**Práctica Flujo Óptico**

**Orión García Gallardo**

Esta práctica ha consistido en la implementación de dos algoritmos para la detección de flujo óptico en video o en varias imágenes. Los algoritmos implementados ha sido Lucas-Kanade y Horn-Schunck.

En primer lugar, hablando del algoritmo de Lucas-Kanade hay que decir que se han realizado dos implementaciones distintas aunque el resultado final es el mismo sólo cambia el orden de las operaciones. Sea [u, v] la velocidad del movimiento en un determinado punto [x, y], el movimiento se puede definir por el algoritmo de Lucas-Kanade según las fórmulas descritas en la tabla 1 o tabla 2, según lo visto en clase.

|  |
| --- |
|  |
| Tabla 1 (implementado en LucasKanade1.m) |

|  |
| --- |
|  |
| Tabla 2 (implementado en LucasKanade2.m) |

Donde se tienen las siguientes definiciones:

|  |
| --- |
|  |
|  |
|  |

El sumatorio de i se hará sobre una ventana seleccionada. Un valor bueno de ventana podría ser 5x5.

Por otro lado el algoritmo de Horn-Schunck sólo se ha realizado una implementación. Este es un algoritmo iterativo cuyo objetivo es ir minimizando el error de las siguientes ecuaciones:

|  |
| --- |
|  |
|  |

Donde las variables tienen el mismo significado que en el algoritmo de Lucas-Kanade. Se puede observar que hay una variable que falta por definir, λ.Este parámetro actúa como ponderación de ambas magnitudes de error. De tal manera que λ es grande si la imagen tiene mucho error y bajo si contiene mucho ruido.

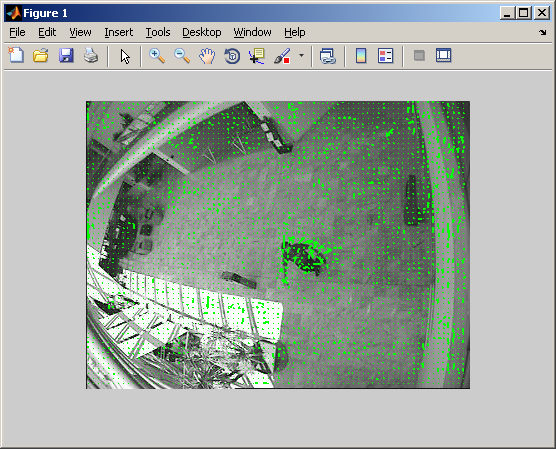
Este algoritmo no tiene un tamaño de ventana. Sin embargo, para hacer más fácil la visión del resultado al usuario se ha creado un tamaño, de tal manera que el pixel central de la ventana mostrará el valor medio de lo pixeles vecinos.

**Resultados**

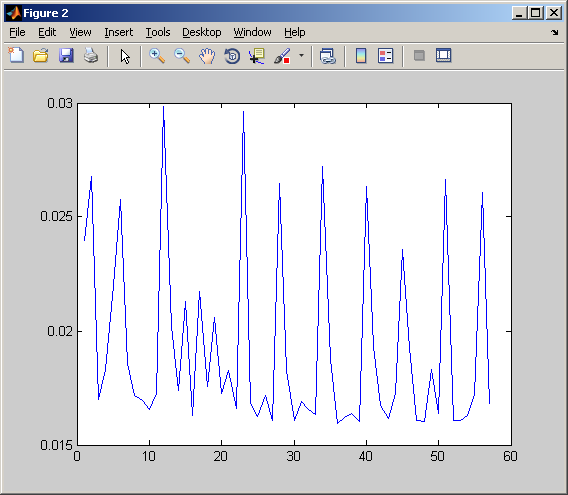
Para la realización de estas pruebas se ha utilizado el “CAVIAR Test Case Scenarios” disponibles en <http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/> en concreto el conjunto de imágenes de “One person walking – straight line and return”.

Empleado un temaño de bloque de 5x5

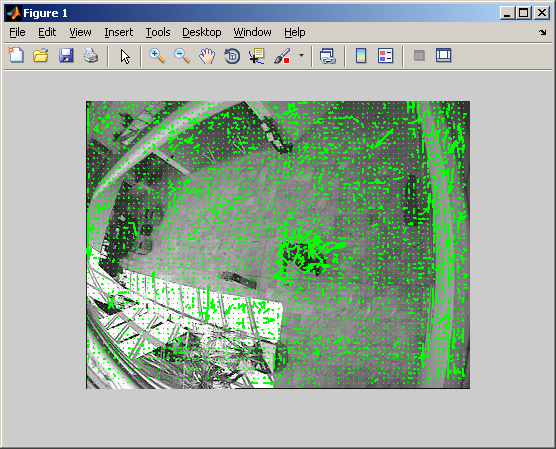
Lucas-Kanade primera implementación (Tiempo máximo total = 1.03 seg):

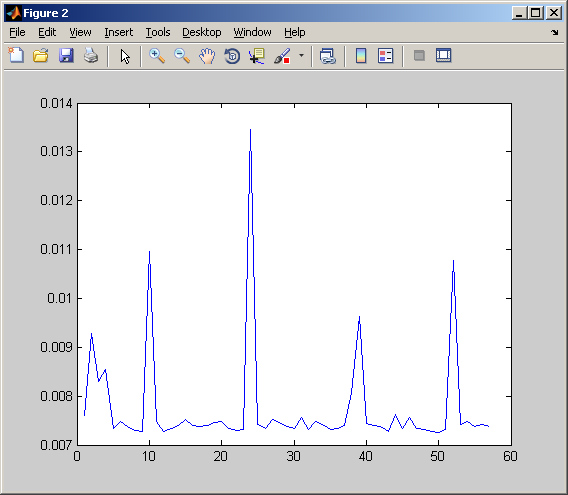


Tiempo por iteración (a nivel fila):

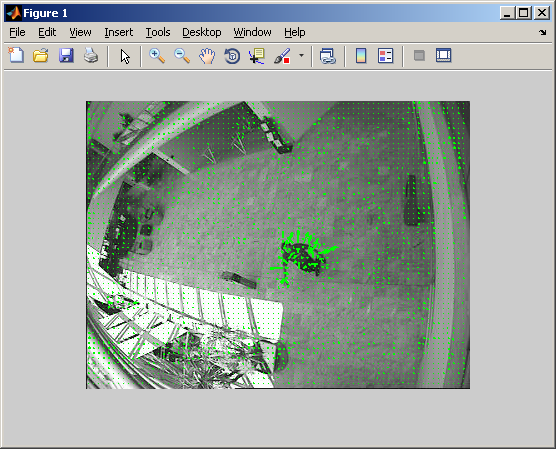


Lucas-Kanade segunda implementación (Tiempo máximo total = 0.44 seg):

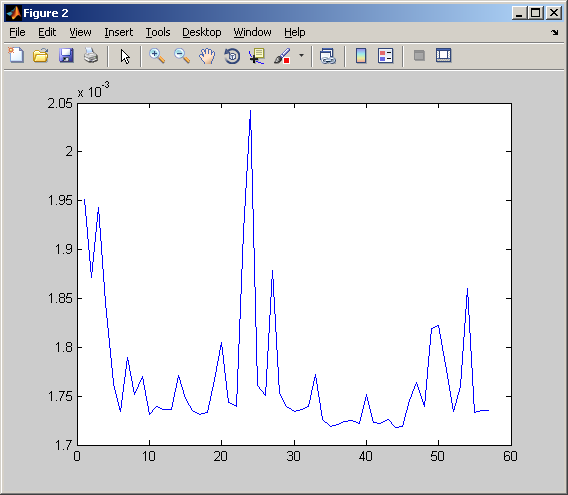


Tiempo por iteración (a nivel fila):

Horn-Schunck segunda implementación (Tiempo máximo total = 0.23 seg):

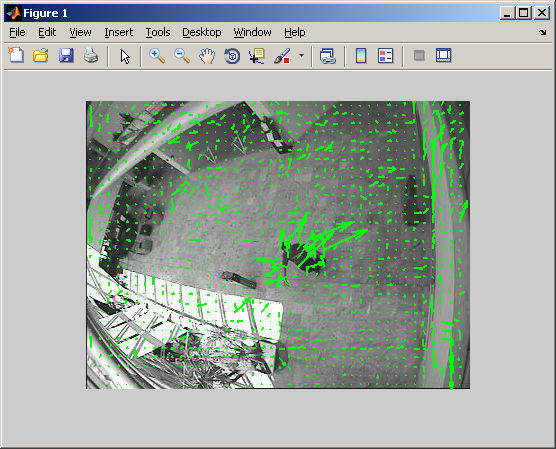


Tiempo por iteración (a nivel fila):

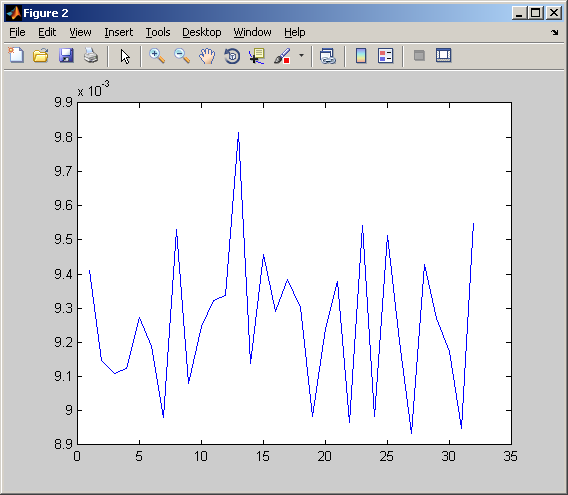


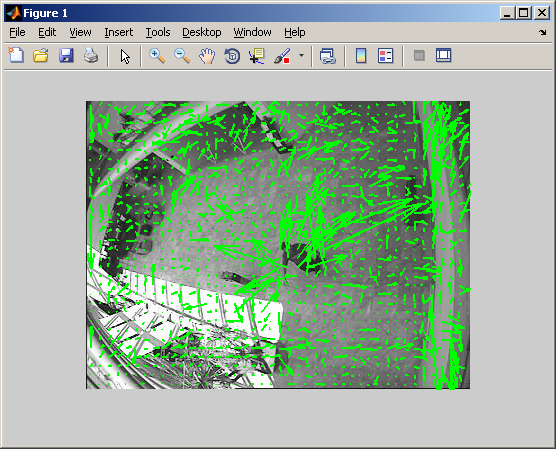
Empleado un temaño de bloque de 9x9

Lucas-Kanade primera implementación (Tiempo máximo total = 0.3 seg):

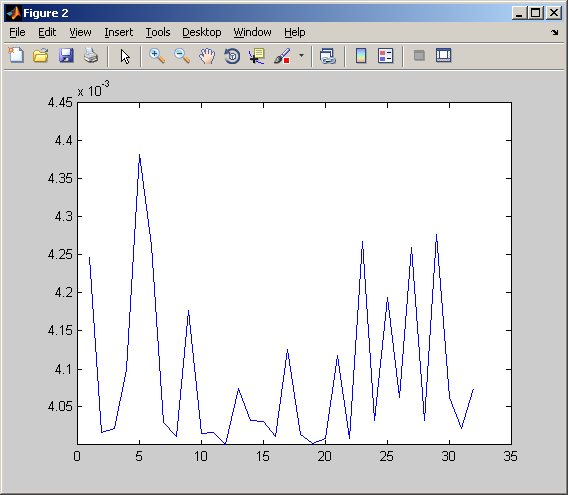


Tiempo por iteración (a nivel fila):

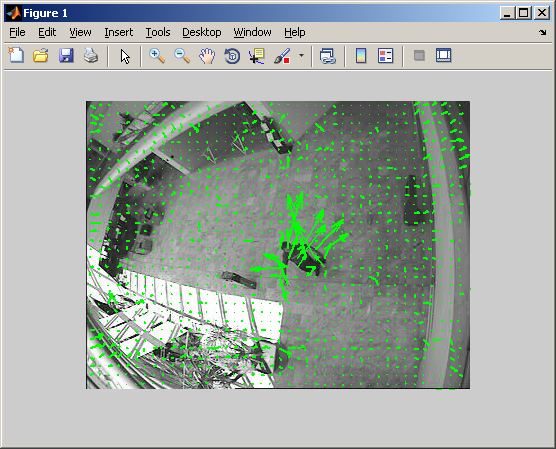


Lucas-Kanade segunda implementación (Tiempo máximo total = 0.13 seg):

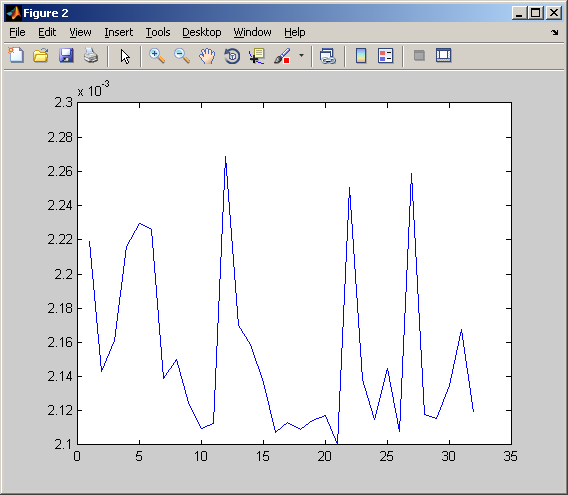
Tiempo por iteración (a nivel fila):



Horn-Schunck segunda implementación (Tiempo máximo total = 0.06 seg):



Tiempo por iteración (a nivel fila):



**Conclusiones**

Como podemos observar de las pruebas realizadas el método que obtiene los mejores resultados es el de Horn-Schunck. Horn-Schunck no sólo tarda menos tiempo en ejecutarse sino que además nos da un resultado más eficiente y menos ruidoso. Comparando las implementaciones de Lucas-Kanade entre sí podemos observar que aunque la segunda de las implementaciones tarda menos en ejecutarse (menos de la mitad que la primera) los resultados obtenidos contienen mucho ruido.