

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/316282519>

DETECCIÓN DE OBJETOS EN IMÁGENES UTILIZANDO OPENCV PARA RASPBERRY

Conference Paper · April 2015

CITATIONS

0

READS

16,259

1 author:



Jorge Benjamin Magaña

Instituto Tecnológico Superior de Motul

4 PUBLICATIONS 0 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



Sistemas de visión por computadora [View project](#)

9. RECONOCIMIENTO DE PATRONES

9.1. DETECCIÓN DE OBJETOS EN IMÁGENES UTILIZANDO OPENCV PARA RASPBERRY

José R. Atoche E.², Jorge B. Magaña Z.¹, Jesús Sandoval G.², Carlos A. Luján R.²

¹Instituto Tecnológico Superior de Motul

Carretera Mérida-Motul, Tablaje Catastral 383, C.P. 97430. Teléfono 01-991-9151632.

²Instituto Tecnológico De Mérida

Av. Tecnológico km. 4.5 S/N C.P. 97118, Tel:(999) 964-5000, 964-5001 Fax:(999) 944-8181

jratoche@gmail.com, jorge.magana@itsmotul.edu.mx, jesus_sandoval_gio@yahoo.com.mx,
clujanitm@gmail.com

RESUMEN

El presente artículo describe la manera de utilizar el algoritmo Surf para la detección de los puntos claves de una imagen determinada así como la generación de los descriptores de la misma. Se presenta la manera de instalar el software Python y la librería Opencv utilizada para visión artificial en el sistema embebido Raspberry PI. Posteriormente se presenta de manera teórica el funcionamiento del algoritmo Surf para luego presentar la implementación, utilizando el software Python, en imágenes predefinidas de internet y con imágenes del objeto real.

Por último se presentan los resultados obtenidos de la implementación del algoritmo surf para la detección de objetos con el software Python desde la tarjeta Raspberry PI.

ABSTRACT

This article describes how to use the Surf algorithm for detecting the key points of a particular image and the generation of descriptors of it. It is presented how to install the software Python and OpenCV computer vision library used for the Raspberry PI embedded system. Subsequently presented the theoretically Surf operation algorithm and then present implementation, using the Python software, Internet predefined images and images of the real object.

Finally the results of the implementation of the algorithm surf for detecting the objects with Python software from the card are presented Raspberry PI.

1. INTRODUCCIÓN

Como parte del proyecto "Detección, Selección y Manipulación de Objetos Utilizando Visión Artificial Apoyada de un Brazo Robótico" surge la necesidad de utilizar un algoritmo capaz de detectar objetos sin importar la posición en la que se encuentre, en base a esta necesidad se plantea el uso del algoritmo surf.

En base a lo anterior se utilizó el programa Python para la implementación del algoritmo surf, debido a que este algoritmo es rápido y detecta homogéneamente los puntos de interés, lo que permite emparejamientos más confiables en condiciones normales [1].

2. INSTALACIÓN DE PYTHON Y OPENCV

El software Python, creado por Guido Van Rossum a principios de los años 90, pertenece a un lenguaje de programación interpretativo, de alto nivel, multiplataforma, de tipado dinámico y multiparadigma [2]. Python es interpretativo debido a que utiliza un programa intermedio llamado

intérprete, por lo que es un lenguaje flexible y portable. Este lenguaje tiene muchas de las características de los lenguajes compilados por lo que se puede considerar como un semi interpretado, también un lenguaje de alto nivel ya que consiste en una estructura sintáctica y semántica legible, puede ser interpretado en diferentes Sistemas Operativos como Windows, Mac OS, GNU/Linux entre otros, por lo que es Multiplataforma; sus variables no requieren ser definidas asignando su tipo de datos, se auto-asigna en tiempo de ejecución por lo que es dinámico; este lenguaje acepta diferentes paradigmas de programación, como la orientada a objetos, aspectos, la programación imperativa y funcional por lo que es multiparadigma [3].

OpenCV es una librería informática de código abierto desarrollado por intel bajo una licencia BSD (Berkeley Software Distribution), funciona en Mac OSX, Windows y Linux. Las principales funciones de OpenCV son: captura en tiempo real, importación de archivos de video, tratamiento básico de imágenes, detección de objetos, entre otras [4]. Para la detección de objetos se utilizan algoritmos que sean adecuados a los objetos a encontrar los cuales se encuentran en la librería de OpenCV, en nuestro caso, utilizaremos el algoritmo Surf.

2.1. Instalación de Python

El software Python se instala en el Sistema Operativo basado en el Kernel Linux instalado en la Raspberry Pi, la cual puede ser cualquier distribución de Linux, preferentemente Debian por su naturaleza ligera y recomendado por la Fundación Raspberry Pi [5].

El primer paso es abrir una terminal presionando Alt+F4 y escribiendo en el campo de búsqueda gnome-terminal, una vez abierta se escribe Python para comprobar si ya viene instalado el software Python, lo que aparecerá en pantalla es el Shell interactivo de Python y para salir se debe presionar Ctrl+D. En caso de que saliera un mensaje de error, similar a "**Python: command not found**" se procederá a actualizar primero la lista de los repositorios tecleando "**sudo apt-get update**", después el Sistema Operativo con "**sudo apt-get upgrade**" y por ultimo instalar Python con "**sudo apt-get install python2.7**" [3].

2.1. Instalación de OpenCV en Python

La forma de instalar OpenCV con la versión más reciente de la librería es primero instalando las herramientas de compilación que nos ayudarán en el proceso con los siguientes comandos:

1. `sudo apt-get install build-essential`
2. `sudo apt-get install cmake`
3. `sudo apt-get install pkg-config`

Terminada la instalación se procede a instalar las librerías de OpenCV y de NumPy usando el comando "**sudo apt-get install libgtk2.0-dev python-dev python-numpy**". Con el comando "**sudo apt-get install libpng12-0 libpng12-dev libpng++-dev libpng3 libpnglite-dev zlib1g-dbg zlib1g zlib1g-dev pngtools libjasper-dev libjasper-runtime libjasper1 libjpeg8 libjpeg8-dbg libjpeg62 libjpeg62-dev libjpeg-progs libtiffxx0c2 libtiff-tools ffmpeg libavcodec-dev libavcodec53 libavformat53 libavformat-dev libswscale2 libswscale-dev openexr libopenexr6 libopenexr-dev**" se podrán abrir y manipular diferentes formatos de imágenes y con "**sudo apt-get install libgstreamer0.10-0-dbg libgstreamer0.10-0 libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libxine1-ffmpeg libxine-dev libxine1-bin libunicap2 libunicap2-dev libucil2 libucil2-dev libdc1394-22-dev libdc1394-22 libdc1394-utils libv4l-0 libv4l-dev**" podremos abrir y manipular diferentes formatos de video [6].

3. ALGORITMO SURF

El algoritmo surf fue desarrollado por herbert Bay en el año 2006, basado en el algoritmo SIFT pero más robusto y libre de patentes. El algoritmo surf hace réplicas de la imagen para buscar puntos que estén en todas las réplicas asegurando la invariancia de escala. Utiliza determinantes Hessianas

para la detección de los puntos de interés localizando a la vez su posición así como su escala; utiliza filtros tipo caja para aproximar las derivadas parciales de segundo orden del filtro de Gauss y ser evaluadas de forma mucho más rápida usando imágenes integrales [7]. El procedimiento del algoritmo SURF para detectar puntos de interés (keypoints), asignación de la orientación y el descriptor SURF se puede apreciar en la Figura 1 [8].

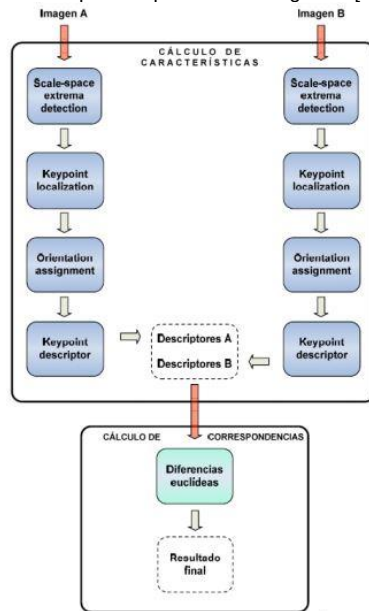


Figura 1. Proceso para realizar un muestro con Surf [8].

El algoritmo SURF extrae los puntos de interés y su localización dentro de la imagen, seguidamente se representa la vecindad del punto de interés como un vector de características que por defecto es de un tamaño de 64 [9]. El algoritmo SURF utiliza una aproximación básica de la matriz Hessiana, utilizada para localizar los puntos y la escala. Por lo que dado un punto $x = (x, y)$ en una imagen "I", la matriz Hessiana $H(x, \sigma)$ en x con escala σ se define como se observa en la ecuación 1.

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \quad (1)$$

Donde $L_{xx}(x, \sigma)$ es la convolución de la derivada de segundo orden de la Gaussiana, $\frac{\delta^2}{\delta x^2} g(\sigma)$ con la imagen "I" en el punto x , y similarmente para $L_{xy}(x, \sigma)$ y $L_{yy}(x, \sigma)$. Para el cálculo del determinante, se realizan aproximaciones a las derivadas de segundo orden de la Gaussiana de modo que se obtienen tres aproximaciones: D_{xx} , D_{xy} , D_{yy} . De esta forma, el determinante de la Hessiana que nos indica la escala del punto se calcula con la ecuación 2 [10].

$$\det(H_{aprox}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \quad (2)$$

Donde el valor 0.9 está relacionado con la aproximación del filtro Gaussiano.

Una vez realizada la selección de puntos de interés, es necesario disponer de la orientación y la vecindad. Primero calculamos la respuesta Haar en X y Y en área circular de radio 6s alrededor del punto de interés, con s la escala del punto de interés [9]. Las wavelets de Haar puede calcularse rápidamente a través de la integración de imágenes, similar al filtro gaussiano de segundo orden. La orientación dominante se estima y se incluye en la información del punto de interés [11].

Para la creación del descriptor se construye una región cuadrada centrada en el punto de interés y orientada a lo largo de la orientación dominante establecida anteriormente, de tamaño 20s como se observa en la Figura 2.



Figura 2. Regiones cuadradas centradas alrededor del punto de interés de tamaño de 20s [9]

La región se divide en 4x4 subregiones con el fin de conservar cierta información espacial. En cada subregión, wavelets de Haar se extraen puntos de muestreo regularmente espaciados. Las respuestas de wavelets en las direcciones horizontal y vertical (dx y dy) se suman a lo largo de cada subregión y los valores absolutos de $|dx|$ y $|dy|$ se suman para obtener información acerca de la polaridad de los cambios de intensidad de la imagen. De ahí que el patrón de intensidad subyacente de cada subregión es descrito por un vector V como se observa en la ecuación 3 las regiones presentadas por la ecuación 3, los puntos resultantes son invariables a la escala.

$$V = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy| \right] \quad (3)$$

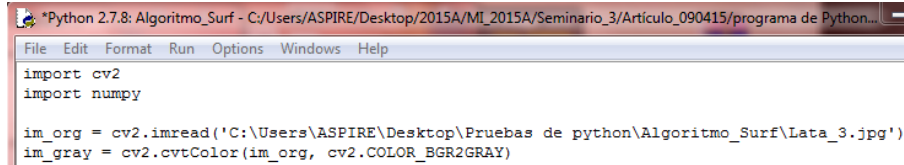
El vector descriptor resultante para todas las 4x4 sub-regiones es de una longitud de 64, dando el descriptor SURF estándar [11].

4. IMPLEMENTACIÓN

Teniendo instalado Python, se procedió a realizar la programación para obtener los descriptores de las imágenes. Para realizar la programación ejecutamos el IDLE de Python (Python GUI). Desde el shell del IDLE abrimos un archivo nuevo para comenzar la programación, los pasos que se siguieron para aplicar el algoritmo Surf a una imagen, desde Python, son los siguientes:

1. Se realizó la importación de librerías de OpenCV y Numpy, nos permite trabajar con imágenes, video y con funciones matemáticas de alto nivel.
2. Se procedió a leer y guardar la imagen a tratar en una variable llamada "im_org" con la función `cv2.imread('imagen')`, entre comillas va la dirección de la imagen que queremos leer, por ejemplo: `'C:\Users\ASPIRE\Desktop\Pruebas de python\captura_img.jpg'`
3. Es necesario convertir la imagen del formato jpg a escala de grises y guardarla en una nueva variable, en nuestro caso "im_gray", para ello utilizamos el comando `cv2.cvtColor(im_org, cv2.COLOR_BGR2GRAY)`. También podemos convertir la imagen directamente del comando `cv2.imread` agregando un 0 dentro del paréntesis `"cv2.imread('imagen',0)"`.

La Figura 3 muestra el código donde se importan las librerías OpenCV y Numpy, para posteriormente leer la imagen y convertirla a escala de grises.



```
*Python 2.7.8: Algoritmo_Surf - C:/Users/ASPIRE/Desktop/2015A/MI_2015A/Seminario_3/Articulo_090415/programa de Python...
File Edit Format Run Options Windows Help
import cv2
import numpy

im_org = cv2.imread('C:\Users\ASPIRE\Desktop\Pruebas de python\Algoritmo_Surf\Lata_3.jpg')
im_gray = cv2.cvtColor(im_org, cv2.COLOR_BGR2GRAY)
```

Figura 3. Código para convertir a escala de grises una imagen en Python

4. Se procedió a determinar los puntos clave (keypoints) y los descriptores, para ello se declara una variable igual al comando **cv2.SURF()** con la cual declaramos el umbral para el punto clave, los valores recomendable son entre 300 y 500. Posteriormente se utilizó el comando **cv2.detectAndCompute (im_gray, None)** con ello buscamos los puntos clave y los descriptores de la imagen. Para poder visualizar los puntos y los vectores descriptores utilizamos el comando **"img2 = cv2.drawKeypoints(im_org,kp,None,(255,0,0),4)"**, nos dibuja círculos de color azul en la imagen original donde están los puntos clave.

El código para visualizar los puntos clave y los descriptores se muestran en la Figura 4 y en la Figura 5 se puede apreciar la imagen con los puntos clave y los descriptores dibujados sin orientación.

```
import cv2
import numpy

im_org = cv2.imread('C:\Users\ASPIRE\Desktop\Pruebas de python\Algoritmo_Surf\Lata_3.jpg')
im_gray = cv2.cvtColor(im_org, cv2.COLOR_BGR2GRAY)

surf=cv2.SURF(22000)#22000 para kp 19

#puntos claves y descriptores
kp,des=surf.detectAndCompute(im_gray,None)

#dibujar círculos en los puntos claves
img2 = cv2.drawKeypoints(im_org,kp,None,(255,0,0),4)

#mostrar imagen con los puntos claves
cv2.imshow('surf',img2)

#Salir de la ventana con cualquier letra
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figura 4. Código para buscar y dibujar los puntos clave y descriptores de una imagen



Figura 5. Imagen de los puntos clave y descriptores desorientados.

5. Para darle orientación a los vectores descriptores se utiliza el comando **surf.uptight=True** con esta función pondremos orientación a los vectores descriptores de manera que todos queden orientados de manera vertical y cambiamos el tamaño descriptores básicos a descriptores extendidos, en la Figura 6 se puede apreciar la implementación de las función encerradas en un rectángulo de color rojo y la imagen con los descriptores ya orientados.

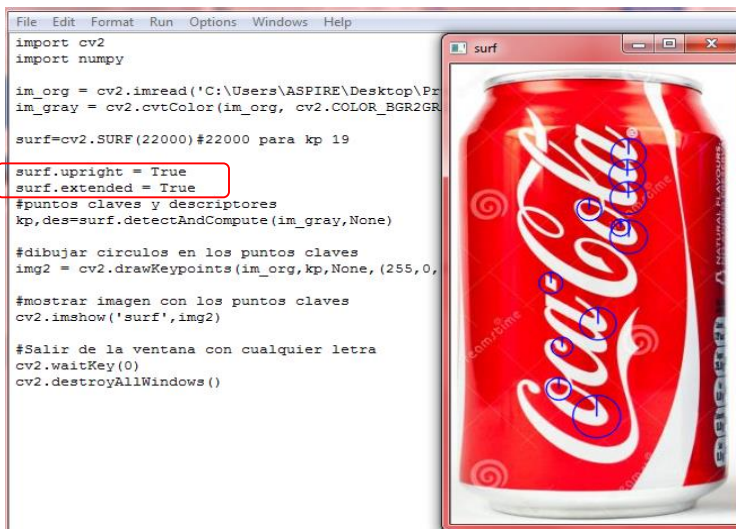


Figura 6. Orientación de los descriptores

Una de las características del algoritmo surf es detectar los descriptores de una imagen sin importar la posición en que se encuentre. En la Figura 7 se puede apreciar la imagen rotada 90° y en ella se observan los mismos descriptores.



Figura 7. Imagen a 90° de la original

Si la imagen se ubica con otras, se observa que los descriptores de la imagen se mantienen, en la Figura 8 se aprecia la imagen con otras similares y se observan los descriptores.



Figura 8. Imagen con la imagen original y descriptores

En la Figura 9 se puede apreciar la imagen del objeto real en diferentes posiciones y con otros objetos. Se puede apreciar que los descriptores del objeto se conservan en las 5 imágenes mostradas con líneas de color verde.

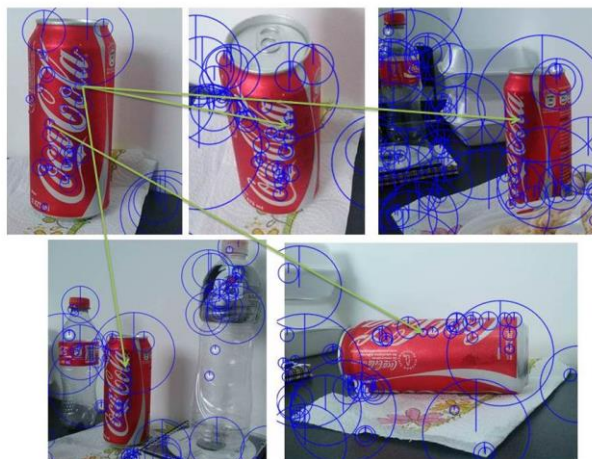


Figura 9. Imagen del objeto real

La imagen 9 muestra los descriptores para un umbral relativamente alto entre 3000 y 4000 para poder visualizarlos, sin embargo el recomendado es de 300 a 400 [9], este rango de umbral mostraría una gran cantidad de descriptores por lo que sería complicado visualizarlo a simple vista.

5. RESULTADOS

Los resultados observados al utilizar el algoritmo surf en el software Python fueron alentadores para su implementación en la detección y selección de objetos con visión artificial y brazo robótico. Se obtuvieron los mismos vectores descriptores de la imagen original girada a 90° mostrada en la Figura 7 y en la Figura 8 se observó que los descriptores de la imagen original se mantienen casi en su totalidad.

Los resultados concuerda con la teoría, el algoritmo es adecuado para detectar objetos sin importar su posición y rotación. En base a estos resultados se pretende realizar la coincidencia de los descriptores de manera que solo se observen los puntos clave del objeto a buscar, así mismo se pretende realizar pruebas con objetos capturados desde una cámara para comprobar el efecto de la variación de la luz en el objeto a detectar.

6. REFERENCIAS

- [1] I. G. Barquero, P. S. González, M. L. Serrano y E. G. Aguilera, *Comparación de algoritmos detectores de puntos singulares para reconocimiento de objetos en vídeo quirúrgico.*, 2012.
- [2] R. G. Duque, *Python para todos*, España: Autoedición, 2010.
- [3] E. Bahit, *Curso:Python para Principiantes*, Buenos Aires, Argentina, 2012.
- [4] EcuRed, «EcuRed Conocimiento con todos y para todos,» [En línea]. Available: <http://www.ecured.cu/index.php/OpenCV>. [Último acceso: 10 Abril 2015].
- [5] E. Upton, *Raspberry Pi Guía del Usuario*, 2014.
- [6] I. V. M. Ruiz, «Operaciones básicas con OpenCV y Pytho,» [En línea]. Available: http://turing.iimas.unam.mx/~ivanvladimir/es/content/teach/curso_aprendizaje_automatico_s4.html. [Último acceso: 11 Abril 2015].

- [7] T. J. P. Ordoñez, Recuperación de Imágenes Basada en Contenidos Utilizando el Método SURF, Cuenca-Ecuador, 2012.
- [8] R. A. López, Desarrollo de un Sistema Cognitivo de Visión para la Navegación Robótica, Valencia, 2012.
- [9] H. Bay, A. Ess, T. Tuytelaars y L. V. Gool, «Speeded-Up Robust Feactures (SURF),» *Computer vision and image understanding*, vol. 110, nº 3, pp. 346-359, 2008.
- [10] A. M. Romero y M. Cazorla, «Comparativa de detectores de características visuales y su aplicación al SLAM,» *X WORKSHOP DE AGENTES FÍSICOS*, nº X, pp. 55-62, 2009.
- [11] A. Murillo, J. J. Guerrero y C. Sagués, «SURF features for efficient robot localization with omnidirectional images,» *I Robotics and Automation, 2007 IEEE international Conference on*, pp. 3901-3907, 2007.