

TAREA 3 - CONSIDERACIONES COGNITIVAS

Carlos E. Jara Vásquez, Christopher J. Morales Acosta
 carlosjaravas@estudiantec.cr christophermorales@estudiantec.cr
 Área académica de Ingeniería Mecatrónica
 Tecnológico de Costa Rica

Resumen

La Tarea 3 consiste en el desarrollo de una aplicación en python capaz de determinar la puntuación total e individual de 3 dados iguales mediante una imagen. Para la implementación de la aplicación se utilizan principalmente las herramientas de *KMeans* y la transformada de Hough, para realizar la segmentación por color y ubicación de los dados, y para identificar las líneas y puntos de cada dado. En este documento se presenta la descripción tanto del problema como de la solución, se muestran y analizan los principales resultados. Se utilizan 3 imágenes con diferentes configuraciones de los dados, donde varían sus posiciones y la puntuación en los dados. Finalmente se logra determinar la puntuación de cada dado individualmente y la puntuación total de forma precisa y exitosa, y presentar los resultados de cada imagen en un archivo de texto. Se demuestra que la transformada de Hough y *KMeans* son herramientas muy flexibles, autónomas y con múltiples aplicaciones.

Palabras clave

Lectura, K-means, Hough Transform, Dados

I. DESCRIPCIÓN DEL PROBLEMA

Una de las aplicaciones principales de los sistemas de visión es la lectura, o sea, extraer información codificada en objetos, mediante captura y procesado [1]. En este problema se tienen datos: objetos cúbicos de cierto color donde cada una de las caras tiene ciertas depresiones, orificios o puntos circulares pintados de un color distinto al color del dado, donde la información codificada es la cantidad de agujeros que tenga la cara superior del dado. En esta aplicación se debe diseñar e implementar un sistema de visión que realice la captura de 3 dados iguales tirados (exponiendo la información codificada) sobre un fondo uniforme y debe analizar la imagen y devolver los valores de cada uno de los dados (extraer la información) [2].



Figura 1: Dado convencional: codifica el número 6. [3]

Además de estos requerimientos, se tiene una restricción: la solución debe hacer fundamentalmente uso de los paradigmas de K-means y/o Transformada de Hough. En otras palabras, la solución utiliza como núcleo de la solución los paradigmas anteriores, sin embargo, es permitido el uso de funciones y operaciones de preprocesado que mejoran o habilitan el funcionamiento de estas funciones [2].

II. DESCRIPCIÓN DE LA SOLUCIÓN

II-A. Captura de Imagen

El primer paso es configurar un ambiente de captura adecuado de forma que se definan los límites y condiciones en las que trabajará el sistema de visión. Para empezar, se dispone de una cámara por medio de un teléfono móvil Samsung A32. Este se configuró de forma que tome imágenes con relación de aspecto de 3:4, además se trabajó en modo PRO (equivalente al modo manual), donde se modificó el parámetro ISO, para obtener un mayor contraste y desactivar los filtros automáticos y autoajustes (de aumento o enfoque).

La escena o espacio de trabajo se diseñó con el principal objetivo de: evitar reflejos, adquirir el mayor contraste en la imagen y tener imágenes con características óptimas (como se menciona adelante). El primer aspecto por destacar es que se utilizó un material textil negro como fondo, pues este genera alto contraste con el dado y provoca muy poca luz especular con respecto a otros materiales probados. El montaje de la escena se muestra a continuación en la Fig. 2:



Figura 2: Montaje de la escena.

De esta escena se pueden destacar características clave. La primera, encerrada con un círculo verde, es el espacio de trabajo, donde se encuentran los dados y el fondo textil negro. También se pueden ver que existen dos fuentes de iluminación: se trata de dos lámparas de bombillo blanco puestas a los lados como se aprecia en la Fig. 2 en círculos rojos, esto con el objetivo de que no se generen sombras sobre el fondo, pues las sombras aumentan el contraste local cerca de los dados y puede generar falsos bordes, por lo tanto las sombras son un objeto no deseado para esta aplicación y la luz blanca provocará obtener el mayor contenido espectral en la iluminación. Además, se puede apreciar el teléfono/cámara señalada con una flecha azul y se puede notar que esta sostenida con un trípode, este último aporta estabilidad y permite la reproducibilidad de resultados. Por último, y muy importante, es la distancia de trabajo a la que se realiza la captura, la cámara se ubica a más de 70cm de altura con respecto a la posición de los dados; esto con el objetivo de que se reduzcan los efectos de perspectiva [4]. Esto es importante porque lo que se desea en la captura de la imagen es fotografiar la cara superior de los dados.

Al reducir el efecto de perspectiva y mantener la cámara nivelada, de forma que la lente quede viendo perpendicularmente la cara superior de los dados se logra que, en la imagen solo se vea la cara superior de los dados. Si esto no se realiza, en la imagen se puede notar que se pueden ver porciones de las caras laterales si los dados están muy lejos del punto focal de la cámara. En resumen, la altura permite aumentar el espacio de trabajo efectivo y que los dados puedan estar más separados entre sí, como es de esperar al tirar dados de forma manual. Para la altura seleccionada (70cm), se determinó una condición de funcionamiento: que los tres dados deben caer en una *zona de fidelidad*, que corresponde a la región interna de una circunferencia de 20cm de diámetro centrada en el punto focal de la cámara.

Cabe resaltar que, como el fondo es de material textil, la fricción no permite que los dados se desplacen muy lejos y facilitan que los dados queden dentro de la *zona de fidelidad* a la hora de tirar los dados. Bajo todas estas condiciones, se obtiene una imagen de entrada de 6936x9248px con una resolución estándar de 64MP y un peso de archivo promedio de 10MB como la que se muestra en la Fig. 3



Figura 3: Imagen de entrada (IMG_38).

Esta es la imagen por ser procesada por el sistema de visión desarrollado. Se realizaron un total de 73 distintas capturas para tener una muestra significativa de distintos resultados de "tiradas" de dado.

II-B. Preprocesado

II-B1. Preprocesado General: Con la imagen de entrada se realiza un preprocesado de forma que se mejoren las características de la imagen antes de implementar las rutinas principales de forma que sean más efectivas y rápidas.

El primer aspecto por corregir es el tamaño de la imagen. La imagen de entrada es de 6936x9248px, por lo tanto es de utilidad reducir el tamaño de la imagen y eliminar la porción de ésta que no aporte información de utilidad, pues trabajar con la imagen original demandaría mucho mas recursos computacionales y tiempo de procesado. Por esta razón, se reutilizó la función *edit_image* (véase las líneas 28-35 del código del anexo V-A) generada en el miniproyecto del curso de sistemas de visión anteriormente (se realizó ciertas modificaciones para esta solución), la cual realiza dos tareas: recortar la imagen de forma que solo quede la cuadrícula central de la imagen original con un pequeño offset de 250px hacia la derecha (operación crop); y escalar la imagen recortada a un tamaño definido por el parámetro SIZE que en este caso es de 1544x1158px (resize) este tamaño resultante permite una mayor rapidez de procesado sin perder la información fundamental de interés [5] (fondo y cara superior de los dados). La imagen resultante se aprecia en la Fig. 4.

Se aprecia que todos los elementos fotografiados que no son de utilidad como las patas del trípode, el trípode o el fotógrafo son eliminados de forma estratégica. Sin embargo, si se presta atención a la Fig. 4 aún se pueden apreciar detalles y dobleces en el fondo textil oscuro, por lo tanto aún es necesario encontrar una forma de simplificar la imagen todavía más.



Figura 4: Imagen después de la operación edit_image (IMG_20).

II-B2. Cuantificación en 2 colores: La imagen ahora tiene muchas mejores características y solo cuenta con la información justa y necesaria por decodificar (lectura), sin embargo, es prudente realizar un paso extra que simplifique todavía más la imagen. Se trata de una cuantificación de colores, de forma que todos los colores de la imagen se reduzcan a únicamente dos colores: claro (para los dados) y oscuro (para el fondo y los puntos de los dados).

Para esto se utiliza el paradigma de K-Means (el cual se explicará con detalle más adelante), con dos centroides. Esto es posible con la función *img_to_2_colors* (véase las líneas 37-57 del código del anexo V-A). Este paso extra es recomendable porque de esta forma se obtienen mejores resultados a la hora de aplicar funciones de detección de bordes y posteriormente, la Transformada de Hough como se mencionará mas adelante en este informe. El resultado se aprecia en la Fig. 5. En resumen esta cuantificación funciona como un filtro.

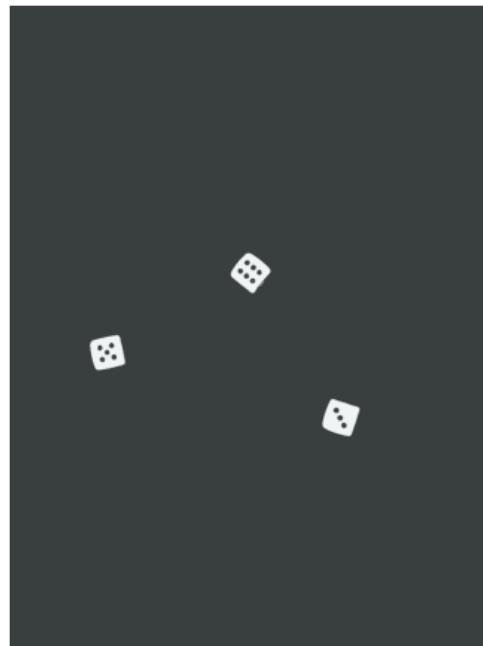


Figura 5: Imagen después de la operación img_to_2_colors (IMG_20).

De esta forma la imagen está lista para ser procesada con las funciones principales de la solución.

II-C. Lectura de tres dados iguales

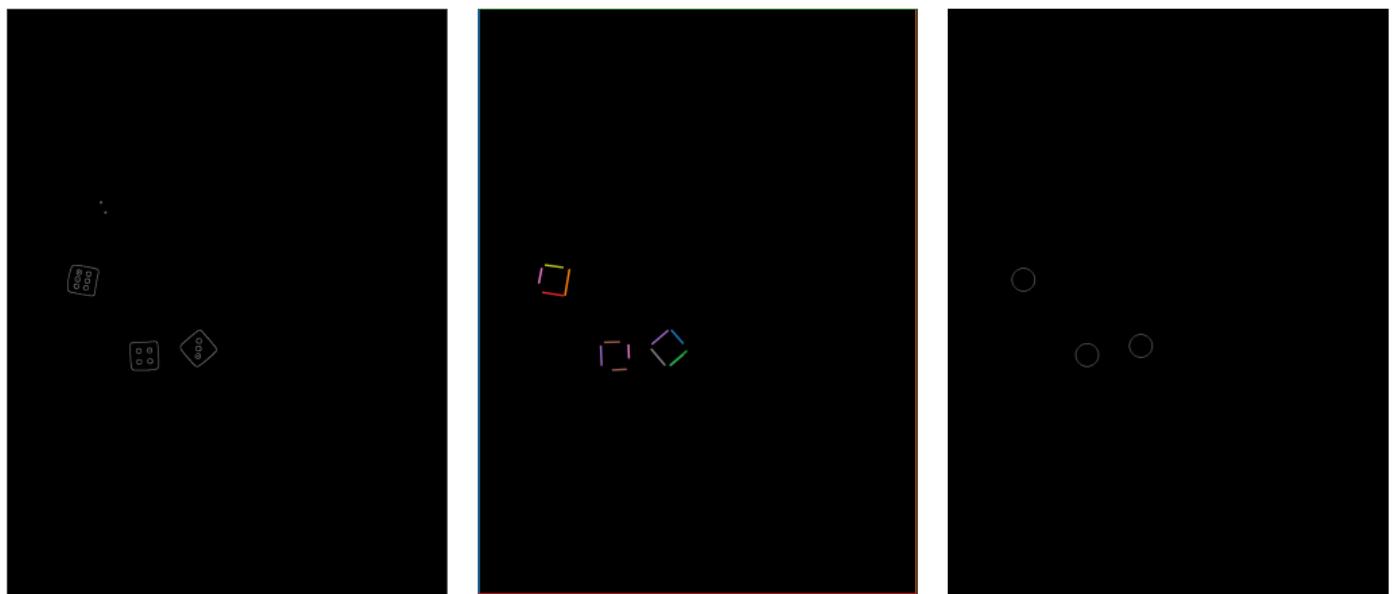
La solución propuesta realiza dos conjuntos de rutinas de forma secuencial. El primer conjunto tiene la función de identificar o segmentar cada uno de los datos independientemente de su valor codificado y la segunda rutina toma cada dato previamente aislado y realiza la lectura de la información que codifica cada uno. En las secciones II-C1 y II-C2 se detallan estos subprocesos.

II-C1. Identificación de cada dado: Este subproceso presenta los siguientes retos: por requerimiento, los tres dados son iguales (misma forma, color, tamaño, etc), esto significa que no hay forma de diferenciar cada uno de los dados mas que por su posición y separación, no se puede utilizar los puntos pues estos valores cambian para cada "tirada" y no sirven como forma de diferenciar datos, pues puede existir la posibilidad de que los tres dados caigan con el mismo valor (e.g 4, 4, 4). Por último, se debe tomar en cuenta que los dados pueden caer en cualquier posición y orientación dentro de la *zona de fidelidad*.

Bien, se tiene tres objetos con relleno claro y un fondo oscuro, además por las características de la captura explicadas en la sección II-A los objetos son fundamentalmente cuadrados (con esquinas redondeadas). El primer paso de este proceso es aprovechar esta naturaleza dual (claro-oscuro) mediante la función de skimage.color *rgb2gray* que convierte una imagen en RGB a escala de grises (toma el valor promedio de los valores R,G y B). Posteriormente se utiliza la función de la librería skimage.feature *canny* para encontrar los bordes de la imagen. El resultado se aprecia en la Fig. 6a.

Sobre los bordes de la imagen, se utiliza la Transformada de Hough. Esta última es un tipo de segmentación "cognitiva" que busca identificar elementos de una imagen mediante ecuaciones, en este caso, rectas [1]. La librería de skimage.transform tiene la función *probabilistic_hough_line* que realiza la transformada de Hough para encontrar líneas: para cada par de puntos determina una recta mediante representación polar (rectas definidas por los parámetros polares ρ y θ) y busca concentraciones de pares ordenados (ρ, θ) (mínimo 10, definido por el parámetro *threshold*); adicionalmente, realiza una subrutina que encuentra los extremos de las líneas, de forma que convierte "rectas en segmentos", la salida de esta función es una lista de segmentos lineales, donde cada segmento está definido por dos puntos (inicial y final) con coordenadas espaciales (e.g segmento_1 = $[(x_i, y_i), (x_f, y_f)]$), para un total de cuatro coordenadas para cada segmento [6]. Mediante los parámetros *line_length* y *line_gap* se define la distancia mínima del segmento lineal (estratégica para el tamaño de los dados) y se creó una función auxiliar *delete_long* que elimina los segmentos demasiados largos (generados en los bordes). Esta función a su vez, hace uso de una función *dist_puntos* (véase las líneas 59-69 y 128 del código del anexo V-A) que calcula de distancia entre dos puntos. El resultado se aprecia en la Fig. 6b.

Una vez que se encuentran los segmentos lineales, estos representan los bordes de todos los dados. Lo que se realiza ahora es utilizar el paradigma de K-Means que es un tipo de segmentación iterativo y autónomo. El proceso *centroides_dados* toma centroídes iniciales, agrupa cada coordenada o pixel al centroide que esté más cerca, luego redefine el centroide en función de los puntos que agrupó y repite el proceso hasta alcanzar un criterio de parada [1]. A diferencia de la función *img_to_2_colors* que trabaja en el espacio de colores, se utiliza la función de KMeans de la librería sklearn.cluster con coordenadas de píxeles, pero utilizando únicamente los puntos que definen los segmentos de línea encontrados previamente (se utiliza la función auxiliar *lineas2puntos* para tratar cada punto por separado) [6]. Al definir tres centroídes, se encuentran los centros de los tres dados como se aprecia en la Fig. 6c. (véase las líneas 71-81 del código del anexo V-A)



(a) Bordes de la imagen con Canny

(b) Transformada de Hough de los bordes.

(c) Centros de cada dado.

Figura 6: Subproceso de identificación de tres dados (Muestra #38).

De esta forma queda identificado cada uno de los dados mediante una coordenada de centro. Ahora se puede pasar al subproceso de lectura.

II-C2. Lectura de cada dado: Este subprocesso tiene como objetivo detectar los puntos de los datos y asociar cada punto a un dato específico de forma que se tienen tres datos y cada dato tendrá una cantidad de puntos asociados a él.

Para empezar, se utiliza la imagen de bordes realizada previamente con Canny. Luego, para identificar la presencia de puntos o orificios en la imagen, se utiliza la función *hough_circle* de la librería *skimage.transform*, que realiza un proceso similar a la Transformada de Hough para rectas, pero en vez de buscar y definir ecuaciones de recta, busca y define ecuaciones de circunferencia y tiene la característica de que retorna la n cantidad de círculos que tengan mejor ajuste.

Para utilizar esta función, se definió las funciones *image_circles* y *uniq_circles*. La primera función define el rango de tamaños que puede tener los círculos a detectar (en este caso de 5 a 10 píxeles pues este es el rango de tamaños del radio de los puntos de los datos) e invoca la función *hough_circle* para que encuentre los mejores 18 círculos [6]. La segunda función se encarga de eliminar los círculos que están repetidos; esto es importante porque la función *image_circles* siempre retorna 18 círculos (máximo posible cuando se tiene la configuración 6,6,6) y se generan círculos repetidos cuando hay menos de 18 círculos. Entonces en caso de que hayan menos de 18 círculos, la función *uniq_circles* se encarga de eliminar los círculos repetidos y solamente dejar la cantidad correcta de círculos por asociar. (véase las líneas 83-107 del código del anexo V-A)

Por último, está la función *segmentacion_por_dado* que toma los círculos generados con las funciones *image_circles* y *uniq_circles* y mediante un criterio de cercanía a los centros generados por la función *centroides_dados*, asocia cada círculo al dato que tenga mayor cercanía con él y si se encuentra a una distancia menor de 40 píxeles. Luego de este asocie, se realiza un conteo de círculos en cada uno de los datos y este es el resultado que se deseaba. (véase las líneas 109-121 del código del anexo V-A).

En este punto, la lectura se realizó correctamente y se tiene una lista de tres valores con la cantidad de puntos o círculos en cada dato. Ahora es necesario generar una forma de visualizar más fácilmente el resultado generado. Esto se realiza en el proceso de reporte de resultados (se explicará más adelante en la sección II-D).

II-D. Reporte de resultados

Para poder realizar la lectura completa de los datos y la generación del reporte con los resultados se crearon dos funciones, *cuenta_puntos* y *main*. La función *cuenta_puntos* se encarga de realizar todo el proceso de preprocesado y procesado de la imagen, de forma que llama a las funciones definidas anteriormente. Esta función recibe una imagen, la recorta y escala, luego la segmenta en 2 colores y la binariza. Seguido se obtienen los bordes de la imagen y los utiliza para encontrar las líneas que delimitan cada dato y con ellas se determina el centroide que representa a cada dato. Luego se localizan los centros de los puntos de cada dato y se procede a segmentar los puntos por dato y calcular el total. Retornando el valor total de puntos y una lista con los puntos que hay en cada dato.

Asimismo, la función *main*, crea el archivo donde se guardan los resultados *report_file*, si el archivo existe primero lo elimina para escribir siempre sobre un archivo en blanco. Luego determina el *path* en el que se encuentran las imágenes por analizar y ejecuta la función *cuenta_puntos* para cada imagen de la carpeta de imágenes, guardando en el archivo *report_file* los resultados de cada imagen, tal que se escribe una línea por imagen con el formato: *IMG_numero de imagen: Valor de cada dato: valor dato 1, valor dato 2 y valor dato 3*, y los puntos totales son *valor total de los datos*.

III. ANÁLISIS DE RESULTADOS

En esta sección se realiza el análisis de los principales resultados obtenidos de la ejecución del código explicado en la sección II. Para esto se utilizaron las imágenes de las Figs. 3, 7 y 8, estas imágenes fueron seleccionadas aleatoriamente entre 73 imágenes que fueron tomadas. Además, en el anexo V-B1 se muestra el archivo *report_file* donde se muestran los resultados obtenidos de cada imagen.



Figura 7: Imagen de entrada (IMG_09).

Como se aprecia, las imágenes varían tanto en la posición de los dados como en la puntuación de los dados. Los resultados obtenidos en el reporte de resultados se resumen en el cuadro I.

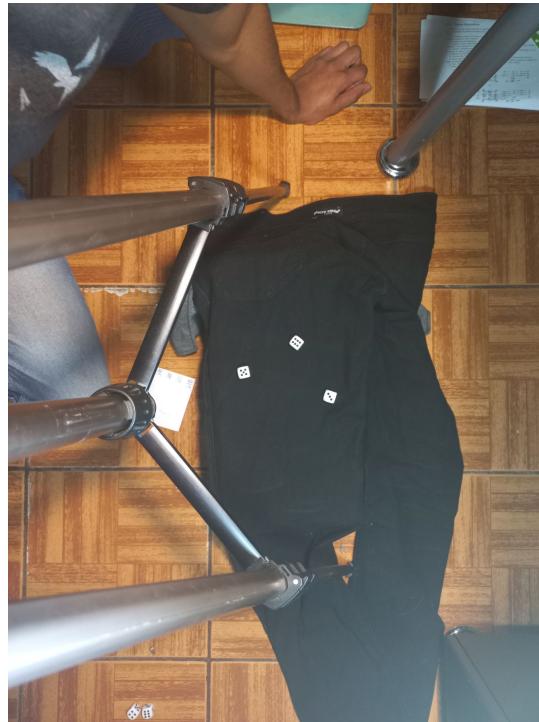


Figura 8: Imagen de entrada (IMG_20).

Al comparar los resultados obtenidos con las imágenes de entrada se observa que los resultados son correctos, y la puntuación por dado y total coinciden con las observadas en las imágenes de las Figs. 7, 8 y 3 respectivamente. Además, se aprecia como la asignación de los dados no se ve afectada por su posición en la imagen o puntuación, por lo tanto se puede obtener la puntuación por dado, sin embargo no se puede indicar cual dado corresponde al dado 1, 2 o 3.

Cuadro I: Resumen de resultados

Imagen	Dado 1	Dado 2	Dado 3	Puntuación Total
IMG_09	4	1	5	10
IMG_20	5	3	6	14
IMG_38	6	3	4	13

En los resultados obtenidos también se evidencia la robustez de la aplicación, capaz de ignorar las áreas despreciables de la imagen, aceptando imágenes donde los datos se posicionen en cualquier configuración ya sea muy juntos o muy dispersos, esto siempre y cuando los datos se posicionen dentro de la zona de fidelidad, un círculo de 20 cm de diámetro centrado en el punto central del campo de visión.

La aplicación funciona con seguridad únicamente si se cumplen con ciertos criterios:

- Los datos se ubican en la zona de fidelidad
- El fondo es homogéneo y regular, de un color diferente al de los datos
- Los tres datos utilizados son iguales

Esto se debe a que si no se cumplen estas condiciones no se puede asegurar que la aplicación opere correctamente.

Una herramienta que fue de suma utilidad para la solución de esta tarea fue *KMeans*, debido a que permitió realizar la segmentación de la imagen por color y su posterior binarización, sino que también permitió ubicar los datos y con esto obtener la puntuación individual de cada dato, segmentando los puntos encontrados.

Adicionalmente, se ejecutó el código para un total de 20 imágenes incluyendo las 3 imágenes de las Figs. 7, 8 y 3. Las 17 imágenes adicionales se muestran en el anexo V-B2, y el reporte de resultados de la ejecución en el anexo V-B3. Esto con la finalidad de probar la aplicación con una muestra más grande y una mayor variedad de configuraciones de los datos.

IV. CONCLUSIONES Y RECOMENDACIONES

De esta tarea se concluyeron las siguientes afirmaciones:

- Si no se conocen los colores principales de una imagen, o no se sabe en qué colores segmentar, una herramienta muy útil es *KMeans* que permite segmentar una imagen en una cantidad k de colores, facilitando así una segmentación por color si no se conocen los colores para segmentar o se desea que cambien según la imagen.
- El algoritmo de *KMeans* funciona con matrices de diferentes tipos, de forma que puede ser aplicado con diferentes propósitos además de la segmentación, en este caso se utiliza para identificar los centroides y con ellos la ubicación de cada dato en la imagen, utilizando una matriz con puntos pertenecientes al perímetro de los diferentes datos.
- La transformada de Hough permite identificar diferentes geometrías en una imagen, algo que es de suma utilidad en visión artificial, en el caso de líneas y círculos hay funciones incluidas en la librería *scikit-image* para python, que permiten identificar ambas geometrías permitiendo la configuración de varios parámetros para identificar únicamente las geometrías que cumplen las condiciones definidas.

Además, se dan las siguientes recomendaciones basadas en la experiencia adquirida durante la realización de la tarea 3:

- En una aplicación de visión artificial la captura de la imagen es una de las partes más importantes, puesto que puede facilitar o dificultar el preprocesado y procesado de la imagen, por lo que se recomienda tener en cuenta aspectos como la distancia de trabajo y resolución espacial que se desea, para que permita capturar los detalles necesarios para la aplicación, como lo es el caso de los puntos en los datos en esta tarea. Asimismo, se debe poner atención a la iluminación y el fondo de la imagen.
- En caso de que no se conozca la ubicación de algún objeto pero se tengan puntos pertenecientes a dicho objeto, si se conoce la cantidad de objetos deseados, utilizando *KMeans* se puede ubicar el centroide de los objetos, y ya sea utilizarlos para segmentación o algún otro proceso.
- Si se desea ejecutar un código para múltiples archivos, la librería *io* en python incluye muchas funciones de gran utilidad, que permiten obtener la ubicación de un archivo y los archivos dentro de una carpeta.
- Para procesado de imágenes en *python* se recomienda utilizar la librería *scikit-image*, puesto que incluye una gran variedad de funciones que facilitan trabajar con imágenes.
- Al utilizar las funciones de la transformada de Hough en python se debe tomar en cuenta que identifican todas las geometrías que cumplen con las condiciones dadas, por lo que si se desea discriminar algunas se pueden crear funciones extras que filtran las geometrías identificadas.

REFERENCIAS

- [1] J. Serrano, V. Staff, A. Diaz, A. Calle, and J. Sanchez-Marin, *Vision Por Computador*. Textos docentes, Dykinson, S.L., 2003.
- [2] J. L. Crespo, “Tarea parcial evaluable número 3: Consideraciones cognitivas,” tech. rep., Tecnológico de Costa Rica, 2022.
- [3] tablademultiplicar.net, “Dado.”
- [4] G. Martinsanz, J. de la Cruz García, and A. TOME, *Visión por Computador: Imágenes digitales y aplicaciones*. RA-MA S.A. Editorial y Publicaciones, 2001.
- [5] C. Jara and C. Morales, “Miniproyecto - uso de visión para caracterización de objetos y determinación de presencia / ausencia,” tech. rep., Tecnológico de Costa Rica, 2022.
- [6] scikit-image development team, “scikit-image: Image processing in python.”

V. ANEXOS

V-A. Código

V-A1. Medición del ancho de un objeto con VA: Código utilizado para determinar el ancho de tres objetos a distintas distancias de trabajo

```

1 # Importar las librerías por utilizar
2 import os
3 import numpy as np
4 from skimage import io
5 from skimage.transform import resize, probabilistic_hough_line, hough_circle, hough_circle_peaks
6 from skimage.feature import canny
7 from skimage.draw import circle_perimeter
8 from sklearn.cluster import KMeans
9 from sklearn.utils import shuffle
10 from skimage.color import rgb2gray
11
12 #-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*
13
14 #PARAMETROS
15 #Tamaño de trabajo
16 SIZE=(1544,1158)
17
18 #numero de colores incluyendo el fondo
19 N_COLORS = 2
20
21 MAX_RHO = 18
22 MAX_THETA = 5*np.pi/180
23
24 #-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*-----*
25
26 #FUNCIONES
27
28 def edit_image(image):
29     y,x = image.shape[:2]
30     y_div = y//3
31     x_div = x//3
32     x_off = 250
33     cropped = image[y_div:2*y_div,x_div+x_off:2*x_div+x_off]
34     resized = resize(cropped,(SIZE[0],SIZE[1]),preserve_range=True).astype(int)
35     return resized
36
37 def recreate_image(codebook, labels, w, h):
38     return codebook[labels].reshape(w, h, -1)
39
40 def img_to_2_colors(img):
41     # Copia de la imagen por cuantificar
42     img_5_colors = img
43     # Convert to floats instead of the default 8 bits integer coding. Dividing by
44     # 255 is important so that plt.imshow behaves works well on float data (need to
45     # be in the range [0-1])
46     img_5_colors = np.array(img_5_colors, dtype=np.float64) / 255
47     # Load Image and transform to a 2D numpy array.
48     w, h, d = original_shape = tuple(img_5_colors.shape)
49     assert d == 3
50     image_array = np.reshape(img_5_colors, (w * h, d))
51     image_array_sample = shuffle(image_array, random_state=0, n_samples=1_000)
52     kmeans = KMeans(n_clusters=N_COLORS, random_state=0).fit(image_array_sample)
53     labels = kmeans.predict(image_array)
54     colores = np.round(np.array(kmeans.cluster_centers_, dtype=np.float64) * 255)
55     colores = np.array(colores,dtype=np.int32)
56     img_quant = recreate_image(kmeans.cluster_centers_, labels, w, h)
57     return colores,img_quant
58
59 def dist_puntos(p0,p1):
60     return np.sqrt((p1[1]-p0[1])**2+(p1[0]-p0[0])**2)
61
62 def delete_long(linesdl):
63     newlinesdl = []
64     for linedl in linesdl:
65         p0dl, p1dl = linedl
66         len = dist_puntos(p0dl,p1dl)
67         if len < SIZE[1]/2:
68             newlinesdl.append(linedl)
69     return newlinesdl
70
71 def lineas2puntos(lineas):
72     puntos = []
73     for i in lineas:

```

```

74     puntos.append(i[0])
75     puntos.append(i[1])
76     return puntos
77
78 def centroides_dados(puntos):
79     kmeans = KMeans(n_clusters=3, random_state=0).fit(puntos)
80     centroides = kmeans.cluster_centers_
81     return centroides
82
83 def image_circles(edges):
84     circles = []
85     hough_radii = np.arange(5, 10)
86     hough_res = hough_circle(edges, hough_radii)
87     # Select the most prominent 18 circles
88     accums, cx, cy, radii = hough_circle_peaks(hough_res, hough_radii, total_num_peaks=18)
89     for i in range(len(cx)):
90         temp = (cx[i], cy[i])
91         circles.append(temp)
92     return circles
93
94 def uniq_circles(circulos):
95     min_diff = 2
96     uniq = []
97     uniq.append(circulos[0])
98     for i in circulos:
99         uniq_circ = True
100        for k in uniq:
101            dist = dist_puntos(i,k)
102            #Si no es unico
103            if dist <= min_diff:
104                uniq_circ = False
105        if uniq_circ:
106            uniq.append(i)
107    return uniq
108
109 def segmentacion_por_dado(dados,puntos_unicos):
110     dist_max = 40
111     dado_correspondiente = []
112     for i in puntos_unicos:
113         for k in range(len(dados)):
114             dist = dist_puntos(i,dados[k])
115             if dist <= dist_max:
116                 dado_correspondiente.append(k)
117     puntos_d1 = dado_correspondiente.count(0)
118     puntos_d2 = dado_correspondiente.count(1)
119     puntos_d3 = dado_correspondiente.count(2)
120     total = puntos_d1 + puntos_d2 + puntos_d3
121     return [puntos_d1,puntos_d2,puntos_d3],total
122
123 def cuenta_puntos(image):
124     img_edit = edit_image(image)
125     COLORES, img_quant=img_to_2_colors(img_edit)
126     image_gray = rgb2gray(img_quant)*255**3/2
127     edgesHT = canny(image_gray, 2, 1, 25)
128     linesHT1 = probabilistic_hough_line(edgesHT, threshold=10, line_length=30, line_gap=3)
129     linesHT2 = delete_long(linesHT1)
130     puntosHT = lineas2puntos(linesHT2)
131     centroides = centroides_dados(puntosHT)
132     c_puntos_dados = image_circles(edgesHT)
133     puntos_unicos = uniq_circles(c_puntos_dados)
134     puntos_por_dado, total = segmentacion_por_dado(centroides,puntos_unicos)
135     return puntos_por_dado, total
136
137 def main():
138     if os.path.exists("report_file.txt"):
139         os.remove("report_file.txt")
140     report_file = open("report_file.txt", "x")
141     path = os.getcwd()
142     folder = 'Imagenes_por_analizar'
143     folder_path = os.path.join(path, folder)
144     files = os.listdir(folder_path)
145     for file in files:
146         res_report = ''
147         file_path = os.path.join(folder_path, file)
148         foto = io.imread(file_path)
149         puntos_por_dado, puntos_totales = cuenta_puntos(foto)
150         resultado = f": Valor de cada dato: {puntos_por_dado[0]}, {puntos_por_dado[1]} y {puntos_por_dado[2]}, y los puntos totales son {puntos_totales}."
151         res_report += resultado + '\n'

```

```
152     report_file.write(res_report)
153     report_file.close()
154
155 if __name__ == "__main__": main()
```

V-B. Resultados adicionales

V-B1. Reporte de resultados: Archivo de texto correspondiente al *report_file*, generado después de la ejecución del código en el anexo V-A utilizando las imágenes de las Figs. 3, 7 y 7.

```
1 IMG_09.jpg: Valor de cada dado: 4, 1 y 5, y los puntos totales son 10.
2 IMG_20.jpg: Valor de cada dado: 5, 3 y 6, y los puntos totales son 14.
3 IMG_38.jpg: Valor de cada dado: 6, 3 y 4, y los puntos totales son 13.
```

V-B2. Imágenes adicionales: Se muestran las 17 imágenes adicionales para la ejecución del código a un total de 20 imágenes.

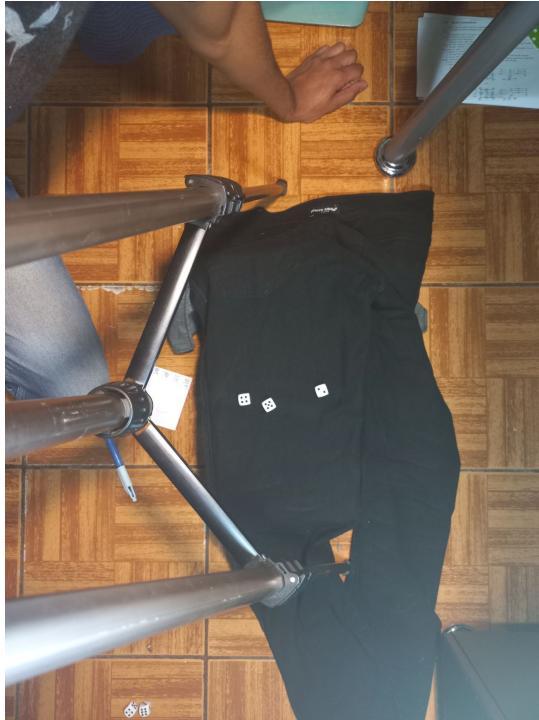


Figura 9: Imagen adicional de entrada (IMG_06).

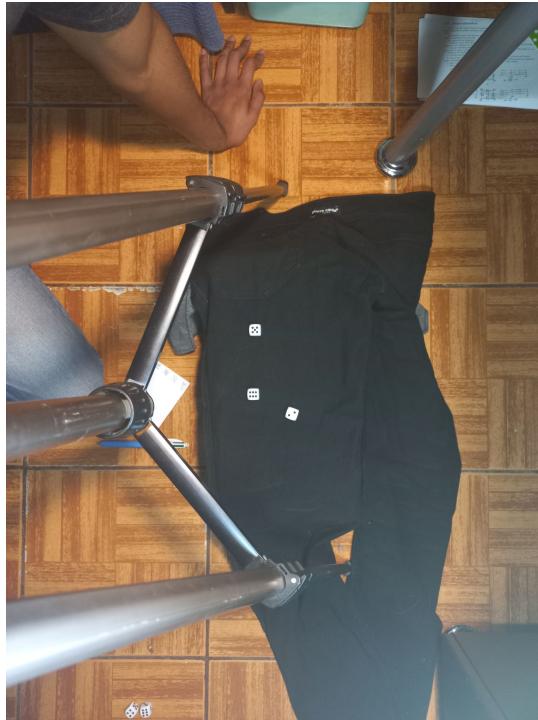


Figura 10: Imagen adicional de entrada (IMG_07).



Figura 11: Imagen adicional de entrada (IMG_12).

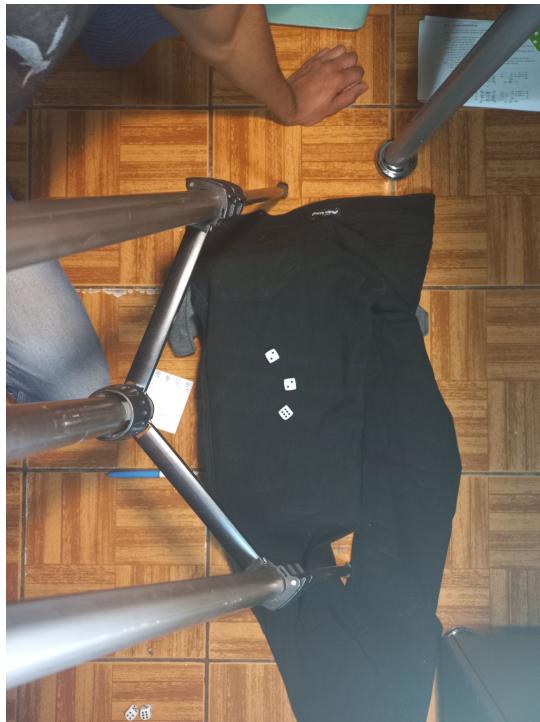


Figura 12: Imagen adicional de entrada (IMG_14).



Figura 13: Imagen adicional de entrada (IMG_25).

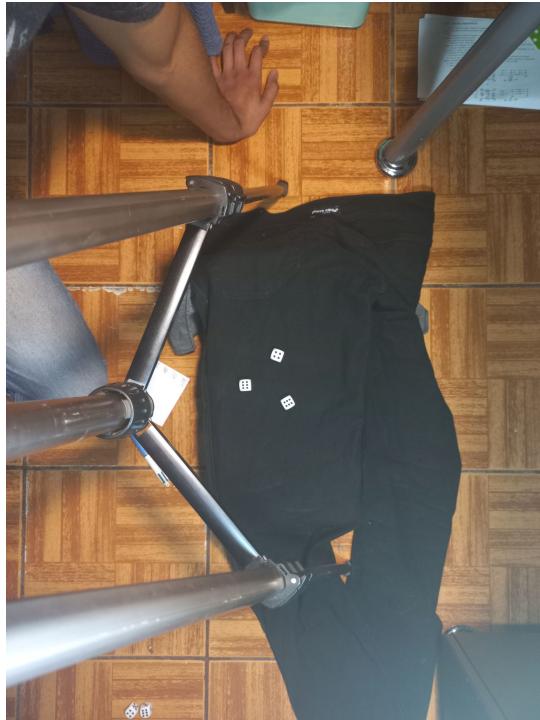


Figura 14: Imagen adicional de entrada (IMG_33).



Figura 15: Imagen adicional de entrada (IMG_35).



Figura 16: Imagen adicional de entrada (IMG_41).



Figura 17: Imagen adicional de entrada (IMG_42).



Figura 18: Imagen adicional de entrada (IMG_49).



Figura 19: Imagen adicional de entrada (IMG_52).

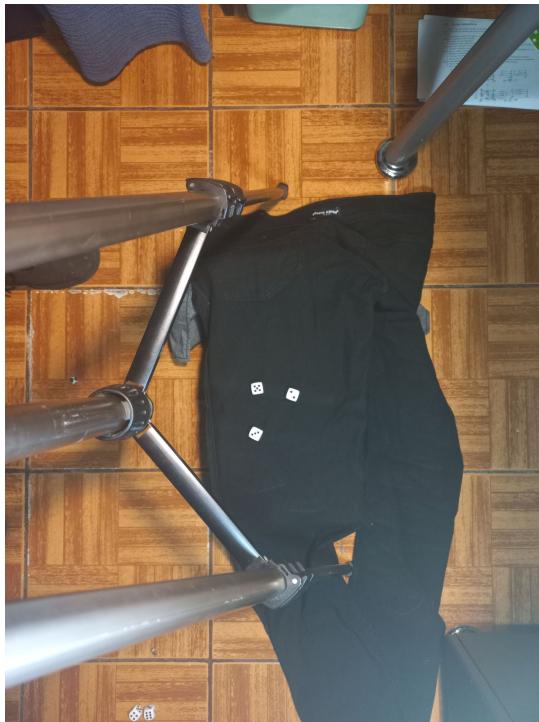


Figura 20: Imagen adicional de entrada (IMG_53).

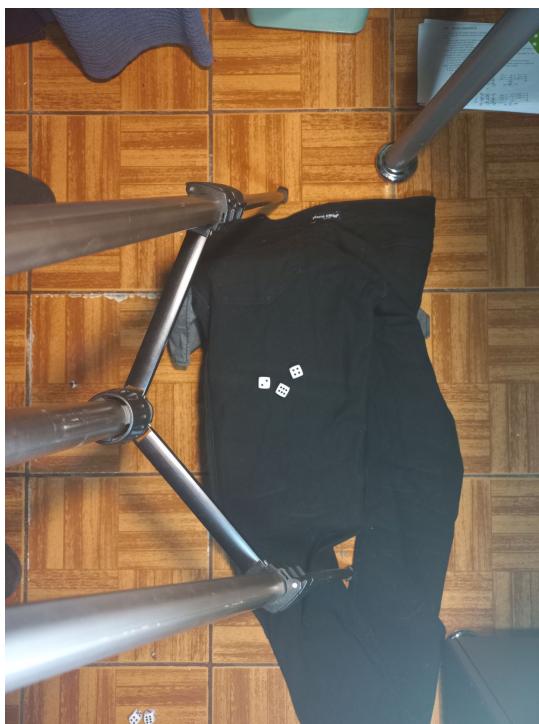


Figura 21: Imagen adicional de entrada (IMG_54).

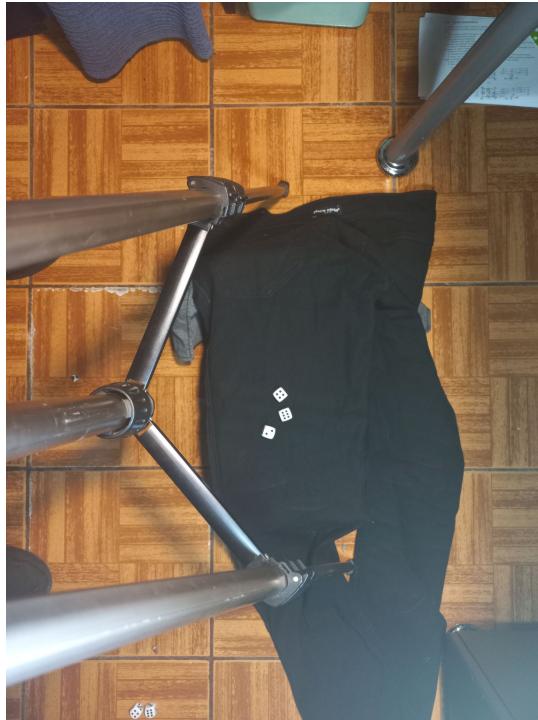


Figura 22: Imagen adicional de entrada (IMG_63).

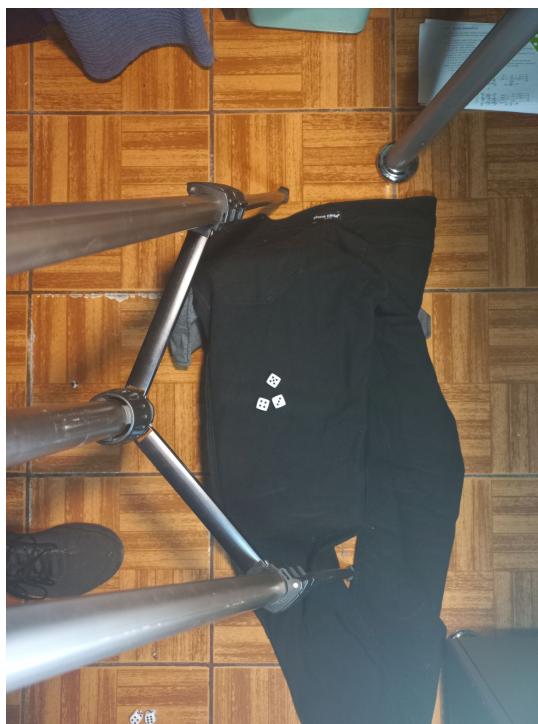


Figura 23: Imagen adicional de entrada (IMG_66).



Figura 24: Imagen adicional de entrada (IMG_71).

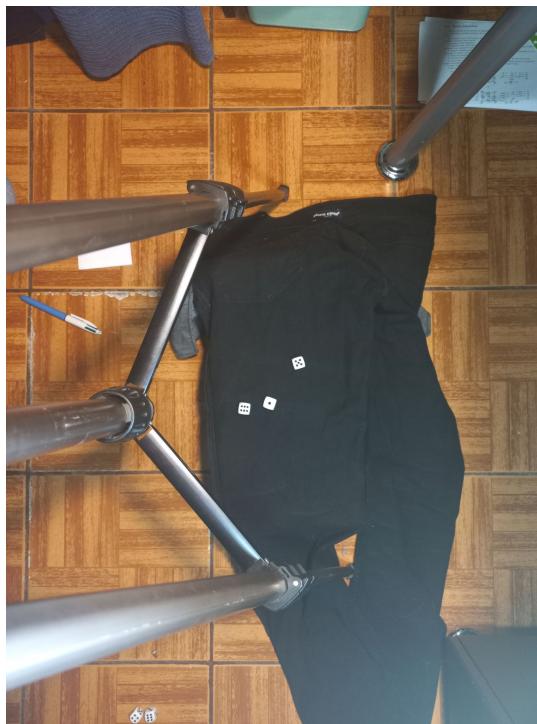


Figura 25: Imagen adicional de entrada (IMG_72).

V-B3. Reporte de resultados adicional: Archivo de texto correspondiente al *report_file*, generado después de la ejecución adicional del código en el anexo V-A para 20 imágenes, utilizando las imágenes de las Figs. 3, 7 y 7, y las imágenes del anexo V-B2.

- 1 IMG_06.jpg: Valor de cada dado: 2, 4 y 5, y los puntos totales son 11.
- 2 IMG_07.jpg: Valor de cada dado: 6, 2 y 5, y los puntos totales son 13.
- 3 IMG_09.jpg: Valor de cada dado: 1, 5 y 4, y los puntos totales son 10.
- 4 IMG_12.jpg: Valor de cada dado: 2, 1 y 4, y los puntos totales son 7.
- 5 IMG_14.jpg: Valor de cada dado: 2, 6 y 2, y los puntos totales son 10.
- 6 IMG_20.jpg: Valor de cada dado: 6, 3 y 5, y los puntos totales son 14.
- 7 IMG_25.jpg: Valor de cada dado: 5, 5 y 3, y los puntos totales son 13.
- 8 IMG_33.jpg: Valor de cada dado: 4, 6 y 6, y los puntos totales son 16.
- 9 IMG_35.jpg: Valor de cada dado: 2, 6 y 4, y los puntos totales son 12.

10 IMG_38.jpg: Valor de cada dado: 4, 6 y 3, y los puntos totales son 13.
11 IMG_41.jpg: Valor de cada dado: 3, 5 y 4, y los puntos totales son 12.
12 IMG_42.jpg: Valor de cada dado: 5, 2 y 4, y los puntos totales son 11.
13 IMG_49.jpg: Valor de cada dado: 2, 3 y 4, y los puntos totales son 9.
14 IMG_52.jpg: Valor de cada dado: 5, 2 y 6, y los puntos totales son 13.
15 IMG_53.jpg: Valor de cada dado: 3, 5 y 2, y los puntos totales son 10.
16 IMG_54.jpg: Valor de cada dado: 4, 6 y 2, y los puntos totales son 12.
17 IMG_63.jpg: Valor de cada dado: 4, 2 y 6, y los puntos totales son 12.
18 IMG_66.jpg: Valor de cada dado: 5, 4 y 3, y los puntos totales son 12.
19 IMG_71.jpg: Valor de cada dado: 6, 4 y 2, y los puntos totales son 12.
20 IMG_72.jpg: Valor de cada dado: 5, 1 y 6, y los puntos totales son 12.