[[1]](#footnote-1)

Análisis del Movimiento Armónico Simple en un péndulo

Santiago Correa Marulanda, Juan Esteban Aristizabal Aristizabal.

Resumen -Este laboratorio presentan métodos de procesamiento digital de imágenes aplicados al análisis de un fenómeno físico, con el objetivo de estimar la velocidad, aceleración y oscilaciones para comprobar si el sistema sigue un Movimiento Armónico Simple. Se utilizaron técnicas en Python con apoyo de las librerías OpenCV, Matplotlib y Numpy. Se aplicaron técnicas y conceptos matemáticos como diferenciaciones numéricas, suavizado de derivadas y convolución para eliminar la mayor cantidad de ruido de los datos experimentales, posteriormente se graficaron los vectores velocidad y aceleración junto con las sus graficas contra el tiempo, finalmente se comparan los resultados del entorno experimental y el modelo teórico y así se verificó la coherencia de los resultados.

**Índice de Términos – Movimiento Armónico Simple, OpenCV, Procesamiento digital de imágenes, estimación, velocidad, aceleración**

# introducción

El estudio del Movimiento Armónico Simple es una pieza importante en la física clásica o mecánica, ya que ayuda a modelar sistemas oscilatorios como movimientos pendulares, circulares, y resortes. En el pasado llegar a un modelo experimental de estos era más complejo que lo que es hoy en día, debido a las ventajas otorgadas por la tecnología y, en este caso, al procesamiento digital de imágenes.

En este laboratorio se explora el uso de las técnicas de procesamiento digital de imágenes para analizar el comportamiento de un sistema oscilatorio a partir de un video en condiciones controladas, como, por ejemplo, un fondo homogéneo que resalte del objeto o péndulo. Para ello se emplearon herramientas como OpenCv para capturar las imágenes, detección del objeto y operaciones morfológicas con la intención de facilitar la obtención de los datos, Numpy para el tratamiento de los datos y métodos matemáticos, y Matplotlib para la representación gráfica de los datos en función del tiempo. El objetivo principal es estimar las componentes cinemáticas del sistema, es decir, la posición, velocidad y aceleración y determinar si el movimiento observado se ajusta al sistema teórico de un Movimiento Armónico Simple. Para mejorar la precisión de los resultados se aplicaron conceptos matemáticos como diferenciación numérica, suavizado de derivadas y convolución.

Esta propuesta no solo automatiza el análisis experimental de un sistema pendular, sino que también permite poner en práctica todos los conocimientos adquiridos de la primera unidad del curso procesamiento digital de imágenes y también aplicar los conocimientos adquiridos en anteriores asignaturas como lo es el concepto físico de la aceleración y velocidad, derivadas, métodos numéricos, convolución, y así mismo, una comprensión más intuitiva del comportamiento de sistemas físicos reales y tratarlos al modelarlos computacionalmente.

# Fundamento teórico

## Movimiento Armónico Simple (MAS)

Un movimiento Armónico Simple es un sistema que describe un tipo especifico de movimiento oscilatorio que se caracteriza por presentar una aceleración proporcional y de sentido opuesto al desplazamiento del objeto respecto a una posición de equilibrio. Este concepto tiene aplicaciones como resortes, orbitas y en este caso, péndulos. Para entender y analizar el comportamiento de este fenómeno, se deben recordar algunos conceptos físicos a utilizar durante el desarrollo del trabajo, empezando con las componentes cinemáticas posición, velocidad y aceleración.

Posición :

Representa la distancia del objeto respecto a su punto de equilibrio en función del tiempo, Físicamente la posición se define como:

Donde:

A es la amplitud del movimiento.

es la frecuencia angular del sistema.

es la fase inicial.

En este laboratorio no se utilizó esta ecuación ideal, sino que se utilizó un modelo experimental, debido a que se trabaja en un medio discreto (capturar y registrar frame a frame la posición), se hizo uso una versión simplificada: , , donde la posición nos dará una coordenada en x al igual que una coordenada en y para luego ser derivadas cada una.

Velocidad :

Es la derivada de la posición con respecto al tiempo, e indica la rapidez y dirección con la que cambia la posición. Matemáticamente:

Aceleración :

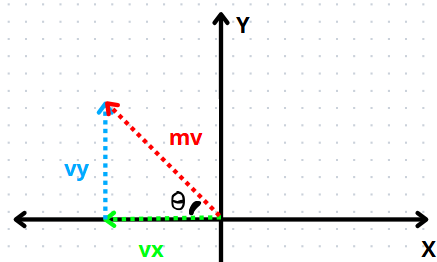
Es la segunda derivada de la posición con respecto al tiempo o primera derivada de la velocidad con respecto al tiempo

La velocidad es máxima cuando la posición es cero, y la aceleración alcanza su mayor valor cuando la posición es extrema. Es importante entender esta relación para una mejor interpretación de los datos.

Las anteriores componentes de la velocidad y aceleración, tienen otros elementos, como sus magnitudes, que se encargan de medir la intensidad del respectivo vector, representándolo con un escalar, pero únicamente sus valores positivos:  
 magnitud de la velocidad

magnitud de la aceleración

En este punto, se tiene la magnitud de los vectores velocidad y aceleración, conocer estos valores no nos permite extraer información sobre la orientación exacta del movimiento en el plano, es aquí cuando se hace necesario encontrar el ángulo que forma el vector resultante con el eje x



Con la tangente inversa podemos hallar el valor en radianes del ángulo , ya que conocemos los catetos que se forman en el triángulo rectángulo. Seguido de esto, un paso que será necesario para el grafico que se planea mostrar más adelante, es encontrar el vector unitario con la dirección a la que apunta cada componente, de la siguiente manera:

, dir

Para el caso de la aceleración, se puede realizar de la misma forma con sus respectivos valores.

## Diferenciación numérica y Convolución

En el inciso anterior se mencionó el hecho de que estamos trabajando en un medio discreto, como en este caso, pixeles de una secuencia de imágenes, fotograma a fotograma, por lo que a la hora de calcular las derivadas se usaron métodos numéricos, como diferencias centradas para la mayoría de pixeles y diferencias hacia adelante y diferencias hacia atrás para los bordes de cada frame

Diferencias centradas:

Donde seria cada cuanto tiempo (segundos) hay un nuevo frame, es decir , por ejemplo, si estamos en un medio de 30 FPS, entonces = 0. 03333.. segundos (Tomar en cuenta que estos pasos se pueden realizar para las componentes x e y de la velocidad y aceleración)

En un medio discreto y experimental, puede haber variables que causen interferencia a la señal o datos que estamos tratando, como puede ser un bajón de FPS, lo que causaría ruido en los datos y finalmente provocando picos fuera de lo común en las derivadas y aún más en la segunda derivada (aceleración), para tratar estos errores, se ha utilizado la convolución discreta lineal para suavizar las derivadas y llegar a una mejor estimación.

La convolución es una operación matemática que combina dos funciones o señales, que genera una tercera función que representa como la segunda “suaviza” a la primera, se define como:

sí = 1, 2,…, N-1 y en otro caso

## Centro de masa.

En el laboratorio se intenta llegar a una estimación de la velocidad y aceleración de un objeto que sigue un movimiento pendular, a partir de un video, y debido a esto, debemos calcular el centro de masa del objeto fotograma a fotograma, en este caso, una esfera y partiendo de allí, calcular aquellas componentes físicas.

Donde:

xc, yc: centro de masa en x, centro de masa en y respectivamente

H: altura (Height), “y” se mueve en el eje vertical (altura del frame)

W: ancho (Width), “x” se mueve en el eje horizontal (ancho del frame)

x, y: coordenadas del pixel actual

𝞀 (x, y): valor de intensidad del pixel en la coordenada (x, y)

De esta forma se tiene el centro de masa del objeto, y siguiendo las ecuaciones, calcular esto fotograma a fotograma e ir guardando estos valores con el objetivo de realizar los demás cálculos y posteriormente ser presentados en la ventana y gráfico.

## Procesamiento Digital de Imágenes

En este inciso se explican algunos conceptos que se aplicaron en la realización del laboratorio, relacionado con el procesamiento digital de imágenes.

### Erosión: Es una operación morfológica que “disminuye” o “reduce” los pixeles claros o blancos, en este caso, con una intensidad de 255, en una imagen binaria. Matemáticamente se define como: Dada una imagen binaria A y un elemento estructurante (kernel) B, la erosión está dada por:

Cada pixel de la imagen de salida se convierte en 1 (blanco) únicamente si todos los pixeles cubiertos por el elemento estructurante en la posición correspondiente de la imagen original también eran 1.

### Dilatación: Es la operación complementaria a la erosión, se encarga de “expandir” los blancos, en una imagen binaria:

Un pixel de salida se convierte en 1 si al menos uno de los pixeles cubiertos por el elemento estructurante en la posición dada de la imagen original era 1.

### Máscara: Es una imagen binaria que se utiliza para aislar objetos de interés de la misma, se construye asignando 1 (o 255) a los pixeles de las regiones que se desean mantener y 0 a los demás, luego se realiza sobre la imagen original una operación AND bit a bit, y de esta forma, se preservan los pixeles “enmascarados” con un valor 1 y los demás se descartan (se les asigna el valor cero), posterior a esto, es posible invertir la mascara o realizar distintas operaciones morfológicas si así se desea.

# METODOLOGÍA

### Escenario Experimental

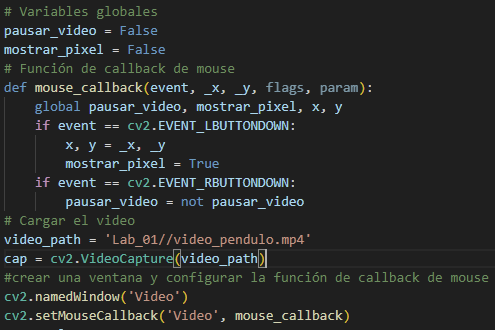
La primera fase del escenario experimental fue la grabación del video y preparación de un ambiente lo más ideal posible. Se montó un fondo homogéneo con un color que resalte y difiera del objeto a analizar, el péndulo es posicionado alejado del fondo para evitar que se toquen y así podemos despreciar la fricción. Con ayuda de un pequeño hilo, nos encargamos de soltar el péndulo con cierto ángulo de amplitud, así se pudo trabajar sin la problemática de que el video capte una parte de nuestras manos.



Anteriormente, se tomaron las medidas de la longitud de la cuerda (L), en nuestro caso L = 0.265 metros y el diámetro del objeto es D = 0.045 metros

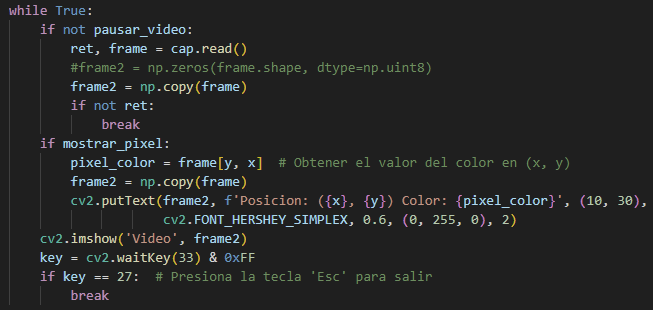
### Desarrollo del algoritmo

A partir de las herramientas que nos otorga la librería OpenCV, se realiza todo el procesamiento del video del péndulo, iniciando con la lectura del mismo. Además, para la facilidad de conversión de pixeles a metros, la detección de pixeles y su intensidad para seleccionar una máscara adecuada y diferentes testeos que se puedan realizar, se hizo uso de cv2.setMouseCallback(), esta función nos permitirá interactuar en la ventana que creemos con diferentes eventos del mouse que indiquemos. Inicializamos unas variables para mostrar el pixel y sus coordenadas en (x, y) y también para pausar el video:

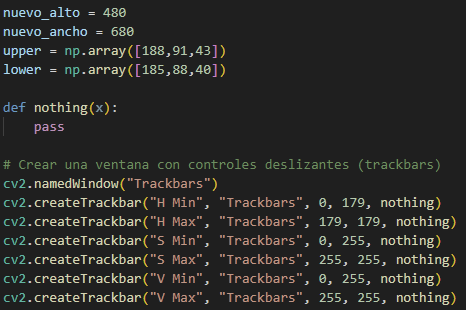


En caso de dar click derecho, el video se pausará o reanudará, si se presiona click izquierdo, se mostrará el pixel seleccionado, y guardaremos sus coordenadas en x, y.

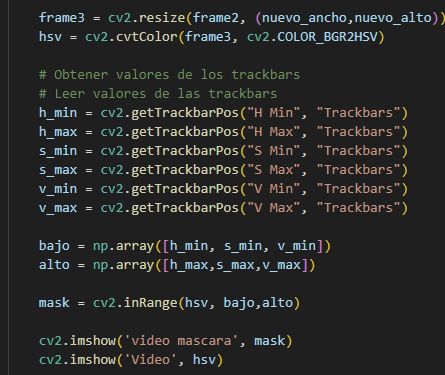
En este punto, resta mostrar el video en la ventana



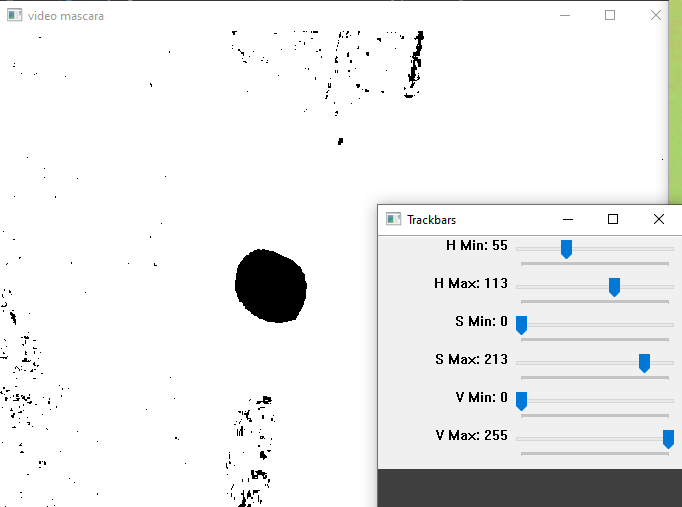
Con este código, el video se mostrará en la ventana, en caso de dar click izquierdo obtendremos el valor en BGR del pixel y lo mostraremos en la ventana, junto con sus coordenadas, si queremos salir del video se debe presionar la tecla “esc”. Para proceder con la obtención de los datos y así realizar los cálculos, se debe elegir una máscara, por lo que la opción mas adecuada en este caso, será usar “track bars” para hallar los valores más precisos posibles.



El anterior fragmento debemos ingresarlo fuera del ciclo while, mientras que el siguiente se escribe dentro.

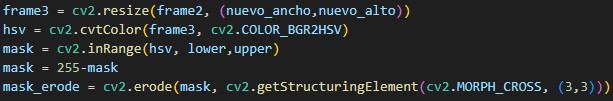


Lo que se realiza allí es crear unos trackBars para calibrar los límites de la máscara, por eso se inicializan primero como lower y upper, la función nothing(x) no hace nada, pero se crea ya que OpenCV tiene una sintaxis específica para los trackbars, además el frame debe ser convertido al formato hsv, con el objetivo de poder enmascarar de mejor forma el objeto sin importar las diferentes tonalidades de color que tenga, posteriormente los mostramos y reflejamos los cambios en tiempo real.



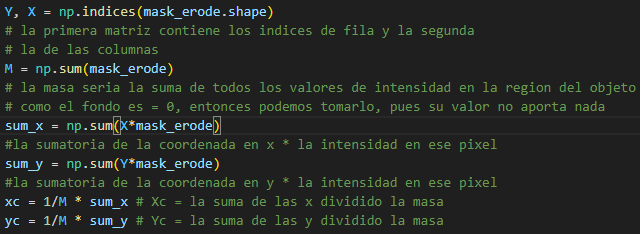
Una vez se seleccionan los limites mas precisos posibles, podemos eliminar los fragmentos de código correspondientes a los trackbars y guardar los valores quemados, en este caso lower = (80,64,0) y upper = (116,255,255) (formato BGR) quiere decir que todo color que supere upper y sea inferior a lower será negro (0) y todo pixel para el que se cumpla que lower ≤ pixel ≤ upper entonces será blanco (255).

Para proceder con todos los cálculos, la máscara debe ser invertida, para que el fondo obtenga valores nulos (0 o negro) y el objeto de interés sea blanco y así trabajar en base a este.



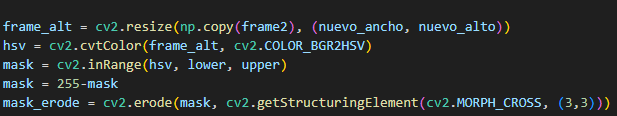
Si es necesario, realizar alguna operación morfológica.

Una vez obtenidos los elementos necesarios para calcular el centro de masa de la esfera, se deben recordar las ecuaciones anteriormente mencionadas y aplicarlas con ayuda de Numpy y sus funciones, así como se muestra en el siguiente fragmento, estos valores podemos luego mostrarlos en la ventana con cv2.puText() y dibujar un círculo justo en la posición calculada del centro de masa con cv2.circle() para verificar que los cálculos se estén realizando correctamente.





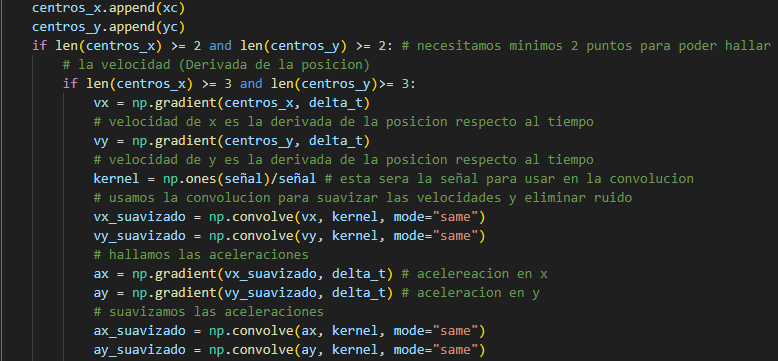
Note que, en este punto, si se llegase a presionar click derecho, se presentaría el problema de que el texto presentado en la ventana interfiere con el cálculo del centro de masa, por lo que se vuelve necesario realizar el cálculo sobre una variable que almacena los frames, distinta de otra, encargada de guardar los fotogramas que se presentan en ventana.





mask\_vis será el frame que mostraremos en ventana, y sobre mask\_erode es sobre quien haremos los cálculos del centro de masa y futuras operaciones.

Con el centro de masa calculado fotograma a fotograma, se prosigue con el cálculo de la velocidad y aceleración, tener en cuenta que, para esto, debemos tener como mínimo 3 puntos de la posición encontrados, es decir, debe haber por lo menos 3 coordenadas en x, y del centro de masa, estas las guardaremos en una lista de centros\_x, centros\_y, luego de verificar esta condición, usaremos funciones Numpy para hallar las derivadas y convoluciones con el objetivo de suavizarlas. Recuerde que = 0.0333… segundos



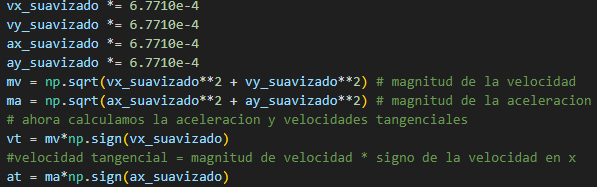
Señal = 3, este dato se elige con prueba y error.

Para este punto, tener en cuenta que la velocidad y aceleración están en pixeles/segundo (o segundo^2) por lo que se debe realizar la conversión de pixeles a metros con la medida que ya tenemos de la esfera (0.045m) y medidas tomadas con el evento click izquierdo, esto mostrará una coordenada en x, y, con la que se calcula posteriormente un promedio de las distancias euclidianas con el objetivo final de, con este promedio usar regla de 3 y así determinar cuantos metros corresponden a 1 pixel.

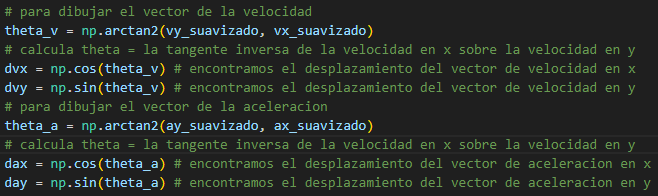
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X1** | **Y1** | **X2** | **Y2** | **Distancia** |
| 585 | 177 | 585 | 249 | 72 |
| 555 | 213 | 615 | 213 | 60 |
| 565 | 240 | 606 | 186 | 67.80117993 |
| 569 | 183 | 604 | 239 | 66.03786792 |

pixeles

Con el factor de conversión calculado, podemos convertir las velocidades y aceleraciones en metros/s y metros/s2. Hallar la magnitud de estas será de utilidad para encontrar la velocidad y aceleración “tangencial”, en este contexto, elegiremos como eje tangencial, el eje x, por lo que según se desplacé el objeto en este eje, se indicará el signo de su dirección (derecha = positivo, izquierda = negativo)

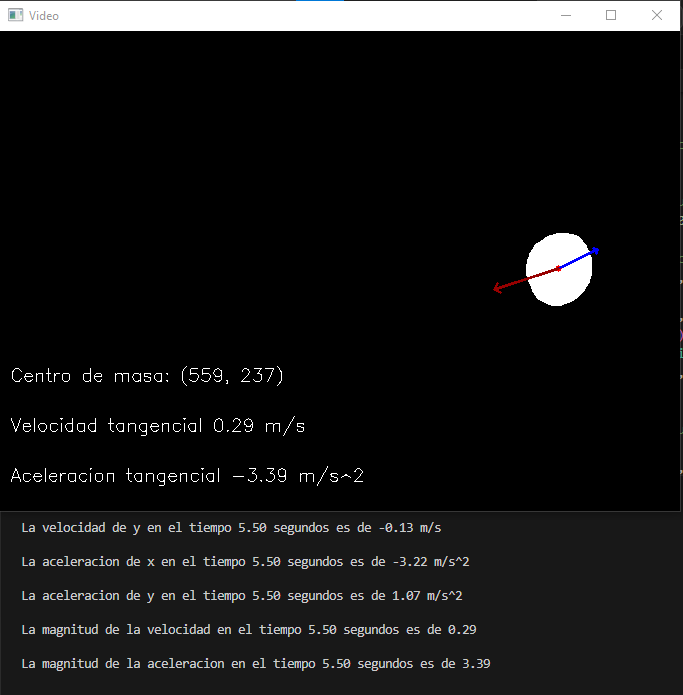


En la sobreposición de los datos en el video, se dibujaron los vectores como flechas usando cv2.arrowedLine(), esta función recibe como tercer parámetro un punto en el eje (x, y) que debemos conocer e indicar, para poder ser calculado se usó la tangente inversa para poder obtener el ángulo theta y posteriormente hallar el seno y coseno, finalmente se grafica sobre la ventana:



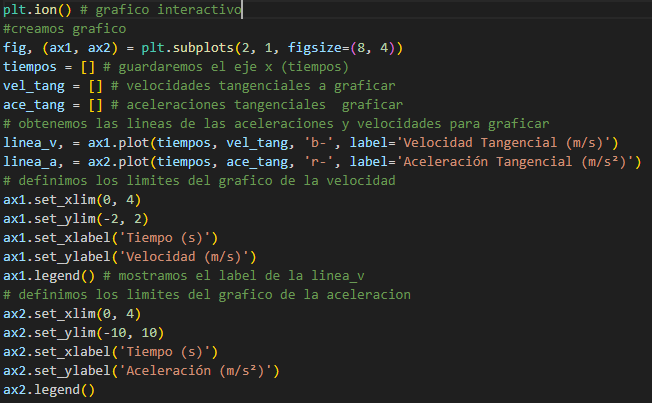


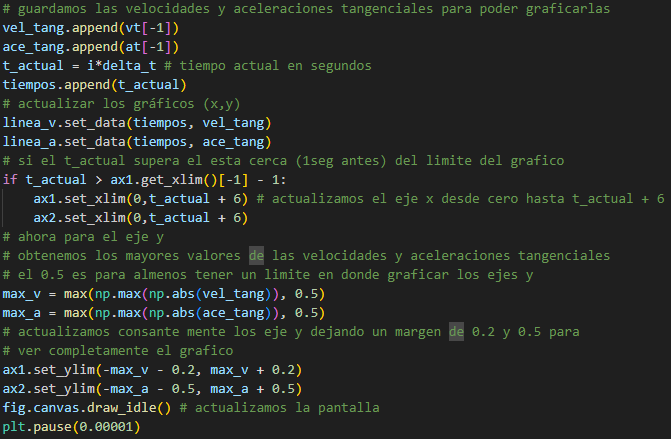
Usando cv2.putText() se agregaron datos relevantes para el laboratorio que se deseen mostrar en la ejecución. Hasta el momento se ha finalizado la fase de procesamiento de imágenes y toma de datos, resta graficar las funciones de velocidad y aceleración en función del tiempo, en este punto, el resultado es el siguiente:



En la terminal se muestra información adicional, se puede realizar por medio de print(), Para tener conocimiento de el segundo que se está transcurriendo en video, se debe inicializar un contador y crear la variable t\_actual = i\*delta\_t donde i debe ingresar al ciclo while con un valor de -1 y en el caso en que el video no este en pausa, su contador debe aumentar una unidad, así nos aseguramos de tomar el frame 0 y aumentar el conteo cada que el video no esté en pausa.

Para el gráfico de las funciones velocidad y aceleración en función del tiempo, se dio uso de la librería Matplotlib y su función de gráficos interactivos en tiempo real plt.ion(), se crearon listas para guardar el tiempo, las velocidades y las aceleraciones tangenciales, definimos limites dinámicos para los ejes x, y de cada componente física y finalmente proyectamos estos valores en pantalla.





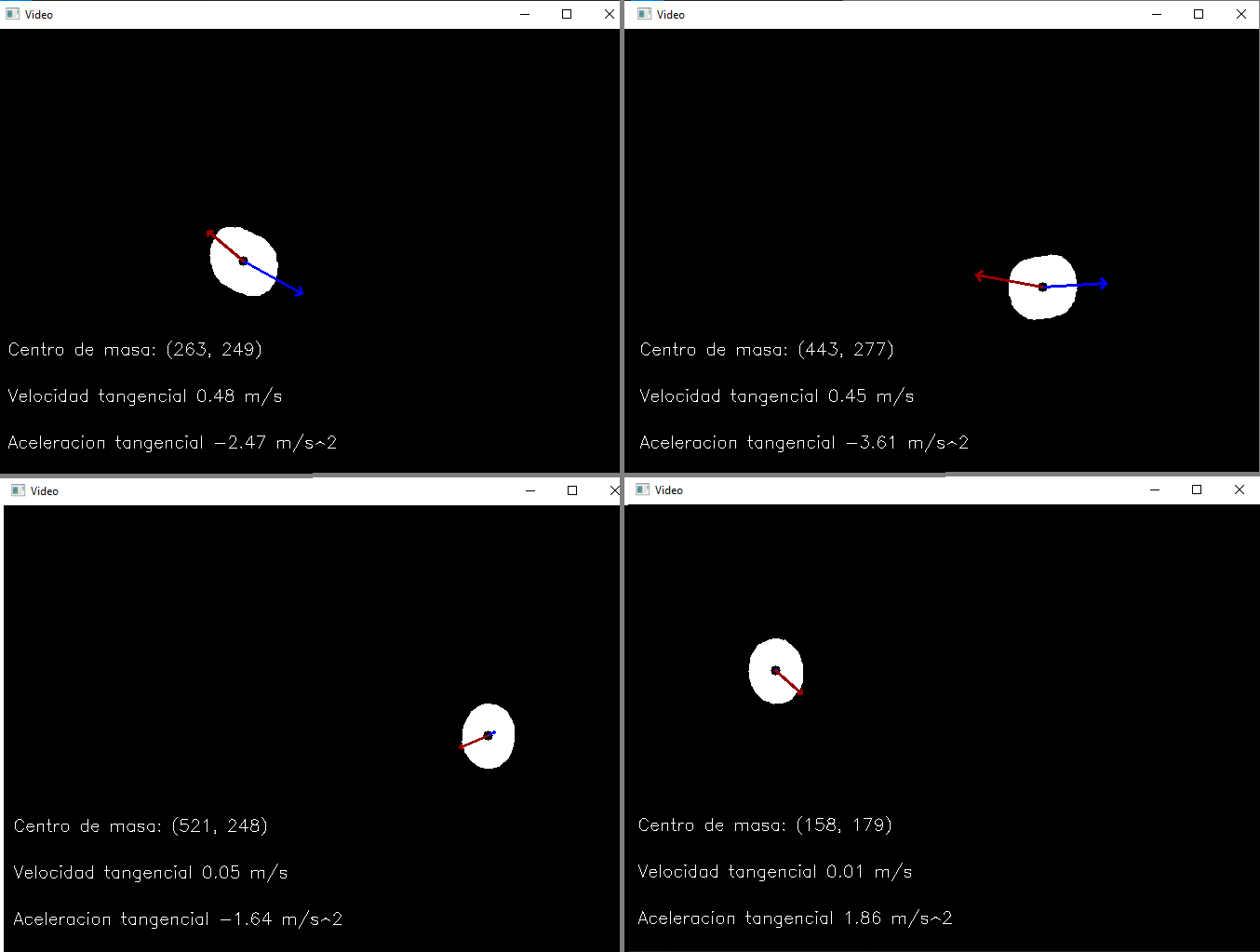
En caso de que el tiempo actual esté apunto de superar el eje x de los gráficos, este debe aumentar su rango, en la implementación se usó un valor de 6 y constantemente se debe aumentar el valor de los ejes y, dependiendo de los valores capturados frame a frame.

Para calcular el ángulo theta, basta con usar la formula:

donde A es el desplazamiento del péndulo de el punto de equilibrio y el extremo del mismo, L es la longitud en metros de la cuerda

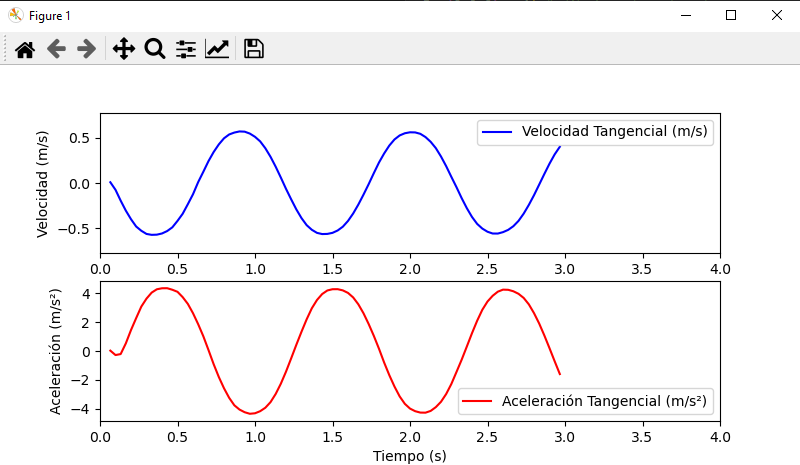
# RESULTADOS

Esta sección está destinada al análisis de los datos obtenidos, visualización de los mismos y algunas observaciones. En primer lugar, se observan los vectores, centro de masa e información presentada en la ventana de procesado de imágenes, donde se desea confirmar que lo indicado por los cálculos es coherente con lo observado. Principalmente nos fijaremos en si el centro de masa frame a frame es obtenido de forma correcta con algunos fotogramas de testeo



Observe que el centro de masa (punto negro en medio de la esfera blanca) en los frames tomados, coinciden con el centro aproximadamente del objeto, además es posible saber la ubicación en x, y por el texto que aparece en la ventana, podemos dar certeza de que este calculo se está realizando de forma correcta.

Con la misma imagen anteriormente presentada, podemos hacer el análisis sobre los vectores sobrepuestos. La aceleración corresponde a la flecha roja, mientras que la velocidad a la azul. Se logra observar que la magnitud de estas cambia durante el trayecto, parten del centro del objeto y los valores numéricos (velocidad tangencial y aceleración tangencial) coinciden con la dirección a la que apuntan, tomar en cuenta que en una imagen, el sistema de referencia tiene valores positivos de y: abajo, valores negativos de y: arriba, valores positivos de x: derecha, valores negativos de x: izquierda. Sabiendo lo anterior, la información que se visualiza en la ventana, es coherente.

A continuación, se visualiza el gráfico realizado por medio de Matplotlib para la velocidad y aceleración en función del tiempo, se espera hallar funciones senoidales o cosenoidales, donde se logre visualizar una oscilación de estas, esto nos indicaría que efectivamente, se cumple un Movimiento Armónico Simple (MAS).

Los gráficos de cada componente, siguen el comportamiento de funciones seno o coseno, por consecuencia, podemos ver como ambas oscilan y cumplen el requisito para ser un Movimiento Armónico Simple.

# COMPARACIÓN CON EL MODELO TEÓRICO

El proceso experimental parece cumplir con precisión la teoría del MAS, pero es necesario realizar el modelo teórico para comparar resultados, hallar fallos, estimar errores e interpretarlos correctamente. Los elementos que se tomarán en cuenta para este análisis son: Velocidad y aceleración en el punto más bajo y punto inicial del péndulo y ángulo de amplitud

*Velocidad y aceleración en los extremos*

Datos iniciales tomados en el modelo experimental:

Longitud de la cuerda L: 0.265 metros

Periodo T = 1.18 segundos

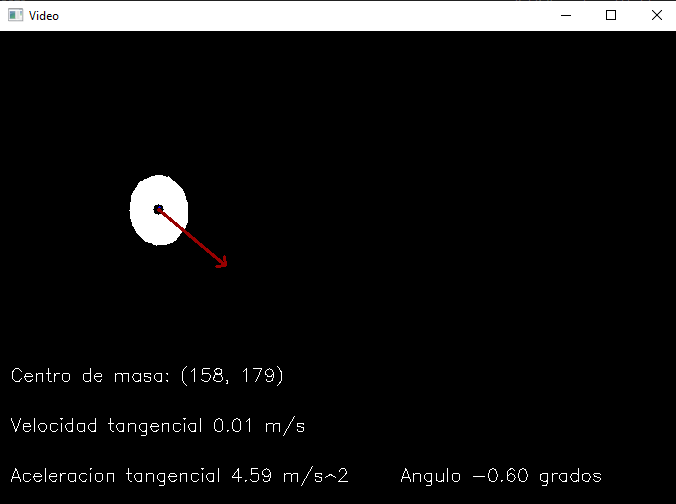
*Extremo y punto inicial en el modelo teórico:*

Tomando cuantos pixeles hay entre el xmax y el punto de equilibrio en el video, hallamos A, que seria la longitud de amplitud del péndulo. Posteriormente se encuentra , ya que . Recordar que = 5.3247 rad/s. Reemplazando en: = 0.1374 mts

Tomamos A = ya que es el extremo opuesto al punto inicial, por eso el valor de la aceleración es positivo.

*Extremo y punto inicial en el modelo experimental:*

Verificando los valores que muestra la ventana al ejecutar el código, cuando el péndulo está en un extremo, muestra:



*Punto más bajo:*

*Punto más bajo en el modelo teórico:*

En el punto más bajo el sistema está en equilibrio, estamos en el punto cero en x, es decir: entonces

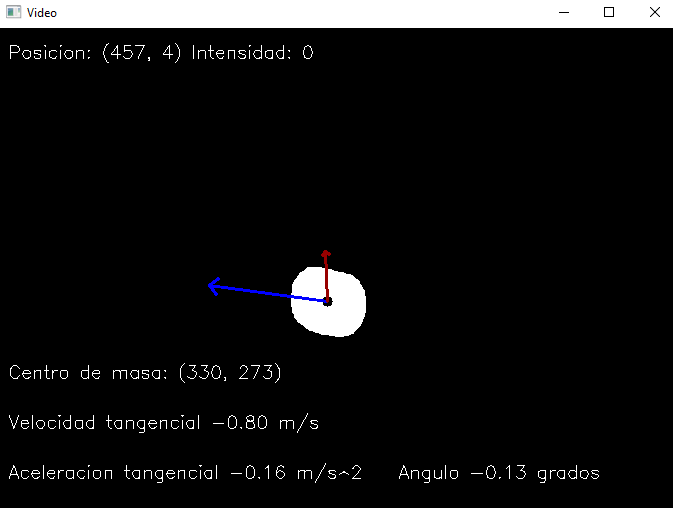
, como

Entonces tenemos que velocidad en el punto más bajo es:

Y la aceleración en el punto más bajo es:

*Punto más bajo en el modelo experimental:*

Recordar que el punto equilibrio anteriormente fue definido como 344px en el eje x, por lo que hay que posicionarse en un pixel cercano en ese y allí observar los datos.



El punto 330 en el eje x, es cercano y podemos realizar el análisis desde esta coordenada

*Angulo de amplitud en el modelo teórico:*

La formula a seguir es: , dando como resultado: , incluso si quisiéramos modificar el parámetro A, por uno más preciso podríamos obtener valores mas precisos, todo esto debido a que en el video no se logra capturar correctamente el inicio del péndulo o punto pivote

*Angulo de amplitud en el modelo experimental:*



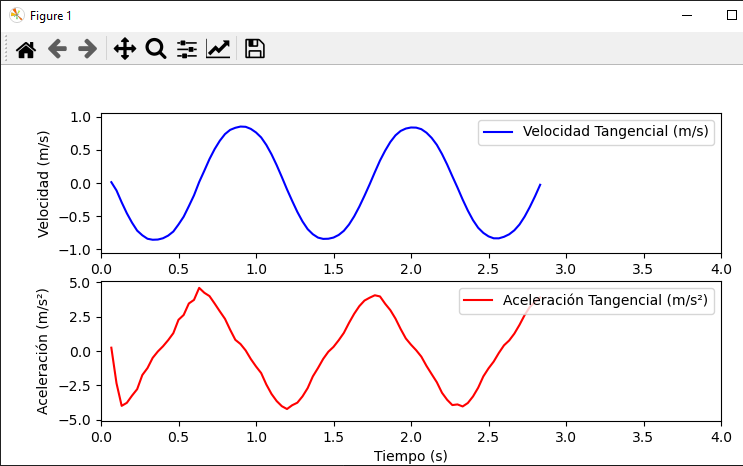
En este componente el error es un poco mas pronunciado, probablemente sea por el parámetro A y algún error al ser calculado.

*Comparación de los datos observados y teóricos:*

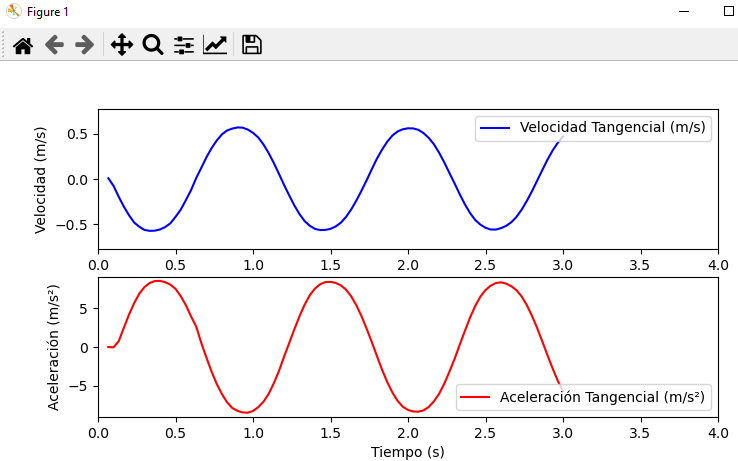
|  |  |  |  |
| --- | --- | --- | --- |
| **Componente** | **Experimental** | **Teórico** | **Error** |
| Velocidad en un extremo | 0,01m/s | 0m/s | 0.01 m/s |
| Aceleración en un extremo | 4.6072 m/s^2 | 4.59 m/s^2 | 0.0172m/s^2 |
| Velocidad en el punto bajo | -0.80 m/s | -0.8652 m/s | 0.0652 m/s |
| Aceleración en el punto bajo | -0.16 m/s^2 | 0 m/s^2 | 0.16 m/s^2 |
| Angulo de amplitud | 0.55 grados | 0.6601 grados | 0.1101 grados |

Además de esto si visualizamos la graficas, podemos afirmar que el movimiento cumple con un MAS:

Señal = 2 (una convolución menos agresiva)



Señal = 3 (Una convolución un poco más agresiva)



El problema de esta última es que las derivadas de los bordes, es decir, el comienzo del video es modificado por la convolución haciendo que las funciones no cumplan con la relación aceleración-velocidad. A parte de esto, el movimiento cumple con los requisitos.

# CONCLUSIONES

El laboratorio presente, demuestra que la combinación de técnicas de procesamiento digital de imágenes permite extraer con un considerable alto nivel de precisión y fidelidad las características dinámicas de un péndulo simple a partir de un video en condiciones controladas. En primer lugar, el cálculo del centro de masa, sus derivadas y conceptos correctamente aplicados mostró un comportamiento de la velocidad y aceleración coherente con el modelo teórico del movimiento armónico simple, La segmentación por medio de HSV y el suavizado por convolución resultaron efectivos para reducir el ruido de los datos al llevar a cabo la derivación numérica. No obstante, se identificaron errores en algunos elementos considerablemente altos, mostrando como el llevar a cabo un ambiente en condiciones controladas no asegura un perfecto modelo y que a pesar de esto se puede considerar que el sistema sigue las normas, leyes o requisitos, como en este caso, para decir que es un MAS.

# REFERENCIAS

[1] A. Sachdev, “Dilation (Morphological Operation)—Image Processing,” Medium, 5 min read, Mar. 17, 2024. https://medium.com/@anshul16/dilation-morphological-operation-image-processing-82d16a619f59.

[2] A. Sachdev, “Erosion (Morphological Operation)—Image Processing,” Medium, 6 min read, Mar. 17, 2024. https://medium.com/@anshul16/erosion-morphological-operation-image-processing-18537f7c66cd.

[3] “Centro de masas,” Wikipedia, la enciclopedia libre, Jan. 3, 2025 (última edición). https://es.wikipedia.org/wiki/Centro\_de\_masas.

[4] P. Negrón, “Diferenciación numérica,” Universidad de Puerto Rico en Humacao: https://mate.uprh.edu/~pnegron/notas4061/numdif/numdif.htm

[5] “OpenCV: Open Source Computer Vision Library,” OpenCV.org. [En línea]. Disponible: https://opencv.org/.

[6] “NumPy — The fundamental package for scientific computing with Python,” NumPy.org. [En línea]. Disponible: https://numpy.org/

[7] “Convolución de Tiempo Discreta,” LibreTexts Señales y Sistemas

https://espanol.libretexts.org/Ingenieria/Senales\_y\_Sistemas\_(Baraniuk\_et\_al.)/04%3A\_Análisis\_de\_dominio\_de\_tiempo\_de\_sistemas\_discretos\_de\_tiempo/4.03%3A\_Convolución\_de\_Tiempo\_Discreta.

[8] “Movimiento armónico simple,” Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Movimiento\_armónico\_simple.

1. [↑](#footnote-ref-1)