Procesamiento de Imágenes

Trabajo práctico N°3 - Informe.

Dimenna, Valentín : D-43366/4

Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Tec. Universitaria en Inteligencia Artificial
2024

Índice

Introducción

Este informe corresponde al trabajo práctico n°3 de la asignatura Procesamiento de Imágenes I de la carrera Tecnicatura Universitaria en Inteligencia Artificial.
El trabajo consta de dos ejercicios: “Detección de dados en video” y “Detección de patentes”.
En este informe se explica cómo se encaró cada ejercicio, con imágenes paso a paso y comentarios u observaciones.

Ejercicio 1: Clasificación de Dados.

Planteo del problema:

Tengo 4 videos con tiradas de 5 dados cada una, el problema a resolver es detectar cuántos puntos se obtienen en cada tirada. Para cada video debe obtenerse cuando se detienen los dados enmarcarlos en color azul, escribir cuanto dio cada dado y finalmente crear un video de cada tirada con los dados enmarcados y los números escritos. Durante el informe se mostrarán los ejemplos basados en el video de la tirada Nro 1.

Solución:

Lectura de videos:

Para cada uno de los videos se leen cada uno de los frames y se guardan en 3 listas, frames_hsv,frames,frames_gr donde se guarda cada frame en formato hsv,rgb y escala de grises respectivamente. Esto supone un problema si el programa se quisiera integrar el programa en algún proyecto que quiera hacer el procesamiento en tiempo real ya que en este trabajo se guardan todos los frames en listas antes de empezar a procesarlas, para este trabajo será suficiente.

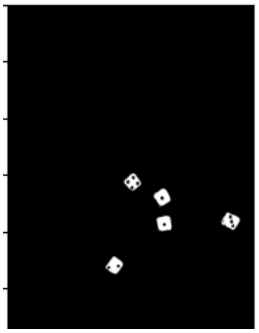
Delimitar región de interés:

En cada video nos centraremos únicamente en la zona de la cartulina verde por lo que como primer paso voy a delimitar esta región para ignorar cualquier cambio que suceda fuera de esta.

En la primer imagen vemos el frame inicial del video, ussaremos la clara diferencia de color entre la cartulina y el resto del frame para poder segmentar esta region y todo lo que no este en el rango del color de esta region se pondra en negro (0,0,0). Voy a suponer que durante el transcurso del video no se mueven ni la cartulina ni la cámara, por lo tanto esta máscara solo se calculará en el primer frame de cada video y se tomará como que siempre está en la misma posición. Esto sugiere algunas limitaciones ya que si se quisiera usar este mismo trabajo para otros videos donde si se mueve esta cartulina no sería de utilidad pero para este caso será suficiente.

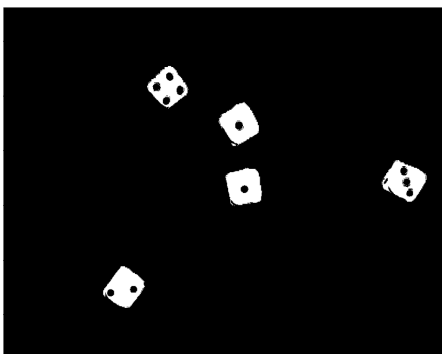
Movimiento de dados:

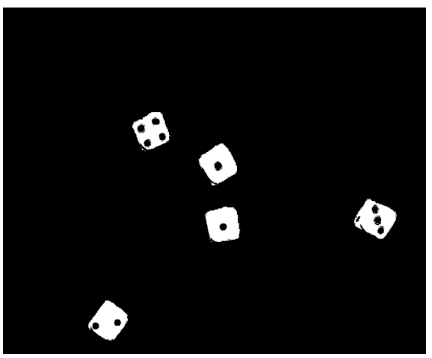
Para detectar si hay movimiento o no voy a usar la diferencia absoluta entre frames consecutivos. Creó una máscara (similar a la usada para para segmentar la cartulina) para segmentar los dados que puede o no haber dentro de la región de interés ya que al ser rojos se diferencian del fondo verde.
Para cada frame de cada video se calculará una máscara para detectar los dados (Rojos) para ese frame y el siguiente, se hace una operación AND con la máscara de la región para descartar cualquier ruido fuera de la zona de interés, después de esto se calcula la diferencia entre ambos (cv2.absdiff(frame1,frame2)), si esta abajo de un umbral entre un frame y el otro significa que no hay movimiento por lo que guardo el índice de la lista en ese frame, una vez detectado cuando se frena revisamos si la diferencia vuelve a superar el umbral por algunos frames consecutivos representa que vuelve a haber movimiento (puede ser cuando el tirador saca los dados) y guardo también este índice.

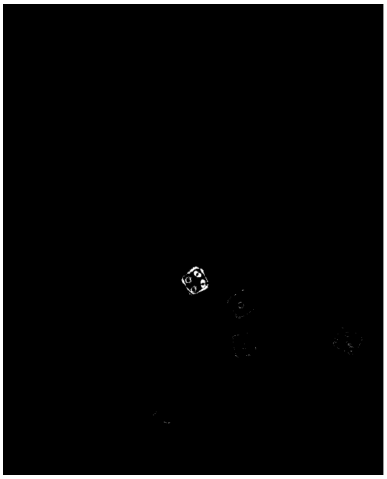


Máscara de los dados una vez detectado el freno de los dados.

Máscara de 2 frames consecutivos los cuales se comparan para detectar movimiento



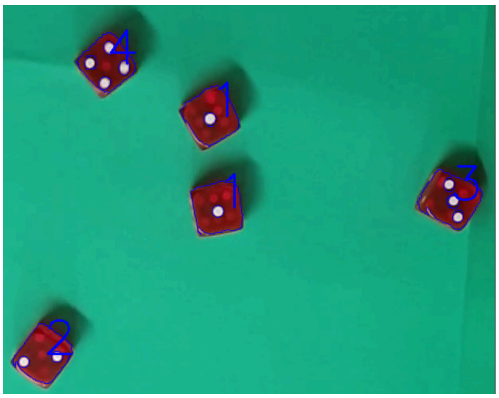




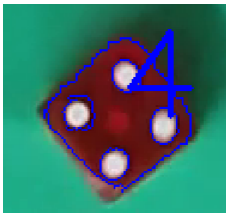
Diferencia entre los 2 frames consecutivos mostrados cuando se están frenando los dados (pueden estar rotando todavía mientras frena)

Contornos:

Teniendo los índices en que se frena y vuelve a haber movimiento en el video, en todos los frames fuera del rango descrito por esos índices no realizó más procesamiento y esos frames van directamente al video de salida. Los frames dentro de ese rango se encuentran los contornos de cada uno, se filtra por área y se calcula la cantidad de hijos que tiene cada padre, en este caso los padres representan los dados y los hijos los puntos de cada dado, se pintan de azul los contornos, se dibuja en el frame los contornos en azul, se dibuja el valor de cada dado y se agregan los frames al video de salida.



Ejemplo de un Frame de salida de los dados frenados.



Conclusión:

Como se fue diciendo en partes del informe el programa no es robusto frente a algunas situaciones ya explicadas, para los videos proporcionados la respuesta resulta suficiente. Algunos cambios posibles para mejorar este programa se puede empezar haciendo el procesamiento a la vez que se leen los frames del video y guardar pocos frames en memoria para poder verificar si hay movimiento o no. Podría calcularse la máscara de la región de interés en cada frame para evitar errores por posibles movimientos de cámara o cartulina. Por otro lado, se podría intentar mejorar el preprocesamiento antes de encontrar contornos en los frames detectados como quietos para mejorar la detección.

Ejercicio 2: Detección de patentes.

Planteo de problema:

Tengo 12 fotos de autos diferentes. El problema a resolver es obtener los 6 caracteres de cada patente de cada foto. Para cada foto debe recortarse los 6 caracteres y mostrarlos en orden.

Solución:

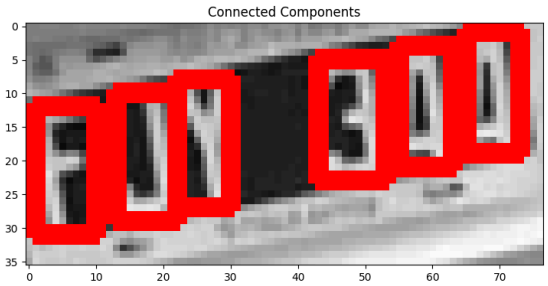
Para resolver este problema se trabajó con las imágenes en escala de grises ya que al ser autos diferentes y de diferentes colores no se pueden utilizar para sacar información ya que difieren mucho entre una imagen y otra.

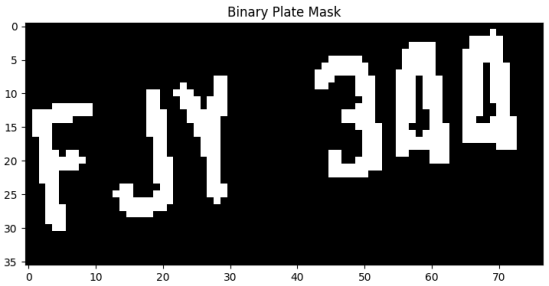
Imagen en escala de grises

Imagen Umbralizada

Comencé leyendo las imágenes en escala de grises y umbralizado usando threshold adaptativo ya que las imágenes presentaban diferentes niveles intensidad lo que produce que un threshold fijo no sea muy eficiente para luego segmentar correctamente la imagen.

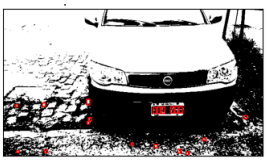
Se encontraron los contornos en las imágenes umbralizadas y se filtraron por área,alto (h),ancho (w) y aspect_ratio (w/h) ya que las patentes argentinas estan normalizadas puedo usar esta información para buscar contornos que tengan forma parecida, las patentes que encontramos en estas fotos son todas del formato antiguo (3 letras y 3 números) y miden 294 x 129mm. Basado en esto guardo las regiones cortadas por esos contornos que son posibles patentes y aquellos que no cumplen estas especificaciones son descartados.



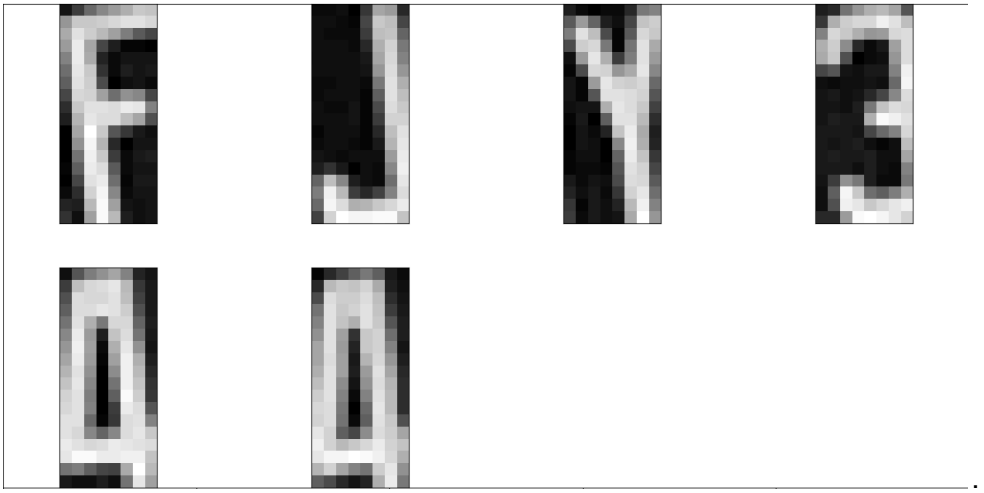


A cada una de las regiones detectadas se las binariza y se encuentran los contornos, nuevamente se filtran por alto,ancho y área. todas las regiones que pasen los filtros se guardan en una lista de componentes válidos. Sabiendo que una patente tiene 6 caracteres en total podemos usar esto para filtrar si la patente fue detectada correctamente o no, por lo que en un diccionario se guardaron pares {nro_imagen_auto : [region_valida]} donde cada región válida representa un carácter distinto. Una vez procesadas todas las imágenes toda clave que tenga un largo (len(dict[nro_imagen_auto]) menor a 6 se toma como fallo y se guarda en una lista para procesarlas nuevamente.

Para cada una de las imágenes que no se encontraron 6 dígitos se procesan con otro método, esta vez se usó un threshold ya que se supuso que aquellas imágenes que no pudieron ser correctamente segmentadas tendrían valores más similares y habria menores problemas al tener una cantidad de imágenes más baja.
Se encuentran los contornos de esas imágenes y se filtran por área, alto y ancho y se guardaron todas las regiones posibles nuevamente en una lista.



En esta imagen se ven de izquierda a derecha la imagen con los contornos dibujados, la imagen umbralizada sobre la que se buscan las componentes conectadas, las componentes conectadas filtradas y por último todas las componentes conectadas encontradas en la imagen.
Para todas las imágenes que se encontraron menos de 6 contornos fueron tomadas como fallo y descartadas ya que no se lograron segmentar todos los caracteres. Aquellas imágenes para las que se obtuvieron más de 6 contornos, se conservan sólo 6 de esos contornos que estuvieran cerca entre ellos, es decir, buscamos un grupo de 6 contornos donde las coordenadas (x1,y1) del primero y las coordenadas (x2,y2) difieran en pequeñas cifras. Por último se procedió a mostrar todas las regiones de las imágenes de las cuales se encontraron 6 caracteres y de las que no se informó por pantalla



Resultado final.

Conclusión:

Se segmentan correctamente los seis caracteres de ocho patentes y una presenta dos errores, las tres restantes no se logran segmentar. Podría realizarse otra iteración cambiando nuevamente los parámetros y/o métodos de umbralizado, búsqueda de contornos, filtros,etc. para intentar encontrar los 3 restantes y mejorar el resto y que quede un programa más robusto frente a nuevas fotos.