INSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA
ESCUELA DE INGENIERÍA Y GESTIÓN

Detección y seguimiento de partículas autopropulsadas

AUTORES: Nardini, Gonzalo Martín (Leg. Nº 54387)
Sakuda, María Eugenia (Leg. Nº 53191)
Vázquez, Diego Nicolás (Leg. Nº 54377)

TUTOR: Patterson, Germán Agustín

TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN INFORMÁTICA

Índice

1- Objetivos

Las partículas autopropulsadas son agentes autónomos que convierten cierta cantidad de energía interna en movimiento. Es de interés estudiar la interacción entre este tipo de agentes ya que pueden producir comportamientos emergentes, como es la formación de rebaños, que modifican fuertemente el comportamiento individual.

En el centro IN3D de la facultad, se desarrollan experimentos con agentes autopropulsados que mediante un sistema de procesamiento de imágenes se determinan las posiciones y orientaciones espaciales. Los agentes utilizados son los HEXBUG nano, también conocidos como VDVs (del inglés, vibration driven vehicles). El tamaño de éstos es de 43 mm de largo, 15 mm de ancho y 18 mm de altura como se puede observar en la Fig. 1a. Con el fin de hacer el trackeo, cada uno de estos agentes está etiquetado con dos circunferencias de colores que permiten ubicarla en el espacio y asignarle la orientación en cada uno de los frames del video.

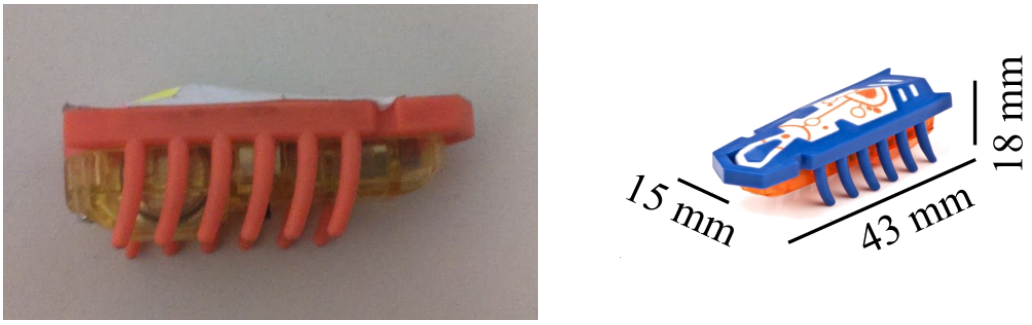


Figura 1: a la izquierda imagen del perfil de un HEXBUG nano modelo 477-2409-00GL12. A la derecha foto en perspectiva.

Sin embargo, este sistema existente no logra identificar cada partícula de forma unívoca introduciendo problemas para realizar el *tracking*, por ejemplo en caso de que la misma no sea detectada durante un frame lo cual puede suceder por diversas causas. Por este motivo, el objetivo de este proyecto es diseñar e implementar un sistema de etiquetas que permita el reconocimiento individual de los agentes y el software necesario para poder realizar el proceso de *tracking* a partir de las mismas.

Otra característica importante que se tuvo en cuenta fue la de implementar un sistema de corrección de errores. En particular, se aplicó dicho sistema en aquellos frames donde por distintas condiciones de iluminación o movimiento de los agentes

no se podían detectar correctamente las componentes del sistema de etiquetado. Además, se determinó que el procesamiento no era necesario que fuese realizado en tiempo real y que el número mínimo de etiquetas requeridas dada la cantidad de agentes utilizados en los experimentos fuera de 50 unidades.

Finalmente, dadas las condiciones de los experimentos realizados por el laboratorio donde la filmación era automáticamente guardada en una secuencia de archivos consecutivos por la cámara utilizada, se decidió permitir la paralelización del procesamiento según la capacidad de la máquina donde se corriera el sistema.

2- Plan de trabajo

El plan de trabajo planteado originalmente estaba segmentado en tres etapas principales: diseño de etiquetas, detección de las etiquetas en el video, generación de los archivos de salida.

Estas etapas se fueron dividiendo en subtareas más específicas y, además, durante el desarrollo del proyecto se fueron agregando otras rutinas que permitan mejorar o solucionar algunas de las etapas de implementación. En la *tabla 1* se puede observar el detalle de tiempo final utilizado para cada tarea.

Tarea	Duración
Investigación de elementos de trackeo (etiquetas)	30 h
Investigación y prueba de frameworks para el procesamiento de imágenes	10 h
Configuración del entorno de desarrollo	40 h
Diseño y prueba de las etiquetas	25 h
Implementación básica (detección de la partícula: posición y orientación)	30 h
Exportación a archivos y parametrización (ver o no formato visual)	10 h
Migración de código - optimización de tiempos	40 h
Ampliación de la cantidad de etiquetas disponibles	28 h
Corrección de la lente	6 h
Interpolación	20 h
Paralelización del procesamiento	10 h
Escritura del manual de usuario	15 h
Escritura del informe	40 h

Tabla 1: tiempos dedicados por tarea.

3- Implementación

En esta sección se explica cómo se llevó a cabo el proceso de implementación del sistema previamente explicado.

Cabe mencionar que originalmente el desarrollo se inició en Python por las facilidades que brindaba el lenguaje y la compatibilidad con OpenCV. Sin embargo, a medida que el proyecto fue avanzando se determinó que los tiempos de procesamiento en algunas secciones del código eran demasiado elevados. Por este motivo, se realizó una prueba de concepto en C++ y se comparó la performance entre ambos lenguajes. Los resultados indicaron que el programa tardaba un décimo del tiempo original, por lo que se optó por migrar la implementación existente y continuar el desarrollo en C++.

En la *tabla 2* se presenta el algoritmo principal de procesamiento de un video. Cada uno de estos pasos será descrito en las siguientes secciones.

Algoritmo 1 Video Processor

```
1:  for i := 1 to frameCount do                    # Itera por cada frame del video
2:      undistortFrame := undistortLens(frames[i]);      # Corrección de la lente
3:      extremizeColorFrame := extremizeColors(undistortFrame);  # Detección
                                                                  # de colores
4:      components := getComponents(extremizeColorFrame);  # Detección de
                                                           # componentes
5:      history[i] := detectAgents(components);              # Detección
                                                             # de agentes

        # Se guardan los frames cuando se procesaron suficientes para
        # realizar la interpolación de caso de ser necesaria
6:      if i > maxFramesToInterpolate then
7:          saveFrame(history, i - maxFramesToInterpolate);
8:      end if
9:  end for
```

```
    # Se guardan los últimos frames a parte porque no se tienen datos para
    interpolar normalmente
10: for j: = frameCount - maxFramesToInterpolate to frameCount:
11:     saveFrame(history, j);
12: end for
13: where
14: proc saveFrame(history, i) □
15:     history[i] := interpolate(history, i);          # Interpolación de posiciones
16:     saveToFiles(history, i);                    # Guardado de archivos de salida
17: end
```

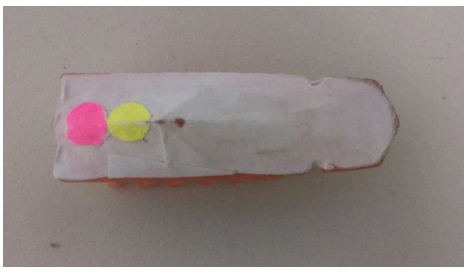
Tabla 2: algoritmo principal de procesamiento de un video.

3.1- Definición y diseño de la etiqueta

Esta fue la etapa inicial que incluyó tiempo de investigación para la interiorización en los diversos métodos utilizados como IDs (del inglés *identity* o identidad). Si bien las opciones analizadas fueron códigos QR (del inglés Quick Response code o "código de respuesta rápida") o códigos de barras, texto y la combinación de colores las primeras opciones fueron descartadas por el tamaño y la velocidad del agente autopropulsado donde se iba a utilizar. Dadas las condiciones de tamaño de los agentes la definición de las primeras etiquetas no puede garantizar ser suficiente para el procesamiento en todos los casos.

Una vez definida la utilización de la combinación única de colores como identificador, se prosiguió a evaluar el diseño de la misma (cantidad, forma, ubicación y material de colores). En la Fig. 2 se puede observar esta evolución comenzando por la etiqueta existente (1a).

(a) (c)



(b) (d)

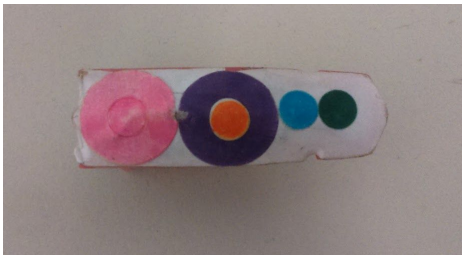


Figura 2: Evolución de las etiquetas desde la versión existente en la imágen 1 hasta la versión final en la imágen 4 (a) - (d).

Primero se evaluaron las geometrías que resultaban más fáciles de detectar alternando entre distintos tamaños de círculos y franjas de colores. Al momento de analizar los frames, las etiquetas realizadas con circunferencias de colores presentaban menor cantidad de píxeles (Fig. 2b) dejando mayor cantidad de espacio sin utilizar. Debido a la forma del modelo del agente utilizado (HEXBUG nano) no era posible reubicar estas geometrías para mejorar esta condición por lo que se descartó esta opción. De este modo, se concluyó que las formas serían franjas de colores que ocuparan la mayor cantidad del espacio disponible.

Una vez definida la forma se prosiguió analizando el material para confeccionar cada etiqueta. En un principio las pruebas se venían realizando con fibras de colores presentando el inconveniente de la poca uniformidad en el color obtenido. Por este motivo, se pasó a utilizar papel glacé de colores. Además, en esta fase se evaluaron cuán distintos eran los colores disponibles y cómo responden los mismos luego de ser fotografiados para el procesamiento digital. Esto será desarrollado con mayor profundidad en la sección 3.2.

Las cotas para la cantidad de franjas para cada etiqueta se determinaron de forma experimental. Se necesitaban la mayor cantidad de franjas posibles para alcanzar la cantidad de etiquetas necesarias. Sin embargo, si las mismas eran demasiado pequeñas el proceso de detección sufría errores. Por este motivo, el número de franjas por etiquetas que se decidió utilizar a partir de las pruebas realizadas fue de 4. Se requerían dos de estas franjas para determinar el sentido u orientación lo que a lo largo de la investigación se nombran como “cola” y “cabeza” del agente. Dado que una vez digitalizados siete colores parecían ser distinguibles, la cantidad de etiquetas que se lograron armar con las condiciones establecidas

(utilizando un color para la “cola”, otro para la “cabeza” y los otros cinco colores en las franjas restantes) hasta el momento eran de veinte unidades calculadas según:

$$\#etiquetas = \binom{5}{1}\binom{4}{1} = 20\,, \qquad (1)$$

donde los símbolos $\binom{a}{b}$ representan la operación combinatoria entre a y b.

Durante el desarrollo del programa se agregaron dos colores más utilizando uno de ellos para la cola y otro para la cabeza superando de este modo el número requerido de cincuenta unidades, según los objetivos especificados. La cantidad de etiquetas alcanzadas es para ochenta agentes obtenidas a partir de:

$$\#etiquetas = \binom{5}{1}\binom{4}{1}\binom{2}{1}\binom{2}{1} = 80\,. \qquad (2)$$

En algunos experimentos, los agentes se superponían entre sí debido a su vibración característica. Esto dificultaba la individualización de los mismas dentro de la imagen (por ejemplo si las cabezas de dos partículas se consideran una única unidad). Para contrarrestar este efecto, se agregó un borde blanco en el perímetro a cada unidad que las delimite. La Fig. 3 muestra un ejemplo de etiqueta utilizada para la identificación de los VDVs. En la misma, se pueden observar los 4 colores de identificación y el perímetro blanco delimitante.

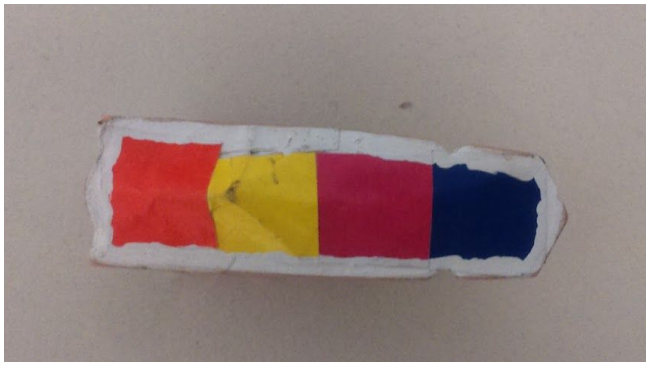


Figura 3: diseño final de la etiqueta. Los colores de de izquierda a derecha: rojo, amarillo, fucsia y azul.

3.2- Caracterización de los colores

Las condiciones para determinar el color de un píxel pueden ser más o menos complejas según el color y su sensibilidad a los cambios de iluminación. Estas reglas consisten en relaciones entre las componentes R (Red/Rojo), G (Green/Verde) y B (Blue/Azul) que pueden tomar valores entre 0 y 255. A continuación se detallan para cada uno de los colores.

- **Azul**: En los casos con poca iluminación, con un valor de B menor a 80, se requiere que este valor supere en por lo menos 18 al G y R. Además, la diferencia entre el R y el G debe ser menor a 10. Cuando

se cuenta con mayor iluminación los valores tienden a subir, pero el B es el más afectado, por lo que cuando el valor de B es mayor a 80 se requiere de una diferencia mayor a 30 con G y R para ser considerado azul.

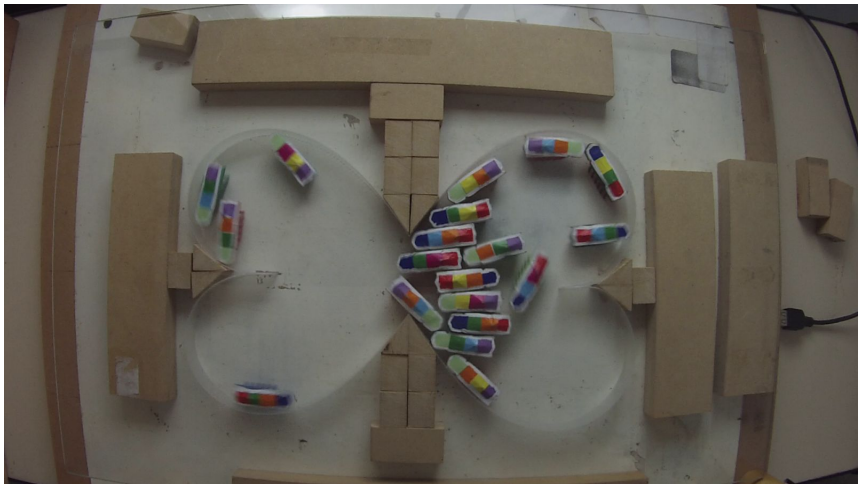
- **Rojo**: No se hace diferencia por iluminación. En ambos casos se pide que la componente R supere a las otras dos por 40 o más y que la diferencia entre B y G sea menor a 10.
- **Naranja**: Es parecido al rojo, pero a diferencia de éste el valor de G siempre es mayor a B por lo que se requiere que R supere a G por 40 o más y que G supere a B por más de 8. Además, por las características que presenta se pide que el valor de R sea mayor o igual a 85.
- **Amarillo:** Puede ser identificado con facilidad. Se requiere que el R supere al B por 50 o más y que la diferencia entre R y G sea menor a 15.
- **Fucsia**: Los componentes están escalonados, con R siendo el mayor, B siendo el siguiente, con una diferencia de 20 o más y G siendo el menor, con otra diferencia de 20 o más. Además, se requiere que la diferencia entre R y B sea menor a 40 para evitar confusión con el rojo.
- **Celeste**: También se encuentra escalonado, con el componente B superando al G por 20 o más y el G superando al R por 10 o más.
- **Violeta**: Las componentes se encuentran más juntas que en otros colores, pero aún así se encuentran escalonadas, con el B superando al R por 5 o más y el R superando al G por 6 o más. Además, se pide que el valor de B supere 50.
- **Verde**: El verde es muy afectado por los cambios de iluminación, por lo que los requerimientos van cambiando según la intensidad de la componente G. Si esta es menor a 85, basta con que los valores de B y R sean por lo menos 10 unidades menores a G y que la diferencia entre ellos sea menor a 10. Si G es mayor a 85 pero menor a 120 se requiere que B sea mayor a R, ya que se observa que la componente B aumenta más que la R ante una situación de mucha iluminación. Si el valor de G es mayor a 120 la condición adicional es que B sea 10 unidades mayor a R, ya que de caso contrario corresponde simplemente a verde claro.
- **Verde claro**: La condición básica es similar al verde, G tiene que superar a R y B por 10 unidades o más y la diferencia entre ellas debe ser menor a 10. Además, se requiere que R supere a B, que es una condición que no ocurre con el verde.

3.3- Corrección de la lente

Este paso no estuvo considerado en las primeras etapas de desarrollo pero a medida que se avanzaba fue necesario para favorecer la detección posterior y ubicar con mayor precisión espacial a las VDVs. Se investigó cómo llevar a cabo esto y hasta el momento no se encontró una solución genérica que pudiera ser implementada. Por este motivo, se calcularon las matrices tanto de las características de la cámara como los coeficientes de distancia y se almacenaron en un archivo que puede ser reemplazado si se realiza algún cambio ya sea de la cámara utilizada como de la distancia del escenario a la cual se ubica.

A partir de esas dos matrices se utiliza una función de OpenCV para corregir la distorsión producida por la lente antes de procesar cada frame. Puede observarse en la Fig. 4 como el centro de la imagen se mantiene sin alteraciones mientras que los costados son recortados y enderezados como si la imagen fuera aplanada.

(a)



(b)

Figura 4: la imagen superior superior es previa a la corrección de la lente. La imagen inferior cuenta con la corrección realizada.

3.4- Detección de colores

Este proceso se lleva a cabo durante el análisis de cada frame teniendo como fin reducir la cantidad de colores a analizar unificando el RGB de cada pixel al color con el cual lo asociamos semánticamente. Esto quiere decir que si la composición del píxel entra en la gama de rojo, es considerado como rojo y se le asigna el RGB designado (*[255, 0, 0]* para el caso del rojo) para agruparlos.

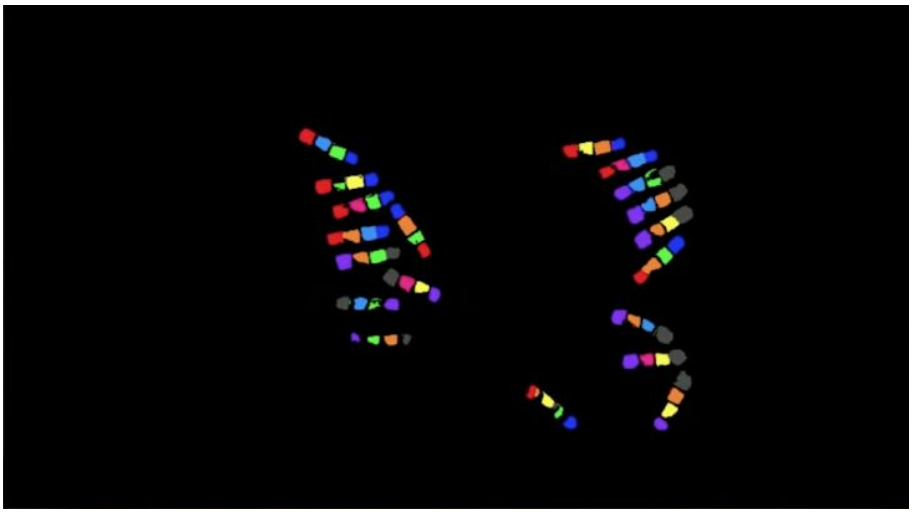
La primera etapa de la detección es el *blureado* o borroneado de la imagen para suavizarla. Para esto se aplica una función de OpenCV con una máscara de 5x5.

Luego se utiliza una función para “extremizar” los colores. Esta función valida cada píxel de la imagen para determinar a cuál de los colores definidos por el sistema pertenece, como se puede observar en la Fig. 5b. En caso de que el píxel no se corresponda con ninguno de los colores definidos para las etiquetas se reemplaza con un píxel negro. Originalmente, esto se lograba restando el fondo a cada frame antes de realizar el análisis, pero este paso se eliminó porque podía obtenerse el mismo resultado en esta etapa.

Una vez extremizados los colores, se realiza una erosión y posterior dilatación para unir posibles componentes que pudiesen quedar aisladas como se puede observar en la Fig. 5c. Durante la erosión se busca reducir o remover irregularidades, es decir, pixeles sueltos que quedaron marcados de algún color y puedan causar ruido. Durante la dilatación, por otro lado, se busca agrandar las componentes que se hayan encontrado de modo de facilitar el trabajo posterior y rellenar cualquier hueco que haya quedado en el medio de una componente.

(a)

(b)



(c)

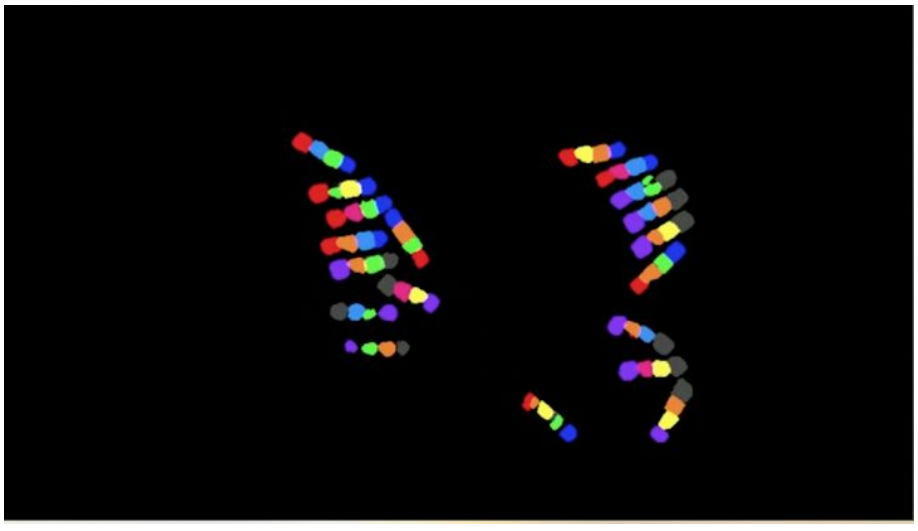


Figura 5: en la imagen (a) se puede observar el frame luego del blureado. En la imagen (b) se observa el mismo frame "extremizado" y posteriormente erosionado. En la imagen (c) se ve el mismo frame luego de la dilatación.

3.5- Detección de componentes

Una componente es un conjunto conexo de puntos de un mismo color. El algoritmo de detección de componentes que se utiliza está basado en un DFS[1] (Depth First Search) donde, una vez detectado el píxel de un color perteneciente al mapa de colores mencionado en la sección 3.4, se revisa en las cuatro direcciones buscando otros píxeles del mismo color. De encontrarse alguno, se guarda esta posición y se repite el proceso teniendo cuidado de no contar ningún píxel más de una vez y parando una vez que se encuentren todos los píxeles que forman la componente conexa. El cálculo del centro se realiza considerando la sumatoria de las posiciones de los píxeles que conforman cada componente y dividiéndola por la cantidad total de píxeles que tiene:

$$(x_c, y_c) = \left(\sum_{k=1}^{n} (x_k, y_k) \right) \frac{1}{n}, \qquad (3)$$

donde *n* es la cantidad total de píxeles del componente. En la Fig. 6 se puede observar los centros dentro de cada una de las componentes detectadas.

Una componente se considera como válida si la cantidad de píxeles que la conforman es superior a una cota parametrizable. El objetivo de este filtrado es evitar el ruido que puede encontrarse en el frame.

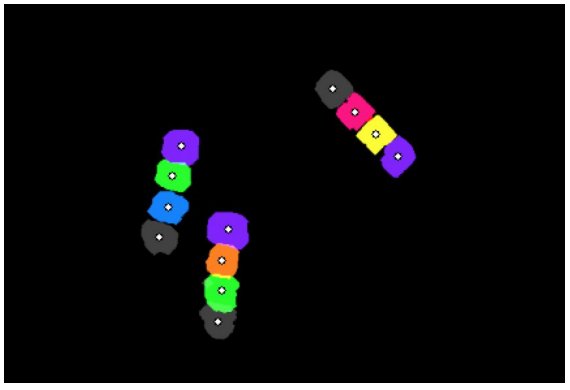


Figura 6: frame procesado luego de ubicar el centro de cada componente

[1]J. L. Gross - Jay Yellen, “Spanning Trees” en Graph Theory and its applications, 2nd ed. Nueva York,
Nueva York, EEUU. Chapman & Hall/CRC, 2006, cap. 4, sec. 4.2.

3.6- Detección

Durante la etapa de detección, se busca establecer las combinaciones de las componentes detectadas en la sección 3.6, que conforman etiquetas válidas. A continuación se enumeran los pasos del algoritmo implementado para hallar estas combinaciones:

1. Se toman las componentes que sean colas o cabezas que como se indicó en la sección de armado de etiquetas están definidas con colores específicos. Entre todas esas componentes se buscan combinaciones de colas y cabezas que se encuentren a una distancia dentro de un umbral parametrizable (Fig. 7a).
2. Para cada uno de estos conjuntos se trazan rectas entre la cola y la cabeza que representan el eje de la posible partícula (Fig. 7b).
3. Por cada uno de estos conjuntos que cumplen esta condición se itera por el resto de las componentes que no son ni colas ni cabezas. Durante ese ciclo sólo se conservan como posibles partes del agente en cuestión aquellas componentes cuyos centros se encuentra a una distancia menor a un umbral parametrizable (distinto al del paso anterior) respecto al eje de la partícula.
4. Una vez analizadas todas las componentes para esa partícula en cuestión pueden darse tres condiciones:
    a. No existe ninguna componente intermedia para esas colas y cabezas establecidas o existe sólo una. En este caso se descarta la posibilidad de que esta combinación pertenezca a la etiqueta válida de un agente autopropulsado.
    b. Se detectaron exactamente dos componentes intermedias. En este caso se suma la distancia entre el centro de las dos componentes intermedias y el eje formado entre la cola y la cabeza y se almacena como posible combinación.
    c. Se detectaron más de dos componentes intermedias válidas. En algunos casos, durante el proceso de dilatación en la detección de colores se generan componentes intermedias que no están presentes en la etiqueta. Las componentes adicionales suelen ser las de menor dimensión, por lo que nos quedamos con las 2 componentes de mayor tamaño y seguimos el proceso descrito en *b* con ellas.
5. Finalmente, si hay agentes que se almacenaron durante el punto 4 se agrega la que tenga menor valor (es decir, que las componentes intermedias están mejor alineadas con la cola y la cabeza) a la colección de partículas correspondientes al frame.

(a) (b)

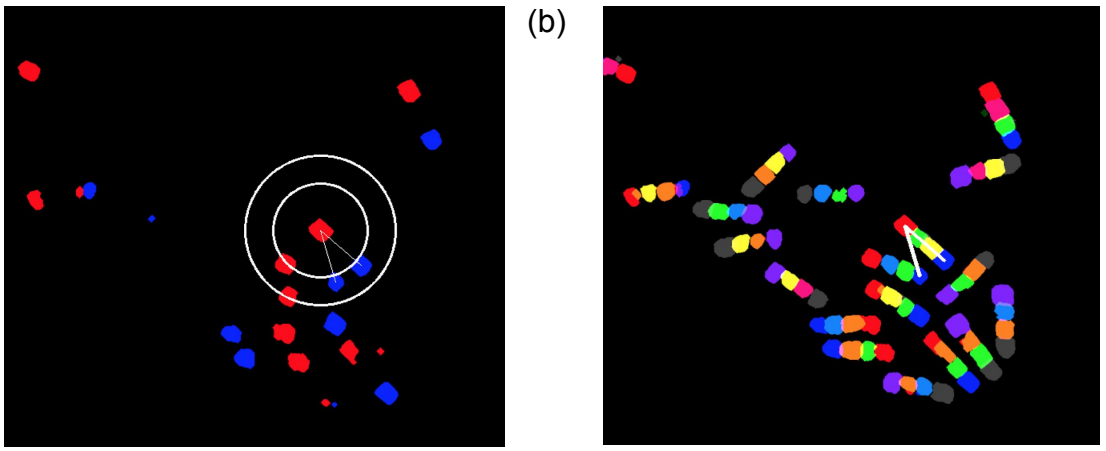


Figura 7: en la imagen (a) se observan las componentes de colas y cabezas identificadas para los colores rojos y azules. Las circunferencias marcan el umbral mínimo y máximo dentro de los cuales tienen que hallarse las colas para la cabeza roja analizada (centro de la circunferencia). Además, se puede contemplar el eje de la partícula uniendo las posibles colas de esa cabeza. En la imagen (b) se puede observar las componentes intermedias detectadas (verde y amarilla) que tienen sus respectivos centros a una distancia menor al umbral establecido de la línea del eje.

3.7- Interpolación y guardado

A medida que se fueron procesando los videos, se observó que varios de los casos en que una partícula no era detectada se debía a la poca definición que tenía la misma en un frame generado en mayor medida debido a problemas con la iluminación. Dado que la cantidad de frames consecutivos en los que estas partículas desaparecen son pocos y que los desplazamientos en esos frames no son considerables, se implementó un algoritmo de interpolación que sirve para obtener las posiciones y orientaciones faltantes. Para esto se considero, en primer lugar, un tipo de interpolación lineal debido a su baja complejidad.

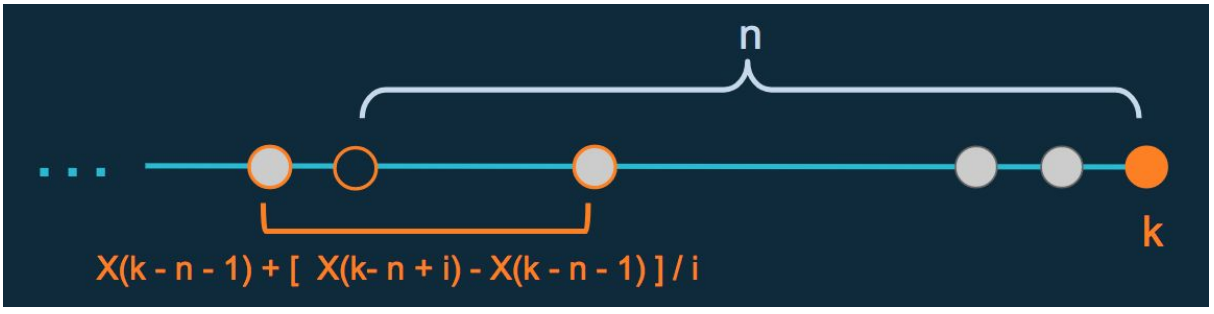


Figura 8: Esquema de condición para la interpolación de dos agentes. La circunferencia naranja rellena de azul representa al agente al que se le quiere hallar la posición con la interpolación . Las circunferencias naranjas rellenas de gris son aquellos agentes que cuentan con posición a partir de las cuales se calcula la posición de la partícula analizada. La circunferencia que se encuentra rellena de color naranja es el último agente que se puede utilizar para respetar la cota de interpolación definida.

Esta interpolación se realiza al momento de guardar los datos en los archivos en caso de que la posición de la partícula a guardar no cuente con su posición definida como se puede observar en la Fig. 8. Al inicio del proceso se revisa que el agente que se intenta interpolar cuente en el frame anterior con un valor definido para su posición. Esa posición anterior se almacena como uno de los puntos conocidos y se procede a buscar el extremo superior. Para esto se recorren los siguientes *n* frames, siendo *n* el mínimo entre la cantidad total de frames restantes del video analizado y el máximo de frames que se desean interpolar (este parámetro se puede modificar desde los parámetros de entrada). Si dentro de ese recorrido por los frames siguientes se encuentra una posición definida para la partícula, se procede a utilizarla junto con la posición del frame anterior al analizado para obtener la posición de la partícula que está siendo interpolada. Esta nueva posición se calcula a partir de:

$$(x_i, y_i) = (x_{i-1} + \frac{(x_{i+k} - x_{i-1})}{k+1}, y_{i-1} + \frac{(y_{i+k} - y_{i-1})}{k+1}), \quad (4)$$

con k < n.

A medida que el video se procesa, las posiciones y orientaciones resultantes se van guardando en diferentes archivos. Se generan un total de cuatro archivos por cada video procesado: *mapping*, *positions*, *orientations* y *interpolations*. El archivo de *mapping* almacena la relación entre el ID de la etiqueta, dado por la combinación única de colores y el número que aparece en el modo de ejecución visual del programa. Además, el número de la fila, *j*, es el número de columna que va a tener designada esa partícula en el resto de los archivos. En el caso de los archivos que almacenan *x* e *y* ese valor se calcula como *2j* y *2j + 1* respectivamente.

El archivo *positions,* así como su nombre lo indica, almacena las posiciones de cada partícula en un frame determinado, donde la información de cada uno de estos frames se encuentra en una fila distinta. Cabe aclarar que para la posición de una partícula se escriben dos columnas en representación de las coordenadas *x* e *y* que se mapean con cada una de las etiquetas como se indicó en el párrafo anterior.

Siguiendo el formato de *positions*, *orientations* almacena las orientaciones de cada partícula. Esta orientación se encuentra expresada por las coordenadas *x* e *y* del vector que la describe.

Finalmente el archivo *interpolations* cuenta con la información sobre las modificaciones que fueron establecidas durante la interpolación pudiendo adoptar para cada partícula los siguientes valores por frame:

- 0 que representa que no hubo necesidad de interpolar dado que la partícula fue detectada por el sistema correctamente.
- 1 se realizó la interpolación y luego de la misma se pudo estimar la posición de la partícula.

- 2 se intentó realizar la interpolación pero aún así no se halló información suficiente para asignarle una posición a la partícula.

3.8- Soporte de Multithreading

La versión inicial del programa utilizaba un único proceso para el procesamiento de los videos ya que estaba pensado para analizar un único video por experimento. Por las condiciones de trabajo y la cámara utilizada para la grabación de los videos, resulta bastante frecuente que se genere más de un video a procesar dado que la cámara fragmentaba el experimento en videos de tiempos más cortos. Este cambio de contexto resultó en la búsqueda de una forma de aprovechar los recursos disponibles para procesar más de un video en simultáneo disminuyendo la cantidad de tiempo de manera significativa.

A nivel implementación se genera un proceso por cada video a analizar permitiendo aprovechar los diversos núcleos de las computadoras modernas. De este modo se puede reducir el tiempo de procesamiento a más de la mitad dependiendo la cantidad de núcleos disponibles.

Una vez finalizados todos los procesos, y en caso de haber sido seleccionada la opción al momento de correr el programa, se pueden unificar estos archivos para obtener un único archivo de salida del experimento compuesto por la combinación de los archivos individuales.

Primeramente se arma un nuevo archivo de mapping dado que en el conjunto original de este tipo de archivos puede suceder que el orden de los IDs dentro de los archivos no sea el mismo y hay que tomar un orden en común. Además, hay que considerar la posibilidad de que los archivos no tengan las mismas etiquetas.

Luego se hace un unificación del resto de los archivos para generar un archivo único de cada uno de los tipos disponibles: *positions*, *orientations* e *interpolations*. Para esta unificación se analiza el nuevo archivo de *mapping* generado en el paso anterior, para hacer coincidir la información en las columnas según las filas de este.

4- Resultados

En esta sección se analizan las distintas pruebas realizadas sobre el programa desarrollado. Se incluyen algunos casos de posibles usos del mismo como es el análisis de errores que se llevó a cabo durante el proceso de desarrollo.

4.1- Análisis de errores

Como se mencionó en la sección 3 se decidió agregar un paso al procesamiento de los videos para intentar solventar los posibles problemas en la detección debido a factores puntuales. A continuación se presenta el análisis de errores realizado sobre dos videos de alrededor de 17 minutos de duración que cuentan con la participación de 19 agentes. Ambos videos fueron procesados con la configuración default del programa. En la Fig. 9, se puede observar la configuración utilizada para todo el análisis de errores. El perímetro que limita el movimiento de las VDVs está realizado con tiras de acetato que le brinda flexibilidad.

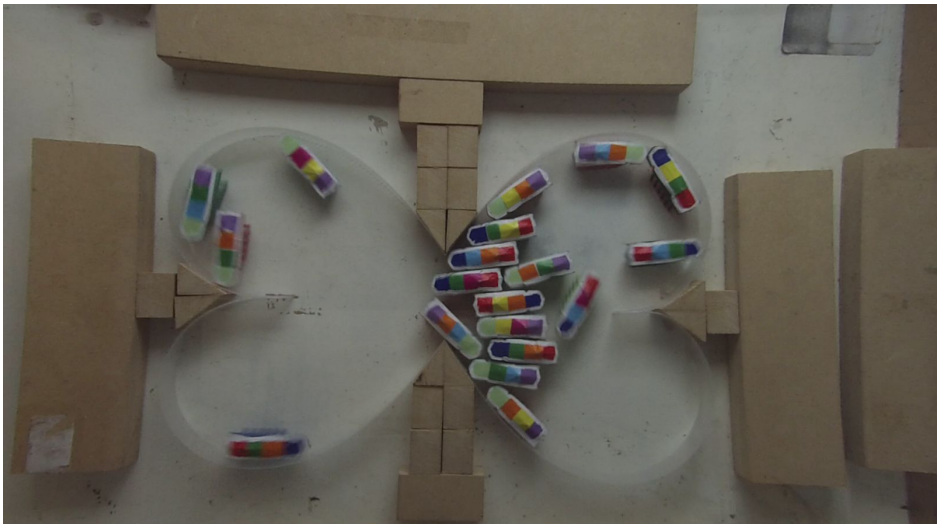


Figura 9: configuración del sistema utilizado para la sección de análisis de resultados.

Una vez implementada la interpolación se necesitaba comprobar cuán positivo fue el impacto generado en la detección de las partículas. Con este fin se analizó cuántos datos se extrajeron de frames interpolados para cada una de las partículas detectadas. Como se puede observar en la Fig. 10a, para todas estas partículas se aplicó interpolación en al menos un frame y en el caso de la partícula con identificación 12 se alcanzó un 7% de frames interpolados. Estos resultados avalan la utilización de la interpolación.

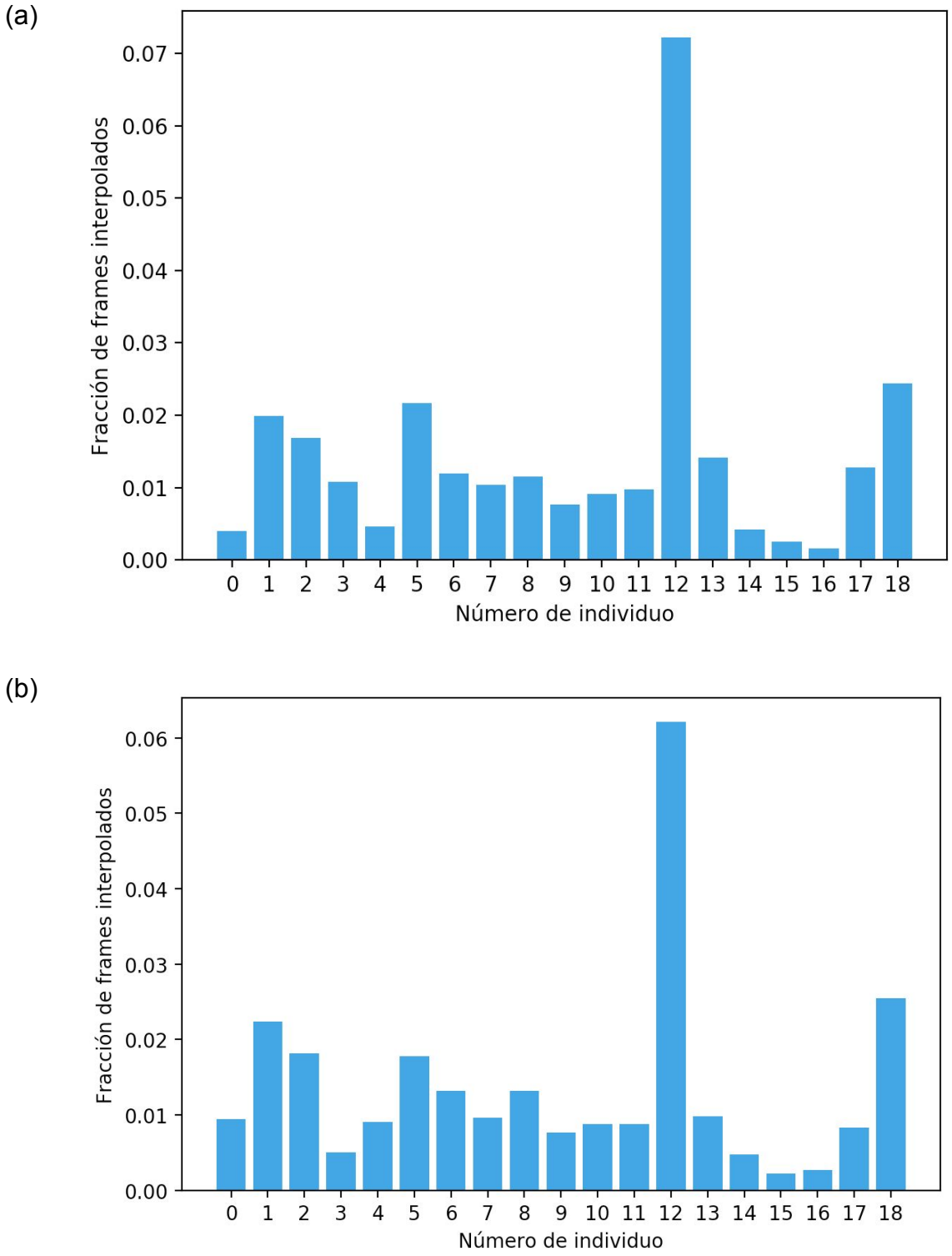
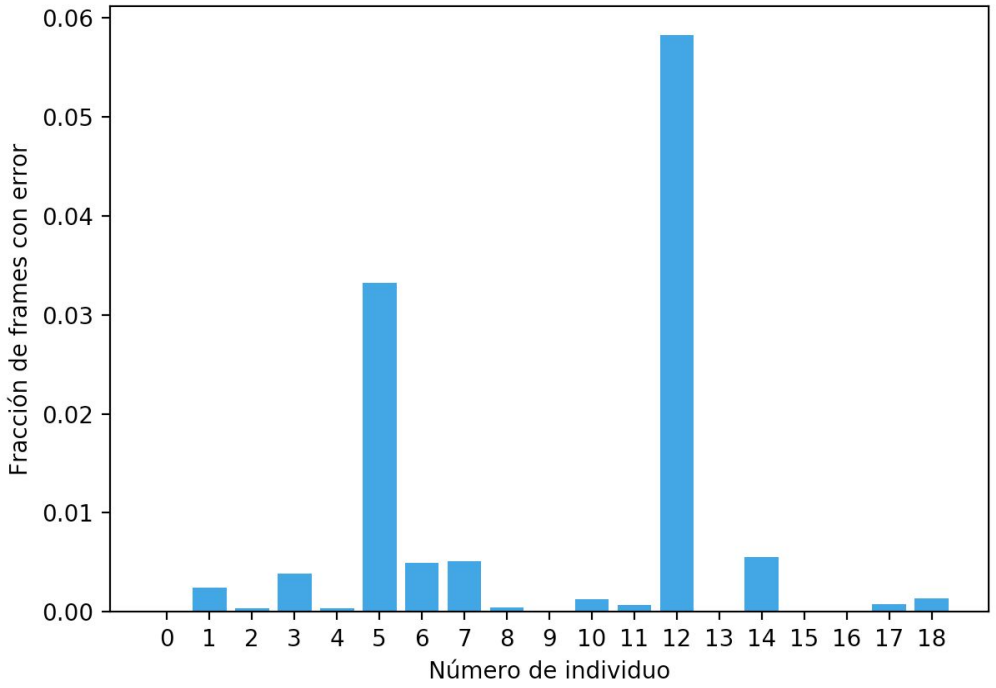


Figura 10: cantidad de frames interpolados durante el procesamiento por cada partícula.

A su vez, la diferencia de porcentaje de interpolación para el agente número 12 incentivó el análisis particular de esa etiqueta con el fin de determinar si esta combinación de colores trae algunos inconvenientes en la detección o se trató de un caso particular. En la Fig. 10b se puede observar el porcentaje de frames

interpolados para los mismos agentes (con la misma combinación de etiquetas) pero en un video diferente. En este caso, también se observó un porcentaje de interpolación similar para la misma combinación de colores concluyendo que la etiqueta correspondiente a este agente, ROJO-FUCSIA-AZUL CLARO-AZUL tiene menor efectividad al momento de la detección.

(a)



(b)

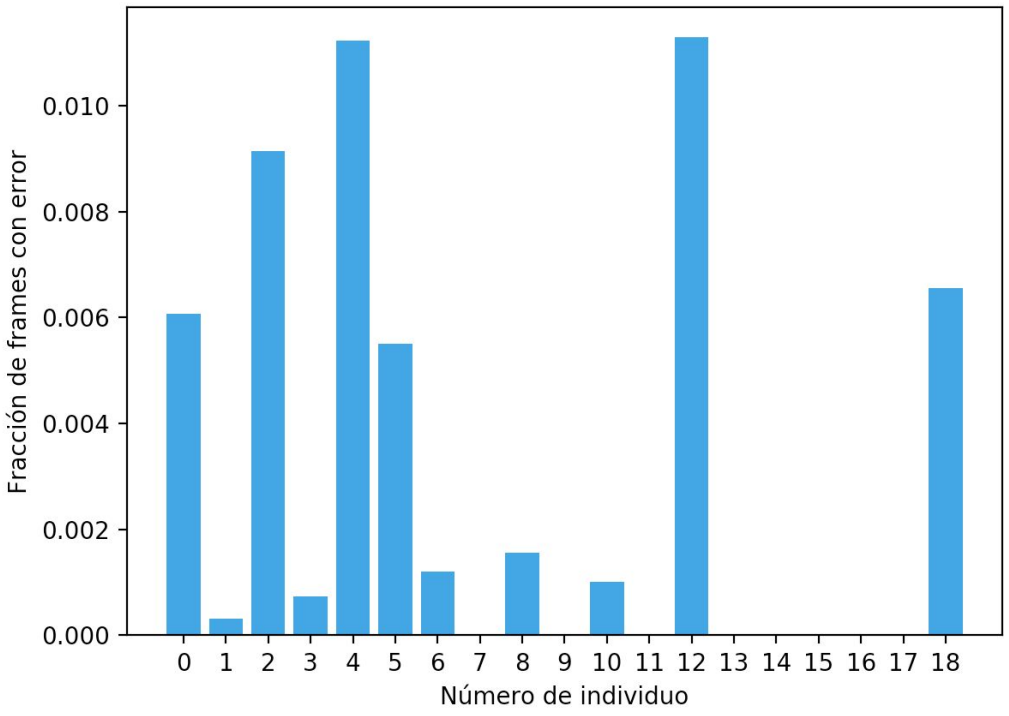


Figura 11: error por individuo calculado en un video que utiliza interpolación de 10.

Teniendo en cuenta cuál fue el aporte de la interpolación se deseaba evaluar cuánto disminuyó el error total luego de aplicarla. Por este motivo se realizó la medición del error por individuo, definiendo este error como el porcentaje de frames durante los cuales cada una de las partículas no pudo ser identificada. En la Fig. 11a se puede apreciar que el error porcentual máximo alcanzado es del 6% y el error promedio es menor al 1% incluso llegando a alcanzar el 0% de error en algunos individuos. Para el segundo video, el error de la interpolación es menor al 1.2%. Esto permite aseverar que el uso de interpolación permite asignar suficiente posiciones para lograr un porcentaje de error aceptable.

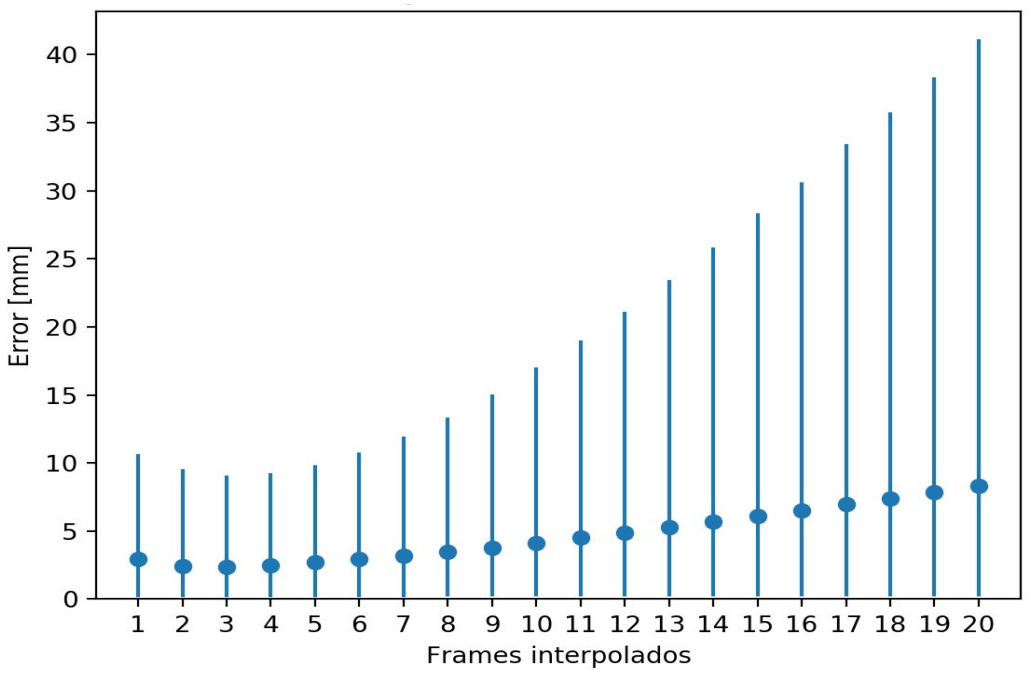


Figura 12: error producido por la interpolación lineal respecto al valor original. Las barras representan los percentiles 2,5 y 97,5.

Finalmente luego de corroborar que efectivamente la interpolación está siendo utilizada y ayuda a la detección de las partículas restaba verificar que los datos que brindaba eran los correctos y no se hallaba introduciendo otro tipo de error al sistema brindando posiciones que distan demasiado de la ubicación real. Para esto se compararon las posiciones detectadas en el video con los datos que se hubieran tenido en caso de que hubiese sido necesario aplicar la interpolación. El valor máximo de frames a interpolar se varió entre 1 y 20 frames que se corresponden con los valores del *eje y* de la Fig. 12. Definimos el error de un agente en un frame como la diferencia en píxeles entre la posición interpolada y la real (no se considera aquellos casos donde se desconoce la posición real del agente):

$$E = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2} \qquad (5)$$

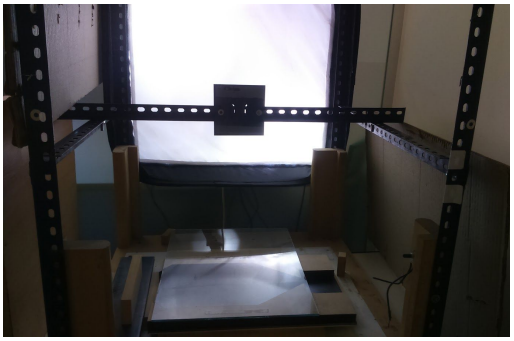
donde (x_i, y_i) indica las coordenadas de la posición interpolada y (x_r, y_r) indica las coordenadas de la posición real. Como se puede observar, la media para todos los valores de distancia estudiados se mantiene por debajo de 20 píxeles, que equivalen aproximadamente a 8 mm, aunque los valores máximos aumentan significativamente al aumentar la distancia en frames entre valores conocidos. Es importante destacar que este gráfico está realizado a partir uno de los vídeos analizados por lo que el resultado no es absoluto sino meramente orientativo.

4.2- Algunos casos de uso

4.2.1- Escenario

Todos los experimentos de esta sección se encuentran realizados dentro del mismo escenario. Éste se encuentra compuesto por diversas piezas de madera que forman el perímetro y cuenta con dos aberturas, una por la que ingresan las partículas y otra por la que salen. La primera se bloquea una vez ingresaron todas para que no puedan salir por la misma (Fig. 13b). Las dimensiones características se encuentran en la Fig. 14.

A su vez, se cuenta con un vidrio que cubre todo el recinto que va a ser evacuado por las partículas. Este elemento tiene como fin evitar que los agentes se den vuelta y se vea modificada la dinámica de los agentes. La desventaja que proporciona es ser fuente de reflejos dependiendo de la posición de la luz y la cámara. Por este motivo, hay una estructura armada (Fig. 13a) donde se ubica la cámara con la que se realiza la captura y una luz blanca potente que se utiliza para iluminar produciendo la menor cantidad de reflejo posible. Así mismo, para favorecer una iluminación pareja los laterales de esta estructura son cubiertos evitando la generación de sombras no deseadas.

(a) 

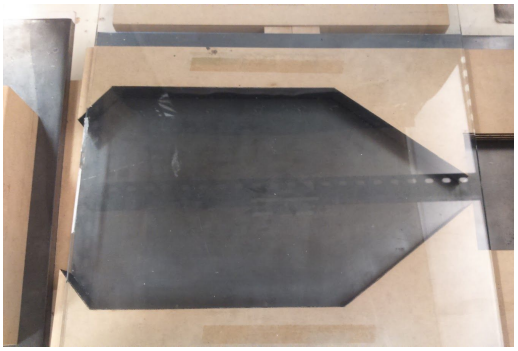
(b) 

Figura 13: en la imagen (a) se puede apreciar la estructura completa utilizada. En la imagen (b) se observa un plano más cercano del contenedor de donde las partículas intentarán escapar.

4.2.2- Características de las pruebas realizadas

Se ejecutaron 30 pruebas que generaron 31 videos a procesar dado que el video de una de las pruebas se encontraba dividido en dos. Estas pruebas contaban todas con 19 agentes. El orden de los videos se encuentran ordenados cronológicamente, es decir, que la carga de la batería fue disminuyendo a lo largo de la secuencia. A partir de los archivos de salida *positions* y *orientations* generados por el software se realizaron los siguientes análisis, considerando que la posición de salida se encuentra en x = 1400 (obtenida de forma experimental) como se puede apreciar en la Fig. 14.

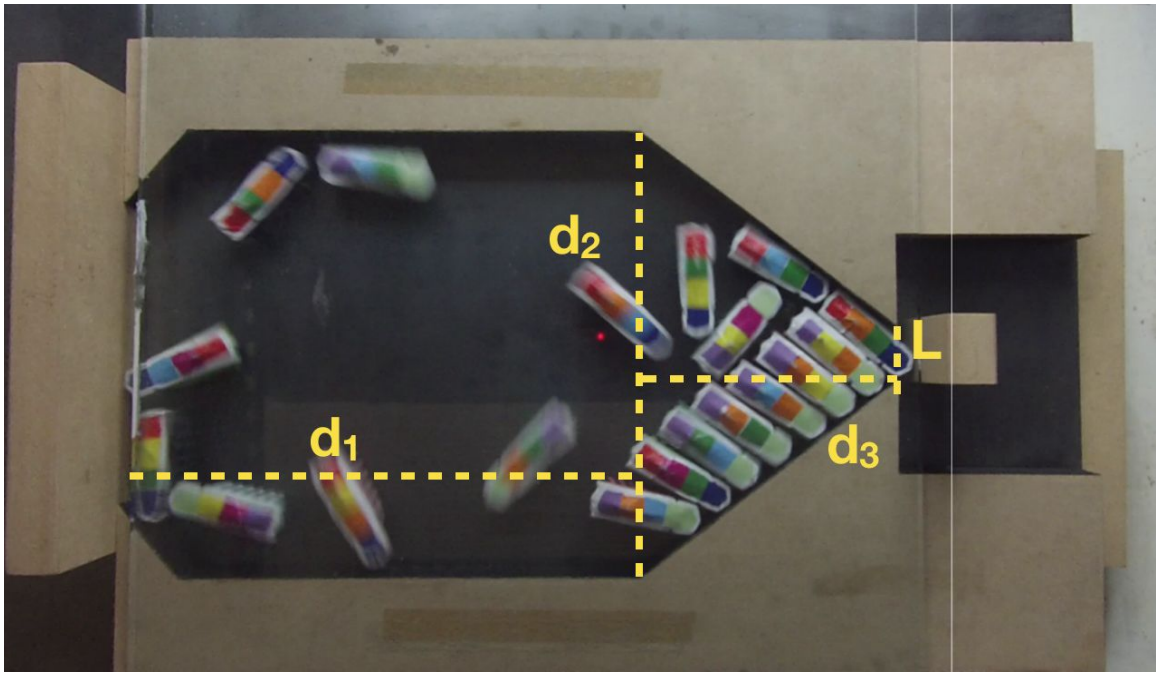


Figura 14: los agentes cuya posición se encuentre a la derecha de la línea vertical blanca se consideran que han salido del recinto.
d_1=200 mm; d_2 = 170mm; d_3=100mm; L=20mm

4.2.3- Tiempo total de escape por video

Este estudio consistió en el análisis de las posiciones resultantes luego del procesamiento para medir el tiempo total que demoró en cada una de las pruebas en escapar del recinto. Para esta medición se consideró el tiempo transcurrido entre que la primera partícula y la última cruzaban la posición de salida. En la Fig. 15 puede observarse que la mayoría de los videos demoraron entre 30 y 50 segundos,

pero en algunos casos llegó a tardar más del doble. Esto se debe a uno o varios individuos que se quedan dando vueltas sin encontrar la salida. Al quedarnos con el tiempo de escape de solo las primeras 15 partículas, como se puede ver en la Fig. 16, los casos con demora se redujeron significativamente. De esta forma se pueden tomar mediciones más relevantes respecto a la cantidad de tiempo medio hasta el escape de los agentes del recinto.

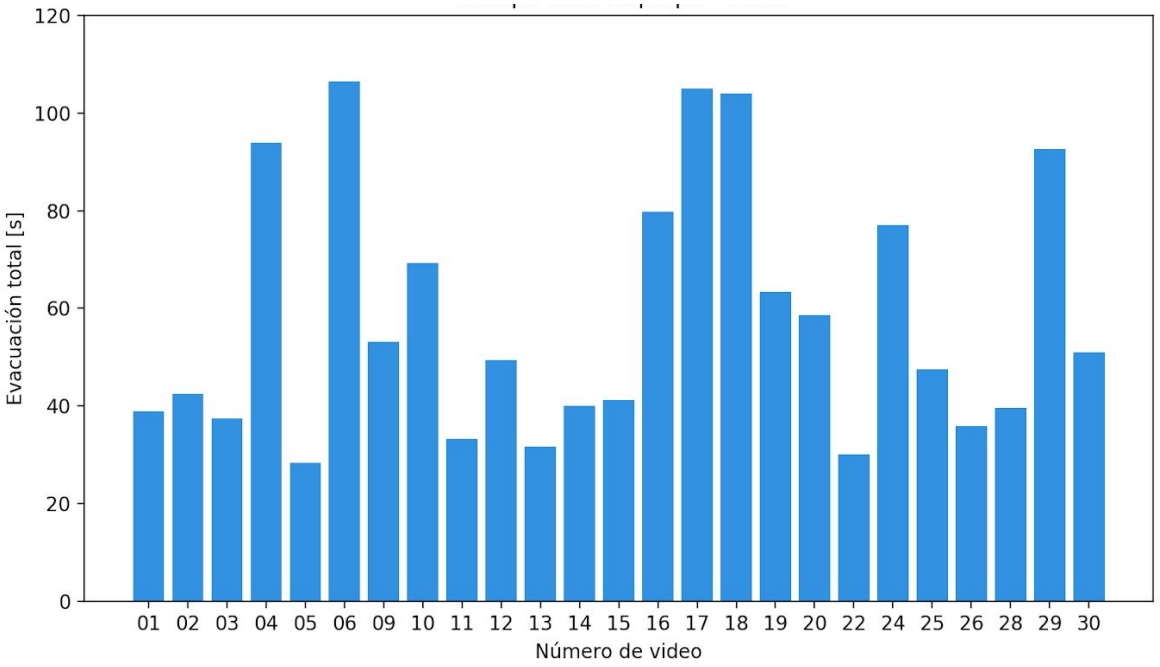


Figura 15: tiempo en segundos de escape de todas las partículas por video

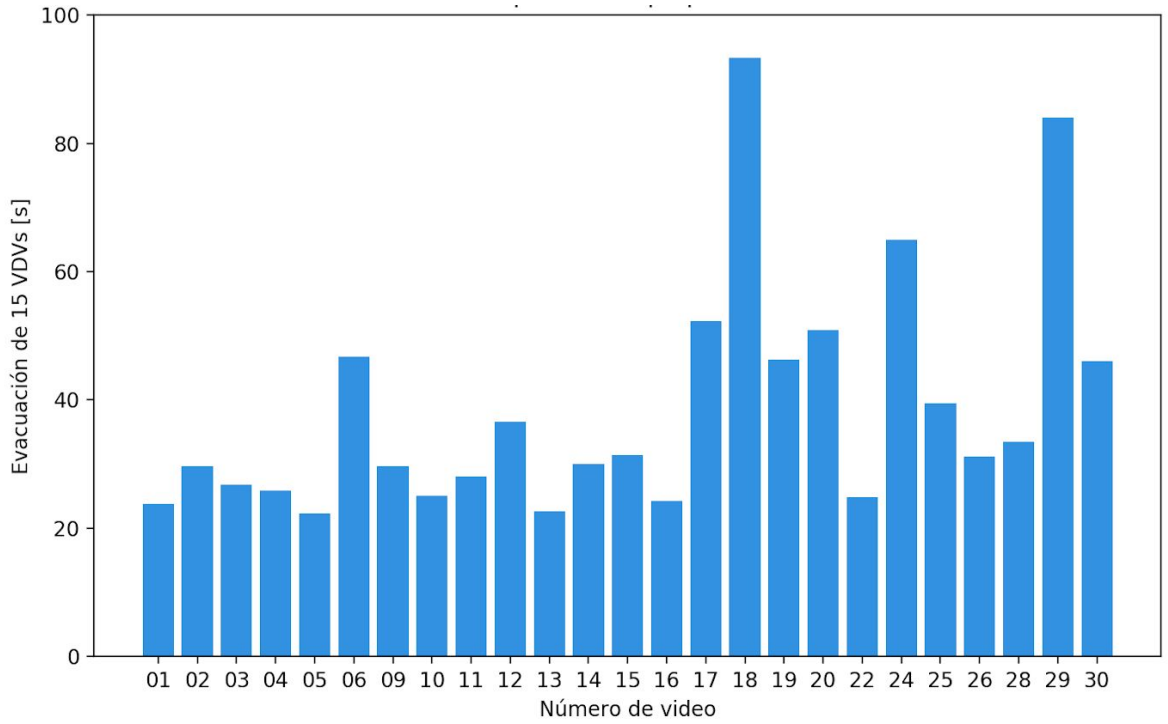


Figura 16: tiempo en segundos de escape de 15 partículas por video

4.2.4- Tiempo de escape por partícula

En esta sección se analizan los tiempos de salida de las partículas, es decir cuánto tiempo tarda una partícula en superar la posición de la salida según el orden en el que salió. Con tal fin se utilizó el archivo de posiciones resultante de cada video y se obtuvieron los tiempos de salida de cada partícula por video. Finalmente se promediaron los tiempos de salida para cada partícula de forma ordenada en los distintos videos, es decir se promediaron los tiempos de todas las que salieron primeras, luego los de las segundas y así sucesivamente con las restantes. En la Fig. 17a y 17b se pueden observar el tiempo de salida de cada agente.

En el caso de la Fig. 17a el tiempo de salida se expresa por cada video mientras que en la Fig. 17b se encuentra promediada entre todos los videos. Esto permite analizar la tendencia a demorar más tiempo en evacuar a medida que quedan menos partículas dentro del recinto. Esto puede apreciarse en los videos, donde se observa cómo aprovechan el espacio para moverse sin la necesidad de buscar salir.

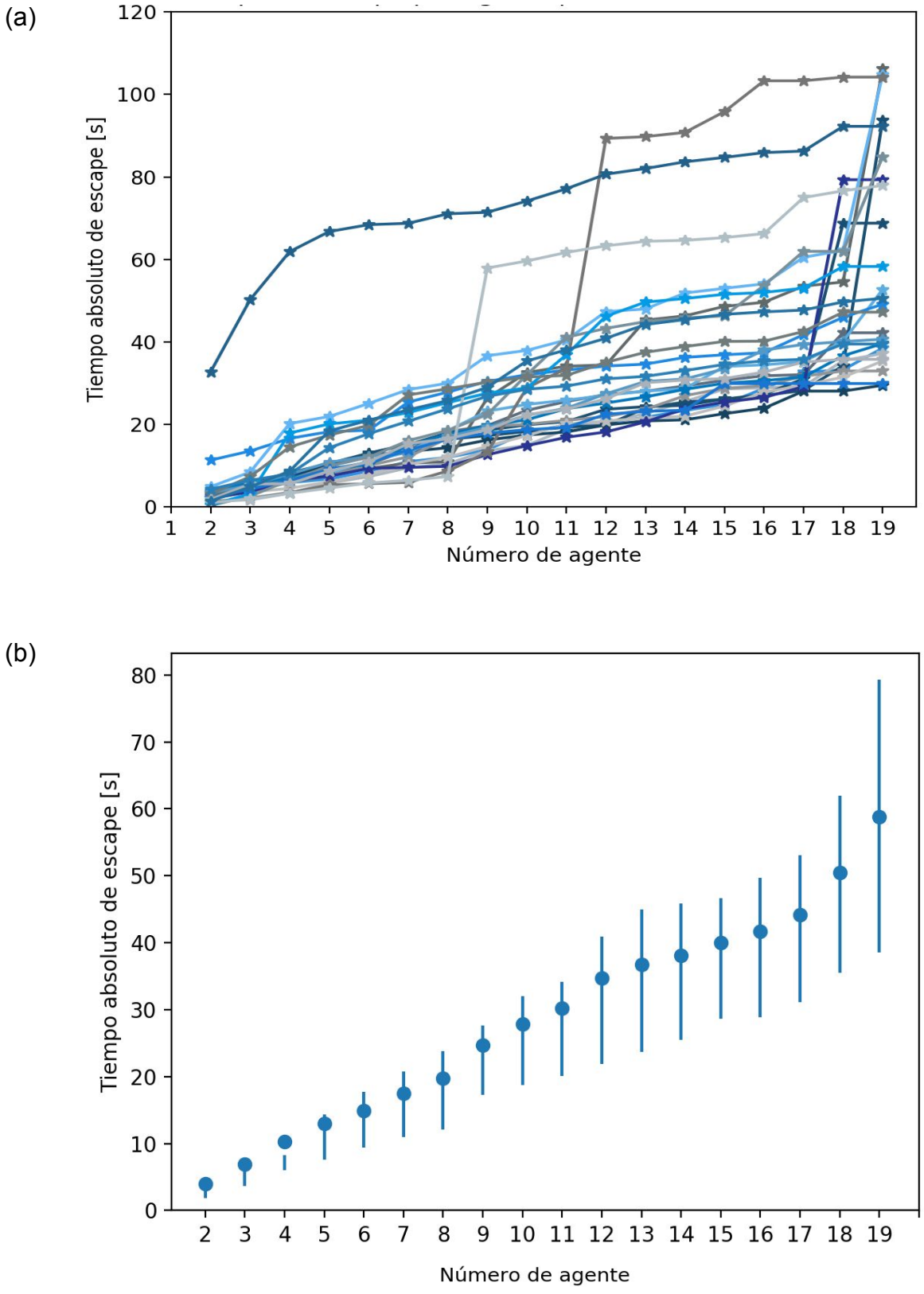


Figura 17: (a) se observan los tiempos de escape de cada uno de los agentes por cada video. En (b) tiempo promedio de escape de cada agente. Las barras representan los percentiles 25 y 75. Se cuenta el tiempo desde el escape de la primer partícula.

4.2.5- Análisis de los agentes rezagados

Finalmente se decidió estudiar la existencia de alguna correlación entre las últimas tres partículas que evacuaban los videos en último lugar. Para esto se realizó un recuento de los IDs de las partículas que se encontraban entre las últimas 3 que abandonaron el recinto en último lugar para cada uno de los videos utilizando los archivos de *mappings* y *positions*.

ID - etiqueta	Apariciones
Azul-Amarillo-Verde-Rojo	10
Verde Claro-Fucsia-Amarillo-Violeta	9
Azul-Verde-Fucsia-Rojo	8
Verde Claro-Amarillo-Fucsia-Violeta	8

Tabla 3: IDs de las etiquetas con más apariciones en las últimas 3 posiciones en evacuar el recinto.

Aparecieron 4 etiquetas cuyas apariciones cubrieron una cota superior al 46% del total. En la *tabla 3* se pueden observar sus etiquetas y la cantidad de apariciones que tuvieron entre este grupo rezagado. La etiqueta AZUL-AMARILLO-VERDE-ROJO es la que se ubica en la posición número uno con diez apariciones entre las últimas 3 posiciones de las cuales tres de ellas coinciden con videos de duración por encima de la media (videos 18, 24 y 29 de la Fig. 12), lo que podría estar indicando un comportamiento diferente. Haciendo un análisis de esos videos, se puede observar como ese agente en particular está bastante más tiempo dando vueltas consumiendo más tiempo.

5- Conclusiones

A partir del desarrollo del trabajo y basándonos en los resultados obtenidos se puede concluir que la implementación del programa fue satisfactoria ya que cumple con los requerimientos funcionales inicialmente planteados. Esto implica la detección unívoca de cada agente participante de un experimento que es exportado a un video a procesar, teniendo como salida del procesamiento un archivo de posiciones y otro de orientaciones. Así mismo se fueron realizando algunas características adicionales para complementar y facilitar el uso del sistema, como el procesamiento en paralelo.

Esta solución fue probada en los experimentos expuestos en la sección de resultados donde se utilizó para determinar tiempos de salida de los agentes, estudiar las estadísticas de los tiempos de salida y analizar la existencia de patrones sistemáticos producidos por algunos de los VDVs.

Si bien la solución propuesta y la implementación posterior al problema planteado cumple los requerimientos, el sistema presenta mayormente dos desventajas:

- Las etiquetas se encuentran generadas manualmente por lo que se requiere la confección de las mismas para cada agente. Así mismo, los materiales utilizados tienen como limitante la diversidad de colores que presentan debido a la necesidad de que los mismos sean claramente distinguibles.
- La solución se encuentra diseñada de forma personalizada para el modelo de agentes en cuestión, HEXBUG nano. Esto genera una dependencia en los tamaños considerados para validar las componentes, umbrales para la aceptación de pares cabeza-cola y visibilidad de la etiqueta.

5.1- Posibles mejoras

De todos modos se considera que este programa puede continuar expandiéndose mediante la prueba e implementación futura de algunas mejoras que se enumeran a continuación:

- Mejorar la detección de colores a partir del uso de machine learning. Se nos ocurren dos opciones: por un lado, podríamos entrenar una red neuronal estándar para que dado el RGB de un píxel lo clasifique entre los colores posibles. Otra opción es usar un algoritmo de K-medias. Como sabemos el total de colores, podemos usarlo para el valor de K (es decir, la cantidad de grupos distintos a ser detectados) y de esa forma agrupar los colores según los valores de sus componentes RGB.

- Inferir la posición de una partícula aunque sólo fuesen detectadas 3 de las 4 componentes de la etiqueta. Para esto habría que agregar la lógica necesaria para, en caso de contar con un caso de estas características validar si esa combinación ya ha aparecido en algún otro frame y cuál es el color que le estaría faltando para obtener la etiqueta completa.