

# Proyecto Final: Análisis y segmentación de mastocitos en imágenes de patología

Serapio Hernández Alexis Arturo  
*Facultad de Ingeniería  
UNAM*  
Ciudad de México, México  
[alexis.serapio@ingenieria.unam.mx](mailto:alexis.serapio@ingenieria.unam.mx)

García López Erik  
*Facultad de Ingeniería  
UNAM*  
Ciudad de México, México  
[erikpumas999@gmail.com](mailto:erikpumas999@gmail.com)

Reyes Herrera Rogelio  
*Facultad de Ingeniería  
UNAM*  
Ciudad de México, México  
[masterfive5of@gmail.com](mailto:masterfive5of@gmail.com)

**Abstract**—This project focuses on developing an automated system for detecting and classifying mast cells in histological images to aid veterinary diagnostics. Using supervised learning with a linear SVM and features derived from GLCM and geometric properties, the model achieved a 94% accuracy. Despite a limited dataset, the approach demonstrates promising results, combining manual annotation, feature extraction, and image segmentation techniques to differentiate mast cells from healthy cells.

**Keywords**—*Mast cells, QuPath, classification , SVM, GLCM, Canny, veterinary pathology*

## OBJETIVOS

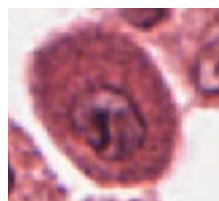
- Aplicar conocimientos adquiridos a lo largo del curso en un proyecto final.
- Desarrollar un sistema automatizado para la detección y clasificación de mastocitos y células sanas
- Contribuir a la medicina veterinaria para mejorar la precisión y diagnóstico en la identificación de mastocitos en perros

## I. INTRODUCCIÓN

El diagnóstico y tratamiento oportuno de enfermedades en animales domésticos representa un reto importante en la medicina veterinaria moderna.

Entre estas enfermedades, los mastocitomas son un tipo común de tumor cutáneo en perros, destacando debido a su potencial agresividad y alta incidencia

El **mastocito** es una célula del sistema inmunológico, reconocible por su núcleo y el citoplasma granular que lo rodea.



Por otra parte, las células que **no son mastocitos** son más pequeñas y carecen de las características granulares visibles en su citoplasma.



La identificación precisa de estas afecciones suele depender de un análisis histopatológico realizado por especialistas, lo que suele ser un proceso laborioso y propenso a variaciones humanas.

En este contexto, el uso de técnicas de visión computacional y el reconocimiento de patrones puede fungir como una herramienta prometedora para automatizar y mejorar la precisión del diagnóstico.

Por lo que este proyecto, busca aplicar métodos de clasificación en imágenes microscópicas de tejidos que permitan identificar y clasificar mastocitos saludables y aquellos que presentan anomalías relacionadas con mastocitomas.

Esto nos permite extraer información cuantitativa sobre las propiedades visuales de las células, como patrones de distribución, contraste, homogeneidad y rugosidad para detectar las regiones de mastocitos y determinar la situación en la que se encontraría un animalito

## II. DESARROLLO

Para dar solución al problema planteado en el proyecto primero había que pensar en cuáles eran las opciones más óptimas.

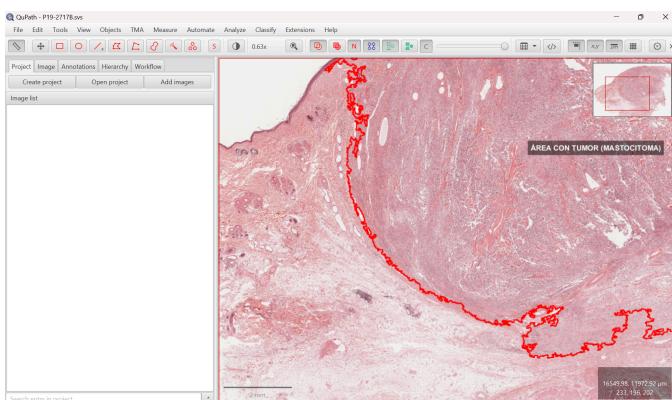
Al principio se desconocía la naturaleza de los datos así que pensamos en un K-means con 2 clusters, pero al ser células tan

parecidas hasta cierto punto clasificaba 50-50 como si las clases estuvieran perfectamente balanceadas; por lo tanto para recabar la información restante consultamos directamente con el Dr. Jacobo.

Una vez teniendo los conocimientos necesarios para distinguir en mayor parte los mastocitos de las células sanas, pensamos en una red neuronal convolucional, pero por falta de tiempo mejor aplicamos un algoritmo tradicional.

Por lo tanto a lo largo de este documento se detallará cómo se recolectó la información, como se ingresó al modelo, como se probó y los resultados:

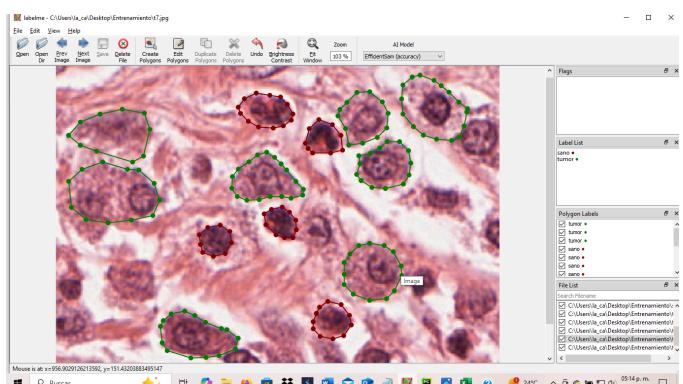
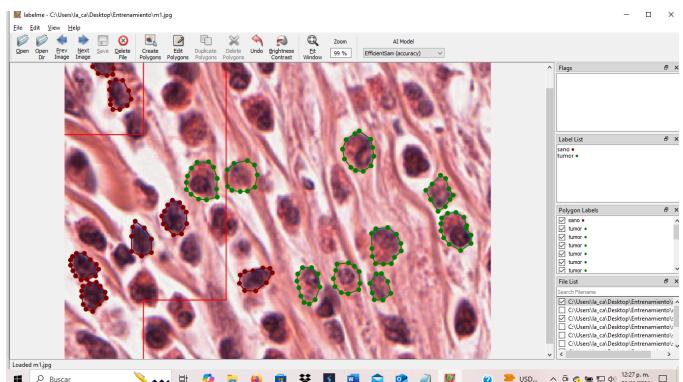
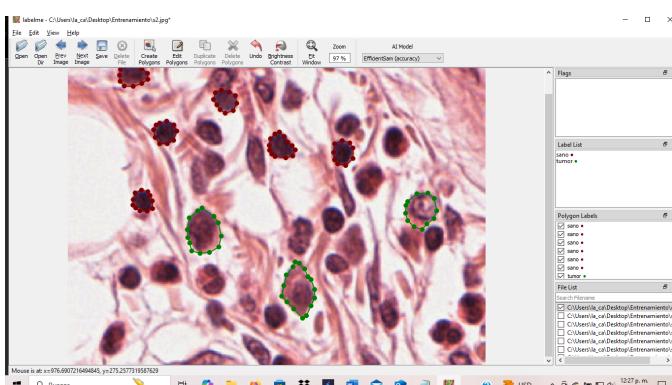
Primero obtuvimos 20 capturas de QuPath a la misma resolución con imágenes de ambas zonas (región sana y región tumoral).



Utilizamos la imagen P19-2717B.svs y para tener las imágenes en buena resolución tomamos las capturas de pantalla con ayuda de File > Export snapshot > Current viewer content (SVG), para después en una página de internet convertirlas a JPG.

Una vez teniendo las imágenes, hicimos un etiquetado manual célula por célula en todas las imágenes, utilizando la biblioteca labelme.

En las células dudosas no pusimos etiqueta, así como tampoco a la totalidad de células porque era un proceso tardado.



Labelme nos devuelve un archivo json con la máscara que se le aplicará a la imagen para solo tomar en cuenta esa región con su respectiva etiqueta.

```

45 def load_data(data_dir):
46     for file_name in os.listdir(data_dir):
47         if file_name.endswith(".json"):
48             with open(os.path.join(data_dir, file_name)) as f:
49                 data = json.load(f)
50                 image_path = os.path.join(data_dir, data["imagePath"])
51                 image = Image.open(image_path)
52
53
54                 for shape in data["shapes"]:
55                     label_name = shape["label"]
56                     points = shape["points"]
57
58                     mask = np.zeros((image.height, image.width), dtype=bool)
59                     poly = np.array(points, dtype=np.int32)
60                     rr, cc = polygon(poly[:, 1], poly[:, 0], mask.shape)
61                     mask[rr, cc] = True

```

En esta imagen podemos ver como se cargan los json con el módulo os y se relacionan con la imagen por medio del parámetro imagePath que viene dentro del archivo.

En base a los puntos que contiene el json, se crea una máscara del tamaño de la imagen original y va seleccionando únicamente los polígonos definidos.

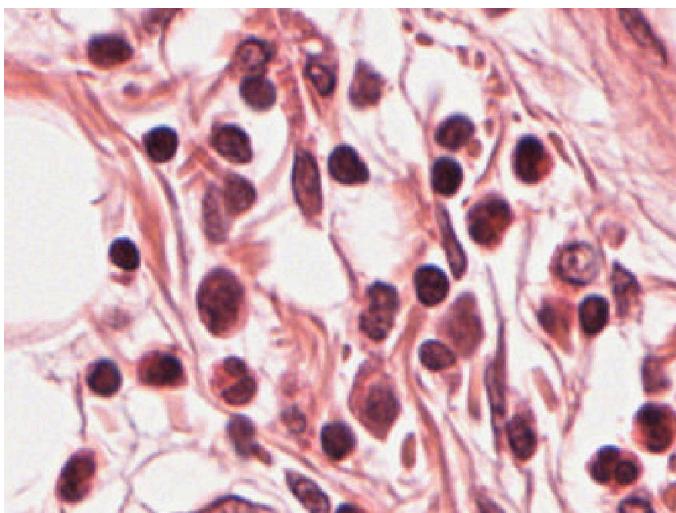
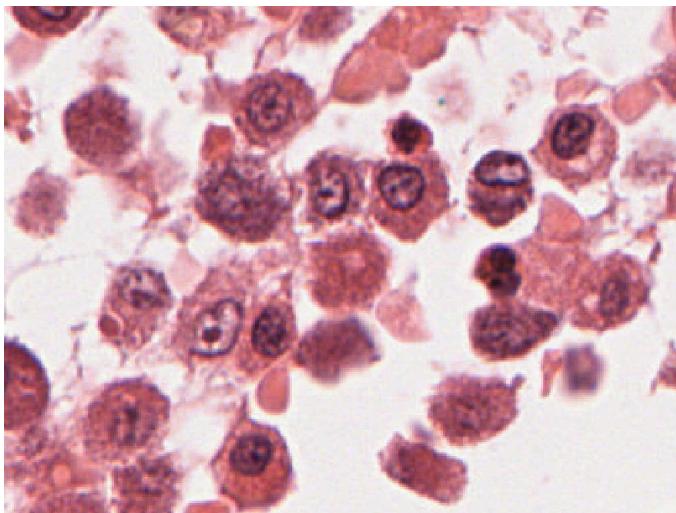
Se manda a llamar esta función para cargar los datos y dividirlos en dos listas, una para entrenamiento y una para prueba.

```
70 train_dir = "Entrenamiento"  
71 test_dir = "Prueba"  
72  
73 X_train, y_train = load_data(train_dir)  
74 X_test, y_test = load_data(test_dir)
```

El número de imágenes en cada directorio no es el factor determinante, si no el número de polígonos que seleccionamos en la etapa previa, que corresponden al número de células.

Con un n mero de c lulas del 80% en entrenamiento y 20% en prueba continuamos con el entrenamiento. En este caso usamos un SVM lineal, ya que siempre nos da buenos resultados en esta asignatura.

Ejemplo de imágenes de entrenamiento:



```
[{"version": "5.5.0",  
 "flags": {},  
 "shapes": [  
   {  
     "label": "sano",  
     "points": [  
       [  
         556.0,  
         218.0  
       ],  
       [  
         558.0,  
         200.0  
       ],  
       [  
         574.0,  
         193.3333333333334  
       ],  
       [  
         590.0,  
         198.0  
       ],  
       [  
         596.6666666666666
```

Para alimentar al modelo necesitamos extraer un grupo determinante de características; para el proyecto empezamos usando solo el contraste y la correlación. Esto nos dio métricas muy bajas, del 55%.

Así que hicimos más robusta la información aumentando primero el set de datos etiquetando más células, y después aumentando el número de características usando todas las disponibles de GLCM.

```
18 def extract_features(image, mask):
19
20     region_area = mask.sum()
21
22     glcm = graycomatrix(masked_image, distances=[1, 2, 3, 4], angles=[0, np.pi / 4, np.pi / 6, np.pi / 8], levels=256, symmetric=True, normed=True)
23     contrast = graycoprops(glcm, prop='contrast').mean()
24     energy = graycoprops(glcm, prop='energy').mean()
25     correlation = graycoprops(glcm, prop='correlation').mean()
26     dissimilarity = graycoprops(glcm, prop='dissimilarity').mean()
27     asm = graycoprops(glcm, prop='ASM').mean()
28
29     mean_intensity = masked_image[mask].mean()
30     std_intensity = masked_image[mask].std()
31
32     labeled_mask = label(mask)
33     props = regionprops(labeled_mask)
```

Además usamos características no definidas en GLCM como el área de la célula, justificándonos en que los mastocitos son más grandes que las células normales, así como la media y desviación estándar ya que creemos que con esto podemos medir un poco la granularidad que comenta el Dr; las células malas son menos “uniformes” que las tumorales que se pueden ver con texturizado más prominente.

También el contraste es determinante, ya que las células sanas tienen una “intensidad” de negros mayor y las otras se perciben medio borrosas.

Fue así como hicimos la elección de características que nos llevó a obtener métricas del 94%.

Entrenamos ahora si el modelo para obtener las métricas mencionadas:

```
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)

y_pred = svm.predict(X_test)
print(classification_report(y_test, y_pred))

precision_global = accuracy_score(y_test, y_pred)
print(f"Precisión global: {precision_global:.2f}")
```

	precision	recall	f1-score	support
sano	0.89	1.00	0.94	17
tumor	1.00	0.88	0.93	16
accuracy			0.94	33
macro avg	0.95	0.94	0.94	33
weighted avg	0.95	0.94	0.94	33
Precisión global:	0.94			

Por último quisimos implementar una técnica de detección para probar con una imagen que el algoritmo no conoce y ver cómo se desempeña.

Para esta validación, ni siquiera tendremos etiquetas, pretendemos que el programa detecte bordes y encierre las células.

En primera instancia intentamos seleccionar el contorno de todas las células, pero siempre había inconsistencias por lo que nos decidimos por detectar células y meterlas en una caja.

Una vez en la caja, se extraerán las características de las células detectadas y se meterán al algoritmo entrenado para ver cómo clasifica.

Con la clasificación lista se despliega una salida la clasificación. Tiene algunos fallos visibles pero los resultados son prometedores.

Rojo para mastocitos y azul para célula sana.

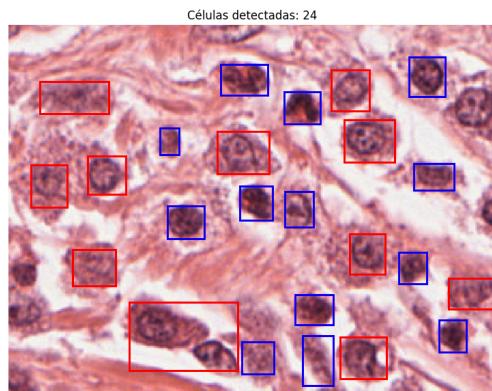
```
98 cleaned_mask = morphology.remove_small_objects(binary_mask, min_size=900)
99 cleaned_mask = remove_small_holes(cleaned_mask, area_threshold=500)
100 cleaned_mask = clear_border(cleaned_mask)
101
102
103 edges = canny(cleaned_mask.astype(float), sigma=2)
104
105
106 labeled_regions = measure.label(cleaned_mask)
107 properties = measure.regionprops(labeled_regions)
108
109
110 valid_regions = []
111 for prop in properties:
112     area = prop.area
113     eccentricity = prop.eccentricity
```

Se hace un filtrado de los objetos detectados en el rango de 1000 a 12000 de área; para que no detecté cosas minúsculas que pueden corresponder únicamente a ruido.

En un principio también filtramos por excentricidad pero resultó casi sin impacto.

```
115 if 1000 < area < 12000:
116     valid_regions.append(prop)
117
118
119 fig, ax = plt.subplots(figsize=(10, 10))
120 ax.imshow(image)
121
122
123 for region in valid_regions:
124     minr, minc, maxr, maxc = region.bbox
125     ax.add_patch(plt.Rectangle((minc, minr), maxc - minc, maxr - minr,
126                               edgecolor='red', facecolor='none', linewidth=2))
127
128 # Extraer características y predecir clase
129 mask = labeled_regions == region.label
130 features = extract_features(image, mask)
131 predicted_class = svm.predict([features])[0]
```

Figure 1



### Analizando los resultados:

Por la parte de las métricas tenemos un buen desempeño aun con un dataset pequeño, como de 200 células aproximadamente y 33 para prueba (94% accuracy), se pueden afinar detalles desde el etiquetado manual, ya que al no ser expertos pudimos cometer varios errores.

Por otra parte, podemos decir que la detección es una tarea difícil y puede ser estudiada más a profundidad para detectar automáticamente las células en una imagen y pasarlas por el modelo entrenado; de esto dependen en gran medida los resultados que estamos obteniendo.

Por ejemplo, vemos que detecta el núcleo de un mastocito como una célula sana y es justamente porque en efecto parece una célula sana si no se toma en cuenta el contexto exterior de la caja detectada.

### III. CÓDIGO

En este apéndice, se anexará el contenido del código para el proyecto.

#### *Código:*

```
import os
import json
from PIL import Image, ImageFilter
from skimage import io, color, filters, measure,
morphology
import numpy as np
from skimage.draw import polygon
from skimage.feature import graycomatrix, graycoprops
from skimage.measure import regionprops, label
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,
classification_report
import matplotlib.pyplot as plt
from skimage.feature import canny
from skimage.filters import threshold_otsu
from skimage.segmentation import clear_border
from skimage.morphology import remove_small_holes

# Función para extraer características de una región
etiquetada
def extract_features(image, mask):
    gray_image = np.array(image.convert("L"))
```

```
masked_image = np.where(mask, gray_image, 0)

region_area = mask.sum()

glcm = graycomatrix(masked_image, distances=[1, 2,
3, 4], angles=[0, np.pi / 4, np.pi / 6, np.pi / 3,
np.pi / 2, 2 * np.pi / 3, 5 * np.pi / 6, 3 * np.pi / 4],
levels=256, symmetric=True,
normed=True)
contrast = graycoprops(glcm, 'contrast').mean()
energy = graycoprops(glcm, 'energy').mean()
correlation = graycoprops(glcm, 'correlation').mean()
dissimilarity = graycoprops(glcm, 'dissimilarity').mean()
asm = graycoprops(glcm, 'ASM').mean()

mean_intensity = masked_image[mask].mean()
std_intensity = masked_image[mask].std()

labeled_mask = label(mask)
props = regionprops(labeled_mask)
eccentricity = props[0].eccentricity if props else 0
solidity = props[0].solidity if props else 0

features = [region_area, contrast, energy,
correlation, mean_intensity,
std_intensity, eccentricity,
solidity, dissimilarity, asm]
return features

def load_data(data_dir):
    X, y = [], []
    for file_name in os.listdir(data_dir):
        if file_name.endswith(".json"):
            with open(os.path.join(data_dir,
file_name)) as f:
                data = json.load(f)
                image_path = os.path.join(data_dir,
data["imagePath"])
                image = Image.open(image_path)

                for shape in data["shapes"]:
                    label_name = shape["label"]
                    points = shape["points"]

                    mask = np.zeros((image.height,
image.width), dtype=bool)
                    poly = np.array(points, dtype=np.int32)
                    rr, cc = polygon(poly[:, 1], poly[:, 0], mask.shape)
                    mask[rr, cc] = True

                    features = extract_features(image,
mask)
```

```

        X.append(features)
        y.append(label_name)

    return X, y

train_dir = "Entrenamiento"
test_dir = "Prueba"

X_train, y_train = load_data(train_dir)
X_test, y_test = load_data(test_dir)

svm = SVC(kernel='linear')
svm.fit(X_train, y_train)

y_pred = svm.predict(X_test)
print(classification_report(y_test, y_pred))

precision_global = accuracy_score(y_test, y_pred)
print(f"Precisión global: {precision_global:.2f}")

image_path = "t7.jpg"
image = Image.open(image_path)

gray_image = np.array(image.convert("L"))

smoothed_image = filters.gaussian(gray_image, sigma=4)

threshold_value = .5
binary_mask = smoothed_image < threshold_value

cleaned_mask = morphology.remove_small_objects(binary_mask,
min_size=900)
cleaned_mask = remove_small_holes(cleaned_mask,
area_threshold=500)
cleaned_mask = clear_border(cleaned_mask)

edges = canny(cleaned_mask.astype(float), sigma=2)

labeled_regions = measure.label(cleaned_mask)
properties = measure.regionprops(labeled_regions)

valid_regions = []
for prop in properties:
    area = prop.area
    eccentricity = prop.eccentricity

    if 1000 < area < 12000:
        valid_regions.append(prop)

```

```

fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(image)

for region in valid_regions:
    minr, minc, maxr, maxc = region.bbox
    ax.add_patch(plt.Rectangle((minc, minr), maxc - minc, maxr - minr,
                               edgecolor='red',
                               facecolor='none', linewidth=2))

    # Extraer características y predecir clase
    mask = labeled_regions == region.label
    features = extract_features(image, mask)
    predicted_class = svm.predict([features])[0]
    print(predicted_class)

    # Asignar color según clase
    if predicted_class == "tumor":
        color = 'red'
    elif predicted_class == "sano":
        color = 'blue'
    else:
        color = 'green'

    ax.add_patch(plt.Rectangle((minc, minr), maxc - minc, maxr - minr,
                               edgecolor=color,
                               facecolor='none', linewidth=2))

ax.set_title(f"Células detectadas: {len(valid_regions)}")
ax.axis("off")
plt.show()

```

## REFERENCES

- [1] Histology Guide - virtual microscopy laboratory. (s. f.). <https://histologyguide.com/>
- [2] Pombal, M. M. P. M. M. Á. (s. f.). Tipos celulares. Mastocito. Atlas de Histología Vegetal y Animal. <https://mmeigas.webs.uvigo.es/8-tipos-celulares/mastocito.php>
- [3] Support Vector Machine (SVM). (s. f.). MATLAB & Simulink. <https://la.mathworks.com/discovery/support-vector-machine.html>
- [4] Wkentaro. (s. f.). GitHub - wkentaro/labelme: Image Polygonal Annotation with Python (polygon, rectangle, circle, line, point and image-level flag annotation). GitHub. <https://github.com/wkentaro/labelme>