

Bloque-3.pdf



Daysapro



Fundamentos de Sistemas Inteligentes en Visión



3º Grado en Ingeniería Informática



**Escuela Politécnica Superior de Córdoba
Universidad de Córdoba**



WUOLAH Print

Lo que faltaba en Wuolah



Imprimir



quieres trabajar
en Wuolah??

TE BUSCAMOS

BLOQUE 3

Las sesiones de las clases resumidas

*Después de leer el resumen intentad hacer las preguntas de otros años que algunas están en
wuolah
- Daysa -*

sin ánimo
de lucro,
chequea esto:



tú puedes
ayudarnos a
llevar
WUOLAH
al siguiente
nivel
(o alguien que
conozcas)

Tema 9: Introducción al análisis de imágenes

9.1 Introducción

Cuando pensamos en comparar 2 imágenes de la misma escena intuitivamente nos surge la duda de cómo podemos sacar información o características en común de ellas. Una de las primeras dudas que nos surgiría es cómo conseguir una correspondencia, y aquí vuelve a entrar un tema del cual ya hemos hablado, los puntos característicos. Son puntos únicos en la imagen y están diferenciados a simple vista. También podríamos llegar a la conclusión que para el análisis de imágenes son importantes las fronteras de los elementos e incluso el propio área de cada elemento.

En este bloque vamos a ver técnicas para estudiar estas características y establecer los fundamentos de análisis de imágenes.

9.2 Características de una imagen

Una característica es una magnitud escalar o vectorial obtenida a partir de una imagen. Deben ser aspectos significativos para cada caso en común, solo debemos valorar las característica que sean realmente de utilidad ante el problema que tengamos. Hay 3 tipos de características:

- Bajo nivel. Colores, texturas, gradiente, flujo óptico...
- Medio nivel. Áreas, bordes y puntos característicos.
- Alto nivel. Histogramas, contornos...

Necesitamos características de alto nivel para resolver los 3 problemas clásicos del análisis de imágenes por computador:

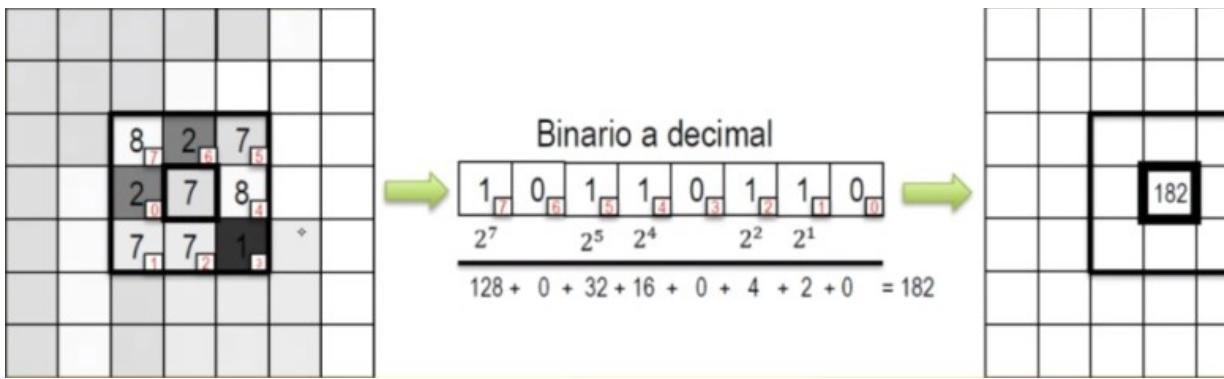
- Categorización. ¿Qué hay en la imagen? ¿Qué es? ¿Es una persona, un animal...?
- Detección. ¿Qué es y dónde está? Encontramos al objeto y lo recogemos en boinding boxes, cajas rectangulares alrededor del mismo.
- Segmentación. Diferenciación de píxeles del objeto buscado.

En los siguientes apartados trataremos las características de medio nivel y qué podemos usar para detectarlas. Después, definiremos los posibles descriptores que podemos usar para determinar valores de utilidad de las mismas, y que nos sirven para comparar el mismo tipo de objeto en caso de que aparezca en otra imagen.

9.3 Áreas

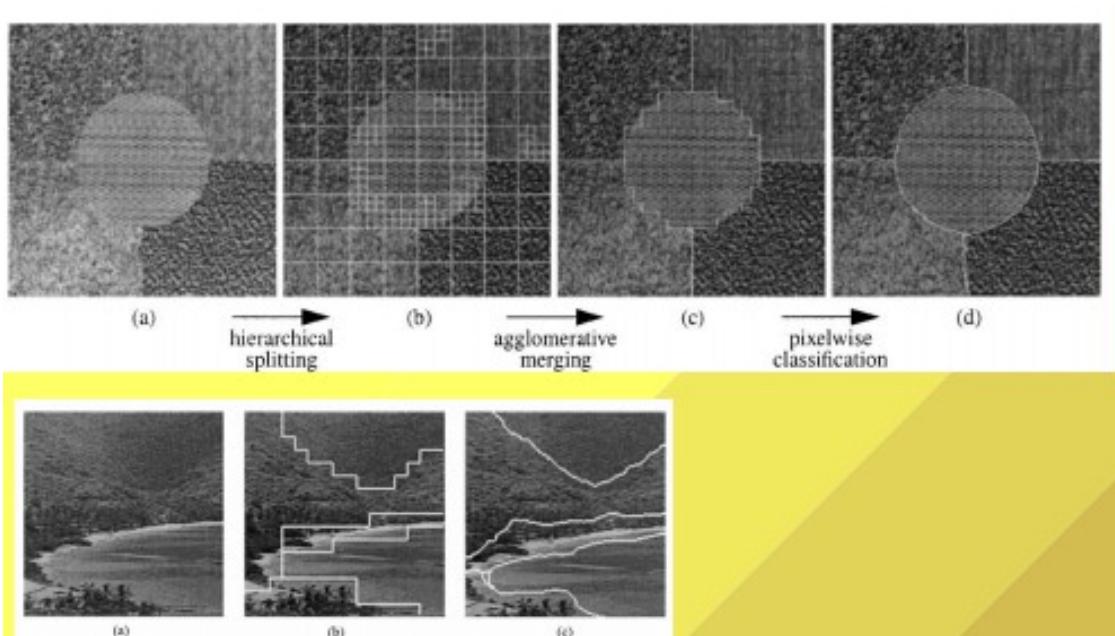
- Características para detección de áreas:
 - Color. Tiene algunos defectos a corregir, y es común procesar la imagen para normalizar la iluminación, con técnicas como el white patch, una normalización cromática, coordenadas cromáticas... Nos ayuda a diferenciar áreas porque, en general, todos los puntos que pertenecen a un mismo área comparten características de color en común.
 - Textura visual. En las texturas encontramos reglas de repetición, que podemos aprovechar para diferenciar objetos. A veces es sencillo si la textura es regular, pero otras veces en texturas irregulares es más complicado.

Podemos introducir una característica de bajo nivel muy útil. El patrón binario local (LBP). Nos recuerda a filtrado no lineal, ya que existe una ventana y hacemos una umbralización en un orden determinado. Como vemos en la imagen a continuación, comparamos si es mayor el valor del píxel central en relación a los valores de sus vecinos y a la salida le vamos sumando una potencia de 2 en un orden de vecinos arbitrario decidido por el programador.



También tiene una versión uniforme (LBP-U), que ayuda a introducir invarianza al patrón. Otra forma de caracterizar la textura de la imagen es el banco de filtros de Gabor. Un filtro de Gabor es un filtro lineal que consigue una función sinusoidal multiplicada por una función gaussiana. Obtenemos entonces un conjunto de filtros distintos de manera que al pasarlo a la imagen original conseguimos una imagen con tantos canales como filtros del banco, y en forma de vector esta imagen nos sirve para distintos usos de análisis de textura.

- Flujo óptico. Es una característica de bajo nivel que determina el patrón del movimiento aparente de los objetos, superficies y bordes en una escena. Se consigue en base a una diferencia de tiempo, y cómo ambas imágenes tomadas desde el mismo punto en la misma escena tienen diferencias entre ellas. Existe un detector de flujo llamado Lucas-Kanade detector. Este, estima el flujo óptico sobre puntos característicos asumiendo que el entorno de vecindad de un punto se mueve de igual forma.
- Umbralización. Es un ejemplo de clustering por partición. Consiste en poner todos los píxeles de la imagen en un intervalo determinado, consiguiendo áreas relevantes en blanco y fondo negro. Para la diferenciación de áreas tenemos 2 algoritmos relevantes: ISODATA Y Otsu. ISODATA minimiza la varianza intra-clases (kmeans() en OpenCV) y Otsu la maximiza entre-clases (threshold() en OpenCV).
- Split-merge. Es clustering jerárquico. Se divide la imagen convolucionada con el LBP, comparando sus histogramas. Si cambia mucho de un valor determinado arbitrariamente por el programador, considera una parte diferente y sigue dividiendo. Así hasta obtener todas las partes iterativamente. Después se unen las partes de otras partes más parecidas para ir refinando el resultado y poco a poco vamos obteniendo un resultado final.



A dynamic, low-angle photograph of two young people running towards the right against a solid teal background. Both individuals are captured mid-stride, leaning forward. They are both holding yellow ColaCao cans to their mouths, suggesting they are drinking from them. The person on the left wears a white and blue striped t-shirt and blue jeans. The person on the right wears a teal t-shirt and maroon pants. The overall composition conveys a sense of motion and energy.

#ESTASREADYCOLACAO

ColaCao®

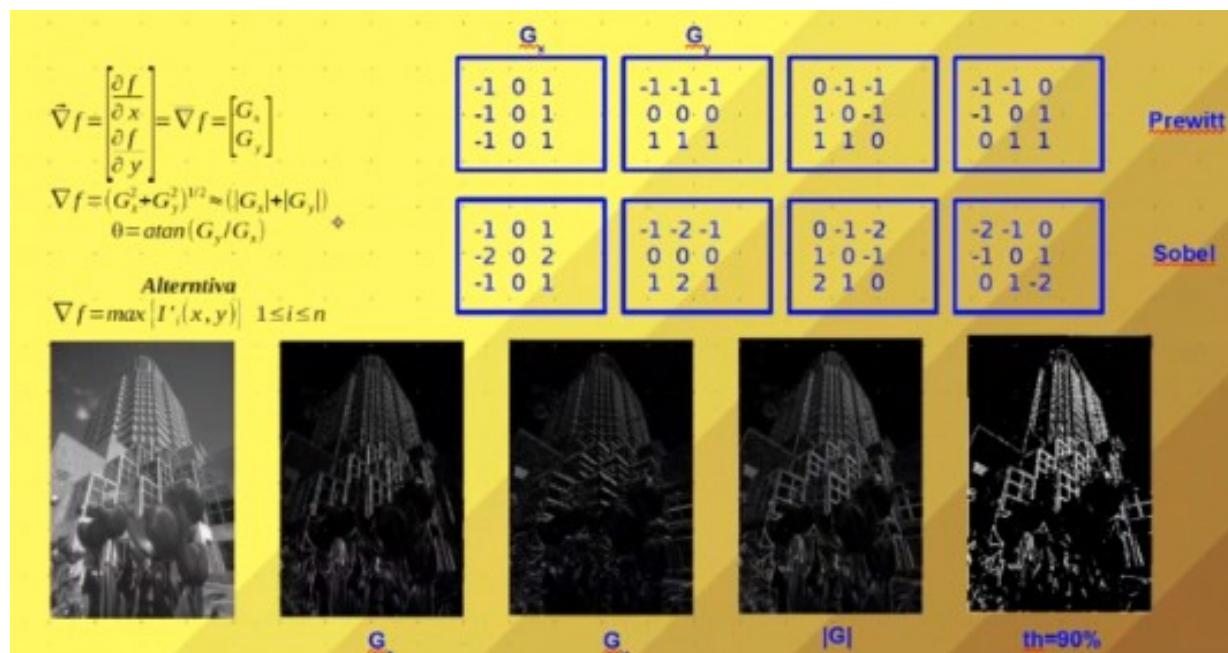
- Etiquetado de componentes conexas. Es una técnica para recuperación de objetos distintos después de obtener las áreas diferenciadas en caso de que las formas sean muy parecidas. Por ejemplo, diferenciar monedas. Todas son circulares, pero no son iguales, son objetos diferentes: monedas de 50 céntimos, de 20... En OpenCV se hace con `findContour()`.
 - Histogramas. Es una forma de reducir los datos de color, donde el eje x son los distintos colores y el eje y el número de píxeles totales de ese color.
 - Momentos invariantes. Son propiedades de la imagen que se pueden recoger en una tabla y no cambian esté en el estado que esté la imagen (ya la rotemos, le cambiemos la escala...), y podemos recogerlos en un vector cuyo elementos describen características de la imagen. Por ejemplo, si calculamos un número determinado de momentos invariantes de muchas caras distintas, a grandes rasgos los momentos serán parecidos, aunque no iguales; pero lo suficientemente parecidos como para determinar que una nueva imagen es una cara y no una señal de tráfico por ejemplo.
- Características para descripción de áreas:
- Descriptores de forma. Ocupación, compacidad, ocupación convexa y solidez son distintos descriptores de forma que recogen datos distintos de una imagen y combinándolos entre ellos o con otras técnicas podemos diferenciar qué objeto es un área determinada.
 - Descriptores de Fourier, basados en el contorno. Son herramientas para recuperar una forma determinada.

9.4 Bordes

Informáticamente, un borde es un paso entre píxeles que al darse cambia el color de la imagen. Es decir, sirven como límite entre 2 regiones de la imagen. Una de las herramientas que usaremos para detectarlos es la derivada. Estas presentan un problema, ya que las derivadas son muy sensibles al ruido y gran parte de los casos obtenemos unos bordes borrosos.

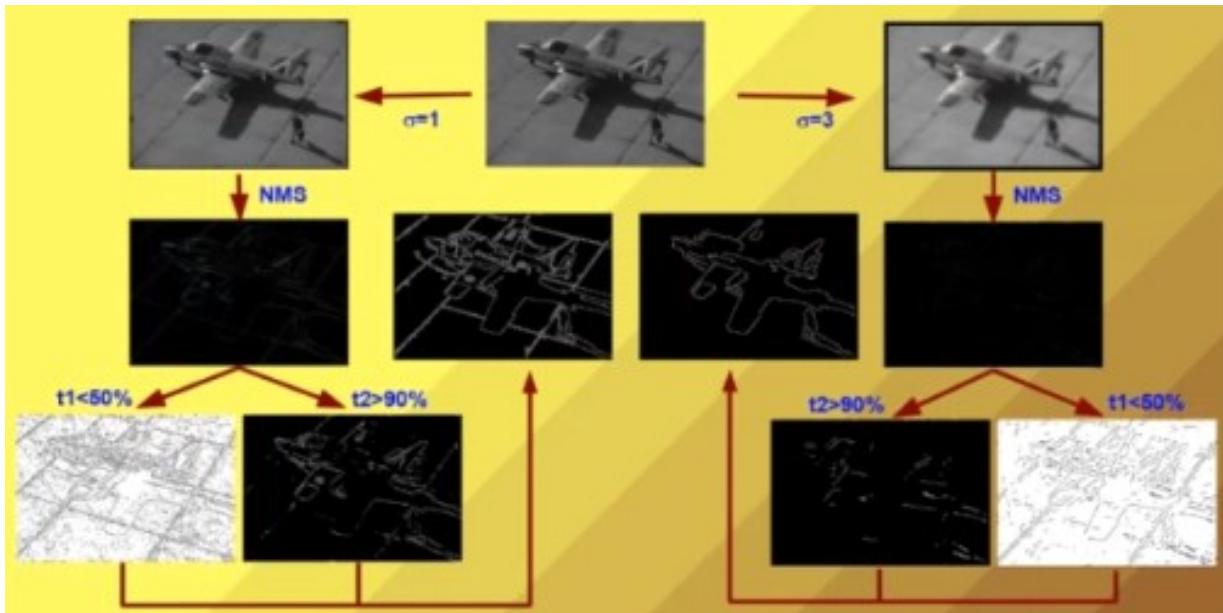
Por ello, usaremos solo la primera derivada, porque ya la segunda nos expone demasiado al ruido. Cabe señalar antes de comenzar con la explicación de las técnicas que en su aplicación, usaremos imágenes en blanco y negro.

- Características para detección de bordes:
 - Gradiante. Se aplica el filtro del gradiante con la convolución y obtenemos los siguientes resultados.



El mejor umbral para conseguir los mejores bordes tras aplicar el filtro está entre el 85% y 90%.

- Detector o método de Canny. Primero obtenemos una magnitud del gradiente, similar al caso anterior, pero con un filtro gaussiano. Tras esto, eliminamos todo píxel que no sea un extremo local en la dirección del gradiente, algo que mejora mucho la localización de los bordes, reduciendo falsos píxeles de borde (NMS). Después debemos realizar un umbral doble (valor inicial y final), eliminando los píxeles que tengan valores fuera del intervalo de umbral y para acabar un enlace de los puntos por un proceso llamado histéresis.



9.5 Puntos característicos

Un punto es característico es un píxel de la imagen sobre el que si nos movemos en distintas direcciones, el sistema detecta mucha variación.

- Características para detección de puntos característicos:
 - Función de autocorrelación. Dado un punto de la imagen, calculamos una ventana alrededor de él y valoramos cuánto se parece el punto en relación a la ventana. El resultado de la función se suele representar en 3D para observar la variación.
 - Detector de Harris. La función de autocorrelación anterior Harris la relacionaba con una matriz, la cual analizaba para obtener conclusiones. Establecía que si los 2 autovalores de la matriz eran altos, tenemos 2 direcciones de variación, así que en esas direcciones se encontrará algún punto característico. El cálculo de autovalores es costoso computacionalmente, así que Harris establecía una puntuación a cada punto de la imagen (Harris score), que establecía que los valores de puntuación más altos, eran puntos característicos. El usuario es el que determina quedarse con los n valores más altos de la imagen.

$$\text{Harris score:}$$

$$R = s_1 \cdot s_2 - k \cdot (s_1 + s_2)^2 = \det(M) - k \cdot \text{Traza}(M)^2$$

con k en $[0.04, 0.15]$ (el autor usa 0.06)

En OpenCV, el detector de Harris se implementa con `goodFeaturesToTrack()`, y se refinan con `cornerSubPixel()`.

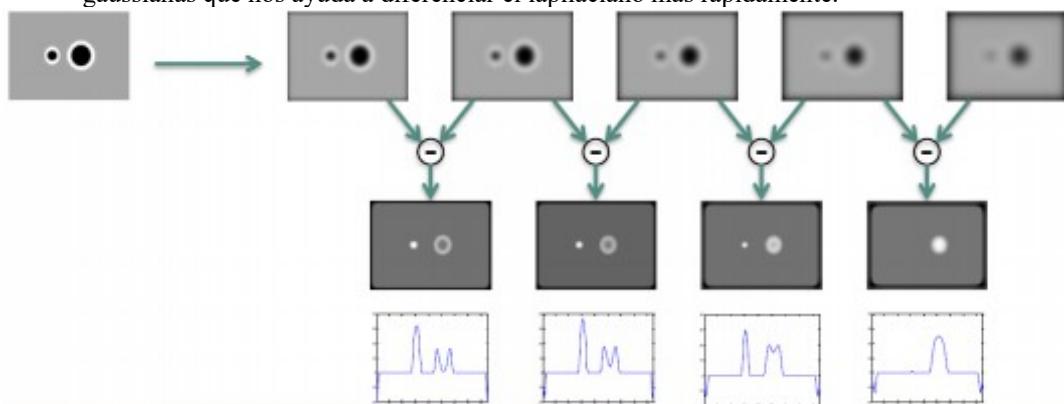
Con estos detectores podían darse el problema de la escala de detección, ya que si obtenemos los puntos característicos con una baja resolución al ampliar la imagen abarcaría regiones de la imagen grandes, y eso no es de utilidad. Igualmente, Harris calcula muy buenos puntos para la imagen a la resolución que esté, pero veamos ahora técnicas de detección más complejas y exactas.

- Detector SIFT, laplaciano. El detector SIFT usa el laplaciano para detectar regiones

quieres trabajar
en Wuolah??

TE BUSCAMOS

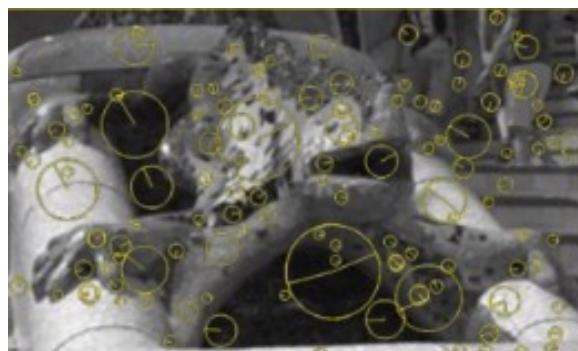
aisladas con cambios claro/oscuro alrededor en todas las direcciones. Si hacemos el mismo proceso con la imagen con gaussiano también podemos controlar la escala de detección. Al restar la imagen gaussiana con una gaussiana con mayor sigma, obtenemos diferencias de gaussianas que nos ayuda a diferenciar el laplaciano más rápidamente.



sin ánimo
de lucro,
chequea esto:



Los puntos los obtenemos en distintas escalas de la misma imagen y podríamos ver todos los puntos característicos sobre la misma, más grandes o más pequeños dependiendo de la escala utilizada para el cálculo de los mismos.



Este método es más lento que el de Harris.

- Características para descripción de puntos característicos:

- Descriptor SIFT. Ya habiendo detectado el punto, conseguimos un histograma de orientación de 8 direcciones en cada celda de una ventana orientada de 16x16, obteniendo un descriptor final de la concatenación de los histogramas, de 16·8 (128) dimensiones. Es costoso, por lo que se plantean otras alternativas.
- Descriptor BRIEF. Está pensado para ser utilizado en tiempo real. Se selecciona un número de pares de puntos aleatorios en una rejilla del punto característico en cuestión y en base a ellos se genera un string binario donde si el primero es menor que el segundo el valor es 1, al contrario 0. Esto mismo se hace en una segunda imagen de la misma escena, y los puntos característicos con el descriptor parecido serán probablemente los mismos.

tú puedes
ayudarnos a
llevar
WUOLAH
al siguiente
nivel
(o alguien que
conozcas)

Tema 10: Ejemplo. Detección y clasificación de señales de tráfico

10.1 Introducción

Queremos diseñar un detector/clasificador de señales de tráfico. En base a una base de datos de imágenes (dataset) de señales de tráfico, queremos diseñar un clasificador basado en aprendizaje automático que diferencie qué es una señal de tráfico y qué no, y también los distintos tipos de ellas.

10.2 Detección

Para empezar con el desarrollo primero debemos detectar qué es una señal de tráfico y qué no. La extracción de características en este ejemplo la realizamos por medio de histogramas de LBP y LBP-U, para conseguir el patrón de la textura de las diferentes imágenes.

Para impedir que los histogramas lleguen a conclusiones erróneas, ya que la acumulación de valores se realiza igual sea la imagen original o la imagen partida y puesto el trozo restante debajo, partimos las imágenes en varios trozos (definidos por el usuario en forma de nxn celdas) y concatenamos sus histogramas. Así, tenemos la información ordenada.

Ya al tener los histogramas, utilizaremos la diferencia entre los puntos con la distancia chi cuadrado. Ahora ya podemos empezar a buscar posibles candidatos de señales en las imágenes. Este recorrido se hará por medio de una ventana deslizante, y compara el histograma de la ventana con los histogramas de referencia. Sin embargo, nos surge un problema de escala, porque una ventana muy grande no será capaz de detectar una señal lejana de la cámara. Por ello, usamos un tamaño canónico, pequeño, y una estructura de pirámide multi-escala en la que hacemos este proceso en distintas escalas de la misma imagen con el mismo tamaño de ventana.

Debemos ya por último diferenciar lo que son señales de tráfico de otros elementos o fondo. Para ello tenemos que enseñarle al algoritmo muestras positivas y muestras negativas de señales, para que sepa con qué se debe quedar, y con qué no.

También es relevante señalar que, como en varias técnicas ya vistas, se necesita un refinamiento final para quedarnos con la mejor detección de cada señal.

Al final de todo tenemos que evaluar el rendimiento de nuestro algoritmo. La forma más intuitiva es que un experto defina manualmente dónde están las señales (proceso denominado ground truth). Buscamos que la tasa de detección de nuestro algoritmo sea lo mayor posible, para que nuestro detector sea lo mejor posible. La tasa de detección es los aciertos entre el número total de imágenes.

10.3 Clasificación

Ahora queremos saber cuál es la señal. Recopilamos ejemplos de cada clase de señal y aprendemos las características visuales que permitan describirlas y distinguirlas. Volvemos a usar LBP con distribución en rejilla y debemos enseñar al clasificador para que obtenga fronteras entre clases. El clasificador debe aprender y, por tanto, introducimos aquí el siguiente tema, para seguir resolviendo el problema con aprendizaje automático.

Tema 11: Aprendizaje automático enfocado a la visión

11.1 Introducción

El aprendizaje automático consiste en dar a las computadoras la habilidad de aprender sin que sean programadas de manera explícita. Existen 2 tipos de algoritmos de aprendizaje automático:

- No supervisado. No requiere de anotaciones previas. 2 ejemplos muy claros que se pueden nombrar son el uso como agrupamiento o clustering, del cual ya hemos hablado anteriormente, y los algoritmos de reducción de la dimensionalidad. Este último consiste en eliminar dimensiones y características irrelevantes que ocupan espacio y tiempo de computación al algoritmo, o simplificarlas sin perder información.
- Supervisado. Requiere que el programador le anote datos y compruebe los resultados. La tarea más presente en este tipo es la regresión. Predice el valor de las variables respecto a los datos dados por el usuario (dataset). Un dataset es una colección de muestras y anotaciones que nos proporciona la información del problema. Suele contener información para el entrenamiento (train), para la validación (validation) y para el período de pruebas (test). Este es el tipo que nosotros necesitamos en el problema de las señales de tráfico.

11.2 Métricas

Nos ayudan a estimar el rendimiento de nuestro sistema. Una de las más importantes es el ISE o el RMSE, recta que estima los valores predichos incluso sin haberlos probado empíricamente.

Para el caso de clasificadores, tenemos la matriz de confusión, herramienta que nos proporciona

información. Es una tabla de frecuencias de clasificaciones correctas o incorrectas para cada clase. Es más sencillo plantearlo gráficamente. En el ejemplo de abajo, siendo la clase 1 el clasificador predice en este ejemplo en 14 muestras correctamente, y 1 vez falla diciendo que era parte de la clase 2.

En base a esta matriz, podemos deducir métricas como el ratio de reconocimiento por clase y la exactitud.

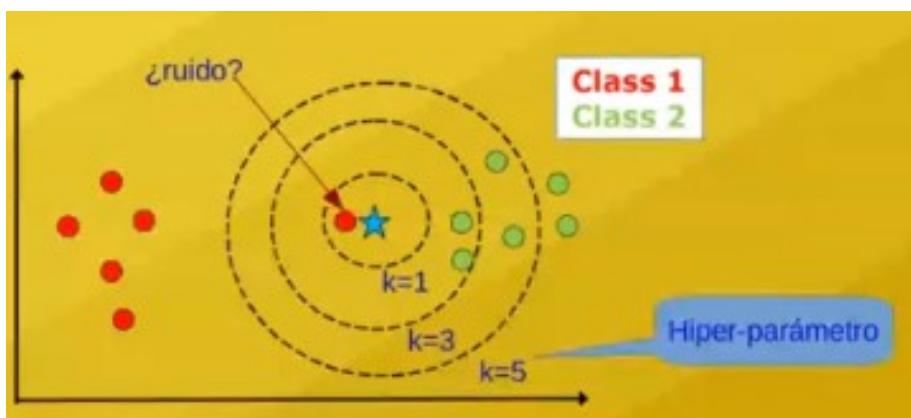
| | P#1 | P#2 | P#3 | Total | Recognition Rate per Class. |
|--|-----|-----|-----|-------|---|
| T#1 | 14 | 1 | 0 | 15 | $R_1 = \frac{14}{14+1+0} \times 100 = 93,3\%$ |
| T#2 | 31 | 15 | 1 | 47 | (Average Recognition Rate) |
| T#3 | 1 | 2 | 12 | 15 | $R = \frac{(R_1 + R_2 + R_3)}{3}$ |
| Accuracy (exactitud) | | | | | $\frac{(93,3 + 31,9 + 80,0)}{3} = 68,4\%$ |
| $Acc = \frac{14+15+12}{15+47+15} = 53,2\%$ | | | | | |

11.3 Principales clasificadores

Trabajamos en OpenCV. Todos los clasificadores de los que vamos a hablar heredan de `cv::ml::StatModel`. Los métodos más destacables son `isTrained()`, `train()`, `predict()` y `save()`.

Clasificadores:

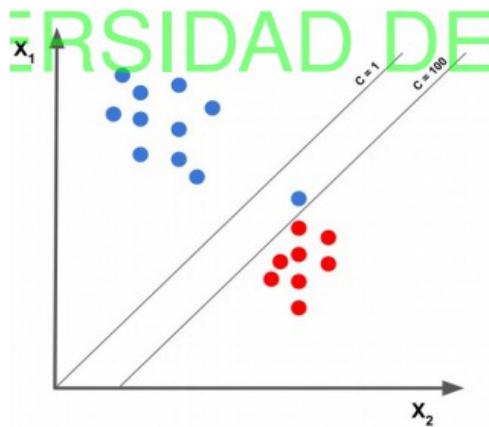
- K-NN (K Nearest Neighbour). Guarda el modelo de entrenamiento y siendo k los k elementos más cercanos elige la clase de la que más elementos hayan dentro.



Implementación:

```
cv::Ptr knn = cv::ml::KNearset::create();
knn->setAlgorithmType(cv::ml::KNearset::BRUTE_FORCE);
knn->setIsClassifier(true);
knn->setDefaultK(knn_K);
```

- SVM (Support Vector Machines). Dadas unas muestras busca seleccionar un hiper-plano que separe las clases con mínimo error y máximo margen.
 - Hiperparámetro C. Permite ajustar el balance entre muestras mal clasificadas y el margen conseguido.



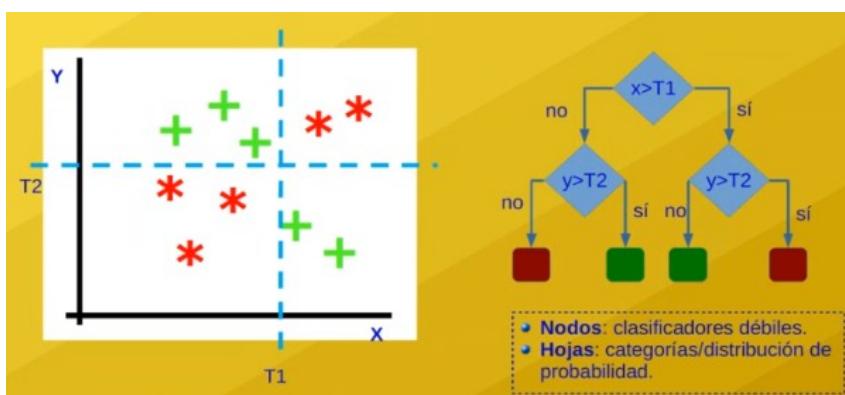
- Hiperparámetro tipo de kernel. No siempre podemos separar las muestras con una función polinómica como una recta, podemos necesitar funciones no lineales. Para ello podemos definir otro tipo de kernel, como RBF, chi cuadrado, sigmoide... y los grados y el gamma si fuera necesario.

Implementación:

```
cv::Ptr<SVM> svm = cv::ml::SVM::create();
svm->setType(cv::ml::SVM::C_SVC);
svm->setKernel(svm_K);
svm->setC(svm_C);
svm->setDegree(svm_D);
svm->setGamma(svm_G);
```

SVM no solo puede funcionar como clasificador binario, si no que también puede funcionar como multiclase. Creamos varios clasificadores y escogemos la clase positiva con mayor margen. Así funciona OpenCV, pero su software lo hace automáticamente, no tenemos que hacer nada como desarrolladores. También se puede solucionar esta situación con un método llamado ECOC (Error Correcting Output Codes), el cual calcula en base a patrones binarios y la mínima distancia de Hamming el posicionamiento de nuevas muestras con posibles pequeños errores.

- Boosting. Entendemos por clasificador débil uno que tiene un error ligeramente por debajo del 50%, y uno fuerte si a infinitas muestras, tendríamos un clasificador fuerte. Se puede construir un clasificador fuerte en base de muchos clasificadores débiles, y eso es a lo que se refiere el boosting. En cada iteración (cada vez que se introduce un clasificador), las muestras mal clasificadas tienen más peso y serán prioritarias para ordenarlas. Un ejemplo es el detector de caras de Viola & Jones. Es un detector de caras usa filtros de Haar, ventanas blancas y negras, indicando 1 si encuentra una cara y 0 en caso contrario. El clasificador fuerte en este caso es una suma de todos los clasificadores con distintas ventanas.
- Random Forest (árboles de decisión). Utiliza boosting. Separamos las muestras con separadores, por ejemplo rectas, y cuando tengamos las 2 clases en sectores independientes hemos resuelto el problema, planteando el siguiente árbol de decisión:



quieres trabajar
en Wuolah??

TE BUSCAMOS

Esto es cuando tenemos unas muestras simples y diferenciadas, pero la realidad no es así y no siempre tenemos solo 2 dimensiones. Por eso, podemos incluir conceptos más interesantes:

- Ensemble. Conjunto de varios clasificadores de tipo árboles, preparados para resolver el mismo problema. La solución problema será la decisión de la mayoría. Usualmente también se denomina Random Forest.
- Bagging. Se unen distintos árboles cada uno especializado en un sector diferenciado del conjunto de entrenamiento y buscamos dejar el que mejor haga una clasificación en específico. Así, cada árbol no es bueno en todo, pero si el mejor en algo determinado, así que todos juntos haciendo cada uno su función cumplen todos los requisitos del sistema.

Implementación:

```
cv::Ptr rtrees = cv::ml::Rtrees::create();
rtrees->setMaxCategories();
rtrees-> setActiveVarCount(sqrt(train.cols));
rtrees->rtrees-> setActiveVarCount(TermCriteria(TermCriteria::MAX_ITERS+TermCriteria::EPS, 50, 0.1));
```

- ANN (Artificial Neural Networks). Tiene la forma de un grafo, y cada nodo es una neurona. A una neurona le llegan una serie de valores de entrada, realizan una función determinada (función de activación), y obtienen una salida. La salida se comunica con otra neurona y así sucesivamente. La introducción de muestras requieren unos pesos, llamados pesos de la red, y existe un método para minimizar las pérdidas por neurona, llamado backpropagation.

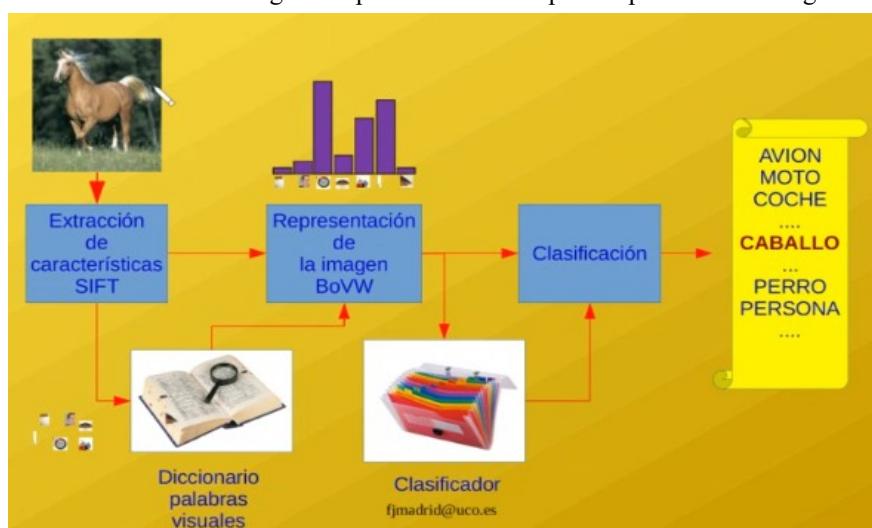
11.4 Categorización de imágenes. BOVW (Bag Of Visual Words)

Se trata de otra forma más de análisis de imágenes, pero buscando introducirle etiquetas de los contenidos de la imagen. Por ejemplo, la etiqueta persona y campo si hay una persona en el campo, o incluso cielo, césped, árboles...

Vamos a utilizar un algoritmo de aprendizaje automático supervisado. Como estamos acostumbrados, tenemos que recopilar ejemplos para dárselas a nuestro algoritmo y aprender características visuales de cada clase para distinguirlas.

Igual que en los casos anteriores tendremos dificultades de iluminación, cambios de posición, tamaño, occlusiones parciales...

En esta parte usaremos un descriptor llamado bolsa de palabras visuales (BOVW). Se basa en una técnica inicialmente utilizada para clasificación de documentos, en la que se recopilaban palabras claves de un documento y si aparecían muchas veces palabras como campo, jugador, balón, partido... se encasillaba en la categoría deportes. Este concepto lo aplicamos en imágenes.



sin ánimo
de lucro,
chequea esto:



tú puedes
ayudarnos a
llevar
WUOLAH
al siguiente
nivel
(o alguien que
conozcas)

Como podemos ver en la ilustración usamos el detector SIFT, comparamos con nuestra referencia, el diccionario, representamos y entrenamos a nuestro clasificador.

Ya conseguidos los descriptores de 128 valores calculados con los métodos SIFT (detector y descriptor) explicados previamente, construimos nuestro diccionario usando k-means. Podemos pensar que, si en nuestro diccionario recopilamos muchas imágenes de personas, e intentamos buscar en el diccionario otra persona no introducida, dentro de los descriptores habrá algún punto característico respectivo, por ejemplo, a los ojos. Podemos suponer además que un punto característico de un ojo es parecido al de todos los ojos, así que ya tenemos una pista sobre qué puede ser la imagen. La comparación en estos casos se realiza por medio de distancia euclídea o con la distancia chi cuadrado. A la hora de clasificar, se suele usar el clasificador K-NN.

Tema 12: Análisis de vídeos: descriptores espacio-temporales

12.1 Introducción

Los motores de búsqueda actuales ya están preparados para reconocer acciones como abrazos, choques de manos... ¿Cómo hemos llegado a hasta este punto? Igual que esta utilidad el análisis de vídeo también sirve para video vigilancia, conducción asistida, análisis biomecánico, estadísticas deportivas...

12.2 Punto de interés espacio-temporal (STIP)

Ya sabíamos obtener puntos característicos, y estas ideas se pueden extender al 3D. Buscamos cambios bruscos en espacio-tiempo. Para calcularlos usamos un método muy afín al de Harris en el caso de imágenes:

- Utilizamos convolución 3D para calcular gradientes.
- Calcular la matriz donde g es una gaussiana 3D.

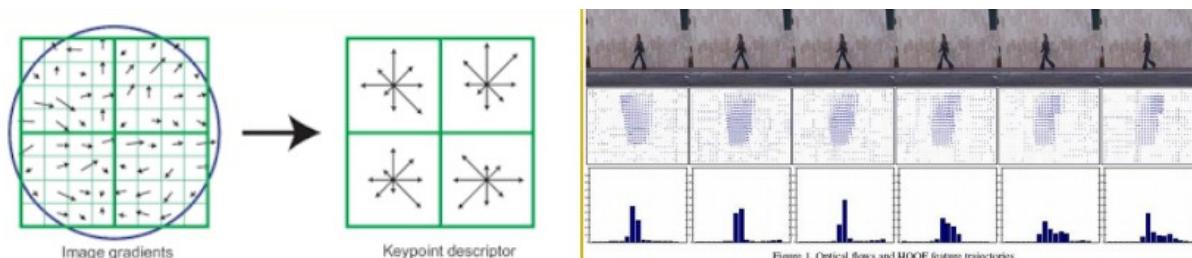
$$\mu = g(\cdot; \sigma_i^2, \tau_i^2) * \begin{pmatrix} L_x^2 & L_x L_y & L_x L_t \\ L_x L_y & L_y^2 & L_y L_t \\ L_x L_t & L_y L_t & L_t^2 \end{pmatrix}$$

- Calcular los autovectores de la matriz en base a un determinante.

$$H = \det(\mu) - k \operatorname{trace}^3(\mu) = \lambda_1 \lambda_2 \lambda_3 - k(\lambda_1 + \lambda_2 + \lambda_3)^3$$

12.3 Descriptores para STIP

Buscamos describir los sucesos que ocurren en el espacio y el tiempo. Para describir el espacio tenemos el histograma de gradientes (HOG) y para describir en el tiempo el histograma de flujo óptico (HOF).



Para su cálculo se divide el vídeo 3D en una rejilla de cubos y para cada uno se calcula los descriptores. Después se concatenan los histogramas normalizados.

Para analizar después las acciones tenemos a nuestra disposición todas las técnicas de imágenes, como la idea de BOVW o incluso clasificadores SVM o Random Trees.

12.4 Otros descriptores

- Track de un punto. Secuencia temporal de las coordenadas de un punto visual, consiguiendo la trayectoria. En el caso de que se combinen con muchos del mismo objeto podemos conseguir algún tipo de descriptor, como HOG/HOF o descriptor de la forma de trayectoria.
- Motion History Image. Se trata de un descriptor holístico, es decir, usa integrales. Para cada fotograma de la secuencia obtiene la silueta en blanco, y si conforme va pasando el tiempo ese punto deja de formar parte de la silueta se le va restando tonos y tiende a escala de grises. Con esto, la silueta del fotograma actual está blanca y el cambio en forma de movimiento de los fotogramas anteriores en escala de grises.
- Gait Energy Image. Es otro ejemplo de descriptor holístico. Consiste en, de nuevo, obtener la silueta de distintos fotogramas para después sumar todos los fotogramas, concluyendo en una imagen con tonos blancos (zonas estáticas) y grises (parte de la silueta en movimiento). Es un método que avanzó mucho el estudio de cómo camina cada persona.

Tema 13: Nuevas tendencias en análisis de imágenes y videos

13.1 Introducción

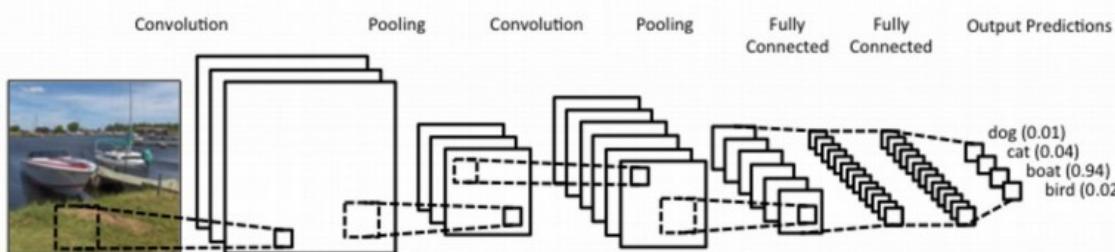
Como hemos visto anteriormente la extracción de características se hace de forma manual, pero estamos tendiendo a que el sistema aprenda automáticamente las características a extraer. Este proceso se realiza con aprendizaje profundo (Deep Learning) y redes de neuronas convolucionales.

13.2 Aprendizaje profundo

Funciona de manera afín con las redes neuronales del aprendizaje automático, pero siendo los pesos de las conexiones aprendidos durante el entrenamiento. Además, gracias a la tecnología que tenemos en la actualidad, conseguimos redes neuronales con una gran cantidad de capas. De ahí proviene el concepto de aprendizaje profundo. Una de las aplicaciones más sonadas en los últimos años son las aplicaciones para envejecer caras y ya las máquinas han superado a los humanos en eficacia de caracterización de imágenes.

13.3 Redes de neuronas convolucionales

Son redes profundas con capas convolucionales. Los filtros son aprendidos de forma automática e igual que en los casos anteriores, el algoritmo necesita aprender los pesos de todas las neuronas. Los pesos de convolución coinciden con los pesos de las uniones entre neuronas.



13.4 Aplicaciones

- Detección de objetos.
- Leer los labios.
- Descripción de imágenes.
- Análisis de imagen médica.
- Coches autónomos.
- Reconocimiento de textos.
- Estimación de la pose de la cabeza.