

enCuadro

Recorrido interactivo en museos con realidad aumentada sobre dispositivos móviles

Juan Braun, Martín Etchart, Pablo Flores y Mauricio González

Resumen—La presente publicación es una breve síntesis de la investigación realizada y los resultados obtenidos durante el proyecto de fin de carrera llamado enCuadro. Aquí se resumen las distintas partes de la guía interactiva para museos que comprende: navegación dentro del museo, identificación de obras y realidad aumentada. También se explican algunos algoritmos elegidos que son base para la solución planteada que logró buenos resultados de realidad aumentada a través del procesamiento de imágenes en tiempo real. Finalmente se hace una breve demostración de algunos de los casos de uso más interesantes que se desarrollaron durante el proyecto.

Keywords—*realidad aumentada, procesamiento, tiempo real, imagen, dispositivo.*

I. INTRODUCCIÓN

Debido a la creciente disponibilidad de las plataformas móviles y el gran poder de procesamiento con el que cuentan, el número de aplicaciones móviles ha crecido de manera significativa en los últimos años. Dichas plataformas cuentan con sistemas de adquisición de audio, video y una variedad de sensores como por ejemplo acelerómetro y giroscopio, lo que las transforma en sistemas ideales para desarrollar aplicaciones de procesamiento multimedia.

Por otro lado, desde hace algunos años varios museos de distintas partes del mundo han comenzado a considerar este tipo de dispositivos, y otras tantas tecnologías, como una alternativa muy interesante para brindar un valor agregado al usuario. Proyecciones de imágenes y videos, recorridos interactivos y aplicaciones de *realidad aumentada* son tan sólo algunos de los ejemplos. Sin embargo, esta es un área muy reciente y en la que todavía queda un camino muy largo por recorrer.

Durante el proyecto de fin de carrera se desarrolló un recorrido interactivo para un museo con realidad aumentada, sobre dispositivos móviles con sistema operativo iOS como iPhone, iPod Touch y iPad. Probablemente, la realidad aumentada sea el mayor atractivo del proyecto por ser un área que se encuentra en pleno desarrollo y que todo el tiempo recibe ideas innovadoras y muy interesantes, lo que la hace por demás apasionante. Vale la pena entonces dar una definición para la misma:

La realidad aumentada (AR del inglés Augmented Reality) es un término que denota la visión de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales generados por computadora, para la creación de una realidad mixta en tiempo real.

Cuando se genera una imagen por medio de realidad aumentada, conviven en ella elementos reales con elementos virtuales. Es básicamente un juego de percepciones. En la Figura 1 se muestra un ejemplo de realidad aumentada desarrollado durante este proyecto. Se puede ver en ella, un cubo virtual sobre la esquina superior izquierda de “Hombre de Vitruvio” de Leonardo da Vinci, de manera coherente con la posición del dispositivo respecto de la obra. Si en esta figura tan sólo se viera a través del dispositivo, cualquiera podría pensar que el cubo es real y que efectivamente forma parte de la escena. Eso es lo que busca la realidad aumentada.

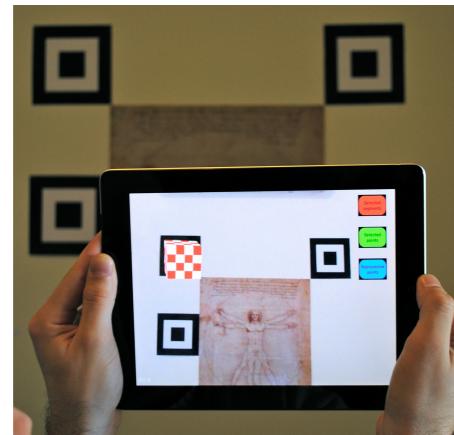


Figura 1: Ejemplo de realidad aumentada. En la figura se puede ver cómo se ubica un cubo virtual en la esquina superior izquierda de una réplica de la obra “Hombre de Vitruvio” de Leonardo da Vinci.

II. OBJETIVOS DEL PROYECTO

El presente proyecto de fin de carrera tiene varios objetivos. En primer lugar se busca evaluar la capacidad de procesamiento de dispositivos móviles para aplicaciones de procesamiento de imágenes y estudiar el desempeño de diferentes algoritmos. Esto implica que se deben estudiar los diferentes dispositivos disponibles en el mercado. Por otro lado poder aplicar lo investigado para desarrollar una aplicación de realidad aumentada completa funcionando sobre un dispositivo móvil y en tiempo real. Finalmente se quiere utilizar dicha aplicación para abordar un problema real, el *recorrido interactivo con realidad aumentada* para museos.

Los objetivos anteriores pueden resumirse en las tres tareas fundamentales del proyecto, que se expresan a continuación:

1. **Investigación:** Comprensión de la arquitectura de las plataformas móviles y de sus plataformas de desarrollo, con el objetivo de implementar los distintos algoritmos y *software* en general en las mismas. Estudio de las diferentes maneras de lograr la realidad aumentada, elección de los algoritmos a utilizar y su comprensión, desarrollo de nuevos algoritmos y variantes de algoritmos existentes. Aprendizaje de herramientas en general.
2. **Implementación:** Integración de los distintos bloques para lograr la realidad aumentada. Implementación de bloques lógicos accesorios que faciliten la integración de los primeros. Validación de los algoritmos utilizados y desarrollados.
3. **Aplicación:** Implementación de una aplicación completa en la que el usuario ingrese al museo, se ubique dentro de él, se dirija a un cuadro, reciba información respecto del mismo y finalmente experimente la realidad aumentada sobre la obra.

Cada una de ellas se jerarquizó en función de la importancia que se les dio en el proyecto, así como también el tiempo que se les dedicó:

Frente de trabajo	Porcentaje
Investigación	50%
Implementación	30%
Aplicación	20%

Lo que la tabla anterior intenta reflejar es que el foco principal del proyecto es la investigación, evaluación de algoritmos y su migración a plataformas móviles. La aplicación es un objetivo secundario que ayuda a validar los conceptos estudiados en las etapas de investigación e implementación.

III. EXPLICACIÓN GLOBAL DE LA APLICACIÓN

Si bien se dijo que la creación de la aplicación integral, correspondiente a un recorrido con realidad aumentada para muesos, corresponde tan sólo a un quinto del alcance total del proyecto; la visualización de la aplicación total es quizás la forma más sencilla de comprender el proyecto en su conjunto y ayuda a evaluar si el sistema desarrollado funciona de manera aceptable.

La aplicación se desglosa en tres grandes bloques que son resumidos individualmente en secciones subsiguientes:

- **Navegación**
- **Identificación de obras**
- **Realidad aumentada**

III-A. Navegación

La navegación es la ubicación del usuario dentro del museo, la importancia de este bloque reside en que permite brindar información contextual de donde se encuentra el usuario. Esta información podría ser una breve descripción de la sala en la que se encuentra o indicaciones para ir de un lugar a otro del museo, entre otras. Desde el punto de vista de la aplicación también es importante porque permite manejar un menor volumen de datos al momento de identificar las

obras, debido a que sabe en qué sala se encuentra el usuario. Se estudiaron distintas alternativas para la navegación. La primera posibilidad analizada fue la utilización de tres o más *access points*, mediante los cuales, una vez mapeadas las características de las señales en cada uno de los puntos de las salas, se puede ubicar al usuario dentro de las mismas. Otra forma de navegación que se tuvo en cuenta fue la localización a través de la tecnología GPS. Sin embargo, se optó por utilizar códigos QR dada su amplia difusión, practicidad y facilidad de implementación. La discusión técnica que justifica estas elecciones se encuentra detallada en la documentación del proyecto.

III-B. Identificación de obras

Por identificación de obras se entiende al proceso mediante el cual la aplicación detecta frente a qué obra se encuentra el usuario para así entonces brindarle información de la misma, una audioguía y si fuera el caso la posibilidad de desplegar realidad aumentada sobre ella. La forma en la que se implementó este bloque fue mediante un algoritmo de detección de características de imágenes llamado SIFT. Es un algoritmo que se basa en el contenido de la imagen únicamente, por lo que para identificar la obra no se necesita ningún tipo de marcador o agregado externo. Los detalles de la implementación se encuentran en la documentación del proyecto.

III-C. Realidad Aumentada

El proceso mediante el cual se logra la realidad aumentada puede verse en el diagrama de bloques de la Figura 2.

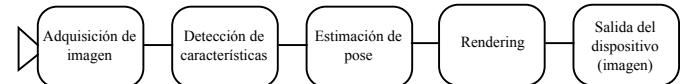


Figura 2: Diagrama de bloques del proceso mediante el cual se logra la realidad aumentada.

El primer bloque corresponde a la cámara que captura una imagen. En el segundo bloque dicha imagen es procesada con el objetivo de detectar en esta características. Estas características pueden ser segmentos, esquinas, descriptores. El tercer bloque corresponde a la estimación de pose, busca estimar en qué posición se encuentra la cámara respecto de cierto eje de coordenadas previamente definido y hacia dónde esta apunta. En el cuarto bloque con la información anterior, se debe poder renderizar una escena de manera consistente con la pose de la cámara, para así entonces lograr la salida del sistema, que será una imagen con la realidad aumentada incorporada.

Debe notarse que en ninguna etapa se mencionó ningún algoritmo en particular, ya sea para la detección de características como para la estimación de pose. Tampoco se menciona nada en particular para la etapa de rendering. Esto se debe a que las etapas se concibieron de forma genérica, de

manera ser libres de utilizar cualquier algoritmo de detección de características con cualquier algoritmo de estimación de pose. Las interfaces entre los bloques están pensadas de forma de poder cambiar un bloque sin generar problemas en los demás bloques.

En lo que sigue se hace una explicación breve del modelo de cámara que fue necesario adoptar a los efectos de vincular las relaciones del mundo con las de las imágenes, se describe parcialmente la detección de características desarrollada y el algoritmo de estimación de pose que se utilizó.

IV. MODELO DE CÁMARA *pin-hole*

Este modelo consiste en un centro óptico O , en donde convergen todos los rayos de la proyección y un plano imagen en el cual la imagen es proyectada. Se define *distanza focal* (f) como la distancia entre el centro óptico O y la intersección del eje óptico con el plano imagen (punto C). Ver Figura 3.

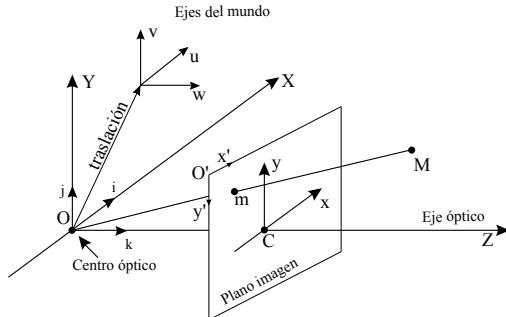


Figura 3: Modelo de cámara pin-hole.

Se llama proceso de proyección al proceso en el que se asocia al punto M del mundo, un punto m en la imagen. Para modelar el mismo es necesario referirse a varias transformaciones y varios ejes de coordenadas.

- *Coordenadas del mundo:* son las coordenadas que describen la posición 3D del punto M respecto de los ejes del mundo (u, v, w). La elección de los ejes del mundo es arbitraria.
- *Coordenadas de la cámara:* son las coordenadas que describen la posición del punto M respecto de los ejes de la cámara (X, Y, Z). i, j y k son los versores de este eje de coordenadas.
- *Coordenadas de la imagen:* son las coordenadas que describen la posición del punto 2D, m respecto del centro del plano imagen, C . Los ejes de este sistema de coordenadas son (x, y).
- *Coordenadas normalizadas de la imagen:* son las coordenadas que describen la posición del punto 2D, m' respecto del eje de coordenadas (x', y') situado en la esquina superior izquierda del plano imagen.

La transformación que lleva al punto M , expresado respecto de los ejes del mundo, al punto m , expresado respecto del sistema de coordenadas normalizadas de la imagen, se puede ver como la composición de dos transformaciones menores.

La primera, es la que realiza la proyección que transforma a un punto definido respecto del sistema de coordenadas de la cámara (X, Y, Z) en otro punto sobre el plano imagen expresado respecto del sistema de coordenadas normalizadas de la imagen (x', y'). Véase que una vez calculada esta transformación, es una constante característica de cada cámara. Al conjunto de valores que definen esta transformación, se le llama “parámetros intrínsecos” de la cámara. La segunda, es la transformación que lleva de expresar un punto respecto de los ejes del mundo (u, v, w), a ser expresado según los ejes de la cámara (X, Y, Z). Esta última transformación varía conforme se mueve la cámara (respecto de los ejes del mundo) y el conjunto de valores que la definen es denominado “parámetros extrínsecos” de la cámara. Del cálculo de estos parámetros es que se obtiene la estimación de la pose de la cámara. De lo anterior se concluye rápidamente que si se le llama H a la matriz proyección total, tal que:

$$m = H \cdot M,$$

entonces:

$$H = I \cdot E$$

donde I corresponde a la matriz proyección asociada a los parámetros intrínsecos y E corresponde a la matriz asociada a los parámetros extrínsecos.

- **Parámetros extrínsecos:** pose de la cámara.
 - Traslación: ubicación del centro óptico de la cámara respecto de los ejes del mundo.
 - Rotación: rotación del sistema de coordenadas de la cámara (X, Y, Z), respecto de los ejes del mundo.
- **Parámetros intrínsecos:** parámetros propios de la cámara. Dependen de su geometría interna y de su óptica.
 - Punto principal ($C = [x'_C, y'_C]$): es el punto intersección entre el eje óptico y el plano imagen. Las coordenadas de este punto vienen dadas en píxeles y son expresadas respecto del sistema normalizado de la imagen.
 - Factores de conversión píxel-milímetros (d_x, d_y): indican el número de píxeles por milímetro que utiliza la cámara en las direcciones x e y respectivamente.
 - Distancia focal (f): distancia entre el centro óptico (O) y el punto principal (C). Su unidad es el milímetro.
 - Factor de proporción (s): indica la proporción entre las dimensiones horizontal y vertical de un píxel.

IV-A. Matriz de proyección

En la sección anterior se vio que es posible hallar una “matriz de proyección” H que dependa tanto de los parámetros intrínsecos de la cámara como de sus parámetros extrínsecos:

$$m = H \cdot M$$

donde M y m son los puntos ya definidos y vienen expresados en “coordenadas homogéneas”. Por más información acerca de este tipo de coordenadas ver [?].

Para determinar la forma de la matriz de proyección se estudia cómo se relacionan las coordenadas de **M** con las coordenadas de **m**; para hallar esta relación se debe analizar cada transformación, entre los sistemas de coordenadas mencionados con anterioridad, por separado.

- **Proyección 3D - 2D:** de las coordenadas homogéneas del punto **M** expresadas en el sistema de coordenadas de la cámara (X_0, Y_0, Z_0, T_0) , a las coordenadas homogéneas del punto **m** expresadas en el sistema de coordenadas de la imagen (x_0, y_0, s_0) :

Se desprende de la Figura 3 y algo de trigonometría la siguiente relación entre las coordenadas en cuestión y la distancia focal (f):

$$\frac{f}{Z_0} = \frac{x_0}{X_0} = \frac{y_0}{Y_0}$$

A partir de la relación anterior:

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \frac{f}{Z_0} \begin{pmatrix} X_0 \\ Y_0 \end{pmatrix}$$

Expresado en forma matricial, en coordenadas homogéneas:

$$\begin{pmatrix} x_0 \\ y_0 \\ s_0 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{pmatrix}$$

- **Transformación imagen - imagen:** de las coordenadas homogéneas del punto **m** expresadas respecto del sistema de coordenadas de la imagen (x_0, y_0, s_0) , a las coordenadas homogéneas de él mismo pero expresadas respecto del sistema de coordenadas normalizadas de la imagen (x'_0, y'_0, s'_0) :

Se les suma, a las coordenadas de **m** respecto del sistema de la imagen, la posición del punto **C** respecto del sistema normalizado de la imagen (x'_C, y'_C) . Las coordenadas de **m** dejan de ser expresadas en milímetros para ser expresadas en píxeles. Aparecen los factores de conversión d_x y d_y :

$$\begin{aligned} x'_0 &= d_x \cdot x_0 + x'_C \\ y'_0 &= d_y \cdot y_0 + y'_C \end{aligned}$$

Se obtiene entonces la siguiente relación matricial, en coordenadas homogéneas:

$$\begin{pmatrix} x'_0 \\ y'_0 \\ s'_0 \end{pmatrix} = \begin{pmatrix} d_x & 0 & x'_C \\ 0 & d_y & y'_C \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}$$

- **Matriz de parámetros intrínsecos (I):** de las coordenadas homogéneas del punto **M** expresadas en el sistema de coordenadas de la cámara $(X_0, Y_0, Z_0, 1)$, a las coordenadas homogéneas del punto **m** expresadas respecto del sistema de coordenadas normalizadas de la imagen (x'_0, y'_0, s'_0) :

Se obtiene combinando las dos últimas transformaciones. Nótese que como ya se aclaró, depende únicamente de parámetros propios de la construcción de la cámara:

$$I = \begin{pmatrix} d_x \cdot f & 0 & x'_C & 0 \\ 0 & d_y \cdot f & y'_C & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

- **Matriz de parámetros extrínsecos (E):** de las coordenadas homogéneas del punto **M** expresadas respecto del sistema de coordenadas del mundo (U_0, V_0, W_0, P_0) , a las coordenadas homogéneas de él mismo pero expresadas respecto del sistema de coordenadas de la cámara (X_0, Y_0, Z_0, T_0) :

Se obtiene de estimar la pose de la cámara respecto de los ejes del mundo y es la combinación de, primero una rotación R_{3x3} , y luego una traslación T_{3x1} . Se obtiene entonces la siguiente representación matricial:

$$\begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \\ T_0 \end{pmatrix} = \begin{pmatrix} R & T \end{pmatrix} \begin{pmatrix} U_0 \\ V_0 \\ W_0 \\ P_0 \end{pmatrix}$$

donde la matriz de parámetros extrínsecos desarrollada toma la forma:

$$E = \begin{pmatrix} i_u & i_v & i_w & t_x \\ j_u & j_v & j_w & t_y \\ k_u & k_v & k_w & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- **Matriz de proyección (H):** de las coordenadas homogéneas del punto **M** expresadas respecto del sistema de coordenadas del mundo (U_0, V_0, W_0, P_0) , a las coordenadas homogéneas del punto **m** expresadas respecto del sistema de coordenadas normalizadas de la imagen (x'_0, y'_0, s'_0) :

Es la proyección total y se obtiene combinando las dos transformaciones anteriores:

V. MARCADORES

La inclusión de *marcadores* en la escena ayuda al problema de extracción de características y por lo tanto al problema de estimación de pose. Estos por construcción son elementos que presentan una detección estable en la imagen para el tipo de característica que se desea extraer así como medidas fácilmente utilizables para la estimación de la pose.

El marcador utilizado está basado en la estructura de detección incluida en los códigos *QR* y se muestra en la Figura 4. Éste consiste en tres grupos idénticos de tres cuadrados concéntricos superpuestos en “capas”. La primer capa contiene el cuadrado negro de mayor tamaño, en la segunda capa se ubica el cuadrado mediano en color blanco y en la última capa un cuadrado negro pequeño. De esta forma se logra un fuerte contraste en los lados de cada uno de los cuadrados facilitando la detección de bordes o líneas. A diferencia de los códigos *QR* la disposición espacial de los grupos de cuadrados es distinta para evitar ambigüedades en la identificación de los mismos entre sí.

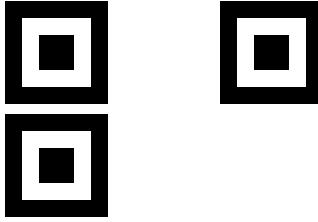


Figura 4: Marcador propuesto basado en la estructura de detección de códigos QR.

V-A. Estructura del marcador

A continuación se presentan algunas definiciones de las estructuras básicas que componen el marcador propuesto. Estas son de utilidad para el diseño y forman un flujo natural y escalable para el desarrollo del algoritmo de determinación de correspondencias.

Los elementos más básicos en la estructura son los *segmentos* los cuales consisten en un par de puntos en la imagen, $\mathbf{p} = (p_x, p_y)$ y $\mathbf{q} = (q_x, q_y)$. Estos *segmentos* forman lo que son los lados del *cuadrilátero*, el próximo elemento estructural del marcador.

Un *cuadrilátero* o *quadrilateral* en inglés, al que se le denomina Ql , está determinado por cuatro segmentos conexos y distintos entre sí. El cuadrilátero tiene dos propiedades notables; el *centro* definido como el punto medio entre sus cuatro vértices y el *perímetro* definido como la suma de el largo de sus cuatro lados.

A un *conjunto de cuadriláteros* o *quadrilateral set* se le denomina $QlSet$ y se construye a partir de M cuadriláteros, con $M > 1$. Los cuadriláteros comparten un mismo centro pero se diferencian en un factor de escala. A partir de dichos cuadriláteros se construye un lista ordenada ($Ql[0], Ql[1], \dots, Ql[M-1]$) en donde el orden viene dado por el valor de perímetro de cada Ql . Se define el *centro del grupo de cuadriláteros*, \mathbf{c}_i , como el promedio de los centros de cada Ql de la lista ordenada.

Finalmente el *marcador QR* está constituido por N conjuntos de cuadriláteros dispuestos en una geometría particular. Esta geometría permite la determinación de un sistema de coordenadas; un origen y dos ejes a utilizar. Se tiene una lista ordenada ($QlSet[0], QlSet[1], \dots, QlSet[N-1]$) en donde el orden se puede determinar mediante la disposición espacial de los mismos o a partir de hipótesis razonables.

V-B. Diseño

En base a las estructuras previamente definidas es que se describe el diseño del marcador. Como ya se explicó se toma un marcador tipo QR basado en cuadriláteros y más específicamente en tres conjuntos de tres cuadrados dispuestos en como se muestra en la Figura 4.

Los tres cuadriláteros correspondientes a un mismo conjunto de cuadriláteros tienen idéntica alineación e idéntico centro. Los diferencia un factor de escala, esto es, $Ql[0]$ tiene lado l mientras que $Ql[1]$ y $Ql[2]$ tienen lado $2l$ y $3l$ respectivamente.

Esto se puede ver en la Figura 5. Adicionalmente se define un sistema de coordenadas con centro en el centro del $QlSet$ y ejes definidos como x horizontal a la derecha e y vertical hacia abajo. Definido el sistema de coordenadas se puede fijar un orden a los vértices v_{j_1} de cada cuadrilátero $Ql[j]$ como,

$$\begin{aligned} v_{j_0} &= (a/2, a/2) & v_{j_2} &= (-a/2, -a/2) \\ v_{j_1} &= (a/2, -a/2) & v_{j_3} &= (-a/2, a/2) \end{aligned}$$

con $a = (j+1) \times l$. El orden aquí explicado se puede ver también junto con el sistema de coordenadas en la Figura 6.

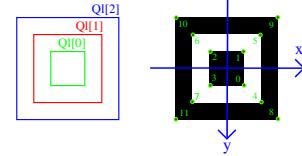


Figura 5: Detalle de un $QlSet$. A la izquierda se muestra el resultado de la detección de un $QlSet$ y el orden interno de sus cuadriláteros y a la derecha el orden de los vértices respecto al sistema de coordenadas local.

Un detalle del marcador completo se muestra en la Figura 6 en donde se define el conjunto i de cuadriláteros concéntricos como el $QlSet[i]$ y se definen los respectivos centros de cada uno de ellos como \mathbf{c}_i . El sistema de coordenadas del marcador QR tiene centro en el centro del $QlSet[0]$ y ejes de coordenadas idénticos al definido para cada Ql . Se tiene además que los ejes de coordenadas pueden ser obtenidos mediante los vectores normalizados,

$$\mathbf{x} = \frac{\mathbf{c}_1 - \mathbf{c}_0}{\|\mathbf{c}_1 - \mathbf{c}_0\|} \quad \mathbf{y} = \frac{\mathbf{c}_2 - \mathbf{c}_0}{\|\mathbf{c}_2 - \mathbf{c}_0\|} \quad (1)$$

La disposición de los $QlSet$ es tal que la distancia indicada d_{01} definida como la norma del vector entre los centros \mathbf{c}_1 y \mathbf{c}_0 es significativamente mayor que la distancia d_{02} definida como la norma del vector entre los centros \mathbf{c}_2 y \mathbf{c}_1 . Esto es, $d_{01} \gg d_{02}$. Este criterio facilita la identificación de los $QlSet$ entre sí basados únicamente en la posición de sus centros y es explicado en la sección de determinación de correspondencias.

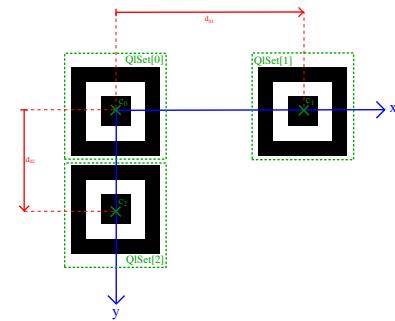


Figura 6: Detalle del marcador propuesto formando un sistema de coordenadas.

V-C. Diseño utilizado

Diseño de Test: Durante el desarrollo e implementación de los algoritmos de detección e identificación de los vértices del marcador se trabajó con determinados parámetros de diseño de dimensiones apropiadas para posibilitar el traslado y las pruebas domésticas.

- $l = 30\text{mm}$
- $d_{01} = 190\text{mm}$
- $d_{02} = 100\text{mm}$

VI. DETECCIÓN

La etapa de detección del marcador se puede separar en tres grandes bloques:

- Detección de segmentos de línea.
- Filtrado y agrupamiento de segmentos.
- Determinación de correspondencias.

En esta sección se muestran algunos resultados para la detección de segmentos de línea por LSD y se centra en profundidad en los algoritmos desarrollados durante el proyecto para el filtrado de segmentos y determinación de correspondencias.

VI-A. Detección de segmentos de línea

La detección de segmentos de línea se realiza mediante el uso del algoritmo LSD. En forma resumida, dicho algoritmo toma como entrada una imagen en escala de grises de tamaño $W \times H$ y devuelve una lista de segmentos en forma de pares de puntos de origen y destino.

VI-B. Filtrado y agrupamiento de segmentos

El filtrado y agrupamiento de segmentos consiste en la búsqueda de conjuntos de cuatro segmentos conexos en la lista de segmentos de línea detectados por LSD. Los conjuntos de segmentos conexos encontrados se devuelven en una lista en el mismo formato a la de LSD pero agrupados de a cuatro. A continuación se realiza una breve descripción del algoritmo de filtrado de segmentos implementado.

Se parte de una lista de m segmentos de línea,

$$\mathbf{L} = (\mathbf{s}_0 \quad \mathbf{s}_1 \quad \dots \quad \mathbf{s}_{m-1})^t \quad (2)$$

y se recorre en i en busca de segmentos vecinos. La estrategia utilizada consiste en buscar, para el i -ésimo segmento \mathbf{s}_i , dos segmentos vecinos. En una primera etapa \mathbf{s}_j y en una segunda etapa \mathbf{s}_k , de forma que se forme una "U" como se muestra en la Figura 7. La tercera etapa de búsqueda consiste en completar ese conjunto con un cuarto segmento \mathbf{s}_l que cierre la "U".

Una vez encontrado el conjunto de cuatro segmentos conexos estos se marcan como utilizados, se guardan en una lista de salida y se continúa con el segmento $i+1$ hasta recorrer los m segmentos de la lista de entrada. De esta forma se obtiene una lista de salida \mathbf{S} de n segmentos en donde n es por construcción múltiplo de cuatro.

En la Figura 8 se muestran los resultados obtenidos para el algoritmo tomando como entrada la lista de segmentos de LSD. Se puede ver que los lados de los cuadrados del marcador son

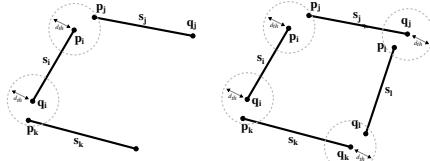
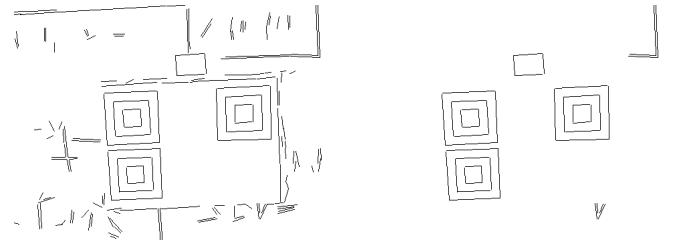


Figura 7: Conjunto de cuadriláteros conexos. A la izquierda la primera y segunda etapa del filtrado completadas para el segmento \mathbf{s}_i en donde se busca una "U". A la derecha la última etapa en donde se cierra la "U" con el segmento \mathbf{s}_l .



(a) Entrada: segmentos de línea detectados por LSD
(b) Salida: segmentos de línea filtrados y agrupados

Figura 8: Resultados del algoritmo de filtrado y agrupamiento de segmentos de línea.

detectados correctamente pero también hay otras detecciones presentes.

El algoritmo descrito es simple y provee resultados aceptables en general pero es propenso a tanto a detectar *falsos positivos* como al *sobre-filtrado* algunos conjuntos.

VI-C. Determinación de correspondencias

Se detalla a continuación el algoritmo de determinación de correspondencias a partir de grupos de cuatro segmentos de línea conexos.

Se toma como entrada la lista de segmentos filtrados y agrupados

$$\mathbf{S} = (\mathbf{s}_0 \quad \mathbf{s}_1 \quad \dots \quad \mathbf{s}_i \quad \mathbf{s}_{i+1} \quad \mathbf{s}_{i+2} \quad \mathbf{s}_{i+3} \quad \dots \quad \mathbf{s}_{n-1})^t \quad (3)$$

en donde cada segmento se compone de un punto inicial \mathbf{p}_i y un punto final \mathbf{q}_i , $\mathbf{s}_i = (\mathbf{p}_i, \mathbf{q}_i)$, con n múltiplo de cuatro. Si i también lo es, entonces el sub-conjunto, $\mathbf{S}_i = (\mathbf{s}_i \quad \mathbf{s}_{i+1} \quad \mathbf{s}_{i+2} \quad \mathbf{s}_{i+3})^t$, corresponde a un conjunto de cuatro segmentos del líneal conexos.

Para cada sub-conjunto \mathbf{S}_i se intersectan entre sí los segmentos obteniendo una lista de cuatro vértices, $\mathbf{V}_i = (\mathbf{v}_i \quad \mathbf{v}_{i+1} \quad \mathbf{v}_{i+2} \quad \mathbf{v}_{i+3})^t$. Se obtienen dos posibles configuraciones que se muestran en la Figura 9, una de ellas tiene sentido horario y la otra antihorario partiendo de \mathbf{v}_i .

Posterior a la intersección se realiza un chequeo sobre el valor de las coordenadas de los vértices. Si alguno de ellos se encuentra fuera de los límites de la imagen, el conjunto de cuatro segmentos es marcado como inválido. Este chequeo

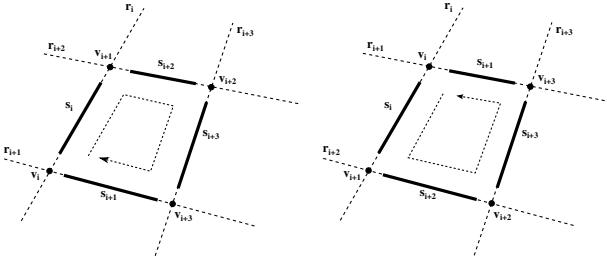


Figura 9: Posibles configuraciones de vértices posterior a la intersección de conjuntos de segmentos pertenecientes a un cuadrilátero.

resulta en el filtrado de “falsos cuadriláteros” corrigiendo un defecto del filtrado de segmentos, como por ejemplo un grupo de segmentos paralelos cercanos como ya se explicó.

Para cada uno de los conjuntos de vértices se construye con ellos un elemento cuadrilátero que se almacena en una lista de cuadriláteros

$$QlList = (Ql[0] \quad Ql[1] \quad \dots \quad Ql[i] \quad \dots \quad Ql[\frac{n}{4}])^t$$

A partir de esa lista de cuadriláteros, se buscan grupos de tres cuadriláteros $QlSet$ que “compartan” un mismo centro. Para esto se recorre ordenadamente la lista en i buscando para cada cuadrilátero dos cuadriláteros j y k que cumplan que la distancia entre sus centros y el del i -ésimo cuadrilátero sea menor a cierto umbral c_{th} ,

$$d_{ij} = \|\mathbf{c}_i - \mathbf{c}_j\| < c_{th}, \quad d_{ik} = \|\mathbf{c}_i - \mathbf{c}_k\| < c_{th}. \quad (4)$$

Estos cuadriláteros se marcan en la lista como utilizados con ellos se forma el l -ésimo $QlSet$ ordenándolos según su perímetro, de menor a mayor como

$$QlSet[l] = (Ql[0] \quad Ql[1] \quad Ql[2])$$

con $l = (0, 1, 2)$. Esta búsqueda se realiza hasta encontrar un total de tres $QlSet$ completos de forma de obtener un marcador completo, esto es, detectando todos los cuadriláteros que lo componen.

Una vez obtenida la lista de tres $QlSet$,

$$QlSetList = (QlSet[0] \quad QlSet[1] \quad QlSet[2])$$

ésta se ordena de forma que su disposición espacial se corresponda con la del marcador QR. Para esto se calculan las distancias entre los centros de cada $QlSet$ y se toma el índice i como el índice que produce el vector de menor distancia, $\mathbf{u}_i = \mathbf{c}_{i+1} - \mathbf{c}_i$. En este punto es importante que la condición de distancia entre los centros de los $QlSet$ se cumpla, $d_{10} \gg d_{20}$, para una simple identificación.

Una vez seleccionado el vector \mathbf{u}_i , se tienen obtiene el juego de vectores $(\mathbf{u}_i, \mathbf{u}_{i+1}, \mathbf{u}_{i+2})$ como se muestra en la Figura 10.

Existen solo dos posibles configuraciones para estos vectores por lo que se utiliza este conocimiento para ordenar los $QlSet$ de la lista realizando el producto vectorial $\hat{\mathbf{u}}_i \times \mathbf{u}_{i+1}$.

Se construye un marcador QR que contiene la lista de tres $QlSet$ ordenados según lo indicado permitiendo la definición

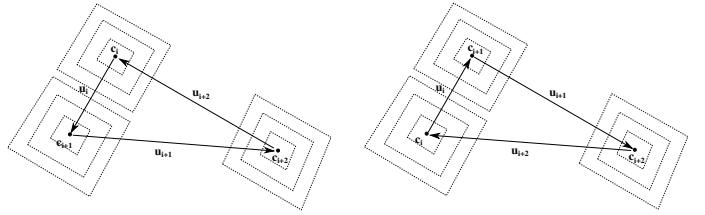


Figura 10: Vértices de cada Ql ordenados respecto al signo de sus proyecciones contra el sistema de coordenadas local a cada $QlSet$.

de un centro de coordenadas como el centro \mathbf{c}_0 del $QlSet[0]$ y ejes de coordenadas definidos en la Ecuación 1. De esta forma, recorriendo ordenadamente los elementos del marcador, se ordenan los vértices de cada Ql del marcador.

Por último, a partir del marcador ordenado, se extrae una lista de vértices que se corresponde con la lista de vértices del marcador en coordenadas del mundo. Se determinan las correspondencias $\mathbf{M}_i \leftrightarrow \mathbf{m}_i$ necesarias para la estimación de pose. El algoritmo de determinación de correspondencias funciona correctamente por lo que los “falsos” cuadriláteros que sobreviven al filtrado de segmentos no son un problema.

VII. POSIT

A continuación se explica el algoritmo utilizado para el cálculo de la pose a partir de una imagen capturada por la cámara. Como lo dice el nombre del algoritmo se utiliza una técnica llamada *POS* (*Pose from Orthography and Scaling*), esta técnica consiste en aproximar la pose de la cámara a partir de la proyección *SOP* (*Scaled Orthographic Projection*).

VII-A. Notación

En la Figura 11 se puede ver nuevamente el modelo de cámara pinhole. O es el centro óptico y G es el plano imagen ubicado a una distancia focal f de O . x e y son los ejes que apuntan en las direcciones de las filas y las columnas del sensor de la cámara respectivamente. z es el eje que esta sobre el eje óptico de la cámara y apunta en sentido saliente. Los versores para estos ejes son \mathbf{i} , \mathbf{j} y \mathbf{k} respectivamente.

Se considera ahora un objeto con puntos característicos $M_0, M_1, \dots, M_i, \dots, M_n$, cuyo eje de coordenadas está centrado en M_0 y está compuesto por los versores (M_0u, M_0v, M_0w) . Como los ejes del mundo son arbitrarios se puede asumir sin pérdida de generalidad que los ejes del objeto coinciden con los ejes del mundo. La geometría del objeto se asume conocida, por ejemplo (U_i, V_i, W_i) son las coordenadas del punto M_i en el marco de referencia del objeto. Los puntos correspondientes a los puntos del objeto M_i en la imagen son conocidos y se identifican como m_i , (x_i, y_i) son las coordenadas de este punto en la imagen¹. Las coordenadas de los puntos M_i en el eje de coordenadas de la cámara, identificadas como (X_i, Y_i, Z_i) , son desconocidas ya que no se conoce la pose del objeto respecto a la cámara.

¹En realidad los puntos m_i no están dados, vienen de la etapa anterior de detección.

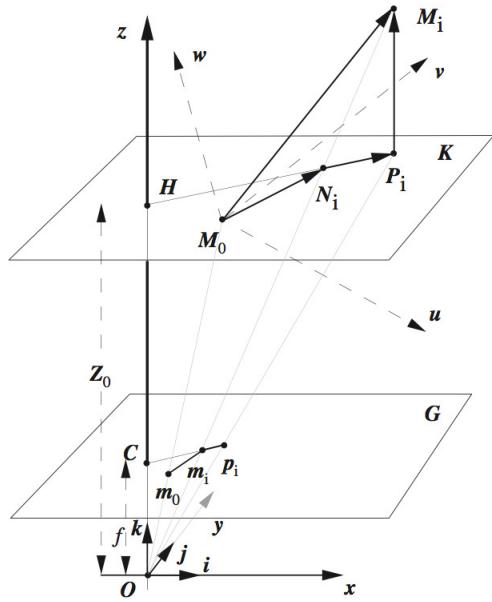


Figura 11: Proyección en perspectiva (m_i) y SOP (p_i) para un punto del modelo 3D M_i y un punto de referencia del modelo M_0 . Tomado de: [?].

Se busca computar la matriz de rotación y el vector de traslación del objeto respecto a la cámara.

VII-B. SOP: Scaled Orthographic Projection

La proyección ortogonal escalada(SOP) es una aproximación a la proyección perspectiva. En esta aproximación se supone que las profundidades Z_i de diferentes puntos M_i en el eje de coordenadas de la cámara no difieren mucho entre sí, y por lo tanto se asume que todos los puntos M_i tienen la misma profundidad que el punto M_0 . Esta suposición es razonable cuando la relación distancia cámara objeto - profundidad del objeto es grande.

Para un punto M_i la proyección perspectiva sobre el plano imagen está dada por:

$$x_i = fX_i/Z_i, \quad y_i = fY_i/Z_i,$$

mientras que la proyección SOP está dada por:

$$x'_i = fX_i/Z_0, \quad y'_i = fY_i/Z_0.$$

Al término $s = f/Z_0$ se lo conoce como el factor de escala de la SOP.

En la Figura 11 se puede ver como se construye la SOP. Primero se realiza la proyección ortogonal de todos los puntos M_i sobre K , el plano paralelo al plano imagen que pasa por el punto M_0 . Las proyecciones de los puntos M_i sobre K se llaman P_i . El segundo paso consiste en hacer la proyección perspectiva de los puntos P_i sobre el plano imagen G para obtener finalmente los puntos p_i .

VII-C. Ecuaciones para calcular la proyección perspectiva

La pose queda determinada si se conocen los vectores \mathbf{i} , \mathbf{j} y la coordenada Z_0 del vector de traslación. La relación entre las coordenadas de los puntos M_i en el sistema de coordenadas del objeto y el sistema de coordenadas de la cámara es

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} i_u & i_v & i_w \\ j_u & j_v & j_w \\ k_u & k_v & k_w \end{pmatrix} \begin{pmatrix} U_i \\ V_i \\ W_i \end{pmatrix} + \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} \quad (5)$$

La condición necesaria para que la pose definida por \mathbf{i} , \mathbf{j} , x_0 , y_0 y Z_0 sea la pose exacta se puede expresar en las siguientes ecuaciones:

$$M_0M_i \frac{f}{Z_0} \mathbf{i} = x_i(1 + \varepsilon_i) - x_0 \quad (6)$$

$$M_0M_i \frac{f}{Z_0} \mathbf{j} = y_i(1 + \varepsilon_i) - y_0 \quad (7)$$

donde ε_i se define como

$$\varepsilon_i = \frac{1}{Z_0} M_0M_i \cdot \mathbf{k} \quad (8)$$

Se puede ver que los términos $x_i(1 + \varepsilon_i)$ y $y_i(1 + \varepsilon_i)$ son las coordenadas (x'_i, y'_i) de la SOP, en el caso en que la pose está determinada.

VII-D. Algoritmo

Las Ecuaciones 6 y 7 se puede reescribir como:

$$M_0M_i \mathbf{I} = x_i(1 + \varepsilon_i) - x_0 \quad (9)$$

$$M_0M_i \mathbf{J} = y_i(1 + \varepsilon_i) - y_0 \quad (10)$$

en donde

$$\mathbf{I} = \frac{f}{Z_0} \mathbf{i} = s \cdot \mathbf{i}, \quad \mathbf{J} = \frac{f}{Z_0} \mathbf{j} = s \cdot \mathbf{j} \quad (11)$$

Si se conocieran los valores exactos de los ε_i la pose obtenida de resolver el sistema de ecuaciones sería la pose exacta del objeto, como no se conocen los valores exactos de ε_i se utiliza un método iterativo que converge a la solución buscada. En la primera iteración se le toma $\varepsilon_i = 0$. La ecuación para un punto cualquiera está dada por:

$$\begin{aligned} M_0M_i \cdot \mathbf{I} &= x'_i - x_0 \\ M_0M_j \cdot \mathbf{J} &= y'_i - y_0 \end{aligned} \quad (12)$$

Si se escribe la Ecuación 12 para los n puntos del modelo, se tiene un sistema de n ecuaciones con \mathbf{I} y \mathbf{J} como incógnitas

$$\begin{aligned} \mathbf{A}\mathbf{I} &= x' - x_0 \\ \mathbf{A}\mathbf{J} &= y' - y_0 \end{aligned} \quad (13)$$

\mathbf{A} es una matriz $n \times 3$ con las coordenadas de los puntos del modelo M_i en el marco de coordenadas del objeto. Si se tienen más de 4 puntos y no son coplanares, la matriz \mathbf{A} es de rango 3, y las soluciones al sistema están dadas por

$$\begin{aligned} \mathbf{I} &= \mathbf{B} \left(x' - x_0 \right) \\ \mathbf{J} &= \mathbf{B} \left(y' - y_0 \right) \end{aligned} \quad (14)$$

donde \mathbf{B} es la pseudo inversa de la matriz \mathbf{A} . Se debe notar que la matriz \mathbf{B} depende únicamente de la geometría del modelo que se asume conocida, por lo tanto solo es necesario calcularla una sola vez.

Una vez obtenidos \mathbf{I} y \mathbf{J} se calculan s y los versores \mathbf{i} , \mathbf{j} y \mathbf{k}

$$s = (|\mathbf{I}| |\mathbf{J}|)^{1/2} \quad (15a)$$

$$\mathbf{i} = \frac{\mathbf{I}}{s} \quad (15b)$$

$$\mathbf{j} = \frac{\mathbf{J}}{s} \quad (15c)$$

$$\mathbf{k} = \mathbf{i} \times \mathbf{j} \quad (15d)$$

El vector traslación del centro del objeto al centro de la cámara es el vector OM_0

$$OM_0 = \frac{Z_0}{f} Om_0 = \frac{Om_0}{s} \quad (16)$$

El vector Om_0 es conocido ya que se conocen las coordenadas de los puntos m_i , en particular m_0 .

Una vez que se calcularon \mathbf{i} , \mathbf{j} , \mathbf{k} y \mathbf{T} se calculan los valores actualizados de ε_i según la Ecuación 8. Si la variación de los ε_i es mayor a un determinado umbral, se repite el procedimiento actualizando las proyecciones SOP en la Ecuación 13, si es menor al umbral se deja de iterar y se guarda la pose calculada.

VII-E. POSIT para puntos coplanares

Como se mencionó anteriormente, el algoritmo POSIT no funciona en el caso en que los puntos del modelo pertenecen a un mismo plano. Como los marcadores utilizados son planos, se buscó una versión de POSIT que resuelve el problema de la estimación de pose para este caso. El algoritmo fue escrito por DeMenthon et al. en [?].

Si todos los puntos son coplanares, hay dos posibles configuraciones de puntos que cuya proyección SOP es la misma. Esto se puede ver en la Figura 12.

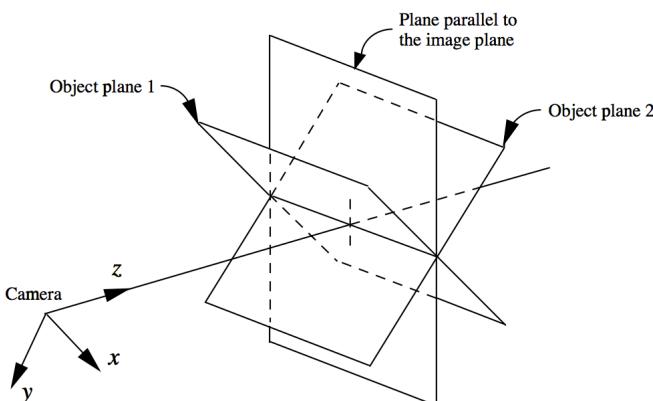


Figura 12: Dos objetos dando la misma proyección SOP.
Fuente: [?].

Analíticamente sucede que el sistema de ecuaciones en 13 queda de rango 2. El vector solución que se obtiene al realizar la pseudo inversa de \mathbf{A} es la proyección de \mathbf{I} sobre el plano paralelo al plano imagen (K en la Figura 11). Para determinar completamente el vector \mathbf{I} hace falta calcular coordenada z del vector \mathbf{I} . Se procede de manera análoga para calcular el vector \mathbf{J} . Finalmente se obtienen dos posibles soluciones. El cálculo detallado de los vectores I y J para configuraciones coplanares se puede encontrar en [?].

En el caso en que las dos soluciones sean válidas para todas las iteraciones, el número de poses posibles sería 2^n a lo largo de n iteraciones. En la práctica se manejan menos soluciones posibles. Se diferencian dos casos:

- Si se tiene que sólo una de las dos primeras poses calculadas es válida, en las siguientes iteraciones se da mismo comportamiento, por lo que hay solo un camino a seguir.
- Si se tiene que las dos primeras poses calculadas son válidas, se abren dos posibles ramas. En la segunda iteración cada rama da lugar a dos nuevas poses, pero en este caso se toma la pose que da menor error de reproyección.

VIII. CASOS DE USO

Se implementaron distintos casos de uso con el fin de integrar los algoritmos comentados en secciones anteriores en pequeñas aplicaciones que funcionen *de punta a punta*. Se buscó resolver individualmente los diferentes desafíos técnicos que una aplicación real de realidad aumentada para museos puede llegar a tener. Estas últimas no serán más que una combinación guionada de cada uno de estos casos de uso.

Se implementaron tres casos de uso: “interactividad”, “video” y “modelos”. El primero presenta un modelo simple sobre el marcador que responde a toques con cierto movimiento y un audio en particular, el segundo soluciona el problema de proyectar un video sobre el marcador de forma consistente con el movimiento del usuario. Para los casos de uso “interactividad” y “modelos” se utilizó la librería *isgl3D* para poder realizar el *rendering*. Esta librería fue ampliamente utilizada dada su fácil incorporación a las aplicaciones desarrolladas. La misma cuenta con modelos propios como el cubo que se ve en “interactividad” y tiene la posibilidad de importar otros modelos (generados con programas específicos para ello) de manera de lograr realidades aumentadas mucho más interesantes que si tan sólo se hicieran con modelos simples.

IX. APLICACIÓN

A continuación se muestra la integración de los conocimientos adquiridos a lo largo del proyecto para poder llevar a cabo la realidad aumentada en una aplicación real. Si bien el objetivo principal del proyecto era la exploración de distintos métodos y algoritmos, parecía importante poder poner en práctica todo lo desarrollado en un producto final que pudiera parecerse a un prototipo de aplicación comercial. Se siguió la línea que se planteó desde un inicio que fue la de tener tres partes fundamentales: Navegación, Identificación y Realidad

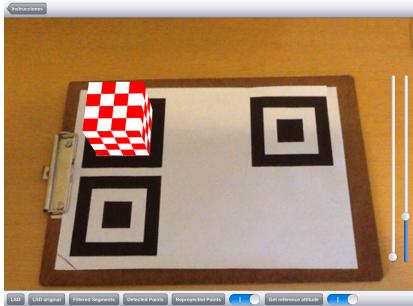


Figura 13: Captura de pantalla del caso de uso “interactividad”. Se puede ver al cubo apoyado sobre el *QlSet* de la esquina superior izquierda y los diferentes controles que ayudan a la depuración del código.



Figura 14: Captura de pantalla del caso de uso “video”. Se puede ver al video proyectado sobre el *QlSet* de la esquina superior izquierda.

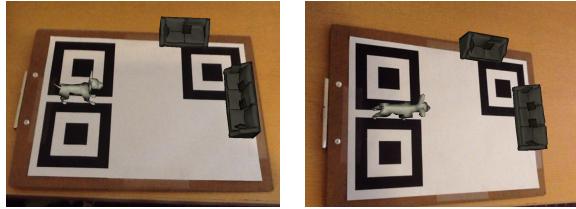


Figura 15: Dos capturas de pantalla del caso de uso “modelos”, desde dos ángulos distintos. Se pueden ver los modelos 3D pertenecientes a un perro chihuahueño y a dos sillones, uno de dos plazas y otro de tres.

Aumentada. Para eso se incorporaron a la aplicación las siguientes funcionalidades:

- Navegación por detección de QRs.
- Identificación de obras mediante SIFT.
- Comunicación con un servidor.
- Navegación por listas de cuadros.
- Interactividad con modelos.

IX-A. Diagrama global de la aplicación

Para que sea más sencilla la comprensión de los bloques que componen la aplicación, en la Figura 16 se muestra un diagrama esquemático de la misma que sirve para visualizar cómo es su flujo *a nivel de usuario*.

Al comenzar el recorrido, el usuario tiene la opción de elegir

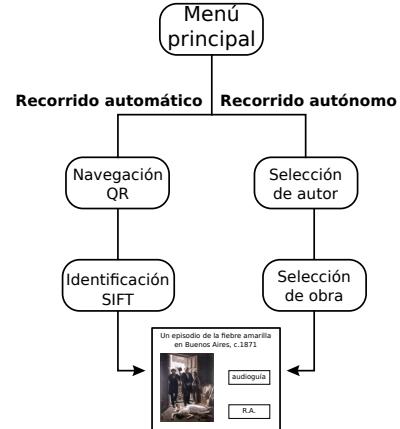


Figura 16: Diagrama global de la aplicación.

cómo recorrer el museo: de manera *autónoma* o de manera *automática*. En la opción autónoma el usuario es el encargado de elegir dentro de una lista de autores el que más le interese, y dentro de la lista de cuadros del autor seleccionado, la obra que desea contemplar en detalle. De esta manera el usuario llega eligiendo opciones al cuadro de interés y está listo para comenzar la interacción con la obra, a través de audioguías o realidad aumentada. De la otra manera de recorrer el museo, con la opción automática, el usuario tiene la opción de navegar por el museo leyendo códigos QR desplegados en las distintas salas o secciones, que sirven para identificar en qué parte del museo se encuentra el usuario. De esta manera una vez que el usuario lee el QR, la aplicación lo reconoce y despliega una foto del autor y un mensaje que invita al usuario a continuar con el recorrido.

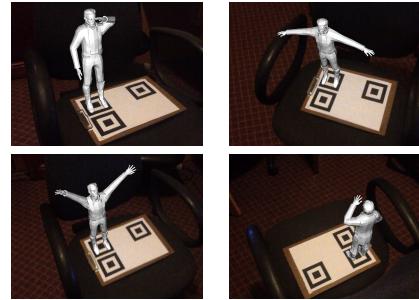


Figura 17: Escultura digital e interactiva de Artigas vista desde ángulos distintos y en posiciones distintas.

Internamente la aplicación guarda la información en la que está el usuario y la utiliza en la siguiente etapa: identificación

de la de obra. La identificación de la obra se da una vez que el usuario está frente a la misma y toma una foto de ella que es procesada y en pocos segundos la aplicación responde con la imagen original de la obra y el usuario puede comenzar la interacción con la obra, a través de audioguías o realidad aumentada. Ver Figura 16

X. CONCLUSIONES

A continuación se detallan conclusiones separadas por los distintos bloques que componen la realidad aumentada.

X-A. Detección de características

El bloque de detección de características se expande en la Figura 18.

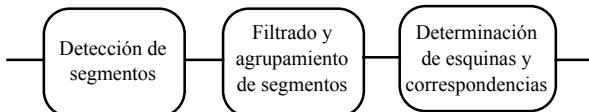


Figura 18: Diagrama de bloques de la detección de características utilizada en este proyecto.

Se vio en secciones anteriores que la detección de segmentos se realizó utilizando LSD. Se está conforme tanto con el desempeño del algoritmo como con la optimización que se le hizo para tiempo real. Sin embargo, se cree que la implementación se puede optimizar aún más.

En lo que respecta a los algoritmos de filtrado y agrupación de segmentos, determinación de esquinas y correspondencias, se puede afirmar que funcionan correctamente. De cualquier manera, se cree que para ciertas aplicaciones en particular se podría utilizar un marcador más sencillo, con una geometría más simple de detectar; lo que simplificaría los algoritmos de detección y aumentaría la aplicabilidad a otros casos de uso.

X-B. Estimación de pose

En la sección se explicaron en detalle diferentes versiones de POSIT, el algoritmo que este proyecto utiliza para estimar la pose de la cámara a partir de ciertas características de la imagen. En particular, se utilizó el **POSIT coplanar moderno** que toma como entrada ciertos puntos en un modelo predefinido y sus correspondientes en la imagen. Este es un algoritmo muy preciso bajo ciertas condiciones y veloz respecto de los algoritmos de estimación de pose tradicionales. Sin embargo, resulta no del todo lógico utilizar un algoritmo que tome puntos como entrada si las características extraídas de la imagen primeramente son segmentos.

Existe por su parte, otra versión de POSIT denominada “Soft POSIT de líneas”. Esta versión, toma como entrada líneas en una imagen y líneas en un modelo y tiene la habilidad de detectar correspondencias entre ellas y luego estimar la pose de la cámara a partir de dichas correspondencias. De haberse usado esta versión de POSIT, el segundo y tercer bloque del diagrama de la Figura 18 no hubieran sido necesarios.

Sin embargo, *Soft POSIT* de líneas no funciona para marcadores planos como los utilizados en este proyecto. Además, requiere de una estimación de la pose inicial del dispositivo lo que no hace tan trivial su uso. Por estos motivos se optó por utilizar POSIT coplanar y resolver las correspondencias en la etapa de filtrado de segmentos.

X-C. Una solución alternativa: SLAM

“*Simultaneous Localization And Mapping*” (SLAM), o en español “Localización y Mapeado Simultáneos”, es el problema de ubicar a un dispositivo de visión artificial en un determinado lugar desconocido y que este, de manera incremental, vaya construyendo un modelo del lugar y a la vez vaya ubicándose dentro del modelo.

Este problema no sólo tiene solución, sino que además muchos de los kits de desarrollo de realidad aumentada lo utilizan. Y por lo que se pudo ver, da muy buenos resultados. En particular, logra resultados sustancialmente mejores a los de este proyecto, en cuanto a los tiempos de procesamiento.

De esta manera, los bloques “Detección de características” y “Estimación de pose” del diagrama de la Figura ??, pueden ser sustituidos por un algoritmo que solucione este problema.

X-D. Demás bloques

El desempeño de los demás bloques involucrados en el proceso mediante el cual se logra la realidad aumentada no requiere de un análisis extra al realizado en secciones anteriores. ISGL3D, la herramienta utilizada para realizar renders sobre las imágenes capturadas, funciona perfectamente y los dispositivos de captura y despliegue de imágenes del iPad son muy buenos. Hubiera sido mejor capturar imágenes de mayor tamaño, para así lograr una mejor calidad a la salida. Pero cuanto mayor es el tamaño de la imagen capturada, también lo es el tiempo de procesamiento de los algoritmos de detección, en particular LSD.

AGRADECIMIENTOS

Los autores quieren agradecer especialmente a Rafael Grompone por sus consejos en la optimización de LSD y a Pablo Musé por su ayuda con el filtrado de Kalman. A Fernando Foglino por facilitar y permitir utilizar su modelo de Artigas.

A las comunidades de Software Libre y de Código Abierto, en particular a la comunidad de Acceso Abierto e Investigación Reproducible: IPOL.

Al Museo Nacional de Artes Visuales (MNAV) y al Museo Blanes. Finalmente también agradecer al Programa de Apoyo a la Investigación Estudiantil (PAIE) de la Comisión Sectorial de Investigación Científica (CSIC) por el apoyo económico para adquirir herramientas de desarrollo.