Desarrollo de un sistema para detectar la posición y orientación de los objetos en el espacio

Autores:
Emiliano Churruca
Hernán Guzmán

Director:
Dr. Juan Miguel Santos

Instituto de Tecnología e Ingeniería
Universidad Nacional de Hurlingham

Tesis para obtener el Título de
Licenciado en Informática

Marzo 2024

Resumen

La creciente demanda de personal calificado en soldadura en Argentina, especialmente en sectores como la industria metalúrgica, ha generado la necesidad de mejorar los métodos de capacitación. Está extensamente demostrado que la incorporación de simuladores de soldadura en los programas de formación incrementan la tasa de certificaciones, reducen el tiempo de aprendizaje y los costos asociados, además de disminuir riesgos para los aprendices.
La Universidad Nacional de Hurlingham, desde el año 2023, impulsa y financia el desarrollo de este tipo de simuladores a través del proyecto PIUNAHUR "Desarrollo de un Prototipo de Simulador de Soldadura para Capacitación en Argentina". Este proyecto usa la técnica de realidad aumentada (RA) para la simulación, superando las limitaciones de los simuladores basados exclusivamente en realidad virtual.
Un requisito que impone la RA es el rastreo de la posición y orientación de los objetos en la escena 3D. Para ese fin, el sistema de simulación hace uso de marcadores fiduciarios. Sin embargo, su implementación enfrenta desafíos técnicos, como la elección de cámaras adecuadas, la gestión del ruido en las imágenes y la optimización del diseño de los marcadores para una identificación precisa.
Esta tesis presenta un enfoque integral para abordar estos desafíos, combinando análisis experimental, algoritmos y metodologías. Se revisa el estado del arte en detección de objetos en RA, se detallan los materiales y métodos utilizados y se describen los experimentos realizados, junto con los resultados obtenidos. Finalmente, se presentan las conclusiones derivadas de este trabajo, que buscan contribuir a la elección de los marcadores adecuados para la simulación de soldadura mediante RA .

Palabras claves: Realidad Aumentada, marcadores fiduciarios, Aruco, Simulación de soldadura

Índice general

1. Introducción **8**

2. Estado del arte **12**

3. Materiales y métodos **20**
3.1. Parámetros de una cámara 20
3.1.1. Parámetros intrínsecos 21
3.1.1.1. Distorsión radial 22
3.1.1.2. Distorsión tangencial 23
3.1.2. Parámetros extrínsecos 24
3.2. Funcionamiento de una cámara 25
3.3. Calibración 28
3.4. Marcadores 33
3.5. Diccionario 33
3.5.1. Generación de marcadores y diccionarios 34
3.5.2. Generación automática de diccionarios 34
3.5.2.1. Descripción del algoritmo 35
3.5.2.2. Generación del marcador 36
3.6. Corrección del error 38

4. Experimentos y resultados **42**
4.1. Identificación de marcadores 42
4.2. Algoritmo y distancia de Hamming 44
4.3. Elección del tamaño de marcador 46
4.4. Oclusión y análisis de error 50

5. Conclusiones **60**
5.1. Artículos derivados de este trabajo 61
5.2. Trabajos futuros 61
5.3. Conclusiones finales 61

Índice de figuras

2.1. La figura de la izquierda sin realidad aumentada, en el centro con realidad aumentada y a la derecha el equipamiento que tiene el tractor (extraído de [10]). 12
2.2. Superposición de una fotografía antigua de un espacio urbano luego de haber reconocido en la imagen obtenida por la cámara de un celular el punto de control donde debe realizarse la superposición (extraída de [11]). 13
2.3. En a. y b. se observan etiquetas superpuestas sobre regiones planas de la imagen. En b. se muestra un detalle de a. y en d. un detalle de c. con una posición relativa de la cámara distinta [12]. 13
2.4. Ejemplo de realidad aumentada utilizando gafas con sensores de movimiento durante una operación de ensamblado. Arriba a la derecha se puede observar un detalle del visor del usuario donde recibe una indicación del paso a ser realizado (extraída de [13]). 14
2.5. Ejemplo de marcador de ArtoolKit (extraída de [6]). 15
2.6. Ejemplo de marcador de ARTag (extraída de [7]). En la región central del marcador se pueden codificar diferentes patrones definiendo los cuadraditos negros y blancos (36 organizados en 6 columnas y 6 filas). 15
2.7. Ejemplo de marcador InterSence [14]. 16
2.8. Ejemplo de marcadores de ReactiVision con su correspondiente gráfico de adyacencia a su derecha (extraída de [15]). 16
2.9. Ejemplo de marcadores de ReactiVision con su correspondiente gráfico de adyacencia a su derecha (Identificación de un marcador usando el método Matrix. Se puede observar la superposición de los ejes de referencia del marcador y las aristas de un cubo cuya base es el propio marcador (extraída de [16]). 16
2.10. Imagen con ejemplo de marcadores respecto a la oclusión, se observa en la imagen del medio que ocluyendo mínimamente, ya no lo detecta (extraída de [18]). 17
2.11. Ejemplo de marcadores visuales Aruco (extraído de [8]). 18

3.1. Esquema de funcionamiento de una cámara. f es la distancia focal y Z la distancia del punto a la cámara (extraída de [20]). 21

3.2. Modelo de cámara. (X_w, Y_w, Z_w) son las coordenadas de un punto en el espacio respecto al sistema de coordenadas del mundo. Los ejes X_c, Y_c y Z_c son los ejes del sistema de coordenadas de la cámara. (c_x, c_y) expresa el centro de la imagen por el cual pasa el eje óptico. (u,v) son las coordenadas del píxel, en el plano de la imagen, donde se proyecta (X_w, Y_w, Z_w) (extraído de [20]). 21
3.3. Distorsión radial observada sobre el plano de la imagen producida por una lente a partir de un objeto rectangular en el espacio (extraída de [20]). . 22
3.4. Deformaciones en la imagen de acuerdo al tipo de distorsión radial. (extraída de [20]). 23
3.5. Imperfección del montaje del sensor lo cual produce distorsión tangencial (extraída de [20]). 23
3.6. Rotación en sentido horario respecto a cada uno de los ejes cartesianos (extraída de [20]). 24
3.7. Esquema de bloques del funcionamiento de una cámara desde el punto de vista del cambio de coordenadas en cada bloque. (extraída de [20]). 26
3.8. Esquema de la proyección de un punto en el mundo sobre el plano de la imagen. 27
3.9. Captura de pantalla que se visualiza ni bien ejecutado el comando para la calibración. 29
3.10. Captura de pantalla con el error luego de capturar 5 imágenes. 30
3.11. Captura de pantalla con el mensaje luego de finalizar el proceso de calibración. 30
3.12. Captura de pantalla del archivo de salida donde se puede observar la matriz de calibración y los coeficientes de distorsión. Notar que el centro no se encuentra en el punto $(320, 240)$ sino en $(374, 235)$. 31
3.13. Tablero utilizado para la calibración usando las funciones de openCV (extraído de [22]). 31
3.14. Captura de pantalla con el patrón dibujado. 32
3.15. En el marcador se puede ver el borde negro, es decir, el contorno está formado por cuadraditos negros, la matriz interna binara es la combinación de colores (blanco y negro) que están por dentro de dicho borde, en este caso, es de 6x6. 33
3.16. Imagen obtenida [8] luego de aplicar umbralización y filtro de Canny [23]. 38
3.17. Imagen con la extracción y filtrado de contornos candidatos (extraída de [8]). 39
3.18. Imagen una vez filtrados los contornos quedando solo los contornos externos de los marcadores (extraído de [8]). 39
3.19. La imagen resultante está umbralizada usando el método Otsu´s [26], esto proporciona el valor de umbral óptimo dado que la imagen es bimodal por lo que la imagen binarizada se divide en una cuadrícula y a cada elemento se le asigna el valor cero o uno dependiendo de los valores de la mayoría de los píxeles. 40

4.1. Contenido del archivo de salida de la calibración (el de menor error de 5 repeticiones) usando el programa para tal fin provisto por la librería Aruco [21]. 43
4.2. Contenido del archivo de salida de la calibración (el de menor error de 5 repeticiones) usando el programa para tal fin provisto por la librería openCV [22] 44
4.3. Captura de pantalla del tablero 1 con los resultados de los marcadores $4x4$ a una distancia de 30 cm. 47
4.4. Captura de pantalla del tablero 1 con los resultados de los marcadores $4x4$ a una distancia de 60 cm. 47
4.5. Captura de pantalla del tablero 2 (marcadores $4x4$ a una distancia de 30 cm). 48
4.6. Captura de pantalla del tablero 2 (marcadores $4x4$ a una distancia de 60 cm). 48
4.7. Captura de pantalla del tablero 3 con los resultados de los marcadores $5x5$ a una distancia de 30 cm. 48
4.8. Captura de pantalla del tablero 3 con los resultados de los marcadores $5x5$ a una distancia de 60 cm. 49
4.9. Captura de pantalla del tablero 4 con los resultados de los marcadores $5x5$ a una distancia de 30 cm. 49
4.10. Captura de pantalla del tablero 4 con los resultados de los marcadores $5x5$ a una distancia de 60 cm. 49
4.11. Imagen del tablero de 5 marcadores con las medidas hacia los bordes utilizado para el experimento. 50
4.12. Captura de pantalla con el rectángulo dibujado. 51
4.13. Captura de pantalla con el rectángulo dibujado. 52
4.14. Captura de pantalla con el rectángulo dibujado. 52
4.15. Captura de pantalla con el rectángulo dibujado. 53
4.16. Distancia en píxeles vs cantidad de marcadores en los tableros. El trazo azul corresponde a la distancia máxima respecto al promedio y el trazo rojo corresponde a la desviación estándar. 54
4.17. Distancia en píxeles vs cantidad de marcadores en los tableros. El trazo azul corresponde a la distancia máxima respecto al promedio y el trazo rojo corresponde a la desviación estándar. 55
4.18. Distancia en píxeles vs cantidad de marcadores en los tableros. El trazo azul corresponde a la distancia máxima respecto al promedio y el trazo rojo corresponde a la desviación estándar. 55
4.19. Distancia en píxeles vs cantidad de marcadores en los tableros. El trazo azul corresponde a la distancia máxima respecto al promedio y el trazo rojo corresponde a la desviación estándar. 56
4.20. Imagen de los tablero con 5, 10, 20 y 40 marcadores. Pese a la oclusión de algunos de los marcadores, la traza del rectángulo sigue al borde del tablero. 56

4.21. Diferencia al dibujar el rectángulo del borde del tablero usando el conjunto de marcadores que tienen el eje z correcto (izquierda) y usando todos los marcadores, independientemente tengan el eje z calculado correctamente o no (derecha). 59

Índice de tablas

4.1. Registro de creación de diccionarios 3x3. 45
4.2. Registro de creación de diccionarios 4x4. 45
4.3. Registro de creación de diccionarios 5x5. 45
4.4. Registro de creación de diccionarios 6x6. 46
4.5. En la tabla se muestra un resumen del experimento con el tablero de 5 marcadores, cada línea es el resultado de tomar 500 imágenes mientras que la última, es el promedio de todas los registros de la tabla. Tanto la DM como la DE están expresadas en píxeles. 51
4.6. En la tabla se muestra un resumen de la experiencia realizada con el tablero de 10 marcadores. Cada línea es el resultado de tomar 500 imágenes mientras que la última, es el promedio de todas los registros de la tabla. Tanto la DM como la DE están expresadas en píxeles. 52
4.7. En la tabla se muestra un resumen de la experiencia con el tablero de 20 marcadores. Cada línea es el resultado de tomar 500 imágenes mientras que la última, es el promedio de todas los registros de la tabla. 53
4.8. En la tabla se muestra un resumen de la experiencia con el tablero de 40 marcadores. Cada línea es el resultado de tomar 500 imágenes mientras que la última, es el promedio de todas los registros de la tabla. Tanto la DM como la DE están expresadas en píxeles. 53
4.9. En la tabla se muestra un resumen del experimento con los 4 tableros a una distancia de 30 cm entre la cámara y los tableros, cada línea es el resultado promedio de todos los registros de cada tablero. Tanto la DM como la DE están expresadas en píxeles. 54

Capítulo 1

Introducción

En la Argentina existe una demanda creciente de personal capacitado para realizar tareas de soldadura en diversas áreas de la industria metalúrgica (industria oil&gas, industria naval, industria nuclear, industria ferroviaria, industria aeroespacial e industrias metal-mecánica en general). Dicha demanda no es completamente satisfecha por la falta de técnicos y profesionales capacitados en diversos tipos de soldadura y, además la previsión, al momento de iniciarse este trabajo de tesis, anunciaba que dicha demanda va a ser sostenida.

Una forma de enfrentar esta dificultad es incorporando simuladores en los cursos de capacitación de soldadura incrementando las certificaciones de aprendices y reduciendo los tiempos de aprendizaje. Un estudio hecho por Lavrentieva y colaboradores [1], a partir de las estimaciones de los principales fabricantes de simuladores de soldadura, reporta que el uso de este tipo de dispositivos permite certificar más soldadores (un 34 % más), reducir el período de aprendizaje en un 56 % y reducir el costo de infraestructura en un 68 %, disminuir las emisiones de CO2, reducir el costo en consumibles y disminuir notablemente los riesgos que corre el aprendiz en un 84 %. Algunas de estas estadísticas fueron validadas por el trabajo de Stone y colaboradores [2] donde un grupo de 22 participantes fueron entrenados usando métodos tradicionales (el 50 % del grupo) y usando métodos asistidos por simuladores (el restante 50 %). Los resultados mostraron calificaciones significativamente más altas por parte del grupo que había tenido asistencia por el simulador durante el aprendizaje contra el grupo que recibió una formación tradicional.

Ya hace casi dos décadas, que este tipo de simuladores están presentes en industrias e instituciones de capacitación de todo el mundo y fueron introducidos, a lo largo del tiempo, por fabricantes de diversas empresas como [3], [4] y [5], entre otras. Sin embargo, escalar la estrategia de incorporar simuladores en los centros de capacitación colisiona con los costos del equipamiento (los equipos, en Argentina, son importados y su costo unitario es de aproximadamente 50000 u$s). Un ejemplo de esto se puede observar en los cursos de formación abierta que la Universidad Nacional de Hurlingham (UNaHur) ofrece en esta temática a la comunidad. En dichos cursos se utiliza un simulador de sol-

dadura que ha sido importado pero dado su costo no es factible adquirir más unidades en la cantidad deseada.

En este contexto, la universidad ha estado impulsando y financiando el proyecto "Desarrollo de un prototipo de simulador de soldadura para capacitación en la técnica de soldado, en Argentina, para instituciones de formación técnica, tecnicaturas e ingenierías que incluyan cursos o materias de capacitación en soldadura".

Dicho proyecto hace uso de la realidad aumentada (RA) para el proceso de simulación. Los simuladores que usan RA tienen como ventaja, sobre los que usan exclusivamente realidad virtual (RV), que permiten al usuario, tener una percepción completa de cualquier tipo de escenario, cuestión que limita a aquellos que usan RV ya que se restringen a un conjunto acotado de situaciones previamente establecidas.

Los sistemas que usan RA, sobreimprimen objetos virtuales en la escena. Una forma de alcanzar esto es conociendo la pose (posición y orientación) de objetos 3D en el espacio y, a partir de ahí, se puede sobre imprimir objetos virtuales que intervienen en la simulación. En el caso de los simuladores de soldadura, es menester conocer la pose de la probeta (donde se deposita el cordón de soldadura) y la torcha (la pistola del soldador). El estado del arte en este tipo de sistemas reconoce el uso de marcadores fiduciarios como la forma más eficiente y eficaz para reconocer la pose de objetos en el espacio 3D.

Los marcadores fiduciarios son patrones gráficos con códigos específicos que se utilizan como puntos de referencia para detectar y rastrear la posición y orientación de los objetos en la escena. Son imágenes bidimensionales con diseños únicos y estructuras que pueden ser fácilmente identificadas por sistemas de visión por computadora.

Existen diversidad de marcadores como ArtoolKit [6], ArTag [7] y Aruco [8], entre otros. Sin embargo, la implementación de un sistema de marcadores para reconocer la pose de los objetos en la escena requiere contemplar una serie de problemas que aparecen durante su uso.
En primer lugar, estos marcadores se identifican con sistemas de visión por computadora lo cual conduce a tener que responder las siguientes preguntas:

- ¿qué tipo de cámara se debe utilizar?,
- ¿cuál es la resolución mínima que debe tener el CCD de la cámara?,
- ¿qué parámetros se deben conocer de una cámara para obtener la pose de un marcador?, y
- ¿cómo se pueden hallar dichos parámetros?.

Además, la adquisición de imágenes por parte de una cámara está afectada por el ruido. Por ejemplo, si preservamos la cámara y un marcador cuadrado fijos, la identificación

de los bordes del marcador puede variar de cuadro en cuadro del video, en general, de manera imperceptible para el ojo humano. Esto puede producir pequeños cambios en la estimación de la pose que puede afectar la simulación alterando la sensación de inmersión que debe tener un sistema basado en realidad aumentada. Luego, surge la pregunta:

- ¿cómo tratar con el problema del ruido en las imágenes?

En segundo lugar, los sistemas de marcadores varían de a uno a otro dependiendo de varias características. Una de ellas es el reconocimiento de la imagen dentro del marcador. Esta imagen no solo es utilizada para identificar al marcador sino que también, en algunos casos, para determinar la orientación del mismo. Una ventaja de los sistemas de marcadores cuya imagen es una cuadrícula de bits, es la posibilidad de caracterizar un sistema de corrección de errores durante su identificación basado en la distancia de Hamming entre las representaciones binarias de dichas cuadrículas. Cuanto menos bits tenga un marcador para su codificación más grande el área de cada cuadradito en la cuadrícula del marcador y más probable que el sistema de visión lo identifique correctamente. Pero cuantos menos bits (cuadraditos en la cuadrícula) tengamos para codificar un marcador, menos marcadores distintos podemos tener (considerando que entre cualquier par de marcadores la distancia de Hamming entre ellos debe ser lo mayor posible). Entonces,

- ¿cómo encontrar una relación adecuada entre la cantidad de marcadores que necesitamos que estén presentes en la escena y la cantidad de bits que cada uno tenga para que sean correctamente identificados?, y

- ¿que tamaño (por ejemplo, en centímetros) debe tener un marcador dada la distancia que un marcador puede estar de la cámara?

Finalmente, la forma de determinar la rotación y traslación que tiene el sistema de coordenadas del marcador respecto al sistema de coordenadas de la cámara es mediante método de estimación de pose [9]. Estos métodos se ven afectados por una ambiguedad que ocurre bajo determinadas condiciones de trabajo (ubicación de la cámara respecto al marcador) que se traduce como una inversión del eje z en el sistema de coordenadas del marcador. Entonces,

- ¿cómo tratar con este inconveniente?.

En este manuscrito se describe la labor realizada para poder responder estas preguntas usando, según el caso, un punto de vista experimental, algoritmico o metodológico. En el capítulo 2 hacemos una revisión del estado del arte en lo que respecta a la detección de la posición y orientación de objetos en la escena en RA, nos centramos en los sistemas de marcadores fiduciarios y describimos las motivaciones que nos condujeron a usar el sistema elegido.
En el capítulo 3, presentamos los materiales y métodos usados durante nuestro trabajo documentando los detalles para facilitar la replicabilidad de nuestros experimentos.
En el capítulo 4, describimos los experimentos realizados especificando las condiciones

en que fueron realizados y detallando la motivación de cada uno. Para cada uno de ellos exponemos los resultados encontrados.

En el capítulo 5 ofrecemos las principales conclusiones obtenidas al cabo de nuestro trabajo.

Capítulo 2

Estado del arte

Trabajos relacionados

La ubicación y la orientación de los objetos en una escena 3D es un requerimiento para una aplicación de realidad aumentada (RA). No existe una estrategia única y general para satisfacer este requerimiento sino que depende del tipo de problema en que la RA se utiliza, el requerimiento de precisión y el tipo de sensado disponible en cada situación.

En el caso que la escena se corresponda con ambientes extensos (típicamente ambientes exteriores) se puede utilizar Localización Basada en GPS. El sistema de RA puede utilizar datos de uno o más GPS para georreferenciar un móvil y puntos de interés. Por ejemplo Santana-Fernández [10] utilizan este tipo de sensado para implementar un sistema guiado de tractores que le permite al conductor ver sobre impuestas indicaciones y líneas de guía de manera natural y de esta forma tener mayor precisión en el trabajo agrícola. En la figura 2.1 se puede observar la escena tomada por una cámara y en el centro de la figura la trayectoria sobre impresa en la imagen.

Figura 2.1: La figura de la izquierda sin realidad aumentada, en el centro con realidad aumentada y a la derecha el equipamiento que tiene el tractor (extraído de [10]).

Cuando el entorno donde se aplica RA varía en cuanto a su extensión, el sensado mediante GPS se puede combinar con el Reconocimiento de Imágenes. Esto se basa en reconocer objetos o características específicas de una imagen para posicionar objetos virtuales en relación con esas características. Por ejemplo, Balint et al. [11] proponen un

método de localización basado en el análisis y tratamiento de las imágenes combinado con la información de un sensor GPS (cuando el usuario se encuentra en el exterior de los edificios) para identificar zonas urbanas. La figura 2.2 muestra la pantalla de un celular con una aplicación que reconoce un punto de control en la imagen capturada por el dispositivo móvil y superpone en la escena una fotografía tomada en el pasado del mismo sitio.

Figura 2.2: Superposición de una fotografía antigua de un espacio urbano luego de haber reconocido en la imagen obtenida por la cámara de un celular el punto de control donde debe realizarse la superposición (extraída de [11]).

Como una alternativa a la detección de características particulares en las imágenes, se puede utilizar el Reconocimiento de Superficies. Esta técnica se basa en la detección y utilización de superficies planas o estructuras físicas del entorno para superponer objetos virtuales. Por ejemplo, Ferrari et al. [12] logran, mediante la detección y seguimiento de superficies planas, superponer objetos virtuales en la escena como se muestra en la figura 2.3.

Figura 2.3: En a. y b. se observan etiquetas superpuestas sobre regiones planas de la imagen. En b. se muestra un detalle de a. y en d. un detalle de c. con una posición relativa de la cámara distinta [12].

Cuando el entorno tiene extensiones acotadas y se requiere rapidez y precisión, surge la posibilidad de adoptar otras estrategias para encontrar la posición y orientación de los objetos.

Una de ellas es el uso de sensores inerciales tales como acelerómetros o giróscopos o medidores de distancia (IR o láser) los cuales aportan información tanto de la posición relativa del usuario respecto a la escena como del o de los objetos de interés. Adicionalmente, se pueden usar técnicas de estéreo visión (cuando se dispone de al menos dos cámaras) para obtener, por ejemplo, la distancia de un objeto con respecto al usuario. Este tipo de situación son frecuentes en la industria y un ejemplo de RA para el ensamblado de piezas es descripto por Pierdicca et. al [13] quienes utilizan RA en la industria 4.0. La figura 2.4 se puede observar un indicativo para el operador mientras está llevando a cabo un procedimiento de ensamblaje.

Figura 2.4: Ejemplo de realidad aumentada utilizando gafas con sensores de movimiento durante una operación de ensamblado. Arriba a la derecha se puede observar un detalle del visor del usuario donde recibe una indicación del paso a ser realizado (extraída de [13]).

Una alternativa al uso de sensores inerciales o medidores de distancia es el uso de los marcadores fiduciarios. Estos son patrones gráficos con códigos específicos que se utilizan como puntos de referencia para detectar y rastrear la posición y orientación de los objetos en el espacio. Suelen ser imágenes bidimensionales con diseños únicos y estructuras que pueden ser fácilmente identificadas por cámaras u otros dispositivos de captura de imágenes (como escáneres láser). Los sistemas de visión por computadora pueden reconocer estos marcadores fiduciarios mediante algoritmos específicos que identifican sus

características visuales únicas, como bordes, esquinas o patrones de alto contraste.

El uso de marcadores fiduciarios fue evolucionando a partir de la aparición de nuevos diseños gráficos y de la posibilidad de definir diccionarios de los mismos (un diccionario es el conjunto de marcadores a ser reconocidos en un sistema de RA). ArtoolKit [6] es un método basado en marcadores cuadrados con un recuadro negro conteniendo en su interior, una imagen a reconocer. Al ser cuadrados, se utilizan las cuatro esquinas como referencia de la posición mientras que la gráfica interior permite reconocer si es o no un patrón válido y la orientación del marcador (ver figura 2.5). Al utilizar imágenes crece la detección de falsos positivos debido a que una vez reconocido el recuadro, depende de una buena iluminación para que la escala de grises, en el proceso de binarización de la imagen interna, pueda separar las partes blancas de las negras y así reconocer el patrón correctamente.

Figura 2.5: Ejemplo de marcador de ArtoolKit (extraída de [6]).

Como una evolución de ArtoolKit, Fiala, M [7] propuso ARTag. Estos marcadores, basados en figuras cuadradas de 10 x 10 cuadraditos usan bordes de 2 cuadraditos de ancho dejando en el medio una región de 6 x 6 cuadraditos disponibles para codificar el marcador. Esto permite disponer de una librería que cuenta con 2002 marcadores diferentes.

Figura 2.6: Ejemplo de marcador de ARTag (extraída de [7]). En la región central del marcador se pueden codificar diferentes patrones definiendo los cuadraditos negros y blancos (36 organizados en 6 columnas y 6 filas).

A diferencia de los marcadores ARTag y ArtoolKit, Naimark et al [14] proponen InterSense. Ellos introducen el uso de marcadores circulares donde, dada la forma, la única referencia es el centro de la circunferencia (ver figura 2.7). Como contrapartida, esto hace que se necesiten al menos cuatro marcadores para el reconocimiento de la orientación de los objetos. Cuando se detectan más de cuatro marcadores, se toman aquellos cuatro que generen el cuadrilátero de mayor área.

A diferencia de las propuestas mencionadas, donde los marcadores conservan la misma forma geométrica en su diccionario, Bencina, R [15] proponen trabajar con un con-

Figura 2.7: Ejemplo de marcador InterSence [14].

junto de marcadores heterogéneos desde el punto de vista geométrico. En este método, denominado ReactiVision, los marcadores no tienen una forma particular ya que pueden ser circulares, cuadrados o rectangulares. Su principal característica es que poseen regiones blancas y negras unas dentro de otras, y para el reconocimiento, se realiza un grafo de adyacencia (un árbol) como se muestra en la figura 2.8.

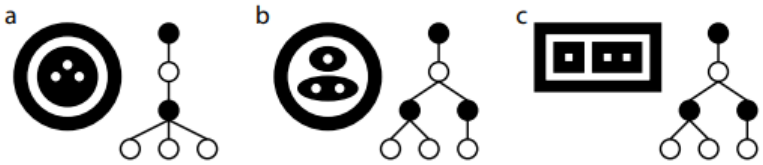

Figura 2.8: Ejemplo de marcadores de ReactiVision con su correspondiente gráfico de adyacencia a su derecha (extraída de [15]).

Existen una variedad más extensa que la mencionada de sistemas de marcadores, sin embargo lo más usual son aquellos de forma cuadrada con marco y con una codificación binaria dentro del mismo. Quizás una de la más simple es Matrix Rekimoto, J [16]. En la figura 2.9 se puede observar un marcador con la identificación de la posición y orientación en la escena. La codificación es interna usando cuadraditos obteniendo una codificación del marcador mediante un sistema binario. Para diferenciar un marcador de otro, se agrega en la codificación bits de redundancia.

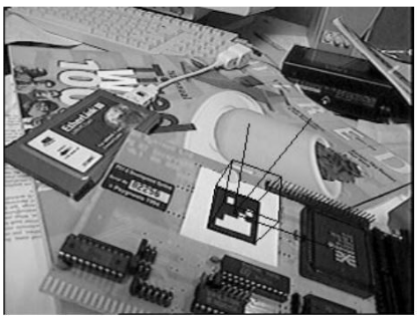

Figura 2.9: Ejemplo de marcadores de ReactiVision con su correspondiente gráfico de adyacencia a su derecha (Identificación de un marcador usando el método Matrix. Se puede observar la superposición de los ejes de referencia del marcador y las aristas de un cubo cuya base es el propio marcador (extraída de [16]).

La diferenciación de la codificación entre marcadores depende de varios factores. Uno de ellos es el método de detección de los bits en la codificación. ARTag usa detección de bordes para llevar a cabo esta tarea lo cual lo hace más robusto a los cambios de iluminación a diferencia de los otros métodos que usan umbralación (donde cuentan la cantidad de píxeles blancos o negros dentro de cada cuadradito de la imagen y así determinan si el significado del bit es 1 o 0).

Otro factor muy relevante para diferenciar la codificación entre marcadores es la distancia de Hamming entre ellos. A mayor distancia entre la codificación de dos marcadores existen más posibilidades de recuperar el error que pudo haber surgido durante la interpretación de su codificación. En este sentido, ARTag es menos flexible que otros métodos que permiten definir su diccionario donde el usuario puede establecer la mínima distancia de Hamming que debe existir entre dos marcadores. ArtoolKit Plus [17] es otro de los métodos con marcadores cuadrados y codificación binaria interna. La versión más evolucionada de este método, ArtoolKit Plus BCH, posee un diccionario de marcadores con una distancia de Hamming mínima de 2. Esto es una limitación ya que sólo se podrá corregir una codificación mal interpretada en un 1 bit.

Además de la mínima distancia de Hamming entre marcadores, otro asunto a tener en cuenta en el reconocimiento de marcadores es la oclusión, es decir, qué ocurre cuando alguna parte del marcador es tapada. La solución más robusta para este tipo de problemas es incrementando la cantidad de marcadores que tiene asociado un objeto o un lugar del entorno pero esto requiere disponer de un diccionario que posea una cantidad suficiente de los mismos. A pesar que ARTag tiene las limitaciones mencionadas más arriba respecto a su diccionario, permite identificar un marcador aun cuando parte del mismo es ocluido. En la figura 2.10 se puede observar que para reconocer un marcador de ArtoolKit Plus se necesita que el recuadro este sin oclusión (izquierda), aunque esta sea mínima (centro) y como ARTag lo reconoce (derecha) aun cuando la oclusión incluye la codificación del mismo [18].

Figura 2.10: Imagen con ejemplo de marcadores respecto a la oclusión, se observa en la imagen del medio que ocluyendo mínimamente, ya no lo detecta (extraída de [18]).

Un método que permite tratar con gran parte de las dificultades mencionadas más arriba se debe a Garrido-Jurado, S et al. [19]. Este método se denomina Aruco y también usa marcadores cuadrados con una identificación basada en codificación binaria (ver fi-

gura 2.11). Uno de los aportes fundamentales de estos autores consiste en la posibilidad de generar diccionarios utilizando una cantidad arbitraria de n x n bits. En su trabajo se presenta el algoritmo de generación de diccionarios donde se puede especificar la distancia de Hamming mínima tolerada entre dos marcadores, la cantidad de bits usados para la codificación (en el formato de n x n) y la cantidad de marcadores necesarios. Esta posibilidad hace muy flexible su uso, en particular para aplicaciones de RA donde la oclusión podría representar una limitación.

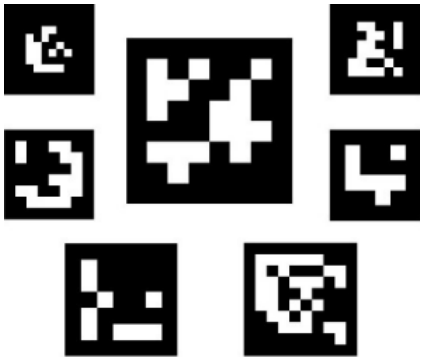

Figura 2.11: Ejemplo de marcadores visuales Aruco (extraído de [8]).

En resumen, hay marcadores circulares que tienen como desventaja tener que reconocer a varios de estos para la detección de la posición y orientación de los objetos en el espacio debido a que la única referencia es su centro. Otros que se reconocen a través de imágenes en el interior del marcador acarreando desventajas respecto a la iluminación. Además, hay métodos con diccionarios fijos, o métodos de codificación que no permiten tener una gran cantidad de marcadores por que usan parte de su código de codificación checksums o redundancia de bits. Cada uno de estos tiene ventajas y desventajas según el tipo de aplicación de RA que se quiera desarrollar.

Se han evaluado las diferentes opciones y hemos elegido el método Aruco para obtener el posicionamiento y orientación de los objetos en la escena. Los principales motivos de esta elección fueron:

- usa marcadores cuadrados, lo que brinda en cada uno, varios puntos de referencia (las esquinas y centro),
- la librería que implementa el método de detección de marcadores, permite especificar la cantidad de bits para la codificación y el tamaño físico del marcador,
- está documentado el método de generación de diccionarios lo cual permite hacer una implementación propia y generar los diccionarios $ad - hoc$ para la aplicación, y

- es compatible con librerías como OpenCV (incluso, Aruco usa varias funciones de OpenCV) y OpenG, lo cual son frecuentemente usadas en aplicaciones de RA.

Capítulo 3

Materiales y métodos

En el presente capítulo vamos a mostrar cuáles son los recursos necesarios para la utilización de marcadores Aruco en el reconocimiento de la pose de los objetos en la escena 3D. Una cámara y los marcadores fiduciarios son elementos principales, pero, ¿qué características deben tener? ¿qué métodos implementan? ¿bajo qué condiciones se deben utilizar?.

En la primera parte describimos el tipo de cámara que utilizamos, los parámetros y el funcionamiento de la misma. También incluimos los métodos usados para la calibración de sus parámetros. En la segunda parte, abordaremos lo referido a marcadores fiduciarios y sus diccionarios, mostramos el algoritmo que permite crearlos y por último, cómo reconocer y corregir el error en el proceso de detección de marcadores.

3.1. Parámetros de una cámara

En cuanto a la cámara, utilizamos el modelo pin-Hole: El dispositivo más común para la formación de imágenes es una cámara con una apertura muy pequeña. Como muestra la figura 3.1, podemos entender a una cámara como un cubo hueco en el que en uno de sus lados existe un agujero muy pequeño. Por este agujero pasan los rayos de luz que forman la imagen en la cara interna opuesta del cubo.
Al tratarse de un agujero tan pequeño, de los rayos que proceden de cada punto del mundo real (3D) exclusivamente uno pasa por él, lo que implica que sobre cada punto de la cara interior del cubo situada enfrente de la apertura solo incide un rayo de luz, formando así una imagen en el plano (2D).

Cabe aclarar que la imagen que se proyecta en la cara opuesta del orificio en el cubo se verá invertida y a una longitud denominada distancia focal f.

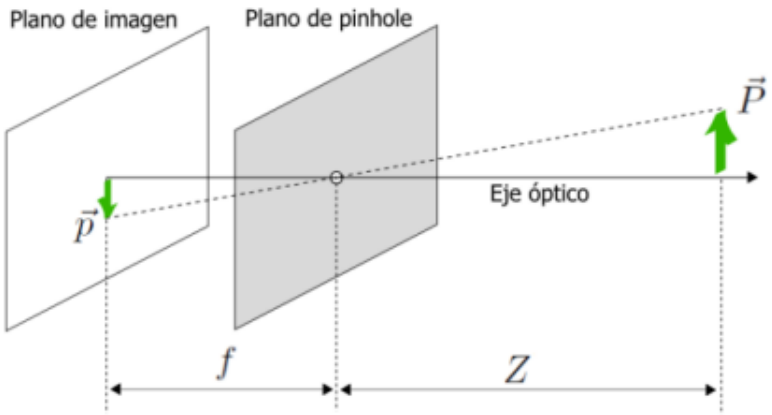

Figura 3.1: Esquema de funcionamiento de una cámara. f es la distancia focal y Z la distancia del punto a la cámara (extraída de [20]).

3.1.1. Parámetros intrínsecos

Los parámetros intrínsecos son todos los parámetros que tienen relación con el conjunto de cámara y óptica que están involucrados en la transformación de puntos 3D en el sistema de referencia de la cámara a puntos 2D en el plano imagen. Podemos encontrar parámetros como el desplazamiento del centro de la imagen (c_x, c_y) y la distancia focal f (ver figura 3.2).

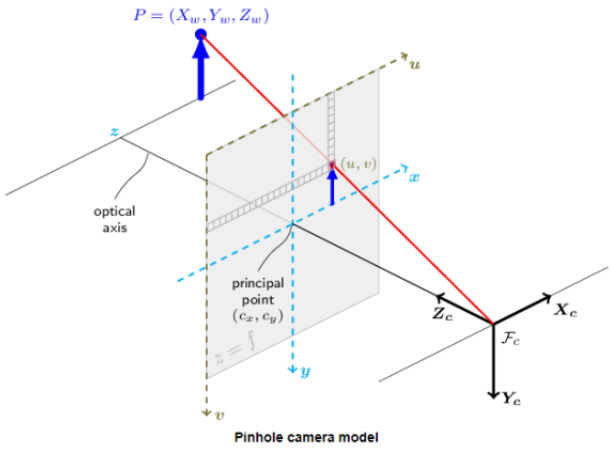

Figura 3.2: Modelo de cámara. (X_w, Y_w, Z_w) son las coordenadas de un punto en el espacio respecto al sistema de coordenadas del mundo. Los ejes X_c, Y_c y Z_c son los ejes del sistema de coordenadas de la cámara. (c_x, c_y) expresa el centro de la imagen por el cual pasa el eje óptico. (u,v) son las coordenadas del píxel, en el plano de la imagen, donde se proyecta (X_w, Y_w, Z_w) (extraído de [20]).

Los parámetros intrínsecos se pueden agrupar dentro de una matriz denominada matriz de calibración o matriz de la cámara o matriz de parámetros intrínsecos:

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

donde f_x y f_y son las longitudes focales de la cámara en el eje x y en el eje y, c_x y c_y son las coordenadas del centro de la cámara (*principal point*) que determinan el desplazamiento del centro de la imagen.

También existen parámetros intrínsecos que permiten expresar la distorsión provocada por las lentes y el tamaño del píxel. La distorsión de la lente deteriora la calidad geométrica de la imagen. Si no se lo tiene en cuenta, la proyección de los objetos en el espacio sobre el plano de la imagen aparecerá distorsionada y afectará el cálculo de la posición de dichos objetos en el sistema de coordenadas del mundo.
Esto ocurre porque los rayos de luz no emergen de las lentes en las direcciones previstas. Pueden existir de dos tipos: radial y tangencial.

3.1.1.1. Distorsión radial

La distorsión radial de la lente provoca que los puntos de las imágenes se desplacen de forma radial a partir del eje óptico. Se puede observar en la figura 3.3 como los rayos más alejados del centro óptico se curvan bastante más que los que inciden en las proximidades del centro de la lente. La causa más común de esta distorsión es el pulido defectuoso de la lente.

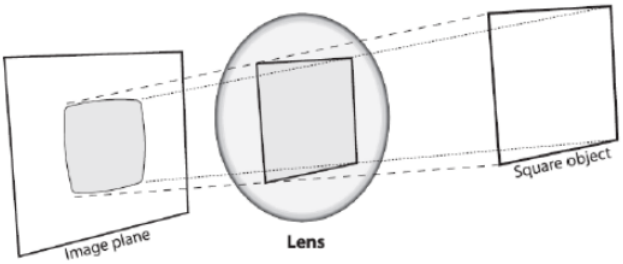

Figura 3.3: Distorsión radial observada sobre el plano de la imagen producida por una lente a partir de un objeto rectangular en el espacio (extraída de [20]).

Para modelar la distorsión con una precisión máxima se necesitaría una serie infinita de elementos (coeficientes) pero, en la práctica, alcanza con tener en cuenta los dos primeros coeficientes de la serie, o incluso el primero, es suficiente.

El grado de distorsión se determina por los coeficientes de distorsión. Según su signo puede haber dos tipos de distorsión radial: positiva y negativa. En la distorsión positiva la imagen toma un tipo corsé y en la distorsión negativa la imagen toma un tipo barril como se puede observar en la figura 3.4.

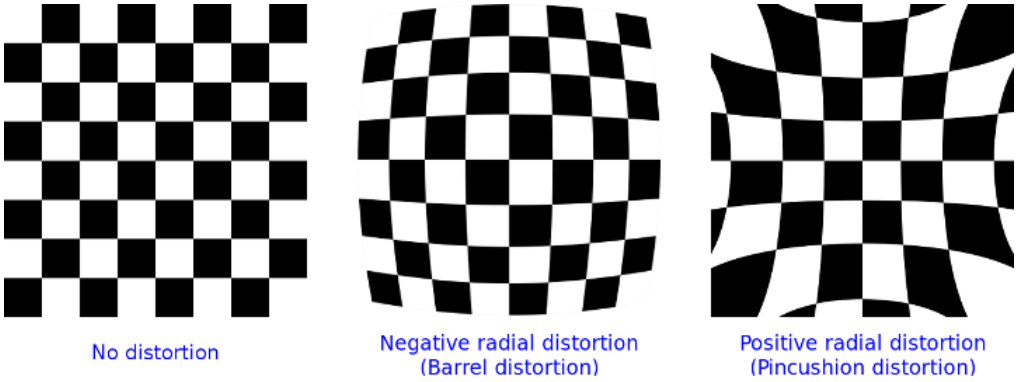

Figura 3.4: Deformaciones en la imagen de acuerdo al tipo de distorsión radial. (extraída de [20]).

3.1.1.2. Distorsión tangencial

Este tipo de distorsión se produce perpendicularmente a las líneas radiales a partir del eje óptico. La principal causa de esta distorsión es el centrado defectuoso de todos los elementos que forman el sistema de lentes, producido por un montaje imperfecto de los mismos o del sensor (CCD) como muestra la figura 3.5.

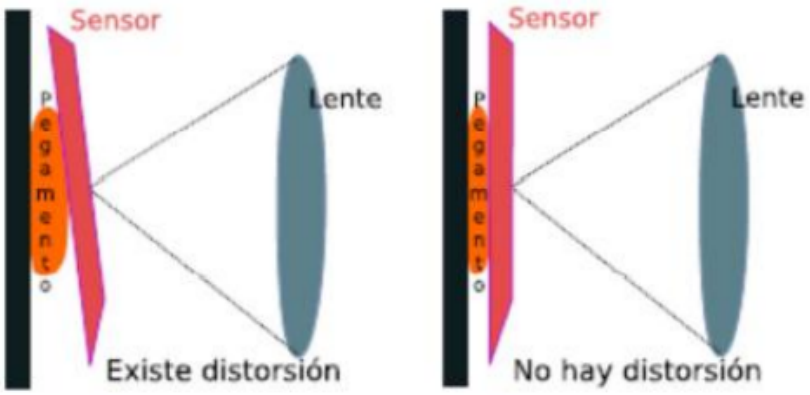

Figura 3.5: Imperfección del montaje del sensor lo cual produce distorsión tangencial (extraída de [20]).

Al contrario que en la distorsión radial los efectos de la distorsión tangencial son menos significativos por lo que en la mayoría de las aplicaciones se considera despreciable teniendo en cuenta únicamente a los efectos de la distorsión radial.

Todos los coeficientes de distorsión, radiales y tangenciales (en el caso que se consideren) se pueden agrupar en un único vector, denominado vector de distorsión.

Podemos, entonces, decir que los parámetros intrínsecos de la cámara lo conforman los pertenecientes a la matriz de calibración y los del vector de distorsión.

3.1.2. Parámetros extrínsecos

Los parámetros extrínsecos hacen referencia a la ubicación física de la cámara en el espacio. Son los que definen la posición y la orientación de la cámara con respecto al sistema de coordenadas global o coordenadas del mundo. Esto es, los parámetros extrínsecos no dependen de la propia cámara sino de su posición espacial. Estos parámetros quedan definidos por el vector de traslación y la matriz de rotación de la cámara.

En la figura 3.6 se puede observar la rotación en sentido horario alrededor de cada uno de los ejes coordenados, debido a que, dependiendo del eje que se adopte como referencia, para realizar la mencionada rotación habrá que utilizar una matriz de rotación u otra.

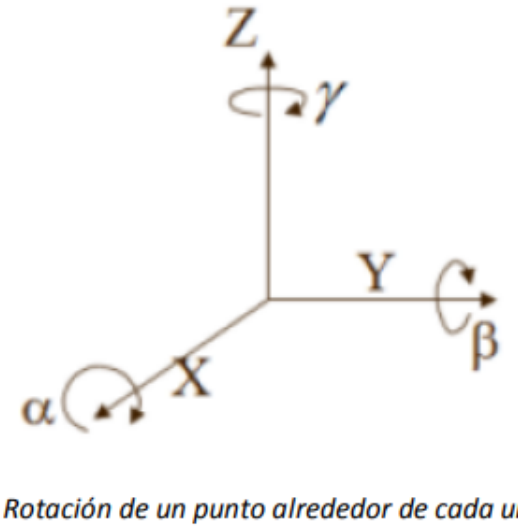

Figura 3.6: Rotación en sentido horario respecto a cada uno de los ejes cartesianos (extraída de [20]).

La matriz de rotación R reproduce un giro de la cámara o de un objeto respecto de ella. Y, como acabamos de mencionar, adoptará distintos valores según el eje coordenado

de referencia:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\alpha & sin\,\alpha \\ 0 & -sin\,\alpha & cos\,\alpha \end{bmatrix}$$

$$R_y = \begin{bmatrix} cos\,\beta & 0 & sin\,\beta \\ 0 & 1 & 0 \\ -sin\,\beta & 0 & cos\,\beta \end{bmatrix}$$

$$R_z = \begin{bmatrix} cos\,\gamma & sin\,\gamma & 0 \\ -sin\,\gamma & cos\,\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

donde α, β y γ son los ángulos de rotación sobre los ejes x, y, z, respectivamente.

Si se tuviera un caso en que la cámara solo esté rotada respecto al eje z la matriz que se debería utilizar sería R_z y las coordenadas de un punto en el espacio (x, y, z) quedarían expresadas respecto del sistema de coordenadas de la cámara como (x', y', z') de acuerdo a:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} cos\,\gamma & sin\,\gamma & 0 \\ -sin\,\gamma & cos\,\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

En el caso que la cámara esté rotada en más de un eje, la matriz de rotación de la cámara puede ser expresada como el producto de las matrices de rotación en cada uno de los ejes de coordenadas:

$R = R_X R_Y R_Z$

Sabemos ahora que la cámara tiene parámetros extrínsecos e intrínsecos, veremos entonces, cómo funciona la misma.

3.2. Funcionamiento de una cámara

Conociendo los parámetros de la cámara podemos entender cómo funciona la proyección de un punto en 3D, expresado en coordenadas del mundo, sobre una imagen 2D en el plano de la imagen (CCD) siguiendo el esquema de la figura 3.7.

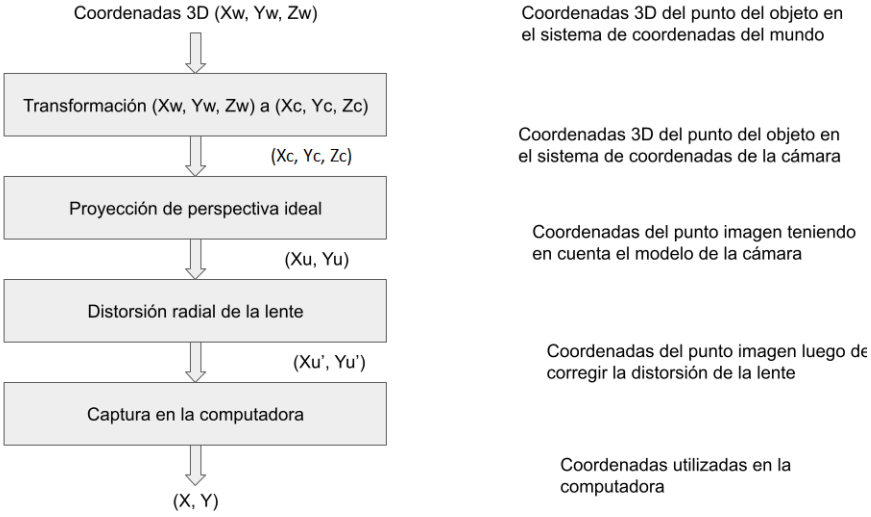

Figura 3.7: Esquema de bloques del funcionamiento de una cámara desde el punto de vista del cambio de coordenadas en cada bloque. (extraída de [20]).

Explicaremos cómo funciona dicha proyección, utilizando los parámetros de la cámara, tanto intrínsecos como extrínsecos.

Sea un punto (x, y, z) expresado en el sistema de coordenadas del mundo, el mismo puede ser expresado respecto del sistema de coordenadas de la cámara primero rotando a (x, y, z) usando la matriz de rotación de la cámara y luego traladando dicho punto usando el vector de traslación:

$$P_c = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} P_w$$

siendo P_w el punto en coordenadas 3D expresadas en una vector de 4x1 donde el último elemento del vector es 1, R la matriz de rotación de 3x3 y t el vector columna de traslación de 3x1. El producto da como resultado el punto rotado y trasladado P_c. Podemos también escribir esta información siguiendo:

$$Z_c \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

donde $P_w = (X_w, Y_w, Z_w)$, $r_{i,j}$ los elementos de la matriz R con $1 \leqslant i, j \leqslant 3$, $t = (t_x, t_y, t_z)^t$ y $x' = \frac{X_c}{Z_c}$, $y' = \frac{Y_c}{Z_c}$ (para Z_c distinto de 0)

Una vez obtenido el punto $Z_c(x', y', 1)^t$ se multiplica por la matriz de calibración quedando:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 1 & r_1 2 & r_1 3 & t_x \\ r_2 1 & r_2 2 & r_2 3 & t_y \\ r_3 1 & r_3 2 & r_3 3 & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

donde u y v expresan la fila y la columna dentro de plano de la imagen (sensor CCD en la cámara) y s es un factor escalar.

Para entender qué es s, podemos introducir el gráfico de la figura 3.8.

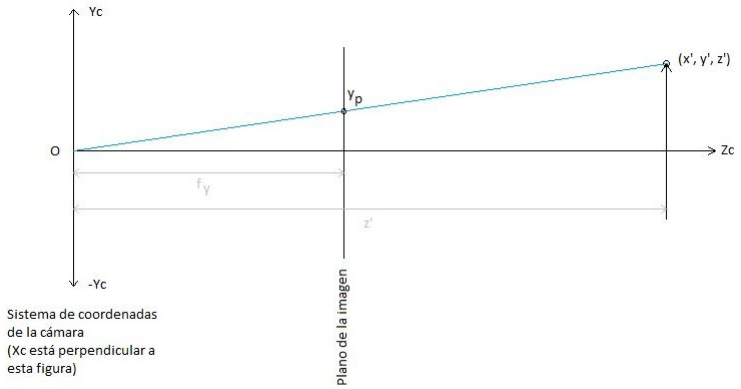

Figura 3.8: Esquema de la proyección de un punto en el mundo sobre el plano de la imagen.

Sea z' la distancia entre el punto (x', y', z') y el centro de proyección O, f_y la distancia focal en y, y_p la coordenada y en el plano de la imagen e y' el valor de la coordenada y del punto (x', y', z') expresado en el sistema de coordenadas de la cámara, podemos establecer, por triángulos semejantes, la siguiente proporción:

$$\frac{y_p}{f_y} = \frac{y'}{z'}.$$

de donde resulta que

$$y_p = \frac{y' * f_y}{z'}.$$

Análogamente, se puede observar que

$$\frac{x_p}{f_x} = \frac{y'}{z'}.$$

de donde resulta que

$x_p = \frac{y' * f_x}{z'}$.

Recordemos que el vector (x', y', z') se obtenía según

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} R & t \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix},$$

de modo que al multiplicar por

$$\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

nos queda

$$\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} f_x x' + C_x z' \\ f_y y' + C_y z' \\ z' \end{pmatrix} = z' \begin{pmatrix} \frac{f_x x'}{z'} + C_x \\ \frac{f_y y'}{z'} + C_y \\ 1 \end{pmatrix}$$

donde llamaremos

$$u = \frac{f_x x}{z'} + C_x \text{ y } v = \frac{f_y y'}{z'} + C_y$$

que coincide con las definiciones de x_p e y_p obtenidas más arriba al considerar semejanzas de triángulos pero, en este caso corrigiendo el centro ya que las filas y columnas en el plano de la imagen se corresponden con el $(0, 0)$ en el vértice superior izquierdo. Vemos así que, $s = z'$.

Ahora que no sólo conocemos que las cámaras tienen parámetros, sino que además sabemos cómo funciona la misma, nos detenemos a considerar una cuestión importante: la calibración de la cámara. ¿Por qué es importante considerarla?. Pues es necesario conocer con precisión la matriz de calibración y los coeficientes de distorsión al momento de determinar la posición y orientación de los objetos en el espacio. Por ejemplo, si una cámara tiene una resolución de 640 x 480 píxeles no necesariamente el eje óptico pasa por el píxel que está en la fila 240 y en la columna 320 del sensor (CCD).

3.3. Calibración

En esta sección mostraremos dos métodos para calibrar una cámara. Esto es muy importante ya que una incorrecta determinación de los parámetros de la cámara influyen

considerablemente en la detección de marcadores.

Se utilizaron dos métodos, el primero es el propuesto por Muñoz Salinas [8].

Método 1: Calibración con Aruco

Para calibrar la cámara con la librería Aruco primero se debe descargar el proyecto de [21]. La versión que se utilizó es la 3.1.12.

Una vez descargado y compilado el proyecto se debe ejecutar desde una ventana de comandos (en Windows) el programa aruco_calibration.exe con el siguiente comando.

aruco_calibration live:0 calibrationTest.yml -size:0.04 -save: enable
donde:

- live:0 es para calibrar en tiempo real con una cámara (ya sea conectada por usb o bien integrada en la laptop),
- calibrationTest.yml es el nombre del archivo de salida donde quedarán registrados los valores de la matriz de calibración y los coeficientes de distorsión,
- size:0.04 es el tamaño del marcador expresada en metros y,
- save: enable es para ir almacenando las imágenes capturadas durante la calibración.

El valor del argumento size debe coincidir con el tamaño de impresión de los marcadores. El proyecto Aruco [21] incluye la imagen de un tablero en un archivo del tipo pdf llamado aruco_calibration_grid_board_a4 para utilizar en la calibración.

Una vez ejecutado el comando se abrirá una pantalla donde se solicitará que se comience a agregar las imágenes presionando la tecla “a” del teclado como muestra la figura 3.9. Se recomienda que las imágenes tomadas sean variadas, desde distintos ángulos y distancias de la cámara respecto al tablero, incluyendo aquellas cuyos bordes estén muy cercanos a los bordes del tablero.

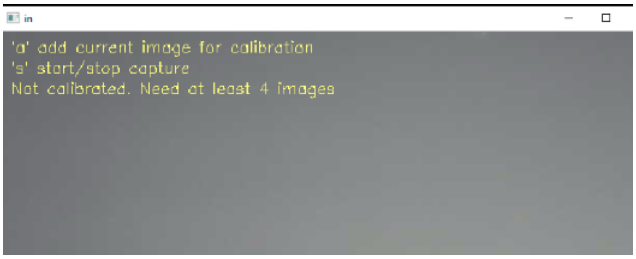

Figura 3.9: Captura de pantalla que se visualiza ni bien ejecutado el comando para la calibración.

A partir que se alcancen las 4 imágenes se mostrará, de ahí en adelante, el error obtenido (ver figura 3.10).

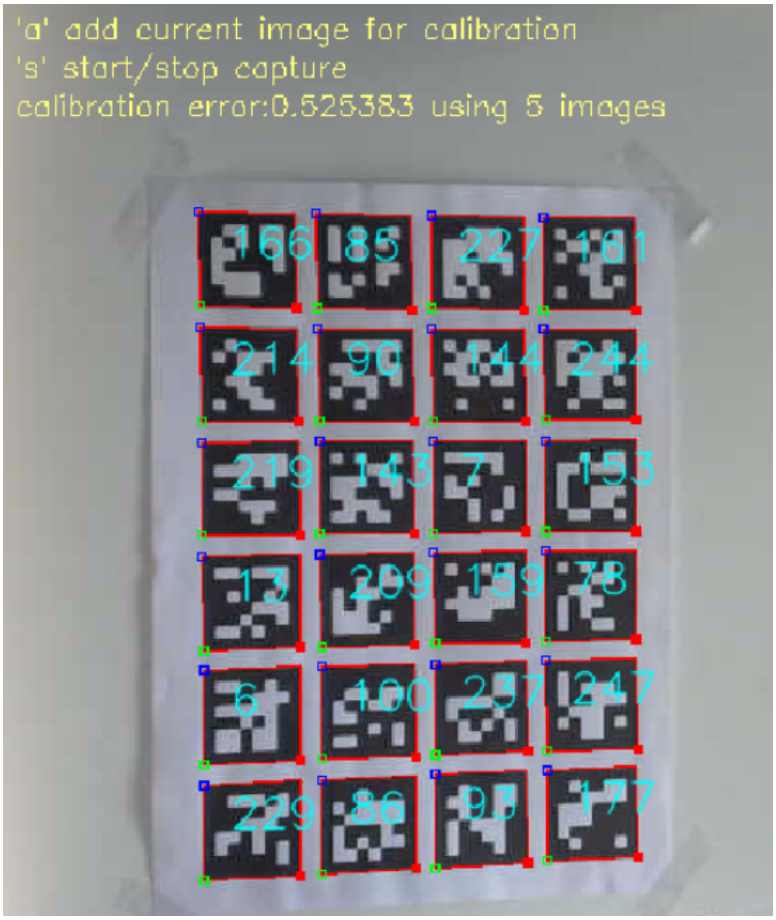

Figura 3.10: Captura de pantalla con el error luego de capturar 5 imágenes.

Al aumentar la cantidad de imágenes, el error tiende a bajar, y una vez que se llega a un error aceptable, se termina el proceso presionando el botón Esc. En la consola (ventana de comandos de Windows) se podrá observar el archivo donde se almacenó la calibración y el error obtenido (ver figura 3.11).

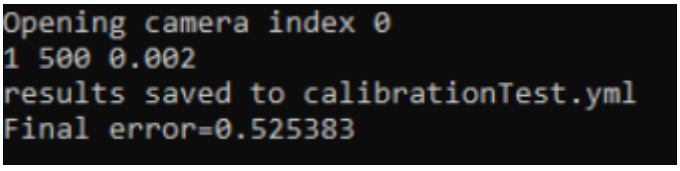

Figura 3.11: Captura de pantalla con el mensaje luego de finalizar el proceso de calibración.

El archivo de calibración se almacena en el formato que se muestra en la figura 3.12.

```
%YAML:1.0
---
image_width: 640
image_height: 480
camera_matrix: !!opencv-matrix
  rows: 3
  cols: 3
  dt: f
  data: [ 9.50486938e+02, 0., 3.74058929e+02, 0., 9.49478882e+02,
    2.35135773e+02, 0., 0., 1. ]
distortion_coefficients: !!opencv-matrix
  rows: 1
  cols: 5
  dt: f
  data: [ 2.94541344e-02, 1.88088298e+00, -7.11962348e-03,
    6.30629668e-03, -7.91974163e+00 ]
```

Figura 3.12: Captura de pantalla del archivo de salida donde se puede observar la matriz de calibración y los coeficientes de distorsión. Notar que el centro no se encuentra en el punto $(320, 240)$ sino en $(374, 235)$.

Método 2: Calibración con openCV

Para este método se deben tomar imágenes de un tablero provisto por la librería OpenCV (ver figura 3.13).

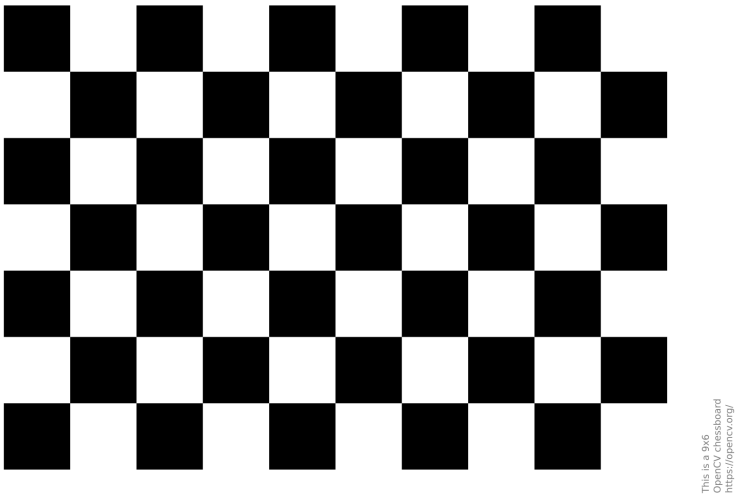

Figura 3.13: Tablero utilizado para la calibración usando las funciones de openCV (extraído de [22]).

Como en el método basado en marcadores Aruco (explicado en la sección anterior), hay que tomar la mayor cantidad de imágenes del tablero con diferentes ángulos y distancias entre la cámara y el tablero para mejorar la calibración. Una diferencia con el otro método, es que se puede calibrar con distintas resoluciones de la cámara, por lo tanto, se debe conocer la misma.
El método utilizado para calibrar está implementado en Python [22], se describen los pasos:

- Se importan las librerías glob, pickle, pandas, cv2 y numpy, con el gestor de paquetes pip. (por ejemplo pip install glob2, pip install pandas, etc)
- se definen las variables de calibración: $frameSize = (1280, 720)$ para la configuración de la resolución y $chessboardSize = (9, 6)$ para el tablero, donde los valores 9 y 6 indican la cantidad de esquinas horizontales y verticales que aparecen cuando dos cuadrados negros se juntan en un vértice. En este caso el tablero tiene $10x7$ cuadrículas, y la variable de calibración $chessboardSize$ es $(9, 6)$
- para hallar el patrón del tablero se utiliza la función $cv.findChessboardCorners()$. Se necesita pasar como parámetro la imagen del tablero y el tamaño del patrón, en este caso $cv.findChessboardCorners(image, (9, 6))$
- Una vez que fueron encontradas las esquinas, se puede aumentar la precisión utilizando la función $cv.cornerSubPix()$. También se puede dibujar el patrón con $cv.drawChessboardCorners()$

En la figura 3.14 se muestra el patrón dibujado en el proceso de calibración.

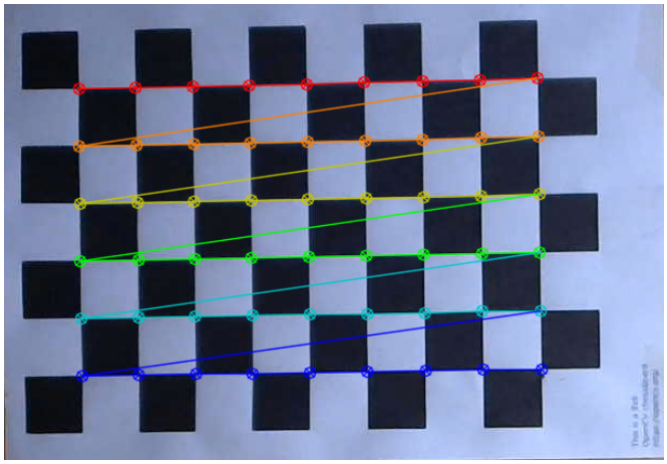

Figura 3.14: Captura de pantalla con el patrón dibujado.

Una vez que se reconoce el patrón del tablero, se puede dar inicio a la calibración con la función $cv.calibrateCamera()$, allí se obtiene la calibración, sin embargo. El

método permite refinar la misma con las funciones $cv.getOptimalNewCameraMatrix()$ y $cv.undistort()$.

Luego de ver los materiales y métodos referido a la cámara, nos adentramos en los marcadores fiduciarios.

3.4. Marcadores

Un marcador Aruco es una figura cuadrada compuesta por un borde negro, una matriz interna binaria (representada por dos colores, en este caso blanco y negro) y un identificador (id). En la figura 3.15 se muestra un ejemplo de marcador.

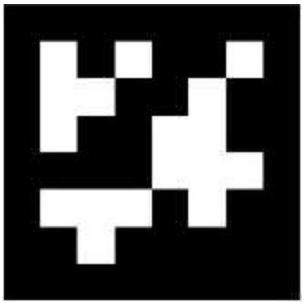

Figura 3.15: En el marcador se puede ver el borde negro, es decir, el contorno está formado por cuadraditos negros, la matriz interna binara es la combinación de colores (blanco y negro) que están por dentro de dicho borde, en este caso, es de 6x6.

En cuanto al identificador, mencionamos que al ser un marcador de 6x6 tiene 36 bits, con esto se podría expresar al valor del id como el número decimal expresado en binario pero el id es representado por la posición que ocupa en un diccionario, y ¿qué es un diccionario?

3.5. Diccionario

Un diccionario es un conjunto de marcadores, cada uno de ellos es único, y en lo posible, que no sea muy parecido a otro para evitar errores durante su reconocimiento. Los diccionarios tienen dos propiedades:

- la cantidad de marcadores,
- el tamaño de cada marcador (bits)

Como mencionamos en el estado del arte, uno de los motivos de la elección de Aruco es que permite generar diccionarios $ad-hoc$, ahora, ¿cómo se genera?

3.5.1. Generación de marcadores y diccionarios

El sistema de marcadores que usamos es de base cuadrada con códigos binarios, sin embargo en lugar de utilizar un conjunto predefinido de marcadores (los que incluye la librería Aruco) se propone generar marcadores configurables, es decir, diccionarios dinámicos donde se pueda fácilmente modificar no sólo la cantidad de marcadores de un diccionario, sino también el tamaño de cada uno de ellos. Para esto utilizamos el algoritmo propuesto por Muñoz Salinas, R y colaboradores [19] que produce marcadores utilizando un criterio para maximizar la distancia entre ellos (lo que permite reducir el error al intentar reconocer un marcador) y el número de transiciones de bits. También se incluye la descripción del método para detectar y corregir errores en base al diccionario propuesto.

3.5.2. Generación automática de diccionarios

Los aspectos más relevantes a tener en cuenta, a la hora de diseñar un diccionario de marcadores, son las tasas de falsos positivos y negativos, la tasa de confusión entre marcadores y el número de marcadores válidos. Las dos primeras se abordan utilizando la detección de errores y bit de corrección, esto por otro lado, reduce el número de marcadores válidos. La tasa de confusión sólo depende de la distancia entre los marcadores empleados. Si están demasiado cerca y durante la identificación se detectan erróneamente algunos bits, esto puede conducir a un error al asignar un número de identificador en el diccionario que sea equivocado.

Otra propiedad deseable de los marcadores es tener un alto número de transición de bit, ya que de esta forma es menos probable que se confundan con objetos del entorno. Por ejemplo los códigos binarios con solo ceros o unos se imprimirán como marcadores completamente negros o blancos respectivamente, con lo cual, pueden confundirse fácilmente con objetos del medio ambiente.

Si bien, en general, las librerías que se usan para trabajar con marcadores fiduciarios emplean diccionarios fijos, Aruco nos brinda un método para generar diccionarios con el número deseado de marcadores y bits por marcador para dar respuesta a las preguntas mencionadas anteriormente. El problema luego, se reduce a seleccionar una cantidad m de marcadores del espacio dentro de todos los marcadores posibles con n bits, logrando que estén lo más lejos posibles entre sí y con tantas transiciones de bits como sea posible.

En general el problema es encontrar el diccionario D^* que maximiza el criterio deseado:

$$D^* = argmax\{t(D)\} \forall D \in \mathbf{D}.$$

donde D^* es el diccionario buscado optimizado, t es el umbral de distancia requerida para un diccionario, D es cada uno de los diccionarios que pertenecen a $\mathbf{D}$ y por último

D expresa el espacio de posibles diccionarios.

3.5.2.1. Descripción del algoritmo

Partimos de un diccionario D vacío, los marcadores son codificados como un cuadrado $(n+2)*(n+2)$ dejando en cada dimensión del cuadrado n bits para la codificación de la matriz interna y los 2 restantes son los bits del borde alrededor de la matriz interna que son fijos y de color negro para facilitar la detección (borde).

Definimos un marcador m como una tupla compuesta por n palabras binarias $w_i, 0 \leqslant i \leqslant n$, cada una de longitud n:

$m = (w_0, w_1, ...w_{n-1})$.

donde cada palabra w_i es m entonces es una tupla compuesta por n palabras binarias (w) de longitud n tales que

$w_i = (b_0, b_1, ...b_{n-1})/b_i \in \{0, 1\}$.

w será entonces el conjunto de todas las palabras posibles de n bit cuyo cardinal es

$|w| = 2^n$.

El algoritmo consiste en un ciclo que donde, en cada iteración del mismo, se selecciona un marcador basado en un proceso estocástico (al azar) para favorecer la obtención de marcadores con un mayor número de transiciones de bits y cuyas palabras no han sido aún agregadas a algún marcador en D.

Si la distancia entre el marcador generado y los marcadores que ya han sido incorporados al diccionario D es mayor que un valor de umbral t (un valor arbitrario que permite reducir o no la cantidad de marcadores cuando es posible) entonces se agrega en el diccionario. De lo contrario, el marcador es rechazado y un nuevo marcador es aleatoriamente generado. Este proceso se detiene cuando se obtiene la cantidad de marcadores que se requería.

Debido a la naturaleza probabilística del algoritmo, la aceptación de nuevos marcadores podría ser improbable o incluso imposible en algunos casos. Para garantizar la convergencia del algoritmo, el umbral de distancia se establece inicialmente al máximo posible entre los marcadores que puede tener el diccionario D. A lo largo del proceso el valor de t se reduce después de una serie de interacciones improductivas, por lo que el valor final de t representará la distancia mínima entre dos marcadores cualesquiera en D y se utilizará como base para la detección y corrección de errores.

El algoritmo que se propone para esta realización es el siguiente:

$D \leftarrow \emptyset$ // Inicializamos vació el diccionario D
$t \leftarrow t_0$ // Inicializamos el umbral de distancia mínima
$q \leftarrow 0$ // Iniciamos en 0 al contador de iteraciones improductivas
while (D no tiene la cantidad de marcadores deseada)
    Generamos al azar un nuevo marcador m
    if (distancia de m al resto de los marcadores de $D \geqslant t$)
        Agregamos m a D
        $q \leftarrow 0$ // Iniciamos en 0 al contador
    else
        $q \leftarrow q + 1$ // Incrementamos el contador de marcadores improductivos
        if ($q = max_q$) //Si se llega a la cantidad máxima de iteraciones improductivas
            $t = t - 1$ //Decrementamos el umbral de distancia mínima
            $q \leftarrow 0$ // Inicializamos en 0 al contador

Algoritmo extraído de [19]

3.5.2.2. Generación del marcador

La selección de los marcadores se realiza con un método aleatorio, aumentando la probabilidad de obtener marcadores con un alto número de transiciones y cuyas palabras aún no están presentes en los marcadores del diccionario D o su frecuencia relativa es muy baja.

El proceso propuesto para generar un marcador consiste en seleccionar n palabras w_i con reemplazo. Para realizar, esto cada palabra w_i que pertenece a W tiene una probabilidad de ser seleccionada en cada iteración que se define como

$$P\{w = w_i\} = \frac{T(w_i)O(w_i,D)}{\sum_{w_j \in w} T(w_j)O(w_j,D)}.$$

Esta ecuación define la probabilidad de seleccionar una palabra basada en la combinación de dos funciones $T(w_i)$ y $O(w_i; D)$. $T(w_i)$ perteneciente a $[0, 1]$ cuenta la cantidad de transiciones en w_i y se define como:

$$T(w_i) = 1 - \frac{\sum_{j=0}^{n-2} \delta(w_i^{j+1}, w_i^j)}{n-1}.$$

siendo w_i^j el bit $-j$ de la palabra w_i, y $\delta(delta)$ es 1 si ambos elementos son iguales y 0 en caso contrario. De este modo, $T(w_i)$ tenderá a uno cuando el número de transiciones entre bit consecutivos aumenta y a cero cuando disminuye. Por ejemplo, las palabras 010110 y 000011 tienen valores de $T(010110) = \frac{4}{5}$ y $T(000011) = \frac{1}{5}$ respectivamente, esto es proporcional al número de transiciones de bits.

Por otro lado, la función $O(w_i, D)$ representa el número de veces que aparece la palabra w_i en los marcadores que están en el diccionario D. La idea es reducir la probabilidad de elegir palabras que ya han sido seleccionadas. La imagen de $O()$ es $[0, 1]$ y se define como:

$$O(w_i, D) = \begin{cases} 1 - \frac{\sum_{m_i \in D} \sum_{w_j \in m_i} \delta'(w_j, w_i)}{n|D|} & if|D| \neq 0 \\ 1 & otherwise \end{cases}.$$

La doble suma cuenta las apariciones de w_i entre los marcadores del diccionario D, mientras que el denominador cuenta el número total de palabras en D. Así $O(w_i, D)$ es uno si w_i no está en D y tiende a cero cuando aparece un mayor número de veces. Finalmente, en la primera interacción del ciclo del algoritmo, la función $O(w_i, D)$ se definirá en 1 para que todas las palabras tengan la misma probabilidad de ser seleccionadas.

Cálculo de distancia

Como se indicó anteriormente se agregará un marcador al diccionario si su distancia a los marcadores que se encuentran en el diccionario es menor que t. El concepto de distancia entre marcadores debe definirse considerando que se imprimen como cuadrículas binarias de nxn bits que se pueden observar bajo rotación. Por lo tanto se puede definir la distancia entre dos marcadores como:

$d(m_i, m_j) = min\{H(m_i, R_k(m_j))\}/k \in \{0, 1, 2, 3\}.$

La función H es la distancia de Hamming entre dos marcadores que se define como la suma de las distancias de Hamming entre cada par de palabras de dos marcadores. La función R_k es un operador que rota el marcador $k * 90$ en el sentido de las agujas del reloj. Con esta función entonces, la distancia es invariante a la rotación entre los marcadores.

Podemos definir la distancia de un marcador m_i a un diccionario D como:

$D(m_i, \mathbf{D}) = mind(m_i, m_j) \forall m_j \in \mathbf{D}$

donde $D(m_i, \mathbf{D})$ resulta la distancia del marcador m_i al marcador más cercano en el diccionario d.

Finalmente, no solo es importante distinguir los marcadores entre sí, sino también identificar correctamente la orientación del marcador, de lo contrario la estimación de la

posición fallaría.

Por lo tanto, un marcador válido también debe garantizar que la distancia mínima a sus propias rotaciones esté por encima de t.

Se puede definir la auto distancia del marcador como:

$$S(m_i) = min\{H(m_i, R_k(m_i))\}/k \in \{0, 1, 2, 3\}$$

En resumen solo agregaremos un marcador al diccionario sí $S(m_i)$ y $D(m_i, D)$ son mayores o iguales a t. De lo contrario se rechaza el marcador y se genera uno nuevo. Después de un número de iteraciones improductivas (max_q), el valor de t se reduce en uno de modo que puedan agregarse nuevos marcadores.

Al final, los marcadores del diccionario generado tienen una distancia mínima entre ellos y consigo mismo, que será, al menos, el último t empleado. Este valor se puede calcular para cualquier diccionario de marcadores de la siguiente manera:

$$t(\mathbf{D}) = min\{min\{S(m_i)\}, min\{d(m_i, m_j)\}\}/m_i, m_j, conm_i \neq m_j, m_i, m_j \in \mathbf{D}$$

3.6. Corrección del error

El proceso se compone de varios pasos:

- Segmentación de imagen: En primer lugar, si la imagen tiene más de un canal (i.e, RGB, BGR u otro formato de color) se la convierte a una imagen de un solo canal en tonos de gris, luego se realiza un proceso de umbralización donde cada pixel que está por encima de un valor de umbral queda en 255 y el que está por debajo de ese valor de umbral pasa a 0. Finalmente, se aplica un filtro de Canny que extrae los contornos de la imagen. La figura 3.16 muestra cuál es el resultado de este proceso.

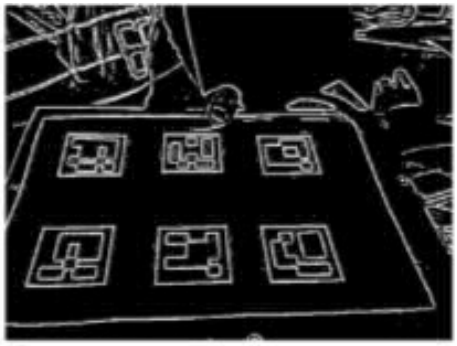

Figura 3.16: Imagen obtenida [8] luego de aplicar umbralización y filtro de Canny [23].

- Extracción y filtrado de contornos: Se realiza una extracción de contornos utilizando el algoritmo Suzuki y Abe [24]. Con esto se obtienen los contornos de la imagen candidatos a ser contornos de marcadores (Ver figura 3.17).

Figura 3.17: Imagen con la extracción y filtrado de contornos candidatos (extraída de [8]).

- A continuación se realiza una aproximación poligonal usando el algoritmo de Douglas-Peucker [25]. Dado que los marcadores están incluidos en contornos rectangulares aquellos que no se aproximan a los polígonos de cuatro vértices se descartarán. Finalmente se simplifican los contornos dejando sólo los extremos como se puede ver en la figura 3.18.

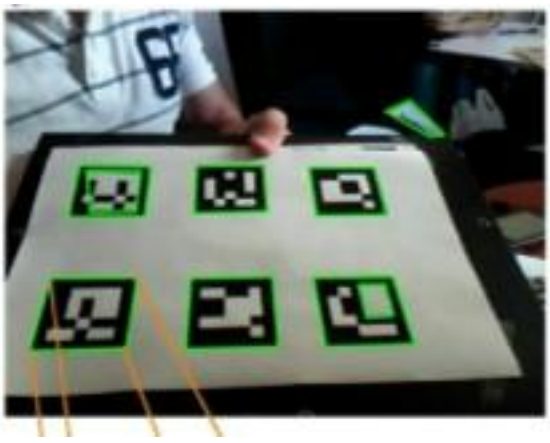

Figura 3.18: Imagen una vez filtrados los contornos quedando solo los contornos externos de los marcadores (extraído de [8]).

- El último paso consiste en analizar la región interna de estos contornos para extraer su código. Primero se elimina la proyección en perspectiva calculando la matriz como se puede observar en la figura 3.19.

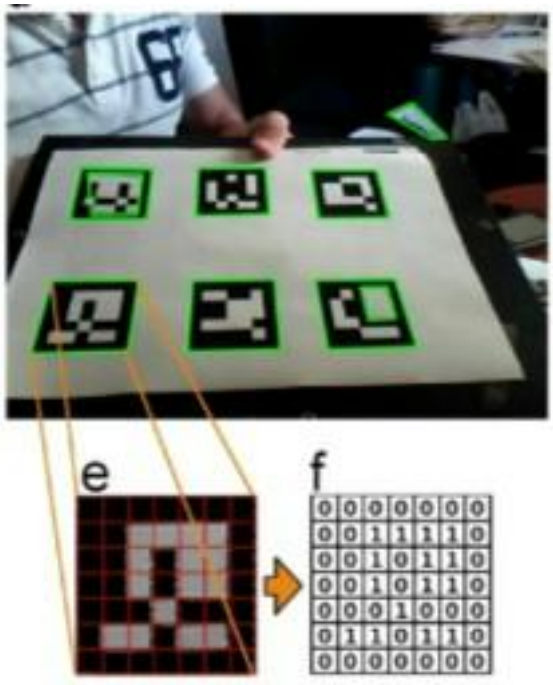

Figura 3.19: La imagen resultante está umbralizada usando el método Otsu´s [26], esto proporciona el valor de umbral óptimo dado que la imagen es bimodal por lo que la imagen binarizada se divide en una cuadrícula y a cada elemento se le asigna el valor cero o uno dependiendo de los valores de la mayoría de los píxeles.

Solamente serán admitidos los marcadores cuyos bordes sean contiguos (todos los píxeles que lo conforman tengan valor 0)

Llegado a este punto es necesario determinar cuál de los candidatos a marcador obtenidos pertenece realmente al diccionario. Una vez que el código de un marcador es extraído se obtienen cuatro identificadores diferentes (uno para cada posible rotación). Si alguno de ellos se encuentra en el diccionario D (el diccionario previamente generado) se toma como válido. Para acelerar este proceso los elementos del diccionario se ordenan como un árbol binario.

Por ello los marcadores están representados por el valor entero obtenido al concatenar todos sus bits, por lo que se puede deducir entonces que este proceso tiene una complejidad logarítmica $O(\log_2(|D|)$.

En el caso que no se haya encontrado ninguna coincidencia de un candidato a marcador con el diccionario que se intenta corregir el error cometido durante la identificación de los bits de la cuadrícula interna del marcador candidato. Teniendo en cuenta que la distancia mínima entre los marcadores cualesquiera en **D** es:

$t(\mathbf{D}) = min\{min\{S(m_i)\}, min\{\mathbf{D}(m_i, m_j)\}\}/m_i \neq m_j \in \mathbf{D}$

podremos corregir un error que como máximo sea $\frac{t-1}{2}$.

El método de corrección de marcadores consiste en calcular la distancia del candidato a marcador erróneo con todos los marcadores en el diccionario **D**: usando

$D(m_i, \mathbf{D}) = minD(m_i, m_j)/m_j \in \mathbf{D}$.

Si la distancia es igual o menor que $\frac{t-1}{2}$ consideramos que el marcador más cercano es el que se debe corregir. Sin embargo este proceso presenta una complejidad lineal $O(4|D|)$, ya que cada rotación del candidato debe compararse con todo el diccionario.

Capítulo 4

Experimentos y resultados

Como mencionamos en el capítulo 1, el uso de marcadores fiduciarios incluye problemáticas. En este capítulo mostramos las experiencias realizadas con el objetivo de dar respuesta a las siguientes de ellas:

- los parámetros que se deben conocer de una cámara para obtener la pose de un marcador,
- ¿cómo hallar esos parámetros?
- ¿cómo tratar el ruido en las imágenes?
- ¿cómo encontrar una relación adecuada entre la cantidad de marcadores, los bits de codificación de cada uno y el tamaño con que se imprimen?

4.1. Identificación de marcadores

En la identificación de marcadores fiduciarios para el reconocimiento de la pose de los objetos en la escena con sistemas de visión por computadora, se deben tener en cuenta el tipo de cámara, la resolución del CCD que ésta tiene, los parámetros que se necesitan y cómo se obtienen.

En el proceso de identificación de marcadores la cámara es muy importante, como se describe en el capítulo anterior el modelo de cámara utilizado es pin-hole y en las experiencias utilizamos una resolución de $1280x720$. También, la función que utiliza la librería Aruco para la detección de objetos en la escena $SolvePnp()$ [27] requiere de los parámetros intrínsecos de la cámara:

- la matriz de calibración (cameraMatrix)

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

donde f_x y f_y son las longitudes focales de la cámara en el eje x y en el eje y, c_x y c_y son las coordenadas del centro de la cámara (principal point) que determinan el desplazamiento del centro de la imagen.

- el vector de coeficientes de distorsión (*distCoeffs*) con 5 elementos.

Es muy importante que estos parámetros sean lo más precisos posible porque brindan el centro de coordenadas de la cámara, las distancias focales, y la corrección de la distorsión. Una incorrecta indeterminación de ellos genera una pose inexacta en la detección.

Para determinar los parámetros de la cámara realizaremos la operación de calibración usando el programa que tiene la librería Aruco para este fin. En la figura 4.1 se pueden observar los resultados del proceso de calibración (el mejor resultado de 5 repeticiones de la operación de calibración).

```
%YAML:1.0
---
image_width: 640
image_height: 480
camera_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: f
   data: [ 2.48036224e+02, 0., 3.29268646e+02, 0., 2.48543991e+02,
       2.52190704e+02, 0., 0., 1. ]
distortion_coefficients: !!opencv-matrix
   rows: 1
   cols: 5
   dt: f
   data: [ 1.03484020e-02, -4.13610265e-02, 4.90170554e-04,
       -8.96315178e-05, 8.92766286e-03 ]
```

Figura 4.1: Contenido del archivo de salida de la calibración (el de menor error de 5 repeticiones) usando el programa para tal fin provisto por la librería Aruco [21].

Dado que los valores de c_x y de c_y en la matriz de calibración parecían muy alejados de los valores esperados (cercanos a 320 y 240 respectivamente) y que además el programa que dispone la librería Aruco para la calibración no tiene una forma de trabajar con una resolución diferente a $640x480$ se realizó una nueva operación de calibración usando las funciones disponibles en la librería openCV para tal fin [22].

```
%YAML:1.0
---
image_width: 1280
image_height: 720
camera_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: f
   data: [1.41714377e+03, 0., 6.471208826e+02, 0., 1.41574110e+03,
      3.66527351e+02, 0., 0., 1.]
distortion_coefficients: !!opencv-matrix
   rows: 1
   cols: 5
   dt: f
   data: [-3.32883206e-02, 9.88403974e-01, 4.66792301e-04,
       6.74877289e-04, -3.31607882e+00]
```

Figura 4.2: Contenido del archivo de salida de la calibración (el de menor error de 5 repeticiones) usando el programa para tal fin provisto por la librería openCV [22]

En cuanto a la calibración, con el método propuesto por la librería Opencv se obtuvieron en general, mejores resultados.

4.2. Algoritmo y distancia de Hamming

La librería Aruco propone un diccionario de 250 marcadores con una codificación de 6x6 bits. Un problema referido a esto es el tamaño en el que se deben imprimir los marcadores sobre la torcha, al ser un espacio reducido, agregar marcadores para reconocer la posición de la misma hace que los bits utilizados para la codificación sean muy pequeños pudiendo haber confusión en la detección. Como consecuencia, desarrollar el algoritmo de creación de diccionarios propuesto por Garrido [19] nos permitió generar diccionarios con marcadores de distinta cantidad de bits y además, conocer la distancia de Hamming entre ellos.

En el algoritmo, la elección de marcadores se realiza con un proceso estocástico, se genera al azar un marcador y, si la distancia de Hamming de ese marcador respecto al resto de los que ya fueron agregados al diccionario es mayor a un umbral, se agrega al diccionario, caso contrario, se incrementa en uno el contador de iteraciones improductivas q hasta que llegue a la cantidad máxima max_q, entonces mientras no se llegue a este valor, aleatoriamente se generan marcadores hasta encontrar el que se puede agregar al diccionario. Creemos entonces que, si aumentamos max_q se pueden generar más marcadores candidatos y en consecuencia aumentar la distancia de Hamming entre ellos.

Para probar lo mencionado ejecutamos el programa para generar 6 diccionarios, cada uno con 8 marcadores con un tamaño de codificación $3x3$ y, en la creación aumentamos la cantidad máxima de iteraciones max_q: 20, 100, 500, 1000, 5000, 10000 y 20000 respectivamente. Registramos la distancia de Hamming obtenida en cada caso. En la tabla 4.1 figuran los resultados de los diccionarios con marcadores 3x3.

Cantidad de marcadores	Tamaño	max_q	Distancia Hamming
8	3x3	20	2
8	3x3	100	2
8	3x3	5000	2
8	3x3	1000	2
8	3x3	5000	2
8	3x3	10000	2
8	3x3	20000	2

Tabla 4.1: Registro de creación de diccionarios 3x3.

Luego repetimos la misma experiencia para diccionarios con tamaño 4x4, 5x5 y 6x6. En la tabla 4.2 figuran los resultados de diccionarios con marcadores 4x4.

Cantidad de marcadores	Tamaño	max_q	Distancia Hamming
8	4x4	20	5
8	4x4	100	6
8	4x4	5000	6
8	4x4	1000	6
8	4x4	5000	6
8	4x4	10000	6
8	4x4	20000	6

Tabla 4.2: Registro de creación de diccionarios 4x4.

En la tabla 4.3 figuran los resultados de diccionarios con marcadores 5x5.

Cantidad de marcadores	Tamaño	max_q	Distancia Hamming
8	5x5	20	9
8	5x5	100	10
8	5x5	5000	10
8	5x5	1000	10
8	5x5	5000	10
8	5x5	10000	10
8	5x5	20000	10

Tabla 4.3: Registro de creación de diccionarios 5x5.

En la tabla 4.4 figuran los resultados de diccionarios con marcadores 6x6.

Cantidad de marcadores	Tamaño	max_q	Distancia Hamming
8	6x6	20	14
8	6x6	100	14
8	6x6	5000	15
8	6x6	1000	15
8	6x6	5000	15
8	6x6	10000	15
8	6x6	20000	15

Tabla 4.4: Registro de creación de diccionarios 6x6.

Luego de esta primera experiencia no podemos concluir que al aumentar el máximo de cantidad de iteraciones, aumente la distancia de Hamming. Sin embargo, podemos concluir que los marcadores $3x3$ no son convenientes puesto que su distancia de Hamming es muy baja, el algoritmo de corrección de errores durante la identificación de cada marcador solo puede corregir 1 bit y por lo tanto, puede conducir a confusión durante la identificación.

4.3. Elección del tamaño de marcador

Habiendo descartado los marcadores $3x3$, nos proponemos encontrar cuál es la codificación que se necesita para la detección de marcadores. Sabemos que los objetos que intervienen en la escena (probeta y torcha) se encuentran a una distancia entre 30 y 60 cm y, que la cantidad de marcadores necesarios para la simulación es aproximadamente 80. Con esta información realizamos el siguiente experimento.

El objetivo entonces es, encontrar la codificación que debe tener los marcadores $4x4$, $5x5$, etc. También, buscamos encontrar el tamaño de dichos marcadores (en cm) más chico posible, el motivo es que mientras más pequeño es el marcador, mayor cantidad de ellos se puede utilizar. Sin embargo, no es tan sencillo porque mientras más pequeña es la impresión, más pequeños son los bits (cuadraditos) y eso puede llevar a confusión e incluso a la no detección, entonces ¿cuál es el equilibrio entra la codificación en bits respecto al tamaño de impresión? Iniciamos la experiencia generando dos diccionarios, uno con una codificación de $4x4$ y otra de $5x5$ con 80 marcadores cada uno. De cada diccionario se seleccionaron 40 marcadores (al azar, sin discriminar la distancia de Hamming) y a partir de ellos, se armaron 4 tableros.

- tablero 1: conformado con 40 marcadores $4x4$ impresos en un tamaño de 1 cm de lado,

- tablero 2: conformado con 40 marcadores $4x4$ impresos en un tamaño de 2 cm de lado,
- tablero 3: conformado con 40 marcadores $5x5$ impresos en un tamaño de 1 cm de lado y,
- tablero 4: conformado con 40 marcadores $5x5$ impresos en un tamaño de 2 cm de lado,

Durante la experiencia, la cámara y el tablero estuvieron en posiciones fijas. Las distancias usadas entre estos elementos fueron de 30 cm y 60 cm. Mientras que el ángulo entre la cámara y el tablero fue de 45°. El objetivo es ver si en la distancia más corta (30 cm) y en la más larga (60 cm) los tableros son detectados, y en caso que así sea, si los reconoce correctamente. En las figuras 4.3 y 4.4 se muestra las capturas de pantalla para marcadores $4x4$ impresos con un tamaño de 1 cm.

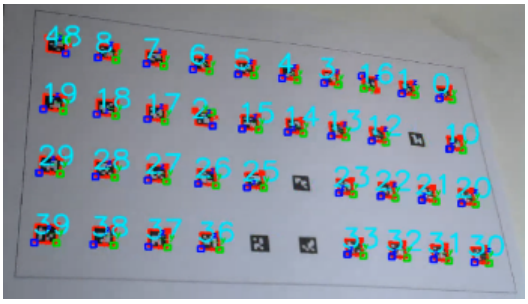

Figura 4.3: Captura de pantalla del tablero 1 con los resultados de los marcadores $4x4$ a una distancia de 30 cm.

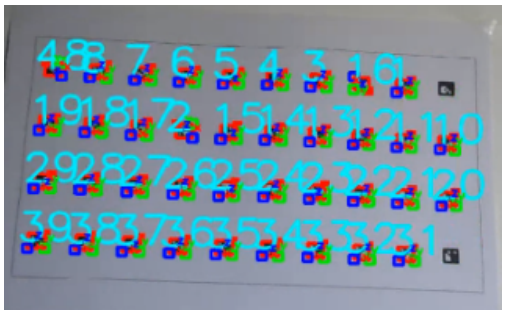

Figura 4.4: Captura de pantalla del tablero 1 con los resultados de los marcadores $4x4$ a una distancia de 60 cm.

Se puede observar que en ambas distancias de la cámara con el tablero 1 hay marcadores que no detecta y también se produce confusión, el marcador que en las imágenes se ve arriba a la izquierda lo reconoce como el marcador 48 mientras que el mismo es el 9. También confunde el 2 con el 16 y viceversa.

Repetimos la experiencia pero con el tablero 2. Los resultados se pueden observar en las figuras 4.5 y 4.6.

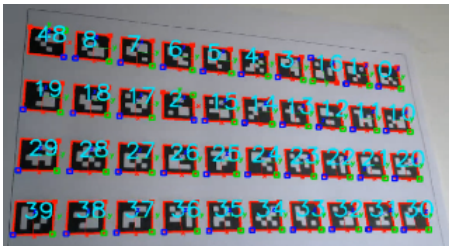

Figura 4.5: Captura de pantalla del tablero 2 (marcadores $4x4$ a una distancia de 30 cm).

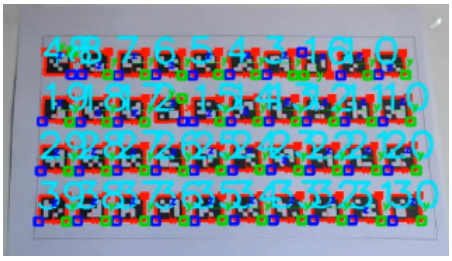

Figura 4.6: Captura de pantalla del tablero 2 (marcadores $4x4$ a una distancia de 60 cm).

Con los marcadores del tablero 2, no hubo problemas de detección, aunque el problema de la confusión continúa.

Probamos ahora si los marcadores del tablero 1 no fueron detectado por ser de 1 cm, o bien por ser de una codificación $4x4$. Para ello, repetimos la prueba con el tablero 3 que utiliza marcadores $5x5$ de 1 cm, los resultados en las figuras 4.7 y 4.8

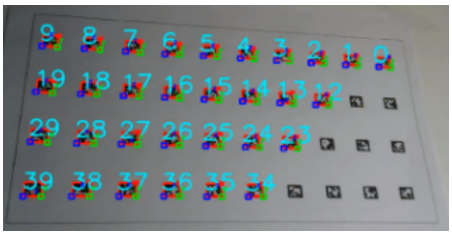

Figura 4.7: Captura de pantalla del tablero 3 con los resultados de los marcadores $5x5$ a una distancia de 30 cm.

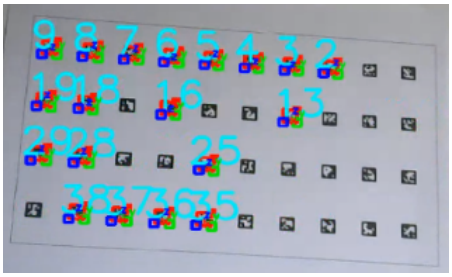

Figura 4.8: Captura de pantalla del tablero 3 con los resultados de los marcadores $5x5$ a una distancia de 60 cm.

En esta parte del experimento se observan dos cosas, la cantidad de marcadores que no reconoce es mayor a los del tablero 1, incluso a una distancia de 60 cm reconoce menos de la mitad. No obstante, los que detecta, no los confunde. Probamos ahora con el tablero 4, los resultados en las figuras 4.9 y 4.10.

Figura 4.9: Captura de pantalla del tablero 4 con los resultados de los marcadores $5x5$ a una distancia de 30 cm.

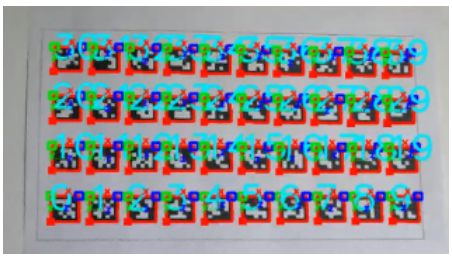

Figura 4.10: Captura de pantalla del tablero 4 con los resultados de los marcadores $5x5$ a una distancia de 60 cm.

En este último ensayo, encontramos que en el tablero 4 se reconocen todos los marcadores en ambas distancias y además, no los confunde.

4.4. Oclusión y análisis de error

Habiendo encontrado el tamaño y la codificación de los marcadores a utilizar, queda por resolver el inconveniente de la oclusión, el ruido y la ambigüedad en el eje z. Separamos este ensayo en tres partes, en la primera buscamos la solución del ruido, en la segunda, la oclusión y en la tercera la ambigüedad.

El ruido es un problema en la detección como explicamos anteriormente, la solución a esto es agregar una cantidad mayor de marcadores para solucionarlo, entonces ¿cuál es esa cantidad?.

Para realizar este ensayo generamos 4 tableros con marcadores de $5x5$ impresos en un tamaño de 2 cm de lado. Los tableros tienen cada uno 5, 10, 20 y 40 marcadores respectivamente. En la figura 4.11 se puede observar la distribución de marcadores en el primer tablero con las medidas hacia los bordes para llevar a cabo el experimento.

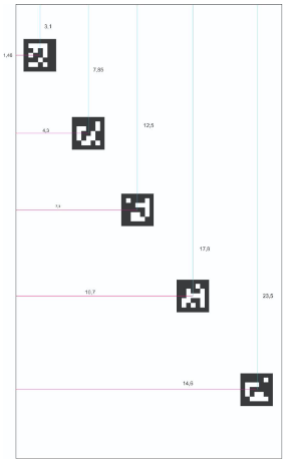

Figura 4.11: Imagen del tablero de 5 marcadores con las medidas hacia los bordes utilizado para el experimento.

Las medidas en cada marcador a los bordes son las coordenadas del marcador. A partir de dichas medidas establecemos la posición de cuatro vértices en el espacio 3D de la escena para dibujar un rectángulo sobre la imagen 2D basado en la proyección de dichos vértices (coincidentes con los vértice de la hoja, la esquina superior derecha es el vértice 1, el resto, en sentido horario. Por ejemplo el vértice 2 es la esquina inferior derecha). La proyección se calcula a partir del vector de traslación y la matriz de rotación de un marcador. En la experiencia ese cálculo lo realizamos tomando el promedio obtenido a partir de cada marcador sobre 500 imágenes (cuadros de video). La cámara y cada tablero se dejan fijos a una distancia de 60 cm, y se repitió la experiencia 5 veces para cada uno de los tableros.

Para tener una medida de error y poder evaluar el ruido, tomamos dos medidas diferentes, la distancia máxima DM (es la mayor distancia entre la proyección de un vértice respecto al promedio de dichas proyecciones) y la desviación estándar. En la figura 42 se puede observar el tablero que tiene 5 marcadores, con el rectángulo dibujado a partir del promedio de proyecciones.

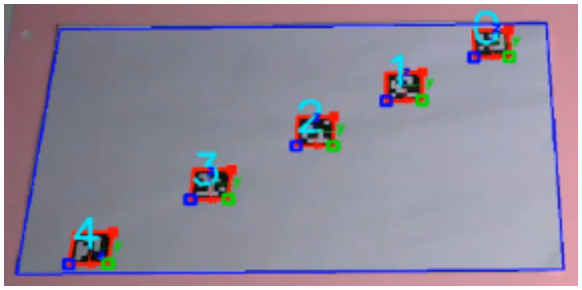

Figura 4.12: Captura de pantalla con el rectángulo dibujado.

Cada experiencia se registra en una tabla como la siguiente.

Vértice 1		Vértice 2		Vértice 3		Vértice 4	
DM	DE	DM	DE	DM	DE	DM	DE

Donde en cada línea se registra una experiencia con:

- DM es la distancia máxima de una proyección respecto al promedio.
- DE es el desvío estándar de la proyección.

En la tabla 4.5 presentamos los resultados de las experiencias realizadas con el tablero de 5 marcadores.

Vértice 1		Vértice 2		Vértice 3		Vértice 4	
DM	DE	DM	DE	DM	DE	DM	DE
0,92	0,4	1,61	0,69	1,04	0,39	1,25	0,55
1,37	0,52	3,22	0,77	2,24	0,53	3,63	0,97
1,54	0,44	3,06	0,74	2,54	0,81	2,66	0,98
1,76	0,62	3,49	0,79	1,65	0,46	3,41	1,24
1,57	0,54	4,15	0,9	2,17	0,65	4,15	1,45
1,432	0,504	3,106	0,778	1,928	0,568	3,02	1,038

Tabla 4.5: En la tabla se muestra un resumen del experimento con el tablero de 5 marcadores, cada línea es el resultado de tomar 500 imágenes mientras que la última, es el promedio de todas los registros de la tabla. Tanto la DM como la DE están expresadas en píxeles.

Repetimos la experiencia ahora con el tablero de 10 marcadores. El dibujo del rectángulo se puede observar en la figura 4.13.

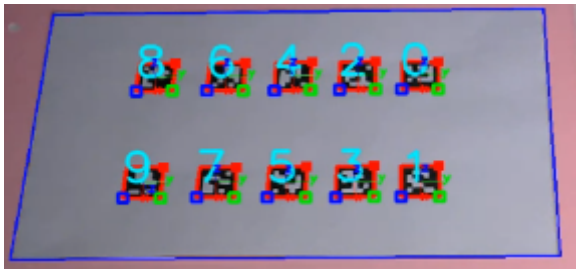

Figura 4.13: Captura de pantalla con el rectángulo dibujado.

En la tabla 4.6 presentamos la experiencia realizada con el tablero de 10 marcadores.

Vértice 1		Vértice 2		Vértice 3		Vértice 4	
DM	DE	DM	DE	DM	DE	DM	DE
1,47	0,36	1,2	0,39	2,14	0,61	1,87	0,89
0,860	0,29	0,86	0,36	1,56	0,38	2,35	0,71
1,22	0,33	0,83	0,32	0,97	0,33	1,74	0,48
0,75	0,26	0,96	0,28	1,72	0,34	2,43	0,53
1,52	0,29	1,04	0,29	2,43	0,41	2,85	0,56
1,164	0,306	0,978	0,328	1,764	0,414	2,248	0,634

Tabla 4.6: En la tabla se muestra un resumen de la experiencia realizada con el tablero de 10 marcadores. Cada línea es el resultado de tomar 500 imágenes mientras que la última, es el promedio de todas los registros de la tabla. Tanto la DM como la DE están expresadas en píxeles.

Nuevamente, llevamos a cabo la experiencia pero en este caso con un tablero de 20 marcadores. El dibujo del rectángulo se puede observar en la figura 4.14.

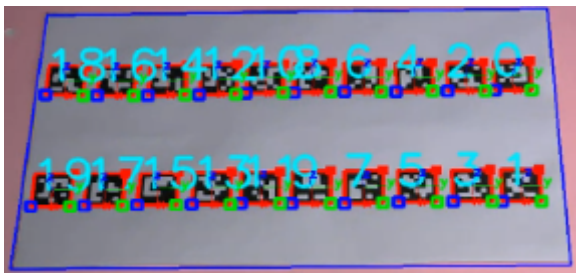

Figura 4.14: Captura de pantalla con el rectángulo dibujado.

Los resultados del tablero de 20 marcadores figuran en la tabla 4.7.

Vértice 1		Vértice 2		Vértice 3		Vértice 4	
DM	DE	DM	DE	DM	DE	DM	DE
0,98	0,27	0,86	0,29	0,72	0,25	0,86	0,29
0,93	0,3	0,99	0,33	1,06	0,39	1,24	0,4
0,96	0,32	0,67	0,28	0,8	0,27	0,99	0,32
1,03	0,28	0,88	0,33	0,73	0,23	0,66	0,26
0,97	0,26	0,8	0,26	0,94	0,25	0,88	0,29
0,976	0,288	0,84	0,298	0,85	0,278	0,926	0,312

Tabla 4.7: En la tabla se muestra un resumen de la experiencia con el tablero de 20 marcadores. Cada línea es el resultado de tomar 500 imágenes mientras que la última, es el promedio de todas los registros de la tabla.

Finalmente, realizamos nuevamente la experiencia pero en este caso con un tablero de 40 marcadores. En la figura 4.15 se puede observar el dibujo del rectángulo sobre la imagen.

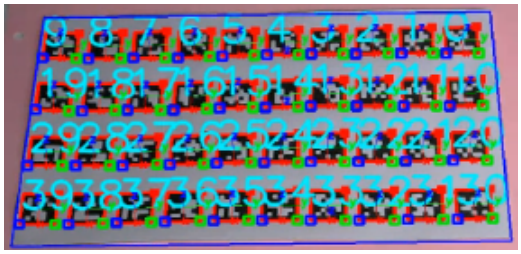

Figura 4.15: Captura de pantalla con el rectángulo dibujado.

En la tabla 4.8 se muestran los resultados.

Vértice 1		Vértice 2		Vértice 3		Vértice 4	
DM	DE	DM	DE	DM	DE	DM	DE
0,77	0,23	0,68	0,21	0,98	0,25,	0,75	0,24
0,64	0,19	0,56	0,17	0,51	0,17	0,58	0,19
0,62	0,25	0,67	0,24	0,46	0,18	0,57	0,2
0,68	0,23	0,68	0,21	0,47	0,17	0,78	0,19
0,79	0,25	0,65	0,23	0,63	0,19	0,77	0,22
0,7	0,23	0,648	0,212	0,61	0,192	0,69	0,208

Tabla 4.8: En la tabla se muestra un resumen de la experiencia con el tablero de 40 marcadores. Cada línea es el resultado de tomar 500 imágenes mientras que la última, es el promedio de todas los registros de la tabla. Tanto la DM como la DE están expresadas en píxeles.

Una vez realizadas las cuatro experiencias recién descritas en las cuales se mantuvo una distancia entre la c´amara y el tablero de 60 cm, se repitió la misma serie pero con una distancia entre la c´amara y el tablero de 30 cm.

En la tabla 4.9 volcamos sólo el promedio de los resultados para cada tablero.

Tablero	Vértice 1		Vértice 2		Vértice 3		Vértice 4	
	DM	DE	DM	DE	DM	DE	DM	DE
5 marcadores	1,444	0,534	2,286	0,704	3,226	0,966	2,446	0,71
10 marcadores	1,248	0,472	1,178	0,428	2,586	0,62	1,806	0,548
20 marcadores	1,028	0,306	0,998	0,328	1,294	0,416	1,114	0,38
40 marcadores	0,726	0,232	0,676	0,23	1,026	0,404	0,768	0,284

Tabla 4.9: En la tabla se muestra un resumen del experimento con los 4 tableros a una distancia de 30 cm entre la cámara y los tableros, cada línea es el resultado promedio de todos los registros de cada tablero. Tanto la DM como la DE están expresadas en píxeles.

Agregando más marcadores, el ruido en las imágenes disminuye, en las figuras 4.16, 4.17, 4.18 y 4.19 se ven gráficos de líneas que muestran cómo se comporta el error al aumentar la cantidad de marcadores en cada uno de los vértices en las experiencias con la cámara a 60 cm de los tableros.

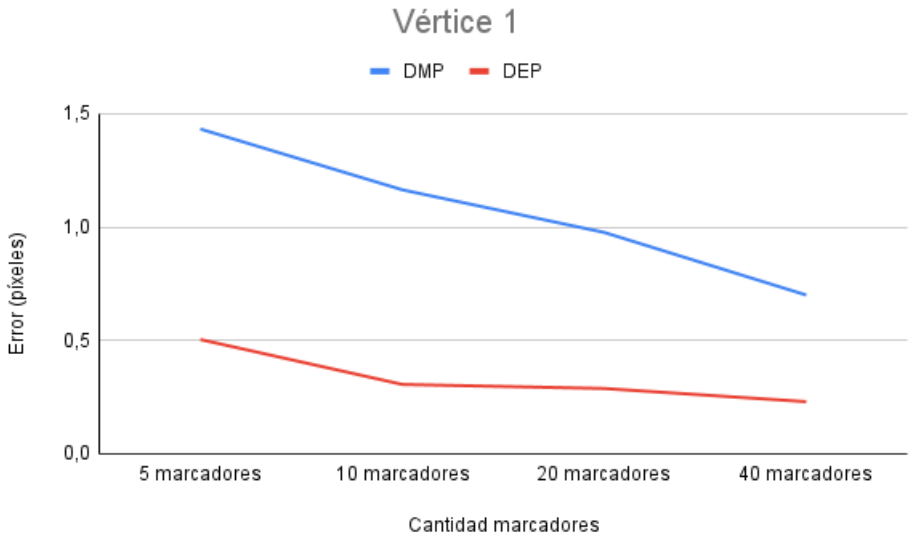

Figura 4.16: Distancia en píxeles vs cantidad de marcadores en los tableros. El trazo azul corresponde a la distancia máxima respecto al promedio y el trazo rojo corresponde a la desviación estándar.

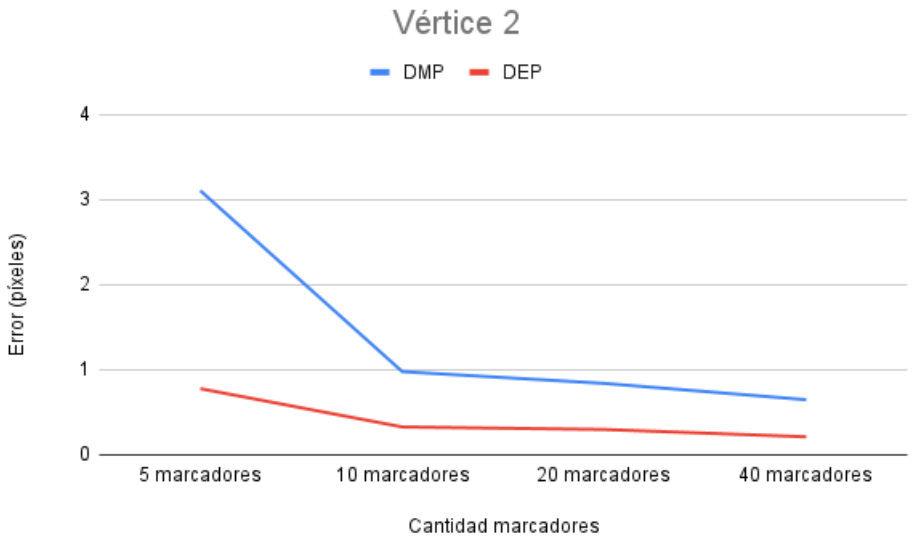

Figura 4.17: Distancia en píxeles vs cantidad de marcadores en los tableros. El trazo azul corresponde a la distancia máxima respecto al promedio y el trazo rojo corresponde a la desviación estándar.

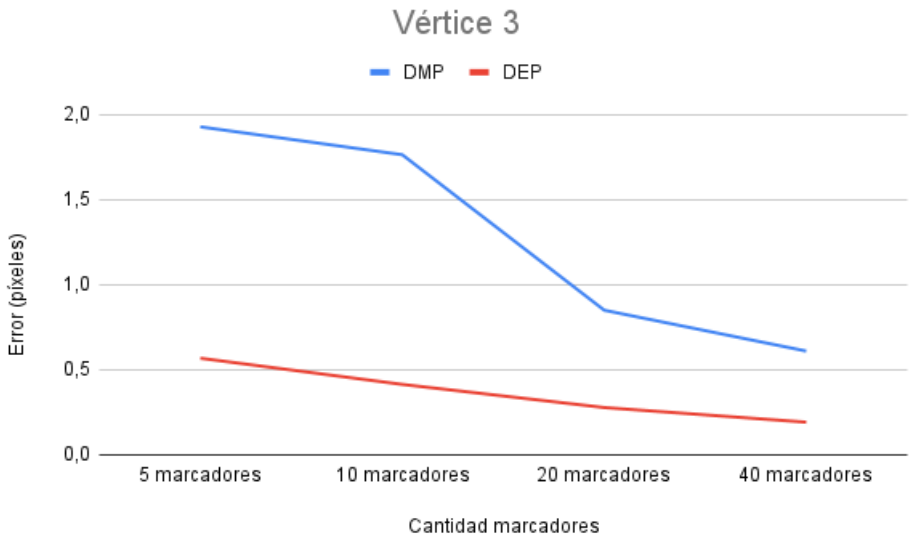

Figura 4.18: Distancia en píxeles vs cantidad de marcadores en los tableros. El trazo azul corresponde a la distancia máxima respecto al promedio y el trazo rojo corresponde a la desviación estándar.

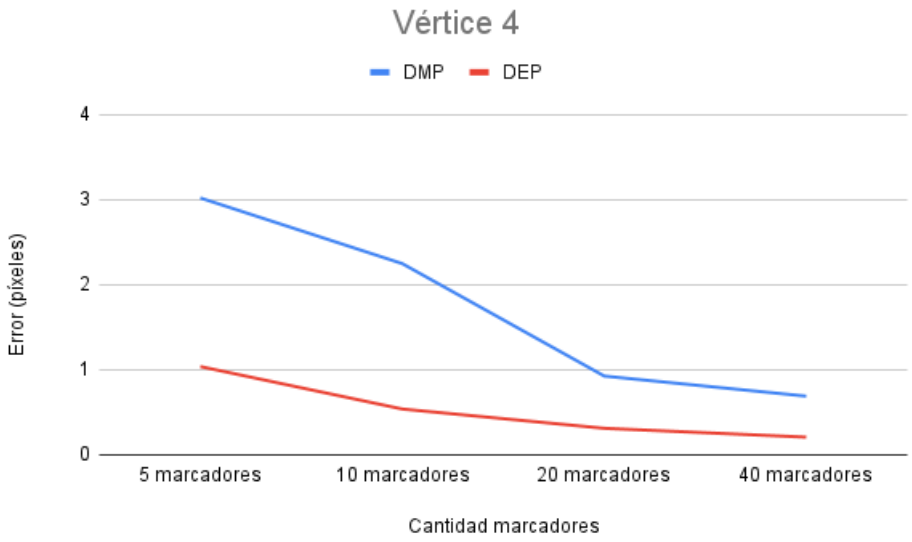

Figura 4.19: Distancia en píxeles vs cantidad de marcadores en los tableros. El trazo azul corresponde a la distancia máxima respecto al promedio y el trazo rojo corresponde a la desviación estándar.

Durante los ensayos hemos notado que si bien la oclusión es un inconveniente en un marcador, cuando se agregan varios, la oclusión de un grupo de marcadores no afecta la ubicación de la traza del rectángulo. Se puede observar en la figura 49. La oclusión entonces, se supera agregando marcadores.

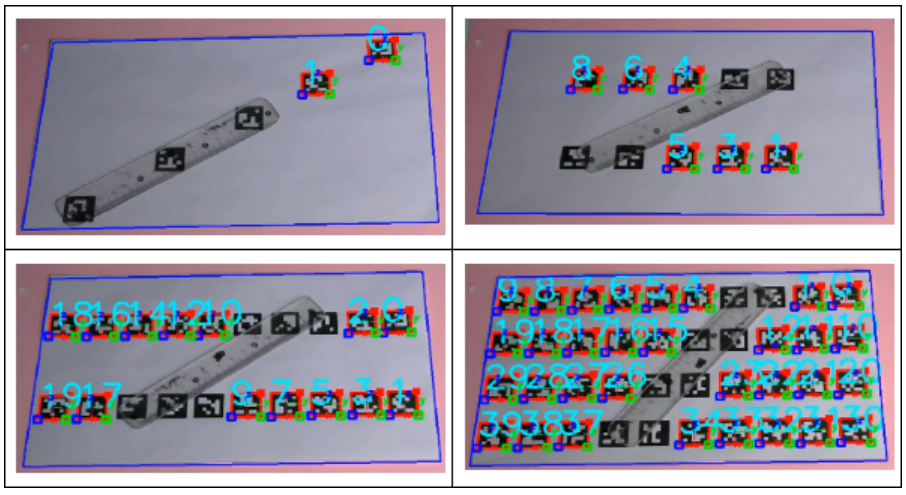

Figura 4.20: Imagen de los tablero con 5, 10, 20 y 40 marcadores. Pese a la oclusión de algunos de los marcadores, la traza del rectángulo sigue al borde del tablero.

Las diversas series de experiencias reportadas en este capítulo requirieron tratar con el problema de la ambigüedad en la determinación del eje z durante el cálculo de la matriz de rotación.
Nuestra propuesta es usar un algoritmo que separa en dos grupos a los marcadores (en el tablero) según la orientación del eje z. A partir de ahí, se usan las matrices del grupo de mayor cardinal. Este enfoque tiene resultados similares a los que surgen de ocluir un subconjunto de marcadores del tablero que, como ya hemos visto, no afecta sensiblemente el cálculo de la proyección en el plano 2D. En la figura 50 se puede observar la diferencia al dibujar el rectángulo del borde del tablero usando el conjunto de marcadores que tienen el eje z correcto (izquierda) y usando todos los marcadores, independientemente tengan el eje z calculado correctamente o no (derecha).
A continuación delineamos los aspectos más relevantes a tener en cuenta para la implementación del algoritmo propuesto. Sea $VectCoeff$ el vector de coeficientes de distorsión de la cámara y $MatCam$ la matriz de la cámara.
Sea *marcadores* una lista de marcadores donde cada *marcador* en ella contiene los siguientes atributos: $MatRot$ la matriz de rotación del marcador, $VecTrasl$ el vector de traslación del marcador e *id* el identificador de marcador en el diccionario.
Sea n la cantidad de marcadores en el tablero.
Sea *vertices* un arreglo de n 4-tuplas donde, en cada tupla están las ubicaciones de los vértices en 3D: izquierdo superior, derecho superior, derecho inferior e izquierdo inferior del tablero para cada marcador. El arreglo es indicado por el identificador del marcador en el diccionario.

Algoritmo
1 while (nueva *imagen*)
2 $ctd_Neg \leftarrow ctd_Pos \leftarrow 0$
3 $Vert2D_Acum_Pos \leftarrow Vert2D_Acum_Neg \leftarrow \varnothing$
4 $marcadores \leftarrow$ DetectarMarcadores($imagen$)
5 $MatRot_ref \leftarrow marcadores[0].MatRot$
6 for each *marc* en la lista *marcadores* detectados
7 $Vert2D_marc \leftarrow$ proyectarEn2D($vertices[marc.id]$), $marc.MatRot$, $marc.VectTrasl$, $VectCoeff$, $MatCam$)
8 AlmacenarVértices2D($marc, Vert2D_Marc, Vert2D_Acum_Pos, Vert2D_Acum_Neg$, ctd_Pos, ctd_Neg, $MatRot_ref$)
9 if ($ctd_Pos \geqslant ctd_Neg$)
10 $vert_rectangulo \leftarrow$ Promediar($Vert2D_Acum_Pos$, ctd_Pos)
11 else
12 $vert_rectangulo \leftarrow$ Promediar($Vert2D_Acum_Neg$, ctd_Neg)
13 ...
Fin

y procede de la siguiente manera. Mientras haya alguna imagen para procesar, se inicia-

lizan los contadores de matrices con eje z positivo y con eje z negativo (paso 2).
Luego se inicializan las listas de 4-tuplas donde cada elemento de una tupla es un vértice 2D $Vert2D_Acum_Pos$ y $Vert2D_Acum_Neg$ (paso 3).
La función DetectarMarcadores() implementada en la librería de Aruco, recibe como argumento la imagen capturada y devuelve la lista de marcadores detectados (paso 4).
En la el paso 5 se guarda como referencia la matriz de rotación correspondiente al primer marcador detectado para luego ser usada en la función AlmacenarVertices2D().
En el paso 6 se proyectan los vértices 3D del tablero, correspondientes al marcador *marc*, sobre la imagen 2D y se guardan en la variable $Vert2D_marc$. Luego la función AlmacenarVértices2D() recibe como argumentos por valor a *marc* y $Vert2D_marc$, y por referencia a las variables $Vert2D_Acum_Pos$, $Vert2D_Acum_Neg$, ctd_Pos, ctd_Neg que serán actualizadas de acuerdo a si la matriz de rotación de *marc* coincide en el signo del eje z con la matriz de referencia $MatRot_ref$ o no. Si hay coincidencia agrega en $Vert2D_Acum_Pos$ las proyecciones calculadas para los vértices de *marc* que habían sido pasadas en $Vert2D_marc$ e incrementa ctd_Pos. Caso contrario, lo hace con las variables $Vert2D_Acum_Neg$ y ctd_Neg.
Al finalizar el ciclo (paso 9) guarda en la variable *vert_rectangulo* el promedio de $Vert2D_Acum_Pos$ calculado por la función Promediar() si $ctd_Pos \geqslant ctd_Neg$ (paso 10). Caso contrario, guarda el promedio de $Vert2D_Acum_Neg$ (paso 12).
Finalmente, los puntos suspensivos del paso 13 indican que, a partir de ese momento, se puede operar con los vértices del rectángulo, por ejemplo, dibujándolos sobre la imagen tal como hemos visto en varias figuras más arriba.
En lo que sigue, se detalla el pseudocódigo de las funciones AlmacenarVértices2D() y Promediar().

AlmacenarVértices2D(*marc*, $Vert2D_Marc$, $Vert2D_Acum_Pos$, $Vert2D_Acum_Neg$, ctd_Pos, ctd_Neg, $MatRot_ref$)
Comenzar
1 if (el signo del eje Z de $marc.MatRot$ coincide con el signo para $MatRot_ref$)
2 Agregar $Vert2D_Marc$ a $Vert2D_Acum_Pos$
3 Incrementar ctd_Pos
4 else
5 Agregar $Vert2D_Marc$ a $Vert2D_Acum_Neg$
6 Incrementar ctd_Neg
Fin

Promediar($Lista_Vert2D$, *ctd*)
Comenzar
1 Inicializar *suma_tuplas* en 0 2 for (cada tupla de $Lista_Vert2D$)
3 Acumular en *suma_tuplas* el valor de la tupla
4 Obtener el promedio como *suma_tuplas* / *ctd*
Fin

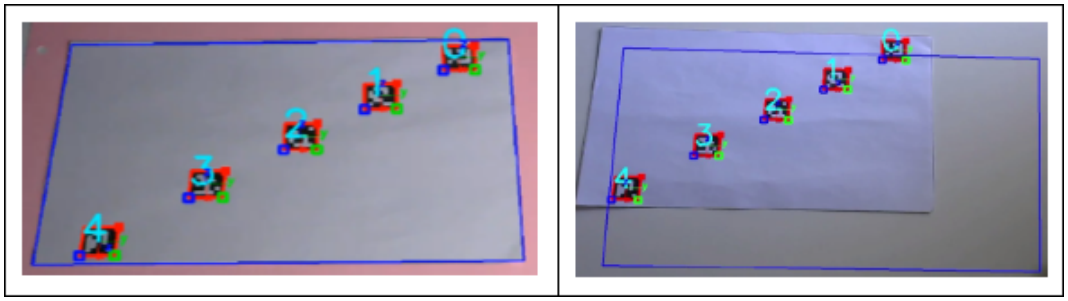

Figura 4.21: Diferencia al dibujar el rectángulo del borde del tablero usando el conjunto de marcadores que tienen el eje z correcto (izquierda) y usando todos los marcadores, independientemente tengan el eje z calculado correctamente o no (derecha).

.

Capítulo 5

Conclusiones

En el desarrollo del trabajo de la tesis, en un primer momento, no fue un motivo de urgencia ver la cámara que utilizamos, comenzamos con una que tiene una resolución de 640x480 y con ella, los marcadores los reconocía aunque el ruido era un problema. Sin embargo, al trabajar con otra cámara de distinta resolución notamos diferencias. A partir de ese momento dejamos de utilizar la resolución de $640x480$ para pasar a una de $1280x720$ si bien esto mejoraba la identificación y el ruido disminuía, aún persistía. Fue entonces que realizamos una calibración que buscara el menor error. Con estas características, no fue suficiente pero si notamos las primeras mejoras.

Como mencionamos en el capítulo anterior iniciamos la calibración con el método propuesto por la librería Aruco [21], los resultados obtenidos no eran los mejores a pesar de aumentar la cantidad de imágenes. Por este motivo buscamos utilizar otro método de calibración, fue entonces que realizamos la propuesta de la librería openCV [22], y el resultado que obtuvimos fue que con el segundo método (openCV) los resultados fueron mejores.

Para realizar un proyecto en realidad aumentada utilizando marcadores fiduciarios para la detección de la pose de los objetos en la escena recomendamos tomarse el tiempo necesario para lograr una buena calibración porque es un aspecto sumamente importante.

En cuanto a los marcadores, mostramos los resultados de las experiencias, aunque no mencionamos en la mismas la importancia que tiene por ejemplo, el estado de los tableros. Esto se debe a que si no se encuentran perfectamente planos (eran hojas), puede llevar a errores debido que la matriz de rotación de los marcadores es la misma por encontrarse todos dentro del mismo plano, entonces si hay una imperfección puede haber marcadores que tengan una matriz distinta produciendo errores.

En el experimento para dibujar el rectángulo pudimos ver el problema de la imperfección en el tablero (cuando no estaba completamente plano) como así también la ambigüedad del eje z.

5.1. Artículos derivados de este trabajo

Hemos escrito un artículo sobre el uso de marcadores fiduciarios para la obtención de la posición y orientación de objetos en la escena para un simulador de soldadura usando realidad aumentada, en el marco del proyecto "Desarrollo de un prototipo de Simulador de Soldadura para Capacitación en Argentina". El artículo se realizó conjuntamente con los siguientes autores: Fernando Puricelli, Leonardo Motillo, Agustín Pérez, Guillermo Zarate, Juan Miguel Santos. Fue enviado (y aceptado) al WICC 2024 (XXVI Workshop de Investigadores en Ciencias de la Computación) a realizarse el 18 y 19 de abril del corriente año para la presentación en la modalidad de póster.

5.2. Trabajos futuros

A partir del trabajo realizado han surgido ideas y preguntas que pueden formar parte de líneas futuras de investigación. Por ejemplo:

- Desarrollo de un sistema propio de marcadores fiduciarios optimizados para el uso necesario en el proyecto. Esto creemos que es importante ya que la librería aruco es muy dependiente del ruido y con la nueva librería se podría contemplar que no tenga esa dependencia.
- Trabajar sobre la resolución analítica de la determinación de la matriz de rotación para salvar el problema de la ambigüedad que mencionamos en el eje Z cuando se detectan los marcadores. [27]
- En este trabajo no se abordó en profundidad los aspectos vinculados con la iluminación del entorno del simulador. Por ello consideramos que podría ser de utilidad proveer de un sistema de iluminación que pudiera graduarse de forma automática para mantener la eficacia en la determinación de las poses de los marcadores fiduciarios.
- Evaluar otra geometría o forma de marcador fiduciario que no haya sido contemplada previamente. Por ejemplo considerando el color.

5.3. Conclusiones finales

Con las características mencionadas y con las experiencias realizadas podemos sacar las siguientes conclusiones:

- utilizar una cámara que tenga modelo pin-hole,
- la misma debe tener un mínimo de resolución 1280x720 y es súmamente que tenga una buena calibración,

- el tamaño en bits del marcador dependerá del proyecto, en nuestro caso para un diccionario de aproximadamente 100 marcadores sugerimos que sean de 5x5
- el tamaño de impresión debe ser de 2 cm,
- la superficie donde se encuentra el marcador debe ser plano, cualquier imperfección puede afectar el correcto funcionamiento, y
- debe haber una buena iluminación.

Desde ya, nuestra propuesta es bajo esas características en cuanto a la cámara y los marcadores, en caso de que se necesite una cantidad mayor o menor de marcadores, distancias más cortas o más grandes, sugerimos hacer las pruebas necesarias para encontrar qué tamaño deben tener los marcadores e incluso utilizar otro tipo de marcadores o métodos. La RA se puede abordar con distintas técnicas, no hay mejores o peores sino la adecuada para el proyecto.

Bibliografía

[1] Olena Lavrentieva, Ihor Arkhypov, Olexander Kuchma, and Aleksandr Uchitel. Use of simulators together with virtual and augmented reality in the system of welders' vocational training: past, present, and future. 2020.

[2] Richard Stone, Kristopher Watts, and Peihan Zhong. Virtual reality integrated welder training. 2011.

[3] RedLab. Redlab simulator, 2024. Acceso el día 8 del mes marzo, 2024.

[4] RedLab. Simbott simulator, 2024. Acceso el día 8 del mes marzo, 2024.

[5] Maquisol simulator.

[6] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pages 85–94. IEEE, 1999.

[7] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596. IEEE, 2005.

[8] Rafael Muñoz Salinas. Aruco: An efficient library for detection of planar markers and camera pose estimation, 2019.

[9] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2015.

[10] Javier Santana-Fernández, Jaime Gómez-Gil, and Laura del Pozo-San-Cirilo. Design and implementation of a gps guidance system for agricultural tractors using augmented reality technology. *Sensors*, 10(11):10435–10447, 2010.

[11] Zsolt Bálint, Botond Kiss, Beáta Magyari, and Károly Simon. Augmented reality and image recognition based framework for treasure hunt games. In *2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics*, pages 147–152. IEEE, 2012.

[12] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Markerless augmented reality with a real-time affine region tracker. In *Proceedings IEEE and ACM International Symposium on Augmented Reality*, pages 87–96. IEEE, 2001.

[13] Roberto Pierdicca, Emanuele Frontoni, Rama Pollini, Matteo Trani, and Lorenzo Verdini. The use of augmented reality glasses for the application in industry 4.0. In *Augmented Reality, Virtual Reality, and Computer Graphics: 4th International Conference, AVR 2017, Ugento, Italy, June 12-15, 2017, Proceedings, Part I 4*, pages 389–401. Springer, 2017.

[14] Leonid Naimark and Eric Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings. International Symposium on Mixed and Augmented Reality*, pages 27–36. IEEE, 2002.

[15] Ross Bencina, Martin Kaltenbrunner, and Sergi Jorda. Improved topological fiducial tracking in the reactivision system. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, pages 99–99. IEEE, 2005.

[16] Jun Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. In *Proceedings. 3rd Asia Pacific Computer Human Interaction (Cat. No. 98EX110)*, pages 63–68. IEEE, 1998.

[17] Daniel Wagner and Dieter Schmalstieg. Artoolkitplus for pose tracking on mobile devices. 2007.

[18] Mark Fiala. Comparing artag and artoolkit plus fiducial marker systems. In *IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, pages 6–pp. IEEE, 2005.

[19] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[20] José Javier Alcalde Sanz. Diseño de un protocolo de calibración de cámaras estéreo. B.S. thesis, 2014.

[21] Muñoz. Librería aruco, 2024. Acceso el día 8 del mes marzo, 2024.

[22] OpenCV. Camera calibration, 2024. Acceso el día 23 del mes febrero, 2024.

[23] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[24] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[25] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973.

[26] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[27] OpenCV. Solvepnp(), 2024. Acceso el día 23 del mes febrero, 2024.