22.90 - Automación Industrial

Instituto Tecnológico de Buenos Aires

Segundo Parcial 2023

Visión

AUTORES:
Ignacio Cutignola (59330)
Olivia De Vincenti (60354)
Valentino Venier Anache (60097)

PROFESORES:
Arias, Rodolfo Enrique
Sofio Avogadro, Federico
Spinelli, Mariano Tomás

Noviembre 2023

1. Introducción

En el siguiente informe, se dará una explicación de como funciona el programa presentado para el segundo parcial de la materia. En base a imágenes de un dado de 20 caras con vista de la cara superior, en una posición y rotación aleatoria, el programa realizará una serie de algoritmos para analizar el resultado final del dado.

Este resultado es uno aleatorio con una probabilidad de éxito de uno a veinte, es decir, todos lo números que pueden salir son equiprobables. Dado esto, se analizarán varios resultados de tiradas para poder abarcar un amplio rango de posibilidades.

Es importante aclarar que el grupo agregó más ejemplos de resultados (entregados por la cátedra) para verificar que el proceso sea correcto y asegurarse de estar haciendo bien el trabajo. Como las imágenes brindadas no eran muy definidas, a estos ejemplos se les aplicó un difuminado usando un kernel gaussiano para que así sean relativamente parecidas a las de la consigna.

Figura 1.1: Dado de 20 caras.

2. Proceso y Análisis

Para realizar un trabajo ordenado, se dividió el proceso en cinco etapas principales. Estas fueron elegidas para poder llegar a la solución final con pasos sencillos y relativamente rápidos y correctos.

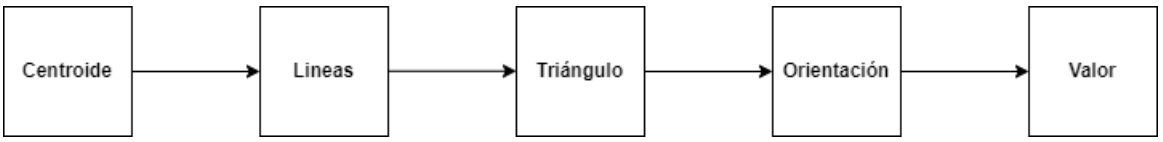

Figura 2.1: Etapas del Proceso.

2.1. Centroide

Primero se procedió a calcular el centroide de la imagen. Dado que las tiradas de los dados son únicas, en una imagen se obtiene un solo dado con un fondo de color blanco. Es por eso que aplicando los siguientes comandos de `MATLAB`, se pudo obtener el centroide del dado de veinte caras y, como consecuencia, su recorte para un mejor análisis.

La búsqueda del centroide se inicia con la aplicación de un *threshold* alto de 0.9 para poder separar el objeto del fondo de imagen. De esta forma se pudo generar un *blobs* con información del objetos en la imagen.

```
%% Buscamos centroide
imblack=imon>0.9;          % Aplicamos threshold que vuelve todo el dado negro
% idisp(imblack)
f = iblobs(imblack, 'class', 0, 'area', [10000, 30000])  % Buscamos blobs
    negros de area parecida al d20
```

Listing 1: Modelo discreto de variable de estado con script `P22023VisionG1.m`

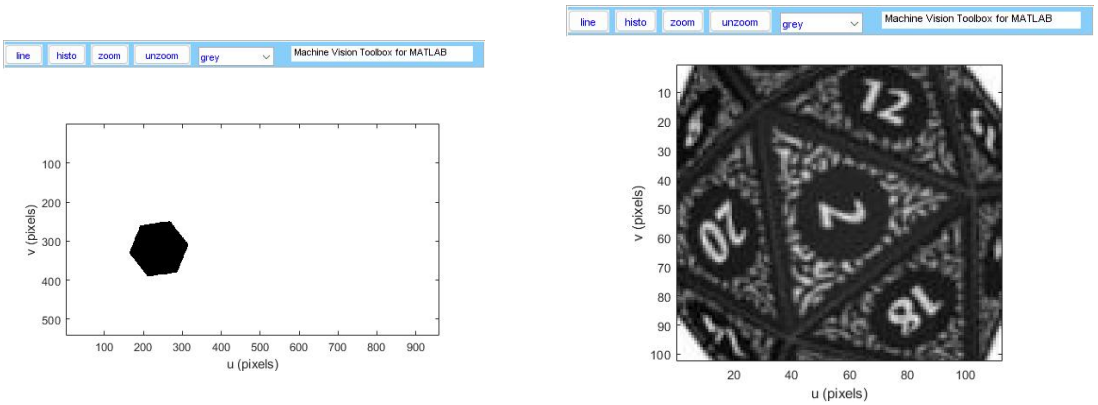

Figura 2.2: Etapa de Búsqueda del Centroide.

Una vez aplicando el código, se pudo obtener el centroide del objeto y concentrarse solamente en ello.

2.2. Líneas

Se ubicó el objeto en el centro de la imagen y se la recortó para extraer el triangulo central, es decir, la forma de la cara superior del dado. Se prosiguió con la obtención de sus lineas utilizando detección de bordes.

Primero, se le aplicó un *Threshold* apropiado a la situación. Inicialmente se probó aplicando el comando `otsu` para calcular el umbral óptimo, sin embargo, el resultado brindado no fue el esperado y la imagen no era correcta para el análisis buscado. Dado a esto, se eligió un umbral de 0.28 que fue un dato obtenido de forma iterativa.

Este proceso se pudo realizar con éxito gracias a la aplicación de dos operaciones de cierre dados primero por un espacio estructural circular de radio 3.5px y luego por una matriz de $5x5$px.

Luego, se utilizó el método de *Hough* para poder obtener las líneas correspondientes. Los parámetros elegidos de forma iterativa fueron muy correctos, se eligió un umbral de votos del 60 % y una supresión de 5, es decir, se evitó el cúmulo de muchas líneas en un mismo lugar.

Por otro lado, se filtraron las líneas de misma o parecida pendientes ya que la supresión sólo elimina las paralelas cercanas. Además, se limitaron las longitudes de las líneas obtener las tres del triángulo principal.

Como resultado se obtiene el triángulo central junto a otras líneas que luego no serán tenidas en cuenta.

```
% Recortamos triangulo del dado
imfoc=im(f.vmin+20:f.vmax-20, f.umin+20:f.umax-20);
%idisp(imfoc)

%% Rellenamos espacios hasta que las lineas principales sean las del
    triangulo
%ithresh(imfoc)
imth=imfoc>0.28;
imfull = iclose(imth, kcircle(3.5));
imfull = iclose(imfull, ones(5, 5));
%idisp(imfull)
%% Detecto lineas
edges = icanny(imfull);     % Obtenemos bordes
% idisp(edges)
% Aplicamos transformada de Hough con 60% de votos y promedio de supresion de
    5
h = Hough(edges, 'houghthresh', 0.6, 'suppress', 5);
lines = h.lines();          % Extraemos lineas
% idisp(imfull, 'dark');

% Filtramos las lineas que miden entre 25 y 60 ya que entre ellas se
% encuentran los bordes del triangulo principal
lines = lines.seglength(edges);
k = find( lines.length > 25 & lines.length <= 60);
% lines(k).plot('b--')
% lines(k)
```

```
25
26 % Suprimimos lineas paralelas con un angulo similar
27 filtered_lines = filter_lines(lines(k));
28 % filtered_lines.plot('b--')
```

Listing 2: Modelo discreto de variable de estado con script `P22023VisionG1.m`

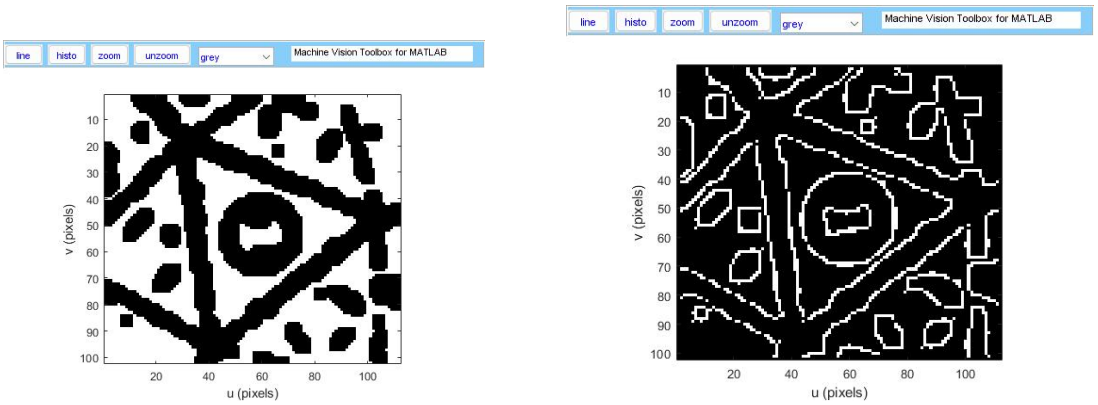

Figura 2.3: Etapa de Detección de Lineas.

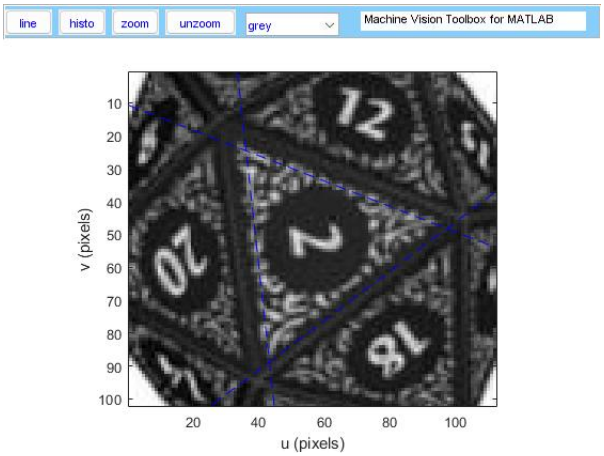

Figura 2.4: Etapa de Detección de Lineas.

2.3. Triángulo

A continuación, se busca formar el triángulo central a partir de la lineas y así obtener su contorno. Como el mismo es equilátero, se buscaron 3 líneas que se intersecan con ángulos iguales de 60°.
Luego buscamos la base del triangulo, consideramos que será la de ángulo menor (la más horizontal) que se encuentre en la mitad inferior de la imagen.

```
%% Chequeamos triangulos
% Buscamos un triangulo equilatero entre las lineas filtradas
% Pedimos que en sus intersecciones se formen angulos de 60 grados
tri_lines = find_triangle(filtered_lines);
% idisp(imfull, 'dark');
% tri_lines.plot('b--')
[~, idx] = sort(abs(tri_lines.theta));  % Ordenamos las lineas
tri_lines = tri_lines(idx)
base = tri_lines(1);    % Determinamos la linea mas horizontal como la base
% Si la linea mas horizontal tiene rho menor que 60 la imagen quedara al
    reves al girarla
if base.rho < 60
    base = tri_lines(2);    % En ese caso se toma la segunda linea mas
    horizontal
end
```

Listing 3: Modelo discreto de variable de estado con script `P22023VisionG1.m`

2.4. Orientación

Gracias a la base, se puede rotar la imagen con su ángulo para orientar correctamente el numero. Así se podremos compararlo y obtener el resultado del dado. Este proceso se realiza con el comando `irotate` que permite girar la imagen para que el número en la imagen esté derecho, es decir, una de las líneas debe estar de forma horizontal.

En el supuesto caso que la linea horizontal no llega a ser la correcta, no es un problema. Esto es así ya que se compararán las imágenes resultantes y las mismas rotadas unos $\pm 120^o$ y se considerará como resultado la imagen con mayor similitud.

```
%% Corrijo orientacion
imm=im(f.vmin:f.vmax, f.umin:f.umax);   % Extraemos el dado completo
vc = (f.vmax - f.vmin)/2;               % Ubicamos su centro
uc = (f.umax - f.umin)/2;
% Giramos la imagen hasta que la base quede horizontal
imrot = irotate(imm, base.theta*180/pi, 'extrapval', 1);
%idisp(imrot)

%% Obtenemos template de los numeros a evaluar
numbers_template = get_numbers();
% disp(numbers_template)
```

Listing 4: Modelo discreto de variable de estado con script `P22023VisionG1.m`

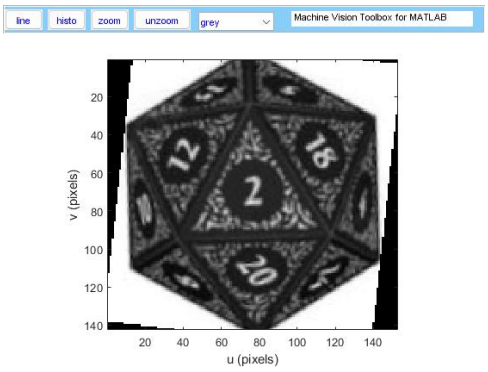

Figura 2.5: Etapa de Orientación: Posición 1.

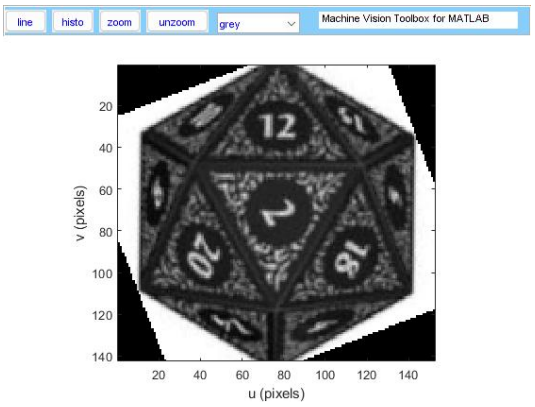

Figura 2.6: Etapa de Orientación: Posición 2.

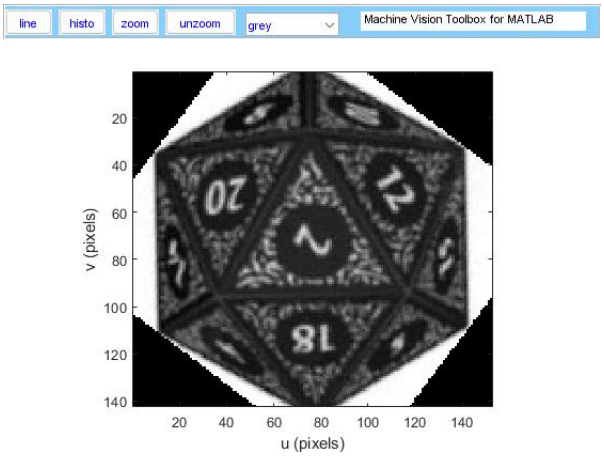

Figura 2.7: Etapa de Orientación: Posición 3.

2.5. Valor

Como se dijo previamente, gracias a la plantilla de dados entregadas por la cátedra, se guardaron diferentes resultados en un arreglo para recorrerlo en bucle y encontrar la mejor respuesta y la que más coincidencia tenga con el número final.

A partir de los siguientes plantillas, obtuvimos los números que utilizaremos para encontrar el resultado de la imagen que estamos analizando:

Figura 2.8: Templates.

De los mismos, aplicando *Threshold* y guiándose por las coordenadas, obtuvimos el siguiente vector compuesto por matrices 23x23. Resultado del siguiente código:

```
numSize = 22;

%********** NUMEROS **********
% 20
%                              Y                      X
Num20 = foto_threashold(90:(90+numSize),85:(85+numSize));
% 19
%                              Y                      X
Num19 = foto_threashold(87:(87+numSize),245:(245+numSize));
% 18
%                               Y                      X
Num18 = foto_threashold(96:(96+numSize),415:(415+numSize));

...
```

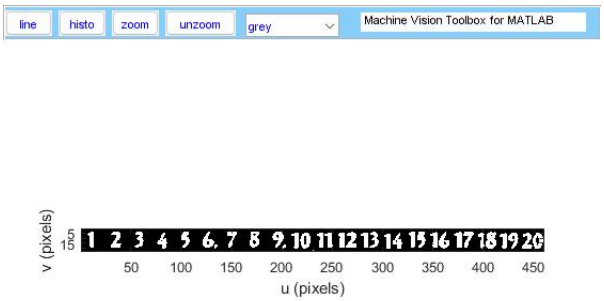

Figura 2.9: Vector que se utiliza como plantilla.

Este proceso se basa en un ciclo que recorre los veinte números de la plantilla, comparando cada uno con los tres tipos de rotaciones. La comparación que tenga más coincidencias de imagen, resulta ser el número correspondiente.

```
%% Buscamos numero
% Es necesario analizar las 3 posibles orientaciones del dado
number=0;
max_coincidence = 0;

for i=1:1:3
    %% Giramos dado
    %imm=im(f.vmin:f.vmax, f.umin:f.umax);
    imrot = irotate(imrot, 120, 'extrapval', 1);
    %idisp(imrot)

    %% Recortamos numero
    imfoc=imrot(vc-11:vc+11, uc-11:uc+11);  % Tiene que ser 23x23
    %idisp(imfoc)
    %[H, W] = size(imfoc)

    %% Aplicamos Threshold
    imth_foc=imfoc>0.45;
    imth_foc = imth_foc*255;
    %idisp(imth_foc)

    %% Buscamos el numero, tienen que ser los dos Threshold
    [number_aux, coincidence_aux] = find_number(numbers_template, imth_foc);
    if coincidence_aux > max_coincidence
        number = number_aux;
        max_coincidence = coincidence_aux;
    end

end

% Resultado
```

```
32 disp(['El valor del dado arrojado es: ', num2str(number)]);
```

Listing 5: Modelo discreto de variable de estado con script `P22023VisionG1.m`

Como resultado, en la terminal del programa se obtiene lo siguiente:

```
El valor del dado arrojado es: 2
```

3. Conclusión

Por todo lo analizado anteriormente, mediante algoritmos de análisis de imágenes que puede ser utilizado para digitalizar un juego de rol sin inconvenientes. Se emplearon distintas técnicas de visión para acelerar el tiempo de procesamiento y así no retrasar el juego.

4. Bibliografía

- Corke, Peter, Pearsion Education, *Machine Vision Toolbox*, Enero - 2015, https://petercorke.com/