Tarea 3 CC5508: Detección de Objetos

Javier Morales

15 de Noviembre del 2020

1 Resumen

En el siguiente informe se busca implementar y entender un algoritmo de detección de objetos para encontrar círculos amarillos en imágenes que contengan varias figuras geométricas de distintos colores. El algoritmo se basa en el manejo los canales RGB de la imagen y a la obtención de componentes conexos, pero también se necesita determinar algunos valores de a través de ensayo y error. Finalmente se concluye que el algoritmo funciona muy bien con el conjunto de imágenes utilizados pero puede dar malos resultados en imágenes con mucho ruido y fondos poco claros.

2 Introducción

La detección de objetos en una imagen es un problema que puede aparecer en una gran cantidad de situaciones reales, por ejemplo, si uno quisiera reconocer rostros de personas en una imagen el primer paso sería identificar donde se encuentran los rostros. El problema puede consistir en solo encontrar siempre la misma figura, pero puede complicarse si se busca ser indiferente a transformaciones como rotaciones o simetrías. También importa que tan complicado es el objeto a encontrar, es muy distinto buscar figuras geométricas a buscar personas, y la forma de abordar estos problemas puede ser muy distinta.

En este informe se busca resolver el problema de encontrar círculos amarillos en una imagen, para enfrentar esto se usaran varias técnicas del curso en conjunto como: *threshold*, componentes conexos y topología digital. El dataset de imágenes a utilizar esta construido específicamente para el problema, y posee varias figuras distintas y de un solo color en un fondo blanco. El resultado del problema consiste de la imagen de entrada con los círculos amarillos envueltos en la caja más pequeña que los contenga.

3 Desarrollo

El programa `yellow_circles.py` recibe una imagen de entrada y trata de encontrar todas las figuras con contornos circulares y que sean de color amarillo dentro de esta, el programa se ejecuta con la siguiente linea:

```
python yellow_circles.py [path to image]
```

Al terminar la ejecución se muestra la imagen de entrada con los objetos circulares amarillos encerrados en rectángulos rojos, además se guarda una imagen binaria que muestra todos los pixeles amarillos y un archivo de texto con la descripción de los componentes conexos encontrados. Los pasos que realiza el código son descritos a continuación.

3.1 Detección de pixeles amarillos

El primer paso consiste en poder discriminar cuales pixeles son de color amarillo, para esto se manipularan los canales de colores del modelo RGB, pues este es el más común. En el modelo RGB el color amarillo se obtiene con altos valores de rojo y verde, pero como los blancos también poseen altos rojos y verdes también es necesario que el valor de azules sea bajo. A través de evidencia empírica se obtiene la siguiente formula:

$$YELLOW = 2 * RED + 3 * GREEN - 4 * BLUE$$

Al aplicar la formula anterior se obtiene una nueva imagen en escala de grises (aunque hay que reescalar los valores entre 0 y 255) con valores altos en los pixeles "más amarillos" y valores bajos en el resto, luego basta realizar una operación de *threshold* para obtener una imagen binaria con los pixeles amarillos.

3.2 Obtención de componentes conexos

Una vez se tiene una imagen binaria se procede a utilizar el algoritmo estándar para detección componentes conexos, con la regla de "si dos pixeles adyacentes tienen valor 1, pertenecen al mismo componente". El algoritmo puede entregar algunos componentes que corresponden a pixeles amarillos que no necesariamente son parte de un objeto amarillo en la imagen, por lo que se deben filtrar los componentes que sean muy pequeños, filtrar los componentes de menos de 50 pixeles muestra buenos resultados.

3.3 Detección de componentes circulares

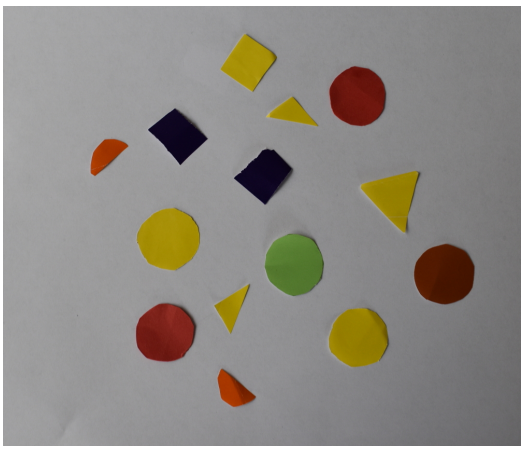
Para determinar si un componente conexo es circular se realiza el siguiente algoritmo:

1. Se obtiene el arreglo de bordes del componente conexo con el algoritmo de topología digital
2. Se definen 3 puntos del borde lo suficientemente espaciados, ubicados aproximadamente en las posiciones $0.33 \cdot len(array)$, $0.66 \cdot len(array)$ y $0.99 \cdot len(array)$ del arreglo de bordes
3. Se obtiene la formula del circulo que pasa por los 3 puntos, resolviendo D, E y F en la siguiente ecuación lineal: $x^2 + y^2 + Dx + Ey + F = 0$

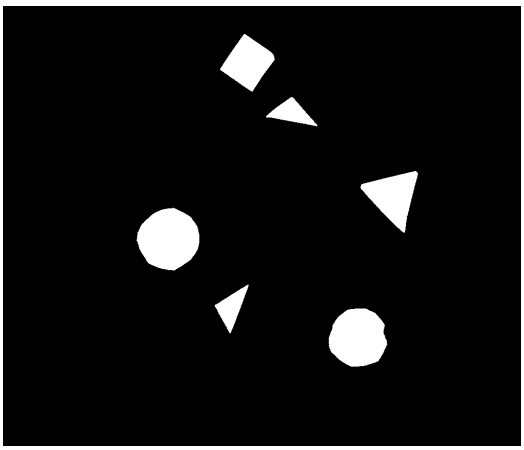
4. Se calcula el centro del circulo $(-D/2, -E/2)$ y su radio $\sqrt{(D^2+E^2)/4-F}$
5. Por cada punto del borde se calcula su radio hacia el centro del circulo y se calcula su diferencia con el radio del circulo
6. Se retorna el error promedio entre los radios calculados y el radio del circulo

El error entregado por este algoritmo es muy cercano cero cuando el componente es perfectamente circular, y su valor aumenta mientras menos circular es. Luego basta definir un error de corte para definir que objetos sean reconocidos como círculos para el programa, a través de prueba y error se decide que 0.1 parece ser un buen valor de corte.

4 Resultados y discusión



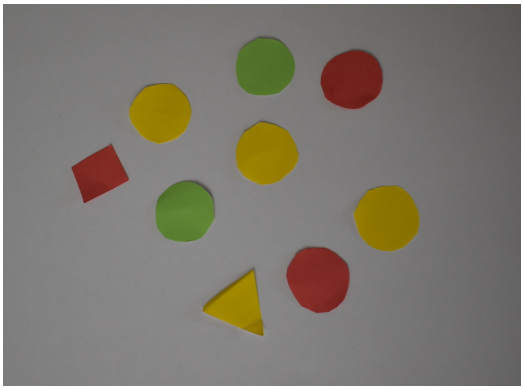
(a) Entrada



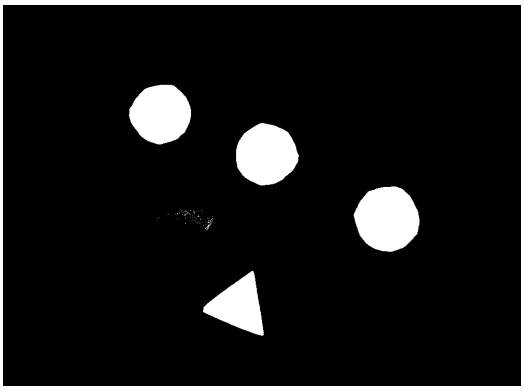
(b) Binario con pixeles amarillos

Resultado 1: Resultado de obtener pixeles amarillos

Se puede notar que con la luz adecuada la ecuación definida para reconocer amarillos muestra muy buenos resultados, reconoce solo los componentes amarillos sin recortarlos ni con puntos negros desntro de estos.



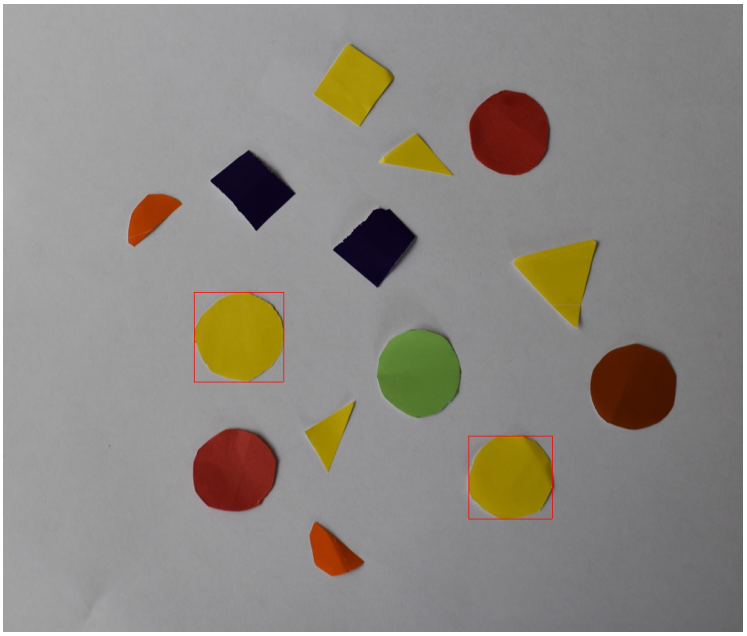
(a) Entrada



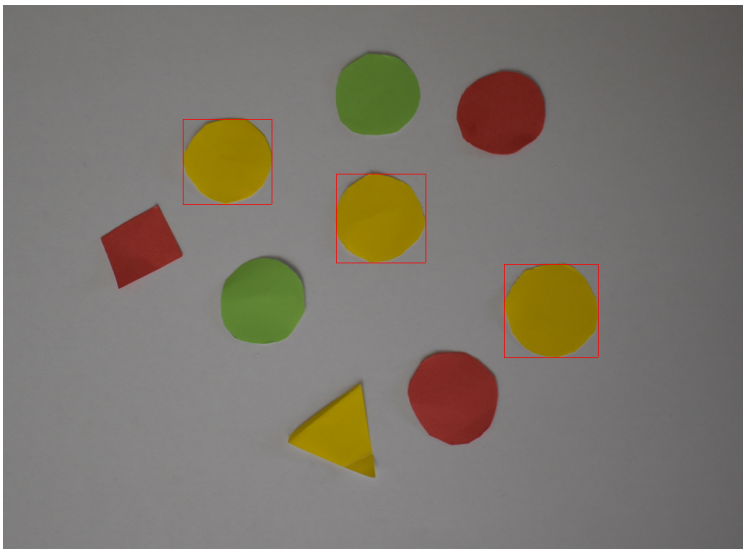
(b) Binario con pixeles amarillos

Resultado 2: Resultado de obtener pixeles amarillos

En este caso se puede ver que reconoce parte de un circulo verde como amarillo, en particular los puntos que están más iluminados, pero estos pixeles no alcanzan a formar un componente conexo muy grande por lo que serán eliminados al filtrar los componentes por tamaño. También, como se ve en el triangulo inferior, el programa puede reconocer amarillos más oscuros como parte del objeto.



Resultado 3: Resultado de obtener pixeles amarillos



Resultado 4: Resultado de obtener pixeles amarillos

En los resultados 3 y 4 se puede ver que el filtro de error a un circulo y el de tamaño de componentes remueven correctamente los pixeles amarillos que no cumplen con los criterios.

5 Conclusiones

El algoritmo funciona sin problemas para el dataset utilizado, es capaz de descartar las figuras que no sean amarillas y las que no sean circulares en todos los casos. Sin embargo, el algoritmo depende de 3 parámetros: la formula para resaltar amarillos, el valor de corte para el tamaño de los componentes y el valor de corte para el error hacia un circulo. Estos valores fueron optimizados probando valores manualmente con el dataset, y deben ser ajustados si se cambia la iluminación o el tamaño de la imagen, por lo queda como trabajo propuesto el encontrar una forma de automatizar la optimización de estos valores según la imagen de entrada.