

MULTIPLES DISPAROS SINCRONIZADOS

Investigacion UHPC

Lukas Wolff Casanova

2 de mayo de 2024

1. INTRODUCCION

Posterior a sincronizar las cámaras, se determinó que la siguiente tarea era lograr sacar 5000 por cámara, lo que da un rango de operación de 25 segundos a 200 fps. Para poder lograrlo se postularon varias ideas como listas concatenadas, determinar el tamaño de la lista o usar el buffer de las cámaras.

2. METODOLOGÍA

Para comenzar se investigó sobre las listas concatenadas o listas con punteros. La diferencia que tienen con una lista normal es que en vez de estar almacenadas de manera conjunta en la memoria RAM, estos datos se almacenan de forma particular, junto a un puntero que indica dónde está el siguiente índice, de esta manera se logra que el proceso de almacenar datos sea mucho más eficiente en términos de velocidad y memoria, pero tiene como contra que recorrer una lista es más lento. Tomando a favor la estructura del código, donde se recorre la lista una vez que las fotos ya fueron tomadas, se determinó que era una buena herramienta a utilizar.

Además, se activó el buffer de la cámara, el cual de igual manera envía una foto por ciclo al computador, pero ayuda a que la cámara no se sature.

En primer lugar se crea la clase *node*, la cual permite crear y recorrer una lista concatenada. La celda más importante del código es la siguiente:

```
# Método para recorrer la lista de nodos
def traverse(self):
    current_node = self.head

    while current_node:

        img = current_node.data.GetArray()
        img = cv2.equalizeHist(img)
        camara = current_node.data.GetCameraContext()
        tiempo = current_node.data.GetTimeStamp()

        # Carpeta donde se almacenarán las imágenes
        folder_path = f'FOTOS/{RPM}/{camara}'
        print(folder_path)

        # Crear la carpeta si no existe
        if not os.path.exists(folder_path):
            os.makedirs(folder_path)
```

```

# Nombre del archivo de la imagen
file_name = f'{tiempo}.png'

# Ruta completa del archivo de la imagen
file_path = os.path.join(folder_path, file_name)

# Crear una imagen PIL a partir de la matriz de la imagen
image = Image.fromarray(img)

# Guardar la imagen como PNG
image.save(file_path)

print(f"Imagen guardada: {file_path}")
current_node = current_node.next

```

Como se puede ver, todo el proceso que se realizaba anteriormente al recorrer la lista de imágenes, ahora se realiza dentro de la clase *node*, la cual es llamada al final del código:

Posteriormente, se configuran los ajustes tanto de cámaras como de fotos:

```

-----#
# DETERMINO EL NOMBRE DE CARPETA
RPM = 12055

# DETERMINO CANTIDAD DE IMAGENES A SACAR
countOfImagesToGrab = 5000
-----#

```

Donde el directorio final de las fotos será el siguiente:

FOTOS/RPM/numero-camara/tiempo-foto.jpg

Posteriormente, se configuran las cámaras:

```

# Create and attach all Pylon Devices.
for i, cam in enumerate(cameras):

    cam.Attach(tlFactory.CreateDevice(devices[i]))

    # Open the camera to set parameters
    cam.Open()

    # Buffer Size Adjustment
    cam.MaxNumBuffer = buffer # Adjust buffer size as needed

    # Configure camera settings
    cam.AcquisitionFrameRateEnable.SetValue(True)
    cam.AcquisitionFrameRate.SetValue(200)
    cam.ExposureTime.SetValue(100)

    # Close the camera after setting parameters
    cam.Close()

```

Comienza el *trigger*:

```
#-----
# Disparo las camaras
cameras.StartGrabbing()
#-----
```

Y se almacenan las fotos:

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

    #recibo la foto y la guardo como array y en la lista concatenada
    R = cameras.RetrieveResult(40000, pylon.TimeoutHandling_ThrowException)
    s.add_at_front(R)
```

Finalmente, se recorre la lista:

```
s.traverse()
print("Fotos Guardadas")
```

3. RESULTADOS

Se hizo una toma de 10,000 fotos en total a 200 fps, lo que equivale a 5,000 fotos por cámara. A continuación se presentan los resultados cada 1,000 fotos para demostrar que se mantiene la sincronización.

3.1. MOMENTO INICIAL

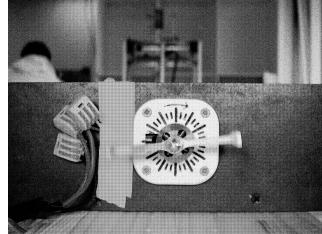


Figura 1: Imagen 0 camara 0

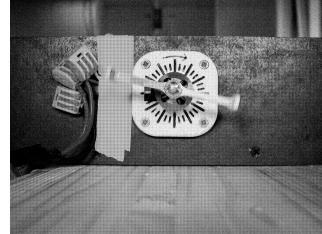


Figura 2: Imagen 0 camara 1

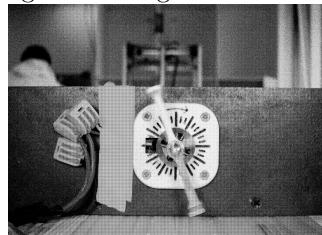


Figura 3: Imagen 1 camara 0

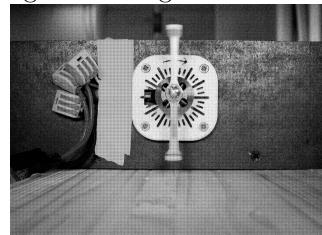


Figura 4: Imagen 1 camara 1

3.2. FOTOS 1000

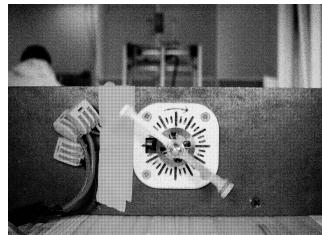


Figura 5: Imagen 500 camara 0

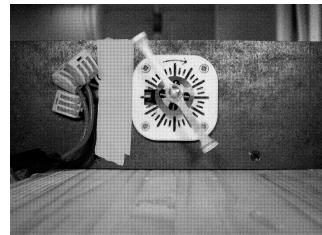


Figura 6: Imagen 499 camara 1

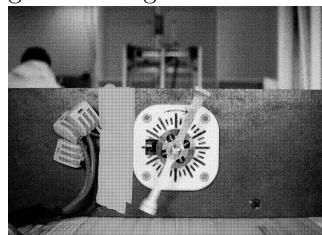


Figura 7: Imagen 501 camara 0

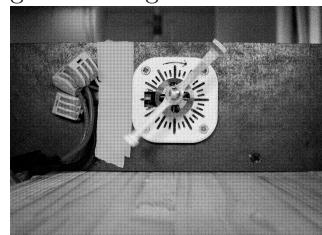


Figura 8: Imagen 500 camara 1

En este caso se puede observar que el índice de la cámara 1 disminuye en 1, lo cual indica que la cámara 1 tiene una foto extra en el intervalo [1000-2000].

3.3. FOTOS 2000

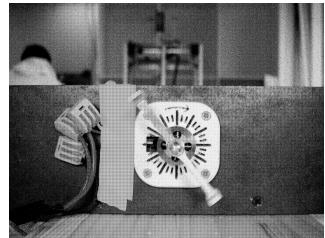


Figura 9: Imagen 1000 camara 0

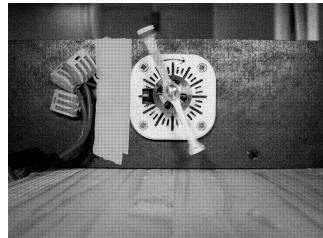


Figura 10: Imagen 999 camara 1

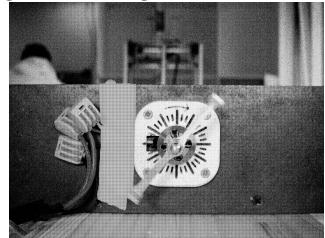


Figura 11: Imagen 1001 camara 0

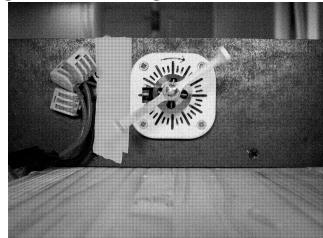


Figura 12: Imagen 1000 camara 1

3.4. FOTOS 3000

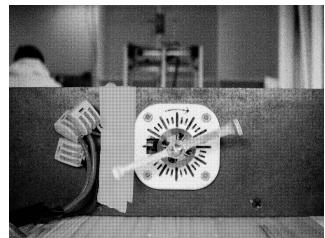


Figura 13: Imagen 1500 camara 0

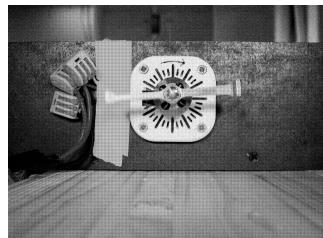


Figura 14: Imagen 1499 camara 1

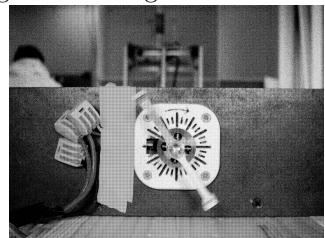


Figura 15: Imagen 1501 camara 0

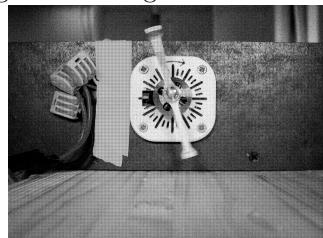


Figura 16: Imagen 1500 camara 1

3.5. FOTOS 4000

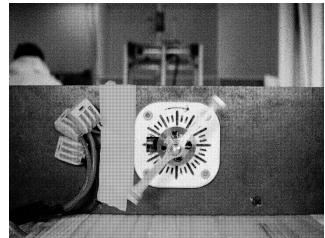


Figura 17: Imagen 2000 camara 0

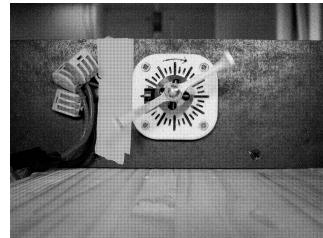


Figura 18: Imagen 1999 camara 1

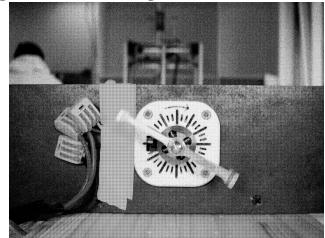


Figura 19: Imagen 2001 camara 0

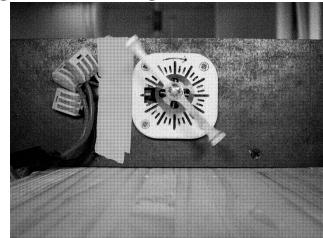


Figura 20: Imagen 2000 camara 1

3.6. FOTOS 5000

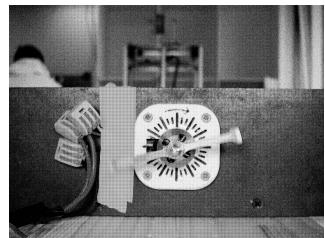


Figura 21: Imagen 2500 camara 0

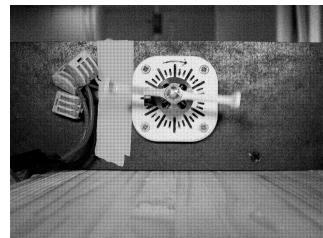


Figura 22: Imagen 2499 camara 1

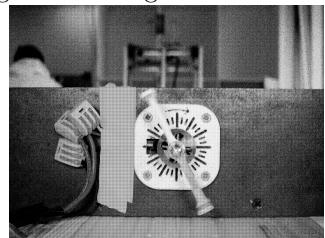


Figura 23: Imagen 2501 camara 0

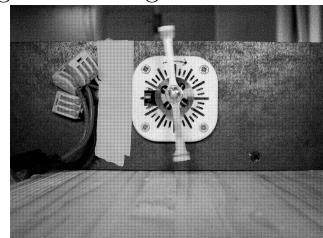


Figura 24: Imagen 2500 camara 1

3.7. FOTOS 6000

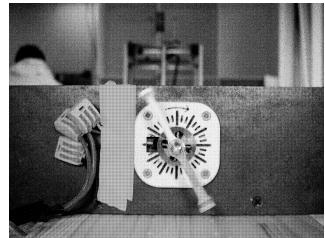


Figura 25: Imagen 3000 camara 0

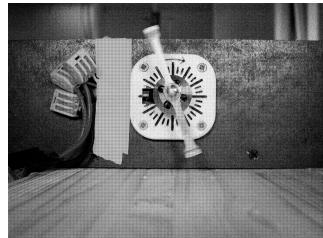


Figura 26: Imagen 2999 camara 1

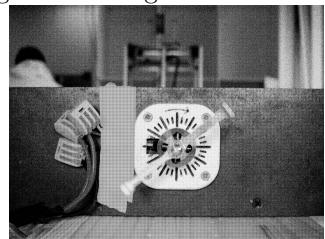


Figura 27: Imagen 3001 camara 0

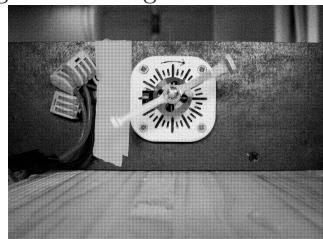


Figura 28: Imagen 3000 camara 1

3.8. FOTOS 7000

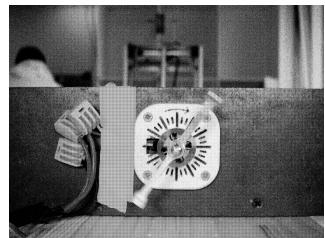


Figura 29: Imagen 3500 camara 0

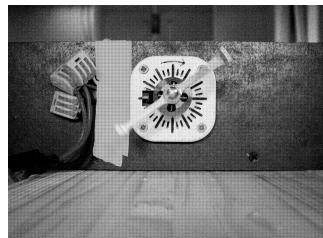


Figura 30: Imagen 3499 camara 1

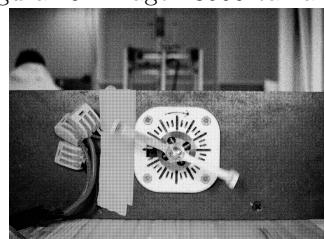


Figura 31: Imagen 3501 camara 0

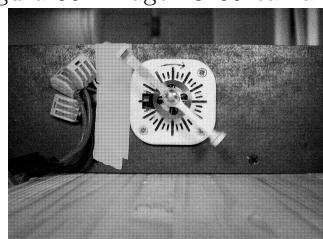


Figura 32: Imagen 3500 camara 1

3.9. FOTOS 8000

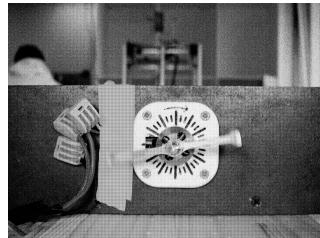


Figura 33: Imagen 4000 camara 0



Figura 34: Imagen 3999 camara 1

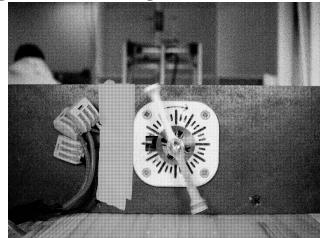


Figura 35: Imagen 4001 camara 0

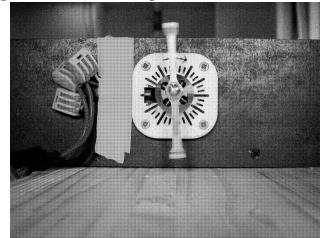


Figura 36: Imagen 4000 camara 1

3.10. FOTOS 9000

Es curioso este caso, ya que al hacer un código que busque las fotos 9000, si encuentra su presencia en las carpetas, pero luego al agregarlas en el informe.tex, este no logra encontrarlas, por lo que quizás no se logró su interpretación de array a imagen de manera correcta.

3.11. ANALISIS

Se puede observar claramente que la sincronización se mantiene a lo largo de las tomas, ahora bien, hay que tener precaución con el análisis de imágenes y la pérdida de datos, por lo que será necesario modificar o crear un código que analice el desfase entre fotos, y sea capaz de determinar si hay fotos extra o vacíos.

4. CONCLUSION

En conclusión, se logró el objetivo principal de sacar 10,000 fotos, ahora bien, surgieron ciertos detalles que hay que arreglar. De todas formas, ya existe una buena base donde se puede empezar a trabajar para la implementación del PTV a medida que se arreglan los detalles finales.

Finalmente, la siguiente gran tarea es comenzar a aprender e implementar el uso de PTV, además de arreglar los detalles antes mencionados.

Enlace a GitHub con toda la información: [GIT-HUB-INVESTIGACION-UHPC](#).

Enlace a GitHub con código base: : [GIT-HUB-CODIGO-BASE](#).