Visión por computador

Práctica 3

Arón Collados
Cristian Román

Gx, Gy, módulo y dirección

Para calcular el gradiente horizontal, vertical, el módulo y la orientación del gradiente se ha usado el operador de sobel con un filtro gaussiano.

Primero se aplica el filtro gaussiano sobre la imagen original.

Filtro Gaussiano 3x3

```
GaussianBlur(bgrMap, bgrMap, Size(3, 3), 0, 0, BORDER_DEFAULT);
```

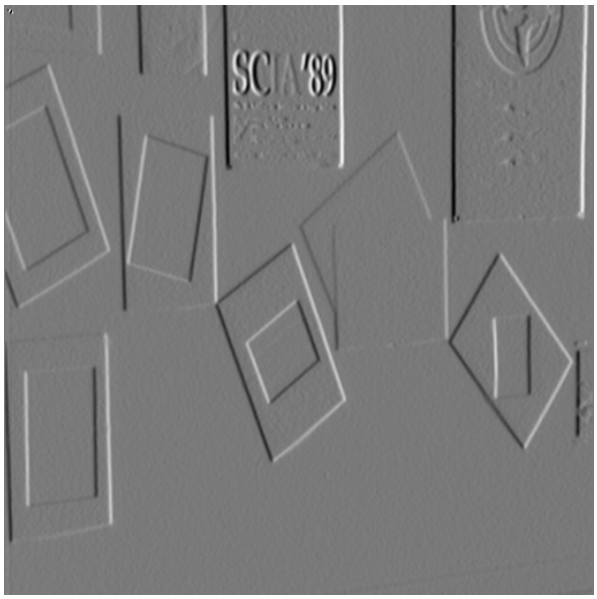
Se convierte la imagen a escala de grises

```
cvtColor(bgrMap, grey, CV_BGR2GRAY);
```

Se calculan los gradientes:

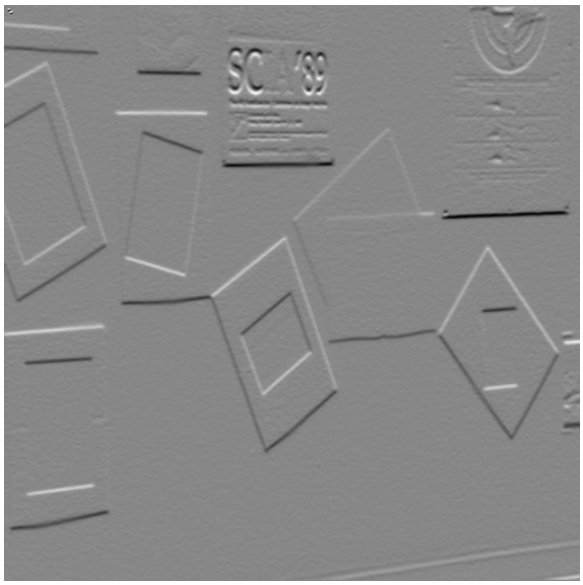
Gradiente Horizontal

```
 Sobel(grey, sobelx, ddepth, 1, 0, 3, scale, delta, BORDER_DEFAULT);
```



Gradiente Vertical

```
Sobel(grey, sobely, ddepth, 0, 1, 3, scale, delta, BORDER_DEFAULT);
```



Módulo del gradiente

```
convertScaleAbs(sobelx, abs_grad_x);
convertScaleAbs(sobely, abs_grad_y);
addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0, grad);
```

Orientación del gradiente

```
dir.at<float>(i, o) = M_PI - atan2(-sobely.at<float>(i, o), -sobelx.at<float>(i, o));
```

Cálculo del punto de fuga

Se aplica el operador de canny sobre la imagen en gris para reducir el posible ruido.

```
  Canny(grey, dst, 50, 200, 3);
```

Se hallan las líneas con un umbral por encima de 65 votos.

```
HoughLines(dst, lines, 1, CV_PI / 180, UMBRAL, 0, 0);
```

Se descartan las líneas horizontales y verticales de hasta cierto ángulo con respecto al horizonte.

```
if(abs(yD)< 0.9 && abs(yD)>0.1)
```

Se toman en cuenta los pixeles de las rectas cercanas al horizonte dentro de un margen en función al tamaño de la imagen.

```
while((intY>=minY&&intY<maxY))
```

Cada pixel vota al resto de los pixeles de la recta.

```
values[intX][valueY]=values[intX][valueY]+1
```

Se toma en cuenta el pixel más votado. Ese será el punto de fuga.

```
if(values[intX][valueY]>max){
    max=values[intX][valueY];
    maxIX=intX;
    maxIY=intY;
}
```