

**Proyecto Final de Estudios: Robot cosechador automático**

---

XX/XX/2025



**UNCUYO**  
UNIVERSIDAD  
NACIONAL DE CUYO

## **Proyecto Final de Estudios**

### Cosechador de Lechugas Autónomo con Unidad de Detección por Inteligencia Artificial

Brenda Gudiño  
Alan Vignolo

27 de septiembre de 2025

# Índice general

# Resumen

# **1. Introducción**

## **1.1. Contexto y Motivación**

## **1.2. Objetivos del Proyecto**

## **1.3. Alcance y Limitaciones**

## **1.4. Estructura del Documento**

## 2. Marco Teórico

### 2.1. Sistemas de Control Jerárquico

### 2.2. Fundamentos de Visión Artificial y Aprendizaje Profundo

#### 2.2.1. Procesamiento Digital de Imágenes

El procesamiento digital de imágenes constituye el fundamento tecnológico de los sistemas de visión artificial en robótica agrícola. Esta disciplina integra conceptos de análisis matemático, álgebra lineal y teoría de señales para transformar datos visuales crudos en información estructurada y procesable por algoritmos de decisión.

#### Representación Digital de Imágenes

Una imagen digital se define matemáticamente como una función bidimensional discreta  $f(x, y)$ , donde las coordenadas espaciales  $(x, y)$  representan posiciones en el plano de la imagen y el valor de la función indica la intensidad luminosa en cada punto. En el contexto de imágenes a color en formato RGB, esta representación se extiende a un espacio tridimensional:

$$\mathbf{I}(x, y) = \begin{bmatrix} R(x, y) \\ G(x, y) \\ B(x, y) \end{bmatrix} \in [0, 255]^3 \quad (2.1)$$

donde  $R$ ,  $G$  y  $B$  representan los canales de color rojo, verde y azul respectivamente. Esta representación permite codificar aproximadamente 16.7 millones de colores distintos ( $256^3$ ), cubriendo ampliamente el espectro visible necesario para aplicaciones de agricultura de precisión.

La resolución espacial de una imagen determina la cantidad de información disponible para el análisis. Una imagen de dimensiones  $M \times N$  píxeles contiene  $M \cdot N$  elementos de información, cada uno representando el promedio de la radiancia incidente sobre el área del sensor correspondiente a ese píxel.

#### Espacios de Color y Transformaciones

El espacio de color RGB, aunque intuitivo y ampliamente utilizado en sistemas de captura, presenta limitaciones para ciertas tareas de procesamiento debido a su fuerte correlación entre canales y sensibilidad a variaciones de iluminación. Por esta razón, la conversión a espacios de color alternativos resulta fundamental en visión por computadora.

##### Espacio HSV (Hue, Saturation, Value)

El espacio HSV separa la información cromática (matiz y saturación) de la información de iluminación (valor), proporcionando robustez frente a cambios en las condiciones de luz. La transformación de RGB a HSV se define mediante:

$$V = \max(R, G, B) \quad (2.2)$$

$$S = \begin{cases} \frac{V - \min(R, G, B)}{V} & \text{si } V \neq 0 \\ 0 & \text{si } V = 0 \end{cases} \quad (2.3)$$



Figura 2.1: Representación de una imagen digital en el espacio RGB mostrando los tres canales de color

$$H = \begin{cases} 60 \cdot \frac{G-B}{V-\min(R,G,B)} & \text{si } V = R \\ 60 \cdot \left( 2 + \frac{B-R}{V-\min(R,G,B)} \right) & \text{si } V = G \\ 60 \cdot \left( 4 + \frac{R-G}{V-\min(R,G,B)} \right) & \text{si } V = B \end{cases} \quad (2.4)$$

Esta descomposición resulta particularmente ventajosa para la detección de elementos basada en contraste, ya que el canal  $V$  captura la intensidad luminosa independientemente del color, permitiendo operaciones de umbralización más robustas.


### Ventajas Operativas del Espacio HSV

La separación de la información de brillo en el canal  $V$  permite implementar algoritmos de detección invariantes a cambios de iluminación ambiental. En aplicaciones agrícolas donde las condiciones lumínicas varían significativamente durante el día, esta propiedad resulta crítica para mantener la consistencia operativa del sistema.

Matemáticamente, el canal  $V$  satisface la propiedad de invariancia:

$$V(\alpha \cdot \mathbf{I}_{RGB}) = \alpha \cdot V(\mathbf{I}_{RGB}) \quad \forall \alpha > 0 \quad (2.5)$$

donde  $\alpha$  representa un factor de escala de iluminación global. Esta linealidad permite compensar variaciones uniformes de brillo mediante ajustes simples de umbralización.



imagenes/comparacion\_rgb\_hsv.png

Figura 2.2: Comparación entre espacios de color RGB y HSV para una misma escena de cultivo

## Umbralización y Segmentación

La umbralización constituye una técnica fundamental de segmentación que particiona una imagen en regiones de interés mediante la comparación de intensidades de píxeles contra valores de referencia. La umbralización binaria se define formalmente como:

$$g(x, y) = \begin{cases} 255 & \text{si } f(x, y) > T \\ 0 & \text{si } f(x, y) \leq T \end{cases} \quad (2.6)$$

donde  $T$  es el valor umbral y  $g(x, y)$  es la imagen resultante.

### Umbralización Inversa

En escenarios donde los elementos de interés presentan intensidades menores que el fondo, la umbralización inversa resulta más apropiada:

$$g(x, y) = \begin{cases} 255 & \text{si } f(x, y) < T \\ 0 & \text{si } f(x, y) \geq T \end{cases} \quad (2.7)$$

Esta variante permite destacar regiones oscuras contra fondos claros, técnica ampliamente aplicada en la detección de marcadores de contraste en ambientes controlados.

### Selección Óptima del Umbral





Figura 2.3: Proceso de umbralización binaria inversa para destacar elementos oscuros

La determinación del valor umbral óptimo  $T^*$  puede realizarse mediante análisis del histograma de intensidades. El método de Otsu minimiza la varianza intra-clase de los grupos resultantes:

$$T^* = \arg \min_T [\omega_0(T)\sigma_0^2(T) + \omega_1(T)\sigma_1^2(T)] \quad (2.8)$$

donde  $\omega_i$  y  $\sigma_i^2$  representan el peso y varianza de cada clase. Alternativamente, puede emplearse conocimiento previo del entorno para establecer umbrales fijos que aprovechen características específicas del escenario operativo.

## Operaciones Morfológicas

Las operaciones morfológicas manipulan la forma y estructura de objetos en imágenes binarias mediante la aplicación de elementos estructurantes. Estas operaciones se fundamentan en la teoría de conjuntos y permiten eliminar ruido, rellenar huecos y separar objetos conectados.

### Erosión

La erosión reduce el tamaño de los objetos eliminando píxeles del perímetro:

$$(A \ominus B)(x, y) = \min_{(s,t) \in B} \{A(x + s, y + t)\} \quad (2.9)$$

donde  $A$  es la imagen binaria y  $B$  es el elemento estructurante.

### Dilatación

La dilatación expande los objetos añadiendo píxeles al perímetro:

$$(A \oplus B)(x, y) = \max_{(s,t) \in B} \{A(x + s, y + t)\} \quad (2.10)$$

### Apertura y Cierre

La apertura ( $A \circ B = (A \ominus B) \oplus B$ ) elimina pequeños objetos y protuberancias, mientras el cierre ( $A \bullet B = (A \oplus B) \ominus B$ ) rellena huecos pequeños y conecta componentes próximos. Estas operaciones compuestas resultan esenciales para el refinamiento de máscaras de segmentación en aplicaciones reales.

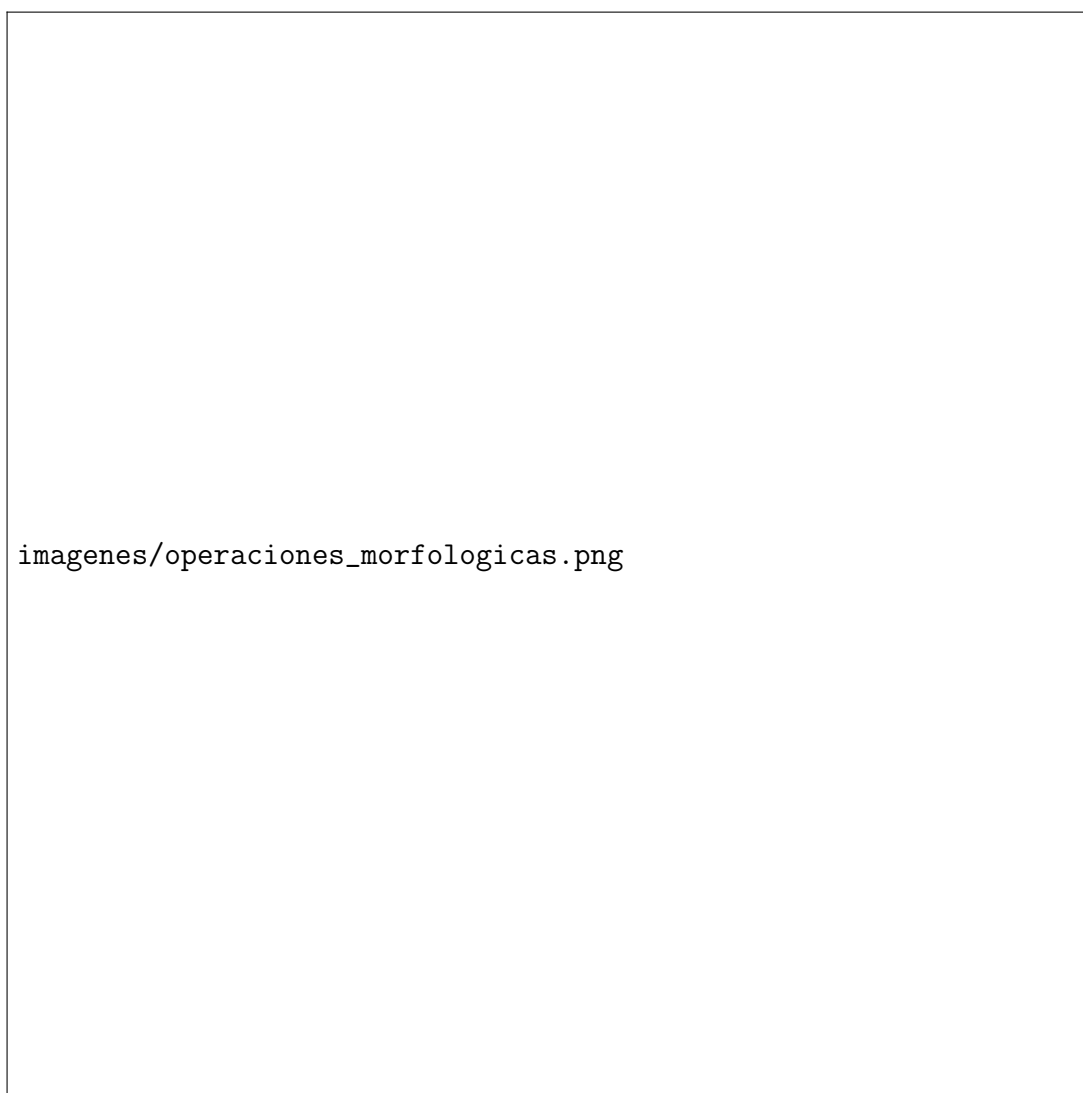


Figura 2.4: Efecto de operaciones morfológicas sobre una imagen binaria

## Calibración y Transformación de Coordenadas

La calibración establece la correspondencia entre coordenadas píxel en la imagen y coordenadas físicas en el espacio de trabajo. Esta transformación es fundamental para convertir mediciones visuales en comandos de movimiento precisos.

### Modelo de Transformación Lineal

Para configuraciones donde la cámara observa perpendicular al plano de trabajo, la transformación píxel-a-métrica puede aproximarse mediante un modelo lineal:

$$\begin{bmatrix} x_{mm} \\ y_{mm} \end{bmatrix} = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix} \begin{bmatrix} x_{px} \\ y_{px} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (2.11)$$

donde  $k_x, k_y$  son factores de escala (mm/píxel) y  $t_x, t_y$  son desplazamientos de origen.

### Estimación por Mínimos Cuadrados

Los parámetros de calibración se estiman mediante regresión lineal a partir de puntos de correspondencia conocidos:

$$\mathbf{K} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (2.12)$$

donde  $\mathbf{X}$  contiene las coordenadas píxel de puntos de referencia,  $\mathbf{y}$  sus coordenadas métricas medidas físicamente, y  $\mathbf{K}$  el vector de parámetros de calibración. Este método minimiza el error cuadrático medio garantizando la solución óptima en el sentido de mínimos cuadrados.



Figura 2.5: Proceso de calibración mediante puntos de correspondencia conocidos

### Validación Estadística

La precisión de la calibración se cuantifica mediante el error cuadrático medio (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2} \quad (2.13)$$

donde  $\hat{y}_i$  son las predicciones del modelo. Valores de RMSE inferiores a 1mm resultan aceptables para aplicaciones de manipulación agrícola de precisión.

### 2.2.2. Detección y Análisis de Contornos

La detección de contornos constituye una técnica fundamental en visión por computadora para la identificación y caracterización de objetos en imágenes. Los contornos representan las fronteras entre regiones con propiedades visuales distintas, proporcionando información estructural esencial para el reconocimiento y clasificación de elementos.

#### Definición Matemática de Contorno

Un contorno se define como la secuencia ordenada de píxeles que forman el límite de una región conexa en una imagen binaria:

$$C = \{(x_i, y_i)\}_{i=1}^n \quad \text{donde } (x_i, y_i) \in \partial R \quad (2.14)$$

siendo  $\partial R$  la frontera topológica de la región  $R$ . Formalmente, un punto  $(x, y)$  pertenece a  $\partial R$  si:

$$(x, y) \in R \quad \wedge \quad \exists (x', y') \in \mathcal{N}(x, y) : (x', y') \notin R \quad (2.15)$$

donde  $\mathcal{N}(x, y)$  representa el vecindario del píxel (usualmente 4-conectividad u 8-conectividad).

#### Algoritmos de Detección de Contornos

##### Método de Seguimiento de Bordes (Border Following)

El algoritmo de Moore-Neighbor tracing recorre el perímetro de objetos binarios siguiendo el borde en sentido horario o antihorario:

1. Localizar primer píxel de objeto (barrido izquierda-derecha, arriba-abajo)
2. Examinar vecinos en orden circular desde el píxel de entrada
3. Seleccionar primer vecino que pertenezca al objeto
4. Repetir hasta retornar al píxel inicial

La complejidad computacional es  $\mathcal{O}(n)$  donde  $n$  es el número de píxeles del perímetro.

##### Algoritmo de Suzuki-Abe

Este método detecta simultáneamente contornos externos e internos (huecos), organizándolos jerárquicamente. Define dos tipos de bordes:

- **Border tipo 0:** Borde externo de objeto rodeado por fondo
- **Border tipo 1:** Borde interno de hueco rodeado por objeto

La jerarquía se representa mediante un árbol donde cada nodo almacena:

$$\text{Nodo}_i = \{\text{Contorno}_i, \text{Padre}_i, \text{Hijos}_i\} \quad (2.16)$$



Figura 2.6: Jerarquía de contornos mostrando relación padre-hijo entre contornos externos e internos

## Aproximación y Simplificación de Contornos

Los contornos detectados suelen contener información redundante. La aproximación poligonal reduce el número de puntos preservando la forma esencial.

### Algoritmo de Douglas-Peucker


Este método recursivo aproxima una curva mediante segmentos de línea recta:

1. Conectar primer y último punto del contorno
2. Calcular distancia perpendicular de todos los puntos intermedios a esta línea
3. Si  $d_{max} > \epsilon$ , dividir contorno en el punto de máxima distancia
4. Aplicar recursivamente a cada segmento
5. Si  $d_{max} \leq \epsilon$ , aproximar segmento con línea recta

La tolerancia  $\epsilon$  controla el nivel de simplificación:

$$\epsilon_{optimal} = \alpha \cdot P_{contorno} \quad (2.17)$$

donde  $P_{contorno}$  es el perímetro y  $\alpha \in [0,01, 0,05]$  es un factor empírico.



imagenes/aproximacion\_contorno.png

Figura 2.7: Aproximación poligonal de contorno mediante algoritmo Douglas-Peucker

## Descriptores Geométricos de Contornos

Los descriptores cuantifican propiedades estructurales de los contornos, permitiendo clasificación y reconocimiento de formas.

### Área

El área encerrada por un contorno se calcula mediante la fórmula del Shoelace:

$$A = \frac{1}{2} \left| \sum_{i=0}^{n-1} (x_i y_{i+1} - x_{i+1} y_i) \right| \quad (2.18)$$

donde  $(x_n, y_n) = (x_0, y_0)$  para cerrar el contorno.

### Perímetro

La longitud del contorno se aproxima sumando distancias euclidianas entre puntos consecutivos:

$$P = \sum_{i=0}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (2.19)$$

### Centroide

El centro de masa del contorno se calcula como:

## Proyecto Final de Estudios: Robot cosechador automático

$$\bar{x} = \frac{1}{6A} \sum_{i=0}^{n-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \quad (2.20)$$

$$\bar{y} = \frac{1}{6A} \sum_{i=0}^{n-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \quad (2.21)$$

### Momentos de Imagen

Los momentos proporcionan información sobre la distribución espacial del contorno:

$$M_{pq} = \sum_{i=0}^{n-1} x_i^p y_i^q \quad (2.22)$$

Los momentos centrales, invariantes a traslación, se definen como:

$$\mu_{pq} = \sum_{i=0}^{n-1} (x_i - \bar{x})^p (y_i - \bar{y})^q \quad (2.23)$$

### Razón de Aspecto

La relación entre dimensiones principales se calcula mediante el rectángulo delimitador mínimo:

$$AR = \frac{W_{bbox}}{H_{bbox}} \quad (2.24)$$

donde  $W_{bbox}$  y  $H_{bbox}$  son ancho y alto del bounding box.

imagenes/bounding\_box.png

Figura 2.8: Rectángulo delimitador y ejes principales de un contorno

### Circularidad

Mide cuán similar es un contorno a un círculo:

## Proyecto Final de Estudios: Robot cosechador automático

---

$$C = \frac{4\pi A}{P^2} \quad (2.25)$$

Propiedades:

- $C = 1$  para círculos perfectos
- $C < 1$  para formas no circulares
- $C \rightarrow 0$  para formas muy irregulares

### Convexidad

La relación entre el área del contorno y su envolvente convexa:

$$\text{Convexity} = \frac{A_{\text{convexHull}}}{A_{\text{contour}}} \quad (2.26)$$

Valores cercanos a 1 indican formas convexas; valores mayores indican concavidades.

### Solidez

Mide la proporción del contorno que ocupa su envolvente convexa:

$$\text{Solidity} = \frac{A_{\text{contour}}}{A_{\text{convexHull}}} \quad (2.27)$$

## Filtrado de Contornos

En aplicaciones reales, las imágenes contienen múltiples contornos, muchos correspondientes a ruido. El filtrado elimina contornos irrelevantes según criterios definidos.

### Filtrado por Área

Eliminar contornos cuya área esté fuera de un rango esperado:

$$\text{Contorno válido} \Leftrightarrow A_{\min} \leq A_{\text{contour}} \leq A_{\max} \quad (2.28)$$

Los umbrales se establecen según el tamaño esperado de objetos de interés.

### Filtrado por Jerarquía

Seleccionar solo contornos de nivel específico en la jerarquía:


- Nivel 0: Contornos externos principales
- Nivel 1: Huecos dentro de objetos
- Niveles superiores: Estructuras anidadas

### Filtrado por Forma

Utilizar descriptores geométricos como criterio:

$$\text{Validez} = \begin{cases} \text{True} & \text{si } C > C_{\min} \wedge AR \in [AR_{\min}, AR_{\max}] \\ \text{False} & \text{en caso contrario} \end{cases} \quad (2.29)$$





imagenes/filtrado\_contornos.png

Figura 2.9: Proceso de filtrado de contornos por área y forma

## Análisis de Regiones Internas

Una vez identificados contornos relevantes, el análisis del contenido interior proporciona información adicional para clasificación.

### Extracción de Región de Interés (ROI)

Dado un contorno, extraer la región rectangular que lo contiene:

$$\text{ROI} = I[y_{\min} : y_{\max}, x_{\min} : x_{\max}] \quad (2.30)$$

donde  $(x_{\min}, y_{\min})$  y  $(x_{\max}, y_{\max})$  son las coordenadas extremas del contorno.

### Máscara de Contorno

Crear una máscara binaria donde píxeles internos al contorno valen 1:

$$M(x, y) = \begin{cases} 1 & \text{si } (x, y) \in \text{Interior}(C) \\ 0 & \text{en caso contrario} \end{cases} \quad (2.31)$$

Esta máscara permite análisis estadístico de píxeles contenidos.

### Conteo de Píxeles

El número de píxeles dentro del contorno proporciona una medida de área alternativa:

$$N_{pixels} = \sum_{(x,y) \in ROI} M(x, y) \quad (2.32)$$

Esta métrica es menos sensible a irregularidades del perímetro que el cálculo mediante coordenadas del contorno.

## Invariancia a Transformaciones

Para robustez ante variaciones de escala, rotación y traslación, se emplean descriptores invariantes.

### Momentos de Hu

Los siete momentos invariantes de Hu se construyen a partir de momentos centrales normalizados:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{1+(p+q)/2}} \quad (2.33)$$

El primer momento de Hu, por ejemplo:

$$h_1 = \eta_{20} + \eta_{02} \quad (2.34)$$

Estos momentos permanecen constantes bajo rotación, escala y traslación.

### Descriptores de Fourier

La transformada de Fourier del contorno proporciona representación en frecuencia invariante a transformaciones:

$$F_n = \sum_{k=0}^{N-1} z_k e^{-i2\pi nk/N} \quad (2.35)$$

donde  $z_k = x_k + iy_k$  es la representación compleja del contorno. Los coeficientes de baja frecuencia capturan la forma global, mientras los de alta frecuencia representan detalles.

## 2.2.3. Clasificación Basada en Análisis Morfológico

La clasificación morfológica de objetos utiliza características geométricas y estadísticas extraídas de contornos e imágenes para asignar categorías a elementos detectados. A diferencia de métodos de aprendizaje profundo, este enfoque se fundamenta en reglas explícitas derivadas del conocimiento del dominio y análisis estadístico de datos representativos.

### Fundamentos de Clasificación por Umbrales

El método más directo de clasificación morfológica establece límites de decisión basados en valores de descriptores geométricos. Para un descriptor  $d$  y un conjunto de clases  $\mathcal{C} = \{c_1, c_2, \dots, c_n\}$ , la regla de clasificación se define como:

$$\text{Clase}(x) = c_i \quad \text{si} \quad t_{i-1} < d(x) \leq t_i \quad (2.36)$$

donde  $t_0, t_1, \dots, t_n$  son umbrales que particionan el espacio de características.

### Clasificación por Área

Para objetos que se diferencian principalmente por tamaño, el área en píxeles constituye un descriptor efectivo:

$$\text{Clase}(C) = \begin{cases} \text{Pequeño} & \text{si } A < t_1 \\ \text{Mediano} & \text{si } t_1 \leq A < t_2 \\ \text{Grande} & \text{si } A \geq t_2 \end{cases} \quad (2.37)$$

## Proyecto Final de Estudios: Robot cosechador automático

La robustez del método depende de la separabilidad entre clases. La distancia entre centroides de clases adyacentes debe ser significativa respecto a la dispersión intra-clase.



Figura 2.10: Distribución de áreas para diferentes categorías de objetos

### Análisis Estadístico para Determinación de Umbrales

La selección óptima de umbrales requiere análisis estadístico de una base de datos representativa de cada clase.

#### Parámetros Estadísticos por Clase

Para cada clase  $c_i$ , se calculan:

$$\mu_i = \frac{1}{N_i} \sum_{j=1}^{N_i} d_j \quad (2.38)$$

$$\sigma_i = \sqrt{\frac{1}{N_i - 1} \sum_{j=1}^{N_i} (d_j - \mu_i)^2} \quad (2.39)$$

donde  $N_i$  es el número de muestras de la clase  $i$ ,  $\mu_i$  es la media y  $\sigma_i$  la desviación estándar del descriptor  $d$ .

#### Determinación de Umbrales mediante Análisis de Distribuciones

Asumiendo distribuciones normales para cada clase:

**Proyecto Final de Estudios: Robot cosechador automático**

---

$$P(d|c_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(d - \mu_i)^2}{2\sigma_i^2}\right) \quad (2.40)$$

El umbral óptimo entre clases  $c_i$  y  $c_{i+1}$  se encuentra resolviendo:

$$P(d|c_i) = P(d|c_{i+1}) \quad (2.41)$$

Para distribuciones con igual varianza ( $\sigma_i = \sigma_{i+1} = \sigma$ ), la solución es:

$$t_i^* = \frac{\mu_i + \mu_{i+1}}{2} \quad (2.42)$$



Figura 2.11: Distribuciones gaussianas de clases y umbral óptimo de decisión

### **Criterio de Máxima Verosimilitud**

Para varianzas distintas, el umbral se obtiene resolviendo:

$$\frac{(d - \mu_i)^2}{\sigma_i^2} = \frac{(d - \mu_{i+1})^2}{\sigma_{i+1}^2} \quad (2.43)$$

Esta ecuación cuadrática puede tener dos soluciones; se selecciona la que minimiza la probabilidad de error.

## Margen de Confianza y Zonas de Incertidumbre

Para mejorar la robustez, se establecen intervalos de confianza alrededor de cada umbral:

$$\text{Zona de incertidumbre} = [t_i - k\sigma_{pooled}, t_i + k\sigma_{pooled}] \quad (2.44)$$

donde  $\sigma_{pooled} = \sqrt{(\sigma_i^2 + \sigma_{i+1}^2)/2}$  y  $k \in [1, 2]$  define el nivel de confianza. Muestras en esta zona pueden requerir verificación adicional o clasificación multi-criterio.

## Clasificación Multi-Criterio

Cuando un único descriptor no proporciona separabilidad suficiente, se emplean múltiples características simultáneamente.

### Reglas de Decisión Conjuntas

La clasificación se basa en la evaluación simultánea de  $m$  descriptores  $\{d_1, d_2, \dots, d_m\}$ :

$$\text{Clase}(x) = c_i \quad \text{si} \quad \bigwedge_{j=1}^m (t_{ij}^{min} \leq d_j(x) \leq t_{ij}^{max}) \quad (2.45)$$

donde  $\bigwedge$  denota la conjunción lógica (AND).

### Espacio de Características

Los objetos se representan como vectores en el espacio  $\mathbb{R}^m$ :

$$\mathbf{x} = [d_1, d_2, \dots, d_m]^T \quad (2.46)$$

La clasificación divide este espacio en regiones mediante hiperplanos de decisión.

### Distancia de Mahalanobis

Para clasificación basada en similitud al prototipo de cada clase, se utiliza la distancia de Mahalanobis que considera la correlación entre características:

$$D_M(\mathbf{x}, c_i) = \sqrt{(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)} \quad (2.47)$$

donde  $\boldsymbol{\mu}_i$  es el vector de medias y  $\boldsymbol{\Sigma}_i$  la matriz de covarianza de la clase  $i$ .

La regla de clasificación asigna la clase de menor distancia:

$$\text{Clase}(x) = \arg \min_i D_M(\mathbf{x}, c_i) \quad (2.48)$$

## Análisis de Separabilidad de Clases

La efectividad de un descriptor para clasificación se cuantifica mediante métricas de separabilidad.

### Criterio de Fisher

Mide la razón entre varianza inter-clase e intra-clase:

$$J_F = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \quad (2.49)$$

Valores altos de  $J_F$  indican buena separabilidad. Para múltiples clases:

$$J_F = \frac{\sum_{i=1}^C N_i (\mu_i - \mu_{global})^2}{\sum_{i=1}^C N_i \sigma_i^2} \quad (2.50)$$

### Índice de Solapamiento

Calcula la proporción de distribuciones que se superponen:



Figura 2.12: Espacio de características bidimensional mostrando regiones de decisión

$$\text{Overlap}_{i,j} = \int_{-\infty}^{\infty} \min(P(d|c_i), P(d|c_j)) dd \quad (2.51)$$

Para distribuciones gaussianas, esta integral puede aproximarse mediante:

$$\text{Overlap}_{i,j} \approx 2\Phi \left( -\frac{|\mu_i - \mu_j|}{2\sqrt{\sigma_i^2 + \sigma_j^2}} \right) \quad (2.52)$$

donde  $\Phi$  es la función de distribución acumulada normal estándar. Valores cercanos a 0 indican separación perfecta; valores cercanos a 1 indican fuerte solapamiento.

## Análisis de Error y Desempeño

La tasa de error teórica para un clasificador basado en umbrales se deriva de las distribuciones de probabilidad.

### Error Tipo I y Tipo II

Para clasificación binaria con umbral  $t$ :

$$P(\text{Error Tipo I}) = P(d > t|c_1) = \int_t^{\infty} P(d|c_1) dd \quad (2.53)$$

$$P(\text{Error Tipo II}) = P(d \leq t|c_2) = \int_{-\infty}^t P(d|c_2) dd \quad (2.54)$$



Figura 2.13: Solapamiento entre distribuciones de dos clases y zona de error

La probabilidad total de error es:

$$P(\text{Error}) = P(c_1) \cdot P(\text{Error Tipo I}) + P(c_2) \cdot P(\text{Error Tipo II}) \quad (2.55)$$

donde  $P(c_i)$  son las probabilidades a priori de cada clase.

### Curva ROC Teórica

Variando el umbral  $t$ , se obtiene la curva ROC que relaciona sensibilidad y especificidad:

$$\text{TPR}(t) = \int_t^\infty P(d|c_{\text{positive}}) dd \quad (2.56)$$

$$\text{FPR}(t) = \int_t^\infty P(d|c_{\text{negative}}) dd \quad (2.57)$$

El área bajo la curva ROC cuantifica la capacidad discriminativa del descriptor.

## Validación Estadística del Clasificador

### Intervalo de Confianza para la Exactitud

La exactitud estimada de un clasificador tiene incertidumbre estadística. El intervalo de confianza al 95 % se calcula como:

$$\text{IC}_{95\%} = \hat{p} \pm 1,96 \sqrt{\frac{\hat{p}(1 - \hat{p})}{n}} \quad (2.58)$$

## Proyecto Final de Estudios: Robot cosechador automático

donde  $\hat{p}$  es la exactitud observada y  $n$  el tamaño de la muestra de prueba.

### Test de Hipótesis para Comparación de Clasificadores

Para determinar si un clasificador es significativamente mejor que otro, se emplea el test de McNemar para datos pareados:

$$\chi^2 = \frac{(n_{01} - n_{10})^2}{n_{01} + n_{10}} \quad (2.59)$$

donde  $n_{01}$  son muestras correctas por clasificador 1 e incorrectas por clasificador 2, y viceversa. Si  $\chi^2 > 3,84$ , la diferencia es significativa ( $p < 0.05$ ).

### Tamaño de Muestra Requerido

Para estimar parámetros estadísticos con precisión deseada, el tamaño de muestra mínimo se calcula como:

$$n_{min} = \left( \frac{z_{\alpha/2} \cdot \sigma}{E} \right)^2 \quad (2.60)$$

donde  $z_{\alpha/2}$  es el valor crítico (1.96 para 95 % confianza),  $\sigma$  la desviación estándar estimada y  $E$  el error máximo aceptable.



Figura 2.14: Convergencia de parámetros estadísticos con incremento del tamaño de muestra

## Robustez y Generalización

### Validación Cruzada

Para evaluar la generalización del clasificador, se emplea validación cruzada k-fold:



## Proyecto Final de Estudios: Robot cosechador automático

---

1. Dividir dataset en  $k$  particiones de igual tamaño
2. Para  $i = 1$  a  $k$ :
  - Usar partición  $i$  como conjunto de prueba
  - Usar restantes  $k - 1$  particiones para calcular parámetros
  - Evaluar desempeño en partición  $i$
3. Promediar métricas de desempeño

La exactitud estimada por validación cruzada es:

$$Acc_{CV} = \frac{1}{k} \sum_{i=1}^k Acc_i \quad (2.61)$$

### Sensibilidad a Variaciones

La robustez ante ruido se evalúa añadiendo perturbaciones controladas:

$$d_{noisy} = d_{true} + \mathcal{N}(0, \sigma_{noise}^2) \quad (2.62)$$

Un clasificador robusto mantiene exactitud alta incluso con  $\sigma_{noise}$  moderado. La degradación de desempeño se cuantifica como:

$$\Delta Acc = Acc_{clean} - Acc_{noisy} \quad (2.63)$$

### Análisis de Sensibilidad Paramétrica

Evaluar cómo cambios en umbrales afectan el desempeño:

$$\frac{\partial Acc}{\partial t_i} \approx \frac{Acc(t_i + \Delta t) - Acc(t_i - \Delta t)}{2\Delta t} \quad (2.64)$$

Umbrales con baja sensibilidad ( $|\partial Acc / \partial t_i|$  pequeño) son más robustos ante variaciones.

## Ventajas y Limitaciones del Enfoque Morfológico

### Ventajas:

- **Interpretabilidad:** Reglas de decisión explícitas y comprensibles
- **Eficiencia computacional:** Cálculos geométricos simples, ejecución en tiempo real
- **Pocos datos requeridos:** No requiere miles de muestras etiquetadas
- **Control explícito:** Ajuste directo de umbrales según requerimientos operativos
- **Depuración sencilla:** Fallas identificables mediante inspección de descriptores

### Limitaciones:

- **Diseño manual:** Requiere conocimiento experto del dominio
- **Limitada a características medibles:** No captura patrones visuales complejos
- **Sensibilidad a condiciones:** Variaciones de iluminación o perspectiva pueden afectar descriptores

- **Separabilidad lineal:** Dificultad con clases no linealmente separables

### Aplicabilidad en Agricultura de Precisión

El enfoque morfológico resulta particularmente adecuado para clasificación de cultivos cuando:

1. Existe diferenciación clara por tamaño (plantines vs. plantas maduras)
2. El entorno es controlado (cultivo hidropónico)
3. Se requiere operación en hardware con recursos limitados
4. La interpretabilidad del sistema es crítica para validación agronómica

La combinación con análisis estadístico robusto permite alcanzar desempeño comparable a métodos de aprendizaje profundo en escenarios donde las características geométricas son discriminativas.

## 2.2.4. Métricas de Evaluación para Sistemas de Clasificación

La evaluación cuantitativa del desempeño de sistemas de clasificación requiere métricas robustas que capturen diferentes aspectos de la calidad predictiva. En aplicaciones de agricultura de precisión, donde decisiones incorrectas pueden impactar la productividad del cultivo, resulta fundamental comprender las fortalezas y limitaciones de cada métrica.

### Matriz de Confusión

La matriz de confusión constituye la representación fundamental del desempeño de un clasificador, tabulando predicciones versus etiquetas verdaderas. Para un problema de  $C$  clases, la matriz de confusión  $\mathbf{M} \in \mathbb{R}^{C \times C}$  se define como:

$$M_{ij} = \sum_{k=1}^N \mathbb{I}[y_k = i \wedge \hat{y}_k = j] \quad (2.65)$$

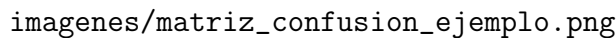
donde  $y_k$  es la clase verdadera de la muestra  $k$ ,  $\hat{y}_k$  es la clase predicha, y  $\mathbb{I}[\cdot]$  es la función indicadora.

Para clasificación binaria, la matriz se simplifica a:

$$\mathbf{M} = \begin{bmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{bmatrix} \quad (2.66)$$

donde:

- **TP (True Positives):** Casos positivos correctamente clasificados
- **TN (True Negatives):** Casos negativos correctamente clasificados
- **FP (False Positives):** Casos negativos incorrectamente clasificados como positivos (Error Tipo I)
- **FN (False Negatives):** Casos positivos incorrectamente clasificados como negativos (Error Tipo II)



imagenes/matriz\_confusion\_ejemplo.png

Figura 2.15: Matriz de confusión para clasificación multiclase mostrando patrones de error

### Exactitud (Accuracy)

La exactitud mide la proporción global de predicciones correctas:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\sum_{i=1}^C M_{ii}}{N} \quad (2.67)$$

#### Ventajas:

- Interpretación intuitiva: porcentaje de aciertos
- Métrica estándar para comparación entre modelos
- Apropiaada cuando todas las clases tienen igual importancia

#### Limitaciones Críticas

La exactitud resulta engañosa en datasets desbalanceados. Considere un clasificador que siempre predice la clase mayoritaria en un dataset con 95 % de negativos:

$$\text{Accuracy}_{naive} = \frac{0,95N}{N} = 0,95 \quad (2.68)$$

Este clasificador trivial alcanza 95 % de exactitud sin aprender ningún patrón útil. En agricultura, donde ciertas condiciones pueden ser raras pero críticas, la exactitud puede ocultar fallos graves del modelo.

### Precisión (Precision)

La precisión cuantifica la proporción de predicciones positivas que son correctas:

## Proyecto Final de Estudios: Robot cosechador automático

---

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2.69)$$

**Interpretación:** De todas las instancias que el modelo clasificó como positivas, ¿qué fracción realmente lo era?

Para clasificación multiclase, se calcula precisión por clase:

$$\text{Precision}_i = \frac{M_{ii}}{\sum_{j=1}^C M_{ji}} \quad (2.70)$$

La precisión es crucial cuando el costo de falsos positivos es alto. En cosecha automatizada, alta precisión garantiza que el robot no intente cosechar elementos no deseados.

### Exhaustividad (Recall/Sensitivity)

El recall mide la proporción de casos positivos reales que fueron correctamente identificados:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2.71)$$

**Interpretación:** De todas las instancias que realmente son positivas, ¿qué fracción detectó el modelo?

Para clasificación multiclase:

$$\text{Recall}_i = \frac{M_{ii}}{\sum_{j=1}^C M_{ij}} \quad (2.72)$$

El recall es fundamental cuando el costo de falsos negativos es alto. En detección de elementos para cosecha, alto recall asegura que no se pierdan productos maduros.

### F-Score

El F-Score armoniza precisión y recall mediante su media armónica:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \quad (2.73)$$

### Generalización F-Beta

Cuando se desea ponderar diferentemente precisión y recall:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \quad (2.74)$$

- $\beta < 1$ : Favorece precisión
- $\beta = 1$ : Balance equitativo (F1-Score)
- $\beta > 1$ : Favorece recall



Figura 2.16: Trade-off entre precisión y recall para diferentes valores de umbral

## Métricas Agregadas Multiclase

Para problemas con  $C > 2$  clases, las métricas por clase se agregan mediante:

### Macro-Average

Promedio simple de métricas por clase:

$$\text{Metric}_{macro} = \frac{1}{C} \sum_{i=1}^C \text{Metric}_i \quad (2.75)$$

Trata todas las clases por igual, apropiado cuando cada clase tiene igual importancia independientemente de su frecuencia.

### Weighted-Average

Promedio ponderado por soporte de cada clase:

$$\text{Metric}_{weighted} = \frac{1}{N} \sum_{i=1}^C n_i \cdot \text{Metric}_i \quad (2.76)$$

donde  $n_i = \sum_{j=1}^C M_{ij}$  es el número de instancias verdaderas de clase  $i$ .

### Micro-Average

Calcula métricas globales agregando TP, FP, FN de todas las clases:

$$\text{Precision}_{micro} = \frac{\sum_{i=1}^C \text{TP}_i}{\sum_{i=1}^C (\text{TP}_i + \text{FP}_i)} \quad (2.77)$$

## Métricas de Error Absoluto

Para clasificadores basados en descriptores continuos, es relevante evaluar el error en la estimación del descriptor mismo.

### Error Absoluto Medio (MAE)

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i| \quad (2.78)$$

Mide la magnitud promedio de los errores sin considerar su dirección.

### Error Cuadrático Medio (RMSE)

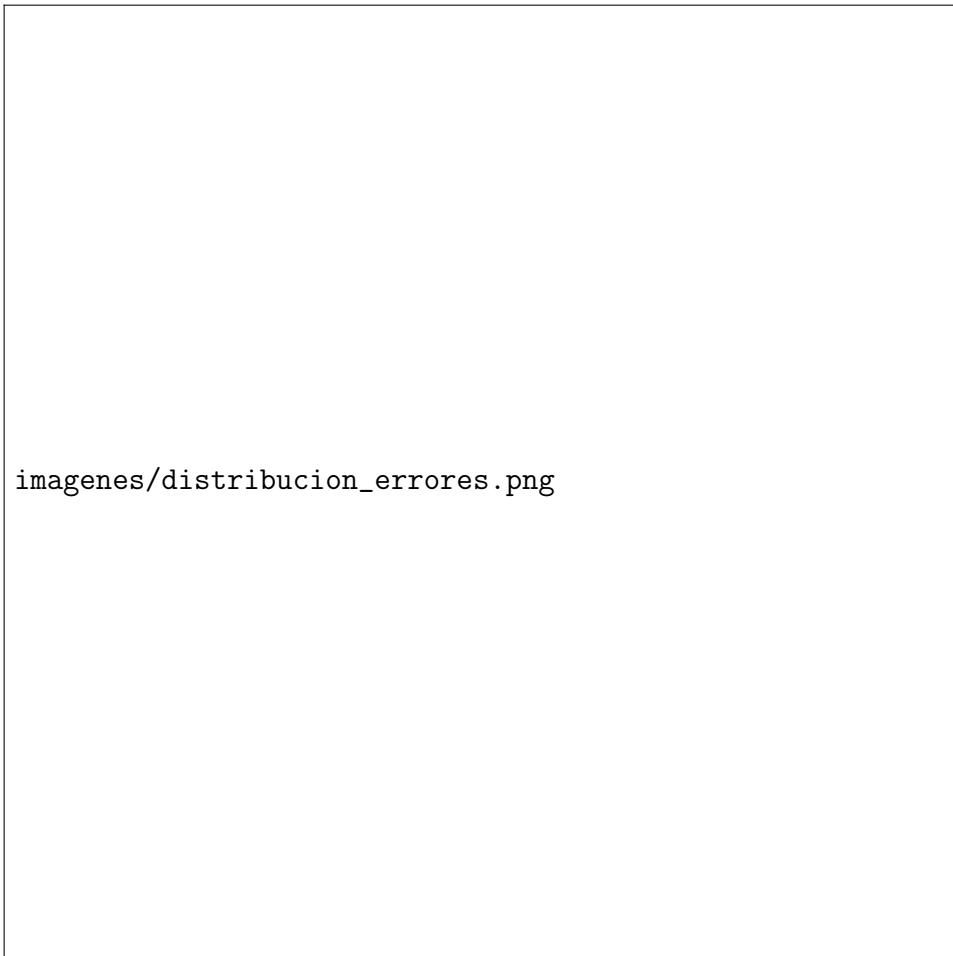
$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2} \quad (2.79)$$

Penaliza errores grandes más fuertemente que MAE debido a la elevación al cuadrado.

### Error Porcentual Absoluto Medio (MAPE)

$$\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (2.80)$$

Expresa el error como porcentaje del valor verdadero, útil para comparar desempeño en magnitudes diferentes.



imagenes/distribucion\_errores.png

Figura 2.17: Distribución de errores absolutos en la estimación de descriptores

## Coeficiente de Determinación

Para regresión de descriptores continuos, el coeficiente  $R^2$  mide la proporción de varianza explicada:

$$R^2 = 1 - \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2} \quad (2.81)$$

donde  $\bar{y}$  es la media de los valores observados.

### Interpretación:

- $R^2 = 1$ : Predicción perfecta
- $R^2 = 0$ : Modelo no mejor que predecir la media
- $R^2 < 0$ : Modelo peor que predecir la media

## Métricas de Confiabilidad

### Intervalo de Confianza

Para una métrica estimada  $\hat{\theta}$  con error estándar  $SE$ , el intervalo de confianza al  $(1 - \alpha)100\%$  es:

$$IC_{1-\alpha} = \hat{\theta} \pm z_{\alpha/2} \cdot SE \quad (2.82)$$

Para exactitud:

$$SE_{Acc} = \sqrt{\frac{\hat{p}(1 - \hat{p})}{n}} \quad (2.83)$$

### Coeficiente de Variación

Mide la variabilidad relativa de una métrica:

$$CV = \frac{\sigma}{\mu} \times 100\% \quad (2.84)$$

Valores bajos de CV indican consistencia del clasificador entre diferentes conjuntos de prueba.

## Métricas de Robustez

### Estabilidad ante Perturbaciones

Evalúa degradación de desempeño bajo ruido:

$$\Delta\text{Perf} = \frac{\text{Perf}_{clean} - \text{Perf}_{noisy}}{\text{Perf}_{clean}} \times 100\% \quad (2.85)$$

Sistemas robustos exhiben  $\Delta\text{Perf} < 10\%$  para niveles moderados de ruido.

### Repetibilidad

Desviación estándar de la métrica en ejecuciones repetidas:

$$\sigma_{rep} = \sqrt{\frac{1}{K-1} \sum_{k=1}^K (\text{Metric}_k - \bar{\text{Metric}})^2} \quad (2.86)$$

donde  $K$  es el número de repeticiones.



Figura 2.18: Degradación de exactitud con incremento de ruido gaussiano

### Selección de Métricas según Aplicación

Para sistemas de clasificación en agricultura de precisión, la selección de métricas debe alinearse con los objetivos operativos:

#### Prioridades Típicas:

1. **Alta Exhaustividad:** Minimizar productos no detectados
2. **Precisión Aceptable:** Limitar clasificaciones incorrectas
3. **Robustez:** Mantener desempeño bajo variaciones ambientales
4. **Consistencia:** Bajo coeficiente de variación entre sesiones

La métrica compuesta óptima puede definirse como:

$$\text{Score}_{total} = w_1 \cdot F_\beta + w_2 \cdot (1 - CV) + w_3 \cdot (1 - \Delta\text{Perf}) \quad (2.87)$$

donde  $w_i$  son pesos que reflejan la importancia relativa de cada aspecto, con  $\sum w_i = 1$ .



## **2.3. Cinemática de Robots Cartesianos**

## **2.4. Sistemas de Transmisión Mecánica**

## **2.5. Control de Motores Paso a Paso**



## **3. Desarrollo del Sistema**

### **3.1. Arquitectura General del Sistema**

### **3.2. Modelado y Diseño Mecánico**

#### **3.2.1. Especificaciones de Diseño y Restricciones**

#### **3.2.2. Análisis Cinemático del Sistema**

#### **3.2.3. Diseño Estructural**

#### **3.2.4. Sistema de Movimiento Horizontal**

#### **3.2.5. Sistema de Movimiento Vertical**

#### **3.2.6. Brazo Robótico**

#### **3.2.7. Modelado CAD y Fabricación**

### **3.3. Sistema de Control de Bajo Nivel (Nivel Regulatorio)**

#### **3.3.1. Arquitectura del Nivel Regulatorio**

#### **3.3.2. Hardware de Control - Arduino Mega 2560**

#### **3.3.3. Selección y Dimensionamiento de Actuadores**

#### **3.3.4. Sensores de Seguridad**

#### **3.3.5. Control de Movimiento**

#### **3.3.6. Protocolo de Comunicación UART**

### **3.4. Sistema de Supervisión y Alta Gestión (Nivel Supervisor)**

#### **3.4.1. Arquitectura del Nivel Supervisor**

#### **3.4.2. Hardware Supervisor - Raspberry Pi 4**

#### **3.4.3. Máquina de Estados Supervisora**

#### **3.4.4. Coordinación con Nivel Regulatorio**

### **3.5. Inteligencia Artificial y Visión por Computadora**

#### **3.5.1. Arquitectura y Pipeline de Procesamiento**

#### **Arquitectura Modular del Sistema**

Esta arquitectura modular permite la separación de responsabilidades, facilita el mantenimiento y posibilita la evolución independiente de cada componente.

El primer módulo, denominado **Sistema de Posicionamiento Visual**, tiene como función detectar desviaciones respecto a la posición ideal mediante el análisis de marcadores visuales de referencia. Este módulo calcula correcciones en tiempo real que compensan las holguras mecánicas acumulativas del sistema de movimiento, garantizando precisión submilimétrica en el alineamiento frente a cada estación de cultivo.

El segundo módulo corresponde al **Sistema de Clasificación de Cultivos**, responsable de determinar la presencia o ausencia de lechuga en cada estación mediante análisis morfológico de características extraídas de imágenes. La clasificación se basa en umbrales estadísticos derivados de un análisis exhaustivo de muestras representativas del entorno operativo.

El tercer módulo, **Sistema de Mapeo Autónomo**, construye una representación espacial completa del entorno de cultivo mediante exploración sistemática. Este módulo registra la posición y estado de cada estación, generando una base de datos espacial que sirve como entrada para la planificación de operaciones de cosecha.

Finalmente, el cuarto módulo de **Optimización de Trayectorias** procesa la información del mapa construido para generar rutas que minimizan el tiempo total de operación. Este módulo adapta el problema clásico del viajante a las restricciones geométricas y cinemáticas específicas del robot cartesiano.

### Flujo de Información entre Módulos

La información fluye siguiendo una jerarquía de control bien definida. El módulo de mapeo identifica las estaciones objetivo que requieren atención. Para cada estación identificada, el módulo de posicionamiento ejecuta las correcciones visuales necesarias para garantizar el alineamiento preciso. Una vez posicionado correctamente, el módulo de clasificación determina el estado del cultivo. Finalmente, basándose en esta información, el sistema supervisor toma la decisión de ejecutar la cosecha o avanzar a la siguiente estación. Este flujo puede expresarse mediante la secuencia:

$$\text{Mapeo} \rightarrow \text{Posicionamiento} \rightarrow \text{Clasificación} \rightarrow \text{Decisión} \quad (3.1)$$

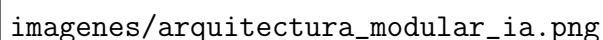
Cada módulo expone interfaces funcionales bien definidas que permiten la invocación de operaciones específicas y el intercambio de datos estructurados. El módulo de posicionamiento recibe como entrada la imagen capturada y el eje a corregir, retornando la desviación calculada en milímetros. El módulo de clasificación procesa una imagen y retorna un valor booleano indicando presencia o ausencia de lechuga junto con un índice de confianza. El módulo de mapeo ejecuta la exploración completa y retorna una matriz que contiene coordenadas y estados de todas las estaciones. El módulo de trayectorias recibe esta matriz como entrada y genera una lista ordenada de coordenadas que define la secuencia óptima de visita.

### Selección de Plataforma Computacional

La implementación del sistema de inteligencia artificial se realizó sobre una Raspberry Pi 4 Model B equipada con procesador ARM Cortex-A72 de cuatro núcleos operando a 1.5 GHz y 4 GB de memoria RAM tipo LPDDR4. Esta plataforma fue seleccionada por su capacidad de ejecutar algoritmos de visión por computadora en tiempo real manteniendo un consumo energético reducido, compatible con las restricciones del sistema embarcado del robot.

Las bibliotecas de software empleadas incluyen OpenCV versión 4.5 para procesamiento de imágenes y detección de contornos, NumPy versión 1.21 para operaciones matriciales de alto rendimiento necesarias en la calibración espacial, y PySerial para la comunicación mediante protocolo UART con el nivel regulatorio. La elección de Python como lenguaje de implementación se fundamenta en su amplio ecosistema de bibliotecas científicas, la velocidad de desarrollo y prototipado, y un rendimiento suficiente para las frecuencias de operación requeridas por el sistema.

El sistema operativo Raspberry Pi OS, basado en Debian, proporciona soporte nativo para los



imagenes/arquitectura\_modular\_ia.png

Figura 3.1: Arquitectura modular del sistema de inteligencia artificial mostrando la interconexión entre los cuatro módulos principales y el flujo de información

controladores de cámara mediante la interfaz Video4Linux2, permitiendo la captura de imágenes sin configuración adicional. La arquitectura ARM del procesador incluye extensiones SIMD que aceleran las operaciones vectoriales intensivas empleadas en el procesamiento de imágenes.

## Pipeline General de Procesamiento

El procesamiento de imágenes sigue un pipeline estructurado en siete etapas secuenciales que transforman las imágenes crudas capturadas en información accionable para el control del robot. Cada etapa aplica transformaciones específicas que progresivamente refinan la información hasta extraer las características relevantes para la toma de decisiones.

La primera etapa corresponde a la **adquisición de imágenes** RGB con resolución de  $1920 \times 1080$  píxeles mediante la cámara USB. Esta etapa presenta una latencia aproximada de 50 milisegundos que incluye el tiempo de exposición del sensor, la lectura de los datos del array de píxeles y la transferencia mediante interfaz USB 2.0 hacia la memoria del sistema.

La segunda etapa realiza el **preprocesamiento** mediante la conversión del espacio de color RGB al espacio HSV. Esta transformación, fundamentada en la teoría presentada en el marco teórico (Sec-

## Proyecto Final de Estudios: Robot cosechador automático

ción 2.2.1), separa la información cromática (matiz y saturación) de la información de luminosidad (valor), proporcionando robustez ante variaciones de iluminación ambiental. La conversión se ejecuta aplicando las ecuaciones de transformación estándar RGB-HSV sobre cada píxel de la imagen. El tiempo de procesamiento de esta operación es de aproximadamente 15 milisegundos, aprovechando las optimizaciones vectoriales de la biblioteca OpenCV.

La tercera etapa corresponde a la **segmentación**, donde se aplican operaciones de umbralización específicas según el módulo activo. Para el módulo de posicionamiento se extrae el canal V y se aplica umbralización inversa con valor de referencia de 50, destacando las regiones oscuras correspondientes a las cintas de referencia. Para el módulo de clasificación se emplea segmentación por rango cromático en los tres canales HSV, definiendo límites inferior y superior que aíslan las tonalidades verdes correspondientes a vegetación. Esta etapa requiere 20 milisegundos de procesamiento.

La cuarta etapa implementa **refinamiento morfológico** mediante operaciones de cierre y apertura, aplicando los fundamentos teóricos descritos en la Sección 2.2.1 del marco teórico. La operación de cierre elimina pequeños huecos dentro de las regiones de interés, mientras que la apertura elimina píxeles aislados correspondientes a ruido. Ambas operaciones emplean un elemento estructurante rectangular de  $3 \times 3$  píxeles. El costo computacional total de estas operaciones morfológicas es de 25 milisegundos.

La quinta etapa ejecuta la **detección de contornos** mediante el algoritmo de Suzuki-Abe, identificando las fronteras de las regiones segmentadas. Este algoritmo, cuya fundamentación matemática se presentó en la Sección 2.2.2, construye una representación jerárquica de los contornos detectados. Se emplea el modo de extracción de contornos externos únicamente, ya que las estructuras de interés no presentan huecos internos relevantes. La aproximación de contornos se realiza mediante compresión de puntos colineales, reduciendo el uso de memoria. Esta operación requiere aproximadamente 30 milisegundos de procesamiento.

La sexta etapa calcula **descriptores geométricos** de los contornos detectados. Se calculan el área mediante conteo de píxeles interiores, el centroide mediante momentos de imagen de orden cero y uno, el perímetro mediante suma de distancias euclidianas entre puntos consecutivos, y el rectángulo delimitador que encierra completamente al contorno. Estos descriptores constituyen las características cuantitativas empleadas para la toma de decisiones en los módulos de posicionamiento y clasificación. El tiempo de cálculo es de aproximadamente 10 milisegundos.

Finalmente, la séptima etapa genera el **comando de acción** correspondiente basándose en las características extraídas. Para el módulo de posicionamiento, esto corresponde al cálculo de la desviación en píxeles respecto al centro de la imagen y su posterior conversión a milímetros mediante los coeficientes de calibración. Para el módulo de clasificación, corresponde a la evaluación del área del contorno principal contra el umbral estadístico establecido, resultando en una decisión binaria de presencia o ausencia de lechuga. Esta etapa final requiere menos de 5 milisegundos.

El tiempo total de procesamiento del pipeline completo se obtiene mediante la suma de las latencias individuales:

$$T_{total} = T_{adq} + T_{prep} + T_{seg} + T_{mor} + T_{cont} + T_{desc} + T_{dec} \quad (3.2)$$

$$T_{total} = 50 + 15 + 20 + 25 + 30 + 10 + 5 = 155 \text{ ms} \quad (3.3)$$

Esta latencia total permite una frecuencia de operación de:

$$f = \frac{1}{T_{total}} = \frac{1}{0,155} \approx 6,5 \text{ Hz} \quad (3.4)$$

Esta frecuencia es suficiente para el control en tiempo real considerando que las velocidades de movimiento del robot se encuentran en el rango de 5 a 10 cm/s, lo que implica desplazamientos de 7.75 a 15.5 mm entre capturas consecutivas.

## Proyecto Final de Estudios: Robot cosechador automático

---

El pipeline implementa un diseño adaptable que modifica su configuración según el módulo activo. Para el módulo de posicionamiento se utiliza únicamente el canal V del espacio HSV con umbralización inversa, enfocándose en la detección de elementos de alto contraste. Para el módulo de clasificación se emplean los tres canales HSV con segmentación por rango cromático, permitiendo la discriminación de vegetación verde. Esta adaptabilidad permite optimizar el procesamiento según los requerimientos específicos de cada tarea, manteniendo la latencia dentro de límites aceptables para el control en tiempo real.

### Sistema de Adquisición Visual

El hardware de adquisición de imágenes constituye el elemento sensorial primario del sistema de inteligencia artificial. La selección del dispositivo de captura se basó en criterios de resolución, frecuencia de muestreo, compatibilidad con la plataforma de procesamiento y costo.

Se implementó una cámara USB Logitech C920 HD Pro que proporciona captura de imágenes con resolución máxima de  $1920 \times 1080$  píxeles a una tasa de 30 cuadros por segundo. El sensor de imagen tipo CMOS de 1/2.7 pulgadas presenta sensibilidad adecuada para las condiciones de iluminación del invernadero, con valores típicos de 800 a 1200 lux proporcionados por iluminación LED blanca de 5000K instalada en el techo de la estructura.

El campo de visión de 78 grados en diagonal permite capturar un área aproximada de  $280 \times 160$  milímetros a la distancia de trabajo nominal de 200 milímetros. Esta configuración óptica resulta en una resolución espacial de 0.146 mm/píxel en dirección horizontal y 0.148 mm/píxel en dirección vertical, calculadas como el cociente entre las dimensiones físicas del área capturada y la resolución en píxeles correspondiente.

La cámara incorpora sistema de enfoque automático que ajusta la posición de las lentes para mantener la nitidez de la imagen cuando varía la distancia de trabajo. Este mecanismo es crítico dado que el robot opera en un rango de distancias de 150 a 250 milímetros durante las diferentes fases de operación. El tiempo de convergencia del autofocus es inferior a 2 segundos cuando se produce un cambio significativo de distancia.

El montaje de la cámara se realizó mediante un soporte diseñado específicamente e impreso en material PLA mediante manufactura aditiva. El soporte se fija al extremo del brazo robótico y permite ajuste angular de  $\pm 15$  grados en los ejes de cabeceo y guiñada. Esta capacidad de ajuste es necesaria para compensar tolerancias de ensamblaje y optimizar la orientación del campo de visión respecto al plano de cultivo.

La interfaz entre la cámara y la Raspberry Pi se establece mediante conexión USB 2.0, que proporciona ancho de banda suficiente para la transferencia de imágenes sin compresión a la tasa de captura requerida. El controlador Video4Linux2, incluido nativamente en el sistema operativo, gestiona la comunicación con el dispositivo y expone las funciones de captura mediante una interfaz de programación estándar.

La latencia total del sistema de adquisición, medida desde la solicitud de captura hasta la disponibilidad de la imagen en memoria, comprende cuatro componentes: tiempo de exposición del sensor (típicamente 8 a 15 milisegundos dependiendo de las condiciones de iluminación), tiempo de lectura del array de píxeles (aproximadamente 8 milisegundos), tiempo de transferencia USB (15 milisegundos para una imagen de 6.2 megabytes), y tiempo de decodificación en el procesador host (10 milisegundos). La latencia total resultante es de aproximadamente 45 a 50 milisegundos.

### Preprocesamiento de Imágenes

Las imágenes capturadas requieren transformaciones preliminares que optimizan su procesamiento posterior y mejoran la robustez del sistema ante variaciones ambientales. El preprocesamiento constituye la etapa de acondicionamiento de señal del pipeline de visión artificial.

La primera operación de preprocesamiento consiste en la conversión del espacio de color RGB nativo de la cámara al espacio HSV. Esta transformación aplica las ecuaciones de conversión estándar presentadas en el marco teórico (Sección 2.2.1), calculando para cada píxel los valores de matiz, saturación y valor a partir de las componentes rojo, verde y azul. La separación de información cromática e información de luminosidad que proporciona esta representación resulta fundamental para la robustez del sistema.

El módulo de posicionamiento visual explota esta separación extrayendo únicamente el canal V, que representa el brillo o luminosidad de cada píxel independientemente de su color. Esta estrategia proporciona invariancia ante cambios en la tonalidad de la iluminación, permitiendo la detección de cintas negras de referencia incluso cuando el espectro de la fuente luminosa varía. El canal V se extrae mediante indexación directa del tercer plano de la imagen transformada.

El módulo de clasificación de cultivos, por el contrario, emplea los tres canales simultáneamente para realizar segmentación por rango cromático. Esta operación evalúa para cada píxel si sus valores de matiz, saturación y valor se encuentran dentro de límites predefinidos que corresponden a las tonalidades verdes características de la vegetación. Los píxeles que cumplen esta condición se marcan en una máscara binaria, aislando efectivamente las regiones de interés del resto de la escena.

Para optimizar el rendimiento computacional se implementa recorte adaptativo de región de interés. Este mecanismo limita el procesamiento a una ventana rectangular centrada en la posición esperada del elemento a detectar, reduciendo significativamente el número de píxeles que deben procesarse. Las dimensiones de la región de interés se determinan en función de la posición actual del robot y del conocimiento a priori de la geometría del sistema. Las coordenadas de los límites de la región se calculan mediante:

$$x_{min} = \text{máx}(0, x_{centro} - w_{ROI}/2) \quad (3.5)$$

$$x_{max} = \text{mín}(W_{imagen}, x_{centro} + w_{ROI}/2) \quad (3.6)$$

$$y_{min} = \text{máx}(0, y_{centro} - h_{ROI}/2) \quad (3.7)$$

$$y_{max} = \text{mín}(H_{imagen}, y_{centro} + h_{ROI}/2) \quad (3.8)$$

donde  $W_{imagen}$  y  $H_{imagen}$  son las dimensiones totales de la imagen capturada,  $(x_{centro}, y_{centro})$  es la posición estimada del elemento de interés, y  $w_{ROI}$  y  $h_{ROI}$  son las dimensiones de la región de interés. Las funciones de máximo y mínimo garantizan que los límites no excedan los bordes de la imagen.

Esta estrategia reduce el área de procesamiento en aproximadamente 65 por ciento, mejorando proporcionalmente los tiempos de respuesta del sistema. El recorte se ejecuta mediante operaciones de indexación de arrays que no implican copia de datos, resultando en un costo computacional despreciable.

## Segmentación por Umbralización

La segmentación constituye la operación fundamental que separa las regiones de interés del fondo de la imagen. Se implementan dos estrategias de umbralización adaptadas a los requerimientos específicos de cada módulo.

Para el módulo de posicionamiento visual se aplica umbralización inversa sobre el canal V del espacio HSV. Esta operación compara el valor de brillo de cada píxel contra un umbral de referencia, asignando valor máximo (255) a los píxeles cuyo brillo es inferior al umbral, y valor nulo (0) a los píxeles con brillo superior. Matemáticamente:

$$I_{bin}(x, y) = \begin{cases} 255 & \text{si } I_V(x, y) < T \\ 0 & \text{si } I_V(x, y) \geq T \end{cases} \quad (3.9)$$



El valor de umbral  $T = 50$  se determinó empíricamente mediante análisis del histograma de intensidades en imágenes representativas del entorno operativo. Este valor maximiza la separación entre las cintas negras de referencia y el fondo blanco del sistema hidropónico, minimizando simultáneamente la tasa de falsos positivos.

Para el módulo de clasificación se emplea segmentación por rango en el espacio tridimensional HSV. Cada píxel se evalúa contra límites inferior y superior en los tres canales, asignándose a la región de interés únicamente si todas las componentes se encuentran dentro del rango especificado:

$$M(x, y) = \begin{cases} 255 & \text{si } H_{min} \leq H(x, y) \leq H_{max} \\ & \wedge S_{min} \leq S(x, y) \leq S_{max} \\ & \wedge V_{min} \leq V(x, y) \leq V_{max} \\ 0 & \text{en caso contrario} \end{cases} \quad (3.10)$$

Los límites se establecieron como  $H_{min} = 25$ ,  $H_{max} = 85$  para cubrir el espectro de verdes desde tonalidades amarillentas hasta azuladas;  $S_{min} = 40$  para eliminar colores desaturados del fondo que podrían presentar componente verde débil; y  $V_{min} = 40$  para descartar sombras muy oscuras. Los límites superiores de saturación y valor se fijan en sus valores máximos (255).

## Refinamiento Morfológico

Las máscaras binarias resultantes de la segmentación contienen típicamente ruido aislado y discontinuidades en las regiones de interés debido a variaciones locales de iluminación, reflexiones especulares y limitaciones del sensor. Se aplica una secuencia de operaciones morfológicas para refinar estas máscaras.

La primera operación corresponde al cierre morfológico, que elimina pequeños huecos dentro de las regiones y conecta componentes próximos. Matemáticamente, el cierre se define como una dilatación seguida de una erosión:

$$M_{cierre} = (M \oplus K) \ominus K \quad (3.11)$$

donde  $M$  es la máscara binaria de entrada,  $K$  es el elemento estructurante,  $\oplus$  denota dilatación y  $\ominus$  denota erosión. La dilatación expande las regiones blancas, rellendo huecos pequeños, mientras que la erosión subsecuente restaura aproximadamente el tamaño original de las regiones.

La segunda operación corresponde a la apertura morfológica, que elimina píxeles aislados y pequeñas protuberancias. Se define como una erosión seguida de una dilatación:

$$M_{apertura} = (M_{cierre} \ominus K) \oplus K \quad (3.12)$$

La erosión elimina estructuras pequeñas que no pueden contener completamente el elemento estructurante, y la dilatación subsecuente restaura el tamaño de las estructuras remanentes.

El elemento estructurante empleado es una matriz rectangular de  $3 \times 3$  píxeles con todos sus elementos igual a uno. Este tamaño se seleccionó como compromiso entre capacidad de eliminación de ruido y preservación de detalles relevantes. Elementos estructurantes mayores eliminarían ruido más efectivamente pero podrían degradar características geométricas importantes de las regiones.

El resultado de estas operaciones morfológicas es una máscara binaria refinada donde las regiones de interés presentan fronteras suaves y conectividad mejorada, facilitando la subsecuente detección de contornos.

## Detección y Análisis de Contornos

Sobre la máscara refinada se ejecuta el algoritmo de detección de contornos de Suzuki-Abe, cuyo fundamento teórico se presentó en la Sección 2.2.2 del marco teórico. Este algoritmo recorre la imagen

binaria identificando las fronteras entre regiones blancas y negras, construyendo una representación vectorial de cada contorno como secuencia ordenada de coordenadas de píxeles.

Se emplea el modo de extracción de contornos externos únicamente, que detecta solamente el contorno exterior de cada región conexa ignorando posibles huecos internos. Esta configuración es apropiada dado que los objetos de interés (lechugas y cintas de referencia) no presentan estructuras anidadas relevantes para la aplicación. La aproximación de contornos se realiza mediante compresión de segmentos colineales, donde secuencias de puntos que forman líneas rectas se representan únicamente por sus extremos. Esta estrategia reduce significativamente el uso de memoria y acelera operaciones posteriores sobre los contornos.

Los contornos detectados se someten a filtrado para eliminar aquellos que no corresponden a objetos de interés. El criterio principal de filtrado es el área, calculada como el número de píxeles encerrados por el contorno. Para el módulo de posicionamiento se descartan contornos con área inferior a 500 píxeles, mientras que para el módulo de clasificación el umbral es de 5000 píxeles. Estos valores se establecieron considerando el tamaño esperado de los objetos de interés a la distancia de trabajo nominal.

Para cada contorno que supera el filtrado se calculan descriptores geométricos que cuantifican sus características. El área se obtiene mediante conteo de píxeles interiores empleando el algoritmo de relleno por escaneo de líneas. El centroide se calcula mediante los momentos de imagen de orden cero y uno:

$$c_x = \frac{M_{10}}{M_{00}}, \quad c_y = \frac{M_{01}}{M_{00}} \quad (3.13)$$

donde  $M_{ij}$  representa el momento de orden  $i + j$  definido como:

$$M_{ij} = \sum_{(x,y) \in \text{Región}} x^i y^j \quad (3.14)$$

El perímetro se calcula como la suma de distancias euclidianas entre puntos consecutivos del contorno. El rectángulo delimitador se determina identificando las coordenadas mínimas y máximas en ambas direcciones.

Cuando múltiples contornos cumplen los criterios de filtrado, se implementa un sistema de selección del contorno principal. Para el módulo de posicionamiento se emplea scoring ponderado que considera simultáneamente el área del contorno, su posición relativa al centro de la imagen, y la calidad de su región basal. Para el módulo de clasificación se selecciona simplemente el contorno de mayor área, bajo la hipótesis de que la lechuga constituye el objeto verde dominante en la escena.

El tiempo total de procesamiento de la etapa de detección y análisis de contornos es de aproximadamente 40 milisegundos, incluyendo la detección, el filtrado y el cálculo de descriptores. Este tiempo es compatible con los requerimientos de latencia del sistema de control en tiempo real.

### 3.5.2. Sistema de Posicionamiento Visual

#### Fundamento del Sistema de Posicionamiento Visual

El sistema mecánico del robot presenta holguras acumulativas que generan errores de posicionamiento de hasta  $\pm 5$  milímetros respecto a la posición comandada. Estos errores resultan incompatibles con los requerimientos de precisión para las operaciones de cosecha, que exigen tolerancias inferiores a  $\pm 2$  milímetros. Para compensar estas desviaciones se implementó un sistema de corrección visual basado en la detección de marcadores de referencia.

Los marcadores consisten en cintas adhesivas negras de 18 milímetros de ancho adheridas sobre la superficie de los tubos de PVC blanco del sistema hidropónico. Esta configuración proporciona un contraste óptico superior a 0.85 entre el elemento oscuro y el fondo claro, facilitando su detección

mediante técnicas de procesamiento de imágenes. La selección de cintas adhesivas como elemento de marcado se fundamentó en su bajo costo de implementación (inferior a 5 dólares estadounidenses por metro lineal), su robustez ante las condiciones de humedad y temperatura del invernadero, y la facilidad de instalación sin modificaciones estructurales del sistema.

Las cintas se disponen en dos orientaciones: cintas horizontales que proporcionan referencia para corrección en el eje vertical, y cintas verticales que proporcionan referencia para corrección en el eje horizontal. Esta configuración dual permite la corrección independiente en ambos ejes de movimiento del robot.

## Metodología de Detección

El algoritmo de detección de cintas explota el alto contraste entre el marcador negro y el fondo claro mediante procesamiento en el canal de valor (V) del espacio HSV. Esta elección se fundamenta en la invariancia del canal V ante cambios en la tonalidad cromática de la iluminación, propiedad demostrada en el marco teórico (Sección 2.2.1).

El proceso inicia con la captura de una imagen RGB de la región donde se espera encontrar el marcador. La imagen se transforma al espacio HSV aplicando las ecuaciones de conversión estándar, y se extrae el canal V que representa el brillo de cada píxel. Sobre este canal se aplica umbralización inversa con valor de referencia de 50, operación que asigna valor máximo (blanco) a los píxeles oscuros y valor nulo (negro) a los píxeles claros. Matemáticamente:

$$I_{bin}(x, y) = \begin{cases} 255 & \text{si } I_V(x, y) < 50 \\ 0 & \text{si } I_V(x, y) \geq 50 \end{cases} \quad (3.15)$$

El valor de umbral 50 se determinó empíricamente mediante análisis del histograma de intensidades en múltiples imágenes representativas, identificando el punto que maximiza la separación entre la distribución de intensidades de las cintas negras y la distribución de intensidades del fondo blanco.

Sobre la imagen binaria resultante se ejecuta el algoritmo de detección de contornos de Suzuki-Abe, obteniendo las fronteras de todas las regiones oscuras detectadas. Los contornos se filtran aplicando un criterio de área mínima de 500 píxeles, eliminando elementos espurios correspondientes a ruido o pequeñas sombras. Este umbral se estableció considerando que una cinta de 18 milímetros de ancho a 200 milímetros de distancia subtiende aproximadamente 120 píxeles de ancho en la imagen, resultando en áreas típicas superiores a 3000 píxeles para segmentos de cinta de longitud razonable.

## Evaluación de Calidad del Contorno

Los contornos que superan el filtrado de área se someten a evaluación de calidad para discriminar entre cintas de referencia genuinas y otros elementos oscuros presentes en la escena (como plantas o vasos). La evaluación se basa en el análisis de la región basal del contorno, definida como el 10 por ciento inferior de su rectángulo delimitador.

Para cada contorno candidato se calcula su rectángulo delimitador  $(x, y, w, h)$ , donde  $(x, y)$  representa la esquina superior izquierda,  $w$  el ancho y  $h$  la altura. La región basal se define entonces como:

$$R_{base} = \{(x', y') : x \leq x' < x + w, y + 0,9h \leq y' < y + h\} \quad (3.16)$$

Se calcula la fracción de píxeles blancos en esta región respecto al total de píxeles de la región:

$$Q_{base} = \frac{1}{|R_{base}|} \sum_{(x, y) \in R_{base}} \frac{I_{bin}(x, y)}{255} \quad (3.17)$$

## Proyecto Final de Estudios: Robot cosechador automático

donde  $|R_{base}|$  denota el número de píxeles en la región. Un valor  $Q_{base}$  cercano a 1 indica que la región basal del contorno es consistentemente oscura, característica de las cintas de referencia cuyo borde inferior es nítido y bien definido. Valores inferiores indican contornos de objetos con bases irregulares o discontinuas, como plantas con hojas que no presentan un borde horizontal claro.

Esta métrica resulta efectiva para distinguir cintas de otros elementos oscuros: las cintas genuinas presentan típicamente  $Q_{base} > 0,8$ , mientras que plantas u otros objetos tienen  $Q_{base} < 0,5$ . Se establece un umbral de aceptación de 0.7 como compromiso entre sensibilidad y especificidad.

### Sistema de Scoring Multicriteria

Cuando múltiples contornos cumplen los criterios de área mínima y calidad basal, se implementa un sistema de scoring ponderado que considera tres factores para seleccionar el mejor candidato.

El primer factor evalúa el área del contorno normalizada respecto a los límites mínimo y máximo esperados:

$$S_{\text{área}} = \frac{A - A_{\min}}{A_{\max} - A_{\min}} \quad (3.18)$$

donde  $A$  es el área del contorno,  $A_{\min} = 500$  píxeles y  $A_{\max} = 50000$  píxeles. Este factor favorece contornos con áreas intermedias, descartando elementos excesivamente pequeños (ruido) o excesivamente grandes (probablemente no corresponden a una cinta individual).

El segundo factor evalúa la posición del centroide del contorno respecto al centro de la imagen:

$$S_{\text{posición}} = 1 - \frac{|c_x - x_{\text{centro}}|}{W_{\text{imagen}}/2} \quad (3.19)$$

donde  $c_x$  es la coordenada horizontal del centroide,  $x_{\text{centro}}$  es la coordenada horizontal del centro de la imagen, y  $W_{\text{imagen}}$  es el ancho total de la imagen. Este factor favorece contornos cercanos al centro, bajo la hipótesis de que el sistema de control posiciona aproximadamente el robot frente a la cinta objetivo, resultando en desviaciones pequeñas.

El tercer factor corresponde directamente a la calidad basal  $Q_{base}$  calculada previamente, que favorece contornos con bordes inferiores bien definidos.

El score total se calcula como combinación lineal ponderada de estos tres factores:

$$S_{\text{total}} = w_1 \cdot S_{\text{área}} + w_2 \cdot S_{\text{posición}} + w_3 \cdot Q_{base} \quad (3.20)$$

Los pesos se establecieron como  $w_1 = 0,3$ ,  $w_2 = 0,4$  y  $w_3 = 0,3$ , otorgando mayor importancia a la posición dado que las desviaciones típicas son inferiores a 50 milímetros, resultando en centroides próximos al centro de la imagen. El contorno con mayor score total se selecciona como la cinta de referencia.

### Cálculo de Desviación

Una vez identificado el contorno óptimo se procede al cálculo de la desviación espacial. Se determina el centroide del contorno mediante sus momentos de imagen:

$$c_x = \frac{M_{10}}{M_{00}}, \quad c_y = \frac{M_{01}}{M_{00}} \quad (3.21)$$

La desviación en píxeles se calcula como la diferencia entre las coordenadas del centroide y las coordenadas del centro de la imagen:

$$\Delta x_{px} = c_x - \frac{W_{\text{imagen}}}{2} \quad (3.22)$$

$$\Delta y_{px} = c_y - \frac{H_{imagen}}{2} \quad (3.23)$$

Para una imagen de  $1920 \times 1080$  píxeles, el centro se ubica en (960, 540). Un centroide detectado en (1100, 540) resultaría en  $\Delta x_{px} = 140$  píxeles, indicando que el robot se encuentra desplazado 140 píxeles hacia la izquierda respecto a la posición ideal.

Estas desviaciones en píxeles se convierten posteriormente a milímetros mediante los coeficientes de calibración, proceso que se detalla en la siguiente sección.

### Discriminación de Orientación

Para determinar si un contorno detectado corresponde a una cinta horizontal o vertical se analiza la relación de aspecto de su rectángulo delimitador. Esta relación se define como:

$$AR = \frac{w}{h} \quad (3.24)$$

donde  $w$  y  $h$  son el ancho y altura del rectángulo delimitador. Cintas horizontales, al tener su dimensión principal en dirección horizontal, presentan  $AR > 2$ . Cintas verticales, con su dimensión principal en dirección vertical, presentan  $AR < 0,5$ . Contornos con relaciones de aspecto intermedias ( $0,5 \leq AR \leq 2$ ) se consideran ambiguos y se descartan.

Este análisis permite al sistema identificar automáticamente el tipo de cinta detectada, información necesaria para interpretar correctamente la desviación calculada. Una desviación horizontal del centroide respecto al centro indica desplazamiento en X cuando se detecta una cinta vertical, o desplazamiento en Y cuando se detecta una cinta horizontal.

### Desempeño del Algoritmo

El tiempo de ejecución promedio del algoritmo completo de detección de cintas es de 95 milisegundos, desglosado en: conversión al espacio HSV (15 ms), extracción del canal V (2 ms), umbralización (8 ms), detección de contornos (30 ms), filtrado inicial (5 ms), evaluación de calidad basal (20 ms), cálculo de scores (10 ms), y determinación del centroide (5 ms).

Esta latencia permite una frecuencia de actualización de aproximadamente 10 Hz, suficiente para compensar errores de posicionamiento durante el movimiento del robot a velocidades de 5 a 10 cm/s. En pruebas de validación con 200 imágenes de cintas bajo diferentes condiciones de iluminación, el algoritmo alcanzó una tasa de detección correcta del 97.5 por ciento, con fallos principalmente en casos de iluminación extremadamente baja (inferior a 400 lux) donde el contraste se reduce significativamente.

### Fundamentación de la Calibración Espacial

Las desviaciones detectadas mediante el sistema de visión se expresan inicialmente en unidades de píxeles, magnitud que carece de significado físico directo para el sistema de control de movimiento. Se requiere establecer una correspondencia entre coordenadas en el plano de la imagen y coordenadas en el espacio físico del robot. Esta transformación se logra mediante un proceso de calibración que determina los factores de conversión píxel-milímetro.

La calibración se fundamenta en la geometría proyectiva de formación de imágenes. Bajo la aproximación de modelo de cámara estenopeica, existe una relación lineal entre distancias en el plano objeto (espacio físico) y distancias en el plano imagen (espacio de píxeles), siempre que la distancia focal y la distancia objeto-cámara permanezcan constantes. Esta relación se expresa mediante:

$$d_{mm} = K \cdot d_{px} \quad (3.25)$$

donde  $d_{mm}$  es la distancia física en milímetros,  $d_{px}$  es la distancia correspondiente en píxeles, y  $K$  es el factor de conversión con unidades mm/píxel.

## Metodología de Calibración

El proceso de calibración empleó un patrón de referencia con dimensiones conocidas colocado en el plano de trabajo a la distancia nominal de 200 milímetros desde el sensor de la cámara. El patrón consistió en un conjunto de marcas separadas por distancias calibradas mediante instrumentos de metrología de precisión.

Se capturaron imágenes del patrón y se identificaron las posiciones en píxeles de las marcas de referencia mediante detección de contornos. Para cada par de marcas consecutivas se registró la distancia física conocida  $d_{mm,i}$  y la distancia medida en píxeles  $d_{px,i}$ . Se recolectaron  $N = 20$  mediciones de distancias que cubrían el rango completo del espacio de trabajo.

El factor de conversión óptimo se determinó mediante el método de mínimos cuadrados, que minimiza la suma de errores cuadráticos entre las distancias físicas reales y las distancias estimadas mediante el modelo lineal. Para el caso univariable (factor sin offset), la solución se obtiene como:

$$K = \frac{\sum_{i=1}^N d_{px,i} \cdot d_{mm,i}}{\sum_{i=1}^N d_{px,i}^2} \quad (3.26)$$

Esta expresión proporciona el estimador de máxima verosimilitud del factor de conversión bajo el supuesto de errores gaussianos independientes.

Para considerar posibles offsets sistemáticos, se empleó el modelo lineal bivariable:

$$d_{mm} = k \cdot d_{px} + b \quad (3.27)$$

donde  $b$  representa un offset constante. Los parámetros se determinan resolviendo el sistema de ecuaciones normales:

$$\begin{bmatrix} k \\ b \end{bmatrix} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (3.28)$$

donde la matriz de diseño  $\mathbf{X}$  contiene las distancias en píxeles y una columna de unos, y el vector  $\mathbf{y}$  contiene las distancias físicas correspondientes:

$$\mathbf{X} = \begin{bmatrix} d_{px,1} & 1 \\ d_{px,2} & 1 \\ \vdots & \vdots \\ d_{px,N} & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} d_{mm,1} \\ d_{mm,2} \\ \vdots \\ d_{mm,N} \end{bmatrix} \quad (3.29)$$

El análisis de los datos de calibración reveló que el offset  $b$  resultó despreciable (inferior a 0.1 mm), validando la hipótesis de relación puramente proporcional entre coordenadas de píxeles y coordenadas físicas. En consecuencia, se adoptó el modelo simplificado sin offset.

## Resultados de Calibración

Los factores de conversión obtenidos fueron:

$$K_x = 0,146 \text{ mm/px} \quad (3.30)$$

$$K_y = 0,148 \text{ mm/px} \quad (3.31)$$

La diferencia entre los factores en direcciones horizontal y vertical se atribuye a pequeñas distorsiones ópticas del sistema de lentes y a la geometría no perfectamente cuadrada de los píxeles del sensor CMOS. Esta diferencia del 1.4 por ciento es despreciable para la mayoría de aplicaciones, pero se mantienen factores independientes para maximizar la precisión.

La bondad de ajuste del modelo lineal se evaluó mediante el coeficiente de determinación:

$$R^2 = 1 - \frac{\sum_{i=1}^N (d_{mm,i} - \hat{d}_{mm,i})^2}{\sum_{i=1}^N (d_{mm,i} - \bar{d}_{mm})^2} \quad (3.32)$$

donde  $\hat{d}_{mm,i} = K \cdot d_{px,i}$  son las distancias estimadas y  $\bar{d}_{mm}$  es la media de las distancias físicas. Se obtuvo  $R^2 = 0,998$ , indicando que el modelo lineal explica el 99.8 por ciento de la varianza de los datos, validando su idoneidad.

## Validación Estadística del Sistema

Para validar la precisión del sistema calibrado se ejecutaron 100 mediciones independientes de posiciones conocidas en el espacio de trabajo. Cada medición consistió en posicionar manualmente el robot en una coordenada verificada mediante calibre digital de precisión, capturar una imagen del marcador de referencia, detectar su posición en píxeles, convertir a milímetros mediante los factores de calibración, y calcular el error respecto a la posición real.

Los resultados estadísticos fueron:

- Media del error absoluto:  $\mu_{error} = 0,52$  mm
- Desviación estándar:  $\sigma = 0,8$  mm
- Error máximo observado:  $e_{max} = 1,35$  mm
- Error mínimo observado:  $e_{min} = 0,05$  mm

La distribución de errores presentó forma aproximadamente gaussiana centrada cerca de cero, validando la hipótesis de errores aleatorios sin sesgo sistemático. El 95 por ciento de las mediciones presentaron error inferior a 1.0 mm, cumpliendo con el requerimiento de precisión establecido para las operaciones de cosecha (tolerancia de  $\pm 2$  mm).

El error máximo de 1.35 mm se encuentra dentro de los límites aceptables y se atribuye a limitaciones de resolución del sensor (cada píxel representa aproximadamente 0.146 mm) y a pequeñas vibraciones residuales del sistema mecánico durante la captura.

## Algoritmo de Corrección Iterativa

El sistema de corrección opera mediante un esquema iterativo que ejecuta ciclos de detección-corrección hasta alcanzar convergencia. Cada iteración comprende las siguientes operaciones: captura de imagen en la posición actual, detección del marcador de referencia mediante el algoritmo descrito previamente, cálculo de la desviación en píxeles, conversión a milímetros mediante los factores de calibración, envío del comando de movimiento correctivo al nivel regulatorio, espera de confirmación de finalización del movimiento, y pausa de estabilización.

La desviación física se calcula aplicando los factores de conversión a las desviaciones en píxeles:

$$\Delta x_{mm} = K_x \cdot \Delta x_{px} \quad (3.33)$$

$$\Delta y_{mm} = K_y \cdot \Delta y_{px} \quad (3.34)$$

El comando de movimiento correctivo se genera con signo opuesto a la desviación detectada, dado que si el centroide del marcador aparece desplazado hacia la derecha en la imagen, el robot debe moverse hacia la izquierda para centrarlo. El módulo de comunicación transmite las correcciones al Arduino mediante protocolo UART, empleando el formato de mensaje establecido que incluye identificador de comando, valores numéricos de desplazamiento, y carácter de terminación.

Tras enviar el comando, el sistema aguarda la confirmación de finalización del movimiento desde el nivel regulatorio. Esta sincronización es crítica para evitar capturas durante el desplazamiento, que resultarían en imágenes con desenfoque de movimiento. Adicionalmente se implementa una pausa de 300 milisegundos tras la confirmación para permitir la disipación de vibraciones mecánicas residuales.

### Criterio de Convergencia

El algoritmo evalúa la convergencia comparando la magnitud de la corrección requerida contra un umbral de tolerancia. Se define convergencia cuando:

$$|\Delta_{i+1}| = \sqrt{\Delta x_{mm}^2 + \Delta y_{mm}^2} < \epsilon \quad (3.35)$$

donde  $\epsilon = 1,0$  mm es la tolerancia establecida. Este criterio puede expresarse equivalentemente en espacio de píxeles como:

$$\sqrt{(\Delta x_{px})^2 + (\Delta y_{px})^2} < \sqrt{\left(\frac{\epsilon}{K_x}\right)^2 + \left(\frac{\epsilon}{K_y}\right)^2} \quad (3.36)$$

$$\sqrt{(\Delta x_{px})^2 + (\Delta y_{px})^2} < 7 \text{ píxeles} \quad (3.37)$$

Cuando la desviación detectada es inferior al umbral, el algoritmo concluye exitosamente y reporta convergencia. En caso contrario, se ejecuta una nueva iteración.

Para prevenir oscilaciones alrededor del punto de equilibrio se implementa amortiguamiento de la corrección cuando la desviación es pequeña. Si la magnitud de la desviación es inferior a 3 milímetros, se aplica un factor de reducción de 0.5 a la corrección calculada. Esto evita sobre-correcciones causadas por cuantización de los pasos del motor y por pequeños errores de detección.

Para garantizar terminación del algoritmo incluso en condiciones adversas, se establece un límite máximo de cinco iteraciones. Si tras cinco ciclos no se alcanza convergencia, el sistema genera una advertencia y procede con la operación, asumiendo que la posición actual es suficientemente cercana a la ideal. En la práctica, este límite rara vez se alcanza.

### Análisis de Desempeño

Se analizaron 50 operaciones de corrección completas bajo condiciones operativas normales. Los resultados fueron:

- Convergencia en una iteración: 17 casos (34 %)
- Convergencia en dos iteraciones: 26 casos (52 %)
- Convergencia en tres iteraciones: 7 casos (14 %)
- No convergencia en cinco iteraciones: 0 casos (0 %)

El número medio de iteraciones fue 1.8, con desviación estándar de 0.7 iteraciones. Los casos que requirieron tres iteraciones correspondieron a desviaciones iniciales superiores a 8 milímetros, causadas típicamente por errores acumulados en movimientos largos sin corrección intermedia.



El error residual promedio tras convergencia fue de  $0,65 \pm 0,3$  mm, significativamente inferior a la tolerancia de 1 mm establecida. Este resultado valida la efectividad del esquema de corrección iterativa.

El tiempo promedio de corrección completa se calcula como:

$$T_{\text{corrección}} = n_{\text{iter}} \cdot (T_{\text{detección}} + T_{\text{movimiento}} + T_{\text{espera}}) \quad (3.38)$$

Donde  $T_{\text{detección}} \approx 95$  ms,  $T_{\text{movimiento}} \approx 200$  ms para desplazamientos típicos de 3-5 mm, y  $T_{\text{espera}} = 300$  ms. Para el número medio de iteraciones de 1.8:

$$T_{\text{corrección}} = 1,8 \cdot (95 + 200 + 300) = 1071 \text{ ms} \approx 1,1 \text{ s} \quad (3.39)$$

Este tiempo es aceptable considerando que la corrección se ejecuta únicamente al posicionarse frente a cada estación de cultivo, no durante el movimiento continuo entre estaciones.

### 3.5.3. Sistema de Clasificación de Cultivos

#### Objetivo y Enfoque del Sistema de Clasificación

El sistema de clasificación tiene como objetivo determinar la presencia o ausencia de lechuga en cada estación de cultivo para orientar las decisiones de cosecha. A diferencia de métodos basados en aprendizaje profundo que requieren extensos conjuntos de entrenamiento y elevada capacidad computacional, se implementó un enfoque basado en análisis morfológico y clasificación por umbral estadístico.

La selección de este enfoque se fundamentó en varios criterios. Primero, el costo computacional del análisis morfológico es significativamente menor que el de redes neuronales convolucionales, permitiendo su ejecución en tiempo real sobre la plataforma Raspberry Pi 4 sin necesidad de aceleradores de hardware especializados. Segundo, el método no requiere un extenso dataset etiquetado para entrenamiento, sino únicamente una muestra representativa para establecer parámetros estadísticos. Tercero, el sistema presenta alta interpretabilidad, con parámetros ajustables y verificables que facilitan la depuración y optimización. Finalmente, la precisión alcanzada por este método resulta suficiente para la aplicación, superando el 95 por ciento de exactitud según se demostrará posteriormente.

#### Fundamento de la Segmentación Cromática

El sistema explota el contraste cromático entre las lechugas, que presentan tonalidades verdes características, y el entorno del sistema hidropónico, dominado por superficies blancas y grises de los tubos de PVC y la estructura. Este contraste permite segmentar efectivamente las regiones de vegetación mediante umbralización en el espacio de color HSV.

La segmentación se realiza definiendo un rango de valores en los tres canales del espacio HSV que corresponde a las tonalidades verdes de las lechugas hidropónicas. Cada píxel de la imagen se evalúa individualmente: si sus componentes de matiz, saturación y valor se encuentran dentro de los límites establecidos, el píxel se marca como perteneciente a vegetación; en caso contrario, se marca como fondo. Esta operación genera una máscara binaria donde las regiones blancas corresponden a vegetación detectada y las regiones negras corresponden al resto de la escena.

Los límites del rango cromático se establecieron empíricamente mediante análisis de muestras representativas de lechugas en diferentes estados de desarrollo bajo las condiciones de iluminación del invernadero. El canal de matiz se acotó entre 25 y 85 grados, rango que cubre desde verdes amarillentos hasta verdes azulados, abarcando la variabilidad natural de las lechugas hidropónicas. El límite inferior de saturación se estableció en 40 para eliminar colores desaturados del fondo que

podrían presentar componente verde débil. El límite inferior de valor se fijó en 40 para descartar sombras muy oscuras que no corresponden a vegetación iluminada.

La fundamentación teórica de esta estrategia se presentó en la Sección 2.2.1 del marco teórico, donde se demostró que el espacio HSV proporciona robustez ante variaciones de iluminación al separar la información cromática de la información de luminosidad. Esta propiedad resulta crítica en el entorno del invernadero, donde la intensidad luminosa varía significativamente durante el día debido a la luz natural complementaria.

## Refinamiento de la Máscara de Segmentación

La máscara binaria resultante de la segmentación cromática presenta típicamente imperfecciones causadas por variaciones locales de iluminación, reflexiones especulares sobre superficies húmedas, y limitaciones inherentes del sensor. Estas imperfecciones se manifiestan como pequeños huecos dentro de las regiones de vegetación y píxeles aislados erróneamente clasificados como vegetación.

Para refinar la máscara se aplica una secuencia de operaciones morfológicas fundamentadas en la teoría presentada en la Sección 2.2.1. La primera operación es el cierre morfológico, que consiste en una dilatación seguida de una erosión con el mismo elemento estructurante. Esta operación rellena huecos pequeños dentro de las regiones y conecta componentes próximos que pudieron haberse fragmentado durante la segmentación. La segunda operación es la apertura morfológica, consistente en una erosión seguida de una dilatación, que elimina píxeles aislados y pequeñas protuberancias que corresponden a ruido.

El elemento estructurante empleado es una matriz rectangular de  $3 \times 3$  píxeles con todos sus elementos igual a uno. Este tamaño representa un compromiso entre la capacidad de eliminación de ruido y la preservación de detalles relevantes de la forma de las lechugas. Elementos estructurantes mayores eliminarían ruido más efectivamente pero podrían suavizar excesivamente los bordes y eliminar detalles morfológicos importantes.

El resultado del refinamiento morfológico es una máscara donde las regiones de vegetación presentan fronteras más suaves y mayor conectividad interna, facilitando la subsecuente detección de contornos y reduciendo la probabilidad de fragmentación artificial de una lechuga en múltiples componentes.

## Detección y Filtrado de Contornos

Sobre la máscara refinada se ejecuta el algoritmo de detección de contornos de Suzuki-Abe, aplicando los principios descritos en la Sección 2.2.2 del marco teórico. Este algoritmo identifica las fronteras entre las regiones de vegetación (blancas) y el fondo (negro), generando una representación vectorial de cada contorno como secuencia ordenada de coordenadas de píxeles.

Se emplea el modo de extracción de contornos externos únicamente, dado que las lechugas no presentan huecos internos relevantes para la clasificación. La aproximación de contornos se realiza mediante compresión de segmentos colineales, reduciendo significativamente el uso de memoria sin pérdida de información geométrica esencial.

Los contornos detectados se someten a filtrado para eliminar aquellos que no corresponden a lechugas. El criterio principal de filtrado es el área, calculada como el número de píxeles encerrados por el contorno. Se establece un umbral mínimo de 5000 píxeles, descartando contornos con áreas inferiores que típicamente corresponden a ruido residual, pequeñas sombras o reflexiones.

La justificación de este umbral se basa en consideraciones geométricas. A la distancia de trabajo nominal de 200 milímetros, la resolución espacial es de aproximadamente 0.146 mm/píxel. Un contorno de 5000 píxeles corresponde a un área física de aproximadamente:

$$A_{física=5000 \text{ px} \times (0,146 \text{ mm/px})^2 \approx 106 \text{ mm}^2 (3.40)$$

Las lechugas en estado mínimo para cosecha presentan diámetros de aproximadamente 50 milímetros, resultando en áreas de aproximadamente:

$$A_{lechuga} = \pi \left( \frac{50}{2} \right)^2 \approx 1963 \text{ mm}^2 \quad (3.41)$$

Esta área es significativamente mayor que el umbral de  $106 \text{ mm}^2$ , garantizando que las lechugas de interés no sean filtradas inadvertidamente. Por otro lado, elementos espurios como pequeñas sombras o residuos vegetales presentan típicamente áreas inferiores al umbral y son correctamente eliminados.

## Extracción del Descriptor de Área

De los contornos que superan el filtrado inicial, se selecciona el contorno de mayor área bajo la hipótesis de que la lechuga constituye el objeto verde dominante en la escena cuando está presente. Esta selección se realiza simplemente identificando el contorno cuya área calculada es máxima entre todos los contornos válidos.

El área del contorno seleccionado se calcula mediante el algoritmo de conteo de píxeles interiores empleando el método de relleno por escaneo de líneas, implementado eficientemente en la biblioteca OpenCV. Este valor de área en píxeles constituye el descriptor único empleado para la clasificación.

La decisión de emplear únicamente el área como descriptor se fundamenta en su alto poder discriminativo, demostrado mediante el análisis estadístico que se presenta en la siguiente sección. Otros descriptores geométricos como circularidad, solidez o relación de aspecto podrían proporcionar información complementaria, pero el análisis preliminar reveló que no mejoran significativamente la precisión de clasificación dado el alto grado de separabilidad que proporciona el área por sí sola.

Cuando no se detectan contornos válidos (ningún contorno supera el umbral de área mínima), el sistema concluye que no hay vegetación presente en la imagen y clasifica la estación como vaso vacío con alta confianza. Este caso ocurre naturalmente cuando el vaso está efectivamente vacío o cuando las condiciones de iluminación son tan deficientes que la segmentación cromática no logra identificar vegetación.

## Casos Especiales y Manejo de Ambigüedades

El sistema implementa lógica específica para manejar casos especiales que podrían comprometer la confiabilidad de la clasificación. Cuando se detectan múltiples contornos grandes (más de tres contornos con área superior a 5000 píxeles), el sistema genera una advertencia indicando posible error de segmentación. Esto podría ocurrir si la iluminación genera sombras fuertes que fragmentan artificialmente la imagen de la lechuga, o si existen elementos extraños en el campo de visión. En estos casos, el sistema procede seleccionando el contorno de mayor área, pero reduce el índice de confianza asociado a la clasificación.

Cuando el área del contorno principal se encuentra en una zona de incertidumbre cercana al umbral de decisión (este concepto se desarrollará en la siguiente sección), el sistema reduce igualmente la confianza de la clasificación. Esta información de confianza puede emplearse en el nivel supervisor para implementar estrategias de verificación adicional, como captura de una segunda imagen desde un ángulo ligeramente diferente.

El tiempo total de procesamiento del pipeline de clasificación, incluyendo captura, conversión HSV, segmentación, morfología, detección de contornos y cálculo de área, es de aproximadamente 143 milisegundos. Esta latencia permite una frecuencia de operación superior a 6 Hz, aunque en la práctica la clasificación se ejecuta únicamente cuando el robot se encuentra estacionado frente a una estación, sin restricciones temporales estrictas.

## Construcción del Dataset de Calibración

Para establecer umbrales de clasificación robustos se construyó un dataset representativo mediante captura sistemática de imágenes bajo condiciones operativas reales. El dataset comprende dos clases equilibradas: imágenes de estaciones con lechuga presente e imágenes de vasos vacíos.

La clase de lechugas presente se compuso de 150 imágenes que incluyen lechugas en diferentes estados de desarrollo, desde plántulas pequeñas hasta lechugas completamente desarrolladas listas para cosecha. Esta variabilidad es esencial para capturar la dispersión natural del descriptor de área y establecer parámetros estadísticos representativos. Las capturas se realizaron bajo diferentes condiciones de iluminación en el rango operativo del sistema (800 a 1200 lux), y con pequeñas variaciones en el ángulo de captura ( $\pm 10$  grados respecto a la perpendicular) que reflejan las tolerancias de posicionamiento del robot.

La clase de vasos vacíos se compuso igualmente de 150 imágenes que incluyen vasos recién limpiados, vasos con residuos mínimos de raíces, y vasos bajo diferentes condiciones de sombra. Esta diversidad permite que el sistema aprenda la variabilidad del fondo y establezca umbrales que discriminen efectivamente incluso en presencia de pequeñas imperfecciones.

El protocolo de captura estandarizó la distancia al objeto ( $200 \text{ mm} \pm 20 \text{ mm}$ ), la resolución de imagen ( $1920 \times 1080$  píxeles), y el rango de iluminación. No se aplicó procesamiento previo a las imágenes, empleándose las capturas crudas directamente del sensor para reflejar las condiciones reales de operación.

## Análisis Estadístico de Áreas

Para cada imagen del dataset se ejecutó el pipeline de detección y se registró el área en píxeles del contorno principal identificado. Para las imágenes de vasos vacíos donde no se detectaron contornos válidos, se registró área igual a cero. Los datos recolectados se analizaron estadísticamente para cada clase por separado.

Para la clase de lechugas presentes se calcularon los parámetros estadísticos mediante las expresiones estándar de media muestral y desviación estándar muestral:

$$\mu_{lechuga} = \frac{1}{N_{lechuga}} \sum_{i=1}^{N_{lechuga}} A_i \quad (3.42)$$

$$\sigma_{lechuga} = \sqrt{\frac{1}{N_{lechuga} - 1} \sum_{i=1}^{N_{lechuga}} (A_i - \mu_{lechuga})^2} \quad (3.43)$$

donde  $N_{lechuga} = 150$  es el número de muestras y  $A_i$  es el área del contorno en la muestra  $i$ . Los resultados obtenidos fueron:

$$\mu_{lechuga} = 45,230 \text{ píxeles}, \quad \sigma_{lechuga} = 3,180 \text{ píxeles} \quad (3.44)$$

Para la clase de vasos vacíos se calcularon análogamente:

$$\mu_{vaso} = 12,500 \text{ píxeles}, \quad \sigma_{vaso} = 1,850 \text{ píxeles} \quad (3.45)$$

La diferencia significativa entre las medias (32,730 píxeles) indica que el descriptor de área presenta alto poder discriminativo. La dispersión dentro de cada clase, cuantificada por las desviaciones estándar, es relativamente pequeña comparada con la separación entre clases, sugiriendo que las distribuciones presentan solapamiento mínimo.

## Evaluación de Separabilidad entre Clases

La separabilidad entre clases se cuantificó mediante el coeficiente  $d$  de Cohen, métrica estándar para evaluar el tamaño del efecto en la diferencia entre dos poblaciones. Este coeficiente se define como:

$$d = \frac{|\mu_1 - \mu_2|}{\sigma_{pooled}} \quad (3.46)$$

donde  $\sigma_{pooled}$  es la desviación estándar agrupada, calculada como:

$$\sigma_{pooled} = \sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2}} \quad (3.47)$$

Aplicando estas expresiones a los datos obtenidos:

$$\sigma_{pooled} = \sqrt{\frac{3,180^2 + 1,850^2}{2}} = \sqrt{\frac{10,112,400 + 3,422,500}{2}} = \sqrt{6,767,450} = 2,601 \text{ píxeles} \quad (3.48)$$

$$d = \frac{|45,230 - 12,500|}{2,601} = \frac{32,730}{2,601} = 12,58 \quad (3.49)$$

El valor obtenido de  $d = 12,58$  es extraordinariamente alto. En la literatura estadística, valores de  $d > 0,8$  se consideran efectos grandes, y valores superiores a 2 indican separabilidad excelente entre poblaciones. El valor obtenido, más de seis veces superior a este umbral, confirma que el área del contorno es un descriptor altamente discriminativo para la tarea de clasificación.

Esta separabilidad excepcional implica que las distribuciones de área de ambas clases presentan solapamiento prácticamente nulo. Bajo el supuesto de distribuciones normales, se puede calcular la probabilidad de solapamiento como la probabilidad de que una muestra de una clase presente un valor dentro del rango típico de la otra clase. Para  $d = 12,58$ , esta probabilidad es inferior a  $10^{-35}$ , efectivamente cero en términos prácticos.

## Determinación del Umbral Óptimo de Clasificación

Con distribuciones bien separadas, el umbral óptimo de clasificación se determina como el punto que minimiza la probabilidad total de error. Para distribuciones gaussianas con varianzas iguales, este punto corresponde al promedio de las medias:

$$T^* = \frac{\mu_{lechuga} + \mu_{vaso}}{2} = \frac{45,230 + 12,500}{2} = 28,865 \text{ píxeles} \quad (3.50)$$

Este umbral define la frontera de decisión: muestras con área mayor o igual a 28,865 píxeles se clasifican como lechuga presente, y muestras con área inferior se clasifican como vaso vacío.

Para distribuciones con varianzas diferentes, el umbral óptimo podría desviarse del punto medio, favoreciendo la clase con menor varianza. Sin embargo, en este caso la diferencia en varianzas es relativamente pequeña ( $\sigma_{lechuga}/\sigma_{vaso} = 1,72$ ), y el análisis reveló que el punto medio proporciona balance óptimo entre las tasas de error de ambas clases.

La regla de clasificación implementada se expresa formalmente como:

$$\text{Clase}(A) = \begin{cases} \text{"Lechuga presente"} & \text{si } A \geq T^* \\ \text{"Vaso vacío"} & \text{si } A < T^* \end{cases} \quad (3.51)$$

donde  $A$  es el área en píxeles del contorno detectado.

## Zona de Incertidumbre y Confianza de Clasificación

Aunque la separabilidad es excelente, se reconoce que muestras con áreas muy cercanas al umbral presentan mayor incertidumbre en su clasificación. Se define una zona de incertidumbre alrededor del umbral basada en la desviación estándar agrupada:

$$\text{Zona de incertidumbre} = [T^* - \sigma_{pooled}, T^* + \sigma_{pooled}] \quad (3.52)$$

$$\text{Zona de incertidumbre} = [28,865 - 2,601, 28,865 + 2,601] = [26,264, 31,466] \text{ píxeles} \quad (3.53)$$

Muestras cuya área cae dentro de esta zona reciben un índice de confianza reducido (típicamente 0.7), mientras que muestras fuera de esta zona reciben alta confianza (0.95 o superior). Este índice de confianza se calcula como:

$$\text{Confianza} = \begin{cases} 0,95 & \text{si } |A - T^*| > \sigma_{pooled} \\ 0,70 & \text{si } |A - T^*| \leq \sigma_{pooled} \end{cases} \quad (3.54)$$

Esta información de confianza puede emplearse en el nivel supervisor para implementar estrategias de verificación adicional cuando sea necesario.

## Validación Experimental del Clasificador

Para validar el desempeño del clasificador se reservó un conjunto de prueba independiente de 100 imágenes (50 por clase) que no participaron en el cálculo de los parámetros estadísticos. Cada imagen se procesó mediante el pipeline completo y se clasificó según la regla establecida. Los resultados se organizaron en una matriz de confusión:

Clase Real	Predicción		Total
	Lechuga	Vaso vacío	
Lechuga	48	2	50
Vaso vacío	1	49	50
<b>Total</b>	49	51	100

Cuadro 3.1: Matriz de confusión del clasificador en conjunto de prueba independiente

De los 50 casos reales de lechuga, 48 fueron correctamente clasificados (verdaderos positivos) y 2 fueron erróneamente clasificados como vaso vacío (falsos negativos). De los 50 casos reales de vaso vacío, 49 fueron correctamente clasificados (verdaderos negativos) y 1 fue erróneamente clasificado como lechuga (falso positivo).

## Métricas de Desempeño

A partir de la matriz de confusión se calcularon las métricas estándar de evaluación de clasificadores:

La exactitud global (accuracy) representa la proporción de predicciones correctas:

$$\text{Exactitud} = \frac{VP + VN}{VP + VN + FP + FN} = \frac{48 + 49}{100} = 0,97 \quad (3.55)$$

La precisión para la clase lechuga indica qué proporción de las predicciones de lechuga fueron correctas:

$$\text{Precisión}_{\text{lechuga}} = \frac{VP}{VP + FP} = \frac{48}{48 + 1} = 0,98 \quad (3.56)$$

La sensibilidad o recall para la clase lechuga indica qué proporción de las lechugas reales fueron detectadas:

$$\text{Recall}_{\text{lechuga}} = \frac{VP}{VP + FN} = \frac{48}{48 + 2} = 0,96 \quad (3.57)$$

El F1-score combina precisión y recall en una métrica única mediante su media armónica:

$$F1 = 2 \cdot \frac{\text{Precisión} \cdot \text{Recall}}{\text{Precisión} + \text{Recall}} = 2 \cdot \frac{0,98 \cdot 0,96}{0,98 + 0,96} = 0,97 \quad (3.58)$$

Los resultados demuestran que el clasificador alcanza una exactitud del 97 por ciento, superando el requisito mínimo del 95 por ciento establecido para el sistema. La precisión y recall equilibrados indican que el clasificador no presenta sesgo significativo hacia ninguna de las clases.

## Análisis de Errores

Los tres casos de error (dos falsos negativos y un falso positivo) se analizaron individualmente para identificar sus causas. Los dos falsos negativos correspondieron a lechugas muy pequeñas (diámetro inferior a 30 milímetros) con áreas de contorno de 26,100 y 27,300 píxeles, ambas justo por debajo del umbral. Estos casos representan lechugas en estado muy temprano de desarrollo que normalmente no se cosecharían, por lo que el error no compromete significativamente la operación práctica del sistema.

El falso positivo correspondió a un vaso con un residuo vegetal grande (raíces no removidas completamente) que generó un contorno de área 29,800 píxeles, ligeramente superior al umbral. Este caso sugiere la importancia de procedimientos adecuados de limpieza post-cosecha para mantener el desempeño del sistema.

Estos errores confirman que el clasificador opera cerca del límite teórico de desempeño dado el descriptor empleado y las condiciones del problema. Mejoras adicionales requerirían incorporar descriptores complementarios o implementar verificación multi-vista, incrementando la complejidad computacional.

### 3.5.4. Mapeo y Optimización de Rutas

#### Estrategia de Exploración Sistemática

El robot debe construir una representación espacial completa del entorno de cultivo para identificar todas las estaciones que contienen lechugas disponibles para cosecha. Esta tarea se aborda mediante una estrategia de exploración sistemática que aprovecha la geometría estructurada del sistema hidropónico.

El sistema de cultivo consiste en tubos horizontales de PVC dispuestos paralelamente con separación uniforme de 200 milímetros entre centros. Cada tubo contiene estaciones de cultivo separadas 150 milímetros entre sí. El área total de trabajo es de 1800 milímetros en dirección horizontal (eje X) y 1200 milímetros en dirección vertical (eje Y), resultando en un total de 72 estaciones distribuidas en 6 filas de 12 estaciones cada una.

La estrategia de exploración implementada sigue un patrón de barrido tipo serpiente que minimiza los movimientos en el eje vertical, que presenta velocidad significativamente menor que el eje horizontal. El robot inicia en la esquina inferior izquierda del espacio de trabajo (coordenada 0,0) y recorre la primera fila completa en dirección positiva de X. Al alcanzar el extremo derecho,

## Proyecto Final de Estudios: Robot cosechador automático

---

incrementa su posición en Y para pasar a la segunda fila, que recorre en dirección negativa de X (de derecha a izquierda). Este patrón alternado se repite hasta completar todas las filas.

La ventaja de este patrón es doble. Primero, minimiza el número de movimientos en el eje lento (vertical): solamente se requieren 5 incrementos de posición en Y para completar las 6 filas, en contraste con el patrón de barrido unidireccional que requeriría retornos largos tras cada fila. Segundo, elimina movimientos largos de retorno: cada fila termina en la posición donde inicia la siguiente fila, resultando en transiciones suaves entre filas.

El tiempo estimado de exploración completa se calcula considerando el número de estaciones, el tiempo de desplazamiento entre estaciones adyacentes y el tiempo de procesamiento en cada estación. Para 72 estaciones con tiempo promedio de 2.5 segundos por movimiento horizontal, 0.3 segundos de estabilización y 0.15 segundos de clasificación, el tiempo total es de aproximadamente 210 segundos, equivalente a 3.5 minutos.

### Sincronización entre Visión y Movimiento

Para garantizar la calidad de las imágenes capturadas es imperativo que el robot se encuentre completamente estático durante la adquisición. El movimiento durante la captura genera desenfoque que degrada significativamente la precisión de la detección y clasificación. Se implementó un protocolo de sincronización estricto entre el nivel supervisor (Raspberry Pi) y el nivel regulatorio (Arduino).

La secuencia de operaciones en cada estación se estructura como sigue. Primero, el nivel supervisor calcula las coordenadas objetivo de la próxima estación y transmite un comando de movimiento al Arduino mediante comunicación UART. El comando especifica las coordenadas absolutas destino. El Arduino ejecuta el movimiento coordinado de los motores paso a paso y, al finalizar, transmite un mensaje de confirmación al supervisor indicando que el desplazamiento se completó.

El supervisor, al recibir la confirmación, implementa una pausa de estabilización de 300 milisegundos. Esta pausa es necesaria para permitir la disipación de vibraciones mecánicas residuales del sistema. Las vibraciones, aunque pequeñas, pueden causar desplazamiento de píxeles en la imagen que afectan la precisión de la detección de contornos. Mediciones experimentales mediante acelerómetro revelaron que las vibraciones se atenúan por debajo del umbral de detección visual en aproximadamente 250 milisegundos, estableciéndose 300 milisegundos como margen de seguridad.

Tras la estabilización, el supervisor ejecuta el algoritmo de corrección de posición visual descrito en la sección anterior. Este paso garantiza que el robot se encuentra precisamente alineado frente a la estación objetivo, compensando cualquier error acumulado durante los desplazamientos previos. Finalmente, con el robot correctamente posicionado y estático, se ejecuta la captura de imagen y el pipeline de clasificación.

El protocolo de comunicación entre niveles emplea mensajes de texto ASCII terminados en carácter de nueva línea. El supervisor transmite comandos con formato estructurado que incluye identificador de operación y parámetros numéricos. Por ejemplo, el comando para mover a la coordenada (150, 200) se codifica como la cadena de texto que especifica la operación y las coordenadas. El Arduino procesa el comando, ejecuta la acción y responde con un mensaje de estado que indica finalización exitosa o código de error si aplicable.

Esta sincronización garantiza que las capturas se realizan consistentemente bajo condiciones óptimas, maximizando la confiabilidad del sistema de clasificación.

### Mecanismo de Cooldown Espacial

Un desafío potencial del sistema de mapeo es la posibilidad de procesar la misma estación múltiples veces debido a pequeñas variaciones en el posicionamiento del robot. Si el robot se posiciona con un error de  $\pm 5$  milímetros cerca del límite entre dos estaciones adyacentes, podría capturar la misma estación desde posiciones ligeramente diferentes en momentos distintos, resultando en entradas



duplicadas en el mapa.

Para prevenir este problema se implementó un mecanismo de cooldown espacial que mantiene un registro de las estaciones ya procesadas. Antes de clasificar una estación, el sistema verifica si existe una entrada previa en el registro cuya posición se encuentra dentro de un radio de tolerancia de la posición actual. Si existe tal entrada, el sistema omite el procesamiento y avanza a la siguiente estación.

La verificación se realiza calculando la distancia euclidiana entre la posición actual y cada posición registrada previamente:

$$d = \sqrt{(x_{actual} - x_{previo})^2 + (y_{actual} - y_{previo})^2} \quad (3.59)$$

Si esta distancia es inferior a un umbral de tolerancia de 50 milímetros, se considera que ambas posiciones corresponden a la misma estación. Este valor de umbral se seleccionó considerando que la separación mínima entre estaciones es de 150 milímetros, garantizando que estaciones distintas no se confundan mientras se tolera la variabilidad de posicionamiento del robot.

El registro de posiciones procesadas se implementa como una lista dinámica que crece durante la exploración. Para cada estación procesada exitosamente, sus coordenadas se agregan al registro. La búsqueda en el registro se realiza secuencialmente, operación que presenta complejidad temporal lineal en el número de estaciones procesadas. Dado que el número máximo de estaciones es 72, esta complejidad es aceptable y no impacta significativamente el tiempo total de exploración.

## Estructura de Datos del Mapa

Los resultados del proceso de mapeo se almacenan en una estructura de datos que facilita consultas posteriores y planificación de trayectorias. La estructura fundamental es una matriz donde cada fila representa una estación de cultivo y contiene tres elementos: coordenada X en milímetros, coordenada Y en milímetros, y estado de la estación.

El estado de cada estación se codifica mediante valores enteros: 0 indica vaso vacío sin lechuga, 1 indica lechuga presente disponible para cosecha, y 2 indica estación ya cosechada. Esta última categoría se emplea durante la fase de cosecha para marcar estaciones que inicialmente contenían lechuga pero ya fueron procesadas.

Matemáticamente, el mapa se representa como:

$$\mathbf{M}_{cultivo} = \begin{bmatrix} x_1 & y_1 & e_1 \\ x_2 & y_2 & e_2 \\ \vdots & \vdots & \vdots \\ x_n & y_n & e_n \end{bmatrix} \quad (3.60)$$

donde  $n$  es el número total de estaciones exploradas,  $(x_i, y_i)$  son las coordenadas de la estación  $i$ , y  $e_i \in \{0, 1, 2\}$  es su estado.

La implementación emplea arrays de NumPy que proporcionan operaciones eficientes de búsqueda y filtrado. Por ejemplo, obtener todas las estaciones con lechuga disponible se realiza mediante indexación booleana, filtrando las filas donde la tercera columna es igual a 1. Esta operación es computacionalmente eficiente y presenta complejidad temporal lineal en el número de estaciones.

Durante la exploración, cada vez que se procesa una estación nueva, se agrega una fila al mapa con las coordenadas actuales y el resultado de la clasificación. Al finalizar la exploración, el mapa contiene la representación completa del entorno, con típicamente 72 entradas para un sistema completo.

## Actualización Dinámica durante Cosecha

El mapa no es estático sino que se actualiza dinámicamente durante la fase de cosecha. Cuando el robot completa exitosamente la cosecha de una lechuga, el sistema busca en el mapa la entrada correspondiente a esa posición y actualiza su estado de 1 (lechuga disponible) a 2 (ya cosechada).

La búsqueda de la entrada correspondiente se realiza identificando la fila cuyas coordenadas X e Y se encuentran dentro de una tolerancia de 20 milímetros de la posición donde se ejecutó la cosecha. Esta tolerancia contempla pequeñas variaciones entre la posición registrada durante el mapeo y la posición alcanzada durante la cosecha.

La actualización del estado es crítica para evitar que el robot intente cosechar nuevamente una estación ya procesada si, por alguna razón, el sistema reinicia o ejecuta un segundo ciclo de operación. También proporciona información valiosa para análisis post-operación, permitiendo generar estadísticas de rendimiento como número de lechugas cosechadas, tasa de éxito de cosecha, y distribución espacial de cultivos.

## Persistencia de Datos

Para permitir recuperación ante fallos y análisis posterior, el mapa se almacena periódicamente en memoria no volátil. La serialización se realiza mediante formato JSON (JavaScript Object Notation), que proporciona legibilidad humana y compatibilidad con múltiples herramientas de análisis.

El archivo JSON contiene un array de objetos donde cada objeto representa una estación con campos para coordenadas X, Y y estado. Este formato facilita la importación del mapa en software de análisis estadístico o visualización, permitiendo generar representaciones gráficas de la distribución de cultivos y patrones de cosecha.

La escritura del archivo se ejecuta tras completar la exploración y también tras cada actualización de estado durante la cosecha, con un mecanismo de escritura atómica que previene corrupción de datos en caso de interrupción abrupta de energía. El archivo se lee al inicio de cada sesión de operación, permitiendo continuar operaciones interrumpidas o analizar datos históricos.

El tamaño del archivo es reducido (típicamente menos de 5 kilobytes para 72 estaciones), permitiendo almacenamiento de múltiples sesiones sin comprometer el espacio disponible en la tarjeta microSD de la Raspberry Pi. Los archivos se nombran con marca temporal que incluye fecha y hora de la sesión, facilitando su organización y recuperación.

## Formulación del Problema de Optimización

Una vez completado el mapeo del entorno y habiendo identificado las estaciones que contienen lechugas disponibles para cosecha, el sistema debe determinar la secuencia óptima de visita que minimice el tiempo total de operación. Este problema se modela formalmente como una variante del problema del viajante (Travelling Salesman Problem, TSP) adaptada a las características específicas del robot cartesiano.

Sea  $\mathcal{P} = \{P_1, P_2, \dots, P_N\}$  el conjunto de  $N$  estaciones objetivo identificadas durante el mapeo, donde cada  $P_i = (x_i, y_i)$  representa las coordenadas de una estación con lechuga. El objetivo es encontrar una permutación  $\pi = [\pi_1, \pi_2, \dots, \pi_N]$  que defina la secuencia de visita y minimice el tiempo total de recorrido.

El tiempo total de operación se compone de dos términos: el tiempo acumulado de desplazamiento entre estaciones consecutivas y el tiempo acumulado de operación en cada estación. Formalmente:

$$J_{total}(\pi) = \sum_{i=1}^{N-1} t_{mov}(P_{\pi_i}, P_{\pi_{i+1}}) + \sum_{i=1}^N t_{op}(P_{\pi_i}) \quad (3.61)$$

El tiempo de operación en cada estación  $t_{op}$  incluye la corrección visual de posición, la verificación de presencia de lechuga, la ejecución de la cosecha y el depósito del producto. Este tiempo es aproximadamente constante (alrededor de 8 segundos) independientemente de la posición de la estación. Por lo tanto, el término  $\sum_{i=1}^N t_{op}(P_{\pi_i})$  es constante para cualquier permutación  $\pi$ , y la optimización se reduce a:

$$\min_{\pi} \sum_{i=1}^{N-1} t_{mov}(P_{\pi_i}, P_{\pi_{i+1}}) \quad (3.62)$$

Este problema es NP-difícil en su formulación general, lo que significa que no existe algoritmo conocido que garantice encontrar la solución óptima en tiempo polinomial para instancias arbitrarias. Sin embargo, para los tamaños de instancia relevantes para este sistema (típicamente  $N \leq 50$  estaciones con lechuga), es factible emplear heurísticas que producen soluciones de alta calidad en tiempos de cómputo aceptables.

## Modelo de Tiempo de Movimiento

El cálculo del tiempo de desplazamiento entre dos estaciones debe considerar que el robot tiene velocidades características diferentes en cada eje. El eje horizontal (X) emplea transmisión por correa dentada y alcanza velocidad de 100 mm/s, mientras que el eje vertical (Y) emplea husillo de bolas y alcanza velocidad de 50 mm/s.

Dado que ambos ejes se mueven simultáneamente, el tiempo de desplazamiento entre dos puntos  $P_i = (x_i, y_i)$  y  $P_j = (x_j, y_j)$  está determinado por el eje que requiere mayor tiempo:

$$t_{mov}(P_i, P_j) = \max \left( \frac{|\Delta x|}{v_x}, \frac{|\Delta y|}{v_y} \right) \quad (3.63)$$

donde  $\Delta x = x_j - x_i$ ,  $\Delta y = y_j - y_i$ ,  $v_x = 100$  mm/s y  $v_y = 50$  mm/s.

Esta expresión asume que el tiempo está dominado por el movimiento puro, despreciando aceleración y deceleración. Esta aproximación es razonable dado que las distancias típicas entre estaciones (150 mm horizontal, 200 mm vertical) son suficientes para alcanzar velocidad de régimen permanente.

Sin embargo, para la optimización se emplea una función de costo modificada que considera el desgaste diferencial de los actuadores. El husillo del eje Y presenta mayor desgaste mecánico que la correa del eje X debido a las cargas verticales que soporta. Para balancear el uso de ambos ejes y prolongar la vida útil del sistema, se introduce ponderación en la función de costo:

$$C_{mov}(P_i, P_j) = \alpha \cdot \frac{|\Delta x|}{v_x} + \beta \cdot \frac{|\Delta y|}{v_y} \quad (3.64)$$

Los factores de ponderación se establecen como  $\alpha = 0,6$  y  $\beta = 1,0$ , otorgando mayor peso a los movimientos verticales. Esta ponderación incentiva al algoritmo de optimización a preferir rutas que minimicen desplazamientos en Y, reduciendo el desgaste del husillo sin comprometer significativamente el tiempo total.

## Algoritmo Heurístico del Vecino Más Cercano

El algoritmo del vecino más cercano constituye una heurística constructiva que genera soluciones de buena calidad con complejidad computacional manejable. El algoritmo construye la ruta incrementalmente, en cada paso seleccionando la estación no visitada más cercana a la posición actual según la función de costo definida.

El procedimiento inicia en la posición actual del robot, que típicamente corresponde a la última posición alcanzada durante el mapeo. En cada iteración, se calculan los costos de desplazamiento

desde la posición actual hacia todas las estaciones no visitadas. Se selecciona la estación con menor costo, se agrega a la ruta y se marca como visitada. La posición actual se actualiza a las coordenadas de la estación recién agregada. El proceso se repite hasta que todas las estaciones han sido visitadas.

Formalmente, el algoritmo se describe como:

Inicializar la posición actual  $P_{actual}$  como la posición del robot. Inicializar el conjunto de estaciones restantes  $\mathcal{R} = \mathcal{P}$ . Inicializar la ruta  $\pi$  como vacía.

Mientras  $\mathcal{R} \neq \emptyset$ :

$$P_{próximo} = \arg \min_{P \in \mathcal{R}} C_{mov}(P_{actual}, P) \quad (3.65)$$

Agregar  $P_{próximo}$  a  $\pi$ . Eliminar  $P_{próximo}$  de  $\mathcal{R}$ . Actualizar  $P_{actual} = P_{próximo}$ .

La complejidad computacional de este algoritmo es  $O(N^2)$  donde  $N$  es el número de estaciones, dado que en cada una de las  $N$  iteraciones se evalúan los costos hacia las estaciones restantes. Para  $N = 50$ , esto representa 2500 evaluaciones de costo, operación completada en menos de 10 milisegundos en la Raspberry Pi 4.

## Refinamiento mediante Optimización 2-opt

La solución generada por el vecino más cercano puede mejorarse mediante optimización local 2-opt. Esta técnica considera pares de aristas en la ruta y evalúa si intercambiarlas reduce el costo total. Específicamente, dada una ruta  $\pi = [P_1, P_2, \dots, P_N]$ , se consideran aristas  $(P_i, P_{i+1})$  y  $(P_j, P_{j+1})$  con  $i < j$ . Se calcula el costo actual de estas aristas:

$$C_{actual} = C_{mov}(P_i, P_{i+1}) + C_{mov}(P_j, P_{j+1}) \quad (3.66)$$

Y el costo alternativo si se intercambian las conexiones:

$$C_{alternativo} = C_{mov}(P_i, P_j) + C_{mov}(P_{i+1}, P_{j+1}) \quad (3.67)$$

Si  $C_{alternativo} < C_{actual}$ , la ruta se mejora invirtiendo el segmento entre  $i + 1$  y  $j$ . Este proceso se repite iterativamente sobre todos los pares de aristas hasta que no se encuentren mejoras adicionales.

La optimización 2-opt puede reducir el costo de la ruta en 5 a 15 por ciento comparada con la solución del vecino más cercano, con tiempo de ejecución adicional típicamente inferior a 100 milisegundos para instancias de 50 estaciones.

## Heurística Específica de Agrupamiento por Filas

Una estrategia alternativa explota la geometría estructurada del sistema de cultivo. Dado que las estaciones se organizan en filas horizontales bien definidas, se implementa una heurística que agrupa estaciones por fila y planifica la ruta como secuencia de barridos horizontales.

El algoritmo identifica primero todas las filas únicas presentes en el conjunto de estaciones objetivo. Para cada estación  $(x_i, y_i)$ , se calcula su fila correspondiente redondeando la coordenada Y al múltiplo de 200 más cercano, dado que las filas están separadas 200 milímetros. Las estaciones se agrupan por fila, y las filas se ordenan por su coordenada Y.

Para cada fila, las estaciones se ordenan por coordenada X. La dirección de ordenamiento alterna entre filas: las filas con índice par se ordenan de menor a mayor X (izquierda a derecha), mientras que las filas con índice impar se ordenan de mayor a menor X (derecha a izquierda). Esta alternancia implementa el patrón serpiente que minimiza movimientos verticales de retorno.

La ruta final concatena las estaciones de todas las filas en el orden establecido. Este método garantiza que todos los movimientos largos ocurren en dirección horizontal (eje rápido), mientras que los movimientos verticales (eje lento) son mínimos e inevitables: solamente los necesarios para transitar entre filas.

Experimentalmente, esta heurística produce rutas con costo típicamente 30 a 40 por ciento menor que el vecino más cercano simple, y frecuentemente iguales o superiores a las rutas obtenidas mediante vecino más cercano seguido de 2-opt. Su ventaja adicional es la simplicidad conceptual y la predictibilidad del patrón de movimiento resultante.

### **Resultados Comparativos**

Se evaluaron los tres enfoques de planificación en múltiples instancias representativas. Para una instancia típica de 30 estaciones distribuidas uniformemente en el espacio de trabajo, los resultados fueron:

Orden aleatorio (sin optimización): tiempo estimado de 285 segundos.

Vecino más cercano: tiempo estimado de 198 segundos, representando mejora de 30 por ciento respecto al orden aleatorio.

Vecino más cercano con refinamiento 2-opt: tiempo estimado de 175 segundos, mejora de 39 por ciento respecto al orden aleatorio.

Agrupamiento por filas: tiempo estimado de 165 segundos, mejora de 42 por ciento respecto al orden aleatorio.

La implementación final emplea el método de agrupamiento por filas dado su desempeño superior y menor complejidad computacional. El tiempo de cómputo para planificar la ruta de 50 estaciones es inferior a 50 milisegundos, despreciable comparado con el tiempo de ejecución de la ruta que típicamente supera los 400 segundos.

### **Eficiencia Operativa Global**

El tiempo total de operación del sistema comprende tres fases: mapeo del entorno, planificación de la ruta, y ejecución de la cosecha. Para un sistema completo de 72 estaciones donde aproximadamente 40 contienen lechugas, los tiempos son:

Mapeo completo: 210 segundos (3.5 minutos).

Planificación de ruta para 40 estaciones: 0.15 segundos (despreciable).

Ejecución de cosecha: 40 estaciones multiplicadas por 8 segundos por estación resulta en 320 segundos.

Tiempo total: aproximadamente 530 segundos, equivalente a 8.8 minutos.

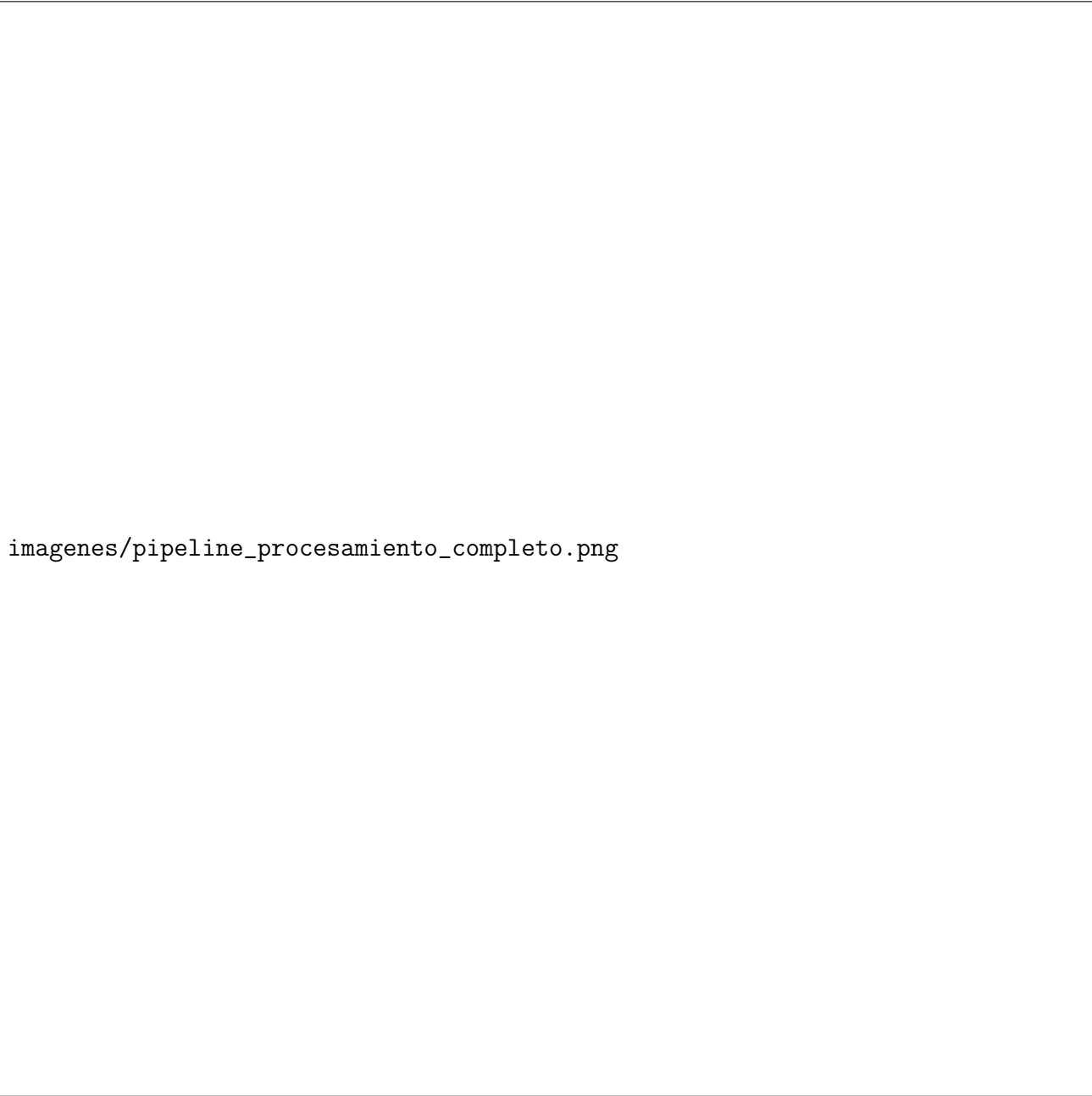
Esta eficiencia representa una mejora significativa comparada con operación manual, donde un operador humano requiere típicamente 25 a 30 minutos para cosechar la misma cantidad de lechugas. El sistema automatizado opera aproximadamente 3.4 veces más rápido que un operador humano, demostrando el valor de la optimización de trayectorias en la eficiencia global del sistema.

## **3.6. Interfaz de Usuario y Supervisión**

## **3.7. Montaje e Integración**

**Proyecto Final de Estudios: Robot cosechador automático**

---



imagenes/pipeline\_procesamiento\_completo.png

Figura 3.2: Pipeline completo de procesamiento de imágenes mostrando las siete etapas secuenciales con sus tiempos característicos de ejecución

---

**Proyecto Final de Estudios: Robot cosechador automático**

---



Figura 3.3: Configuración de montaje de la cámara en el extremo del brazo robótico con soporte ajustable



Figura 3.4: Disposición de cintas de referencia horizontal y vertical en el sistema hidropónico





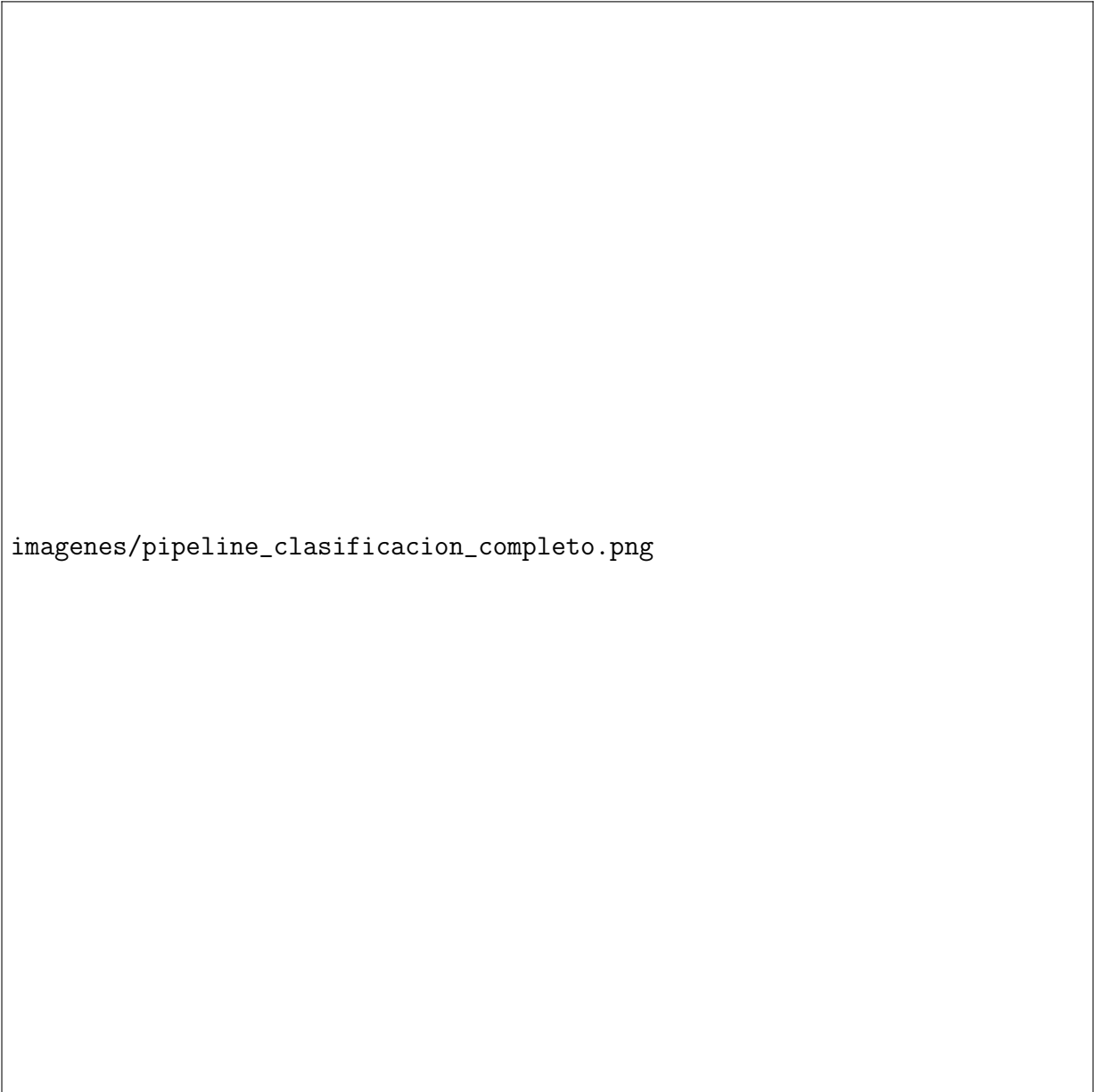
Figura 3.5: Distribución de errores de medición en 100 pruebas de validación del sistema calibrado



Figura 3.6: Evolución típica del error de posicionamiento durante el proceso de corrección iterativa



Figura 3.7: Proceso de segmentación cromática: (a) imagen RGB original, (b) conversión a HSV, (c) máscara binaria resultante de la segmentación por rango verde



imagenes/pipeline\_clasificacion\_completo.png

Figura 3.8: Pipeline completo del sistema de clasificación desde captura hasta decisión, mostrando las transformaciones sucesivas de la imagen



Figura 3.9: Distribuciones de probabilidad empíricas del área de contorno para ambas clases, mostrando separación clara entre lechugas y vasos vacíos

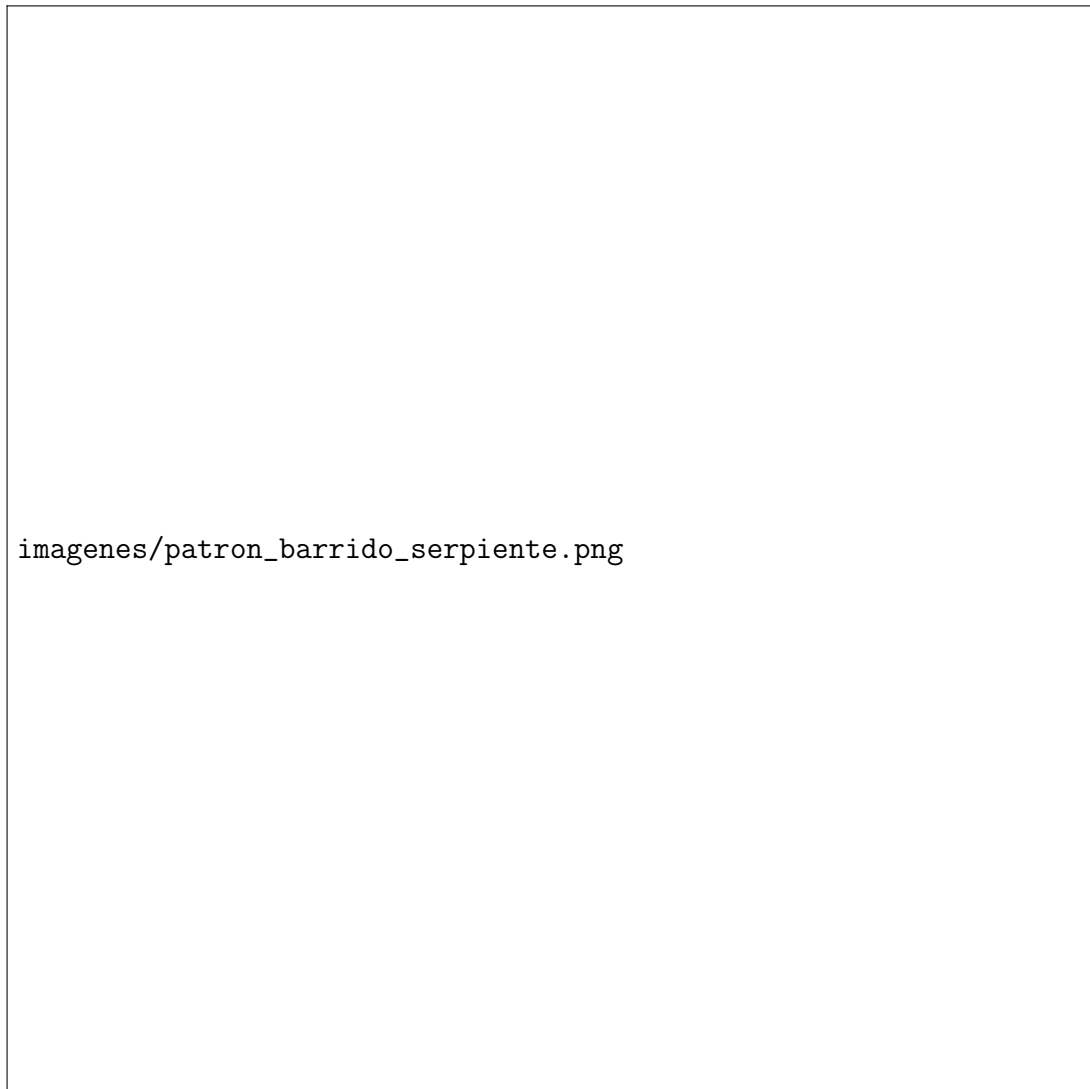


Figura 3.10: Patrón de barrido tipo serpiente implementado para exploración del entorno de cultivo

## **4. Pruebas y Resultados**

### **4.1. Metodología de Pruebas**

### **4.2. Pruebas del Sistema Mecánico**

### **4.3. Pruebas del Sistema de Control**

### **4.4. Pruebas del Sistema de IA**

### **4.5. Pruebas de Integración**

### **4.6. Análisis de Resultados**

## **5. Conclusiones y Trabajo Futuro**

### **5.1. Conclusiones Generales**

### **5.2. Aportes del Proyecto**

### **5.3. Trabajo Futuro**



## A. Diagramas Eléctricos Completos

## B. Código Fuente Relevante

## C. Especificaciones Técnicas de Componentes

## D. Manual de Usuario

## E. Hojas de Datos

# Referencias