Trabajo Final Inteligencia Artificial I – año 2018: Visión Artifical

Fernández, Gonzalo Gabriel

12 de diciembre de 2018

Resumen

El presente trabajo sobre visión artificial fue realizado como trabajo final de la cátedra Inteligencia Artificial I, de la Facultad de Ingeniería de la Universidad Nacional de Cuyo.

Consiste en el diseño e implementación de un agente capaz de clasificar frutas en base a fotografías. Se realiza un breve estudio teórico del agente. En la implementación se analiza el procesado de imágenes en sus diferentes etápas: filtrado, segmentado y extracción de características. Para clasificar las imágenes se utilizan algoritmos de aprendizaje K-means y K-nn, y se hace un estudio de desempeño comparativo entre los dos algoritmos. El lenguaje de programación utilizado a lo largo del proyecto es Python.

Como objetivo final se desea obtener un agente que en un entorno con condiciones controladas se desempeñe con un rendimiento por encima de las exigencias.

1. Introducción

1.1. Visión Artificial

"*La **visión artificial** o visión por ordenador es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por un ordenador.*" de Wikipedia, la Enciclopedia Libre. "*En la visión por ordenador, intentamos describir el mundo que vemos en una o más imágenes e intentamos reconstruir sus propiedades, tal como la forma, iluminación y distribuciones de color.*" Szeliski [3]

1.2. Problema a resolver

Se está desarrollando un **sistema de reconocimiento de frutas por visión artificial** con la intención de agilizar el proceso de cobro en las cajas de un supermercado. Para ello se pide desarrollar un agente prototipo que pueda **reconocer bananas, naranjas y limones** a partir de fotografías.

Las condiciones impuestas son que las **imágenes deben tomarse en escala de grises**, y que debe utilizarse los **algoritmos K-means y K-nn** para realizar la clasificación, y a partir de los resultados obtenidos sugerir uno de ellos para ser implementado.

Es recomendado una extracción de características mediante la aplicación de un análisis textural a través de una matriz de covarianza con un vector de características que se construya a partir de los autovalores de dicha matriz.

2. Especificación del agente

La descripción del agente y su entorno se realiza en base a la clasificación de Peter Norving [2], Capítulo 1.

2.1. Tipo de agente

El agente es **racional**: "*En cada posible secuencia de percepciones, un agente racional deberá emprender aquella acción que supuestamente maximice su medida de rendimiento, basándose en las evidencias aportadas por la secuencia de percepciones y en el conocimiento que el agente mantiene almacenado.*" Peter Norving [2]. Es **no omnisciente**, ya que no conoce el resultado de sus acciones, no posee total información de todo su entorno. El objetivo de utilizar algoritmos como K-means o K-nn es que el agente **aprenda** a lo largo de sus percepciones, y también que sea **autónomo**, y pueda apoyarse más en la experiencia adquirida gracias a sus propias percepciones que en el conocimiento inicial proveído por el diseñador.

El agente es un **agente que aprende**, debido al tipo de algoritmos que utiliza. La base de datos provee un "modelo"del mundo, y los algoritmo también actúan en base a una ütilidad"que se analiza más adelante.

2.2. Tabla REAS: Rendimiento, Entorno, Actuadores, Sensores

- **Tipo de Agente:**

  Computadora, microcontrolador o microprocesador en el cuál se encuentran los algoritmos entrenados, y los dispositivos de *entrada y salida*, o más correcto en éste análisis, *sensores y actuadores*: cámara fotográfica, monitor, etc.

- **Medidas de Rendimiento:**

  A la hora de implementar los algoritmos, se posee una base de datos con toda la información recopilada etiquetada por clase, en este caso fotografías de bananas, naranjas y limones. A esta base de datos se la divide en dos conjuntos: entrenamiento y testeo. El conjunto de testeo nos permitirá dar una medida de rendimiento del agente, siendo esta medida la cantidad de predicciones correctas de todos los elementos de testeo.

- **Entorno:**

  El entorno del agente será el área donde esté ubicada la fruta a la hora de tomar la fotografía, y la fruta en sí. Al entorno también lo integran todos los factores externos que se exponen más adelante, como luminosidad, fondo, calidad de la fruta, etc.

- **Actuadores:**

  El actuador del agente es sencillamente el medio por el que éste se comunicará con el usuario para darle a conocer su predicción. Puede ser un display, una consola, una señal luminosa o auditiva, o cualquier otro medio que resulte adecuado.

- **Sensores:**

  El sensor del agente es la cámara fotográfica que tomará la imagen de la fruta. Esto es así, si se considera la cámara fotográfica como parte del agente. Sin embargo, la imagen puede provenir de otras fuentes exteriores que no se consideren parte del agente. En este caso, podría considerarse como sensor las herramientas (librerías y sus funciones) que nos permiten procesar la imagen en el formato que se reciba (*PNG, JPEG*), en un formato de información con la que el agente puede operar.

2.3. Propiedades del entorno de trabajo

- **Parcialmente observable:**

  El entorno es parcialmente observable, ya que el agente está recibiendo una representación del mundo real, en dos dimensiones; una imagen que describe el entorno mediante un arreglo discreto de pixeles con un determinado valor. Esta limitación está asociada a lo explicado en la introducción, sección 1.1.

- **Estocástico:**

  Que el entorno sea parcialmente observable hace que desde el punto de vista del agente, el entorno parezca en gran medida estocástico. Debido a la gran diferencia entre la representación del entorno en la imagen y el mundo real, el agente nunca puede estar seguro de los hechos en el exterior.

- **Episódico:**

  El entorno es episódico ya que el estado del entorno en un determinado instante de tiempo no depende de la secuencia de estados previos. La cámara fotográfica tomará imágenes de las frutas en el orden que se le presenten, y este orden no tiene ninguna inferencia en el desempeño del agente.

- **Estático:**

  El entorno no cambia mientras el agente está deliberando. Una vez tomada la fotografía, comenzado el procesamiento el agente se aísla en cierto modo del entorno.

- **Discreto:**

  Que el entorno sea discreto está relacionado con que también es estático. El agente no interactúa en tiempo real con el entorno, sino solo en el corto lapso en que toma la fotografía.

- **Agente individual:**

  El entorno es agente individual ya que en todo el proceso de interacción agente-entorno, el único ente racional es el agente.

3. Diseño del agente

Se inicia el proyecto eligiendo con que lenguaje de programación se encarará. Se decide utilizar ***Python***, por ser tan popular en las disciplinas relacionadas con inteligencia artificial mundialmente. Entre las opciones, además de *Python* se contempló utilizar *Matlab* o *R*.

Siendo la primer experiencia con análisis de imágenes, la primer interrogante que surge es cómo obtener de una imagen cruda de la fruta, un vector representativo de todas sus características que sea de utilidad como entrada a los algoritmos K-means y K-nn.

En base a distintas referencias, se decide dividir el problema con cuatro etapas principales: **representación de la imagen** en un arreglo de números fáciles de computar, **filtrar** la imagen para eliminar ruido, **segmentar** la imagen para resaltar características que puede ser relevantes para el posterior análisis y la **extracción de características**, donde como resultado se obtendría un vector representativo de la imagen capaz de ser una de las tantas entradas a los algoritmos antes mencionados, Van Der Walt [4].

Las librerías que se utilizaron para resolver el problema propuesto son *SciPy* y *OpenCV*.

SciPy es un ecosistema basado en *Python* de software *open-source* para matemática, ciencia e ingeniería. Los paquetes que se utilizaron específicamente en este trabajo son:

- NumPy: Es un paquete para computación científica. Contiene herramientas como arreglos n-dimensionales, funciones sofisticadas, herramientas para integrar *C/C++* y *Fortran*, álgebra linear, transformada de Fourier, operaciones random, etc.
- Matplotlib: Es una librería para gráficos 2D con calidad de publicación.
- IPython: Provee una arquitectura para computación interactiva con una potente consola, un kernel para *Jupyter*, visualización de datos interactiva, etc.

OpenCV (Open Source Computer Vision Library), es una librería de funciones para *C++*, *Python* y otros lenguajes, donde su propósito principal es la visión artificial en tiempo real.

Para exponer el código y la investigación realizada de una forma clara, sensilla y reproducible, se utilizó Jupyter. Jupyter es una aplicación web open-source que permite crear y compartir documentos con bloques de código, ecuaciones, gráficos y texto descriptivo.

3.1. Etapas de la implementación

La descripción más específica y técnica de la implementación se encuentra en los diferentes *Jupyter notebook* del proyecto. Para acceder a ellos, ir al link provisto correspondiente.

3.1.1. Representación adecuada de la información

Jupyter notebook: 01-representacion. En el caso de no disponer del proyecto completo: link de GitHub.

Al analizar la conversión a escala de grises se utiliza la fórmula:

$$Y'_{709} = 0{,}2125R' + 0{,}7154G' + 0{,}0721B' \tag{1}$$

La ecuación 1 se obtiene de Szeliski [3], capitulo 2 “Image Formation”, 2.3 “The digial camera”.

El resultado obtenido en esta etapa es la imagen en escala de grises en un tamaño normalizado, a partir de la fotografía original recibida por el agente. Se puede observar un ejemplo en la figura 1. El proceso se puede hacer por medio de la librería *skimage* o también con la librería *OpenCV* (más específicamente *cv2*).

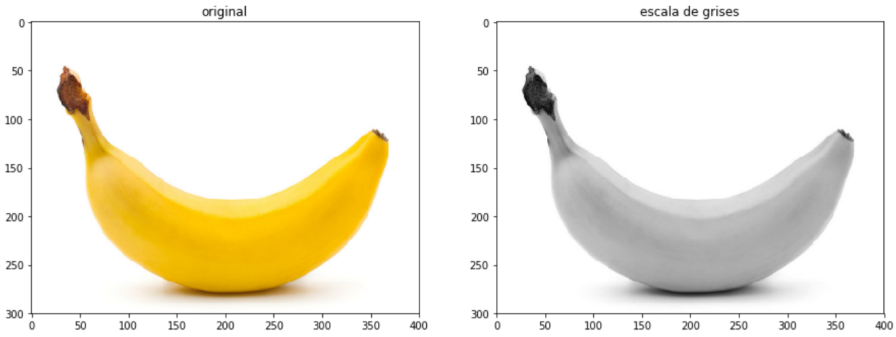

Figura 1: Resultado obtenido en *Jupyter notebook* 01-representacion.

3.1.2. Filtrado de imágenes

Jupyter notebook: 02-filtracion. En el caso de no disponer del proyecto completo: link de GitHub.

De *Wikipedia, la Enciclopedia Libre*: "Es el conjunto de técnicas englobadas dentro del preprocesamiento de imágenes cuyo objetivo fundamental es obtener, a partir de una imagen origen, otra final cuyo resultado sea más adecuado para una aplicación específica mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella. Los principales objetivos que se persiguen con la aplicación de filtros son:

- Suavizar la imagen: reducir la cantidad de variaciones de intensidad entre píxeles vecinos.
- Eliminar ruido: eliminar aquellos píxeles cuyo nivel de intensidad es muy diferente al de sus vecinos y cuyo origen puede estar tanto en el proceso de adquisición de la imagen como en el de transmisión.
- Realzar bordes: destacar los bordes que se localizan en una imagen.
- Detectar bordes: detectar los pixeles donde se produce un cambio brusco en la función intensidad.

Por tanto, se consideran los filtros como operaciones que se aplican a los pixeles de una imagen digital para optimizarla, enfatizar cierta información o conseguir un efecto especial en ella.".

En este trabajo se estudiaron los siguientes filtros:

- *Gaussian*: Altamente recomendado y utilizado ampliamente para eliminar ruido, reducir detalle de las fotografías, y como una etapa de preprocesamiento de imágenes.
- *Diferencial*: Se utiliza para obtener los cambios en la intensidad de la imagen. El resultado es el gradiente, en la dirección que se realizó el filtro, de la intensidad de la imagen. Es un primer paso al detectar bordes de la imagen.
- *Sobel*: Es un filtro muy utilizado para la detección de bordes. Es una aproximación al gradiente de la imagen. En el trabajo se realiza una comparación entre el filtro *Sobel* y la composición de dos filtros *diferenciales* ortogonales, obteniendo resultados similares. En la librería *OpenCV* a esta composición se le denomina filtro *Laplacian*, por estar relacionado con la ecuación de Laplace.
- *Median*: Este filtro a diferencia de los anteriores es no lineal. Se utiliza de forma similar que el filtro *Gaussian*, para reducir ruido en las imágenes.

Los resultados obtenidos en esta etapa es la imagen en escala de grises (o no) en un tamaño normalizado, filtrada por *Gaussian* y/o *Sobel* dependiendo la necesidad, a partir de la fotografía original recibida por el agente. Se puede observar un ejemplo en la figura 2. El proceso se puede hacer por medio de la librería *skimage* o también con la librería *OpenCV* (más específicamente *cv2*).

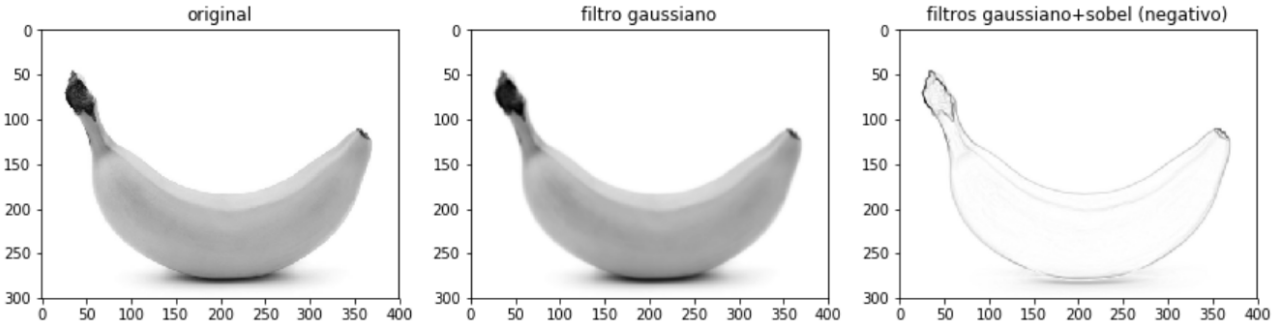

Figura 2: Resultado obtenido en *Jupyter notebook* 02-filtracion.

3.1.3. Segmentado de imágenes

Jupyter notebook: 03-segmentacion. En el caso de no disponer del proyecto completo: link de GitHub.

El resultado del proceso de segmentación, es una imagen binaria que contendría los bordes del objeto a analizar. Esto puede utilizarse o no, según el método de extracción de características que se utilice. Se puede observar un ejemplo en la figura **??**. El proceso se puede hacer por medio de la librería *skimage* o también con la librería *OpenCV* (más específicamente *cv2*).

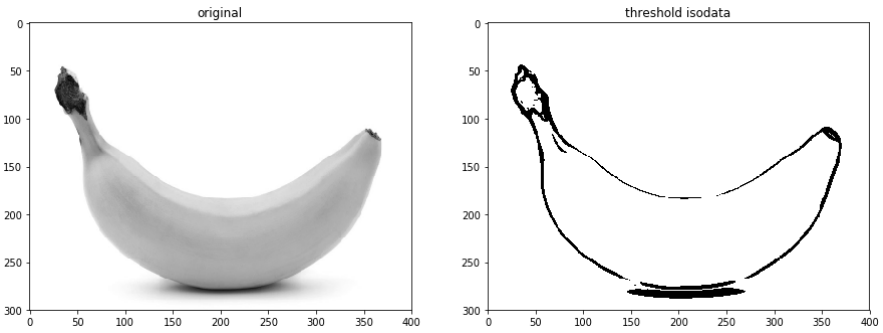

Figura 3: Resultado obtenido en *Jupyter notebook* 03-segmentacion.

3.1.4. Extracción de características

Jupyter notebook: 04-extraccion. En el caso de no disponer del proyecto completo: link de GitHub.
Para la extracción de características se utilizaron diferentes métodos:

- **Histogram of oriented gradients (HOG):** Es utilizado en visión por computador y procesado de imágenes con el propósito de detección de objetos. La técnica cuenta la ocurrencia de orientaciones del gradiente en porciones localizadas de la imagen. Este método es similar a los histogramas de orientación de bordes, transformación de características invariante en escala, y contextos de forma, pero se diferencia en que se calcula una cuadrícula densa de celdas espaciadas uniformemente y utiliza normalización de contraste local superpuesto para una mayor precisión. Resultados obtenidos (reducidos a media y desviación estándar) en la figura 4.

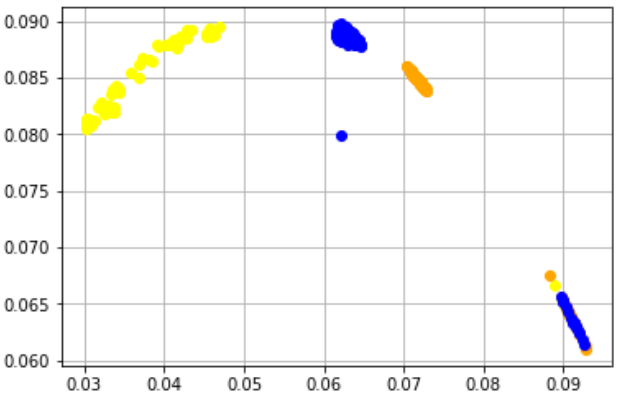

Figura 4: Histogram of Oriented Gradients (HOG) para la base de datos, reducida a media y desviación estándar.

- **Hu Moments:** Generalmente es utilizado para la extracción de la silueta o contorno del objeto en la imagen. Da una valorización de la forma de éste, que puede ser medida comparando vectores de características *Hu Moments* a través de distancias, por ejemplo. Resultados obtenidos (reducidos a media y desviación estándar) en la figura 5.
- **Haralick Textures:** Las características de textura están basadas en estadísticas que resumen la distribución de frecuencia relativa (que describe cuán a menudo un tono gris aparece en una zona espacial específica en relación a otro tono de gris en la imagen), Haralick [1]. Resultados obtenidos (reducidos a media y desviación estándar) en la figura 6.
- **Color Histogram:** Es sencillamente una representación de la distribución de colores en la imágen. En imágenes digitales, el histograma de color representa el número de pixeles que tienen determinado color en una lista fija con el rango de colores. Resultados obtenidos (reducidos a media y desviación estándar) en la figura 7.

Referencias

[1] Robert M. Haralick. “Textural Features for Image Classification”. En: (1973).

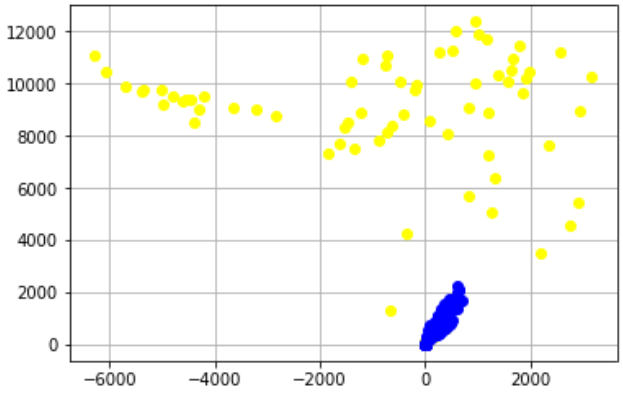

Figura 5: Hu Moments para la base de datos, reducida a media y desviación estándar.

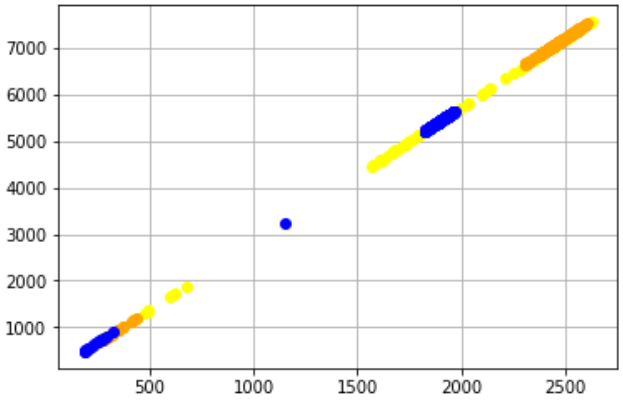

Figura 6: Haralick Textures para la base de datos, reducida a media y desviación estándar.

[2] Stuart Rusell y Peter Norving. *Inteligencia Artificial: Un Enfoque Moderno*. Pearson Educación S. A., 2004.

[3] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[4] Stefan Van Der Walt. *Image Analysis in Python with SciPy and scikit-image — SciPy 2018 Tutorial — Stefan van der Walt*. 2018. URL: `https://www.youtube.com/watch?v=arXiv-TM7DY&t=4343s&list=LLbi4i-j4bBUgaeJvDIn96aw&index=27` (visitado 12-07-2018).

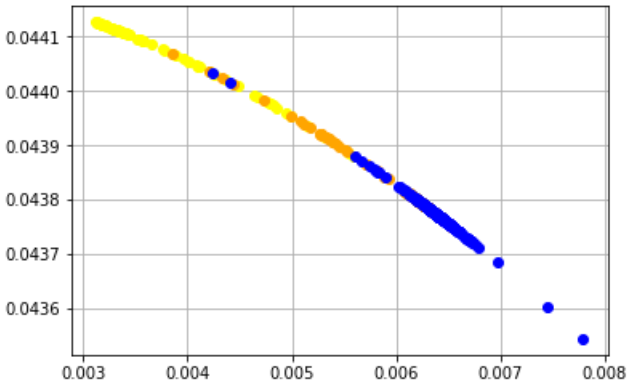

Figura 7: Color Histogram para la base de datos, reducida a media y desviación estándar.

Índice