operation between the computer and the Arduino microcontroller for each curve traced.

Algorithm 1: Steps performed on the Arduino card for each brush stroke

input : N control points of a Bézier curve
1 Read n control points;
2 Reconstruct the Bézier curve of degree $n < 20$;
3 Draw straight lines on the robot using Bresenham's algorithm [34] for each point of parameter t of the Bézier curve;
output: Brush stroke painted by the robot

5. Results and discussion

The equations for interactive segmentation and for interactive field design were implemented in a Matlab graphical interface. This allowed to generate the stroke map in an average time of 20 s, for an image of 550 × 550 pixels. Table 1 shows the established parameters and the number of strokes created for the 3 regions of Fig. 12(a) in the case of the first layer.

Additionally, the parameters defined for the trace of each brush stroke were: right angle of the brush with respect to the application surface, long stroke lengths for the first layers and short stroke lengths for the sub-regional layers.

When specifying a working area of 25 × 25 cm the total time used to paint the artwork was 3 h for a total of 2800 strokes.

As for the materials, acrylic paint, sable hair brush, paper and five tones of red paint previously prepared were used.

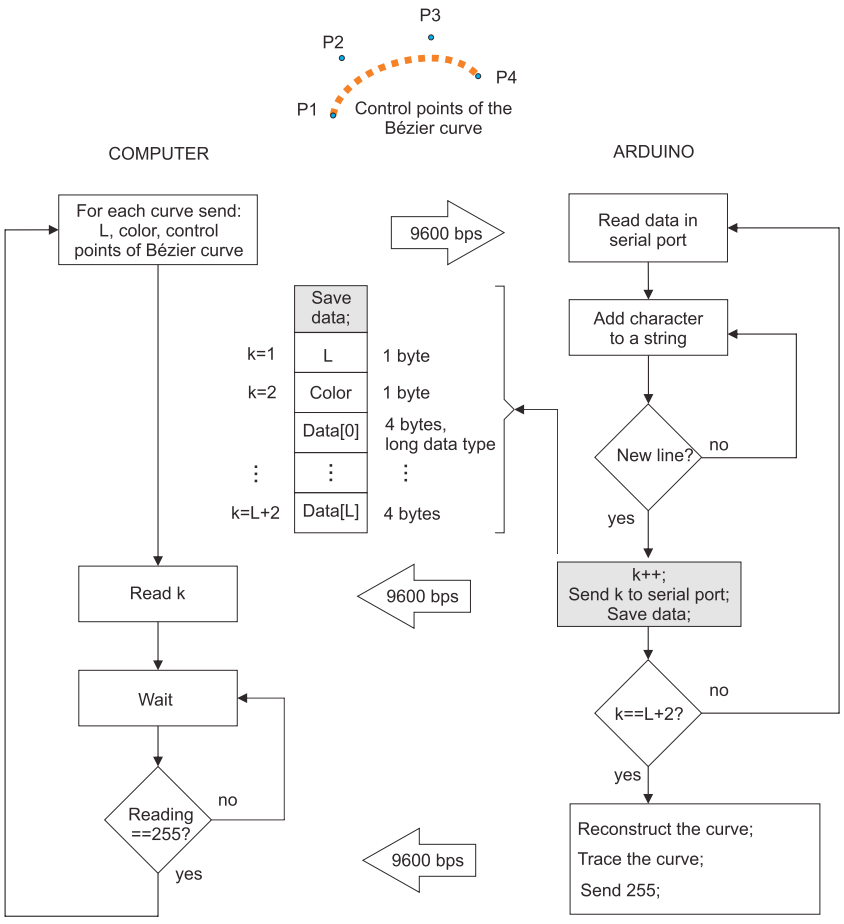


Fig. 11. Operation between the computer and the Arduino card for each curve traced.

Table 1
Brush strokes created in the regions of Fig. 12 (a) for the first layer.

Region	Length	Width	No. of strokes
Background	50 mm	3 mm	334
Apple	50 mm	3 mm	175
Leaf	50 mm	3 mm	52

The procedure we followed for creating the painting is described in Algorithm 2, where the input is the image and the output is robotic painting.

As for the automatic segmentation of sub-regions, 5 segmentations were implemented both for the leaf and for the apple in Fig. 12(a). This number corresponds to the number of colors on the paint palette.

With regard to the interactive design of the vector field, the behavior of the field lines observed on the leaf (Fig. 8(c)) shows that it is possible to join two fields by drawing short curves between the boundaries of both fields.

In the case of the generation of trajectories, traces were created for the first layer and then for the additional layers corresponding to the sub-regions, in this case additional noise could be added to the position of the seeds to obtain more realistic results.

Fig. 12 shows the application of acrylic paint, using the palette of monochromatic paint shown in Fig. 12(b). In first place, the first layer was painted with long strokes using the intermediate paint tone P3 as shown Fig. 12(c) where the complete region Region 2 is painted. Then, Region 2 was segmented into 5 sub-regions and one tone was applied to each sub-region, starting from the light tone to the dark tone one. The sub-regions were painted with short strokes. Finally, following this procedure, Region 1 and Region 3 were painted. Fig. 13 shows the finished painting with the robot and the physical color palette.

In order to compare our method with the gradient technique, Fig. 14 shows the painting based on the vector field generated with the Edge tangent flow technique [14]. In this case the boundary was also painted in the final painting. As can be seen, the brush strokes are not uniform as they follow the gradient.

Algorithm 2: Procedure for creating a painting

input : Input image
1 Set the painting dimensions;
2 Interactively segmenting the regions R in the object;
3 Set the number of tones P in the color palette;
4 Interactively designing the field for brush strokes;
5 Create the brush stroke paths according to the chosen width;
6 Segment R into S sub-regions using the k-means algorithm. Segmentation is performed by gray levels;
7 Paint the first layer in R with long strokes using the intermediate paint tone;
8 Add layers of paint, applying the tone P_i to each sub-region S_i. Paint sub-regions with short strokes. The application of paint tones is from dark to clear;
9 Continue with the other regions using steps 7 and 8;
10 Painting the boundary of regions using the intermediate paint tone;
output: Robotic painting

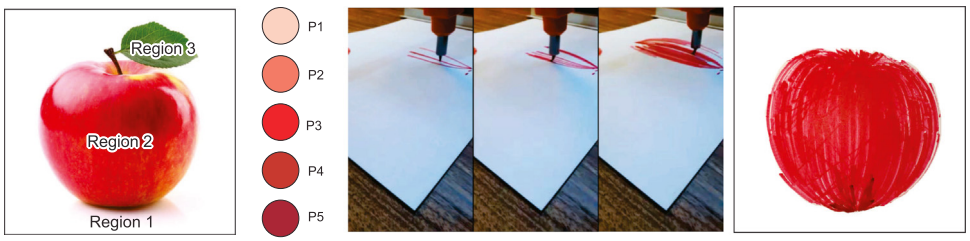


Fig. 12. Paint application progress, (a) input image, (b) color palette, (c) first layer for the Region 2 painted with long strokes using the intermediate paint tone P3.

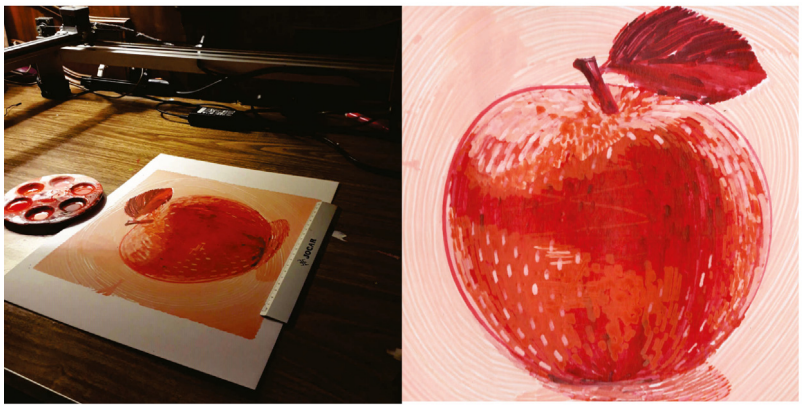


Fig. 13. Finished robotic painting using our method, (a) painting with the cartesian robot and the physical color palette, (b) painting.

Fig. 14. Example of painting with the gradient method, (a) reference image, (b) robotic painting with 5-color palette.

Some interesting aspects to investigate in the future are, other forms of automatic segmentation for producing sub-regions for paint application and other strategies for sampling seed brush strokes. Also, it is considered, the application of more layers of paint and the use of a video camera so the system performs feedback.

6. Conclusion

In this paper we presented a interactive system capable of producing realistic artworks with acrylic paint based on a pen plotter. The system focuses on the painting of objects by regions, and can be applied for example in the painting of objects such as fruits, flowers, leaves and other individual objects.

One of the advantages of interactive paint application is that the user decides when to stop the paint application, as the paint can be further refined by using more layers of paint.

The combination of interactive segmentation with automatic segmentation allowed us to complete the painting process of a whole object. For this purpose, different brush stroke lengths and several regions of the same object obtained automatically or interactively were used.

In this work the generation of strokes has no feedback, which would allow the brush strokes to be dynamic. In other words, if a brush stroke did not cover the desired area, the strokes would be redrawn or generated as the painting work progressed. This requires a video system that captures and processes the paint application.

Acrylic paint, which is diluted with water, has a faster drying time than oil-based paint, allowing the addition of paint layers in relatively short time periods. However, it was observed that the carton used was deformed by humidity, which caused the brush not to reach some areas or was overpainted when applying the secondary layers.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors are grateful for the support of the National Polytechnic Institute through the projects SIP (2019), EDI, and COFFA. Claudia Hernandez-Aguilar thanks the collaboration of educational institutions and research centers: Colegio de Postgraduados-Montecillo, Texcoco, and Cinvestav, through the Department of Physics in particular to Esther Ayala, for his assistance and support.

References

[1] H. Cohen, How to draw three people in a botanical garden, in: AAAI, 1988, pp. 846–855, URL http://dl.acm.org/citation.cfm?id=2887965.2888115.
[2] T. Lindemeier, J. Metzner, L. Pollak, O. Deussen, Hardware-based non-photorealistic rendering using a painting robot, Comput. Graph. Forum 34 (2015) 311–323, http://dx.doi.org/10.1111/cgf.12562.
[3] Cloudpainter project, available in: http://www.cloudpainter.com/. (Accessed 7 June 2019).
[4] C. Aguilar, H. Lipson, A robotic system for interpreting images into painted artwork, in: International Conference on Generative Art, Vol. 11, 2008.
[5] P.A. Tresset, F. Leymarie, Portrait drawing by paul the robot, Comput. Graph. 37 (2013) 348–363, http://dx.doi.org/10.1016/j.cag.2013.01.012.
[6] O. Deussen, T. Lindemeier, S. Pirk, M. Tautzenberger, Feedback-guided stroke placement for a painting machine, in: Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging, 2012, pp. 25–33. URL http://dl.acm.org/citation.cfm?id=2328888.2328894.
[7] R. Luo, M. Hong, P. Chung, Robot Artist for colorful picture painting with visual control system, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 2998–3003, http://dx.doi.org/10.1109/IROS.2016.7759464.
[8] D. Song, T. Lee, Y. Kim, S. Sohn, Y. Kim, Artistic pen drawing on an arbitrary surface using an impedance-controlled robot, in: IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 4085–4090, http://dx.doi.org/10.1109/ICRA.2018.8461084.
[9] L. Scalera, S. Seriani, A. Gasparetto, P. Gallina, Watercolour robotic painting: a novel automatic system for artistic rendering, J. Intell. Robot. Syst. (2018) 1–16, http://dx.doi.org/10.1007/s10846-018-0937-y.
[10] A.I. Karimov, E.E. Kopets, V.G. Rybin, S.V. Leonov, A.I. Voroshilova, D.N. Butusov, Advanced tone rendition technique for a painting robot, Robot. Auton. Syst. 115 (2019) 17–27, http://dx.doi.org/10.1016/j.robot.2019.02.009.
[11] The Robot Art Competition, available in: https://robotart.org/. (Accessed 7 June 2019).
[12] J.E. Kyprianidis, J. Collomosse, T. Wang, T. Isenberg, State of the art: A taxonomy of artistic stylization techniques for images and video, IEEE Trans. Vis. Comput. Graphics 19 (2013) 866–885, http://dx.doi.org/10.1109/TVCG.2012.160.
[13] B. Cabral, L.C. Leedom, Imaging vector fields using line integral convolution, in: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, 1993, pp. 263–270. http://dx.doi.org/10.1145/280814.280951.
[14] H. Kang, L. Seungyong, C.K. Chui, Flow-based image abstraction, IEEE Trans. Vis. Comput. Graphics 15 (2009) 0–76, http://dx.doi.org/10.1109/TVCG.2008.81.
[15] A. Hertzmann, Painterly rendering with curved brush strokes of multiple sizes, in: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH'98, 1998, pp. 453–460. http://dx.doi.org/10.1145/280814.280951.
[16] J. Hays, I. Essa, Image and video based painterly animation, in: Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering, 2004, pp. 113–120. http://dx.doi.org/10.1145/987657.987676.
[17] P. Haeberli, Paint by numbers: abstract image representations, SIGGRAPH Comput. Graph. 24 (4) (1990) 207–214, http://dx.doi.org/10.1145/97880.97902.
[18] S.C. Olsen, B.A. Maxwell, B. Gooch, Interactive vector fields for painterly rendering, in: Proceedings of Graphics Interface 2005, 2005, pp. 241–247. URL http://dl.acm.org/citation.cfm?id=1089508.1089548.
[19] S.R. Vantaram, E. Saber, Survey of contemporary trends in color image segmentation, J. Electron. Imaging 21 (2012) 1–28, http://dx.doi.org/10.1117/1.JEI.21.4.040901.
[20] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models, Int. J. Comput. Vis. 1 (1988) 321–331, http://dx.doi.org/10.1007/BF00133570.
[21] A. Blake, C. Rother, M. Brown, P. Perez, P. Torr, Interactive image segmentation using an adaptive GMMRF model, in: Computer Vision - ECCV 2004, 2004, pp. 428–441.
[22] J. Wang, M. Agrawala, M.F. Cohen, Soft scissors: An interactive tool for realtime high quality matting, ACM Trans. Graph. 26 (2007) http://dx.doi.org/10.1145/1276377.1276389.
[23] X. Bai, G. Sapiro, A geodesic framework for fast interactive image and video segmentation and matting, in: 2007 IEEE 11th International Conference on Computer Vision, 2007, pp. 1–8. http://dx.doi.org/10.1109/ICCV.2007.4408931.
[24] Y. Boykov, G. Funka-Lea, Graph cuts and efficient n-d image segmentation, Int. J. Comput. Vis. 70 (2006) 109–131, http://dx.doi.org/10.1007/s11263-006-7934-5.
[25] L. Grady, Random walks for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006) 1768–1783, http://dx.doi.org/10.1109/TPAMI.2006.233.
[26] V. Lempitsky, P. Kohli, C. Rother, T. Sharp, Image segmentation with a bounding box prior, in: 2009 IEEE 12th International Conference on Computer Vision, 2009, pp. 277–284. http://dx.doi.org/10.1109/ICCV.2009.5459262.
[27] C. Rother, V. Kolmogorov, A. Blake, Grabcut: Interactive foreground extraction using iterated graph cuts, ACM Trans. Graph. 23 (2004) 309–314, http://dx.doi.org/10.1145/1015706.1015720.
[28] E.N. Mortensen, W.A. Barrett, Intelligent scissors for image composition, in: Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques, 1995, pp. 191–198. http://dx.doi.org/10.1145/218380.218442.
[29] A. Rosenfeld, J. Weszka, An improved method of angle detection on digital curves, IEEE Trans. Comput. C-24 (1975) 940–941, http://dx.doi.org/10.1109/T-C.1975.224342.
[30] J. Wang, W. Zhang, A survey of corner detection methods, in: 2018 2nd International Conference on Electrical Engineering and Automation, ICEEA 2018, 2018. http://dx.doi.org/10.2991/iceea-18.2018.47.
[31] K. Abe, R. Morii, K. Nishida, T. Kadonaga, Comparison of methods for detecting corner points from digital curves-a preliminary report, in: Proceedings of 2nd International Conference on Document Analysis and Recognition, ICDAR '93, 1993, pp. 854–857, http://dx.doi.org/10.1109/ICDAR.1993.395603.
[32] J. Munoz-Rodrí guez, R. Rodrí guez Vera, A. Asundi, G. Garnica-Campos, Shape detection using light line and Bezier approximation network, J. Imaging Sci. 55 (2007) 29–39, http://dx.doi.org/10.1179/174313107X165236.
[33] E.T.Y. Lee, Choosing nodes in parametric curve interpolation, Comput. Aided Des. 21 (1989) 363–370, http://dx.doi.org/10.1016/0010-4485(89)90003-1.
[34] J.E. Bresenham, Algorithm for computer control of a digital plotter, IBM Syst. J. 4 (1965) 25–30, http://dx.doi.org/10.1147/sj.41.0025.
[35] S. Biswas, B.C. Lovell, Bezier and Splines in Image Processing and Machine Vision, Springer Publishing Company, Incorporated, 2007.
[36] I. Amidror, Scattered data interpolation methods for electronic imaging systems: a survey, J. Electron. Imaging 11 (2002) 157–176, http://dx.doi.org/10.1117/1.1455013.
[37] D. Watson, Compound signed decomposition, the core of natural neighbor interpolation in n-dimensional space, 2001, Unpublished.
[38] R. Franke, Scattered data interpolation: tests of some methods, Math. Comp. 38 (1982) 181–200, http://dx.doi.org/10.2307/2007474.
[39] J. Macqueen, Some methods for classification and analysis of multivariate observations, in: 5-Th Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297. http://dx.doi.org/10.1109/ICRA.2018.8461084.

Otoniel Igno is currently a doctoral student at National Polytechnic Institute of Mexico within the Graduate Program in Systems Engineering of ESIME Zacatenco. He received master's degree in optomechatronics from the Center for Research in Optics in Mexico. His research interests include robotics, optical metrology, computer vision, digital image processing, art and design.

Dr. C. Hernandez-Aguilar, Professor-researcher of the National Polytechnic Institute, within the Graduate Program in Systems Engineering of ESIME Zacatenco. Member of the Mexican Academy of Sciences and the National System of Researchers (Mexico). International distinction as member of the Editorial Committee of the journal: International Agrophysics (period:2012–present). She currently has articles published in international indexed journals and more than 300 citations to her articles. Concerned and occupied in improving the quality of life of society. Trainer of researchers in the last 12 years, making a call to conscience, to rescue a human attitude in the research process and the impact obtained from it. Motto efforts, transforming auto-self to transform his world.

Dr. Alfredo Cruz Orea, Degree in Physics and Mathematics from the National Polytechnic Institute of Mexico. His Master and Doctorate studies in Physics were carried out at the Universidad de Estadual de Campiñas, Sao Paulo, Brazil. Since April 1999, he has been a researcher in the Physics Department of the IPN's Center for Research and Advanced Studies. He is a member of the National System of Researchers, with level III since 2008. His line of research is the study of optical and thermal properties of materials by photothermal techniques. He currently has 136 articles published in indexed international journals, four chapters in books and more than 600 citations to his articles. He has directed and co-directed 8 doctoral theses, 10 master's and 5 bachelor's theses.

Dr. Arturo Dominguez Pacheco, Doctorate in Systems Engineering with a postdoctoral degree in PHYSICS from Cinvestav, Mexico (two periods). Research Professor of the Postgraduate Program in Systems Engineering at ESIME-Zacatenco-IPN in the line of research in Engineering Systems. Member of the Research Group on Sustainable Biophysical Systems (SBS) for Agriculture, Food and Medicine. Main scientific contributions in the area of Characterization of Materials and Development of Irradiator Prototypes. Active collaborator and participant in research projects with the Cinvestav Photothermal Techniques group, as well as Director of research projects at the IPN since 2010. Currently SNI level I of the area VI.