

SINCRONIZACION CAMARAS

Investigacion UHPC

Lukas Wolff Casanova

1 de mayo de 2024

1. INTRODUCCION

Actualmente se está utilizando el método PIV en la simulación del carbopol, el cual, en pocas palabras, utiliza un plano bidimensional para rastrear partículas. El problema radica en que buscamos hacer la simulación en torno a fibras, las cuales pueden rotar, por lo cual es necesario implementar una simulación tridimensional para poder utilizar la tecnología PTV.

En base a lo anterior, se plantea lograr una sincronización de las cámaras para así poder implementar el software.

2. METODOLOGIA

Para poder realizar una sincronización de las cámaras es necesario implementar un trigger, el cual puede ser mediante software (SW) o hardware (HW). Previo a decidir cuál era el trigger correcto a implementar, se comenzó a armar el setup para poder sincronizar las cámaras.

El setup consistió básicamente de una hélice sobre un motor brushless, donde, mediante un sensor infrarrojo, se podía medir sus revoluciones en tiempo real. Además, se diseñó un soporte para las cámaras, de modo que estas estuvieran alineadas en todo momento.

Los diseños 3D realizados son los siguientes:

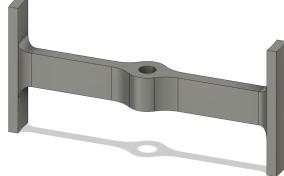


Figura 1: Helice

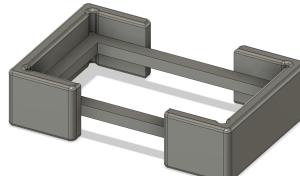


Figura 2: Soporte camaras



Figura 3: Soporte motor y sensor

A continuación se mostrarán las iteraciones para llegar al setup final.

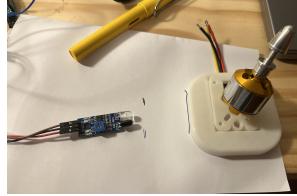


Figura 4: Prueba sensor



Figura 5: Helice



Figura 6: Prueba soporte
camaras

Se determinó que el sensor infrarrojo no lograba medir correctamente las revoluciones, ya que este utilizaba el reflejo de la luz y no alcanzaba a recibir la señal de vuelta. El profesor Jorge sugirió utilizar un medidor de revoluciones láser.

Al mismo tiempo que se realizaba el setup, se contactó a Basler, quienes nos redirigieron con un contacto en Chile. Hablando con Jorge (el contacto), se concluyó que la mejor forma de realizar el trigger es mediante software, donde además nos dio un hub de Python como guía.

El código Python se optimizó de modo que durante el ciclo de fotos, estas solo se guardan en una variable sin procesar, y posteriormente, se almacenan como imágenes visibles y no como un array.

Finalmente, el setup es el siguiente:



Figura 7: Setup final 1

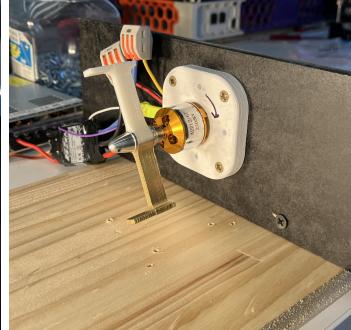


Figura 8: Setup final 2

Por lo tanto, el proceso para realizar una medición es el siguiente:

- Encender motor brushless
- Medir RPM
- Disparar cámaras

Es importante notar que el motor gira en sentido horario. Además, las mediciones fueron realizadas a 200 FPS.

3. RESULTADOS

A continuación se mostrarán las fotos de una medición a 5645 RPM.

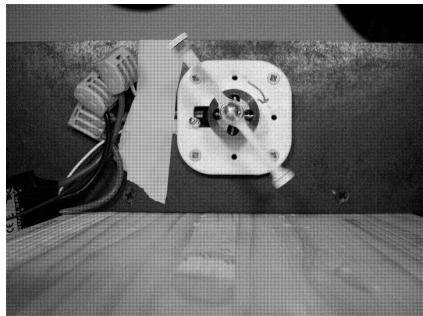


Figura 9: Imagen 1 camara 0

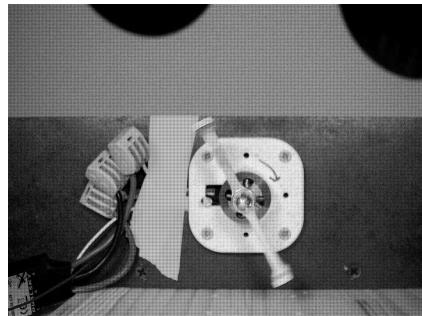


Figura 10: Imagen 1 camara 1

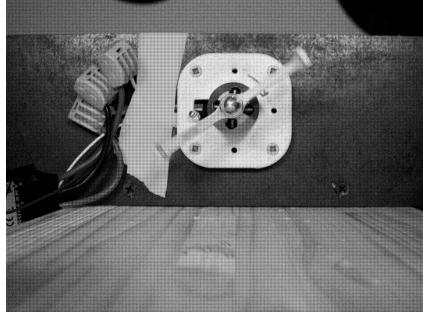


Figura 11: Imagen 2 camara 0

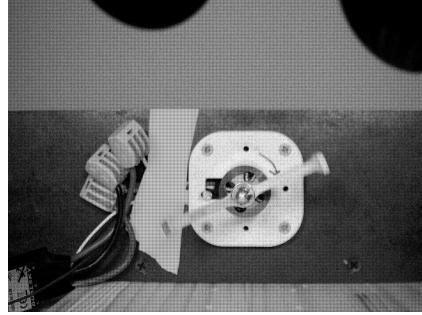


Figura 12: Imagen 2 camara 1

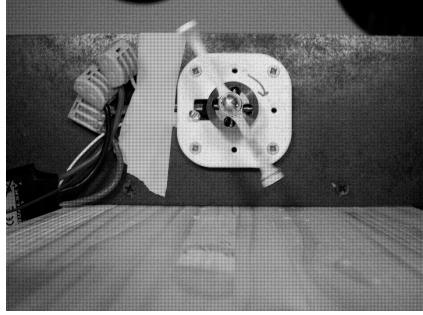


Figura 13: Imagen 3 camara 0



Figura 14: Imagen 3 camara 1

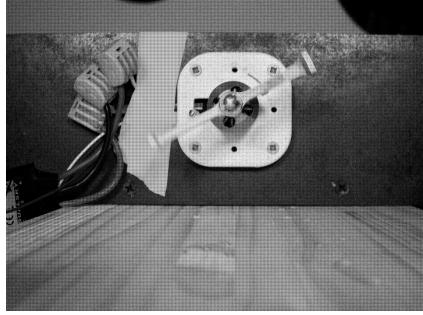


Figura 15: Imagen 4 camara 0

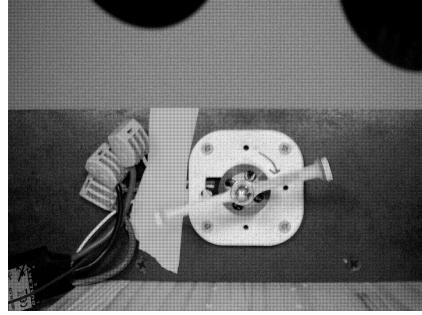


Figura 16: Imagen 4 camara 1

A continuación se mostrarán las fotos de una medición a 8519 RPM.

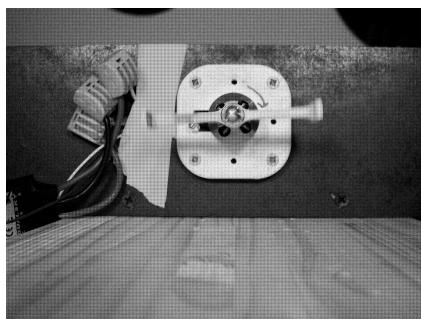


Figura 17: Imagen 1 camara 0



Figura 18: Imagen 1 camara 1

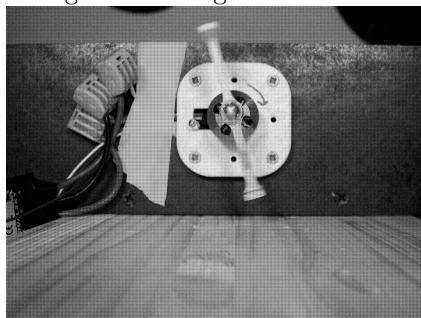


Figura 19: Imagen 2 camara 0

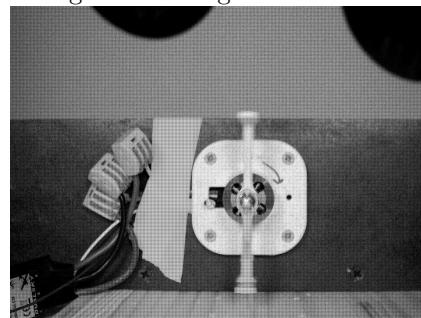


Figura 20: Imagen 2 camara 1

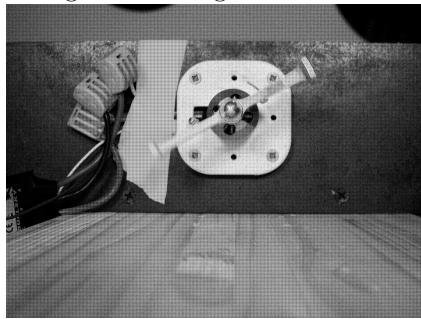


Figura 21: Imagen 3 camara 0

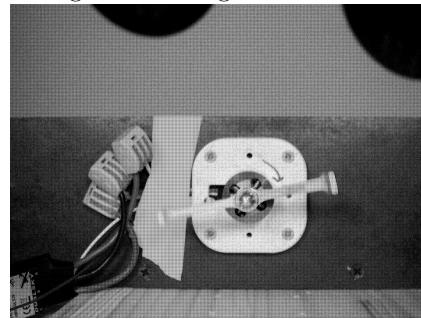


Figura 22: Imagen 3 camara 1

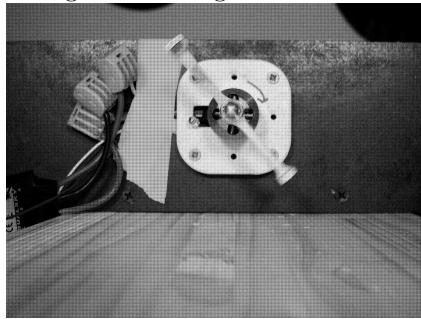


Figura 23: Imagen 4 camara 0

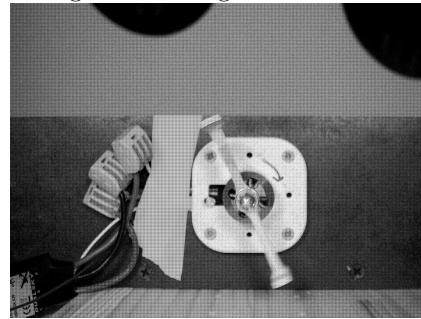


Figura 24: Imagen 4 camara 1

A continuación se mostrarán las fotos de una medición a 12055 RPM.



Figura 25: Imagen 1 camara 0

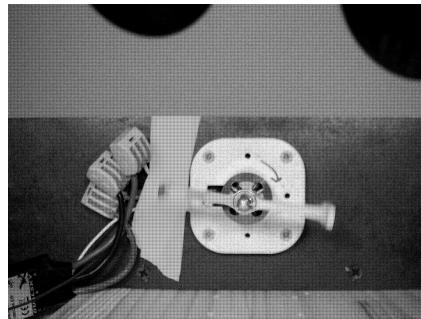


Figura 26: Imagen 1 camara 1

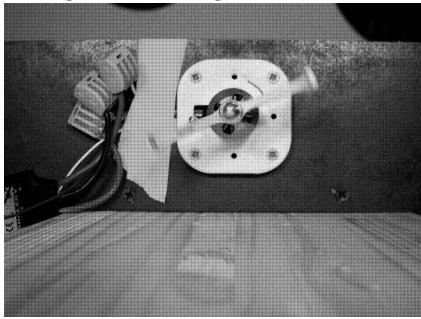


Figura 27: Imagen 2 camara 0

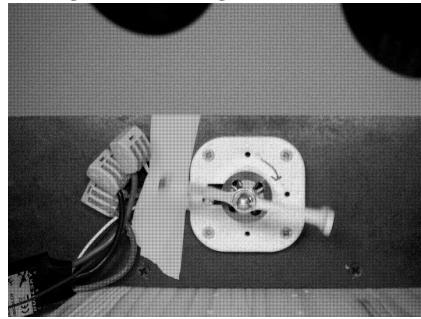


Figura 28: Imagen 2 camara 1

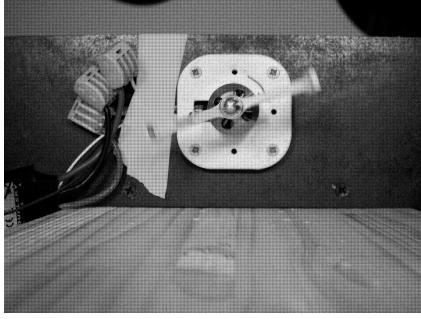


Figura 29: Imagen 3 camara 0

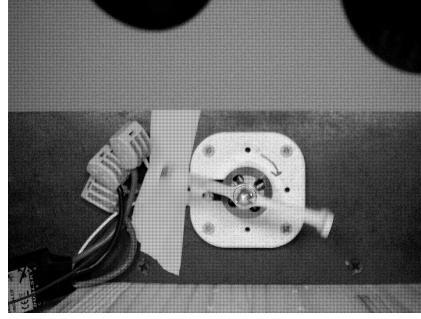


Figura 30: Imagen 3 camara 1

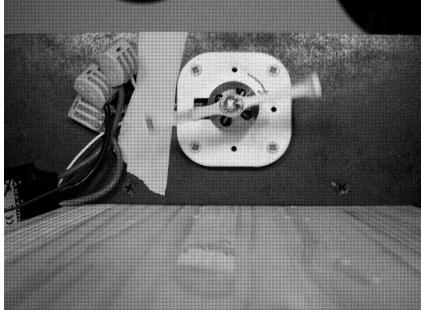


Figura 31: Imagen 4 camara 0

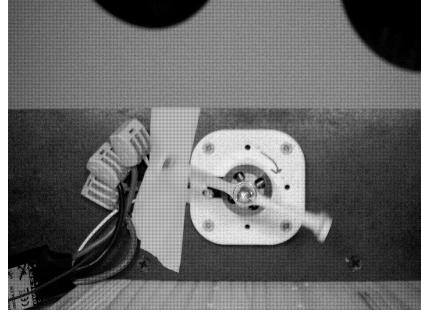


Figura 32: Imagen 4 camara 1

De esta última toma, es interesante notar que todas las fotos son prácticamente iguales en posición. Esto se debe a que la hélice giraba a aproximadamente la misma frecuencia que los FPS.

Además, se realizó una medición a 1 FPS para verificar los resultados.



Figura 33: Imagen 1 camara 0

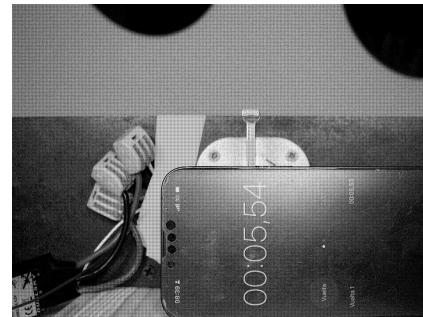


Figura 34: Imagen 1 camara 1



Figura 35: Imagen 2 camara 0

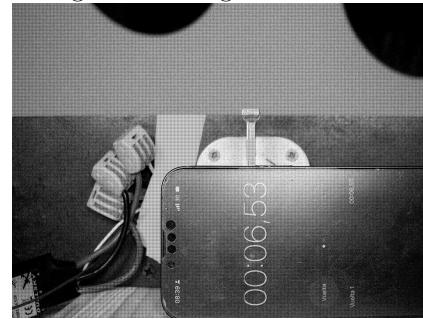


Figura 36: Imagen 2 camara 1



Figura 37: Imagen 3 camara 0

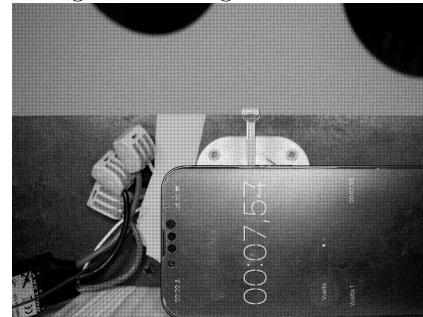


Figura 38: Imagen 3 camara 1



Figura 39: Imagen 4 camara 0

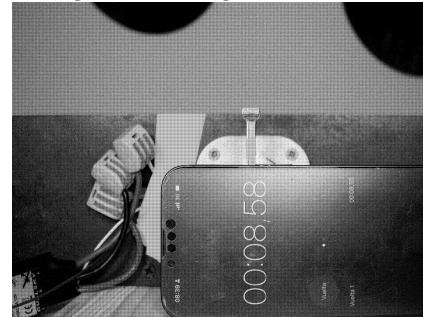


Figura 40: Imagen 4 camara 1

4. ANALISIS

En primer lugar, surgió la incógnita de si el desfase era una pequeña porción de vuelta o una vuelta completa más fracción. Sin embargo, como se están sacando más fotos que RPS, significa que solo hay un desfase muy pequeño.

En base a lo anterior, y aproximando en grados, se obtiene lo siguiente:

RPM	DESFASE [s]
5645	$1,6 * 10^{-5}$
8519	$3,4 * 10^{-5}$
12055	$3,9 * 10^{-5}$

Una vez determinado que se logró la sincronización, se intentó sacar una gran cantidad de fotos, pero el programa tiene como límite 20, donde aún se desconoce la razón. Puede ser por la RAM, el software en sí mismo, etc. Aun así, el objetivo principal fue resuelto.

5. CONCLUSION

En conclusión, se logró sincronizar las cámaras con muy poco desfase, el cual siempre va a existir ya que el trigger es mediante SW, donde se corre un ciclo for. Si se hace mediante HW se podría erradicar esta diferencia, pero puede llegar a ser más costoso y complejo. Además, tomando a favor la velocidad del carbopol, se puede determinar que esta diferencia es despreciable en las mediciones.

Finalmente, se determinó que el próximo desafío es cómo tomar y procesar un gran número de fotos.

6. ESTRUCTURA CODIGO

A continuación se mostrarán las partes clave del código. Primero se configuran las cámaras.

```
for i, cam in enumerate(cameras):
    cam.Attach(tlFactory.CreateDevice(devices[i]))

    cam.Open()
    cam.AcquisitionFrameRateEnable.SetValue(True)

    #Seteo FPS
    cam.AcquisitionFrameRate.SetValue(200)
    #Seteo exposicion
    cam.ExposureTime.SetValue(100)
    cam.Close()
```

Posteriormente, se capturan las fotos, las cuales son guardadas inmediatamente en una lista en forma de array. De esta forma, se pierde el menor tiempo posible procesándolas.

```
for i in range(countOfImagesToGrab):
    if not cameras.IsGrabbing():
        break

    R = cameras.RetrieveResult(5000, pylon.TimeoutHandling_ThrowException)
    imagenes.append(R)
```

Finalmente, se procesan los arrays para obtener la imagen, el tiempo y la cámara que la tomó. De esta forma, es posible guardar las fotos de forma ordenada en sus respectivas carpetas.

```
for i, elementos in enumerate(imagenes):
    img = elementos.GetArray()
    img = cv2.equalizeHist(img)
    camara = elementos.GetCameraContext()
    tiempo = elementos.GetTimeStamp()

    folder_path = f'FOTOS/{RPM}/{camara}'
    print(folder_path)

    if not os.path.exists(folder_path):
        os.makedirs(folder_path)

    file_name = f'{tiempo}.png'
    file_path = os.path.join(folder_path, file_name)

    image = Image.fromarray(img)
    image.save(file_path)

    print(f"Imagen guardada: {file_path}")
```

Enlace a GitHub con toda la información: GIT-HUB-INVESTIGACION-UHPC.

Enlace a GitHub con código base: : GIT-HUB-CODIGO-BASE.