

## **Detección de rectas a través la transformada de Hough implementación en PHP**

El objetivo de esto es simple, necesitamos reconocer formas a partir de simples líneas, cuadrados, rectángulos, triángulos y hasta círculos. Esto es un reto para los métodos de inteligencia artificial, desde hace muchos años estos métodos ya estaban en teoría desarrollados pero por las limitaciones de los computadores, velocidad de procesamiento, memoria entre otras cosas era muy pesado o poco probable desarrollar exitosamente estas técnicas.

### **Transformada de Hough**

La transformada de Hough consiste en permitir descubrir las formas en una imagen. Se basa en transformar puntos de la imagen en un espacio de parámetros. La idea es encontrar rectas, como se menciono en teoría se pueden encontrar varias formas, pero el costo computacional es elevado.

La aplicación mas simple de la transformada de hough es para la detección de rectas, lo primero es tener una imagen binarizada, resultado de la detección de bordes. Para cada punto  $x, y$  de la imagen que estemos utilizando pasan infinitas rectas de la forma:  $y=ax+b$ , la ecuación de la recta. Luego cada punto  $x, y$  vota para cada pareja de puntos que satisface  $b=y-ax$ .

El algoritmo tiene que recorrer todos los puntos  $x, y$ , y para cada uno votar en el espacio de parámetros  $m, b$  o también llamado acumulador. Luego la pareja  $m, b$  con más votos o el máximo del acumulador da la ecuación de la recta.

Hasta aquí hemos llegado de manera sencilla con nuestro algoritmo, pero existe un problema, los valores de  $m$  tienden al infinito, por lo que almacenarlos y graficarlos será un impedimento para el uso de la transformada.

La solución planteada es la siguiente, si bien  $y = ax + b$ , se dice que toda ecuación puede ser representada en senos y cósenos es así que la ecuación de la recta de la forma  $y = ax + b$ , puede ser representada como:

$$p = x\cos(\theta) + y\sin(\theta)$$

Esta es la llamada ecuación en la forma normal de la recta, donde  $p$  es la distancia de la recta al origen y  $\theta$  es el ángulo entre la perpendicular y el eje  $x$ , de esta forma son menos los puntos que hay que recorrer y por lo tanto más rápido es el algoritmo.

Los límites de estos están dados por las siguientes condiciones,  $\theta$  varia entre 0 y 180 ya que a partir de 180 hasta 360 grados se vuelven a cruzar las curvas y  $p$  varia en la diagonal de la imagen es decir la hipotenusa con respecto a los lados de la imagen original.

## **Implementación en el lenguaje PHP**

Si bien PHP esta orientado para el desarrollo de páginas dinámicas para entorno Web, eso no le quita potencialidad a la hora de usarlo como lenguaje de desarrollo para algoritmos complejos, ya que la gran facilidad que nos brinda en la manipulación de los tipos de datos lo hacen flexible a estas situaciones

PHP usa la librería GD para manipulación de imágenes, los pasos que seguiremos en la implementación son los siguientes:

- Implementación de un frontend simple
- Detección de bordes, aplicando el filtro Sobel.
- Binarización de las líneas en contraste con el fondo.
- Implementación del acumulador.
- Graficar el acumulador en una nueva imagen.

La interfaz será sencilla, constara de un formulario base donde podamos subir una imagen y cajas de entrada para la personalización del método a emplear:

## INTELIGENCIA ARTIFICIAL

Ingresar imagen

Seleccionar imagen:

Binarizacion

Horizonte:

Deteccion de bordes

Transformada de hough

Proporcion:

Proporcion Y:

Estirar X:

Estirar Y:

Intensidad:

Fondo:  claro  oscuro

Al momento de subir la imagen, la convertiremos dinámicamente al formato PNG de acuerdo a su actual extensión

```
switch ($ext) {  
    case 'jpeg':  
        case 'jpg': $imEntrada = imagecreatefromjpeg($archivo['tmp_name']);  
        break;  
        case 'gif': $imEntrada = imagecreatefromgif($archivo['tmp_name']);  
        break;  
        case 'png': $imEntrada = imagecreatefromgif($archivo['tmp_name']);  
        break;  
}  
  
imagepng($imEntrada,$this->IMG_ENTADA);
```

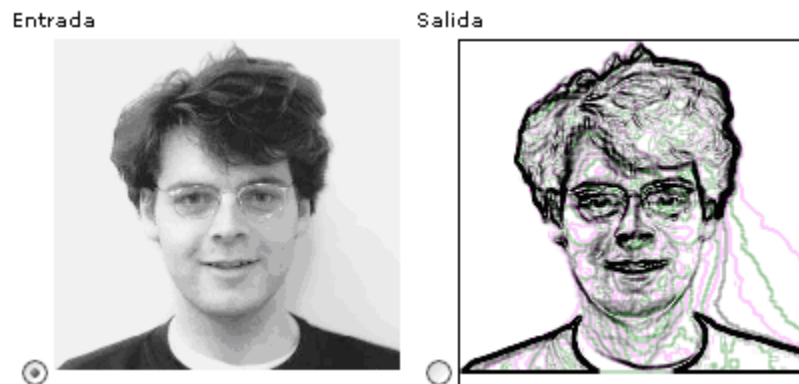
Donde `this->IMG_ENTADA` es el nombre de entrada que definimos previamente junto con el nombre de salida de las imágenes

```

$tratamiento = new TratamientoImagen();
$tratamiento->IMG_ENTADA = "entrada.png";
$tratamiento->IMG_SALIDA = "salida.png";

```

Necesitamos detectar los bordes de la imagen, en este resumen no incluiré la forma como se desarrollara la misma, en el proyecto presentado se uso el filtro Sobel



Ahora podemos aplicar la transformada de Hough:

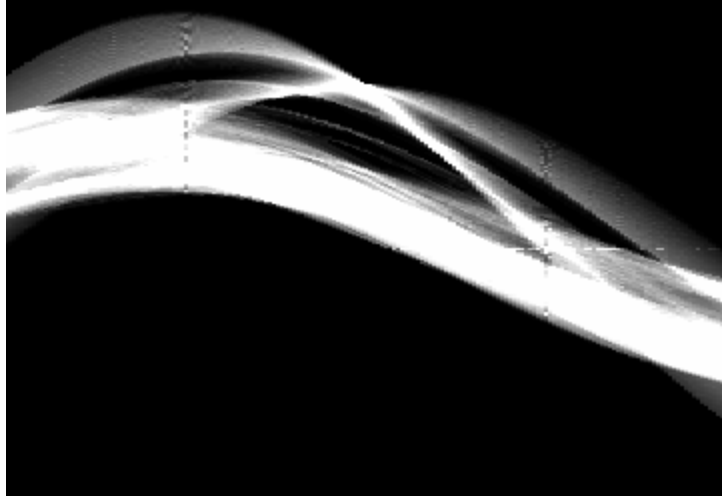
```

for ($y=0; $y<$salto; $y++) {
    for ($x=0; $x<$ancho; $x++) {
        if ($matrImag[$y][$x] == 1) {
            for ($theta=0; $theta<$maximoTheta; $theta+=1) {
                $r = $x * cos(deg2rad($theta)) + $y * sin(deg2rad($theta));

                if (!isset($data[(int)$theta][(int)$r])) {
                    $data[(int)$theta][(int)$r] = 1;
                } else {
                    $data[(int)$theta][(int)$r] ++;
                }
            }
        }
    }
}

```

Donde la matriz \$data es nuestro acumulador, finalmente graficamos los votos recibidos en esta matriz



Christian Huamaní Gutiérrez - MPIA