Tratamiento digital de la imagen

MEMORIA

Proyecto Final

Detector de bicicletas de montaña

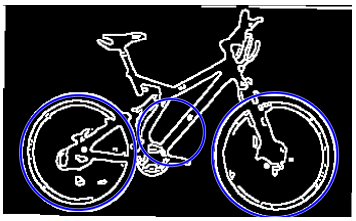
María Íñigo Romillo
Andrés Gómez del Río

ÍNDICE

DESCRIPTORES SELECCIONADOS

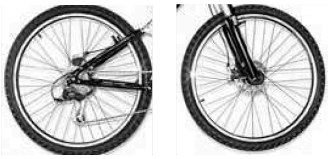
Detector de círculos

El primer detector que hemos incluido, y el que más interviene a la hora de ejecutar los demás detectores, es el detector de círculos. Al ser nuestra temática bicicletas de montaña, lo más destacable son las ruedas, por lo que para detectar correctamente que en la imagen aparezca una bicicleta, tendrá tendrá que haber detectado al menos dos círculos: uno por cada rueda. En el caso de no detectar ningún círculo los valores del vector características serán nulos, es decir, no se considerará como una bicicleta.



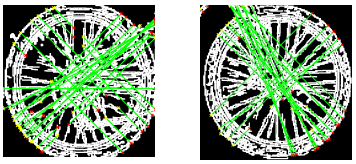
Detector de ruedas semejantes a cierta distancia

Otra característica de las bicicletas de montaña es que el tamaño de sus ruedas, que son bastante parecidas y que se encuentran a una cierta distancia. Por ello, nuestro segundo detector implementa una comparación entre los radios de los círculos detectados previamente y en el caso de haber detectado más de un círculo (como ocurre en el ejemplo), comparará todos los círculos con todos los restantes y seleccionará únicamente dos círculos con tamaños dentro de un rango de (0.7xcírculo_comparandose, 1.3xcírculo_comparandose). A continuación verificará si la distancia a la que se encuentran estos círculos es un múltiplo (0.8x4xcírculo_comparandose, 1.2x4xcírculo_comparandose). En caso de no detectarse círculos con radios semejantes y a una distancia determinada, tanto esa variable, como la del detector de líneas con cierto ángulo (última característica) serán nulas, si por el contrario si existen círculos con radios semejantes la característica de este descriptor será la división de un radio entre el otro.



Detector de radios de las ruedas

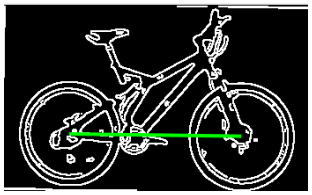
Las ruedas de las bicicletas tienen un conjunto de alambres que conectan la rueda de goma con el eje giratorio, los radios. Nuestro tercer detector se inspira en esta idea e implementa un descriptor para detectar las lineas, con un cierto tamaño (minlenth=7), que haya dentro de los círculos seleccionados previamente.

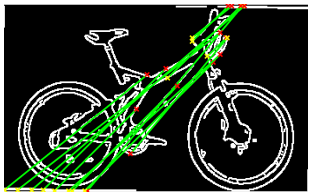


Este detector se aplicará siempre que haya algún círculo detectado en la imagen, por lo que es posible que se ejecute y se cumplan las condiciones incluso si la característica anterior (círculos semejantes a cierta distancia) no se verifica.

Detector de lineas con cierto ángulo

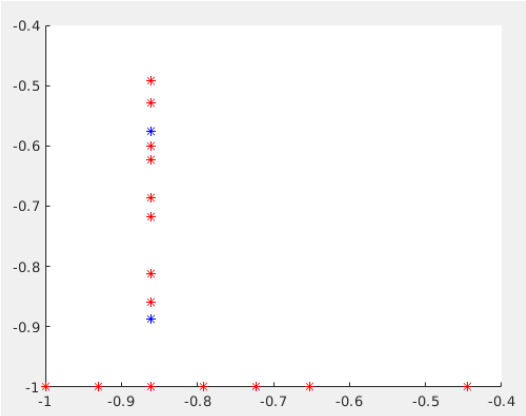
Toda bici cuenta con un tronco o eje que suele tener gran tamaño y forma rectilínea y que suele tener un cierto ángulo característico respecto a la base de las ruedas en todas las bicicletas. De manera que podemos detectar, usando el mismo método que en el detector de radios, todas las lineas que tengan un cierto tamaño (minlength=7) y que formen un determinado ángulo (entre 35º y 55º) con el segmento que une los centros de las ruedas, por lo que es necesario que se cumpla la segunda caracteristica, es decir que se hayan detectado dos circules de tamaños semejantes y a una cierta distancia relativa a su tamaño. Para ello se implementa una condición de selección de ángulos.



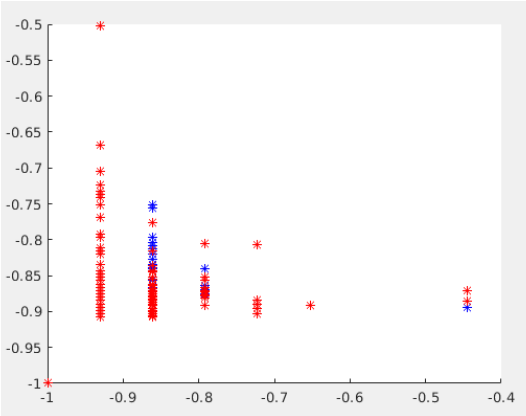


GRÁFICAS DE DISPERSIÓN

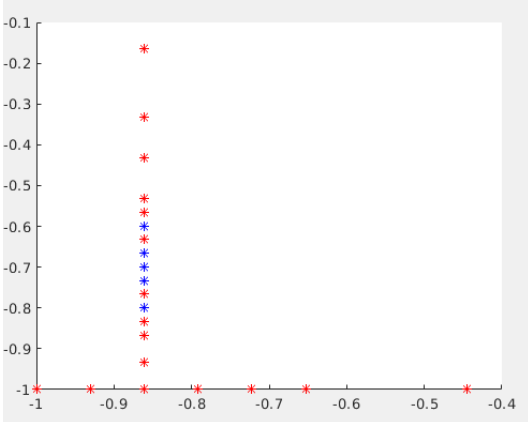
Detector de círculos-ruedas semejantes



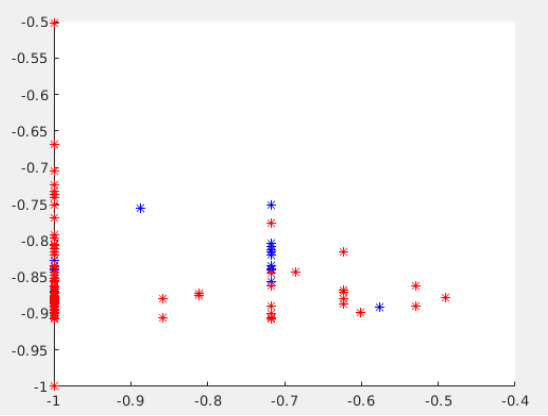
Detector de círculos-radios de las ruedas



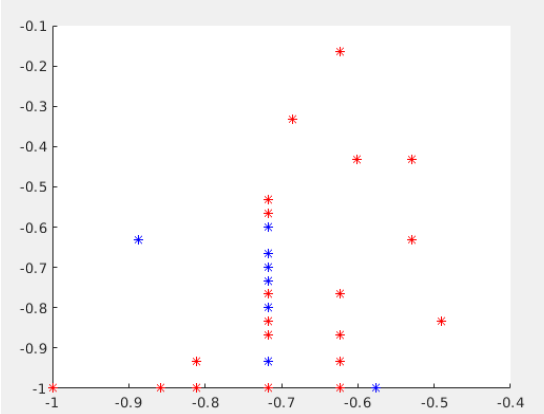
Detector de círculos-lineas con cierto ángulo



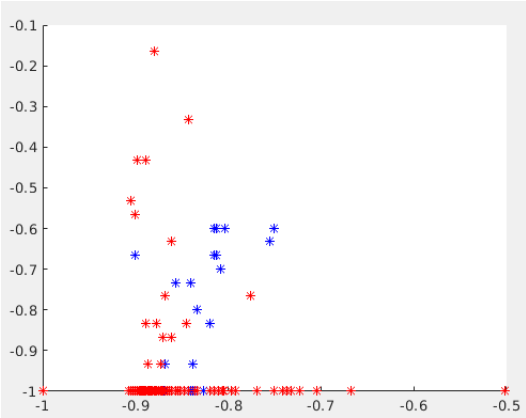
Detector de ruedas semejantes-radios de las ruedas



Detector de ruedas semejantes-lineas con cierto ángulo



Detector de radios de las ruedas-lineas con cierto ángulo



Estas gráficas ilustran las comparaciones de características para imágenes que contienen alguna bicicleta e imágenes que no.

Aparecen 303 imágenes: 42 de bicicletas (*) y el resto de distintas categorías de objetos (*)

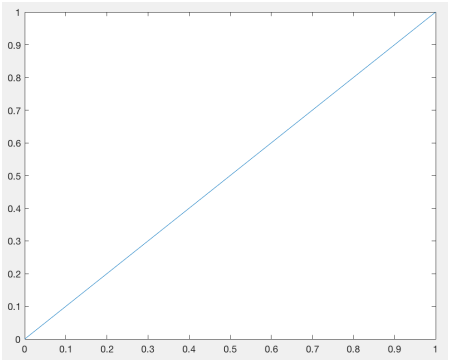
CLASIFICADOR UTILIZADO

En esta parte del proyecto hemos querido ir más allá de nuestra selección subjetiva de un modelo de clasificador. Esto ha supuesto algunos cambios notables pero el resultado será siempre más eficiente y fiable. Para empezar, ha sido necesario incluir una carpeta nueva llamada *validation* cuya función principal es servir de ejemplo de como podría ser el resultado total, es decir, del clasificador final seleccionado, con unas imágenes insertadas que deben ser una mezcla entre algunas de las imágenes recogidas en las carpetas *Train* y *Test.*

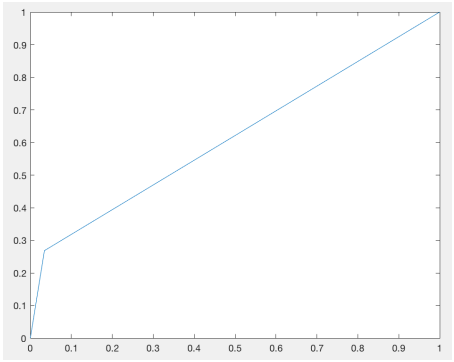
Las imágenes de esta nueva carpeta, se leerán exactamente igual que se ha hecho con las imágenes de la carpeta *Train* y únicamente se recurre a dichas imágenes, con sus respectivos valores y características, a la hora de seleccionar el modelo final con el que se hará la predicción. Para esta selección, se realiza una simulación de predicciones con cada uno de los modelos. El resultado final y modelo seleccionado será aquel que devuelva una predicción de mayor probabilidad, por lo que también se implementa una comparación.

Este método de selección proporciona una mayor seguridad a la hora de elegir un modelo de clasificación, ya que al no ser elegido subjetivamente y para un único caso no nos arriesgamos a que para en algún caso concreto la predicción no sea la mejor.

Sin embargo, para el caso concreto de nuestras imágenes nuestro predictor ha tenido un valor mayor para el modelo de k-NN, concretamente para cinco vecinos (k=5) por lo que es probable que este vaya a ser el modelo seleccionado generalmente por la máquina.



Predicción con el modelo SVM (0.5)



Predicción con el modelo 5-NN (0.67)

EVALUACIÓN Y DISCUSIÓN DE RESULTADOS

Con las características implementadas para la detección y el decisor k-NN obtenemos un 67% de probabilidad de acierto en nuestra máquina, es decir que existe un 67% de probabilidad de que detecte la presencia de una bicicleta en una imagen con una bicicleta. Teniendo en cuenta que el 50% es una probabilidad de acierto prácticamente aleatoria nuestro resultado es bastante decente, ya que se aproxima al 70%.

Es posible que algunas de nuestras características sean demasiado discriminativas, como la detección de círculos. Como se puede observar en la primera gráfica de dispersion el resultado es bastante tajante y solo aparecen dos valores que concentran todas las posibles imágenes donde se detecten círculos. Sin embargo todas juntas crean un conjunto bastante fiable de características restrictivas.

BREVE DESCRIPCIÓN DE LA ESTRUCTURA DE SOFTWARE SELECCIONADO

Función *Clasifica()*

Es la función principal de la aplicación, en ella se llama a todas las demás funciones y se realizan las funciones más básicas como el recorrido de los archivos de las carpetas *Train*, *Test* y *Validation* y la lectura de las imágenes.

Función *ExtraeCaracteristicas()*

Es importante destacar que antes de realizar cualquier método discriminatorio, las imágenes son leídas (`imread()`), transformadas a nivel de grises (`rgb2gray()`), rotadas (`imrotate()`) y dilatadas (`imdilate()`) los bordes (`edge()`).

Para el detector de círculos ha sido necesario implementar el método `imfindcircles()` que se basa en el método de la transformada de Hough. También hemos empleado la transformada en el método de `houghlines()` del cual también hemos obtenido el ángulo de las lineas aplicando `.theta` a los valores obtenidos con `houghlines()`.

Tanto la segunda como la segunda como la última característica se basan en condiciones comparativas para restringir los valores.

Función *DiagramasDeDispersion()*

Esta función es únicamente una forma de comprobar que los resultados obtenidos con las características establecidas son razonables.

El proceso requiere la implementación del método `scatter()` y de un bucle que separa las imágenes con bicicletas de las que no las tienen.

Función *EntrenaClasificador()*

Es en esta función donde se selecciona el decisor con el que se van a realizar las predicciones. Por lo tanto es necesario que aparezcan tanto `fitcknn()` como `fitcsvm()` para obtener los modelos de cada decisor, `predict()` para saber las predicciones que devolvería cada uno de estos modelos, `perfcurve()` para finalmente obtener la predicción final y por último hará falta un comprador para seleccionar el modelo óptimo.

Función *Clasifica()*

En esta sección se usaran los mismos métodos que se han empleado para seleccionar el decisor en la función *EntrenaClasificador()*.