Universidad Nacional de San Agustín

Curso: Robótica

Seguimiento Visual de Objetos Utilizando Robot con Sensor Ultrasónico

---

Integrantes:
Sumari Huayta, Oliver
Castillo Galdos, Lorena
Pariguana Medina, Eyner
Dávila Guillén, Grimaldo
López Condori, Juan José

18 de diciembre de 2018

Índice

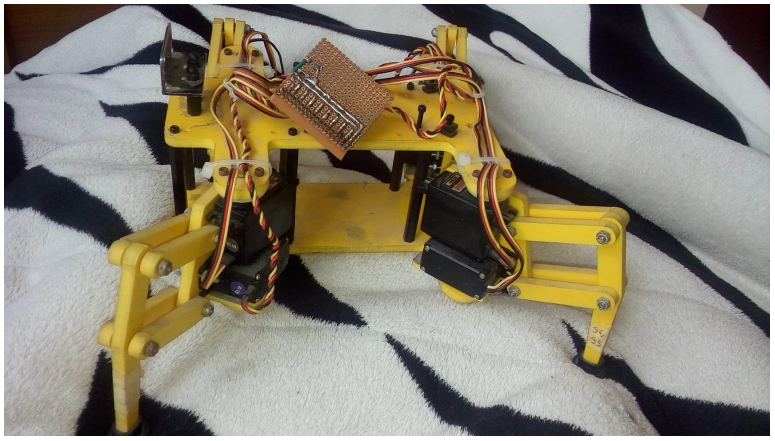
Proyecto: Seguimiento Visual de Objetos Utilizando Robot con Sensor Ultrasónico

1. Introducción

La identificación visual de objetos, personas y sus movimientos es una característica importante en múltiples aplicaciones, tales como sistemas de vigilancia, interfaz de realidad virtual, percepción robótica, reconocimiento de acciones, bases de datos de video, cuartos inteligentes, entre otras. De tal manera que a partir de un *frame* inicial con la ubicación de un objeto se pueda hacer un seguimiento estimando de forma autónoma hacia los estados del objeto en *frames* posteriores.

Mediante el uso de sensores electrónicos podemos extraer diversos datos para poder simular movimiento. Los sensores son dispositivos que consisten en "transformar una variable física que puede ser de cualquier tipo ya sea temperatura, presión, gas, velocidad, etc. Con otra variable más fácil de trabajar", esta vez analizaremos los sensores que convierten una variable relacionada con la distancia hacia un objeto con una señal de tipo electrónica y nos centraremos en el funcionamiento de los sensores ópticos y ultrasónicos los cuales son muy utilizables en muchas aplicaciones prácticas dentro de la rama de robótica al momento de realizar cualquier proyecto.

La habilidad de encontrar y seguir las partes del cuerpo de una persona es un problema visual importante. El presente trabajo está enfocado a modelar, localizar y seguir a un objeto dado. Algunas aplicaciones en la vida real son: inspección automática en fabricas, sistema de identificación de especias, robots industriales, videovigilancia, interacción humano-computador, análisis de imágenes médicas, navegación autónoma de robots, indexación de imágenes, etc. Lo que se busca es mejorar la interacción hombre-máquina y conseguir que las computadoras funcionen como asistentes humanos.



2. Problema

Un paso importante para ello es dotar a las computadoras de inteligencia perceptual, es decir, dotarlas de habilidades que les permitan determinar que aspectos de una situación son significativos, y elegir en consecuencia una acción adecuada.

El problema radica allí. Sin un algoritmo específico, el movimiento del robot puede llegar a ser irregular y no suave como se puede pensar. Los algoritmos ACC son un sistema inteligente para vehículos donde automáticamente ajusta la velocidad del vehículo para mantener una distancia segura. Mediante el uso del sensor ultrasónico interconectado con un Arduino Mega, el robot deseado puede ser construido y el algoritmo puede ser implementado fácilmente debido a la metodología de código abierto, dando como resultado un movimiento suave del robot.

Al utilizar un vehículo no tripulado (robot) para grabar videos, nos percatamos que la grabación del video sufre movimientos bruscos de *frame* en *frame*. Esto sucede porque el movimiento dado por el vehículo sufre distintos posibles inconvenientes. Depende de muchos factores como; el viento, la estabilidad del modelo del robot, las dificultades de los sensores o la resolución de la imagen de la cámara con el que adquiere los datos visuales. También influye la luminosidad del ambiente o los posibles inconvenientes repentinos que se pueda tener al moverse el vehículo.

Los resultados de distintos algoritmos en videos grabados tienen una disminución muy alta de su precisión. El problema es como modelar el movimiento del vehículo en base a los videos que son visualizados en 2D.

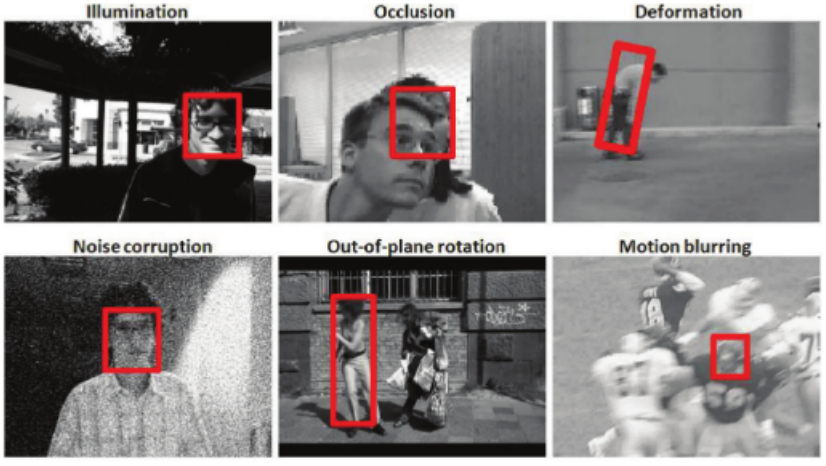


Figura 1: Problemas y Desafios.

3. Temas Relacionados

- Los enfoques generativos rastrean un objeto buscando la región mas similar a un modelo de referencia. Estos métodos suelen basarse en templates (Alt et al.

2010 [AHN10]); dictionary learning (Arandjelovi´c[Ara15] y Wang et al.2013 [WY13]).

- En contraste los enfoques discriminativos aprenden a clasificar siendo capaces de separar el objetivo del fondo o puntos de no interes en los *frames*. Se han usado técnicas avanzadas de Machine Learning, incluyendo boosting (Grabner et al. 2008 [GLB08]), Multiple Instance Learning (Babenko et al. 2011 [BYB11]), entre otras.
- Las Redes Neuronales Convolucionales Profundas (CNN por el ingles Convolutional Neural Network) han demostrado un gran éxito en seguimiento de objetos. Hong et al. [HYKH15].

4. Concepto: Visión Computacional

La visión computacional es un área de investigación que se encarga de analizar y procesar imágenes y/o videos para poder comprender y procesar esta información por medio de la computadora. Clasificación de acuerdo a visión y cámara:

- CEOE: Cámara estática, objetos estáticos.
- CEOM: Cámara estática, objetos en movimiento.
- CMOE: Cámara en movimiento, objetos estáticos.
- CMOM: Cámara en movimiento, objetos en movimiento.

Uno de los objetivos principales de la visión computacional es emular las funciones básicas del ojo humano como percepción del movimiento y comprensión de escenas. El seguimiento visual de objetos es el proceso de localizar un target en movimiento a lo largo del transcurso de un video. A continuación la selección de características.

- Color: Normalmente, el espacio de color RGB se usa para representar el color.
- Bordes: Una propiedad importante de los bordes es que a diferencia de las características basadas en color, estas son menos sensibles a los cambios de iluminación.
- Flujo Óptico: El flujo óptico es un campo denso de vectores de desplazamiento que define la traslación de cada píxel en una región.
- Textura: La textura es una medida de la variación de intensidad de una superficie que cuantifica propiedades tales como la suavidad y la regularidad.

5. Propuesta

Para implementar un sistema que cumpla los objetivos propuestos, necesitamos un robot programable, que proporcione una serie de servos para su movilidad tanto para sus movimientos horizontales como los verticales (en sus patas).

Será necesario también un sensor ultrasónico para captar la distancia del robot con el objeto a seguir, a su vez alguna cámara que pueda procesar imágenes en tiempo real, y capacidad computacional suficiente para procesar dichas imágenes y proveer de datos al robot sobre la posición del objetivo en cada *frame*.

Respecto a la cámara, la limitación de esta solución se encuentra en la baja calidad de las imágenes que proporcionan estas cámaras de bajo coste, y sobretodo en la reducida capacidad de cómputo que tiene la CPU de este tipo de robots, que normalmente trabajan en un solo núcleo y a una velocidad entre 20-50 MHz, frente al procesador de un dispositivo móvil de gama baja que puede trabajar entre los 400 y 800 MHz, o uno de gama media-alta que pueden trabajar con 2-4 núcleos y a velocidades superiores a 1 GHz.

Además, utilizando un dispositivo móvil se cuenta con una ventaja adicional: podemos utilizar librerías externas especializadas en visión por computador que implementan potentes algoritmos y estructuras de datos para el tratamiento de imágenes. Los smartphones contienen cámara, sensor de distancia, micrófono, y muchos otros componentes más. Además que son muy comunes, en muchos casos es menos costoso hacer uso de un smartphone doméstico en vez de comprar distintos sensores para un robot.

Se propone un conjunto formado por un robot con su sensor de ultra sonido para obtener la distancia generada por las emisiones de sonido que generará dicho dispositivo, y un dispositivo móvil que se usará para adquirir y procesar las imágenes a través de su propia cámara. La dos partes que formarán el sistema (robot y móvil) se integrarán mediante una comunicación bluetooth para la sincronización y el intercambio de datos.

Una vez planteado el sistema, se han de elegir las tecnologías que usaremos para el desarrollo del proyecto. En este caso son tres: plataforma robótica, tipo de dispositivo móvil y librería para implementar los algoritmos de visión por computador.

6. Diseño

En la figura 2 se muestra el diseño de nuestro robot.

Figura 2: Diseño

Como se observa cuenta con 4 patas, 8 servomotores, arduino mega, una batería y un sensor ultrasónico.

7. Materiales

Los materiales empleados en el robot fueron los siguientes:

- **Arduino Mega 2560** las especificaciones se muestran en la Figura 3.
- **Memoria:** El Atmega2560 tiene 256 KB de memoria flash para almacenar el código (de la que se utilizan 8 KB para el cargador de arranque), 8 KB de SRAM y 4 KB de EEPROM (que puede ser leída y escrita con la biblioteca EEPROM)
- **Alimentación eléctrica:** El Mega 2560 puede ser alimentado a través de la conexión USB o con una fuente de alimentación externa. La fuente de alimentación se selecciona automáticamente. La alimentación externa (no USB) puede venir de un adaptador de CA a CC o de una batería. El adaptador se puede conectar al enchufe de 2,1 mm de centro-positivo en la clavija de alimentación de la placa. Los cables desde una batería pueden ser insertados en GND y en el pin Vin del conector de alimentación. La tarjeta puede funcionar con un suministro externo de 6 a 20 voltios.

- **Ultrasonic Distance Sensor (28015)**: Los sensores de distancia son dispositivos que consisten en transformar una variable física que puede ser de cualquier tipo.

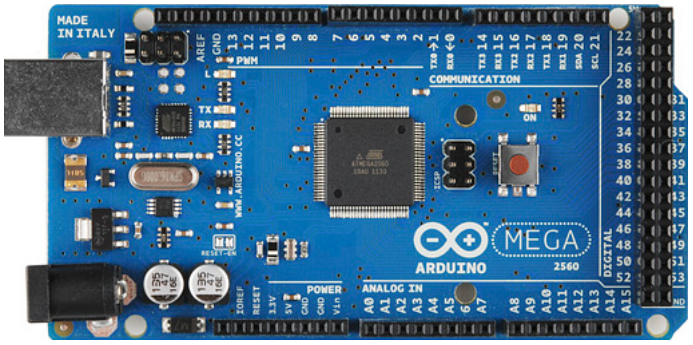


Figura 3: Arduino Mega 2560

Microcontrolador	ATmega2560
Tensión de trabajo	5V
Tensión de entrada (recomendada)	7-12V
Tensión de entrada (límite)	6-20V
Pines Digitales I/O	54 (de los cuales 15 proporcionan salida PWM)
Pines de entradas Analógicas	16
DC Corriente por Pin I/O	20 mA
DC Corriente por Pin 3.3V	50 mA
Memoria Flash	256 KB de los cuales 8 KB se usan por el bootloader
SRAM	8 KB
EEPROM	4 KB
Velocidad del reloj	16 MHz
Largo	101.52 mm
Anchu	53.3 mm
Peso	37 g

Figura 4: Arduino Mega 2560

Figura 5: Ultrasonic Distance Sensor (28015)

Rango	Símbolo	Rango	unidad
Voltaje de alimentación	Vcc	5	V
coriente de alimentacion	Ia	-30-35	mA
Temperatura de operacion	T_{opr}	0-70	C

Figura 6: Rangos absolutos de operación del sensor 28015

8. Costo

Material	Costo
Arduino Mega 2560	60 soles
Smarphone	600 soles
Cables para conectar	10 soles
Batería	20 soles
Servomotores(x8)	160 soles
Sensor	20 soles
Total	870 soles

9. Script para el Arduino y el Sensor Ultrasónico

Inicialmente incluimos las librerías para hacer uso de los servomotores y a su vez para poder borrar la memoria de tareas del arduino cada vez que intentamos enviar nuevas configuraciones de nuestro programa.

```
#include <Servo.h>
#include <EEPROM.h>
```

Inicializamos nuestras variables, inicialmente las conexiones del sensor con el arduino (en las variables Trigger y Echo que recepciona y transmite los datos obtenidos por el sensor). Seguidamente nuestras vaiables de las cuatro patas, los servomotores, añadiendo las variables de las posiciones que radica entre (0 y 180 grados).

```
const int Trigger = 2;   //Pin digital 2 para el Trigger del sensor
const int Echo = 3;   //Pin digital 3 para el Echo del sensor

Servo myservo9;
Servo myservo11;
```

```
Servo myservo8;
Servo myservo10;

int pos = 0;
int pos1 = 170;
```

Usamos tres funciones *Setup*, *Planchas* y *Loop*. La primera función (Setup) se encarga de borrar la memoria del arduino con las antiguas ordenes obtenidas. Inicializar nuestro sensor de ultra-sonido y sincronizamos con el arduino y sus slots. Y por último sincronizamos nuestros servos con los slots del arduino e inicializamos el ángulo con el que comenzará cada pata de nuestro robot.

```
void setup()
{
  for (int i = 0; i < 512; i++)
    EEPROM.write(i, 0);

  digitalWrite(13, HIGH);

  Serial.begin(9600);//iniciailzamos la comunicaci[U+FFFD]n
  pinMode(Trigger, OUTPUT); //pin como salida
  pinMode(Echo, INPUT);  //pin como entrada
  digitalWrite(Trigger, LOW);//Inicializamos el pin con 0

  myservo9.attach(9);
  myservo11.attach(11);
  myservo8.attach(8);
  myservo10.attach(10);

  myservo9.write(170);
  myservo11.write(170);
  myservo8.write(10);
  myservo10.write(10);
}
```

En la segunda función básicamente indicamos el movimiento de nuestro robot que deseamos que obtenga, ya sea para caminar, girar, o algún otro movimiento en particular.

```
  for(pos = 10; pos < 170; pos += 1)  //de 10 a 170 grados
  {
    pos1 -=1;
    myservo8.write(pos);
    myservo10.write(pos);
    //////////
    myservo9.write(pos1);
    myservo11.write(pos1);
```

```
    delay(15); // espera 15ms
  }

  for(pos = 170; pos>=10; pos-=1)  // de 170 grados a 10
  {
    pos1 += 1;
    myservo8.write(pos);
    myservo10.write(pos);
    myservo9.write(pos1);
    myservo11.write(pos1);

    delay(15);  // espera 15ms
  }
```

Ya en la última función podemos repetir el movimiento creado por la función ya mencionada anteriormente. Aquí extraemos la ditancia obtenida del objeto con el robot mediante el sensor ultrasónico y si sobrepasa o no esa distancia podemmos indicarle que haga alguna acción en dicho bucle.

```
void loop()
{
  long t; //timepo que demora en llegar el eco
  long d; //distancia en centimetros

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);          //Enviamos un pulso de 10us
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH); //obtenemos el ancho del pulso
  d = t/59;              //escalamos el tiempo a una distancia en cm

  Serial.print("Distancia: ");
  Serial.print(d);       //Enviamos serialmente el valor de la distancia
  Serial.print("cm");
  Serial.println();

  if(d<120){
    planchas();
  }
}
```

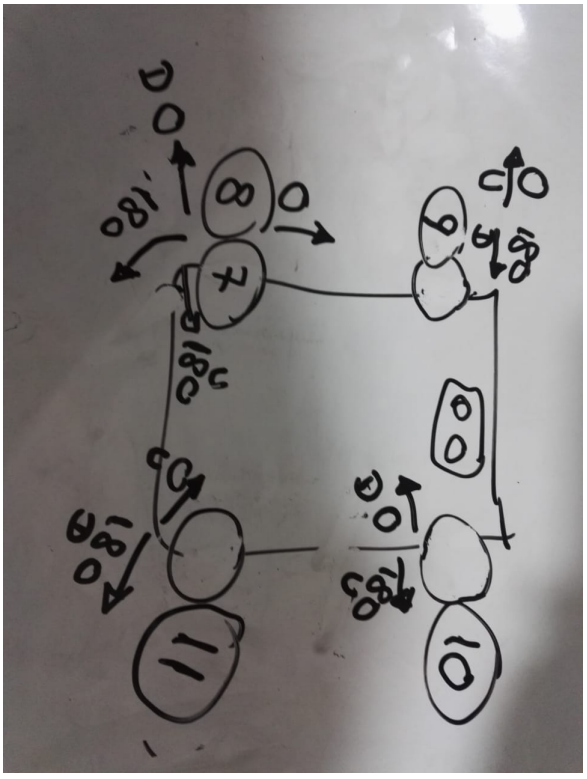


Figura 7: Angulos de cada uno de los servomotores,especificados en el código

10. Algoritmo de Seguimiento de Objeto

Para estimar el movimiento se formula como un problema de estimación de estado dinámico: *Xt = F(Xt-1; Vt-1)* y *Zt = H(Xt; Wt)*, donde Xt es el estado actual, F es la función de evolución de estado, Vt-1 es el ruido del proceso de evolución, Zt es la observación actual, h denota la función de medición y Wt es el ruido de medición. El procesamiento del reconocimiento de objetos consta de dos pasos:

- Primero se debe enseñar a reconocer un objeto determinado y este debe ejecutarse antes de la operación principal del robot. Durante este paso, el objeto se presenta al sistema de visión, la imagen y el conjunto de características extraídas se guardan como un patrón. Muchos objetos pueden ser presentados al sistema, es decir puede aprender mas de un objeto.
- El segundo paso es el reconocimiento real que se ejecuta constantemente durante la operación del robot. Cada fotograma de la cámara se procesa, las características de la imagen se extraen y se comparan con los datos establecidos en la memoria. Si suficientes características coinciden con el patrón, entonces el objeto es reconocido.

En este proyecto utilizaremos el *find_object_2d* función del paquete *find_object_2d* para la enseñanza y el reconocimiento.

Find_object_2d:
Una de las ventajas de ROS es que tiene toneladas de paquetes que se pueden reutilizar en nuestras aplicaciones. En nuestro caso, lo que queremos es implementar un reconocimiento de objetos y un Sistema de detección. El paquete find_object_2d implementa los detectores de funciones SURF, SIFT, FAST y BREVE y los descriptores para la detección de objetos.

- *SURF(Speeded-Up Robust Features):* Es un algoritmo de visión por computador, capaz de obtener una representación visual de una imagen y extraer una información detallada y específica del contenido. Esta información es tratada para realizar operaciones como por ejemplo la localización y reconocimiento de determinados objetos, personas o caras, realización de escenas 3D, seguimiento de objetos y extracción de puntos de interés. Este algoritmo forma parte de la mencionada inteligencia artificial, capaz de entrenar un sistema para que interprete imágenes y determine el contenido. SURF es un detector y un descriptor de alto rendimiento de los puntos de interés de una imagen, donde se transforma la imagen en coordenadas, utilizando una técnica llamada multi-resolución. Consiste en hacer una réplica de la imagen original de forma Piramidal Gaussiana o Piramidal Laplaciana, y obtener imágenes del mismo tamaño pero con el ancho de banda reducido. De esta manera se consigue un efecto de borrosidad sobre la imagen original, llamado Scale-Space. Esta técnica asegura que los puntos de interés son invariantes en el escalado. El algoritmo SURF está basado en el predecesor SIFT.
- *SIFT(Scale-invariant feature transform):* Es un algoritmo usado en visión artificial para extraer características relevantes de las imágenes que posteriormente pueden usarse en reconocimiento de objetos, detección de movimiento, estereopsis, registro de la imagen y otras tareas.
- *FAST(Features from Accelerated Segment Test)* El algoritmo FAST es un detector de esquinas que se caracteriza por producir puntos relevantes muy estables. Este método pertenece a la categoría AST (Accelerated Segment Test), que es una versión modificada del criterio SUSAN para la detección de esquinas. Es el detector de esquinas computacionalmente más eficiente que existe hasta el momento. Este algoritmo trabaja mediante ventanas circulares centradas en cada uno de los puntos de la imagen. Para que un punto sea detectado como esquina deben existir un número mínimo de puntos dentro de su ventana que sean una cantidad umbral más claros que el centro u otra cantidad umbral más oscuros que el centro.

11. Viabilidad

Cuadro 1: Posible Cronograma de Actividades para el desarrollo de nuestro proyecto

Actividad	Inicio Aprox.	Fin Aprox.
Elaboración del primer borrador del proyecto	*3-Set-18*	*10-set-18*
Investigar sobre los precios y la viabilidad del proyecto	*10-set-18*	*17-set-18*
Empezar a programar los algoritmos que usaremos	*24-set-18*	*08-Oct-18*
Comprar los implementos	*08-Oct-18*	*22-Oct-18*
Ensamblar nuestro robot	*22-oct-18*	*29-oct-18*
Hacer pruebas tanto con los algoritmos como con el robot	*29-Oct-18*	*05-Nov-18*
Unir las dos partes de desarrollo	*05-Nov-18*	*19-Nov-18*
Corrección de errores	*19-Nov-18*	*03-Dic-18*
Últimos ajustes/presentación final	*03-dic-18*	*07-dic-18*

12. Conclusiones

Los sensores de proximidad son de gran utilidad en el campo de la electrónica la mayoría de proyectos y micro proyectos llevan sensores tanto ópticos como ultrasónicos por su precisión para localizar un objeto en la practica podríamos llamarles los ojos de nuestros proyectos ya que el sensor nos brinda la señal cuando un objeto se encuentra dentro de su zona de visualización lo cual es aplicable para realizar diversas funciones por ejemplo poner en marcha motores al detectar un objeto como lo hacen la mayoría de robots sumos de batalla electrónicos.

Visual Object Tracking tiene muchas aplicaciones en el mundo real por eso causa interés en varias áreas de investigación. A pesar de ser un tema muy investigado, el desafío de obtener robustes, velocidad y precisión es todavía muy alto. En el campo de los Drones no tiene mucha investigación, es un campo grande para investigadores futuros. El siguiente paso o los trabajos a seguir sería de implementar algoritmos generados por nosotros en Tracking usando Drones.

Referencias

[1] https://husarion.com/tutorials/ros-tutorials/4-visual-object-recognition/4-visual-object-recognition-teaching-objects

[2] https://husarion.com/tutorials/husarnet/following-object-using-your-smartphone/

[3] Ognjen Arandjelovic, Automatic vehicle tracking and recognition from aerial image sequences, arXiv preprint arXiv:1506.06881 (2015).

[4] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han, *Online tracking by learning discriminative saliency map with convolutional neural network*, International Conference on Machine Learning, 2015

[5] Naiyan Wang and Dit-Yan Yeung, *Learning a deep compact image representation for visual tracking*, Advances in neural information processing systems, 2013

[6] Nicolas Alt, Stefan Hinterstoisser, and Nassir Navab, *Rapid selection of reliable templates for visual tracking*, Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 1355–1362.

[7] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie, *Robust object tracking with online multiple instance learning*, IEEE transactions on pattern analysis and machine intelligence 33 (2011), no. 8, 1619–1632.

[8] Helmut Grabner, Christian Leistner, and Horst Bischof, *Semi-supervised on-line boosting for robust tracking*, European conference on computer vision, Springer, 2008, pp. 234–247.