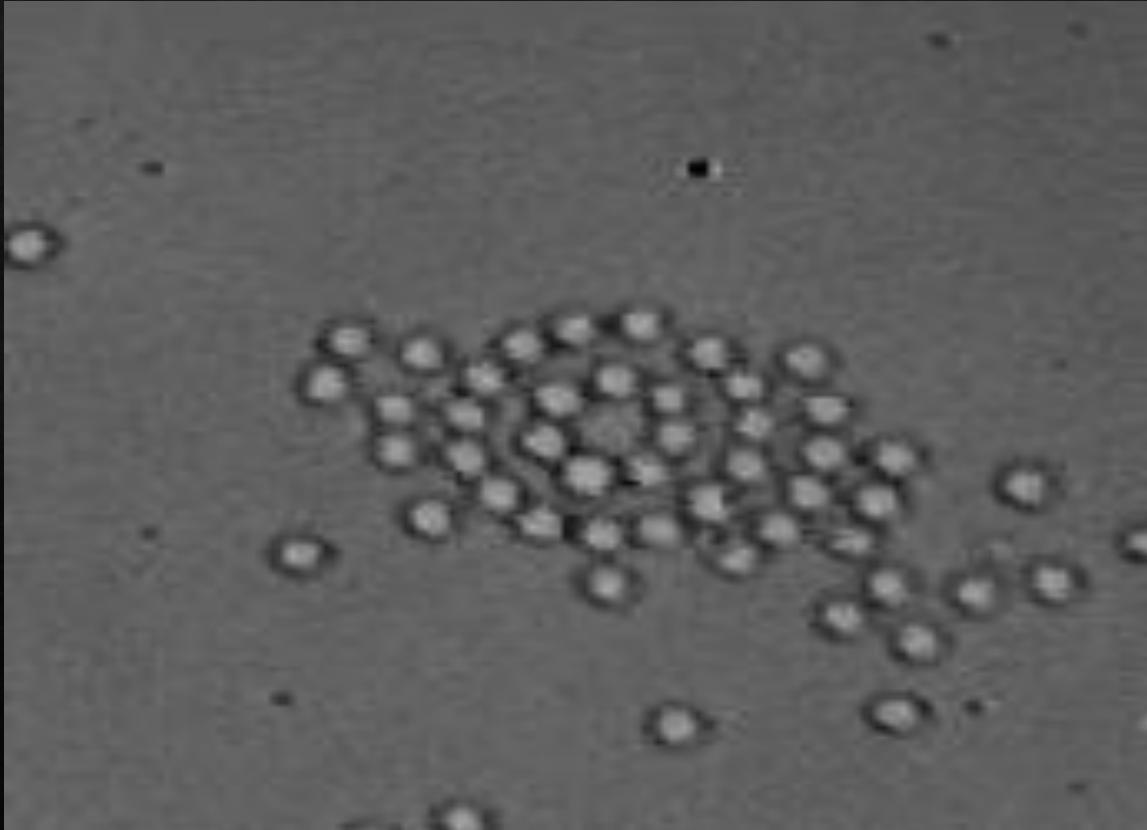


Proyecto Final:

Análisis y Seguimiento de Partículas en un Tamiz Óptico por Medio de Métodos Kernel y visualización con un bot de Telegram.



Física Computacional 2022-2

Profesor: Ricardo Atahualpa
Solórzano Kraemer
Fecha: 24/06/2022

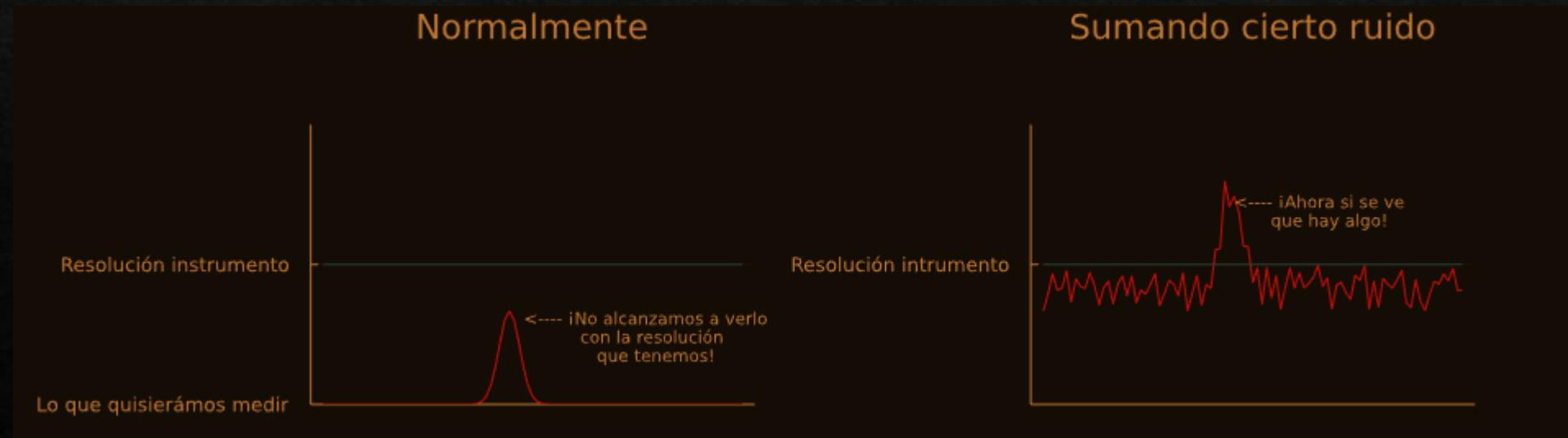
Manuel Jesús Casillas Olivier

The left half of the slide features a minimalist abstract design composed of large, light-colored, faceted geometric shapes, primarily white and light gray, set against a dark gray background. These shapes overlap and interlock, creating a sense of depth and volume.

Motivación:

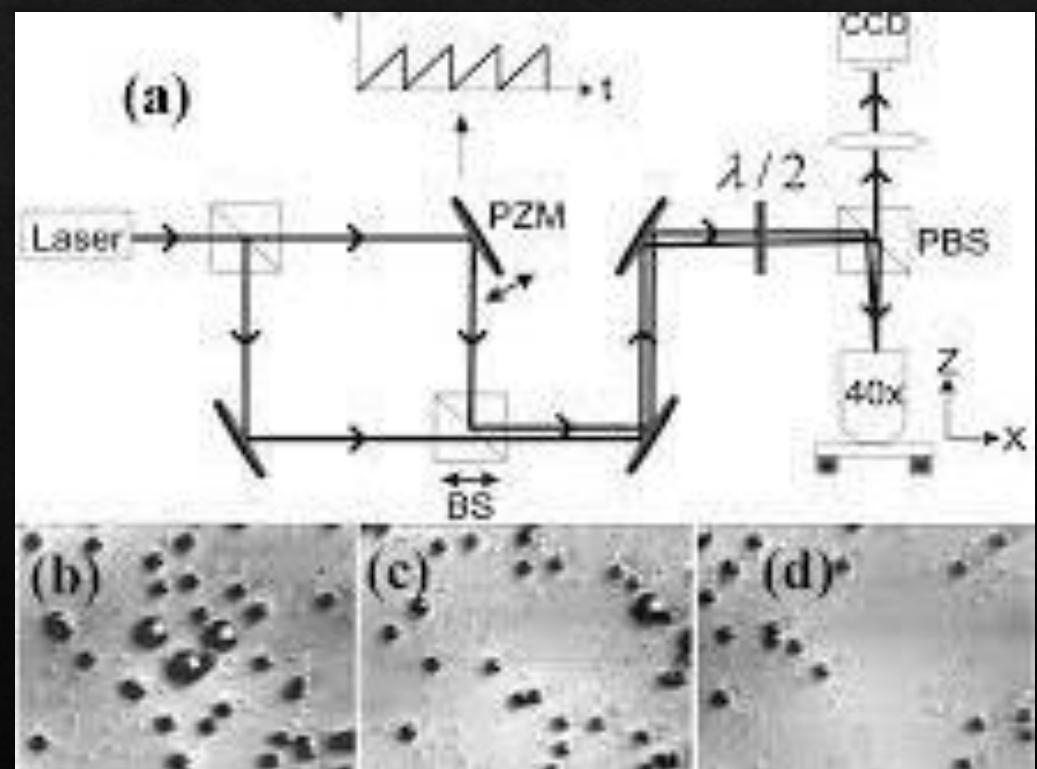
Ruido:

- En ocasiones el ruido en un sistema físico es altamente perjudicial. A veces impide tener mediciones precisas.
- Sin embargo hay otras ocasiones donde el ruido puede no ser tan dañino o incluso puede ser benéfico. Unos de estos casos interesantes donde el ruido nos ayuda sucede en el ámbito de la óptica.



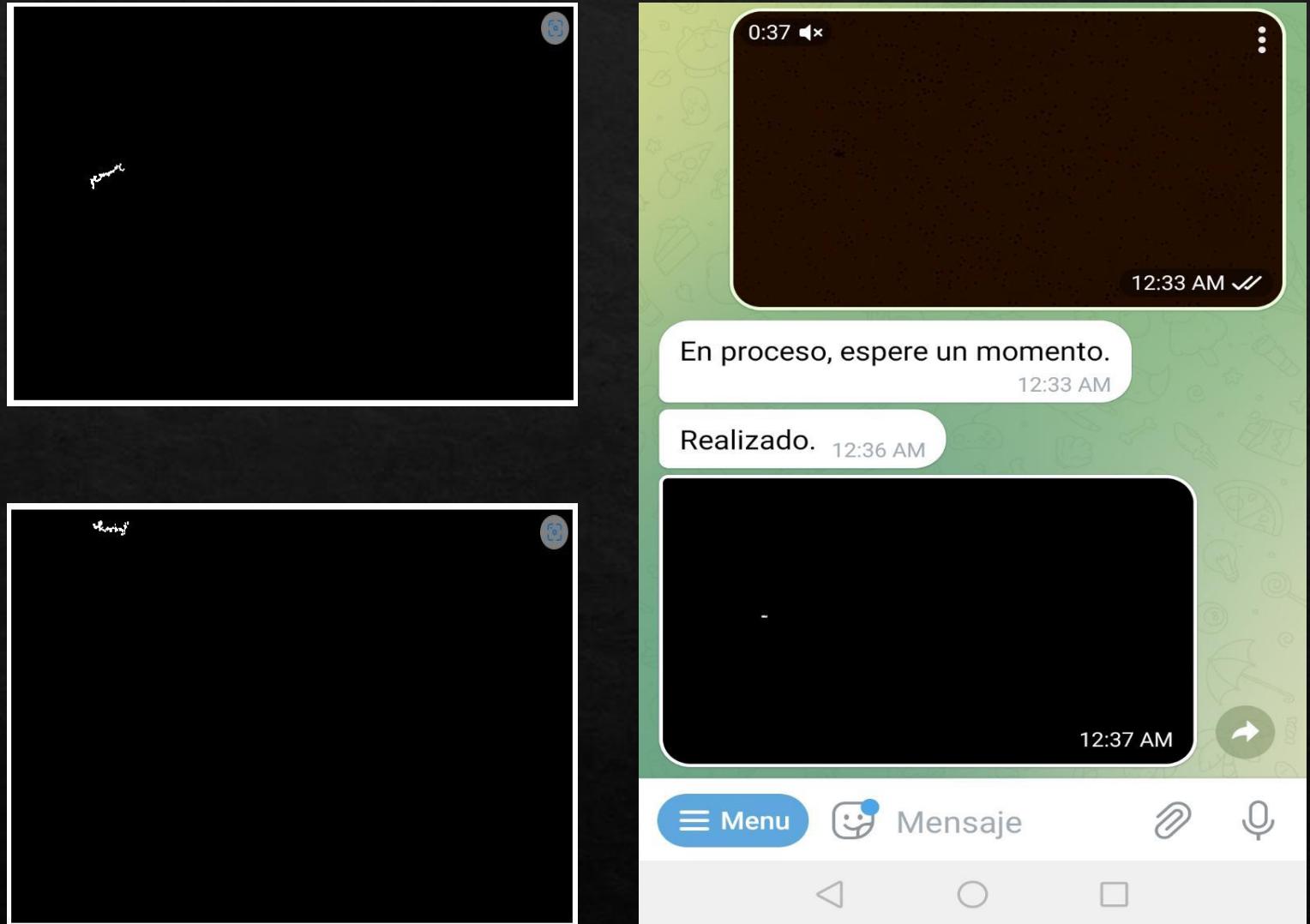
Análisis de un tamiz óptico.

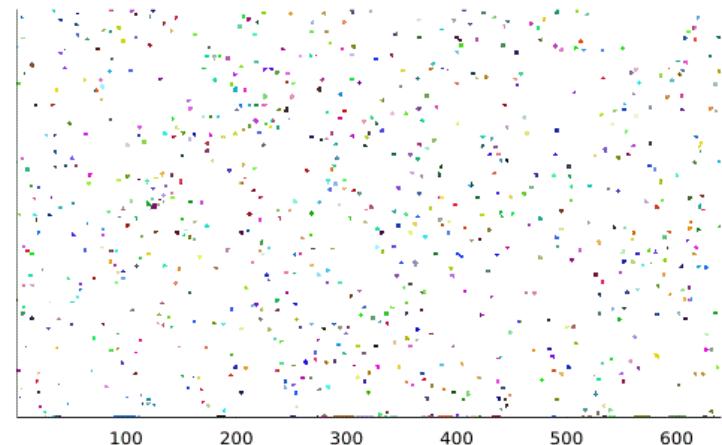
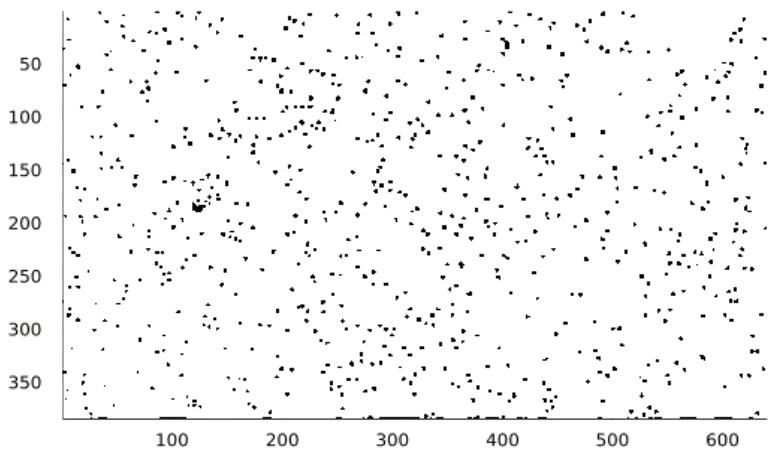
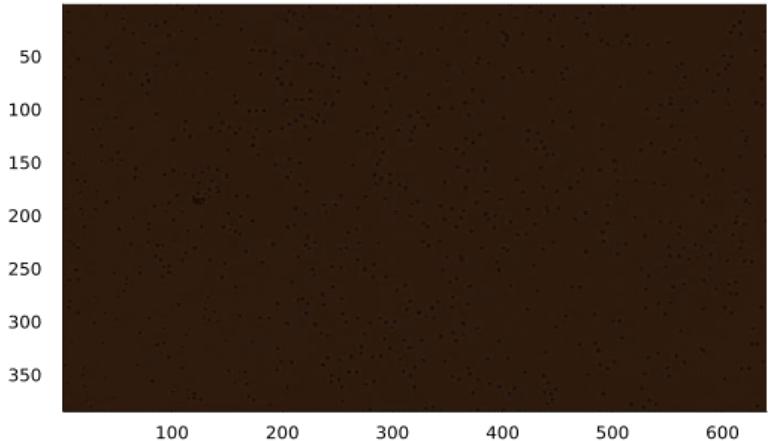
- ❖ Analizar la dinámica de las partículas en un tamiz normalmente requeriría de horas de análisis con un programa de análisis de video convencional y es ineficiente cuando el experimento se quiere repetir muchísimas veces.



Propuesta:

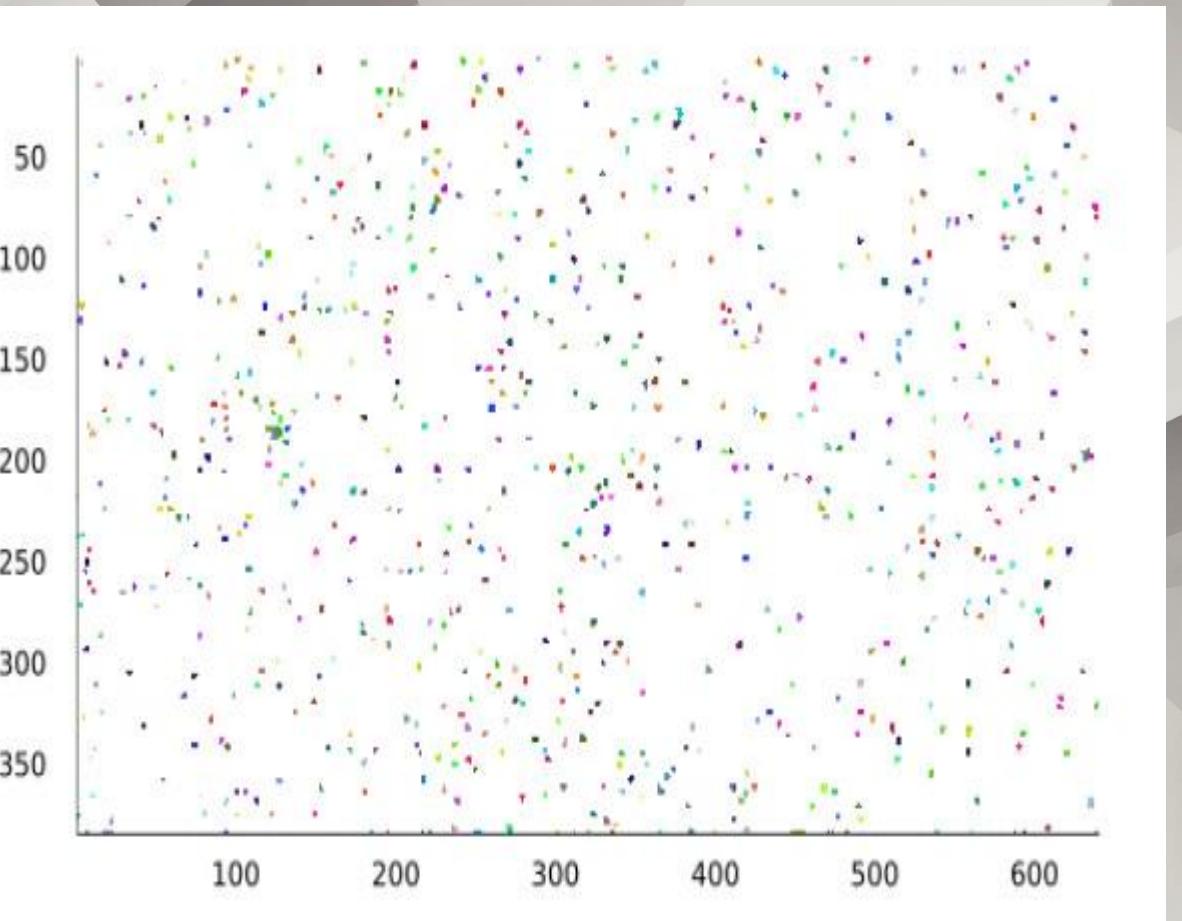
Automatizar el proceso por medio de un bot de Telegram, diseñando un código que cumplirá con los requisitos funcionales, tendrá un diseño pertinente, pulcritud en la implementación, documentación pertinente y sobre todo eficiencia y desempeño general.





Resultados esperados:

- ❖ Recibir por parte del bot la difusión que en este caso de movimiento aleatorio corresponde con el desplazamiento cuadrático medio.



Trabajo realizado:

Primer intento de implementación:

- ❖ Se usaron las paqueterías Telegram para diseñar al bot y VideoIO para hacer una lectura de los frames de cada video.
- ❖ En primera instancia se optó por realizar una función que a cada partícula se le asociará una velocidad en función de los dos frames anteriores:

$$\bar{v}_i^{(frame)} = \frac{\bar{x}_i^{(frame)} - \bar{x}_i^{(frame-1)}}{t^{(frame)} - t^{(frame-1)}}$$

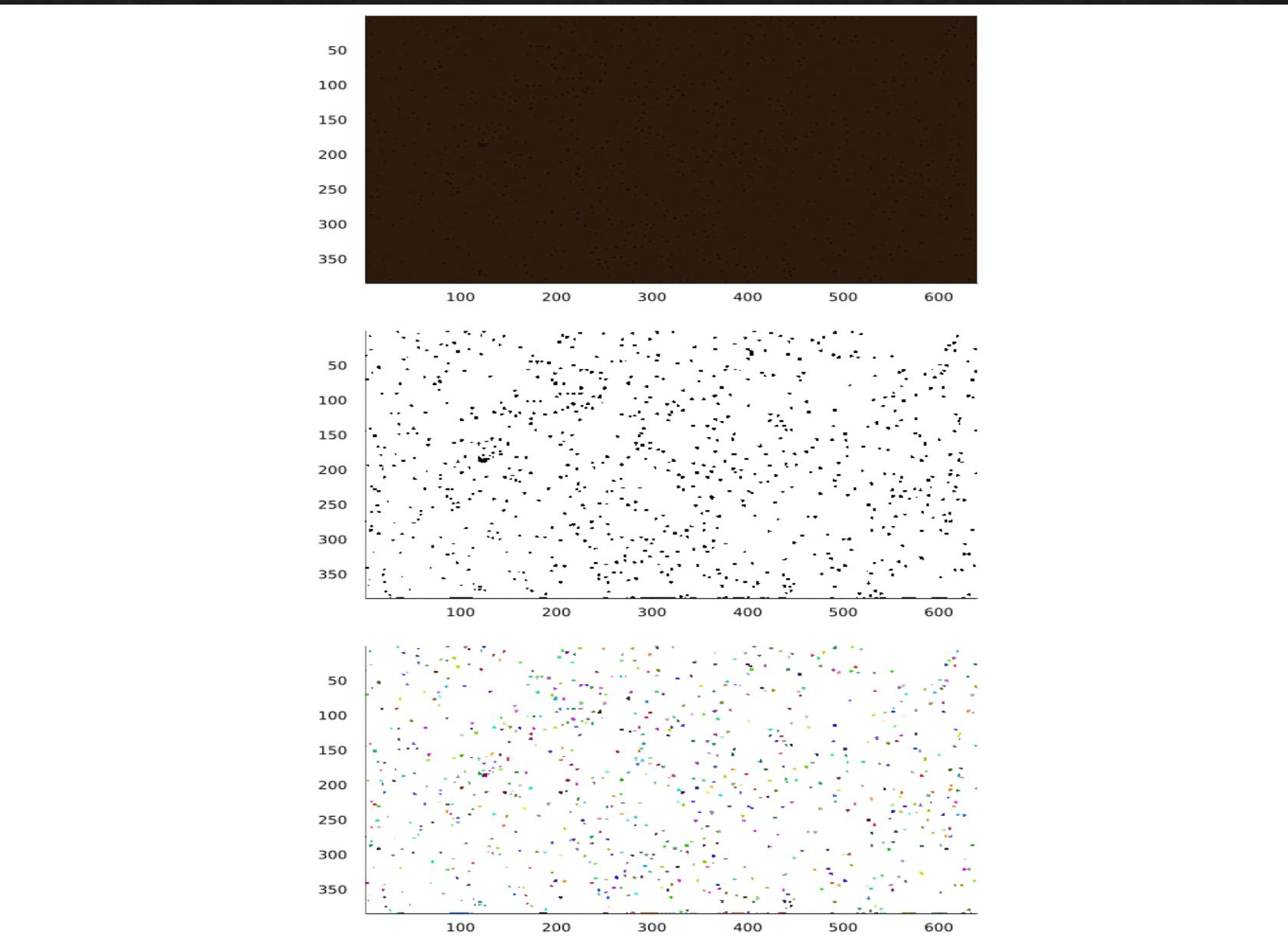
Después se hizo una predicción de la ubicación del frame siguiente:

$$\bar{x}_i^{(frame+1)} = \bar{x}_i^{(frame)} + (t^{(frame+1)} - t^{(frame)})\bar{v}_i^{(frame)}$$

Dado este resultado, se buscaría la partícula más cercana a la posición predicha, sin embargo solamente se tomó la más cercana porque se movían lo suficientemente lento y además era demasiado aleatorio para que la velocidad fuera “casi” constante.

Segundo Intento de implementación:

- ❖ Por grupos:



Algoritmo de análisis del segundo intento

Implementación:

Pseudocódigo:

importar paquetterías

Construir un objeto AVInput para acceder a los flujos de vídeo y audio de un contenedor de vídeo

Acceder al flujo de vídeo en un AVInput

devolver un objeto VideoReader, también puede utilizar un nombre de archivo, en lugar de un AVInput

Generar matriz que contiene cada pixel

Agrupar matrices que contienen por cada pixel el numero de particula correspondiente

funcion graficar grupos:

para cada grupo

definir colores RGB

continuar:

regresar colores

regresar grafica en una imagen.png

```
import VideoIO
using Plots
using Images

function grup2!(grupos,cantidadgrupo,x,y;pixelpp=20)

    size_x=size(img)[1]
    size_y=size(img)[2]
    #print(x,",",y==size_y,"*")
    if x==1
        if y==1
            grupos[x,y]=cantidadgrupo
            cantidadgrupo+=1
        elseif y==size_y
            if grupos[x,y-1]!=0
                if count(i->(i== grupos[x,y-1]),grupos)<pixelpp
                    grupos[x,y]=grupos[x,y-1]
                end
            else
                grupos[x,y]=cantidadgrupo
                grupos[x+1,y]=cantidadgrupo
                cantidadgrupo+=1
            end
        else
            if grupos[x,y-1]!=0
                if count(i->(i== grupos[x,y-1]),grupos)<pixelpp
                    grupos[x,y]=grupos[x,y-1]
                end
            else
                grupos[x,y]=cantidadgrupo
                cantidadgrupo+=1
            end
        end
    end
end
```

Pruebas con kernels:

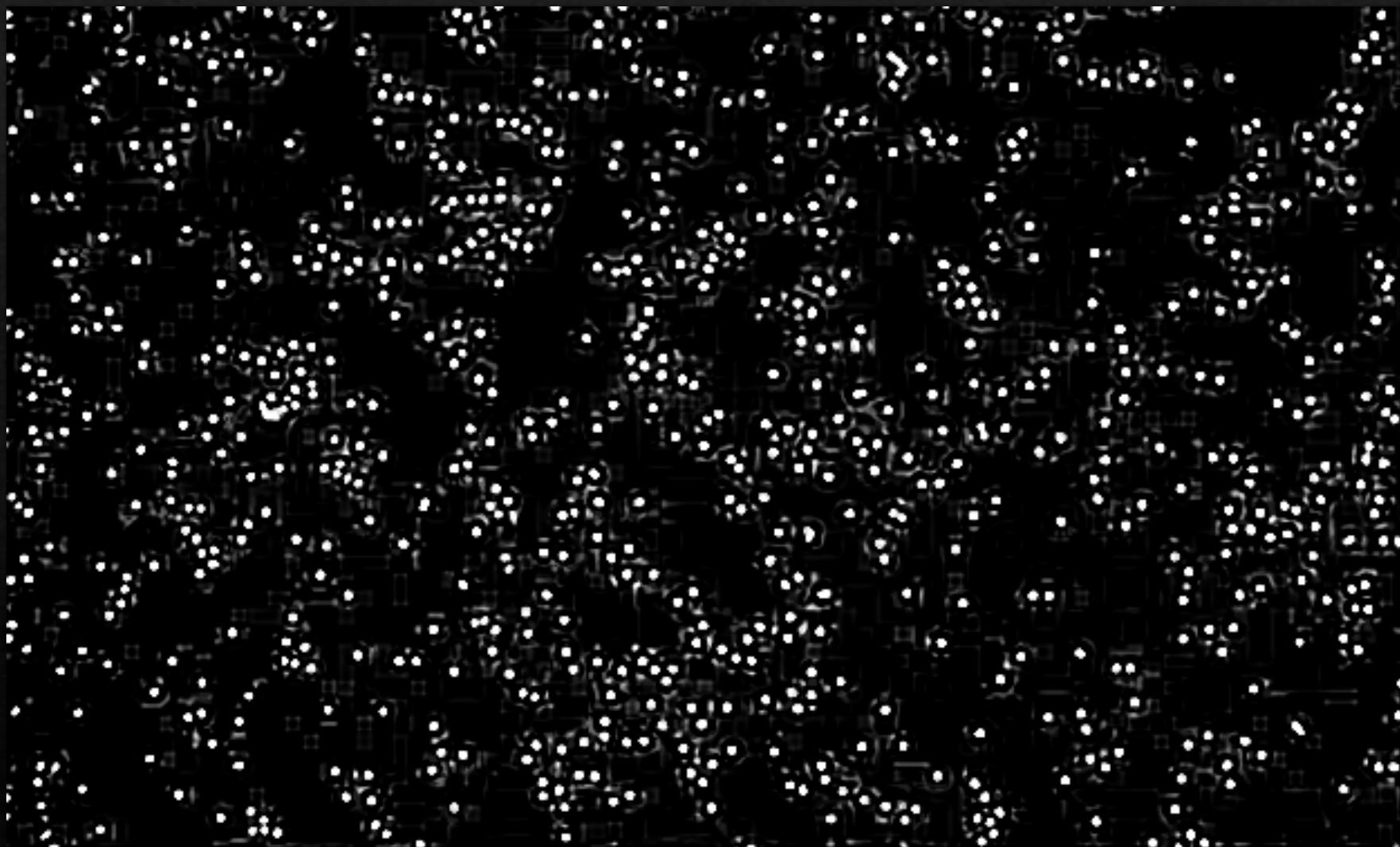
laplacian+1-8:

1	1	1
1	-8	1
1	1	1

laplacian-1+8:

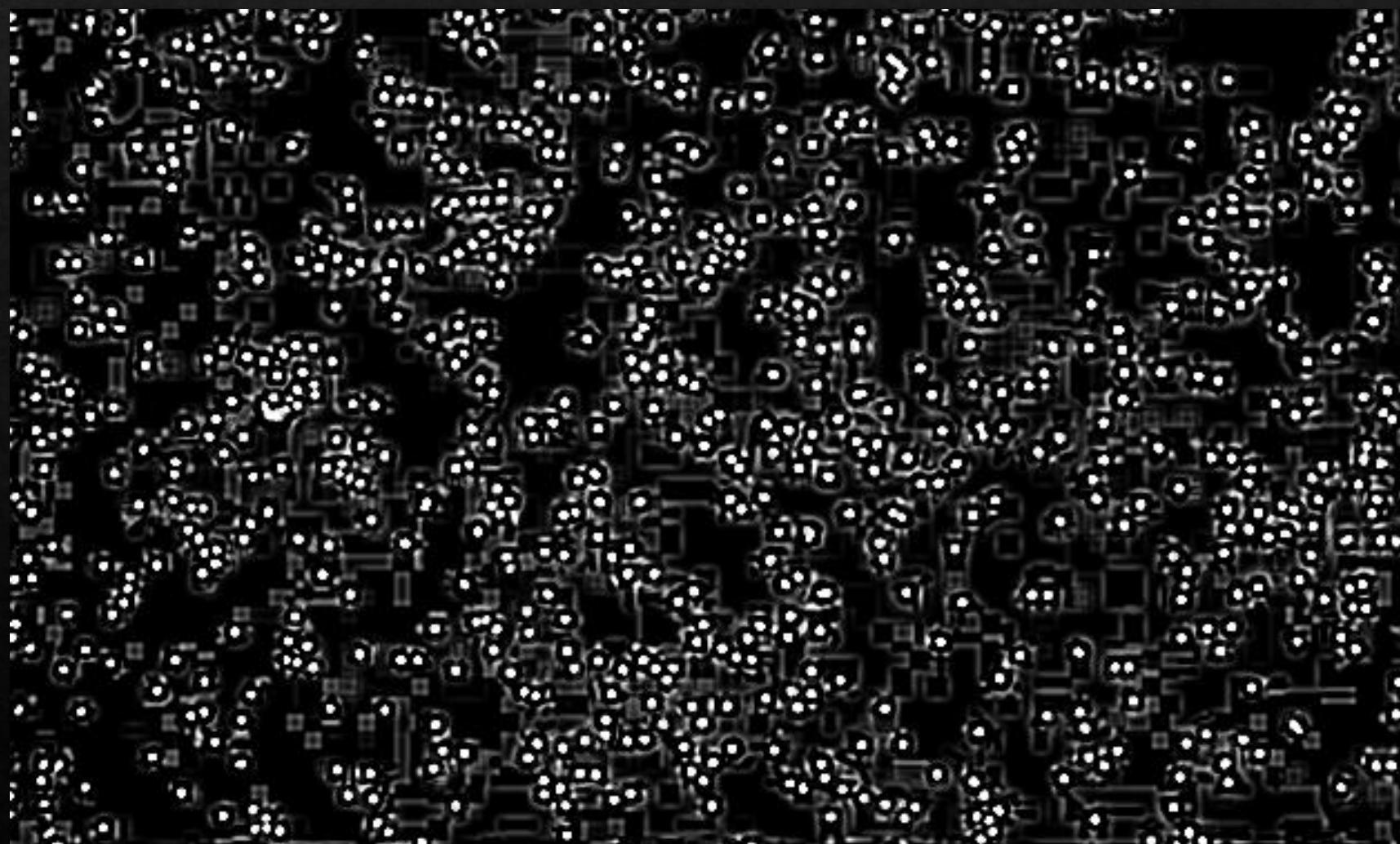
-1	-1	-1
-1	8	-1
-1	-1	-1

log00322:



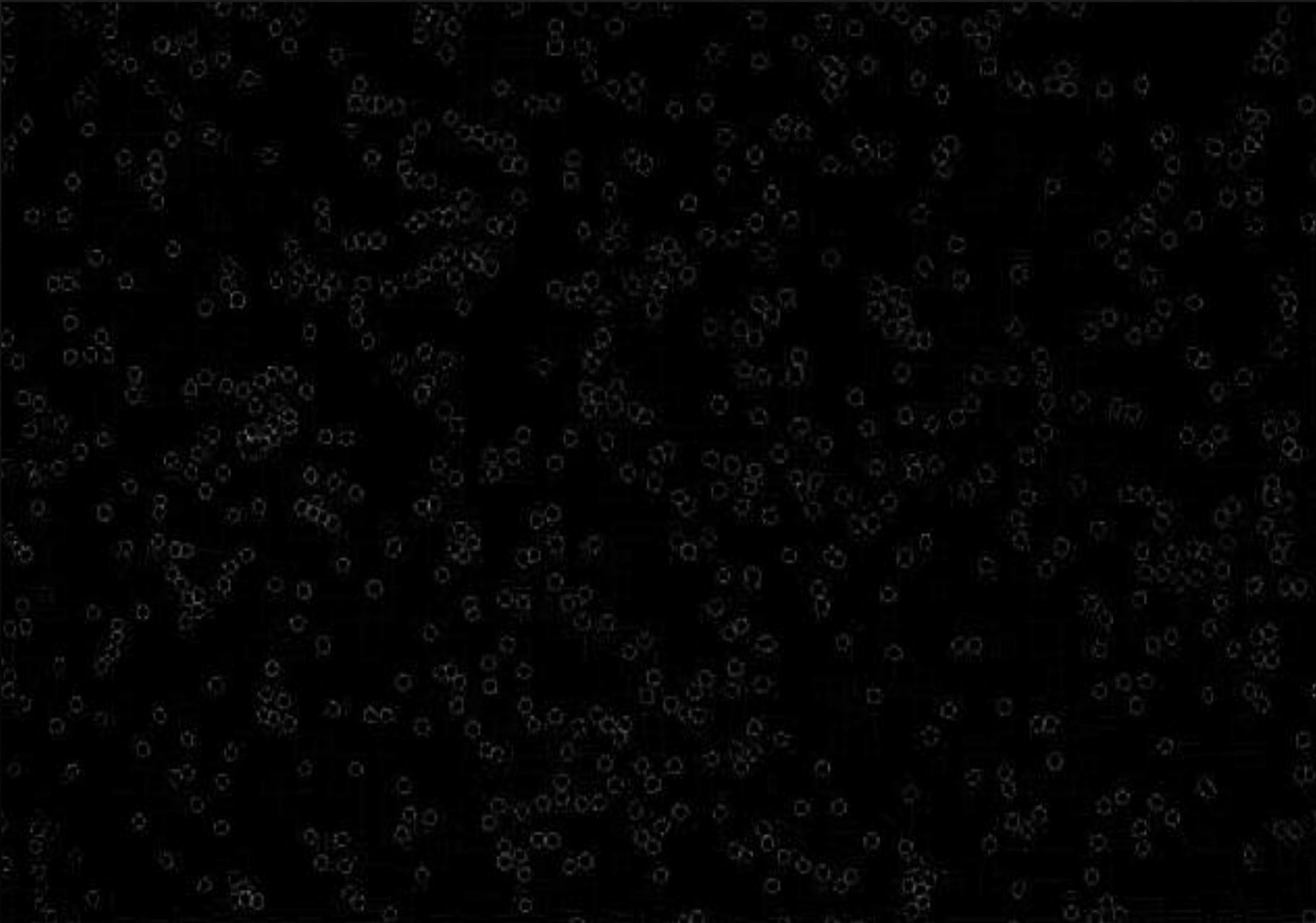
0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

log01122:



$$\begin{pmatrix} 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \\ 0 & 2 & 3 & 5 & 5 & 5 & 3 & 2 & 0 \\ 3 & 3 & 5 & 3 & 0 & 3 & 5 & 3 & 3 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \\ 2 & 5 & 0 & -23 & -40 & -23 & 0 & 5 & 2 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \\ 3 & 3 & 5 & 3 & 0 & 3 & 5 & 3 & 3 \\ 0 & 2 & 3 & 5 & 5 & 5 & 3 & 2 & 0 \\ 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \end{pmatrix}$$

log00-1:



0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Imagen binaria:



Imagen con centroides:



Diseño del algoritmo de análisis:

Pseudocódigo:

```
importar paqueteterías

Definir a cada particula como un objeto con las propiedades:
    * número de identidad = entero
    * posición inicial = vector en 2D
    * trayectoria = vector en 2D
    * frames = vector en Int64
    * posición final = vector en 2D

--ENTRADA--
<Ingresa imagen>
Definir función1 que regresa una imagen binaria:
    1. Aplicar un filtro gris a la imagen
        1.0. Hacer un gradiente de color sencillo
    2. Clasificar de acuerdo a intensidad relativa
        2.1. Se vuelve blanco o negro de acuerdo con la intensidad de grises
        2.2.
        Si es muy claro, entonces:
            se hace blanco
        sino:
            se hace negro
    3. Regresar imagen

Definir función2 que encuntra las posiciones de las particulas por frame:
    1. Se procesa la imagen
    2. Se encuentran blobs mediante Laplacian of Gaussian
    3. Dfinición de vector localizador de las posiciones de los blobs si no se encuentran en la frontera
    4. Regresar vector localizador

Definir función3 que saca la distancia entre dos vectores:
    distancia = raíz_cuadrada((entrada1(punto1)-entrada1(punto2))^2 + (entrada2(punto1)-entrada2(punto2))^2)

Definir función4 encuentra la posición más cercana a otra:
    1. regresar sort (localizador, sustituir en indices)
```

```
| Definir función5 que sigue las partículas:
|     1. Iterar cada frame del video
|     2. Identificar a la misma partícula en el siguiente frame
|     3. Almacenar trayectorias:
|         3.1. Si las partículas se unen, terminar de almacenar y unir las dos o más trayectorias
|         3.2. Terminar cuando alguna partícula pasa por la frontera
|             3.2.1. Mandar trayectoria a un arreglo de partículas que aparecieron pero se dejaron de seguir

Definir función6 que hace el desplazamiento cuadrático:
    1. Recibir seguimiento de partículas de función5
    2. Regresar desplazamiento cuadrático medio como una lista de desplazamientos hasta el enésimo frame

Definir función7 que grafica el desplazamiento cuadrático como función del tiempo.
    1. Usar paquete plots
    2. definir difusión
    3. Regresar imagen
```

Diseño del bot de telegram:

Pseudocódigo:

```
importar paqueteterías
conectar con el algoritmo de análisis incluyendo todas sus funciones.jl

definir token y cliente
--Iniciar bot--
--ENTRADA--

<mensaje del usuario>

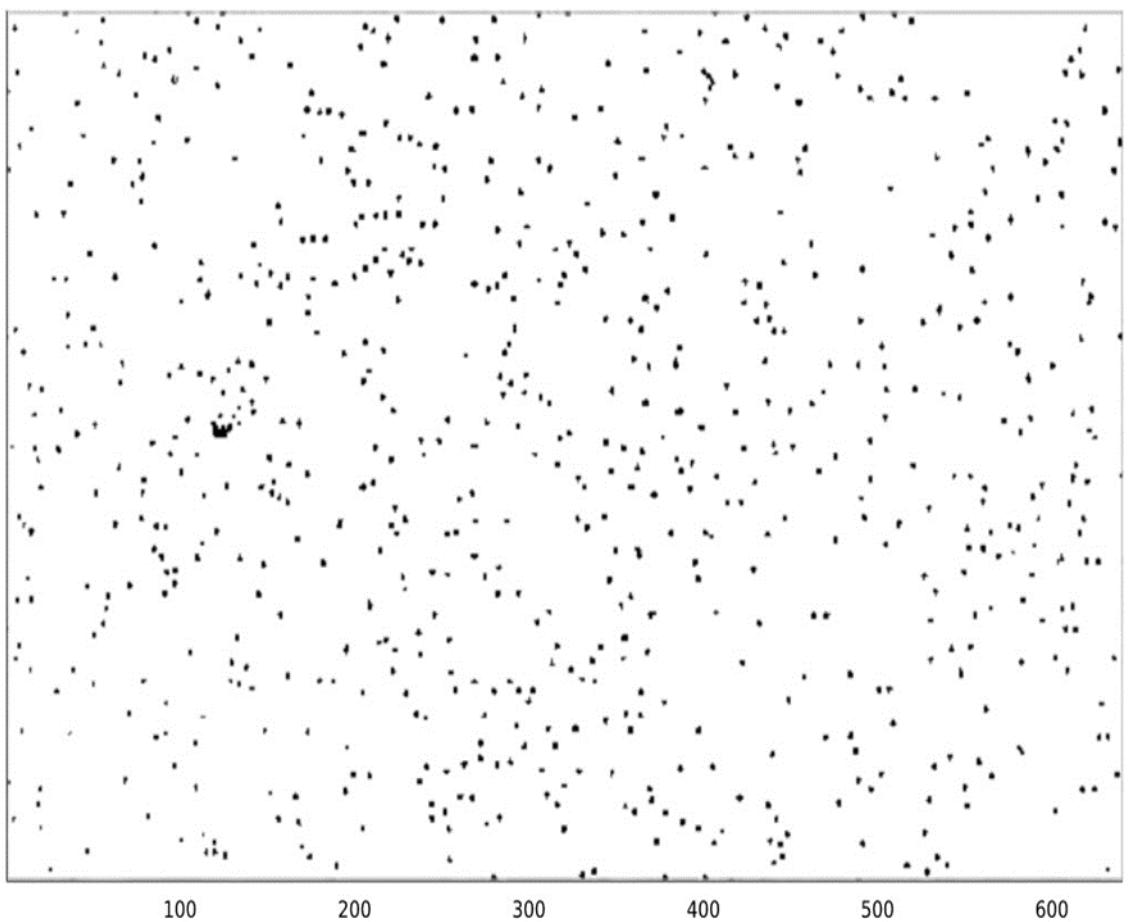
Si se envía un video:
    1. Asignar ID al video
    2. Almacenar video
    3. Llamar al servidor de telegram y guardar
    4. Enviar mensaje al usuario "Análisis realizado"

sino:
    definir comando "iniciar", entonces:
        1.Mandar mensaje al usuario "Este es un bot para analizar videos.
        Manda un video para analizar."

sino:
    definir comando "mostrar análisis", entonces:
        1. Graficar MSD:
            1.1. Conectar con funciones.jl
            1.2. Graficar
            1.3. Guardar gráfica en una imagen.png
        2. Mandar al usuario la imagen.png con la gráfica
        3. Calcular difusión:
            3.1. Conectar con funciones.jl
            3.2. Realizar método para calcular la difusión
            3.3. Guardar la difusión como variable temporal
        4. Mandar al usuario la difusión.
```

```
sino:
    definir comando "trayectoria"
        1. Conectar con funciones.jl
        2. Realizar método para calcular la trayectoria
        3. Graficar trayectoria
        4. Guardar trayectoria como imagen.png
        5. Mandar al usuario la imagen.png

Si el usuario no escribe un comando como los definidos anteriormente, entonces:
    imprimir "Error" y mandar al usuario|
```



Resultados y
mediciones:



Actividades

Telegram Desktop

Jue 23 de jun 21:29

Zoom Meeting

Bot de análisis

bot

What can this bot do?

Este es un bot al cual se le puede enviar un video del movimiento de varias partículas para identificar la trayectoria individual de cada una de ellas y posteriormente realizar un análisis.

/start Comienza.

/trayectoria Muestra la trayectoria de una de las partículas.

/mostrar_análisis Envía el análisis completo.

/graficar_msd_log Manda gráfica log del MSD.

/graficar_md_completo Manda los análisis de MD completo.

Write a message...

329

Bot de análisis
bot

What can this bot do?
Este es un bot al cual se le puede enviar un video del movimiento de varias partículas para identificar la trayectoria individual de cada una de ellas y posteriormente realizar un análisis.

June 23

/start 22:42 ✓

Este es un bot para analizar videos.
Manda un video para analizar. 22:45

00:36 🔍

Analizando, espere un momento. 22:45

Análisis realizado. 22:49

Menu Write a message...



Bot de análisis
bot

Análisis realizado. 22:49

/mostrar_análisis 23:06 ✓

$\langle x^2 \rangle$ vs t

Desplazamiento cuadrático medio $\langle x^2 \rangle(t)$

Tiempo t [número de frames]

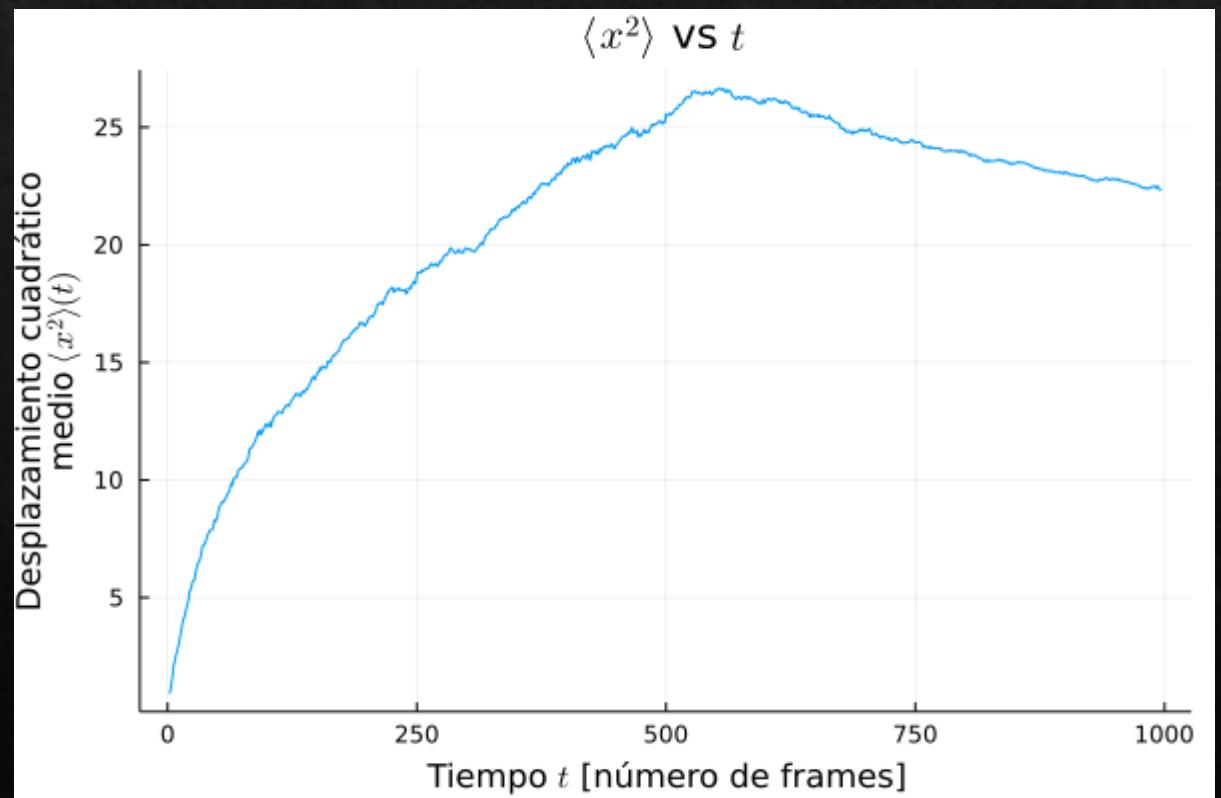
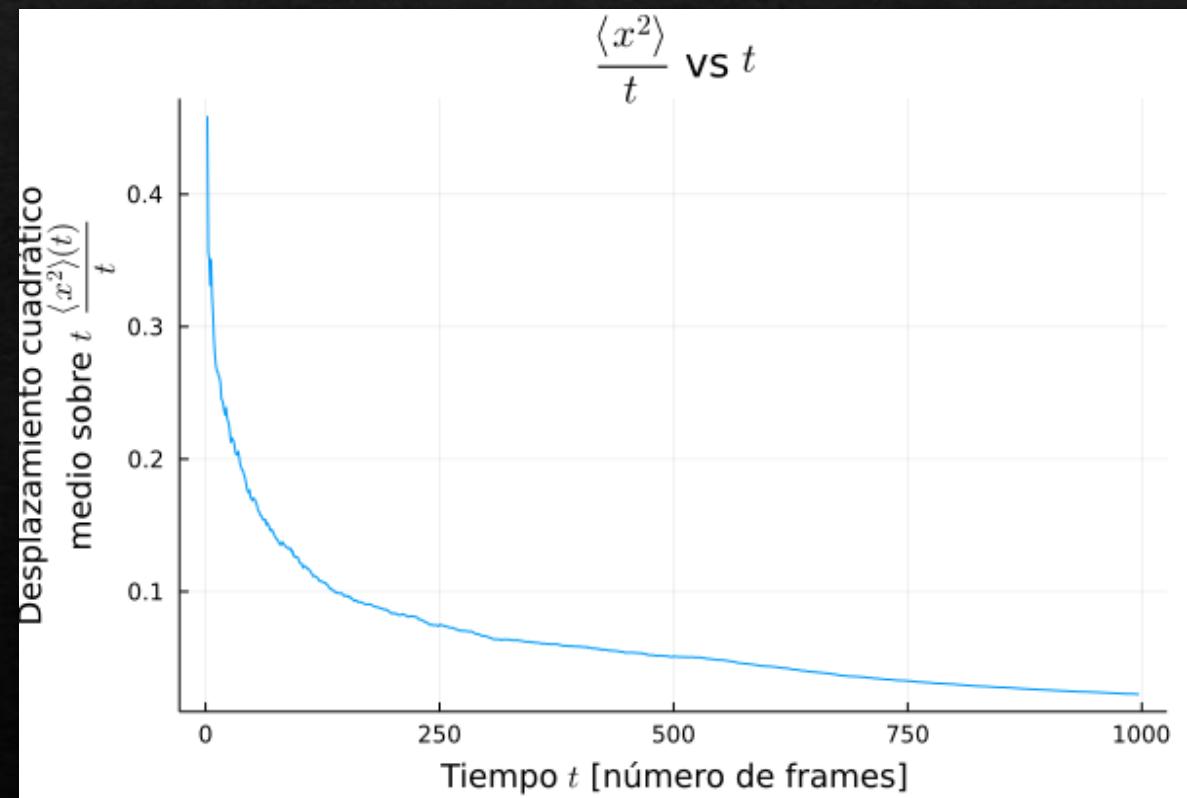
$\langle x^2 \rangle$ vs t

Desplazamiento cuadrático medio sobre t $\langle x^2 \rangle(t)/t$

Tiempo t [número de frames]

El valor de la difusión es 0.02237844206622619 23:06

Menu Write a message...



The left half of the slide features a complex, abstract geometric pattern composed of numerous white and light gray triangles. These triangles overlap and interlock to create a sense of depth and complexity, resembling a crystal lattice or a molecular structure.

Conclusiones:

- I. Se logró diseñar un algoritmo de análisis que cumple con los requisitos mínimos de funcionamiento.
- II. Se logró obtener un bot funcional que ejecuta los comandos requeridos
- III. Se logró establecer una correcta implementación/conexión entre el bot y las funciones.jl
- IV. Se lograron los objetivos propuestos en la sección de motivación estableciendo un bot que devuelva resultados según el comando.

Pensando a futuro:

- ❖ Tanto el código de funciones.jl como el bot podrían optimizarse para obtener un mejor rendimiento y desempeño en general.
- ❖ Un posible mantenimiento sería la utilización de paqueterías nuevas que todavía no se han creado que sean más especializadas para el análisis de la dinámica de muchas partículas, reduciendo los tiempos de carga y consumo de memoria.

Referencias:

Volke Sepúlveda, K., Ricárdez Vargas, I., Rodríguez Montero, P., & Ramos García, R. (2006, 23 marzo). *Modulated optical sieve for sorting of polydisperse microparticles*. Instituto de Física. Recuperado 23 de junio de 2022, de https://www.fisica.unam.mx/personales/karen/karen_docs/APL2006_inaoe.pdf

colaboradores de Wikipedia. (2022, 18 junio). *Desplazamiento cuadrático medio*. Wikipedia, la enciclopedia libre. Recuperado 23 de junio de 2022, de https://es.wikipedia.org/wiki/Desplazamiento_cuadr%C3%A1tico_medio

Volpe, G., Volpe, G., & Gigan, S. (2014, 5 febrero). Brownian Motion in a Speckle Light Field: Tunable Anomalous Diffusion and Selective Optical Manipulation. *Nature*. Recuperado 23 de junio de 2022, de <https://www.nature.com/articles/srep03936.pdf>