Universidad Zaragoza
1542

Trabajo Fin de Grado

Simulación dinámica de un rebaño mediante técnicas de visión y realidad aumentada

Dynamic simulation of a herd using vision and augmented reality techniques

Autor/es

Adrián García Remiro

Director/es

Rosario Aragüés Muñoz
Alejandro Pérez Yus

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2020/2021

Simulación dinámica de un rebaño mediante técnicas de visión y realidad aumentada

RESUMEN

La automatización y robotización de procesos agrarios y ganaderos es un campo de investigación de gran interés en la actualidad. Por ejemplo, la compañía tecnológica Rocos está explorando en la actualidad el uso del robot Spot de Boston Dynamics para llevar a cabo tareas de pastoreo autónomo. Previa implantación real, es importante contar con un entorno en simulación en el que validar las estrategias de pastoreo propuestas. Es importante asimismo que estos entornos de simulación sean intuitivos y los resultados sean fácilmente interpretables.

En este Trabajo de Fin de Grado, se va a construir un entorno en el que se va a representar de una manera más visual, mediante realidad aumentada, un rebaño de ovejas moviéndose libremente y actuando en consecuencia con la presencia de un perro pastor.

Para conseguir este propósito, se han adquirido conocimientos y aplicado técnicas de realidad aumentada y visión por computador, incluyendo el uso de marcadores ArUco para identificar el plano sobre el que se proyectan las ovejas y los robots de pastoreo.

Para simular el comportamiento de las ovejas, se han desarrollado modelos individuales de movimiento, y se han implementado comportamientos colectivos de tipo swarm, basados en términos atractivos y repulsivos entre las ovejas y los robots.

Finalmente, se ha realizado una serie de pruebas y experimentos, dando diferentes valores a los parámetros, de manera que se pueda comprobar el correcto funcionamiento.

Las simulaciones llevadas a cabo, se estudia el comportamiento de un rebaño de ovejas en el que hay incorporado un perro. De esta forma las ovejas se han de

mover libremente siguiendo unas pautas según las cuales se mantengan cerca entre ellas, y, además, si el perro se les acerca, reaccionarán llevando a cabo un movimiento de repulsión.

En este trabajo se ha utilizado el entorno de desarrollo integrado (IDE) Spyder, el lenguaje de programación Python y la librería de visión por computador OpenCV.

Índice de figuras

Índice general

1. Introducción

En este capítulo se presentan de forma detallada los principales aspectos relacionados con el trabajo. Para ello habrá un apartado de motivación que contribuirá a conocer el tema de este trabajo, un apartado en el que se detalla dónde se ha realizado el proyecto, uno más detallando los objetivos principales, y otro que detalla brevemente las herramientas y entorno con el que se ha trabajado Finalmente se refleja el contenido de este documento.

1.1. Motivación

Sabemos que en la actualidad el mundo de la robótica está en auge, que se utiliza para infinidad de cosas con las que, gracias a un simple robot, podemos facilitar de manera increíble el trabajo de un operario en una empresa. Es por eso, que, en relación con este trabajo ha aparecido últimamente un robot que hace de perro pastor ante un rebaño, programado por la empresa norteamericana Rocos. Esta empresa, centrada en tareas de agricultura y ganadería, pretende demostrar que los robots pueden ayudar a obtener información más precisa en tiempo real, así como añadir mayor automatización a procesos dentro de este ámbito, como es el pastoreo. Aún es pronto para equiparar a un perro tradicional en cuanto a prestaciones, sin embargo, el rápido avance en este campo ha demostrado que podría estar más cerca de lo que nosotros pensamos.

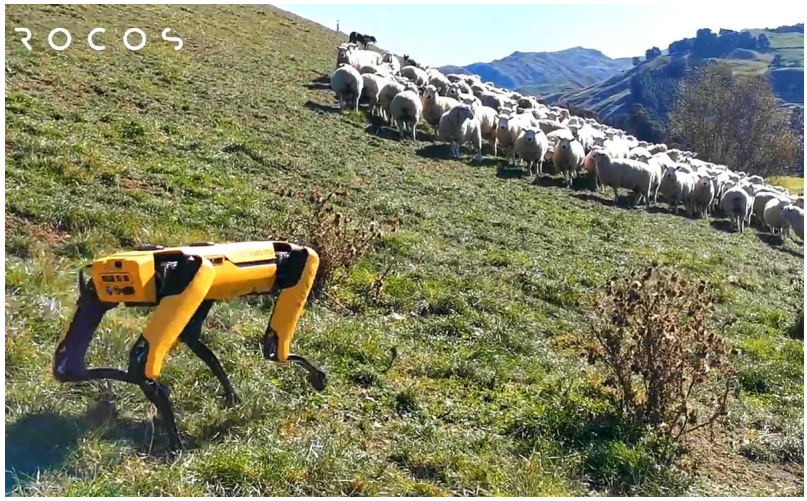

Figura 1.1: Robot pastor [12]

Lo que se pretende al realizar este proyecto es representar el funcionamiento de estos sistemas de pastoreo de una forma más gráfica, de manera que aquellas personas que no conozcan en profundidad estos sistemas capten la idea que queremos presentar de una forma más intuitiva y representativa. Se ha decidido simular por computador el comportamiento de este perro, ya que la simulación es un método de comprobación más rápido. Además, también nos permite un ahorro de costes frente a la experimentación con el robot directamente en el campo, contribuyendo así mismo a obtener medidas de forma más automatizada. También, es muy importante tener una interfaz lo suficientemente clara, intuitiva y representativa para el usuario, de tal modo que los resultados sean fácilmente interpretables.

Para ello, lo que se va a tratar es la representación mediante realidad aumentada de un rebaño de ovejas, entre las cuales se incorporará un perro pastor. Este rebaño estará representado en 3D sobre un plano que estará predeterminado con la ayuda de unos marcadores detectados por la cámara.

Posteriormente, se realizará una demostración en la cual se hará uso de un robot que hará de perro pastor. Este robot tendrá en su parte superior un marcador ArUco incorporado, de manera que nos facilite el reconocimiento del robot y poder hacer las simulaciones que sean necesarias. [12]

1.2. Contexto de realización

El presente proyecto se ha realizado dentro del departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza. Además, se ha tenido el apoyo de investigadores del Instituto de Ingeniería e Investigación de Aragón (I3A) que presentan una amplia experiencia en este ámbito.

Este proyecto pretende servir de ayuda para aquellas personas que estén interesadas en el mundo de la robótica, así como aquellas que quieran aprender a proyectar figuras en realidad aumentada con la ayuda de un simple ordenador y una cámara.

1.3. Objetivos y tareas

El objetivo principal de este trabajo es el desarrollo de una aplicación que nos permita representar en realidad aumentada un rebaño de ovejas simulado, y que a su vez obedecen a un comportamiento en grupo el cual está relacionado con la separación de las ovejas y una fuerza de repulsión ante un perro pastor.
Seguido a esto, le siguen los siguientes sub-objetivos:

- Analizar las distintas matrices de transformación que existen entre los dos marcadores ArUco, y la webcam.
- Representar un objeto sobre el plano real, además de incorporar un control de movimiento en "tiempo real". Se ha representado cubos 3D, casitas, y posteriormente ovejas y un perro.
- Movimiento de comportamiento en grupo, de manera que las ovejas correspondientes al rebaño tengan coherencia entre ellas y se muevan en grupo, sin dejar de tener un movimiento independiente cada una. También deberán reaccionar ante la presencia de un perro pastor.
- Desarrollo de una serie de pruebas y experimentos garantizando así un correcto funcionamiento.
- Elaboración de la documentación.

1.4. Entorno de trabajo

El trabajo desarrollado consiste en la simulación de un rebaño de ovejas representado utilizando técnicas avanzadas de visión por computador y realidad aumentada. Para ello, ha sido necesario un estudio previo de los conceptos básicos de realidad aumentada. Además, se ha implementado un perro pastor, de tal forma que las ovejas tienen que reaccionar ante él, tal y como sucedería en un rebaño real.
Para la realización de este trabajo, se ha hecho uso de un ordenador portátil mediante el cual se ha podido desarrollar el software necesario para poder alcanzar los objetivos propuestos. También se ha utilizado una webcam externa que ha permitido tener una visión del plano donde se ha trabajado, así como el reconocimiento de los diferentes marcadores donde proyectar las ovejas y el perro.

Figura 1.2: Descripción plano de trabajo

Todo el trabajo se ha escrito en Python, a través del entorno de desarrollo integrado (IDE) Spyder. He utilizado este porque es de código abierto y está escrito en Python. [2]

También se ha usado Open CV, una librería software de visión por computador y machine learning. Es la librería que se ha utilizado para la realización de este proyecto debido a la cantidad de funciones que posee para la detección de ArUcos y la consiguiente representación de elementos.

Estas funciones nos han servido de ayuda para la inicial calibración de la cámara, para detectar los distintos marcadores ArUcos visibles en la escena real, dibujar elementos sobre la imagen, etc.

Además, la propia página web de OpenCV contiene una sección tutorial, en la cual podemos encontrar todas las funciones que tiene, así como la descripción de cada una de ellas, permitiendo así entender con mayor profundidad lo que se está realizando. [5]

1.5. Contenido de la memoria

Este documento está organizado de la siguiente manera:

- Capítulo 1: El capítulo en el que nos encontramos. Aquí es donde se introduce el proyecto que se ha realizado.
- Capítulo 2: Se describe de forma detallada las herramientas teóricas que han sido útiles para poder llevar a cabo el trabajo.
- Capítulo 3: Se explica las técnicas de implementación que se han llevado a cabo para conseguir representar objetos virtuales en el plano real, así como la solución a algunos problemas que han surgido.
- Capítulo 4: Aquí se puede encontrar el análisis que se ha llevado a cabo para poder asignar el movimiento adecuado a los elementos que se han representado. También se detalla el modelo de comportamiento el grupo que se ha desarrollado para un correcto funcionamiento.
- Capítulo 5: Se puede observar los diferentes experimentos que se han llevado a cabo para demostrar que el trabajo ha sido realizado exitosamente.
- Capítulo 6: Están desarrolladas las conclusiones que se han obtenido tras el estudio de este proyecto, así como las posibilidades que pueden surgir de cara a futuros proyectos a partir de éste.

2. Aproximación a la Realidad Aumentada y a la Visión por Computador

2.1. Realidad aumentada

La realidad aumentada (RA) consiste en la proyección de elementos virtuales sobre el mundo real que nosotros visualizamos.
Ésta se reproduce a través de un dispositivo que contenga una cámara y además ha de disponer de un software para poder representar el elemento que queremos sobre la escena real.
Los diferentes elementos que se pueden proyectar pueden ser imágenes, vídeos, elementos descriptivos sobre un objeto en concreto, archivos de audio, etc.
Para poder realizar un proyecto basado en realidad aumentada, es necesario una serie de elementos:

- *Dispositivo con cámara:*
  - *Ordenador de sobremesa con webcam*
  - *Ordenador portátil con webcam*
  - *Tablet*
  - *Smartphone*
  - *Wearable con cámara (relojes, gafas, etc.)*
- *Software: encargado de hacer las transformaciones necesarias para facilitar la información adicional.*
- *Disparador, conocido también como "trigger" o activador de la información:*
  - *Imagen*
  - *Entorno físico (paisaje, espacio urbano, medio observado)*
  - *Marcador*
  - *Objeto*
  - *Código QR [1]*

La realidad aumentada puede tener aplicaciones no sólo en el ámbito del ocio, sino también para ayudarnos en ciertas tareas. Por ejemplo, existen aplicaciones dedicadas a la decoración del hogar, en las cuales es posible visualizar previamente a la compra cómo quedará un mueble en una estancia de una casa. [4]
También se aplica en el ámbito educativo y en la sanidad, donde se dispone de realidad aumentada para poder visualizar sobre un cuerpo humano, los músculos o huesos que lo componen.
Otro ejemplo de la utilidad de la realidad aumentada podría ser en la fontanería, pudiendo ver a través de un dispositivo móvil, por donde van las tuberías en la pared sin necesidad de romperla.

Existen varios niveles que pueden definir la realidad aumentada, según [1]. Los distintos niveles son definidos según el grado de complejidad que presentan las aplicaciones según las tecnologías que implementan. La clasificación quedaría como sigue:

- ***Nivel 0*** (enlazado con el mundo físico). En este nivel, se representan los elementos mediante códigos de barras. En este nivel no se pueden representar elementos muy completos, siendo esta categoría la más frecuente, a día de hoy, en el mundo que nos rodea.

- ***Nivel 1*** (RA con marcadores). En este nivel intervienen los marcadores ArUco y QR, facilitando la opción de poder representar como máximo objetos en 3D.

- ***Nivel 2*** (RA sin marcadores). En este nivel la complejidad aumenta en relación con el anterior, de forma que no es necesario un marcador para poder representar un elemento. La posición y orientación de la cámara con respecto de la escena se puede obtener en este caso aplicando métodos de SLAM u odometría visual con la propia cámara, o ayudándose de otros sensores como una IMU (Inertial Measurement Unit), brújula, o GPS.

- ***Nivel 3*** (Visión aumentada). Este sería el nivel máximo. Existen unos dispositivos como las "Google Glass" que disponen de una alta tecnología que nos permite representar cualquier cosa sin necesidad de otros elementos. También existen las Hololens de Microsoft, las cuales llevan incorporados muchos más sensores, como puede ser el de profundidad que nos ayuda a mapear todo el entorno. Estos dispositivos permiten superponer las inserciones de realidad aumentada directamente en la visión humana a través de un cristal transparente que permite visualizaciones de tipo holograma, en vez de verse a través de una pantalla como ocurría en niveles inferiores.

Figura 2.1: Hololens Microsoft [7]

En este caso, se ha empleado un ordenador portátil con webcam incorporada como elemento externo, un software de programación basado en el lenguaje Python y una serie de marcadores ArUco.

2.1.1. Marcadores

Un marcador es una imagen o pegatina fija que nos ayuda a conocer la posición y orientación de un objeto o de un plano donde queremos proyectar. Hay una serie de procesos para reconocer cada marcador debido a que cada uno es único y exclusivo. Por ello, hay unos diccionarios predefinidos donde se almacenan cada uno de los diferentes marcadores. Se podría decir que es un activador que, al detectarlo mediante la cámara, representa la información que queremos. Entre ellos están:

- **Códigos QR:** tienen una forma cuadrada en la que, en su interior, hay una serie de puntos o "pixeles" dibujados. Se representa de forma que, dependiendo de la distribución de estos puntos, cada QR hace que sea único. Mediante su lectura, se pueden representar infinidad de elementos. Por ejemplo, hoy en día los podemos ver para descargar una app de la tienda Store de nuestro dispositivo móvil, también los podemos encontrar recientemente debido a la pandemia del Covid-19 en las mesas de los restaurantes, de forma que, al enfocar con nuestra cámara del dispositivo móvil, nos redirige a una página web en la que podemos ver la carta de ese restaurante. También los podemos encontrar en museos, ofreciéndonos una información detallada del cuadro o escultura que queremos ver.
  Su apariencia es la siguiente:

Figura 2.2: Ejemplo código QR [1]

- **ArUco:** a diferencia de los marcadores QR, estos pueden considerarse más simples. Se basan en unos cuadrados también, los cuales están divididos en unas filas y unas columnas. Existen de 4x4, 5x5, 6x6, etc. De esta forma, queda dividido en una forma similar a la de un tablero de ajedrez, de tal forma que, si unos recuadros están en negro y otros en blanco, representan una figura única para que pueda ser detectada mediante nuestro software. Con un marcador, no solamente podemos reconocer la posición del ArUco, sino que podemos obtener la posición de cualquier elemento de la escena con respecto al marcador. [1]

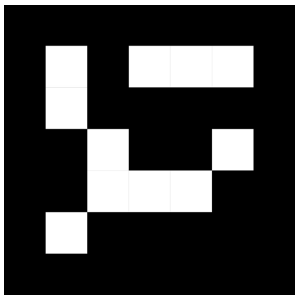

Figura 2.3: Ejemplo de marcador ArUco [8]

2.2. Visión por computador

El término de visión por computador consiste básicamente en la obtención de las propiedades de un mundo tridimensional, a partir de una serie de imágenes bidimensionales de ese mundo.

Hay muchas tecnologías que utilizan actualmente la visión por computador, con múltiples utilidades: para el reconocimiento de objetos, restauración de imágenes, reconstrucción de una escena, detección de sucesos, etc.

No sólo la podemos encontrar en el mundo de la robótica, sino en muchos otros campos, por ejemplo, en el mundo sanitario es utilizada ya para diagnosticar enfermedades de forma autorizada a través de una imagen bidimensional. También se emplea en el fútbol, para el consiguiente seguimiento de un jugador por la cámara.

Asimismo, se usa en el ámbito de la industria para la detección de una anomalía en el proceso de fabricación de una pieza.

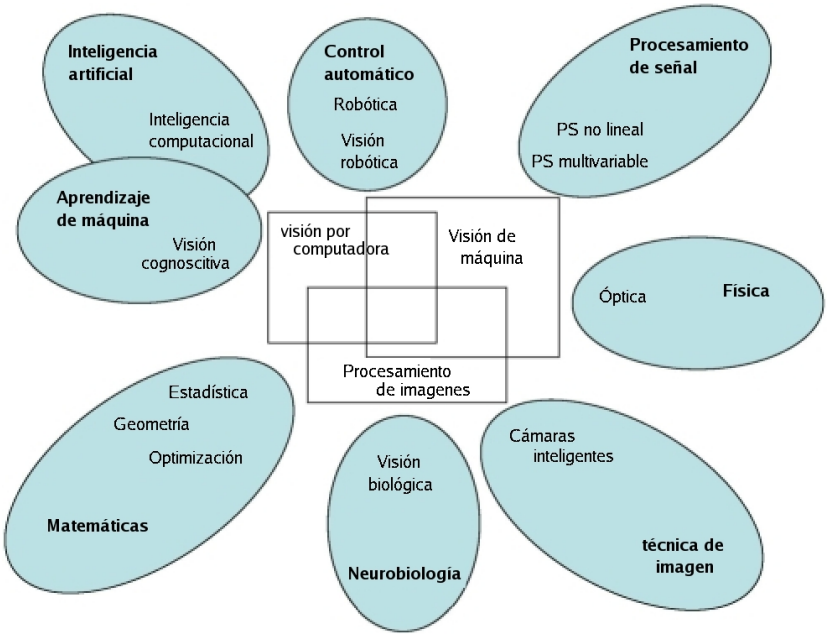

Figura 2.4: Relaciones entre la visión por ordenador y otras áreas afines [9]

Con la ayuda de la visión por computador podemos trabajar en todas las ramas que figuran en la imagen, procesar señales, automatizar robots, programas matemáticos y físicos, inteligencia artificial, etc.

Se ha utilizado la visión por computador principalmente para poder reconocer los marcadores ArUco. De esta manera resulta más sencillo, aunque existen otros métodos para reconocer la escena sin necesidad de marcadores (por ejemplo, usar un método de SLAM), pero estos presentan ciertas desventajas respecto al método empleado, como puede ser un mayor coste computacional o errores acumulados por la deriva. Es cierto que con estos métodos no haría falta tener que visualizar constantemente el marcador ArUco para conocer la posición relativa de la cámara a la escena, pero usando marcadores, aparte de hacer este problema mucho más sencillo, permite conocer también la posición de otros elementos situados en la escena (por ejemplo, los perros pastores).

3. Representación de elementos

En este capítulo se va a detallar de una forma más precisa el procedimiento que se ha seguido para la correcta proyección de un elemento sobre el plano referenciado por un marcador Aruco. Vamos a asumir que todos los elementos se mueven en un plano, por eso utilizamos los marcadores que nos permiten obtener tanto la posición y orientación de los elementos, así como el origen del plano respecto a la cámara. Para ello, es necesario una previa calibración de la cámara, así como tener en cuenta la posibilidad de que aparezcan oclusiones.

3.1. Calibración de la cámara

Para la realización de este trabajo, he tenido que hacer uso de una cámara externa para poder enfocar el plano de trabajo, ya que desde la propia cámara incorporada en el ordenador portátil no hubiera podido conseguir el enfoque del plano donde va a ser proyectado el rebaño con una movilidad y amplitud adecuada. En este trabajo se ha usado una cámara web modelo Logitech C920 Pro HD.

Figura 3.1: Cámara Logitech C920 Pro HD [10]

La cámara presenta unos parámetros importantes que tendrán que ser calculados, como la distancia focal y los centros ópticos, resumiéndose estos dos parámetros en una matriz llamada matriz de calibración de la cámara. Sin conocer la distancia focal y los centros ópticos, sería imposible representar sobre la imagen, ya que son los fundamentales para poder pasar de puntos 3D a píxeles de la imagen o viceversa.

Por otro lado, las cámaras de hoy en día presentan mucha distorsión en las imágenes. Entre ellas se encuentran la distorsión radial y la distorsión tangencial. Para poder realizar una correcta detección de los marcadores que iba a utilizar, debía calibrar la cámara de manera que se resolvieran al máximo estas distorsiones.

La distorsión radial hará que nuestras líneas que son rectas aparezcan curvadas, siendo mayor el efecto conforme nos alejamos del centro de la imagen.

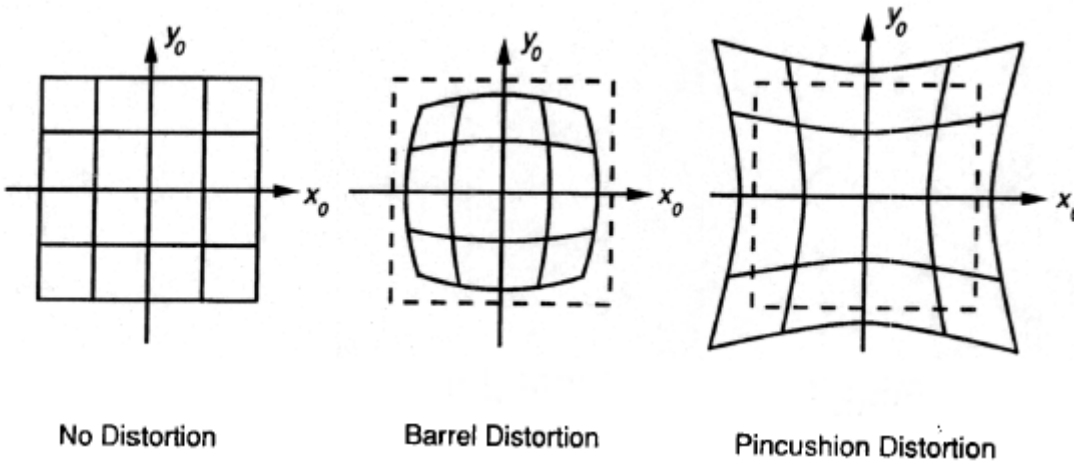

Figura 3.2: Distorsión radial de una imagen [11]

Por otro lado, la distorsión tangencial se presenta cuando la imagen no está perfectamente alineada con el plano, de forma que es posible que veamos un área de la imagen más cerca de lo que esperamos.

A partir de estas dos distorsiones podemos concluir que cada cámara tiene unos coeficientes de distorsión que tendremos que calcular para una correcta calibración de la cámara.

Estos parámetros mencionados son los llamados parámetros intrínsecos de la cámara, pero también existen unos parámetros extrínsecos que así mismo es

necesario conocer para la calibración y que se corresponden con vectores de rotación y de traslación de la cámara respecto del plano de calibración. Existe un vector de rotación y otro de traslación para cada una de las fotos o vistas realizadas.

Para realizar la calibración de la cámara es necesario imprimir un tablero similar al de un ajedrez que contiene una serie de aristas verticales y horizontales. Este tablero se imprime como patrón conocido, de manera que conocemos todas las dimensiones del mismo, que posteriormente deberán ser introducidas en el código realizado.

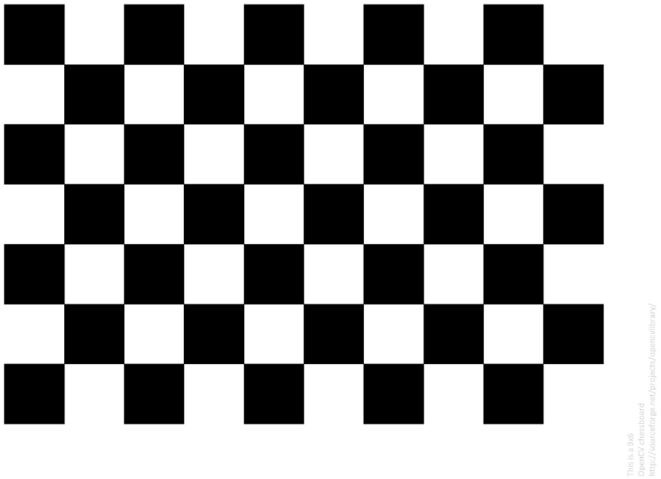

Figura 3.3: Tablero calibración cámara

Una vez el tablero impreso, hay que realizar un programa que ayude a calibrar la cámara, obteniendo también los diferentes parámetros arriba mencionados (intrínsecos, extrínsecos y coeficientes de distorsión). Lo que se pretende es que cuando se reproyecte un punto 3D estimado del tablero, cada reproyección deberá quedar lo más cerca posible de donde se han encontrado esos mismos puntos en la imagen. El procedimiento de calibración consiste en estimar los parámetros que minimicen el error de reproyección de todas las imágenes. Con la ayuda de algunos módulos de la librería OpenCV ("*findChessboardCorners, calibrateCamera*"), es posible realizar dicha optimización y obtener los parámetros.

Para realizar la calibración hay que tomar una serie de fotografías al tablero desde diferentes ángulos y perspectivas, ya que cuantas más fotografías existan en diferentes ángulos y vistas variadas, menor será el error de reproyección.

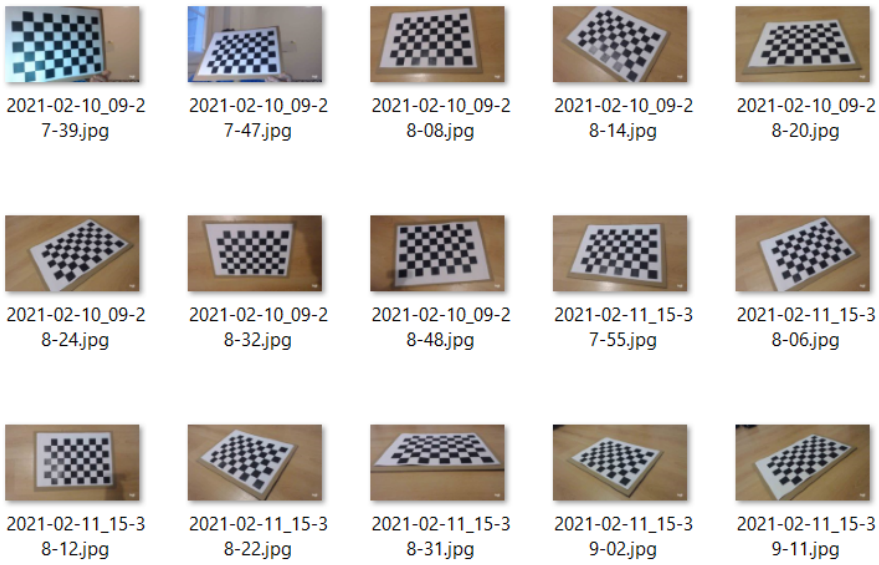

Figura 3.4: Fotografías tomadas a tablero de calibración

Lo siguiente que se obtiene es el reconocimiento de las aristas de cada una de las imágenes, como se puede observar en la siguiente imagen:

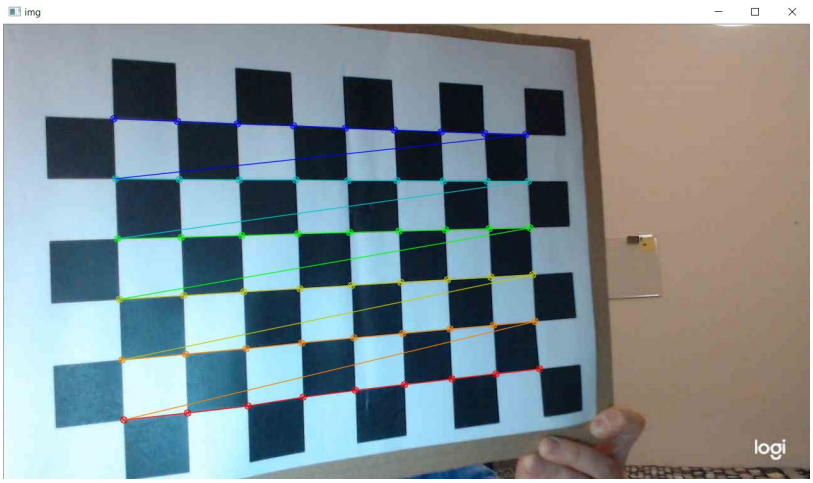

Figura 3.5: Detección aristas tablero

Una vez se han reconocido las aristas del tablero para todas las vistas, se calculan los parámetros extrínsecos e intrínsecos de la cámara mediante la optimización arriba comentada. Para ello, se procede a desarrollar un código con la finalidad de que se nos devuelva la matriz de calibración, los coeficientes de distorsión, los vectores de rotación y los vectores de traslación.

El programa devuelve la matriz de calibración al principio, con las correspondientes distancias focales y centros ópticos, así como los diferentes coeficientes de distorsión y devuelve el vector de rotación y de traslación de cada una de las distintas fotografías que he realizado previamente. A continuación, mostramos los resultados de calibración, en particular la matriz de calibración (1) y los coeficientes de distorsión (2).

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 634.66652243 & 0 & 330.26593214 \\ 0 & 634.65895534 & 250.96867883 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

$$\text{dist} = [-0.0133161 \quad 0.05979842 \quad 0.00547063 \quad 0.00427521 \quad -0.22533614] \quad (2)$$

Para continuar con el trabajo, fue necesario que conservase la matriz de calibración y los coeficientes de distorsión, para poder introducirlos en funciones posteriores y así detectar correctamente los marcadores ArUco y poder lograr representar correctamente los elementos a dibujar.

3.2. Reconocimiento de marcadores

Lo primero que se ha de llevar a cabo es el correcto reconocimiento de un marcador ArUco, obteniendo así la matriz de transformación de la cámara al marcador, y poder hacer las distintas conversiones para representar un elemento adecuadamente.

El procedimiento para detectar un marcador se basa en que el negro se diferencia muy bien del blanco y así, a base de extraer contornos, es posible determinar formas que parecen cuadrados. Estos cuadrados están divididos en píxeles blancos y negros, de manera que cada uno sea único. Existen diferentes diccionarios de marcadores, de manera que, entre los cuadrados extraídos, se compara con los cuadrados que posee el diccionario que hemos decidido utilizar, de forma que se identifique el marcador ArUco que la cámara está visualizando.

Para conocer la posición y la orientación del ArUco, es necesaria la previa calibración de la cámara puesto que vamos a utilizar la información de los puntos

del patrón del marcador en la imagen para compararlos con los puntos del patrón del ArUco correspondiente (que es conocido). Con esa correspondencia entre imagen y patrón conocido, se obtiene la rotación y traslación del objeto con respecto a la cámara mediante un algoritmo de PNP (perspective-N-point) que viene implementado en la librearía OpenCV. Como resultado, obtenemos la transformación entre los ejes de la cámara y los ejes del ArUco.

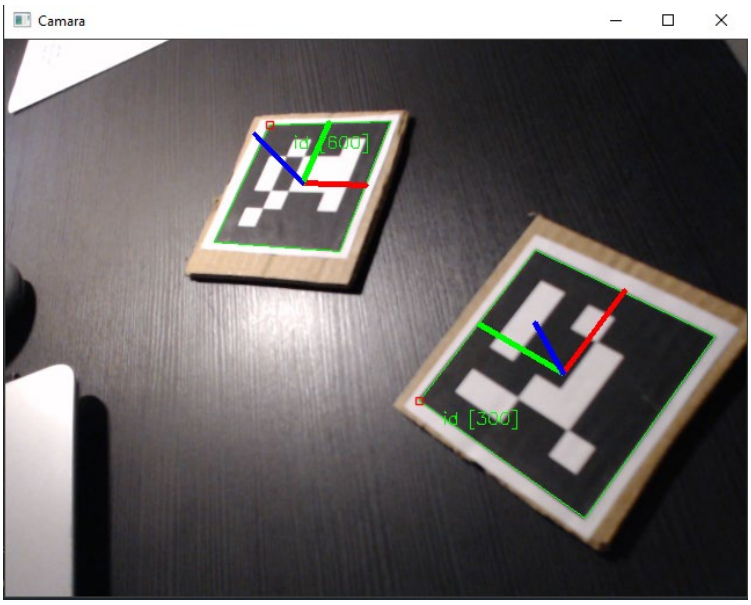

Figura 3.6: Reconocimiento de marcadores ArUco

3.3. Proyección de elementos

Lo siguiente a realizar es la proyección de un punto o de una línea sobre el plano.

La proyección de un punto 3D sobre la imagen se transforma mediante la matriz de calibración K. Para la correcta proyección es necesario transformar las coordenadas de un punto del mundo a las coordenadas de la cámara. Se tiene un punto en 3D que corresponde con x=[x,y,z]', y se pretende proyectar en la imagen en la coordenada (u,v). Con la matriz de calibración K (1), resulta lo siguiente:

$$\begin{bmatrix} s*u \\ s*v \\ s \end{bmatrix} = K * \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (3)$$

siendo s la distancia de la cámara al plano 2D en el que está proyectando, de manera que las coordenadas (u,v) se obtienen dividiendo por s, quedando s*u/s y s*v/s. Sustituyendo K por la matriz (1), queda lo siguiente:

$$u = f_x * \frac{x}{z} + c_x \quad (4)$$

$$v = f_y * \frac{y}{z} + c_y \quad (5)$$

Estas son las operaciones a realizar para convertir un punto 3D a un punto 2D, pero solamente es válido si el punto 3D está en la referencia de la cámara. Como nuestro trabajo lo vamos a situar en la referencia del plano que nos da el marcador ArUco, se tiene que transformar a la referencia de la cámara con la transformación que nos devuelve el ArUco.

En la sección anterior hemos obtenido las transformaciones (rotaciones y traslaciones) entre los ejes del ArUco y la referencia de la cámara. En este trabajo utilizaremos un ArUco, al cual nos referiremos con el número 1, que será la referencia del mundo sobre el que situaremos los elementos de realidad aumentada. Con su transformación ${}_cT_1$ podemos llevar los elementos situados en el plano a la referencia de la cámara para su proyección en la imagen. A su vez, también introduciremos un segundo ArUco, el del perro (numerado con 2), el cual tendremos que situar en el plano para saber cómo interactúa con el rebaño. Entonces, nos hace falta calcular la posición relativa (rotación y traslación) del segundo ArUco respecto al primero, que obtenemos con la siguiente ecuación:

$${}_1T_2 = {}_1T_c * {}_cT_2 = \quad (6)$$

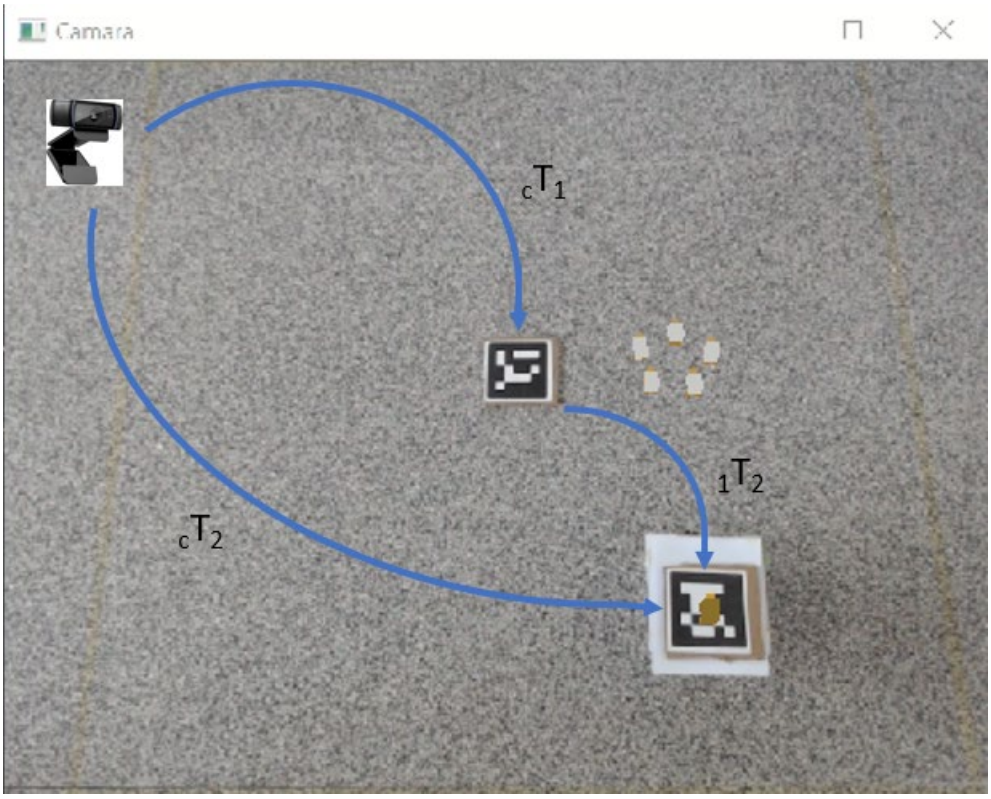

Figura 3.7: Matrices de transformación

Para situar elementos en el plano, se ha considerado que las dimensiones del marcador eran de 1x1, de tal forma que si establecíamos el punto (-0.5, 0.5, 0) se corresponde con el punto rojo marcado como origen en la figura anterior, siendo el eje rojo la X, el eje verde la Y y el eje azul la Z.

Una vez conocida la posición del marcador ArUco y, por ende, el origen respecto al cuál queremos situar los elementos, la matriz de calibración K, y las unidades de medida del eje de coordenadas, ya podemos representar puntos o líneas formando un plano y dibujar prácticamente cualquier elemento, como puede ser un cubo 3D. Para ello, se marcan los 8 puntos correspondientes al cubo, se proyectan a 2D y se dibujan sobre el plano. Para dibujarlos, se hace uso de una función que traza una línea entre dos puntos, de tal manera que, uniendo los puntos de forma coherente, obtenemos el cubo 3D deseado.

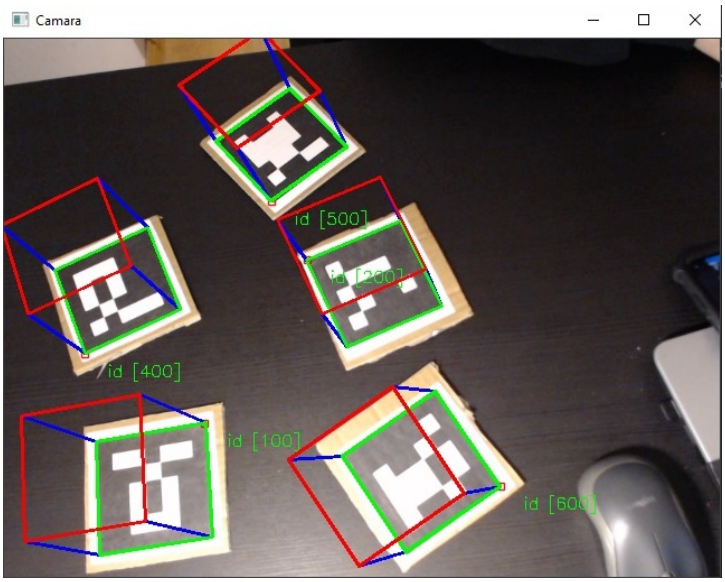

Figura 3.8: Representación de cubo 3D

También, se puede representar figuras geométricas más complejas, como se puede observar en la siguiente imagen la representación de una casita. Para ello ha sido necesario aumentar el número de puntos sobre el mundo 3D, y transformarlos a las coordenadas de la cámara, trazando las líneas y contornos correspondientes.

Figura 1.9: Representación de casitas

Como objetivo final, se decide representar una oveja. Para ello, asumimos que la representación va a consistir en un conjunto de cubos que simbolizan el cuerpo, la cabeza y las patas de la oveja, de manera que sea más sencillo dibujarlo. Se desarrolla una función que nos representa una pata en el centro del ArUco, definida con los puntos que se han determinado en 3D. A partir de esta función,

se desarrolla una segunda que nos permita crear 4 patas a partir de la función anterior, pero cada una desplazada a la posición correspondiente en coherencia con una oveja.

Una vez tenemos las 4 patas de la oveja, se procede a desarrollar la cabeza y el cuerpo. Para ello, dentro de la función donde se va a llevar a cabo, se invoca a la función que nos representa las 4 patas, y posteriormente se representa la oveja y el cuerpo, determinando unos puntos 3D y proyectándolos a 2D. En la siguiente figura, siendo el eje verde X, el eje rojo Y y el eje azul Z, se puede observar el resultado final de representar una oveja y, además, ésta está rotada -90º con respecto al ArUco.

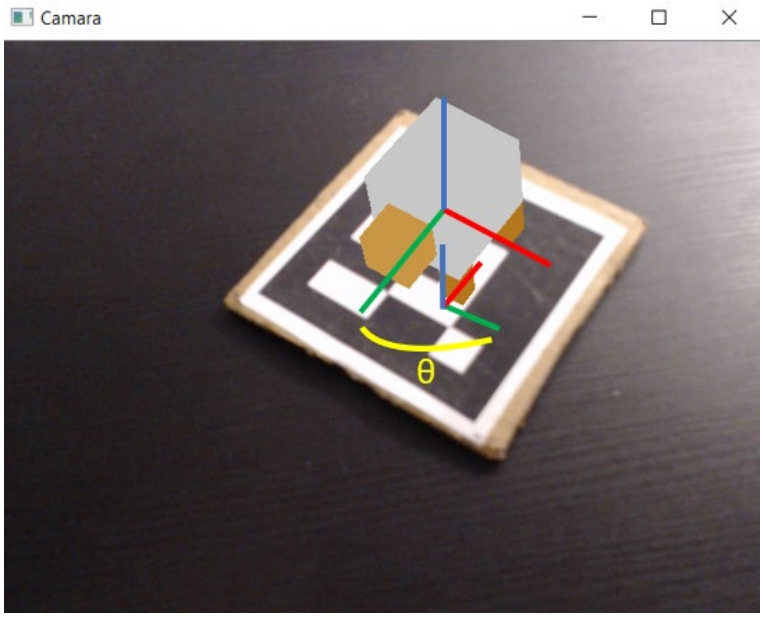

Figura 3.10: Representación oveja

Se ha incorporado el código necesario para poder colocarla en posiciones diferentes, añadiendo una componente X y una componente Y, de tal manera que, con la siguiente fórmula, conseguimos desplazar todos los vértices de la oveja, que se encuentran guardados en un vector:

$$\overrightarrow{desplazamiento} = (x, y, 0) \quad (7)$$

$$axesPoints = axesPoints + \overrightarrow{desplazamiento} \quad (8)$$

siendo que axesPoints es el vector donde se encuentran todos los puntos correspondientes a la oveja.
Por otro lado, también se ha implementado el código necesario para poder orientar cada oveja en la dirección que se desee, para ello hemos empleado las siguientes ecuaciones:

$$axesPoints_{x_k} = \cos(\theta) * axesPoints_{x_{k-1}} - \sin(\theta) * axesPoints_{y_{k-1}} \quad (9)$$

$$axesPoints_{y_k} = \sin(\theta) * axesPoints_{x_{k-1}} + \cos(\theta) * axesPoints_{y_{k-1}} \quad (10)$$

Con estas ecuaciones, lo que se consigue es obtener los puntos correspondientes a la rotación de un ángulo θ dado.
Esto nos ayudará cuando se le quiere dar una velocidad a una oveja, ya que, si ésta se mueve en una dirección determinada, la orientación de la oveja deberá ser en la dirección hacia donde se está dirigiendo, según el modelo de movimiento que se ha implementado.

3.4. Oclusiones

Se ha conseguido dibujar elementos de líneas y planos, pero se debe conocer en qué orden han de pintarse las superficies, ya que si se dibuja arbitrariamente puede que algún elemento no tenga sentido y aparezcan problemas de oclusión.
Para ello, se va a analizar la forma geométrica más sencilla que hemos realizado, que es el cubo3D debido a que, si se da solución de oclusión a un cubo 3D, también se le podrá dar a las ovejas, ya que están construidas por varios cubos.
Partiendo de la observación, se pudo comprobar que, en algunos casos los cubos se solapaban cuando un cubo estaba delante de otro. Comprobamos que, en el mundo real, la superficie que se ve es la parte que está más próxima a nuestros ojos. Lo mismo ocurre con la cámara. Por ello, la manera correcta de proceder es representar primero aquellos puntos que estén más lejanos, e ir superponiendo los puntos más cercanos.

Para resolverlo, se desarrolló una función que devolviera la distancia que hay desde cada punto del cubo a la cámara, para así poder determinar los criterios a la hora de representar el cubo. Lo que hice fue calcular el punto que quedaba más lejano a la cámara, y desde ahí empezar a representarlo en función de las aristas que vería realmente. Tras unas cuántas pruebas, éste fue el resultado:

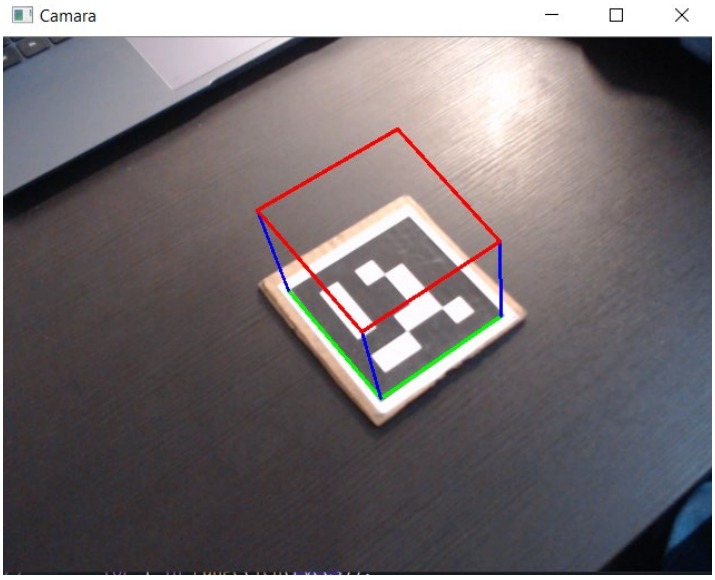

Figura 3.11: Cubo 3D con oclusiones solucionadas

Tras poner solución a las oclusiones en un cubo 3D, se procede a representar una oveja, comprobando que ocurre exactamente lo mismo, ya que no dibujaba los "componentes" de la oveja de la manera correcta. En la siguiente imagen se puede ver cómo representa el cuerpo de la oveja sobre la cabeza, cuando mirándola de frente debería ser al revés.

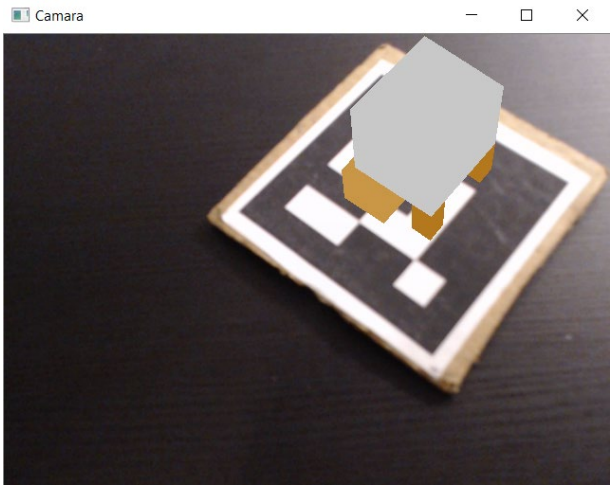

Figura 3.12: Oveja mal representada

De manera similar a lo realizado con el cubo 3D, se tuvo que calcular la distancia de cada vértice de la oveja para poder determinar qué debería ser dibujado primero. Dado que la oveja está formada por varios cubos, y que éstos ya los sabemos dibujar correctamente, lo que se trata de realizar ahora es la colocación de cada "parte" de la oveja correctamente. Para ello, siendo que la oveja está constantemente enfocada desde una vista superior, lo primero que se dibuja son las patas y posteriormente el cuerpo.

En cambio, para dibujar la cabeza tenemos que tener en cuenta cuál de los vértices del cubo que constituye el cuerpo queda más lejano, obteniéndose así las condiciones necesarias para la correcta representación.

El resultado fue el siguiente:

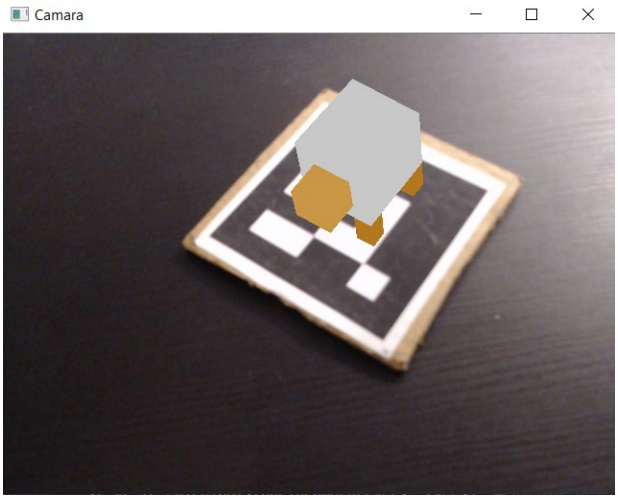

Figura 3.13: Oveja bien representada

Una vez se representa una oveja correctamente, existe el problema de que cuando hay más de una oveja, también puede darse el caso de que se solapen. Cuando una oveja se pone delante de la otra, se mezclan entre ellas de la siguiente forma.

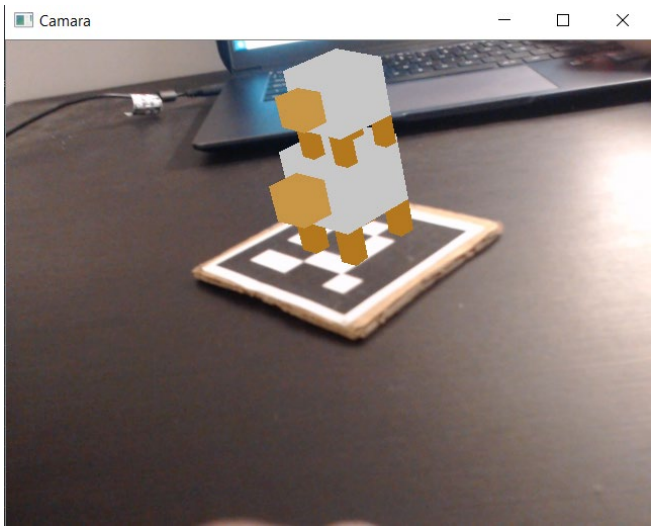

Figura 3.14: Ovejas con problema de oclusión

Para poder solucionar este problema, se realiza algo similar a la función de representar una oveja, pero con matices diferentes también. Calculamos la distancia de cada oveja desde el centro de ella misma a la cámara. De esta manera, se empieza a dibujar aquellas ovejas que estén más lejanas y así ir superponiendo las ovejas que deberán verse por delante de ella, quedando el resultado final de la siguiente manera:

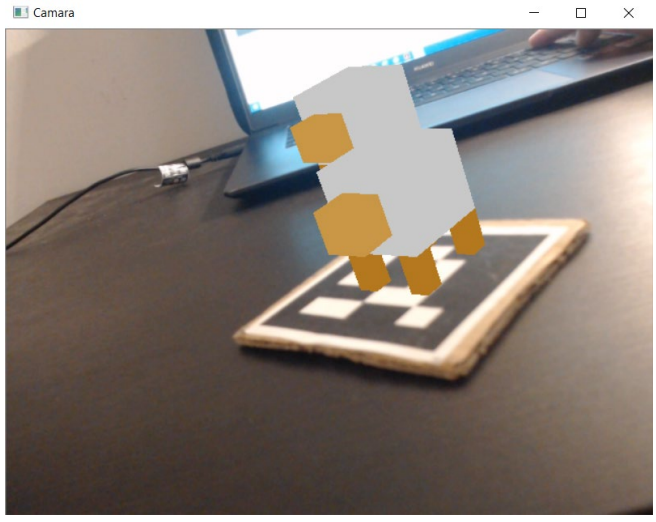

Figura 3.15: Ovejas con problema de oclusión solucionado

4. Movimiento de las ovejas

En este capítulo se va a desarrollar la implementación que se ha seguido para incorporar velocidades a cada uno de los elementos dibujados en el plano para que simulen un comportamiento natural de rebaño y reaccionen ante la presencia de un perro pastor.

4.1. Movimiento independiente

Manteniendo el mismo objetivo de realizar la simulación de un rebaño de ovejas, se deberá implementar el código necesario para que cada una de ellas tenga un movimiento independiente sobre un plano. Para ello se ha llevado a cabo dos modelos de movimiento, un controlador integrador y un controlador de tipo differential drive.

4.1.1. Integrador

Este modelo dará como resultado una velocidad lineal en una dirección x e y, determinada por dos variables de velocidad. Para ello, el criterio inicial que se ha seguido es que, a la hora de definir una oveja, ésta tendrá como parámetros los siguientes:

- Variable inicial x e y, correspondientes con las coordenadas de posición de la oveja.
- Variable inicial θ, que nos determina la orientación.
- Variable v_x y v_y, que indican la velocidad que debe seguir la oveja entorno al eje X y al eje Y.
- Variable v_w, que nos indica la velocidad de rotación de la oveja.
- Variable radio, que hace referencia al radio de giro.

Habida cuenta de esto y, para poder darles una velocidad, se debe crear una función que nos devuelta una nueva posición de la oveja, dada una variable v_x y otra v_y deseadas.
Para que las ovejas se muevan de una manera más realista, se ha decidido orientar previamente la oveja en la dirección lineal en la que se va a desplazar. Aunque

esto no determina el movimiento integrador, se considera que es una simulación más aproximada a la realidad. Esto se obtiene mediante la siguiente fórmula:

$$\theta_{ori} = \text{atan}\left(\frac{v_y}{v_x}\right) \quad (11)$$

Además, se ha normalizado entre 0º y 360º, de manera que, si el ángulo orientativo (θ_{ori}) resultado es menor que 0º, se suma 360º automáticamente.
Una vez tenemos el ángulo orientativo al que debe estar cada oveja, se realizan una serie de conversiones con la finalidad de tener todo mejor parametrizado, en concreto se divide la variable v_x e v_y entre 7.5, que es la dimensión real del marcador ArUco (7.5 cm x 7.5 cm), para que así si asignamos una v_x=1, se mueva 1 cm cada segundo.

Con ello, se realizan las siguientes fórmulas:

$$x = x_{anterior} + v_{x_0} * T + v_{xx} * T \quad (12)$$

$$y = y_{anterior} + v_{y_0} * T + v_{yy} * T \quad (13)$$

$$\theta = \theta_{ori} \quad (14)$$

siendo v_{xo} y v_{yo} las componentes de velocidad iniciales que se le asigna a cada oveja, v_{xx} y v_{yy} las componentes de velocidad asociadas al modelo de comportamiento en grupo del que hablaremos más adelante, θ el ángulo que hemos calculado anteriormente. T es el tiempo que tarde el bucle principal en ejecutarse.
Una vez realizado esto, debemos volver a multiplicar cada componente v_x, v_y, v_{xx} y v_{yy} por 7.5 para dejarlas en su estado inicial, así como normalizar el ángulo θ.
Para la normalización del ángulo θ, se ha realizado una función dada la cual si el ángulo es mayor que 180º, se le resta 360º. Y si por el contrario es menor que -180º, se le suma 360º. De esta forma, el ángulo θ queda delimitado entre -180º y 180º.

4.1.2. Differential drive

Este modelo de movimiento define un comportamiento más realista de las ovejas. En este caso, los parámetros de la función que se ha de definir para llevarlo a cabo este modelo son una velocidad lineal v, una velocidad angular w, las posiciones x e y de la oveja, y su respectivo ángulo θ que define su orientación.
Por ello, se han aplicado las siguientes fórmulas:

$$x = x_{anterior} + \frac{v}{7.5} * T * \cos(\theta + w * T) \quad (15)$$

$$y = y_{anterior} + \frac{v}{7.5} * T * sin(\theta + w * T) \quad (16)$$

$$\theta = \theta_{anterior} + w * T \quad (17)$$

Como se puede observar, es necesario dividir la velocidad v entre 7.5 que son los cm que mide el marcador ArUco, haciendo así que la velocidad implementada en cm/s coincida con la realidad.
La diferencia entre un movimiento y el otro, es que en el differential drive, las ovejas no pueden moverse lateralmente, mientras que en el otro sí.
En la sección de experimentos se podrá visualizar las diferencias entre ambos movimientos.

4.2.Movimiento en grupo

Para la correcta simulación de este trabajo, una vez que las ovejas son capaces de tener una velocidad y orientación independientes respecto a otras ovejas, es necesario que obtengan un comportamiento en grupo, de manera que cada oveja tenga una cierta cohesión con respecto a las otras, que mantengan una cierta distancia que permita que no se choquen, y que reaccionen de manera repulsiva ante la presencia de un perro pastor. Para ello, se han considerado diferentes modelos de comportamiento en grupo y, a partir de ellos, se ha implementado el código necesario para que este trabajo funcione correctamente. Los modelos de comportamiento en grupo que se han estudiado han sido los siguientes:

4.2.1. Algoritmo de Reynolds

Para la realización de este proyecto, se ha obtenido información acerca del modelo de comportamiento en grupo realizado por Craig Reynolds. [6]
Este modelo se basa en una simulación que replica el comportamiento de bandadas de aves. Según dicho modelo existen unas reglas que se aplican individualmente a cada ave, de forma que el comportamiento total simula un movimiento en grupo. Cada ave tiene su propia posición, velocidad y orientación. Con el objetivo de conseguir un comportamiento dinámico en grupo, se han de aplicar estas tres reglas:

- Separación: es la distancia que debe respetar cada ave con sus aves cercanas, de forma que, modificando unos parámetros, esta fuerza de repulsión/atracción sea mayor o menor.
- Alineación: las aves modifican su posición para que corresponda con la alineación promedio de otras aves cercanas.
- Cohesión: cada ave intenta moverse hacia la posición promedio de otras aves cercanas.

Con estas tres simples reglas, es posible conseguir que cada oveja se vea influenciada en cierta medida por las ovejas que tiene a su alrededor, de forma que se logra casi de forma realista un movimiento de rebaño.
Aunque no se ha realizado finalmente este algoritmo, el método que se ha desarrollado sigue esta misma filosofía.

4.2.2. Algoritmo de flock control

Este modelo se basa también en unas fuerzas de atracción y repulsión entre las ovejas, el perro y, en caso de haberlo, un muro o barrera.
Según Richard Vaughan, Neil Sumpter, Andy Frost y Stephen Cameron en su documento de “Experiments in automatic flock control” [3], este modelo incluye una fuerza atractiva que actúa entre los animales, cuya magnitud de atracción varía con el cuadrado inverso de la distancia mutua entre ambos animales.

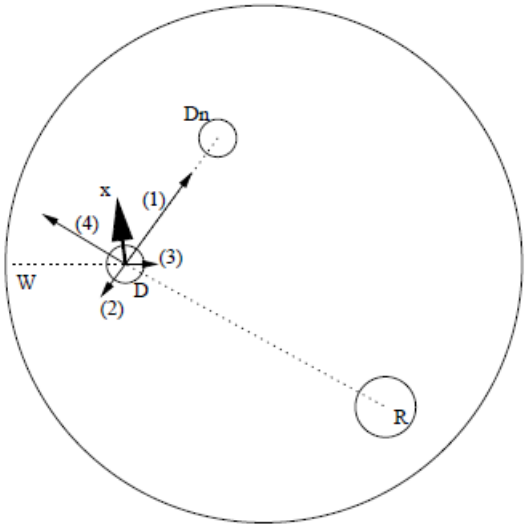

Figura 4.1: Descripción flock model [3]

En este caso, tenemos una fuerza de atracción (1), una fuerza de repulsión (2), una fuerza de repulsión a las paredes (3) y una fuerza de repulsión al perro (4). Por ello, la ecuación que define este comportamiento es la siguiente:

$$\mathbf{d} = \sum_{n=1}^{N} \left(\left(\frac{K_{f1}}{|\mathbf{DD_n}|^2} \right) \widehat{\mathbf{DD_n}} - L - \left(\frac{K_{f2}}{|\mathbf{DD_n}|^2} \right) \widehat{\mathbf{DD_n}} \right) - \left(\frac{K_{f3}}{|\mathbf{DW}|^2} \right) \widehat{\mathbf{DW}} - \left(\frac{K_{f4}}{|\mathbf{DR}|^2} \right) \widehat{\mathbf{DR}} \qquad (18)$$

(1) (2) (3) (4)

En esta ecuación obtenemos un vector de movimiento compuesto por la suma de todos los términos de atracción y repulsión de la oveja en concreto respecto al resto de ovejas, depredadores y obstáculos que hay a su alrededor. Para ello, K_{f1}, K_{f2}, K_{f3}, y K_{f4} son constantes que se han de modificar para un correcto funcionamiento.

Además de estas constantes, se tiene la constante L. Este parámetro es el que nos ayuda a definir la distancia a la que queremos que se quede una oveja respecto a otra, logrando así que no se interpongan entre ellas.

Dado que los términos están divididos por la distancia entre las ovejas elevada al cuadrado, no hemos utilizado este modelo, ya que cuando una oveja se aproxima demasiado a otra, la fuerza que se crea entre ellas es demasiado elevada.

4.3. Implementación asociada a nuestro trabajo

Basándome en los modelos anteriores, se ha creado un modelo con las especificaciones de mi trabajo. Las ovejas deben de obedecer ante fuerzas de atracción y de repulsión dependiendo del objeto que tengan cerca.
Para la realización de este modelo, he tenido en cuenta las tres fuerzas comentadas en el apartado anterior.

4.3.1. Atracción al centroide del rebaño

Para ello, he implementado una ecuación que calcula el centroide de todas las ovejas. Dicha ecuación es la suma de todas las posiciones x divididas entre el número de ovejas y la suma de todas las posiciones y divididas entre el número de ovejas, quedando de la siguiente forma:

$$centroide_x = \sum_{i=0}^{nº\ ovejas} \frac{posición_{x_i}}{nº\ ovejas} \quad (19)$$

$$centroide_y = \sum_{i=0}^{nº\ ovejas} \frac{posición_{y_i}}{nº\ ovejas} \quad (20)$$

Con estas dos fórmulas, tendré un vector del centroide (x,y). Posteriormente, calculo para cada una de las ovejas el vector que une la oveja con el centroide y, con ello, la consiguiente distancia que hay entre ambos. Para ello se realiza lo siguiente:

$$vectorOvejaCentroide_i = [centroide_x - posición_{x_i}, centroide_y - posición_{y_i}] \quad (21)$$

$$distanciaCentroide_i = \sqrt{(centroide_x - posición_{x_i})^2 + (centroide_y - posición_{y_i})^2} \quad (22)$$

Una vez obtenidos estos datos, se desarrolla la ecuación de la que resultará la fuerza de repulsión/atracción al centroide:

$$F_x{}^{centroide} = \frac{vectorOvejaCentroide_x}{distanciaCentroide} * Kc * distanciaCentroide^{\alpha_1} \quad (23)$$

$$F_y{}^{centroide} = \frac{vectorOvejaCentroide_y}{distanciaCentroide} * Kc * distanciaCentroide^{\alpha_1} \quad (24)$$

siendo Kc la constante de proporcionalidad. Con respecto a las alternativas anteriores, en lugar de realizar una atracción oveja-oveja, desarrollamos una atracción al centroide del rebaño que es proporcional a la distancia, en lugar de ser proporcional inversamente como ocurría en el modelo anterior. De esta manera, si están alejadas tienden a acercarse rápidamente, mientras que, si están cerca, apenas tienen una fuerza de atracción al centroide.

4.3.2. Repulsión entre ovejas

Una vez obtenida la fuerza de atracción al centroide, todas las ovejas tenderán a ir a dicho punto. Por lo tanto, es necesario que exista una fuerza de repulsión entre ellas para que no se choquen por el camino, y tampoco en la disposición final.

De manera muy similar a lo que hemos hecho con el centroide, se calcula el vector distancia que hay entre cada una de las ovejas, además del valor numérico de distancia entre ellas.

$$F_x{}^{ovejas} = -\frac{vectorEntreOvejas_x}{distanciaOvejas} * \frac{Kr}{distanciaOvejas^{\alpha_2}} \quad (25)$$

$$F_y{}^{ovejas} = -\frac{vectorEntreOvejas_y}{distanciaOvejas} * \frac{Kr}{distanciaOvejas^{\alpha_2}} \quad (26)$$

siendo Kr la constante de proporcionalidad. Como en este caso la distancia está dividiendo, este término tiene un valor mayor cuando más cerca están las ovejas entre sí.

4.3.3. Repulsión respecto al perro

Finalmente, debido a la aparición de un perro, tengo que representar la fuerza de repulsión respecto al perro para que dichas ovejas se alejen de él a medida que éste se acerque a ellas.

Para representar esta fuerza, se realiza la misma operación que en el apartado anterior, pero en este caso se considera el vector distancia de cada oveja respecto al perro, así como el módulo distancia. Por ende, resultarían las siguientes fórmulas:

$$F_x^{perro} = \frac{vectorOvejaPerro_x}{distanciaOvejaPerro} * \frac{Krr}{distanciaOvejaPerro^{\alpha_3}} \quad (27)$$

$$F_y^{perro} = \frac{vectorOvejaPerro_y}{distanciaOvejaPerro} * \frac{Krr}{distanciaOvejaPerro^{\alpha_3}} \quad (28)$$

siendo Kc, Kr y Krr parámetros a modificar para una funcionalidad correcta. Una vez obtenidas las 3 fuerzas F_x y F_y, hay que sumarlas para obtener el resultado total, el cual se aplicará directamente a la velocidad de cada oveja.

$$v_x = v_{x_{centroide}} + dv_x + dv_{xp} \quad (29)$$

$$v_y = v_{y_{centroide}} + dv_y + dv_{yp} \quad (30)$$

siendo $v_{xcentroide}$ e $v_{ycentroide}$ un vector compuesto por las fuerzas de cada oveja al centroide en dirección de la coordenada x e y, respectivamente, dv_x y dv_y el sumatorio de las fuerzas asociadas a la repulsión entre ovejas, y dv_{xp} y dv_{yp} el sumatorio de las fuerzas correspondientes a la repulsión ante un perro. Estos dos últimos sumatorios son la suma de todos los elementos de cada fila de la matriz resultante.

La suma de todos los términos determina las nuevas componentes de velocidad que tiene cada oveja.

5. Demostraciones

En este capítulo se puede observar todos los experimentos que se han realizado para comprobar un correcto funcionamiento. En la demostración 5.1 – 5.2 se muestra el movimiento individual de una oveja, usando los algoritmos de la sección 4.1.1. En 5.3 – 5.4 se muestra el comportamiento individual igualmente, pero en este caso haciendo uso de los algoritmos de la sección 4.1.2. En 5.5 – 5.6 – 5.7 – 5.8 – 5.9 se muestran comportamientos de grupo (swarm) del rebaño de ovejas, variando los parámetros del algoritmo descrito en la sección 4.3.

Para la correcta simulación de los experimentos, se ha procedido a desarrollar una función que nos permita visualizar el plano donde vamos a representar los distintos elementos. Esta función hace que al ejecutar el programa se abra una ventana emergente de la misma manera que se abre la cámara. En ella se puede visualizar un fondo blanco con unos ejes X e Y, donde se verán representadas las ovejas y el perro. Cada línea que corta a los ejes, mantiene una distancia con la siguiente de 7.5 cm. Se ha elegido este valor al ser intuitivo en nuestro caso, ya que corresponde con la longitud del lateral de las marcas ArUco utilizadas.

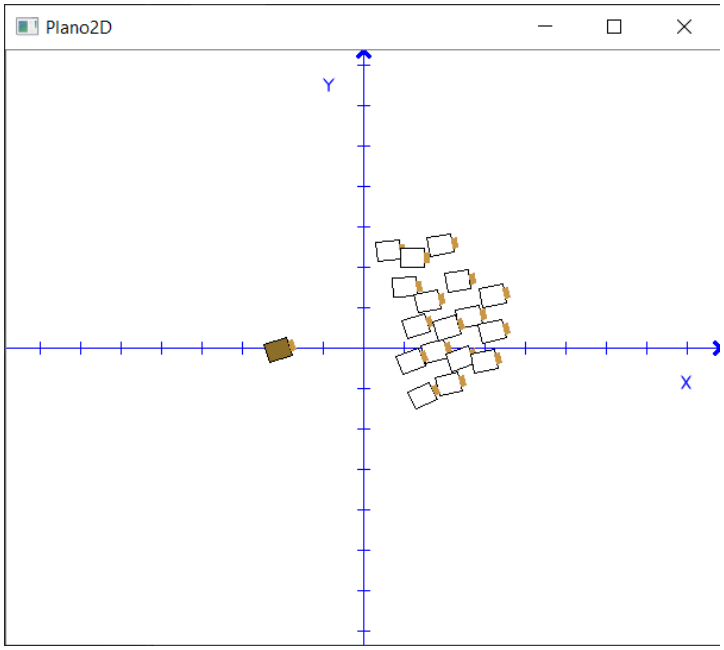

Figura 5.1: Representación plano 2D

Por otro lado, también se ha desarrollado una función que nos permita visualizar el recorrido realizado tanto por cada oveja como por el perro al finalizar la simulación. Para ello, se ha creado una serie de variables que recogen todas las posiciones por las que pasa cada elemento, y así poder dibujar una gráfica posteriormente. El comienzo de cada elemento se representa por una X, y el final del recorrido de cada elemento se ve representado por un triángulo apuntando hacia arriba, como se puede observar en la siguiente figura. Cabe decir que cada oveja está representada por un color distinto, mientras que el perro está representado por una línea de color negro.

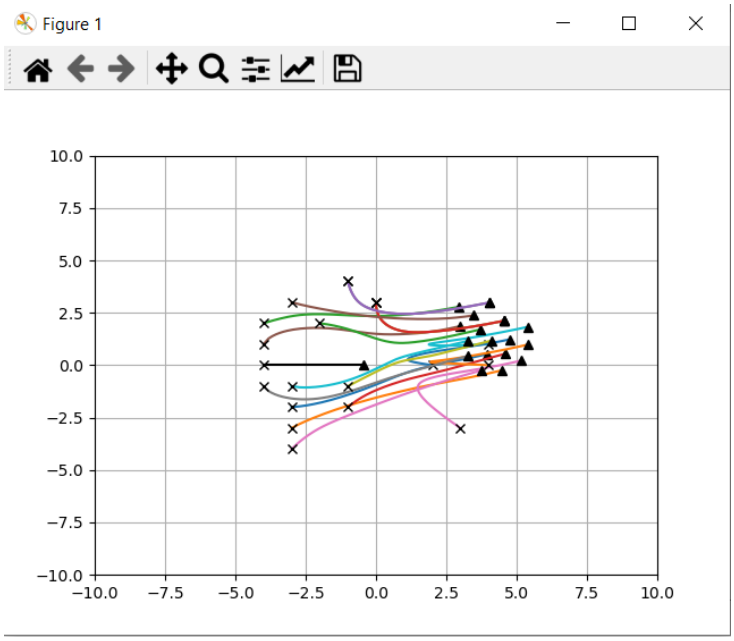

Figura 5.2: Gráfica de los distintos recorridos

Una vez detallado esto, se procede a adjuntar un ejemplo de simulación. Este ejemplo se realiza para poder visualizar cómo en la pantalla del ordenador sí se ven representadas las ovejas, mientras que en la imagen captada por un smartphone externo no se verán las ovejas en el mundo real.

Con esto, las siguientes demostraciones serán capturas de la pantalla del ordenador, así como grabaciones de video que se adjuntarán a través de enlaces de YouTube.

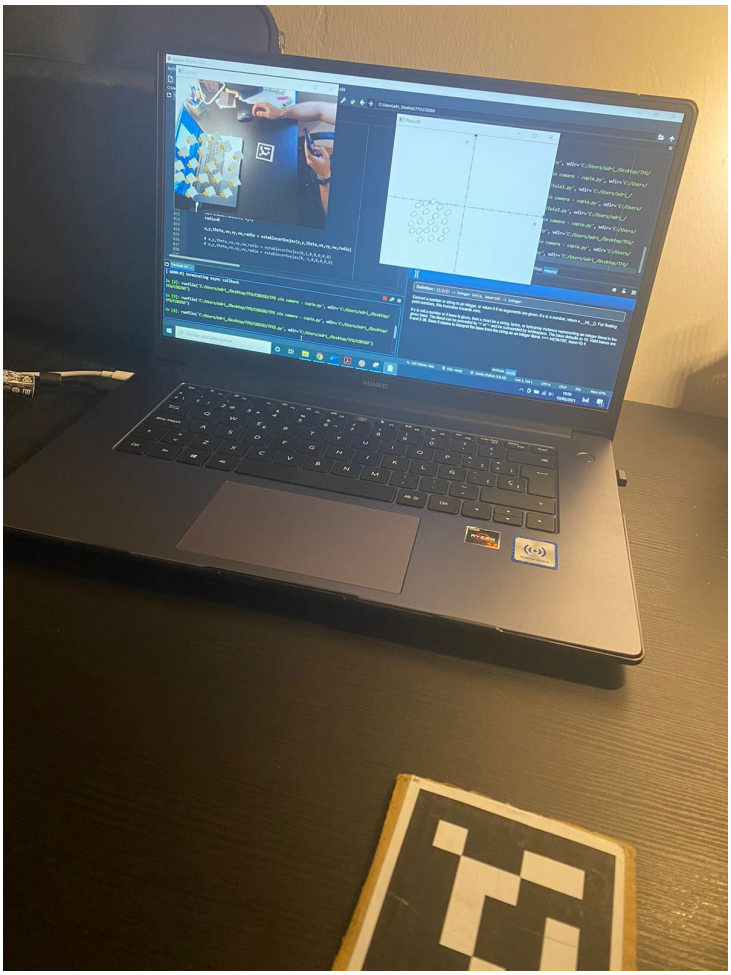

Figura 5.3: Imagen del plano de trabajo

En la imagen se puede observar como en el teclado del ordenador no hay ningún elemento visible, pero si se mira en la pantalla del ordenador, en la ventana que representa la cámara, se observa cómo hay un rebaño encima del teclado del ordenador.

5.1. Una oveja en movimiento lineal

En esta simulación se va a visualizar una oveja posicionada inicialmente en el centro del marcador ArUco, y con una velocidad lineal de manera que se desplace diagonalmente en la recta x=y. Para ello, se ha utilizado el movimiento integrador (sección 4.1.1) dando unos valores de v_x y de v_y de 7.5 cm/s. Por tanto, lo primero que hace la oveja es calcular el ángulo orientativo, como se explica en la sección 4.1.1, que resulta ser 45º, y posteriormente comienza el movimiento.
Enlace de YouTube: https://youtu.be/FBPdt4fbrOY

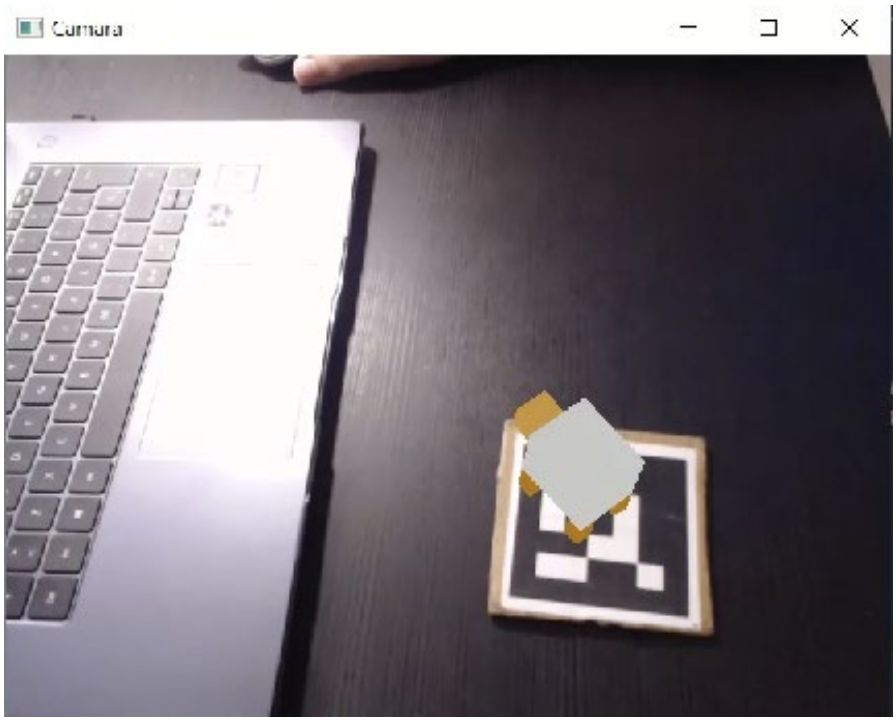

Figura 5.4: Posición inicial

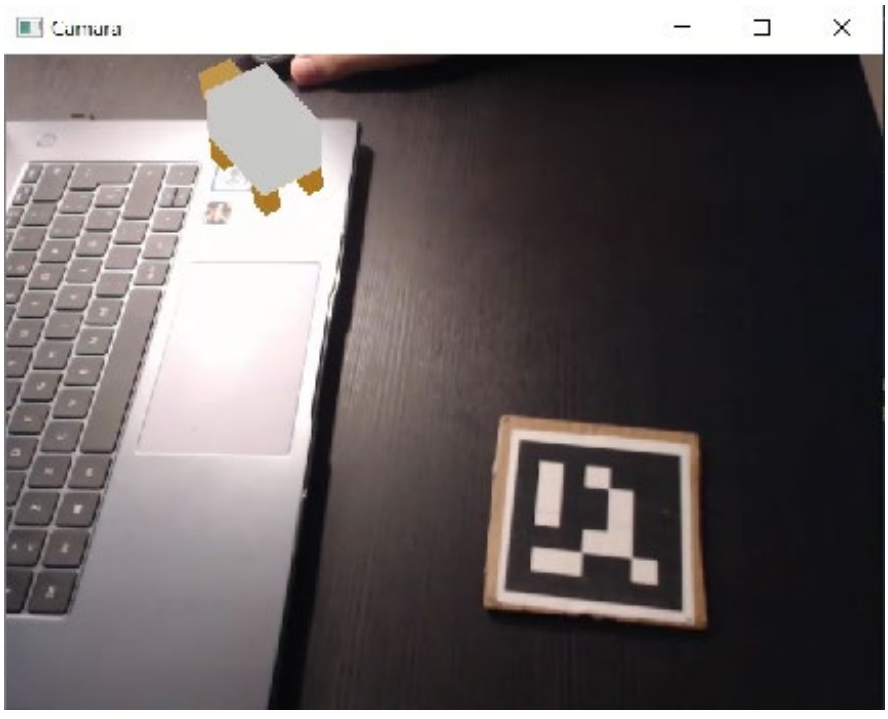

Figura 5.5: Posición final

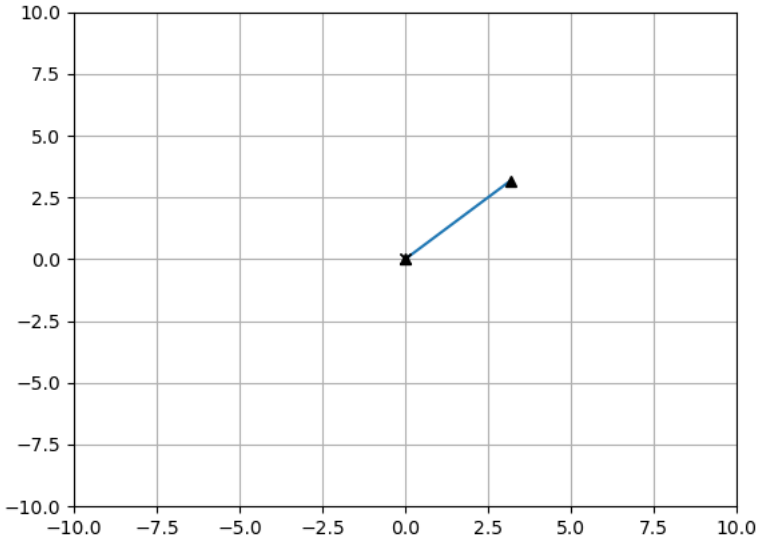

Figura 5.6: Gráfica recorrido simulación 1

5.2. Una oveja en rotación

En este caso, procedemos a visualizar una oveja estática en el centro del marcador ArUco. Utilizando el movimiento integrador (sección 4.1.1), hemos asignado un valor de 0 cm/s a las variables v_x y v_y, y un valor de velocidad angular w de 20º/s.

Enlace de YouTube: https://youtu.be/WbeZF5UN2bo

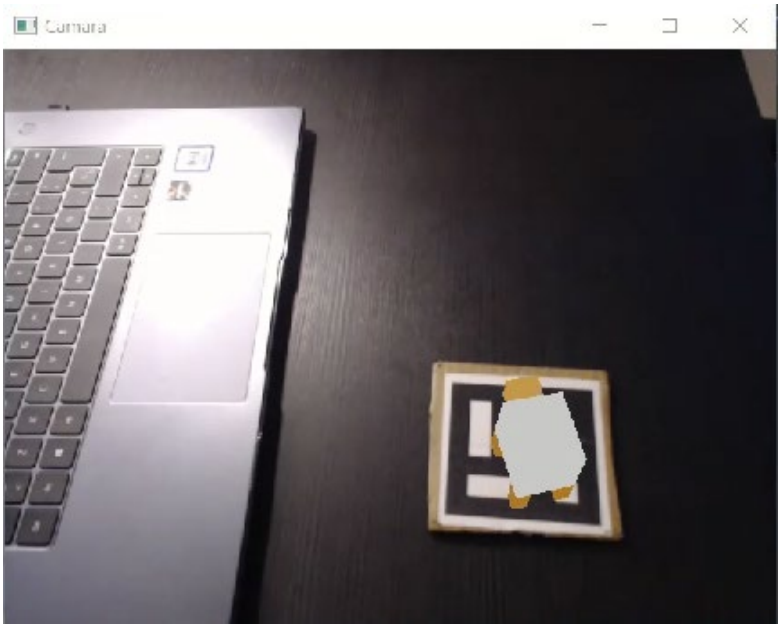

Figura 5.7: Posición inicial

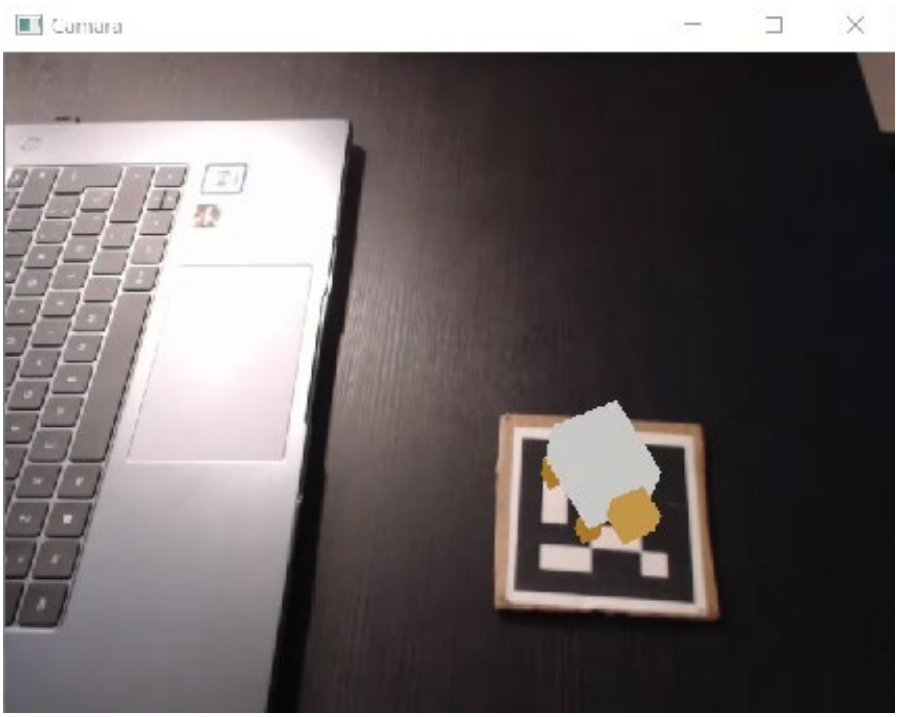

Figura 5.8: Posición intermedia 1

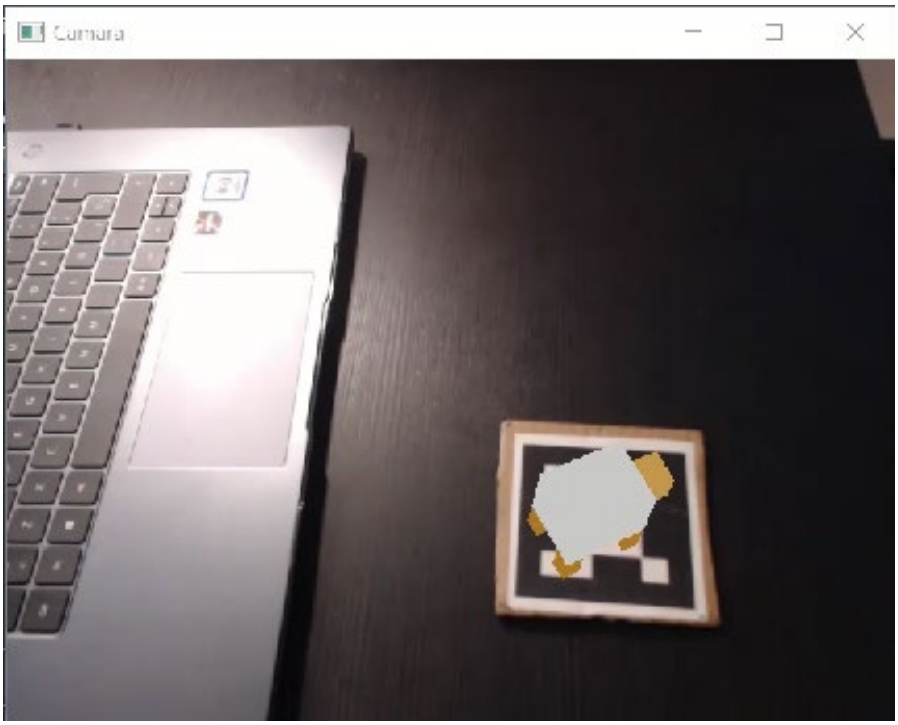

Figura 5.9: Posición intermedia 2

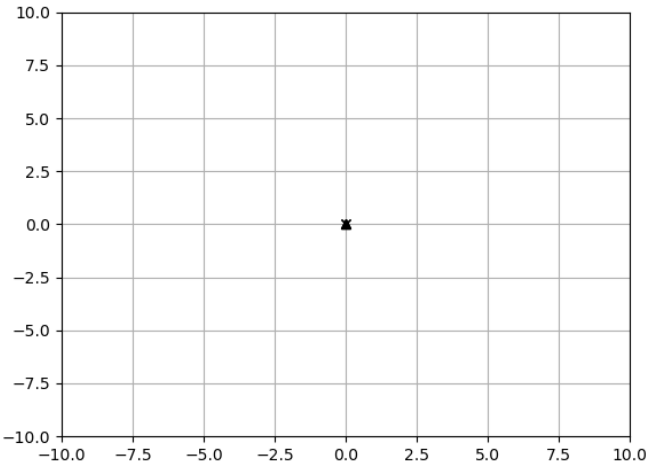

Figura 5.10: Gráfica recorrido simulación 2

5.3. Simulación de una oveja en movimiento differencial drive

La realización de esta simulación se ha llevado a cabo a través de un movimiento differential drive (sección 4.2.1), asignando una v=2 cm/s y una w=20º/s. Con estos datos podremos visualizar un movimiento curvilíneo de la oveja.

Enlace de YouTube: https://youtu.be/CyUSqqYVUPU

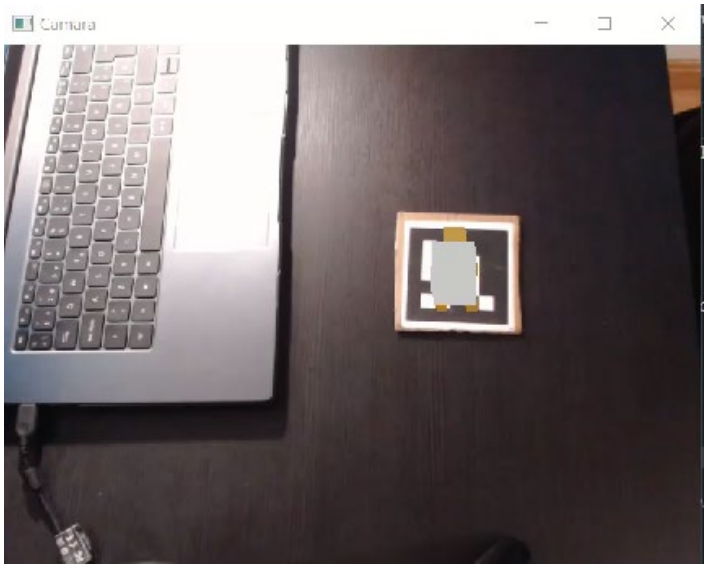

Figura 5.11: Posición inicial

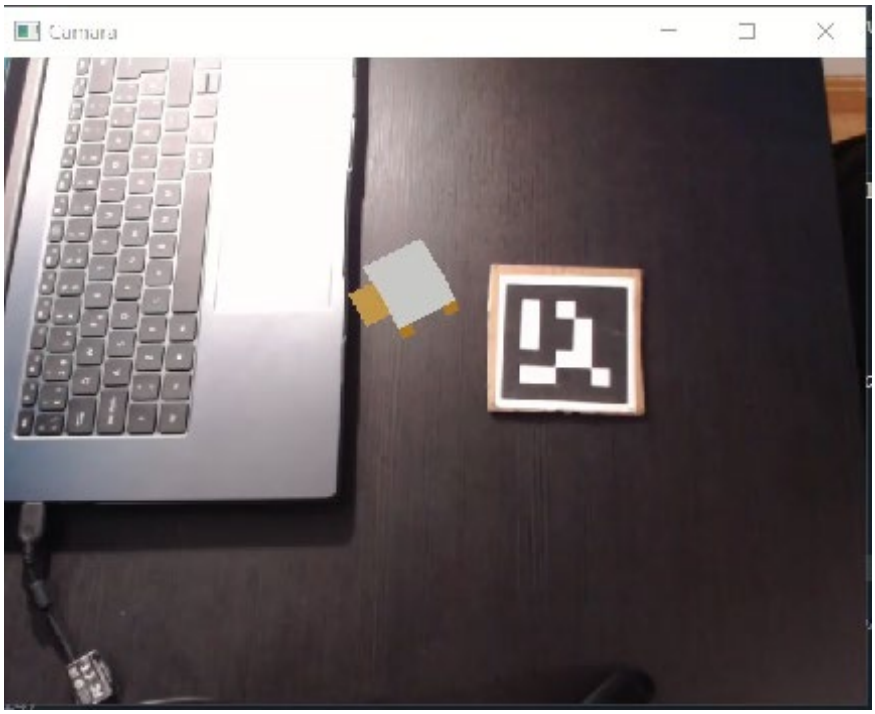

Figura 5.12: Posición intermedia

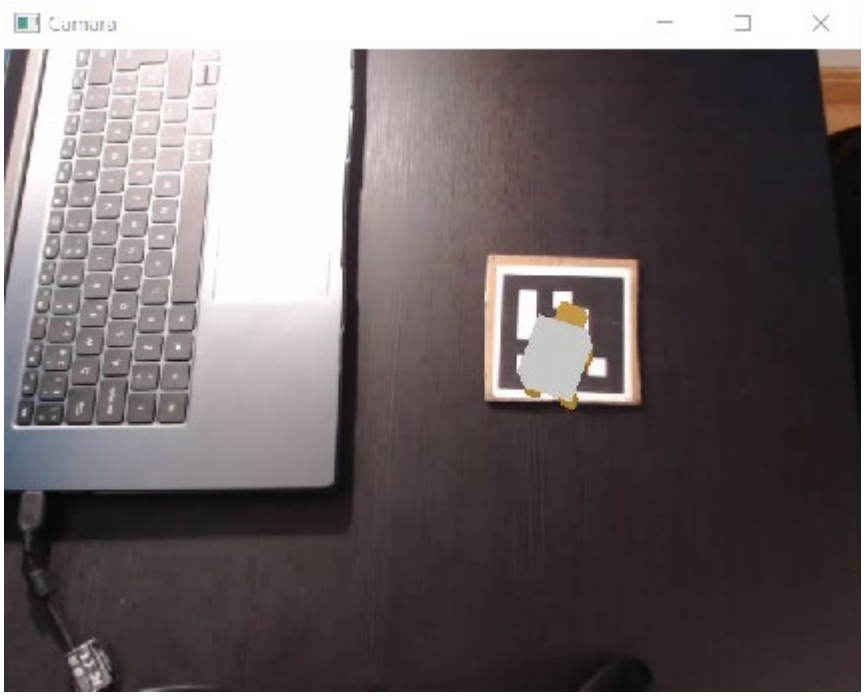

Figura 5.13: Posición final

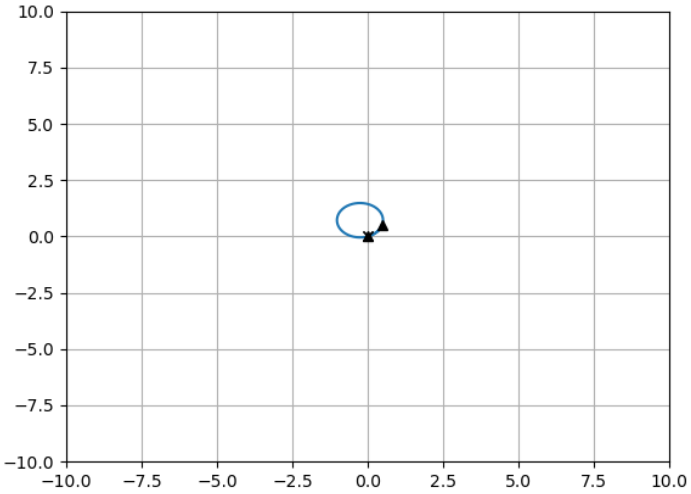

Figura 5.14: Gráfica recorrido simulación 3

5.4. Una oveja que cambia cada 4 segundos

En la siguiente simulación se puede observar una oveja que va a realizar movimientos de forma aleatoria siguiente un control de differential drive (sección 4.2.1). Los parámetros de entrada que se le asignen cambiarán mediante una función que genera números aleatorios cada 4 segundos.
Enlace de YouTube: https://youtu.be/EiD5R3a_b7U

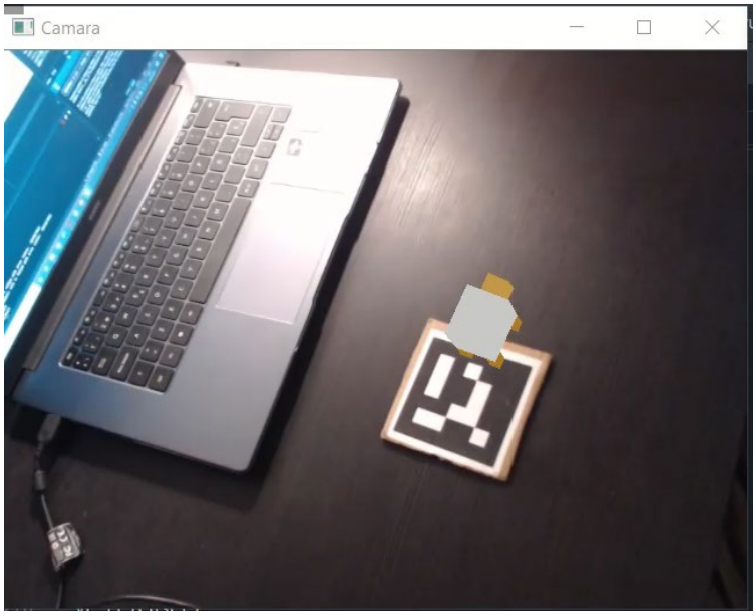

Figura 5.15: Posición inicial

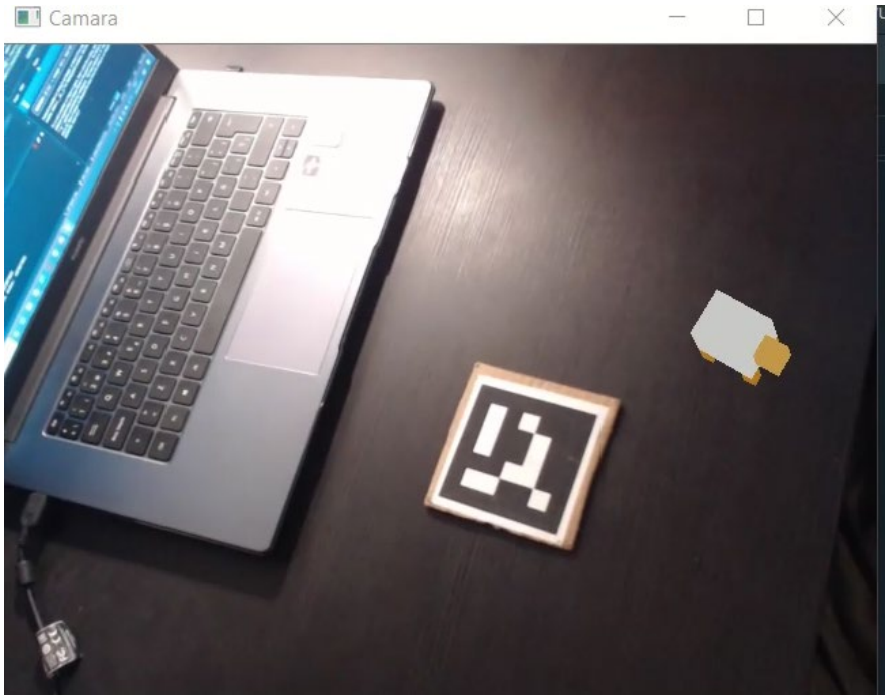

Figura 5.16: Posición intermedia

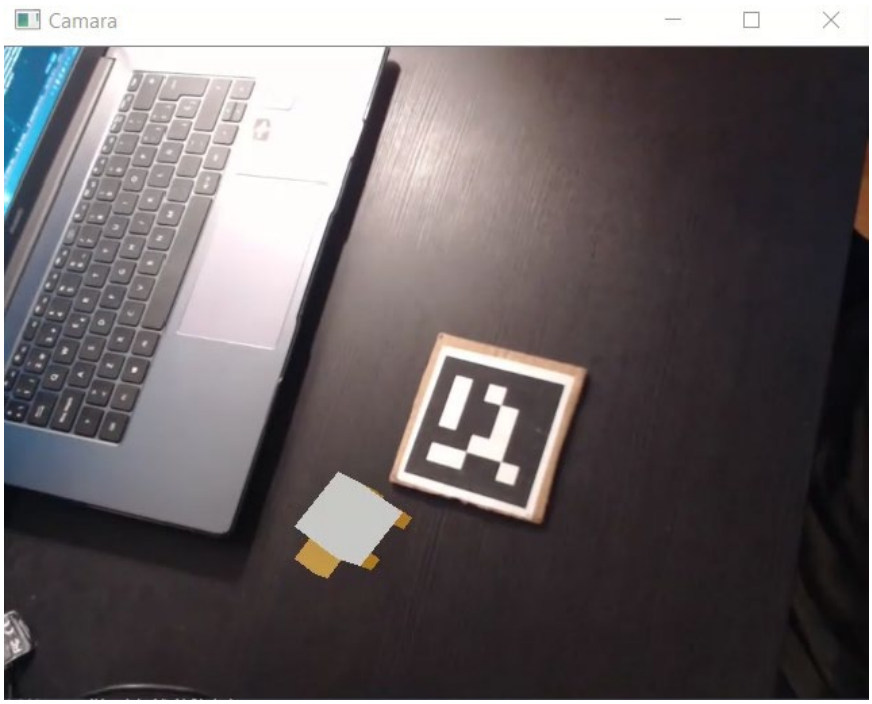

Figura 5.17: Posición final

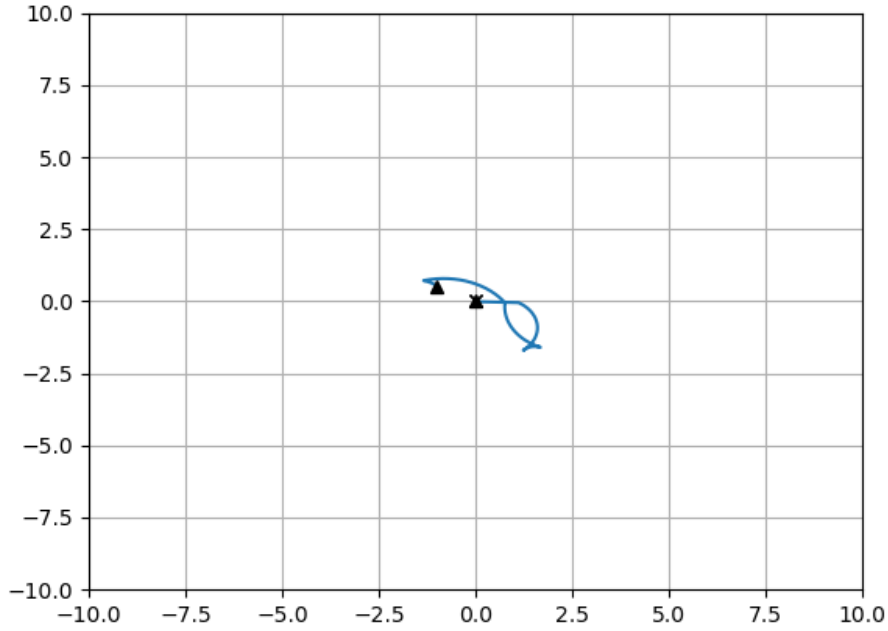

Figura 5.18: Gráfica recorrido simulación 4

5.5. Simulación 5 ovejas moviendo marcador

En esta simulación se observa cómo se mueven las ovejas a medida que se mueve el marcador ArUco, debido a que están referenciadas al origen de este. Para ello, se ha seguido un comportamiento en grupo, explicado en la sección 4.3, cuyos parámetros son los siguientes:

- Kp=1, siendo la constante de proporcionalidad.
- Kc=6
- alpha1=1
- Kr=1
- alpha2=1

Enlace de YouTube: https://youtu.be/iDsoQlozix4

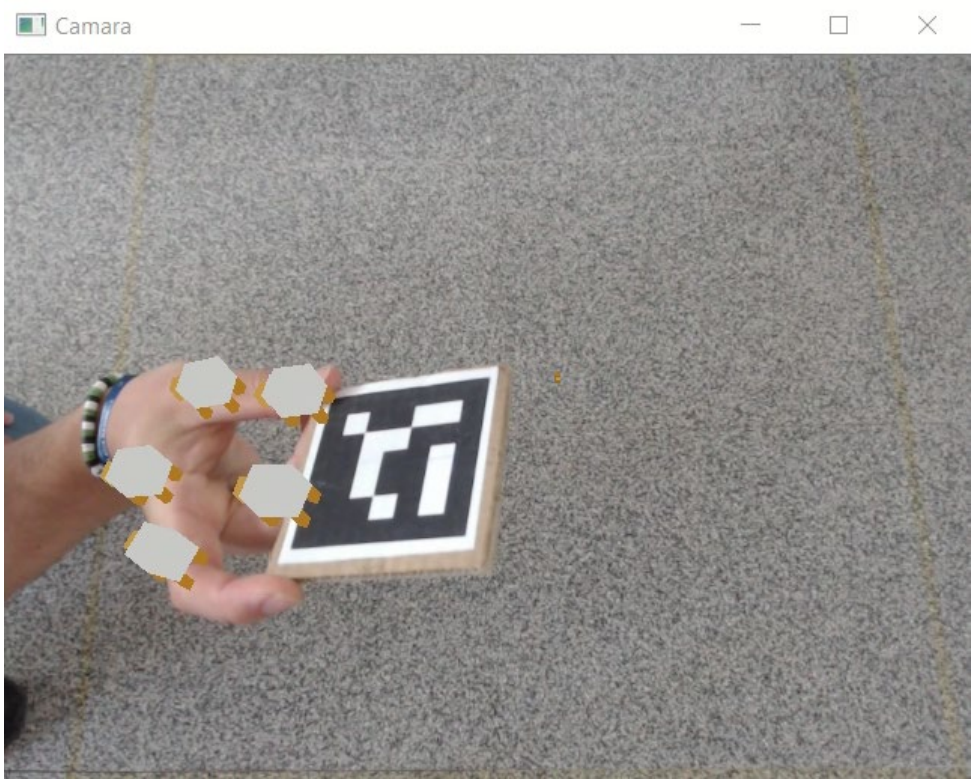

Figura 5.19: Posición inicial

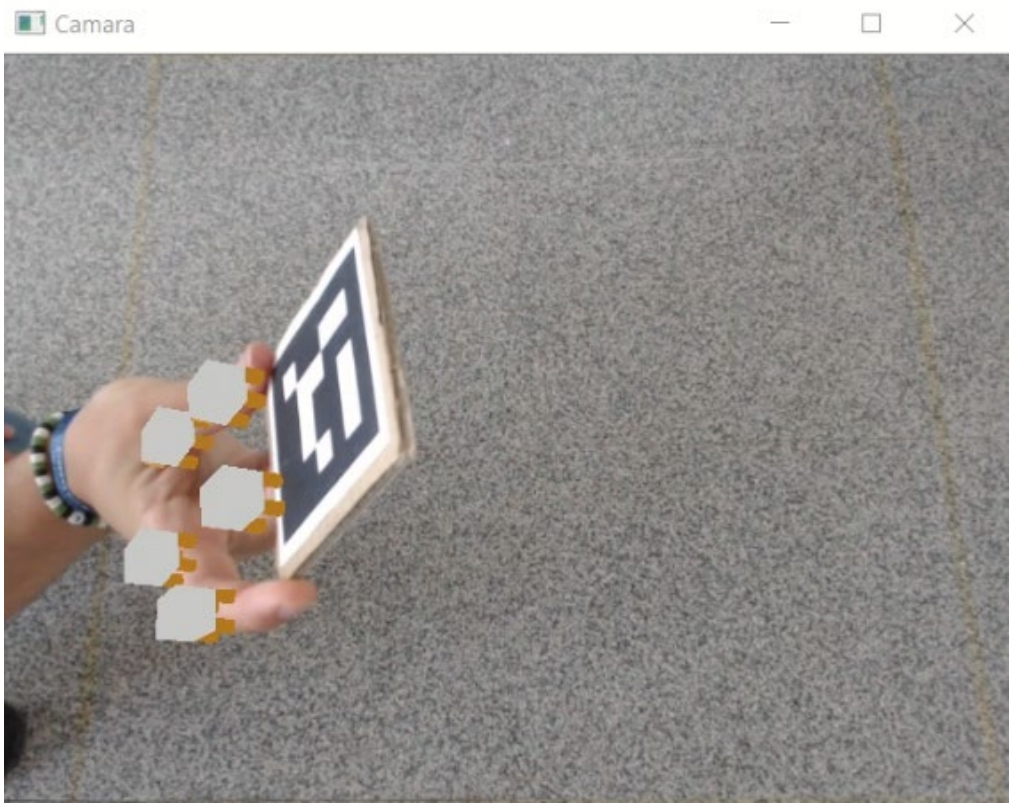

Figura 5.20: Posición intermedia

5.6. Simulación 5 ovejas cambiando parámetros

Se comprueba los parámetros de repulsión entre ovejas. De manera que introduciendo una Kr=4, la fuerza de repulsión resulta mayor con respecto al caso anterior, que se tenía una Kr=1, dando como resultado una posición final en la que las ovejas están más separadas las unas de las otras. Los parámetros definidos son los siguientes:

- Kp=1
- Kc=6
- alpha1=1
- Kr=4
- alpha2=1

Enlace de YouTube: https://youtu.be/LhOByyfKmOA

Figura 5.21: Posición inicial

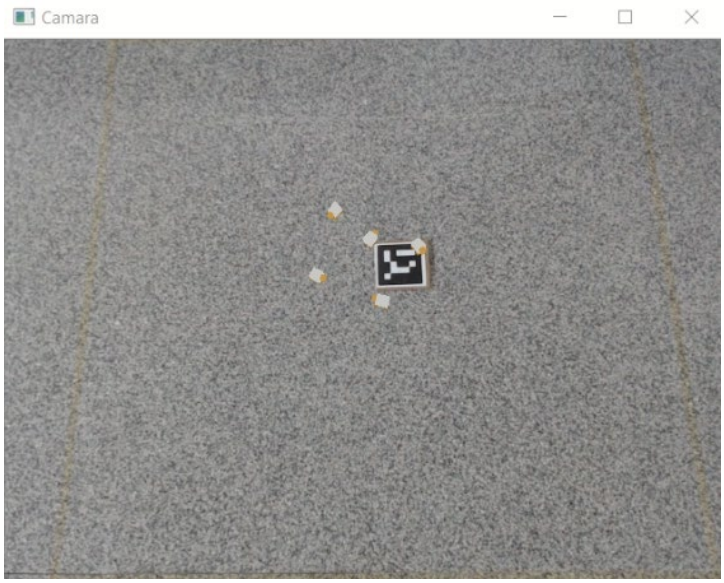

Figura 5.22: Posición final

En este caso, se ha cambiado los parámetros de manera que tengan una fuerza de atracción al centroide menor. Cómo se puede observar en el siguiente vídeo, las ovejas se desplazan al centroide con una velocidad menor que en el caso anterior. Los parámetros asignados son:

- Kp=1
- Kc=2
- alpha1=1
- Kr=1
- alpha2=1

Enlace de YouTube: https://youtu.be/ZIC4on6z_lI

5.7. Simulación de 20 ovejas sin perro

Esta simulación se ha realizado con el fin de observar cómo 20 ovejas son capaces de juntarse formando un rebaño, sin que los centros de las ovejas lleguen a colisionar entre ellos. Las ovejas se desplazan según el modelo de comportamiento en grupo implantado, asignando unas velocidades al

controlador de tipo integrador (sección 4.1.1). Para ello, hemos asignado los siguientes parámetros:

- Kp=1
- Kc=6
- alpha1=1
- Kr=1
- alpha2=1

Se observa como inicialmente aparecen en unas posiciones arbitrarias, y rápidamente se concentran en el centroide del rebaño.
Enlace de YouTube: https://youtu.be/dlswjj0Sg30

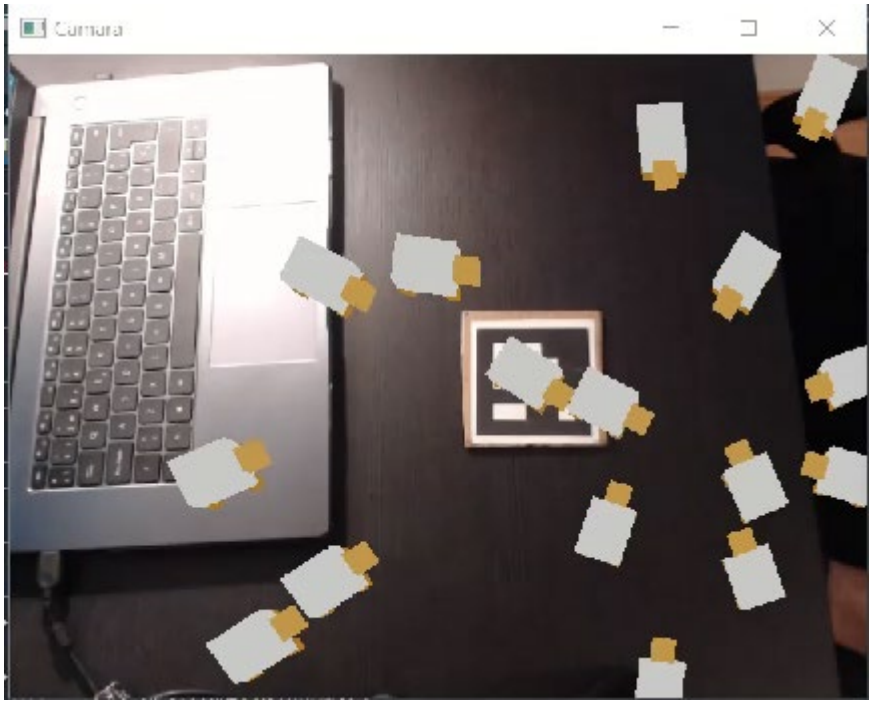

Figura 5.23: Posiciones iniciales

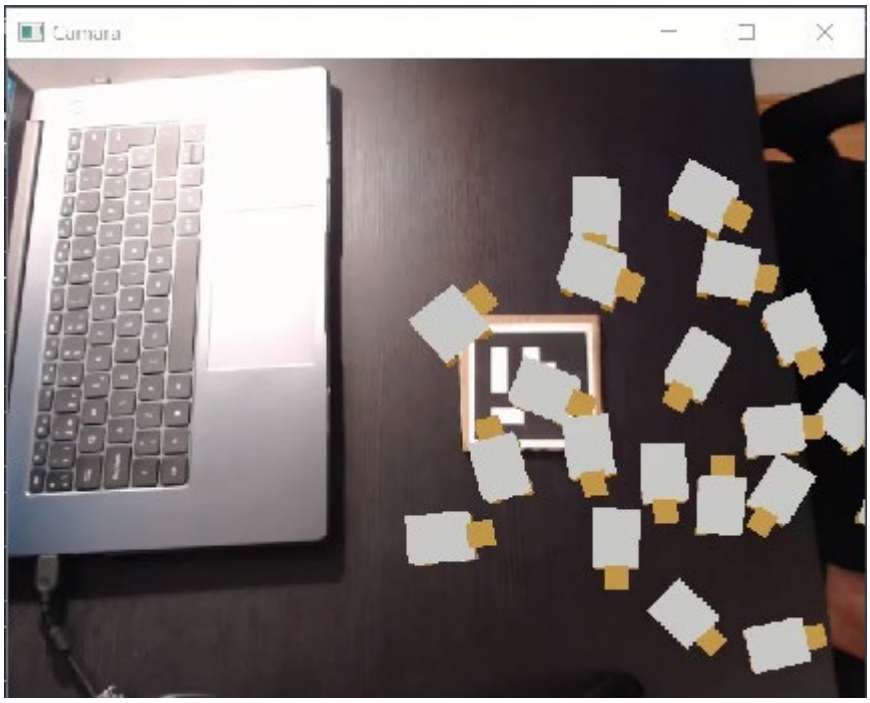

Figura 5.24: Posiciones finales

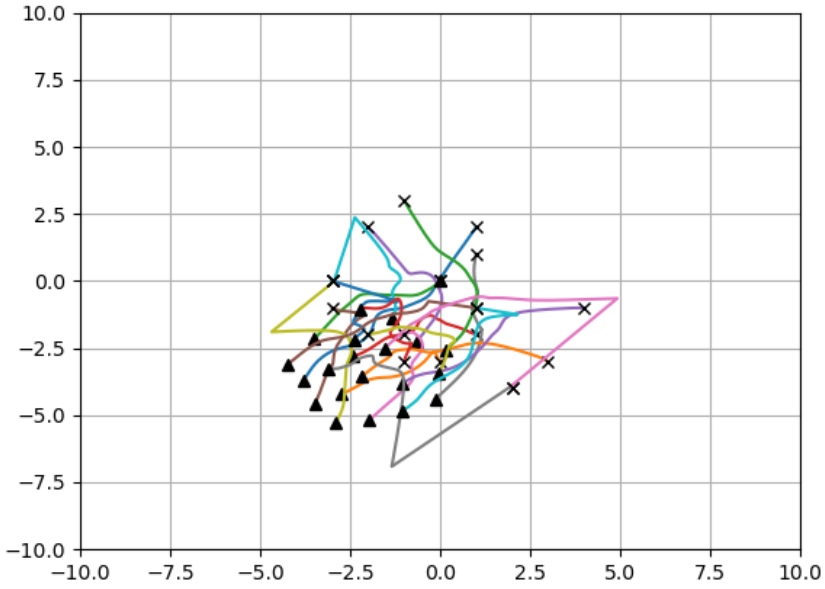

Figura 5.25: Gráfica recorrido simulación 5

5.8.Simulación de 5 ovejas con perro

En este experimento, se hace uso de un robot real, al cual hemos incorporado en su parte superior el marcador ArUco que se corresponde con el perro. De esta manera, vamos a poder realizar el movimiento del perro mediante control remoto.

La simulación se va a llevar a cabo con 5 ovejas, que deberán de reaccionar ante el acercamiento del robot, siendo éstas orientadas en la dirección contraria del perro.

Las ovejas van a estar colocadas mediante unas funciones aleatorias que le indiquen la posición X e Y, la orientación, las velocidades lineales, las velocidades angulares y el radio de giro.

Implementamos el comportamiento (sección 4.3) modificando varios parámetros. Esta simulación la hemos llevado a cabo con los siguientes valores:

- Kp=1
- Kc=6
- alpha1=1
- Kr=1
- alpha2=1
- Krr=30
- alpha3=2

Enlace de YouTube: https://youtu.be/FNE8jgwqrcc

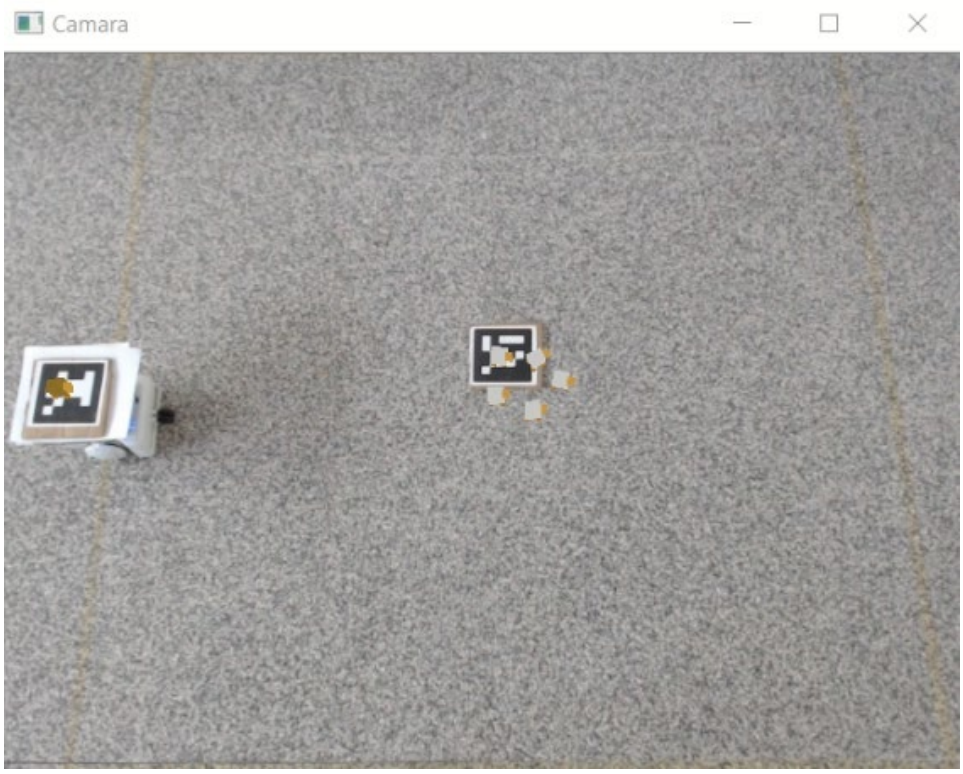

Figura 5.26: Posición inicial

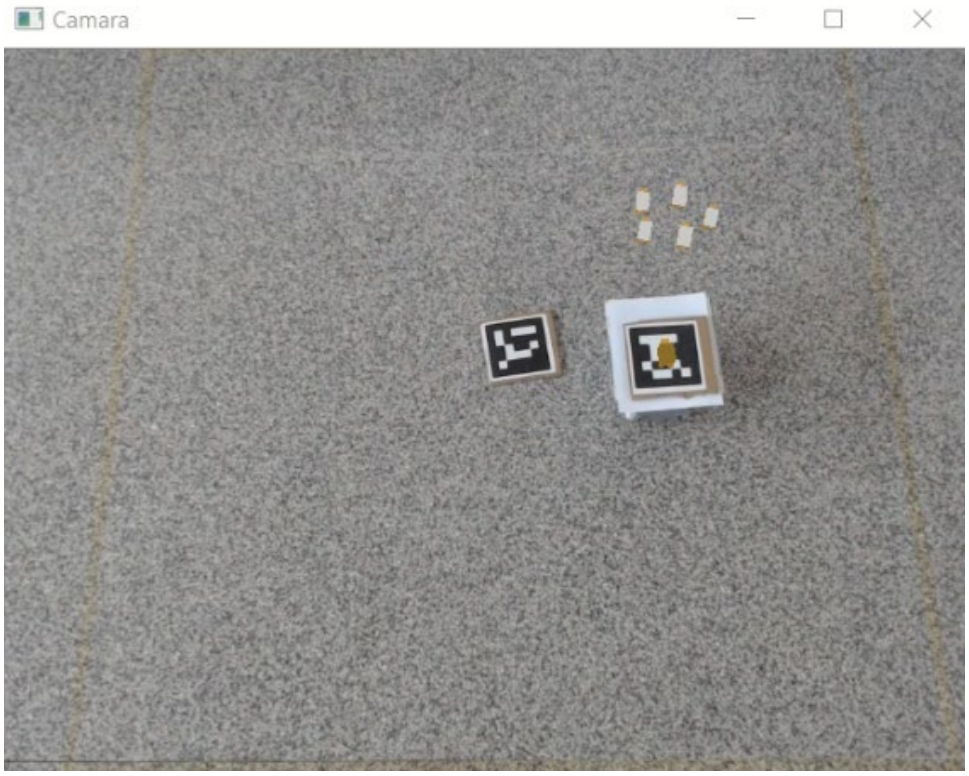

Figura 5.27: Posición intermedia

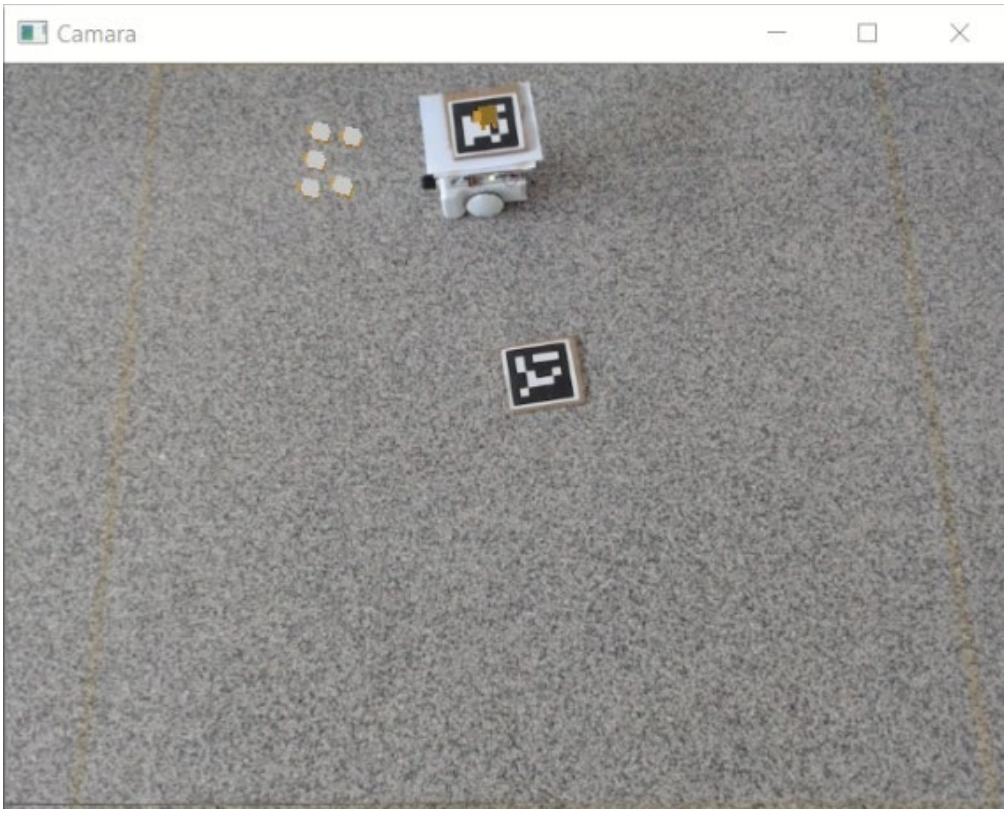

Figura 5.28: Posición final

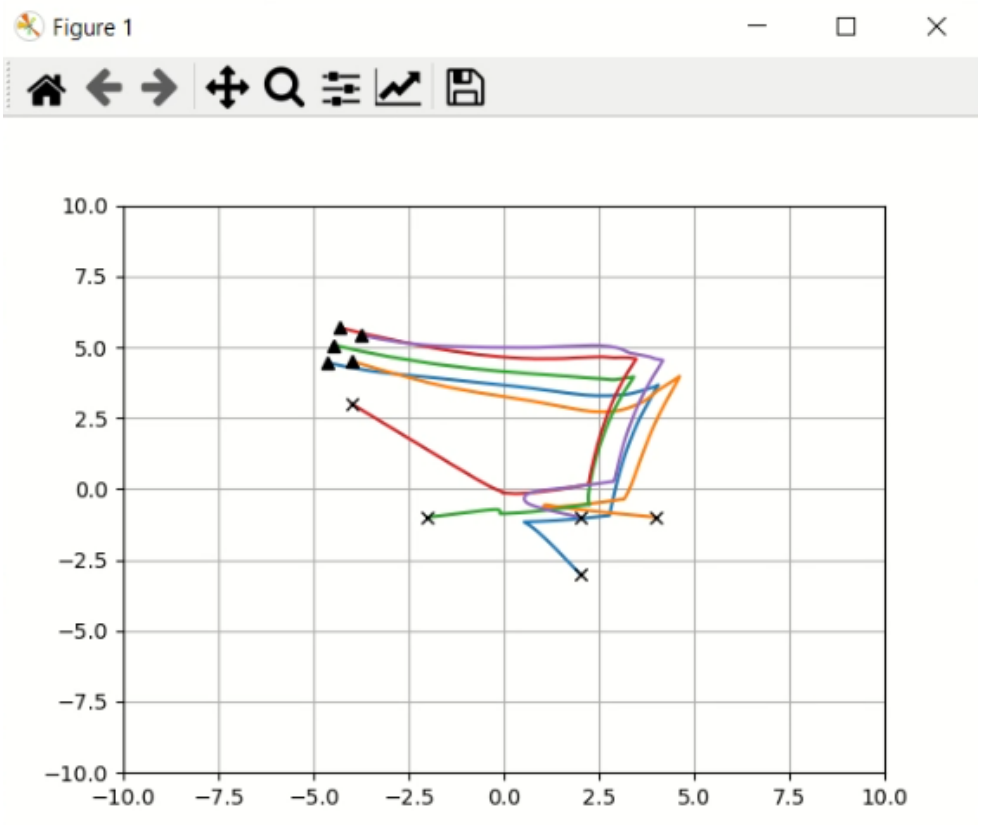

Figura 5.29: Recorrido que han seguido las ovejas

5.9.Simulación de 20 ovejas con perro

Al igual que en la simulación anterior, vamos a hacer uso de un robot para poder representar de manera más real el movimiento del perro.
En este caso, se va a realizar una simulación de 20 ovejas que reaccionen ante la presencia del perro pastor.
Las posiciones, la orientación, las velocidades lineales y angulares, y el radio de giro de las ovejas son asignadas aleatoriamente.
Implementamos el comportamiento (sección 4.3) modificando varios parámetros. Esta simulación la hemos llevado a cabo con los siguientes valores:

- Kp=1
- Kc=6
- alpha1=1
- Kr=1
- alpha2=1
- Krr=30
- alpha3=2

Enlace de Youtube: https://youtu.be/dOgpF6f6n-Q

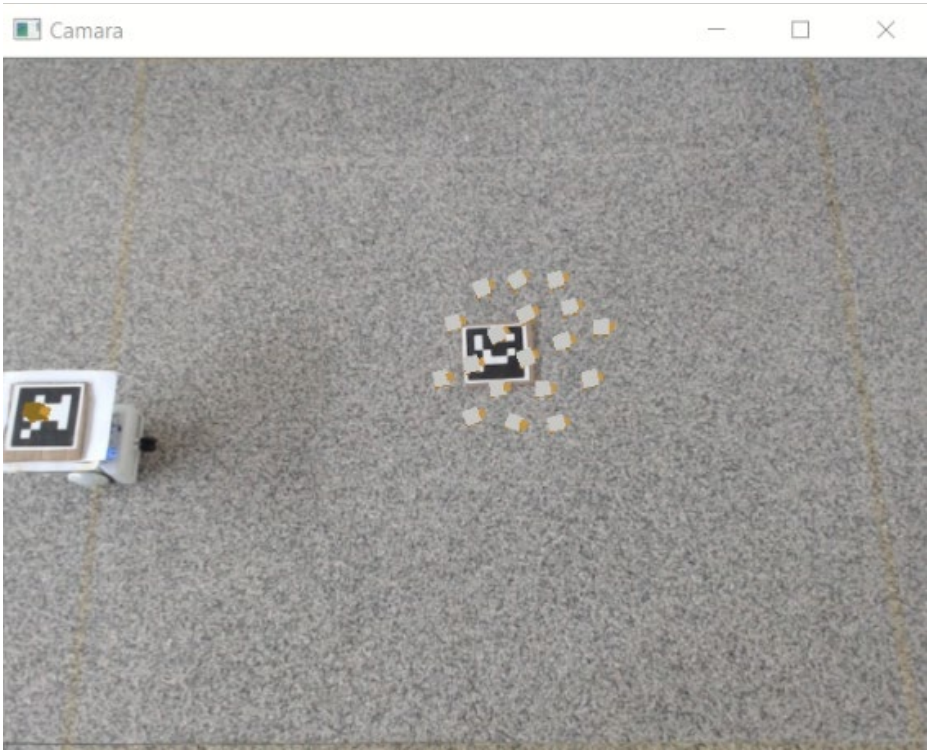

Figura 5.30: Posición inicial

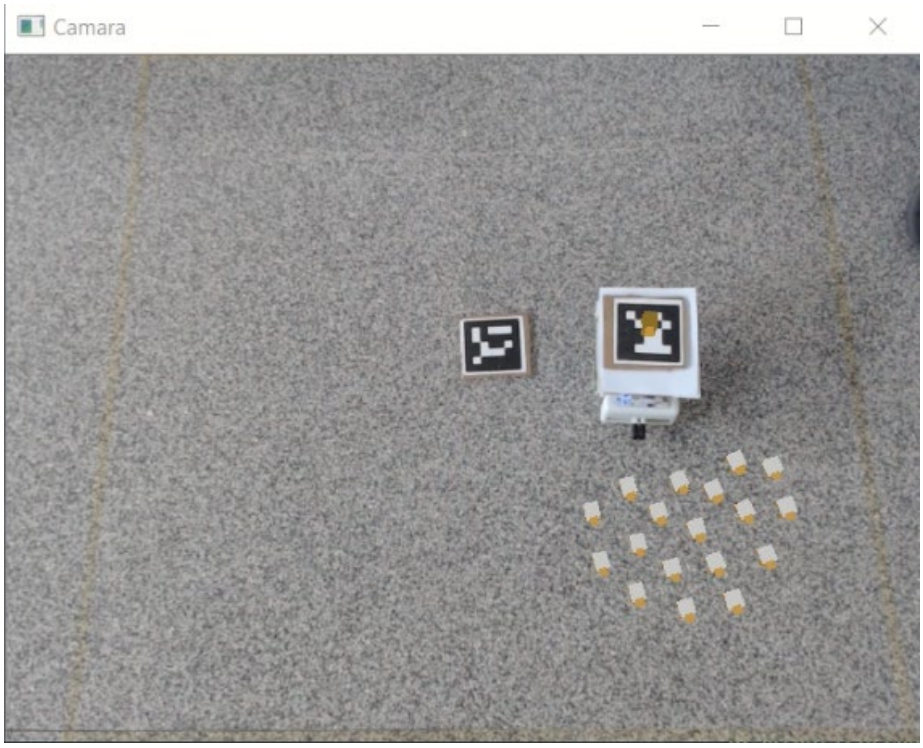

Figura 5.31: Posición intermedia

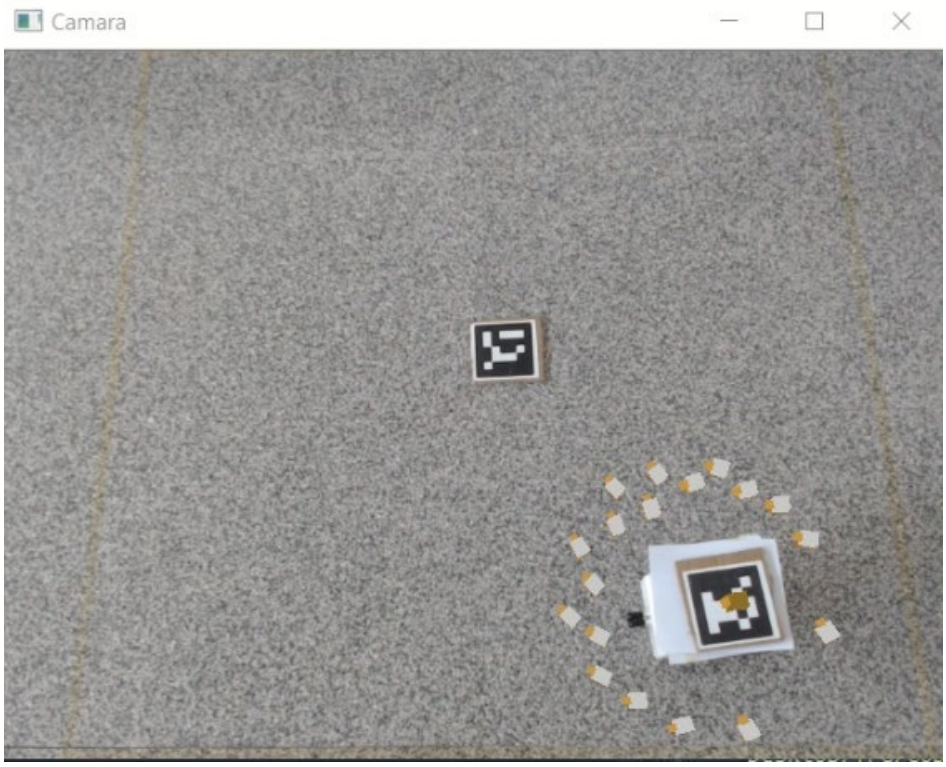

Figura 5.32: Posición final

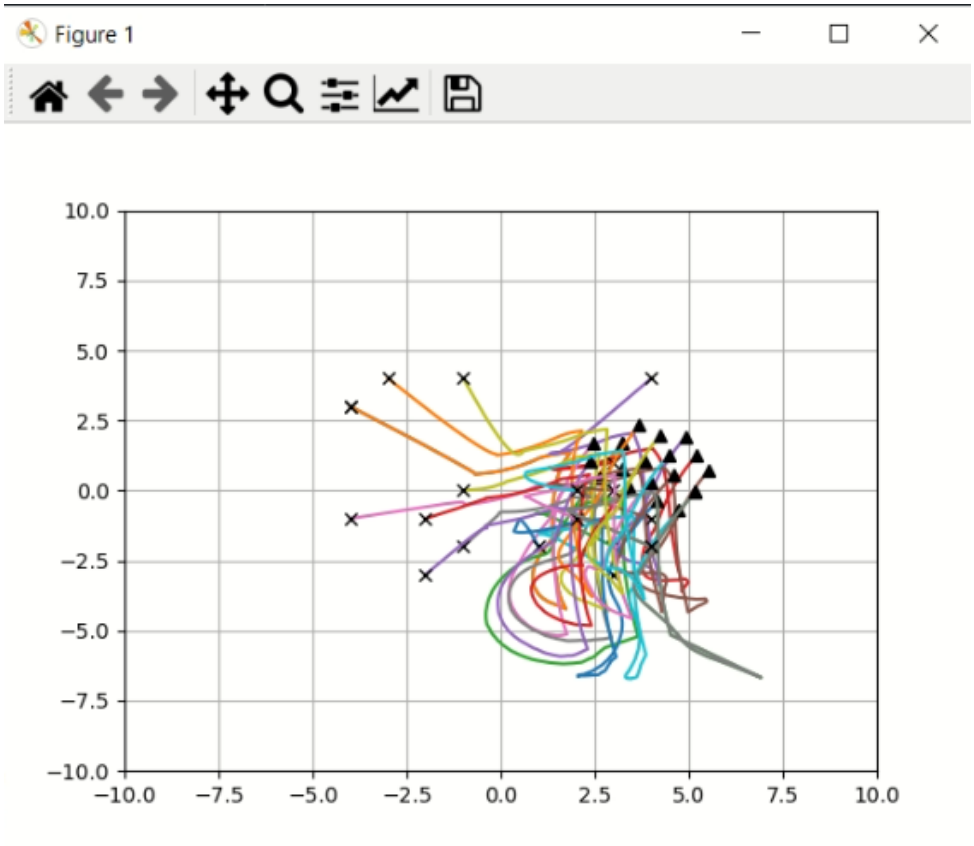

Figura 5.33: Recorrido que han seguido las ovejas

Como se ha podido comprobar, existen una serie de parámetros modificables por el usuario que hay que ajustar. Cada modificación de estos parámetros cambia la funcionalidad del programa, ya que se puede modificar el número de ovejas, las fuerzas de atracción y repulsión, las velocidades, etc. Tras realizar las pruebas convenientes, considero que unos parámetros que podrían adecuarse a un correcto funcionamiento podrían ser los siguientes:

- Kp=1
- Kc=6
- alpha1=1
- Kr=1
- alpha2=1
- Krr=30
- alpha3=2

6. Conclusiones

6.1. Conclusión

En el desarrollo de este proyecto se han alcanzado todos los objetivos propuestos inicialmente. El objetivo principal era realizar una representación virtual de un rebaño de ovejas reaccionando ante la presencia de un perro pastor. Para ello, ha sido necesario abordar una serie de objetivos específicos: se ha conseguido representar en el plano real elementos virtuales sobre marcadores ArUco, combinando técnicas de visión por computador, transformaciones 3D y proyecciones 3D-2D. Se ha llevado a cabo la calibración de la cámara. Se ha estructurado la escena utilizando matrices de transformación entre marcadores y la cámara, conociendo así las posiciones y orientaciones de cualquier elemento dibujado sobre la escena real. Se han definido elementos (ovejas, perros) en 3D mediante el uso de planos, que se proyectan posteriormente sobre la imagen adquirida por la cámara. También se ha dado solución al problema de las oclusiones, llegando a buscar la mejor solución posible para resolver esta cuestión. En este caso se hizo con la idea de pintar primero lo que esté más alejado de la cámara, para después ir superponiendo elementos que fueran tapando los que estén detrás.

Por otro lado, se han desarrollado diferentes métodos de movimiento individual para cada elemento (oveja) del swarm. Se han analizado e implementado comportamientos en grupo (swarm, herd) que, basados en términos de atracción y repulsión, generan las velocidades individuales que se aplican sobre cada oveja. Para ello, se ha tenido que adquirir los conocimientos necesarios sobre realidad aumentada y visión por computador, así como un gran manejo de programación en Python.

El resultado obtenido tras la realización de este trabajo ha sido satisfactorio, al cumplimentarse todos los objetivos propuestos inicialmente. Las ovejas se mueven independientemente las unas de las otras, a la vez que mantienen una relación de cercanía, simulando el rebaño que componen. Incorporando un perro al plano, las ovejas son capaces de tener una acción repulsiva ante él, de tal forma que el perro pueda guiarlas y orientarlas hacia el destino que se desee.

Así pues, el proyecto ha alcanzado un alto grado de eficacia al aproximarse estos resultados a los susodichos objetivos iniciales.

6.2.Conclusión personal

Personalmente estoy muy satisfecho con el trabajo que he desarrollado. La decisión que tomé en cuanto a la realización de este trabajo vino influenciada por el hecho de haber cursado, en este mismo año, la asignatura de Robots Autónomos. En esta materia se realizaron unas prácticas, conjuntamente con un trabajo final, en las que el objetivo último era programar un robot en Python capaz de realizar un circuito. Este trabajo me gustó mucho ya que la programación y los resultados de su aplicación en el mundo real me parecen muy interesantes.

Por eso, con algunos conocimientos que aprendí de visión por computador, me puse en contacto con mis directores del TFG, que me proporcionaron la idea de este proyecto, que me ha permitido profundizar en conceptos de modelos de movimientos para robots autónomos y técnicas de visión por computador y realidad aumentada. También me ha permitido realizar una aproximación en sistemas multi-robot y multi-agente, afianzando así conocimientos de programación.

He aprendido mucho acerca de la realidad aumentada y de cómo es posible proyectar elementos virtuales en el mundo real. Además, estoy satisfecho por haber conseguido modificar los elementos que lo han requerido, planificar las actuaciones necesarias y resolver los problemas que han ido surgiendo a lo lardo del proyecto a fin de alcanzar la aproximación a los resultados propuestos en los objetivos iniciales.

6.3. Trabajo futuro

Como se ha podido leer en el capítulo 2, la realidad aumentada puede ser muy útil en el mundo industrial, educativo, en la ciencia médica, etc.

Este trabajo puede ser de gran ayuda para cualquier persona que esté interesada en el mundo de la realidad aumentada, ya que supone el desarrollo del paso inicial que hay que dar para descubrir este mundo. Como bien se ha podido estudiar a lo largo de este proyecto, se ha desarrollado un trabajo de proyección virtual de elementos sobre el mundo real desde cero, partiendo desde el reconocimiento inicial de un marcador ArUco para disponer así del plano donde se va a representar. También, dado que se ha empezado con elementos simples como un cubo o unas ovejas, se podría seguir desarrollando objetos con una forma más compleja. Por otro lado, el hecho de dar una velocidad a los elementos, permite ver el estudio inicial por el cual se consigue mover cada objeto, teniendo así la base para dotarlos de un movimiento con mayor complejidad si así lo requiriese algún proyecto específico.

Aunque este trabajo se haya centrado en la representación mediante realidad aumentada de ovejas y perros, es a tener en cuenta su utilidad como base para la aplicación en muchos ámbitos. Por ejemplo, en el caso del mobiliario de Ikea, se podría representar un mueble de la misma forma que se ha representado una oveja, y poder posicionarlo en la estancia del hogar que se desee, así como muchas otras aplicaciones. [4]

7. Bibliografía

[1] Alegría Blázquez Sevilla, Universidad Politécnica de Madrid Gabinete de Tele-Educación, 2017. “Realidad Aumentada en Educación”.

[2] Ligdi Gonzalez, AprendeIA, 2018. “Introducción al IDE Spyder”

[3] Richard Vaughan, Neil Sumpter, Andy Frost y Stephen Cameron. “Experiments in automatic flock control”.

[4] Cosmos, Editor Xataka Android, 2018. Ikea Place, su aplicación de realidad aumentada para decorar tu casa llega a los móviles Android con ARCore. URL: https://www.xatakandroid.com/aplicaciones-android/ikea-place-su-aplicacion-de-realidad-aumentada-para-decorar-tu-casa-llega-a-los-moviles-android-con-arcore

[5] Equipo OpenCV, junio 2000. “OpenCV Tutorials”. URL: https://docs.opencv.org/4.5.2/index.html

[6] Reynolds, Craig (1987). Flocks, herds and schools: A distributed behavioral model. SIGGRAPH '87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques. Association for Computing Machinery. pp. 25–
34. CiteSeerX 10.1.1.103.7187. doi:10.1145/37401.37406. ISBN 978-0-89791-227-3. S2CID 546350.

[7] Enrique Dans, 2015. “Microsoft HoloLens y la generación de ecosisteas de innovación”. URL: https://www.enriquedans.com/2015/07/microsoft-hololens-y-la-generacion-de-ecosistemas-de-innovacion.html

[8] Oleg Kalachev, 2018. “ArUco markers generator”. URL: https://chev.me/

[9] Linda G. Shapiro and George C. Stockman (2001). Computer Vision. Prentice Hall. ISBN 0-13-030796-3.

[10] Logitech. URL: https://www.logitech.com/es-mx/products/webcams/c920-pro-hd-webcam.960-000764.html

[11] Garda Muhammad, 2016. URL: https://gardamuhammad.wordpress.com/analytical-photogrammetry-and-remote-sensing/

[12] Juan Antonio Pascual Estapé, 2020. “El perro pastor robot que no gusta a los pastores”. URL: https://computerhoy.com/noticias/tecnologia/perro-pastor-robot-no-gusta-pastores-654443