

Universidad del Valle De Guatemala

Facultad de Ingeniería

Computación Paralela



Proyecto 3 - Programación Híbrida con CUDA

Javier Valle 20159
Mario de León 19019

Guatemala, 5 de noviembre 2023

Transformada de Hough

La transformación de Hough es una técnica bastante conocida que sirve para poder detectar cualquier forma, siempre y cuando esta forma se pueda representar de manera matemática. Este algoritmo es capaz de detectar una forma incluso si ésta está rota o un poco rota o distorsionada.

El algoritmo de la transformación de Hough puede llegar a necesitar mucha computación, incluso si se está analizando una línea con dos argumentos. La transformada probabilística de Hough es una optimización de la Transformada de Hough. Esta optimización lo que hace es no tomar todos los puntos en consideración, a cambio toma un solo subconjunto de puntos al azar y eso es suficiente para la detección de líneas. Para lo anterior solo se necesitaría disminuir el umbral. (Sosa-Costa, 2019)

Esta técnica también es altamente utilizada para los autos autónomos y se usa principalmente para poder realizar recorridos estables y con bastante seguridad. Para ello, lo que se hace es que por medio de las imágenes que van capturando las cámaras de un carro, va encontrando la línea más recta que mejor se adapte para cada par de puntos dado un plano cartesiano en cuestión. (Méndez, 2021)

Por otro lado, es de alta importancia mencionar que la Transformada de Hough es de gran utilidad para poder detectar glóbulos rojos. Esta aplicación es de bastante utilidad, dado que en muchas ocasiones se dan muestras con cantidades erróneas de glóbulos rojos; con esta aplicación se logra prevenir el riesgo de padecer al 100% una enfermedad y se puede iniciar su tratamiento en una época temprana. (Rodríguez Espinoza et al., 2017)

Explicación de las operaciones realizadas con “xCoord” y “yCoord”

Las operaciones “xCoord” y “yCoord” están calculando las coordenadas relativas de un pixel actual en cuestión con respecto a la imagen que se está analizando. El anterior cálculo se hace para convertir las coordenadas de pixel a coordenadas cartesianas relativas al centro de la imagen.

En cuanto a los cálculos específicos, se hace lo siguiente:

- **“xCoord”**: su resultado se obtiene de la resta de la coordenada horizontal del pixel “i” en el centro de la imagen “xCent”, obteniendo así como resultado la distancia horizontal del pixel actual con respecto al centro. Este cálculo se hace para medir la distancia horizontal del pixel al centro de la imagen, lo cual es necesario para calcular la distancia desde el centro al pixel en coordenadas polares.
- **“yCoord”**: el cálculo de yCoord se hace casi igual que xCoord, con la diferencia que se calcula mediante la resta del centro de la imagen “yCent” con la coordenada vertical del pixel “j”. Lo anterior se hace para poder medir la distancia vertical del

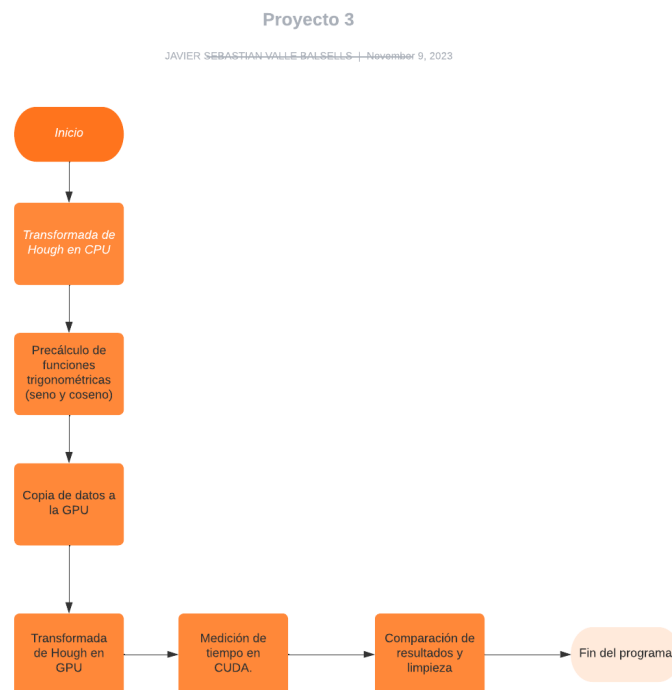
pixel al centro de la imagen, lo cual es necesario para poder calcular la distancia desde el centro al pixel en coordenadas.

Uso de memoria constante

En el código que usa memoria constante, lo que se hace es usar dos variables para guardar los valores que se calculan para las funciones trigonométricas seno y coseno. El valor de estas funciones se calculan una única vez en la CPU y se copian a la memoria constante en el dispositivo CUDA mediante las funciones de “`cudaMemcpyToSymbol`”.

Asimismo, en el código las constantes “`d_Cos`” y “`d_Sin`” son arreglos de tipo float declarados como memoria constante en el espacio global del dispositivo que se está usando. Los arreglos anteriormente mencionados se guardan los valores precalculados de seno y coseno para diferentes ángulos de una imagen proporcionada y son utilizados por los hilos en el kernel “`GPU_HoughTranShared`” para evitar cálculos redundantes y mejorar el trabajo del algoritmo.

Diagrama funcional



Registro de tiempo con la memoria global

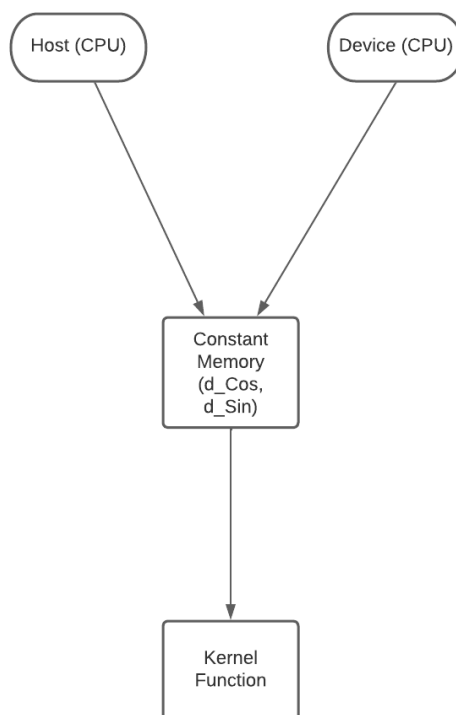
```
!./houghGlobal runaway.pgm  
  
Done!  
Tiempo transcurrido: 0.024768 ms
```

Registro de tiempo con la memoria constante

```
[7] !./houghConstant runaway.pgm  
  
Done!  
Tiempo transcurrido: 0.004768 ms
```

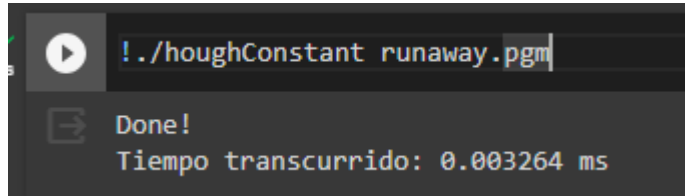
Explicación de lo que se cambió en el código con memoria global

En la versión de memoria global, se aplicó la memoria constante para almacenar las tablas de valores que ya se calcularon en coseno (d_Cos) y seno (d_Sin) de los ángulos a evaluar en el programa con respecto a la imagen. Los valores de d_Cos y d_Sin se usan intensamente dentro del kernel para evitar cálculos repetitivos costosos de funciones trigonométricas. Las variables en memoria constante mencionadas anteriormente se declararon fuera de la función main con el modificador `__constant__`. Luego, se transfirieron los valores precalculados desde el host a la memoria constante en el dispositivo usado `cudaMemcpyToSymbol`.



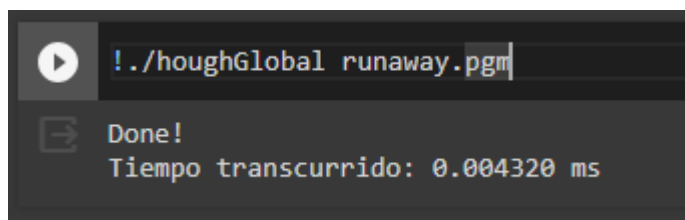
Compilación de las 3 versiones (con bitácoras)

Registro de memoria global



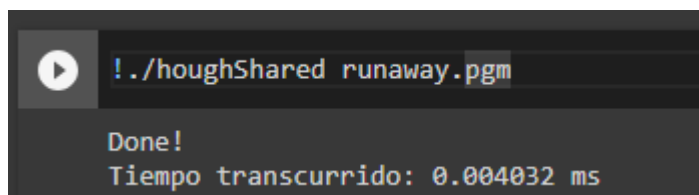
```
!./houghConstant runaway.pgm
Done!
Tiempo transcurrido: 0.003264 ms
```

Registro de memoria constante



```
!./houghGlobal runaway.pgm
Done!
Tiempo transcurrido: 0.004320 ms
```

Registro de memoria compartida

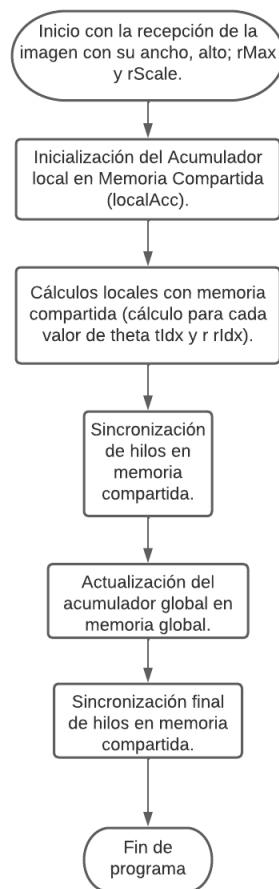


```
!./houghShared runaway.pgm
Done!
Tiempo transcurrido: 0.004032 ms
```

Descripción de los cambios realizados a la Transformada de Hough

En el nuevo código de la Transformada de Hough, lo que se hizo fue utilizar memoria compartida, implementando así una estrategia de optimización para mejorar el rendimiento reduciendo así el acceso a la memoria global, lo cual puede ser un poco más lento. La memoria compartida se usa para almacenar un acumulador local (el cual es localAcc) que se comparte entre los hilos de un bloque. Este acumulador local se usa para realizar cálculos intermedios y así reducir las operaciones de escritura y lectura en la memoria global.

Diagrama de flujo



Bitácora de cada versión del programa

Corrida con la versión global

```
!./houghGlobal runaway.pgm

Done!
Tiempo transcurrido: 0.004064 ms
```

A terminal window showing the execution of the command `!./houghGlobal runaway.pgm`. The output is `Done!` followed by `Tiempo transcurrido: 0.004064 ms`.

```
!./houghGlobal runaway.pgm

Done!
Tiempo transcurrido: 0.004096 ms
```

A terminal window showing the execution of the command `!./houghGlobal runaway.pgm`. The output is `Done!` followed by `Tiempo transcurrido: 0.004096 ms`.

```
▶ !./houghGlobal runaway.pgm
```

Done!
Tiempo transcurrido: 0.002880 ms

```
▶ !./houghGlobal runaway.pgm
```

Done!
Tiempo transcurrido: 0.003936 ms

```
▶ !./houghGlobal runaway.pgm
```

Done!
Tiempo transcurrido: 0.004128 ms

```
▶ !./houghGlobal runaway.pgm
```

Done!
Tiempo transcurrido: 0.003456 ms

```
▶ !./houghGlobal runaway.pgm
```

Done!
Tiempo transcurrido: 0.003648 ms

```
▶ !./houghGlobal runaway.pgm
```

Done!
Tiempo transcurrido: 0.003520 ms

```
▶ !./houghGlobal runaway.pgm
```

Done!
Tiempo transcurrido: 0.003520 ms

```
▶ !./houghGlobal runaway.pgm
```

Done!
Tiempo transcurrido: 0.003904 ms

Corrida con la versión contante y global

```
✓ [54] !./houghGlobal runaway.pgm
0s Done!
    Tiempo transcurrido: 0.004128 ms

✓ [55] !./houghConstant runaway.pgm
0s Done!
    Tiempo transcurrido: 0.003936 ms
```

```
✓ [56] !./houghGlobal runaway.pgm
0s Done!
    Tiempo transcurrido: 0.004000 ms

✓ [57] !./houghConstant runaway.pgm
0s Done!
    Tiempo transcurrido: 0.005472 ms
```

```
[58] !./houghGlobal runaway.pgm
Done!
    Tiempo transcurrido: 0.004096 ms

[59] !./houghConstant runaway.pgm
Done!
    Tiempo transcurrido: 0.002816 ms
```

```
✓ [60] !./houghGlobal runaway.pgm
0s Done!
    Tiempo transcurrido: 0.003584 ms

✓ [61] !./houghConstant runaway.pgm
0s Done!
    Tiempo transcurrido: 0.002112 ms
```


[62] !./houghGlobal runaway.pgm

Done!
Tiempo transcurrido: 0.003776 ms



!./houghConstant runaway.pgm

Done!
Tiempo transcurrido: 0.003328 ms

[64] !./houghGlobal runaway.pgm

Done!
Tiempo transcurrido: 0.008288 ms



!./houghConstant runaway.pgm

Done!
Tiempo transcurrido: 0.002784 ms

[66] !./houghGlobal runaway.pgm

Done!
Tiempo transcurrido: 0.002464 ms



!./houghConstant runaway.pgm

Done!
Tiempo transcurrido: 0.002304 ms

[68] !./houghGlobal runaway.pgm

Done!
Tiempo transcurrido: 0.003520 ms



!./houghConstant runaway.pgm

Done!
Tiempo transcurrido: 0.002752 ms

```
[70] !./houghGlobal runaway.pgm

Done!
Tiempo transcurrido: 0.004096 ms

!./houghConstant runaway.pgm

Done!
Tiempo transcurrido: 0.004032 ms
```

```
[72] !./houghGlobal runaway.pgm

Done!
Tiempo transcurrido: 0.003520 ms

!./houghConstant runaway.pgm

Done!
Tiempo transcurrido: 0.003744 ms
```

Corrida con las tres versiones

```
✓ 0s !./houghConstant runaway.pgm

Done!
Tiempo transcurrido: 0.004096 ms

✓ 0s [15] !./houghGlobal runaway.pgm

Done!
Tiempo transcurrido: 0.002336 ms

✓ 1s [16] !./houghShared runaway.pgm


Done!
Tiempo transcurrido: 0.003744 ms
```

```
[17] !./houghConstant runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.004064 ms
```

```
[18] !./houghGlobal runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.002624 ms
```

```
 !./houghShared runaway.pgm
```


```
Done!  
Tiempo transcurrido: 0.004128 ms
```

```
[20] !./houghConstant runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.003328 ms
```

```
[21] !./houghGlobal runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.003520 ms
```

```
 !./houghShared runaway.pgm
```


```
Done!  
Tiempo transcurrido: 0.004096 ms
```

```
[23] !./houghConstant runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.002368 ms
```

```
[24] !./houghGlobal runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.004096 ms
```

```
 !./houghShared runaway.pgm
```


```
Done!  
Tiempo transcurrido: 0.003808 ms
```

```
[23] !./houghConstant runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.002368 ms
```

```
[24] !./houghGlobal runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.004096 ms
```

```
 !./houghShared runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.003808 ms
```

```
[29] !./houghConstant runaway.pgm
```

Done!

Tiempo transcurrido: 0.003552 ms

```
[30] !./houghGlobal runaway.pgm
```

Done!

Tiempo transcurrido: 0.002912 ms



```
!./houghShared runaway.pgm
```

Done!

Tiempo transcurrido: 0.003712 ms

```
[32] !./houghConstant runaway.pgm
```

Done!

Tiempo transcurrido: 0.004128 ms

```
[33] !./houghGlobal runaway.pgm
```

Done!

Tiempo transcurrido: 0.003680 ms



```
!./houghShared runaway.pgm
```

Done!

Tiempo transcurrido: 0.002240 ms

```
[35] !./houghConstant runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.002368 ms
```

```
[36] !./houghGlobal runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.003328 ms
```



```
!./houghShared runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.003072 ms
```

```
[38] !./houghConstant runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.004000 ms
```

```
[39] !./houghGlobal runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.003616 ms
```



```
!./houghShared runaway.pgm
```

```
Done!  
Tiempo transcurrido: 0.003456 ms
```

```
[41] !./houghConstant runaway.pgm

Done!
Tiempo transcurrido: 0.005536 ms

[42] !./houghGlobal runaway.pgm

Done!
Tiempo transcurrido: 0.004128 ms

[43] !./houghShared runaway.pgm

Done!
Tiempo transcurrido: 0.002432 ms
```

Conclusiones

1. La versión que mejor rendimiento presentó fue la que contenía la versión compartida, dado que no tenía que estar haciendo comparaciones en ninguna parte.
2. El uso de la memoria compartida permitió la reducción del acceso a la memoria global, lo cual mejoró significativamente la eficiencia del acceso a la memoria.
3. La ventaja de tener una memoria constante es que el cálculo del seno y el coseno no genera cálculos redundantes y mejora el rendimiento en las operaciones trigonométricas.
4. El hecho de que se haya implementado una versión probabilística de la Transformada de Hough permitió optimizar el rendimiento del código, tomando así un subconjunto aleatorio de puntos con el fin de reducir la carga computacional para poder acelerar la detección de líneas.

Referencias

1. Sosa-Costa, A. (2019, August 30). *La transformada de línea de Hough* - ▷ *Cursos de Programación de 0 a Experto* © Garantizados. ▷ *Cursos De Programación De 0 a Experto* © Garantizados. <https://unipython.com/la-transformada-linea-hough/>
2. Méndez, I. M. R. (2021). DETECCIÓN DE LÍNEAS CON TRANSFORMADA DE HOUGH. *Uat-mx*. https://www.academia.edu/44827397/DETECCI%C3%93N_DE_L%C3%8DNEAS_CON_TRANSFORMADA_DE_HOUGH

3. Rodríguez Espinoza, M., López Rubio, E., & Molina Cabello, M. (2017, June). Detección automática de glóbulos rojos mediante la Transformada de Hough. *Riuma*. Retrieved November 13, 2023, from [https://riuma.uma.es/xmlui/bitstream/handle/10630/15410/Mariajesusrodriguezepino saMemoria.pdf?sequence=1&isAllowed=y](https://riuma.uma.es/xmlui/bitstream/handle/10630/15410/Mariajesusrodriguezepino%20saMemoria.pdf?sequence=1&isAllowed=y)