RADAR

Creación de un radar mediante una raspberry pi y una cámara

Antonio López Hernández
Jose Sierra Ibañez

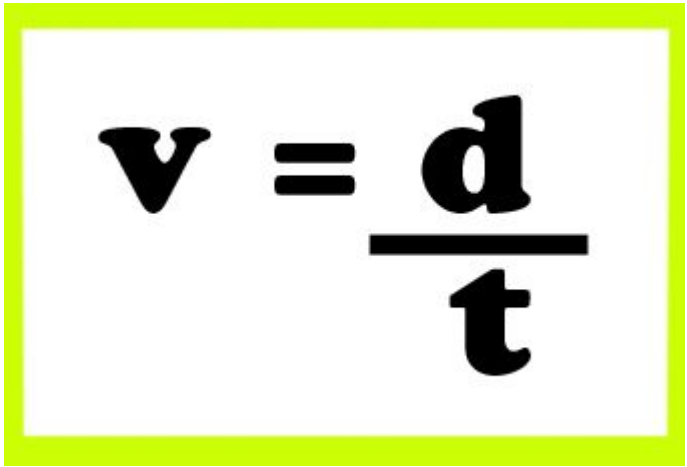
Idea principal

La velocidad no es más que un cálculo del tiempo que tarda un objeto en recorrer un espacio.

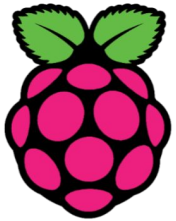
Velocidad nuestro objetivo

Distancia que recorre el objeto

Tiempo que tarda en recorrerla



Programas utilizados y Descartados



RaspberryPi

Primera y más importante consideración

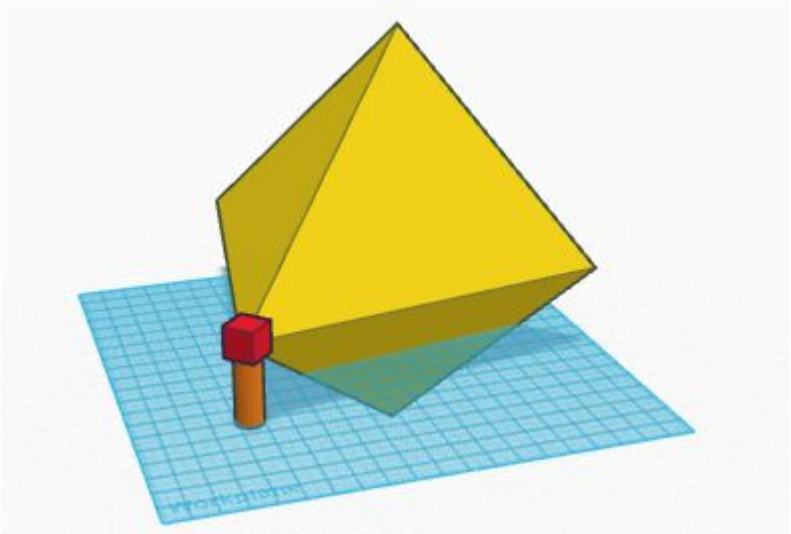
¿Qué espacio recorre el objeto si está siendo grabado con una cámara?

Es imposible hacer una perfecta medición sin dos sensores así que gracias a documentación se puede calcular una estimación. Esto lo descubrimos después de haber comenzado la práctica.

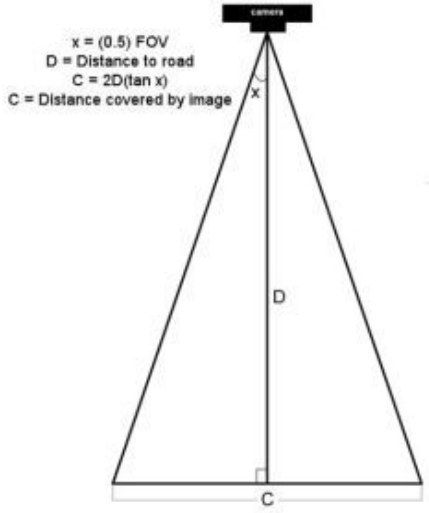
Como lo hemos solucionado y porque ocurre esto.

Problema de cámara única

Motivo la cámara graba en pico



Aproximación, distancia del objeto es igual a distancia recorrida



Código - partes importantes

```
# Distancia desde la cámara al objeto
DISTANCE = 2
# El numero de pixeles que hace falta que cambien para tenerlo en cuenta
THRESHOLD = 50
# El tamaño minimo del objeto para tenerlo en cuenta
MIN_AREA = 175
```

```
# Calculamos el tiempo transcurrido
def secs_diff(endTime, begTime):
    diff = (endTime - begTime).total_seconds()
    return diff
```

```
# Calculo de la velocidad
def get_speed(pixels, ftperpixel, secs):
    if secs > 0.0:
        return ((pixels * ftperpixel)/ secs) * 3.6
    else:
        return 0.0
```

Código - partes importantes

```
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    # Cogemos el tiempo actual
    timestamp = datetime.datetime.now()

    # Cogemos los 3 colores
    image = frame.array
    gray = image
    # Pasamos la imagen a escala de grises, que lo que hace es asignarle un numer entre 0-255
    gray = cv2.cvtColor(gray, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, BLURSIZE, 0)

    # Si no se ha definido la base de la imagen, es decir esta echando fotos y pasandolas a blanco
    if base_image is None:
        base_image = gray.copy().astype("float")
        lastTime = timestamp
        rawCapture.truncate(0)
        continue

    # Ahora comparamos las imagenes para saber cuantos pixeles han cambiado entre la base imagen y la continua
    frameDelta = cv2.absdiff(gray, cv2.convertScaleAbs(base_image))
    thresh = cv2.threshold(frameDelta, THRESHOLD, 255, cv2.THRESH_BINARY)[1]

    # Dilatamos la imagen para encontrar contornos
    thresh = cv2.dilate(thresh, None, iterations=2)
    (cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

    # Buscamos movimiento
    motion_found = False
    biggest_area = 0
```

```
# Vamos a encontrar los contornos de esta imagen para ver si hay movimiento
for c in cnts:
    (x, y, w, h) = cv2.boundingRect(c)
    # Aproximación del tamaño del objeto
    found_area = w*h
    # Ver si supera nuestro limite
    if (found_area > MIN_AREA) and (found_area > biggest_area):
        biggest_area = found_area
        motion_found = True
```

Código - partes importantes

Si encontramos moviemiento

```
#Si esta en espera se cambia a detección
if state == WAITING:
    # Inicializamos la detección
    state = TRACKING
# Obtenemos la posición y el tiempo
    initial_x = x
    last_x = x
    initial_time = timestamp
    last_kph = 0
    text_on_image = 'Detectando'
    print(text_on_image)
else:
    #Si está en modo detección obtenemos la dirección del objeto y calculamos los cambios
    # y los mostramos por consola
    if state == TRACKING:
        #Si va de izquierda a derecha
        if x >= last_x:
            direction = LEFT_TO_RIGHT
            abs_chg = x + w - initial_x
        #Si va de derecha a izquierda
        else:
            direction = RIGHT_TO_LEFT
            abs_chg = initial_x - x
        # Calculamos la velocidad a partir de los cambios en la imagen
        secs = secs_diff(timestamp,initial_time)
        kph = get_speed(abs_chg,ftperpixel,secs)
        #Imprimimos por consola
        print("--> chg={}  secs={}  km/h={}".format(abs_chg,secs,"%.0f" % kph))
        # Una vez el objeto ha salido del área, guardamos
        if ((x <= 2) and (direction == RIGHT_TO_LEFT)) \
                or ((x+w >= 440 - 2) \
                and (direction == LEFT_TO_RIGHT)):
            # Guardamos la foto obtenida que más cambios tenga
            cv2.imwrite("Foto.jpg",image)
            state = SAVING
        # Una vez salga el objeto, obtener la última velocidad y última posición
        last_kph = kph
        last_x = x
```

Si el objeto está parado y es detectado 20 veces, no lo consideramos como movimiento

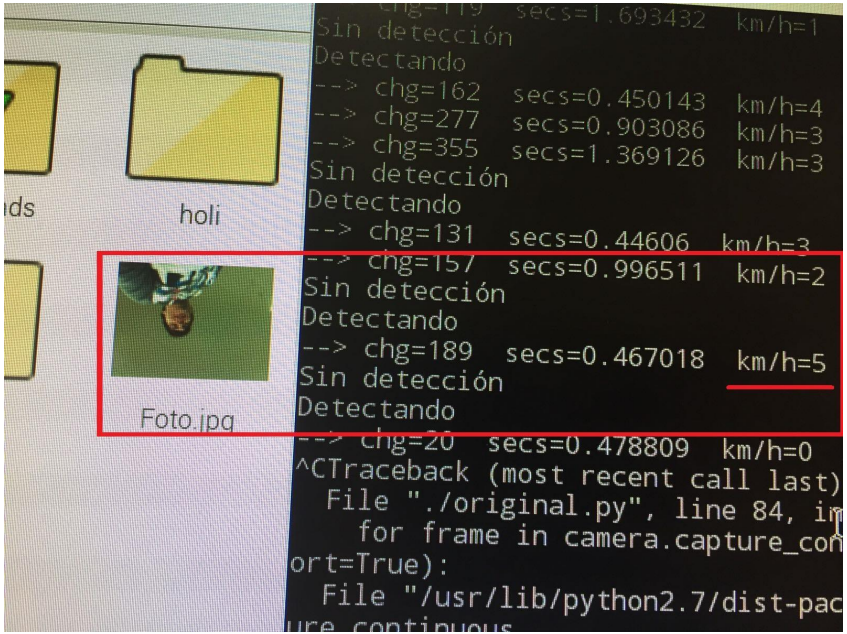
```
# Si el objeto está parado, no lo contamos como movimiento e impedimos
# que la cámara este detectándolo
if (state == WAITING) or (loop_count > 20):

    state=WAITING;
    loop_count = 0

# Limpiamos el stream
rawCapture.truncate(0)
loop_count = loop_count + 1
```

Salidas y el porqué de esas salidas

La mejor aproximación qué podemos hacer a la velocidad es el km/h que nos muestra el mayor número de cambios.



- DUDAS
- PREGUNTAS
- RECOMENDACIONES