CEIG'08, Barcelona, Sept. 3-5 (2008)
L. Matey and J. C. Torres (Editors)

Cálculo de la Pose de la Cámara ante Oclusiones de un Marcador

H. Álvarez, D. Borro
CEIT and Tecnun (University of Navarra)

{halvarez, dborro}@ceit.es
Manuel de Lardizabal 15, 20018 San Sebastian, Spain

Resumen

Uno de los principales problemas de la realidad aumentada consiste en el seguimiento de la cámara (tracking), es decir, conocer la posición de la cámara dentro de la escena en todo momento. Para conseguir este objetivo existen numerosas alternativas, entre las que se encuentra el posicionamiento basado en marcadores. Estos marcadores son identificados en cada fotograma, y son utilizados como sistema de referencia a partir del cual se calcula la posición y orientación de la cámara (camera pose). Sin embargo, ante oclusiones parciales del marcador, este no puede ser identificado correctamente, y por tanto no es posible calcular la nueva pose de la cámara. Para combatir este problema han aparecido propuestas basadas en añadir múltiples marcadores al entorno, pero tienen el inconveniente de agravar el principal problema que tiene la solución basada en marcadores: adecuación del entorno. En este artículo se propone una solución al problema de oclusión de los marcadores basándose en un único marcador y utilizando coherencia temporal. El coste computacional es despreciable, lo cual lo convierte también en una solución ideal para los dispositivos móviles, que carecen de gran capacidad de cómputo. Para poder aplicar el método y mostrar los resultados que se consiguen, se ha utilizado la librería de software ARToolkitPlus.

CD: I.4.8 [Image Processing and Computer Vision]: Scene Analysis --- *Tracking*
Palabras clave: Realidad Aumentada, oclusión, ARToolkitPlus.

1. Introducción

La realidad aumentada (RA) tiene como objetivo el embeber objetos virtuales dentro del mundo real, mostrando al usuario el conjunto de objetos (virtuales y reales) como un solo mundo. Para conseguirlo se pueden utilizar diferentes dispositivos, entre los que se encuentran gafas see-through y video-through, proyectores o dispositivos móviles tales como PDAs. Una comparación de todos ellos puede encontrarse en [Azu97, ABB*01].

Los principales problemas que envuelve la RA son el tracking de la cámara, y el acoplamiento del objeto virtual con el mundo real, intentando que el usuario perciba que los dos mundos coexisten en el mismo espacio.

Dentro del problema del tracking de la cámara existen diferentes técnicas para su resolución [RBG01], entre las que se encuentra el tracking óptico. Esta alternativa basada en visión por computador se ha convertido en una de las más populares debido a sus buenos resultados y a su bajo coste en comparación con el resto. El tracking óptico puede dividirse en dos grupos: basado en marcadores y sin marcadores. El segundo de ellos utiliza las características naturales del entorno tales como bordes, esquinas, texturas, etc. para calcular la pose de la cámara. Sin embargo, además del problema de la inicialización tiene dos grandes problemas como son la acumulación del error y la reinicialización [Yua06a]. Dentro de este grupo también aparece un nuevo subgrupo, basado en el modelo, que consiste en utilizar un modelo 3D de la escena. Esta solución tiene el inconveniente de que algunas veces la generación del modelo 3D puede resultar difícil y puede que requiera mucho tiempo su elaboración.

En la alternativa basada en marcadores, por el contrario, el principal problema viene dado por el trabajo y limitación que supone intervenir el entorno. Siempre que se quiera exportar un sistema a otro lugar habrá que adecuar el nuevo entorno al sistema, añadiendo los marcadores a sus nuevas localizaciones. Además, debido a que se necesita tener en todo momento un marcador visible para obtener la pose de la cámara, existen varias propuestas [Uml02, KB99, Tat07] que utilizan múltiples marcadores para conseguir un sistema más robusto. No obstante, el aumento de robustez supone incrementar la intervención del entorno y una menor adecuación del entorno supone un sistema menos

robusto. Como solución a esta relación, el método que es expuesto en este trabajo no incorpora ningún marcador extra al entorno y consigue aumentar la robustez del sistema al permitir que los marcadores sean parcialmente ocluidos. ARTag [Fia05] es un software comercial que también aumenta la robustez del sistema al permitir ciertas oclusiones, pero nuestros experimentos muestran que nuestro sistema es más robusto, aunque menos preciso.

Existen diferentes tipos de marcadores [SW07, Fia05], tal y como aparece en la Figura 1. En el proceso de identificación de los marcadores basados en patrones, un criterio común a utilizar para comparar la imagen actual con una base de datos de *templates* es por medio del *L2 norm*. Se calcula el *L2 norm* del marcador actual con cada uno de los marcadores almacenados y se devuelve aquel marcador almacenado cuyo *L2 norm* sea menor. Para calcular el *L2 norm* se utilizan los niveles de gris del interior de los marcadores, mostrando una correlación entre ambos. Tras las comparaciones, se aceptará un emparejamiento entre marcadores en relación a un umbral prefijado. Este proceso tiene el problema de que dos marcadores similares darán resultados muy parecidos, provocando interpretaciones erróneas, y la diferente configuración de los niveles de gris en función de la rotación del marcador. Además, se necesita una fase de entrenamiento para fijar el umbral de aceptación. Estos marcadores son utilizados por las primeras versiones de ARToolkit.

Las otras dos opciones codifican el interior del marcador digitalmente generando un identificador único para cada marcador. Además no todos los bits del interior pertenecen al identificador, sino que parte del conjunto de bits se utiliza para generar el identificador y el resto como redundancia para corregir errores. La diferencia entre ambos reside en el número de bits que se codifican en el interior (BCH 6x6 pixels y DataMatrix 144x144 pixels, permitiendo la codificación de texto).

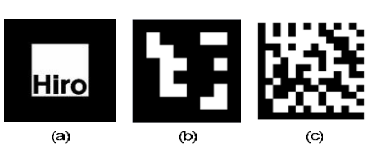


Figura 1: *Marcador basado en patrones (a), marcador BCH (Bose, Chaudhuri, Hocquenghem) (b), marcador DataMatrix codificando el texto "CEIG 2008"(c).*

Para la implementación del método se ha utilizado la librería de software de acceso público ARToolkitPlus [WS07], que ha sido usada en numerosas ocasiones para calcular en tiempo real la pose de la cámara relativa a un marcador físico (Figura 2). Esta librería utiliza marcadores BCH (Figura 1b).

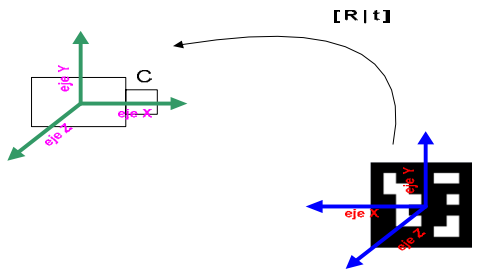


Figura 2: *Pose de la cámara relativa al marcador.*

En la Sección 2 se expondrá cómo interactúa ARToolkitPlus y nuestro método. Posteriormente, en la Sección 3 se presentará una descripción detalla del método, y en la Sección 4 se mostrarán los resultados obtenidos, incluyendo una comparativa de nuestro método y ARTag. Finalmente, en la Sección 5 se ofrecerá una visión sobre los trabajos futuros y en la Sección 6 aparecerá una conclusión de todo el trabajo.

2. Interacción ARToolkitPlus - Occlusion

Se ha modificado ARToolkitPlus para lograr un tracking más robusto ante oclusiones parciales de los marcadores. Cuando se detecta una oclusión, es decir, cuando ARToolkitPlus no es capaz de calcular la nueva pose, se hacen ciertos cálculos y estimaciones para obtener la nueva traslación en los 3 ejes (X, Y, Z) y la nueva rotación sobre el eje Z (eje perpendicular al plano de imagen), ya que para el resto de rotaciones no se dispone de información suficiente. Las poses calculadas por ARToolkitPlus se referenciarán como *poses visibles*, al conjunto de cálculos y estimaciones se le ha asignado el nombre de *Occlusion*, y a las poses calculadas por *Occlusion poses de oclusión*.

Para inicializar todos los cálculos y obtener las *poses de oclusión* se necesitará al menos haber calculado dos *poses visibles*, ya que como se mostrará en los siguientes apartados esta información será sobre la que se basen todos los cálculos y estimaciones. Por tanto, el esquema resultante es el que aparece en la Figura 3:

```
SI ARToolkitPlus es capaz de calcular la nueva pose ENTONCES
        #poses_visibles = #poses_visibles + 1;
        DEVOLVER poses visibles
SINO
        SI número de poses visibles desde la última oclusión >= 2 ENTONCES
                #poses_visibles = 0;
                DEVOLVER pose de oclusión
        FIN SI
FIN SI
```

Figura 3: *Interacción entre ARToolkitPlus y Occlusion.*

Nótese que *Occlusion* inicializa el número de *poses visibles* calculadas, por lo que cada vez que se comience a

ejecutar *Occlusion* se hará con las poses visibles correspondientes a la última secuencia visible, es decir, la información es actualizada dinámicamente.

Además, las dos *poses visibles* que se almacenarán generalmente se corresponderán con las dos últimas, ya que de esta forma se utilizará la información más actual que es proporcionada por el comportamiento del marcador. No obstante, debido a la necesidad de que exista una diferencia cuantificable entre ambas poses almacenadas, para dispositivos en los que la velocidad de renderizado no sea suficientemente alta será necesario guardar la última pose visible y alguna otra pose visible desplazada *k* fotogramas.

$$Poses\_Visibles\_Alacenadas = \{ P(i), P(i-k) \} \qquad (Eq.\ 1)$$

Donde **P**(*i*) se refiere a la pose de ejecución en el fotograma *i* y **P**(*i*-*k*) la pose de ejecución en el fotograma *i*-*k*, siendo *k* un desplazamiento definido en preproceso. Tal y como aparece en la Sección 4.1, para tasas de ~8 fps (fotogramas por segundo) se utilizará $k = 2$, y para tasas del orden de 30 fps $k = 1$. A partir de ahora, cuando se haga referencia a las dos últimas poses visibles no se especificará el valor de *k*, aunque implícitamente se habrá tenido en consideración.

3. Cálculo de la poses de oclusión

En la sección anterior se ha enunciado la necesidad de almacenar la pose de la cámara, pero además de ella también se almacenarán los siguientes datos, que serán accedidos por las diferentes fases de *Occlusion*:

- Coordenadas (en pixel) de las 4 esquinas de la caja contenedora del marcador y que es paralela al eje X (AABB, *Axis Align Bounding Box*, Figura 4a)
- Coordenadas (en pixel) del centro del AABB.
- Coordenadas (en pixel) de las 4 esquinas de la caja contenedora del marcador y que está orientada (OBB, *Oriented Bounding Box*, Figura 4b).

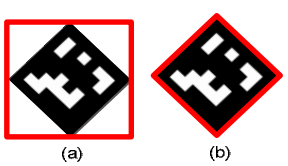


Figura 4: *AABB (a) y OBB (b) de un marcador.*

ARToolkitPlus busca en la imagen objetos negros, cuadrados y con figuras blancas en su interior, ya que son características que definen a sus marcadores [SW07]. Por tanto, cuando un marcador está parcialmente ocluido, su parte visible será detectada por ARToolkitPlus, pero no será interpretado como un marcador. Basándonos en ello, tendremos que aplicar un filtro entre los objetos detectados por ARToolkitPlus para conocer cuál es el que se corresponde con la parte visible del marcador. Nos referiremos a este proceso como la selección del candidato, donde los objetos negros, cuadrados y con figuras blancas detectados por ARToolkitPlus son candidatos, y la parte visible del marcador es el candidato a seleccionar.

Para la selección del candidato ejecutaremos el siguiente algoritmo

```
Distancia_min = INTEGER_MAX;
Id_candidato = -1;

Para cada candidato i hacer
  si | distancia(i) | < VENTANA entonces                                   (a)
    si ( distancia(i) < Distancia_min ) entonces                          (b)
      si ( ( area(i) < 1.25 * area_marcador ) && ( area(i) > 0.75 * area_marcador) ) entonces  (c)
          Distancia_min = distancia(i);
          Id_candidato = i;
      Fin si
    Fin si
  Fin si
Fin para cada
```

Figura 5: *Algoritmo para calcular el marcador dentro de un conjunto de candidatos.*

Donde *distancia(i)* es la distancia entre el centro del candidato *i* y el centro del marcador de referencia, y *VENTANA* es un umbral.

La primera condición (a) elimina los candidatos que no están cerca del marcador de referencia. La idea en la que se apoya este filtro es que el movimiento realizado entre dos fotogramas consecutivos es pequeño (coherencia temporal), y por tanto el marcador realizará pequeños desplazamientos entre fotogramas. *VENTANA* representa el lado del cuadrado centrado en el centro del marcador de referencia, y es un parámetro fijado por el usuario. Con un valor grande se permitirán movimientos bruscos, pero a cambio se aumentará la probabilidad de asociar erróneamente un candidato como marcador, ya que el área de búsqueda es mayor. Con una resolución de 320x240 se han obtenido buenos resultados con valores de *VENTANA* comprendidos entre 50 y 75.

La condición (b) selecciona aquel candidato que se encuentre más cercano al marcador de referencia, pero que se encuentre dentro de la ventana de búsqueda. Asume la misma idea que el filtro anterior.

Por último, la condición (c) asegura que las áreas del candidato y el marcador de referencia sean parecidas, no permitiendo una diferencia entre áreas mayor que el 25%. Esta condición reduce la posibilidad de detectar un nuevo candidato como marcador cuando el marcador ha sido totalmente ocluido, pero ese nuevo candidato no se corresponde con el marcador.

Una vez se ha seleccionado el candidato simplemente habrá que extraer las 4 esquinas del AABB del candidato, que es una información ya calculada por ARToolkitPlus en la fase de procesamiento de la imagen.

El centro del AABB se obtendrá de forma trivial a partir de las equinas del AABB, y el OBB requerirá un procesamiento extra (Sección 3.2.2).

Dentro del cálculo de las poses de oclusión se distingue entre la forma de calcular la primera pose de oclusión y el resto.

3.1. Primera pose de oclusión

La primera pose de oclusión se estima únicamente en función de las últimas dos poses visibles. Se calcula cuál fue la traslación entre las dos últimas poses visibles y se aplica esa misma traslación a la última de ellas (la rotación se mantiene), obteniendo la nueva pose de oclusión.

$$Pose\_fotograma\_i{+}1 = Pose\_fotograma\_i + Traslación\,(fotograma\_i, fotograma\_i{-}k); \qquad (Eq\ 2)$$

Esta primera estimación se basa en que el movimiento del marcador respecto a la cámara seguirá la misma trayectoria en el intervalo de dos fotogramas consecutivos. Tras este número de fotogramas, la cámara comenzará a describir cualquier trayectoria o podrá detener su movimiento, de manera que esta premisa no será válida.

Este paso se utiliza como inicialización de los datos utilizados por *Occlusion*. Se obtienen diferentes proporciones que serán utilizadas en pasos posteriores. Se calcula la proporción *Desp_Pixel* → *Desp_Mundo3D*. Como se ha asumido un movimiento idéntico al de la dos últimas poses visibles, podemos calcular el desplazamiento (*TrX*, *TrY*) 3D (diferencia entre las traslaciones de las poses) y el 2D (diferencia ente las traslaciones de los centros del marcador en la imagen); y por consiguiente la relación *Desp_Pixel* → *Desp_Mundo3D*. Para cada coordenada obtendremos su propia proporción.

$$\begin{aligned} propX &= TrX(pose_i - pose_{i-1}) / TrX(centro\_frame_i - centro\_frame_{i-1}); \\ propY &= TrY(pose_i - pose_{i-1}) / TrY(centro\_frame_i - centro\_frame_{i-1}); \end{aligned} \qquad (Eq\ 3)$$

3.2. Resto de poses de oclusión

Dentro del resto de poses de oclusión también se distingue entre el proceso de cálculo de las traslaciones y el procedimiento llevado a cabo para obtener la rotación. Se recuerda que cuando hablamos de traslaciones y rotaciones, estamos hablando de calcular la traslación y rotación de la cámara mientras el marcador está parcialmente ocluido.

3.2.1. Traslación

La traslación X e Y se calcula llevando a cabo un seguimiento del centro del marcador. Se calcula cuál ha sido la traslación en pixels del centro del marcador entre el último fotograma y el actual y se extrapola a la pose de la cámara en función de la proporción *Desp_Pixel* → *Desp_Mundo3D* calculada en la sección anterior.

$$\begin{aligned} tX' &= propX * distancia\_centro\_x; \\ tY' &= propY * distancia\_centro\_y; \end{aligned} \qquad (Eq.4)$$

Además, el centro del marcador es estimado por el centro del rectángulo contenedor (AABB) de la parte visible del marcador (punto azul de la Figura 6). Puesto que ambos puntos sufrirán el mismo movimiento, y el procesamiento del segundo de ellos es menos costoso, se ha optado por esta segunda alternativa. Este punto se calculará en base las esquinas del AABB, que son proporcionadas por el análisis de ARToolkitPlus.

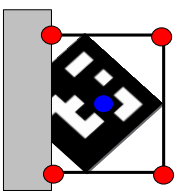


Figura 6: *AABB de la parte visible del marcador*.

Nótese que, cuando el tamaño de uno de los lados del AABB se decrementa (el marcador continua ocluyéndose) en 2 unidades entonces el centro se desplaza una única unidad. Esta relación también es análoga cuando el marcador se ocluye menos. Así, un desplazamiento de *d* unidades del centro del AABB de la parte visible del marcador equivaldrá a un desplazamiento *2*d* del marcador. Sin embargo, esta proporción es dependiente del eje con el que se esté ocluyendo. Una oclusión con los bordes verticales (eje Y) supone aplicar la relación a la coordenada X y viceversa. Además, en las esquinas se aplicará a ambas coordenadas al colisionar con ambos bordes. Así pues, será necesario conocer con que borde está siendo ocluido el marcador. Puesto que la resolución de la imagen es un parámetro conocido, podemos utilizar esta información junto con la posición de las 4 esquinas del AABB para conocer el borde de colisión (Figura 7).

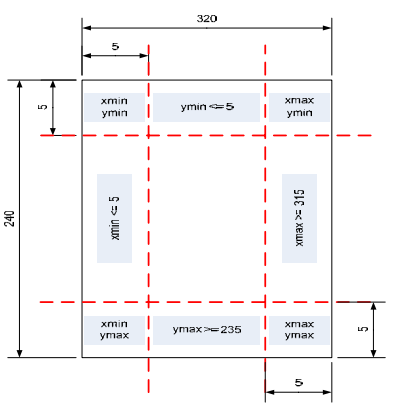


Figura 7: *Detección del borde de colisión para una imagen 320x240. Se comparan las coordenadas máximas y mínimas de las 4 esquinas del AABB (xmin, xmax, ymin, ymax) con las posiciones de los bordes*.

Para el cálculo de la traslación Z, por el contrario, será necesario observar algún cambio de escala en el marcador. Si la distancia entre la cámara y el marcador se reduce, los objetos aparecerán más grandes en la imagen, y cuando se aumenta, el tamaño de los objetos en la imagen disminuirá. Para aplicar esta relación, será necesario realizar un seguimiento del lado visible del marcador, es decir, el lado que no está en colisión con ningún borde. Así, la traslación en Z se obtiene como la diferencia de tamaño del lado visible ($visible_i - visible_{i-1}$) entre dos fotogramas consecutivos multiplicada por la proporción correspondiente a Z (*propZ*).

$$tZ' = propZ * (visible_i - visible_{i-1}); \qquad (Eq.5)$$

La proporción Z refleja la relación que existe entre el incremento del lado en la imagen (en pixels) y la variación de la distancia entre el marcador y la cámara (en coordenadas 3D). Además, esta proporción no es fija, sino que depende de la distancia entre la cámara y el marcador (valor de Z). Cuando el marcador y la cámara se encuentran próximos, una variación en el tamaño del lado supone una pequeña traslación Z (propZ pequeño), mientras que cuando estén alejados supondrá una traslación Z notable (propZ grande). Esto requiere que antes de comenzar con la ejecución (*offline*) se hayan calculado diferentes valores de *propZ* para diferentes valores de Z (Sección 4.1). Además, esta dependencia con el valor de Z no ocurre sólo con el tamaño del objeto, sino también con el movimiento de los objetos, por lo que también se requerirá un ajuste de las proporciones X e Y al cambiar el valor de Z. Esto mismo cuando por nuestro campo de visión vemos pasar a un motorista cerca de nosotros, que aparece y desaparece en pocos segundos, y cuando vemos al mismo motorista a kilómetros de distancia, que tarda varios segundos en salir de nuestro campo de visión.. Por tanto, cuanto mayor sea Z, mayor serán las proporciones propX y propY y viceversa, ya que un pequeño desplazamiento en la imagen puede suponer un gran desplazamiento en el mundo 3D (el motorista que circula a distancia recorre cientos de metros que para el observador son pocos centímetros en la imagen). Bastará con realizarlo sólo la primera vez para cada cámara, a modo de calibración.

Si la resolución de la imagen no es cuadrada, también influirá en la proporción, ya que cuanto menor número de pixels haya en un eje, una diferencia de un pixel supondrá un mayor incremento/decremento en el desplazamiento 3D. Como en este caso se ha trabajado con imágenes de resolución no cuadrada, se ha declarado una proporción Z para el eje X (*propZX*) y otra para el eje Y (*propZY*). Esta es la misma razón por la cual las proporciones utilizadas en la Sección 3.1 son diferentes para el eje X e Y. Todo ello queda resumido en la Figura 8.

El cálculo de la oclusión en las esquinas se ha tratado de forma diferente, ya que no existe ningún lado visible por completo. Por tanto, al ejecutarse una traslación proporcional en X e Y, ambos lados se incrementan (decrementan) proporcionalmente y no se puede saber si ese cambio en el tamaño de los lados se debe a una traslación conjunta y proporcional de X e Y o sólo a una traslación en Z. Aparece una ambigüedad. Así, si alguna de las esquinas del rectángulo contenedor del marcador se encuentra en una de las esquinas de la imagen (Figura 7), entonces la variación del tamaño de los lados X e Y debe ser muy parecida (prácticamente proporcional) para no confundir una traslación Z con una traslación conjunta de X e Y. Basándose en que la probabilidad de que un usuario realice una traslación proporcional en X e Y a la vez es baja, se ha añadido una restricción adicional, que consiste en que la diferencia entre las proporciones de los tamaños de cada lado sea prácticamente igual, reduciendo la probabilidad de error. Así, se considerará una traslación Z cuando los lados crezcan y decrezcan prácticamente con la misma proporción; y traslación X e Y en caso contrario (Figura 9).

```
si se detecta traslación en Z entonces
        si lado visible horizontal entonces
                propZ = proporciónZX ( Z );
        sino
                propZ = proporciónZY ( Z );
        fin si;
        propX = proporciónX ( Z );
        propY = proporciónY ( Z );
fin si;
```

Figura 8: *Gestión de las proporciones en función del valor de Z. proporcionZX, proporciónZY, proporciónX y proporciónY son funciones que devuelven la correspondiente proporción en función del valor de Z que se les pasa como parámetro. Las proporciones que se devuelven son las almacenadas como consecuencia del proceso de entrenamiento de la Sección 4.1.*

```
si | ladoX_i / ladoX_i-1 − ladoY_i / ladoY_i-1 | ~ 0 entonces
        Traslación Z
sino
        Traslación X + Traslación Y
fin si
```

Figura 9: *Restricción para diferenciar entre traslación Z y traslación X e Y.*

Destacar que se ha decidido asignar el caso con menor probabilidad a la traslación en Z debido a que se ha partido del supuesto de que el usuario cuando detecte el marcador

en oclusión intentará trasladarlo a una zona visible, y por tanto necesitará hacer una traslación en X o Y.

Para que esta estimación funcione correctamente, se deberán evitar las rotaciones sobre los ejes X e Y, ya que de lo contrario afectarán al tamaño del lado visible y se interpretarán como traslación en Z.

3.2.2. Rotación

La rotación sólo se calcula para el eje Z, ya que para el resto de los ejes no se dispone de información suficiente. No obstante, se podría haber utilizado la diferencia del tamaño del lado visible para interpretarlo como rotación sobre el eje X (diferencia en el tamaño del lado visible cuando este es horizontal) o sobre el eje Y (diferencia en el tamaño del lado visible cuando este es vertical), pero entonces se eliminaría la posibilidad de traslación Z, ya que esa es la información que es utilizada para la detección de la traslación en Z (Sección 3.2.1) En nuestro caso, este método se ha desarrollado paralelamente a una aplicación en la que los marcadores van a estar ubicados en paredes, y el usuario va a ver la información aumentada a través de una PDA. Por tanto, limitar la traslación en Z (acercamiento del usuario al marcador), resulta más restrictivo que eliminar las rotaciones en los ejes X (girar la PDA como las ruedas de un coche) e Y (girar la PDA como una batidora).

Para calcular la rotación sobre el eje Z (eje perpendicular al plano de imagen), se observará el ángulo rotado por el OBB del marcador entre dos fotogramas consecutivos. El OBB es similar al AABB, ya que ambos encierran en un rectángulo/cuadrado un objeto; pero el primero de ellos lo realiza con la misma orientación que el objeto que encierra y el segundo de ellos es paralelo al eje de coordenadas (Figura 4).

El procedimiento para calcular la rotación sobre el eje Z se resume en los siguientes pasos:

- Calcular el OBB del marcador en el fotograma actual.
- Calcular el orden correcto de las esquinas.
- Calcular el ángulo rotado entre el OBB del fotograma anterior y el fotograma actual.

Para poder calcular el OBB será necesario procesar la imagen, pero bastará con analizar el área definida por el AABB dentro de la imagen segmentada y binarizada por ARToolkitPlus.

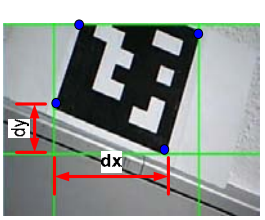


Figura 10: *Relación entre el AABB y OBB de un marcador. Las rectas verdes representan los límites del AABB, y los puntos azules las posibles esquinas del OBB.*

Para encontrar las esquinas del OBB, por cada esquina del AABB recorreremos la imagen en el eje X y en el eje Y hasta encontrar el primer pixel que pertenezca al marcador, obteniendo para cada esquina del AABB dos valores: *dx* y *dy*. Estos valores reflejan la distancia (en pixels) que hay desde la esquina del AABB a una posible esquina del OBB. El siguiente paso consiste en escoger un sólo valor por cada esquina del AABB, que se corresponderá con el mínimo entre *dx* y *dy*. Así, cada esquina del AABB generará una posible esquina del OBB (puntos azules de la Figura 10).

Sin embargo, sólo 3 de los 4 puntos seleccionados pertenecen al OBB. Para encontrar los 3 correctos, en cada punto se calculan los dos vectores que se forman con los dos puntos vecinos y se calcula el ángulo que forman los dos vectores. Los 3 puntos seleccionados serán aquellos que formen los dos vectores cuyo ángulo se aproxime más a un ángulo recto.

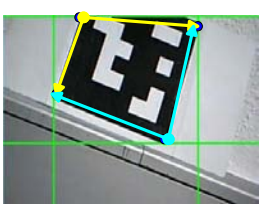


Figura 11: *Selección de los puntos que pertenecen al OBB. Los 3 puntos seleccionados son los que forman los dos vectores azules.*

El cuarto punto que forma el OBB se obtendrá atendiendo a la regla del paralelogramo.

Para calcular el orden correcto de las esquinas del OBB hace falta identificar en cada fotograma el mismo lado del OBB, objetivo que se conseguirá mediante la gestión de un historial de las esquinas del OBB. En cada fotograma se almacenan las 4 esquinas del OBB en el sentido de las agujas del reloj, partiendo siempre desde el mismo punto inicial. Para conseguir en cada fotograma el mismo punto de partida, las esquinas del OBB en el fotograma actual se ordenan en el sentido de las agujas del reloj, comenzando por cualquiera de ellas. Posteriormente se permutan las posiciones de esos 4 puntos (también en el sentido de las agujas del reloj), hasta que se encuentra la configuración que minimice la distancia de los puntos entre la configuración actual y la configuración que se estableció como óptima en el fotograma anterior.

Una vez se han calculado las esquinas del OBB y se han almacenado en el orden correspondiente, sólo falta calcular el ángulo de rotación entre dos OBBs consecutivos. Para ello bastará con comparar el ángulo rotado por uno de los lados, siempre y cuando se compare el mismo lado en los dos OBBs (Figura 12). Para garantizar esto último seleccionaremos siempre las dos primeras esquinas almacenadas, ya que la correspondencia entre los puntos (lados) será la correcta debido al paso anterior. Por tanto, el

ángulo rotado entre los dos OBBs será el ángulo rotado por los dos vectores de dicho lado (un vector por cada uno de los dos fotogramas).

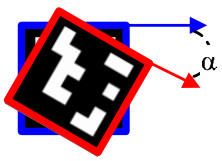


Figura 12: *Cálculo del ángulo de rotación (α) entre 2 OBBs consecutivos (OBB azul y OBB rojo).*

3.2.3. Actualización de la pose de la cámara

La pose de la cámara es la matriz 3x4 [**R** | **t**], donde **R** es la matriz 3x3 de rotación y **t** es el vector 3x1 de traslación. Esta matriz representa la rotación y traslación necesaria para pasar del eje de coordenadas centrado en el marcador al eje de coordenadas centrado en la cámara.
En el apartado 3.2.1 se han obtenido *tX'*, *tY' y tZ'*, y en el apartado 3.2.2 el ángulo de rotación α en el eje Z. Por tanto, la nueva pose de la cámara vendrá definida por la siguiente matriz de rotación y vector de traslación:

$$t' = t + \begin{pmatrix} tX' \\ tY' \\ tZ' \end{pmatrix} \qquad (Eq.6)$$

$$R' = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} * R \qquad (Eq.7)$$

4. Resultados experimentales

4.1. Proporciones 2D-3D

Para calcular las proporciones *Desp_Pixel* → *Desp_Mundo3D* se han realizado diferentes experimentos a dos estudiantes. Uno de ellos no conocía el objetivo del experimento, mientras que el otro sí sabía que se quería obtener. Con esto se ha tratado de evitar que el conocimiento total del contexto pudiese influir en el tipo de movimientos realizados, condicionando los resultados. Introduciendo al estudiante sin conocimiento se ha garantizado la obtención de todo tipo de movimientos y velocidades.

Cada uno de ellos ejecutó ARToolkitPlus 2 veces durante 60 segundos, con total libertad de movimiento. La única restricción que se les impuso a ambos fue la necesidad de ocluir el marcador, ya que de lo contrario, no se obtendrían los valores que se deseaban analizar.

En la primera ejecución se sentaron enfrente del ordenador, con la cámara web apuntado hacia ellos. Se les proporcionó un marcador y se les cronometró. Las características y configuración del ordenador son las siguientes:

- *Pentium 4 CPU 3.00 GHz, 1 GB RAM*
- *Nvidia GeForce 6800*
- *Logitech QuickCam Connect (~30 fps)*
- *Resolución de imagen = 320x240*
- *ARToolkitPlus 2.1.1*
- *VENTANA = 75*
- *k = 2*

Para la segunda ejecución se les proporcionó una PDA Dell Axim X 50v. Se les proporcionó el mismo marcador y se les cronometró también durante 60 segundos. Las características de la PDA son las siguientes:

- *Intel PXA270 a 624MHz, 64MB RAM*
- *Intel 2700G (Marathon), 16MB RAM*
- *Cámara Spectec SDC-003A SD (~10fps)*
- *ARToolkitPlus 2.1.1*
- *VENTANA = 75*
- *k = 1*

Cada uno de los dos estudiantes realizó un orden de ejecución diferente. Uno de ellos utilizó primero el ordenador y luego la PDA; y el otro usó primero la PDA y luego el ordenador. Con esto se trató de equilibrar en la segunda ejecución la habilidad adquirida por cada uno de ellos en la primera ejecución respecto a los movimientos con el marcador [DF00].

Los resultados obtenidos se muestran en las dos siguientes tablas, una mostrando las proporciones para la cámara web, y otra con las proporciones de la cámara de la PDA.

PC execution Z value	**propX**	**propY**	**propZX**	**propZY**
< 400	0.80	0.80	3.18	2.51
[400, 500]	1.00	0.99	5.36	5.88
> 500	1.40	1.34	8.64	9.10

Tabla 1: *Proporciones 2D-3D para la ejecución del PC, clasificados por los valores de Z.*

PDA execution Z value	**propX**	**propY**	**propZX**	**propZY**
< 400	0.73	0.70	2.28	2.38
[400, 500]	0.87	1.00	4.01	4.85
> 500	1.30	1.41	6.76	6.04

Tabla 2: *Proporciones 2D-3D para la ejecución de la PDA, clasificados por los valores de Z.*

La primera columna representa el rango de los valores de Z, que representa la distancia entre el marcador y la cámara en coordenadas 3D. Son valores dependientes del contexto de la aplicación, tal y como se ha citado en la Sección 3.2.1, y si se desea precisión cuando la cámara y el marcador están próximos será necesario crear varios rangos y de pequeña longitud para pequeños valores de Z.

El valor de la proporción se ha calculado obteniendo la mediana de todos los valores registrados durante las dos ejecuciones de los dos estudiantes. Se ha escogido la mediana y no la media ya que esta primera reduce más la influencia de valores erróneos (*outliers*).

4.2. Coste computacional

En la siguiente tabla se muestra el coste computacional que supone añadir las oclusiones a ARToolkitPlus. Para valorarlo, se ha contabilizado la diferencia en el número de fotogramas con y sin la presencia de oclusiones. Los valores de la tabla representan la velocidad de renderizado, es decir, los fotogramas por segundo.

	ARToolkitPlus	ARToolkitPlus + Occlusion		
		Sin ocluir	Ocluyendo	
			Tr	Tr + Rt
PC	27-29	27-29	27-29	27-29
PDA	~8	~6	~8	~8

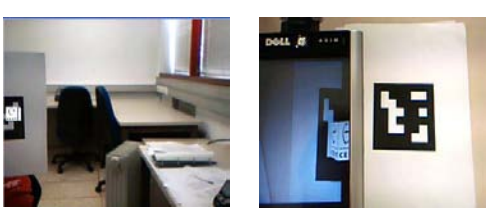
Tabla 3: *Comparativa en la velocidad de renderizado entre usar y no usar las oclusiones. Los valores representan los fotogramas por segundo. (Tr = traslación, Rt = rotación).*

Prácticamente no hay diferencias en la velocidad de renderizado entre utilizar y no utilizar las oclusiones. La única pequeña diferencia aparece en la PDA, cuando ARToolkitPlus está en ejecución, sin ocluir el marcador. Esta disminución de velocidad es debida al coste de almacenar los datos que serán utilizados por las oclusiones cuando el marcador se ocluya.

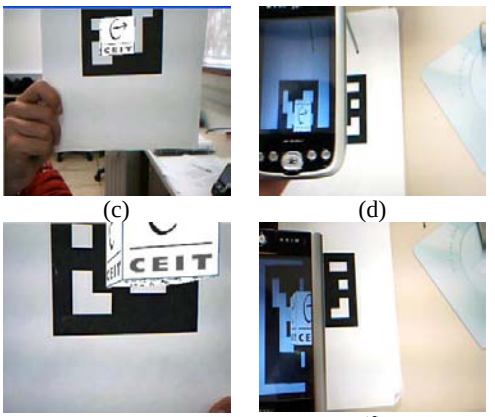
Cuando el marcador está ocluido, no hay diferencias en la velocidad de renderizado entre usar y no usar las oclusiones debido a que se evita todo el coste computacional que ARToolkitPlus requiere para obtener la pose de la cámara a partir de cuatro puntos coplanares.

4.3. Resultados

Las siguientes imágenes muestran el estado de ejecución cuando el marcador está parcialmente ocluido y se realiza una traslación sobre cualquiera de los 3 ejes. La columna de la izquierda se corresponde con imágenes obtenidas con el PC, y la columna de la derecha se corresponde con imágenes obtenidas a través de la PDA.



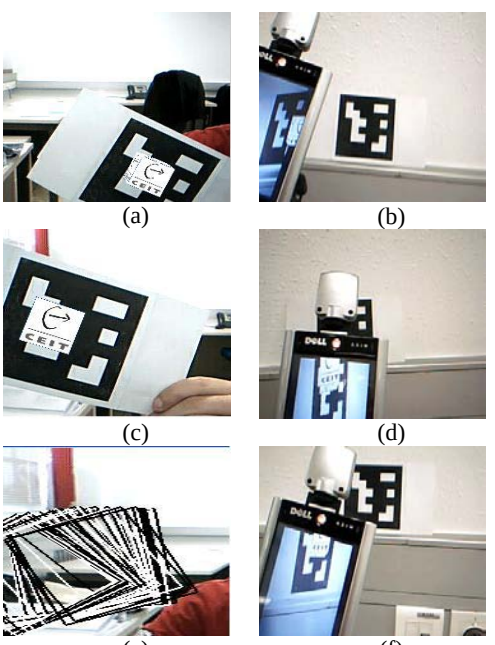
(a) (b)



(c) (d) (e) (f)

Figura 13: *Diferentes traslaciones cuando el marcador está parcialmente ocluido. Traslación en el eje X (a) y (b). Traslación en el eje Y (c) y (d). Traslación en el eje Z (e) y (f).*

Las siguientes imágenes muestran el estado de ejecución cuando el marcador está parcialmente ocluido y se realiza una rotación en el eje Z (perpendicular al plano de la imagen). La columna de la izquierda se corresponde con imágenes obtenidas con el PC, y la columna de la derecha se corresponde con imágenes obtenidas a través de la PDA.

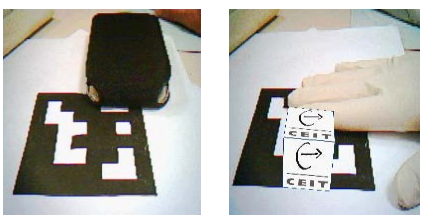


(a) (b) (c) (d) (e) (f)

Figura 14: *Diferentes rotaciones cuando el marcador está parcialmente ocluido. La imagen (e) es una secuencia continua del OBB del marcador entero, donde se aprecia la*

rotación aplicada, aun estando el marcador parcialmente ocluido (los OBBs de los fotogramas pares están dibujados en blanco y los fotogramas impares en negro).

En la Figura 15 también se pone de manifiesto la ventaja de utilizar objetos blancos a la hora de interactuar con los marcadores, ya que esto favorece la segmentación de la imagen y la selección del candidato adecuado (Sección 3).



(a) (b)

Figura 15: *Comportamiento de Occlusion ante la interacción con objetos negros (a) e interacción con objetos blancos (b).*

4.4. Occlusion vs ARTag

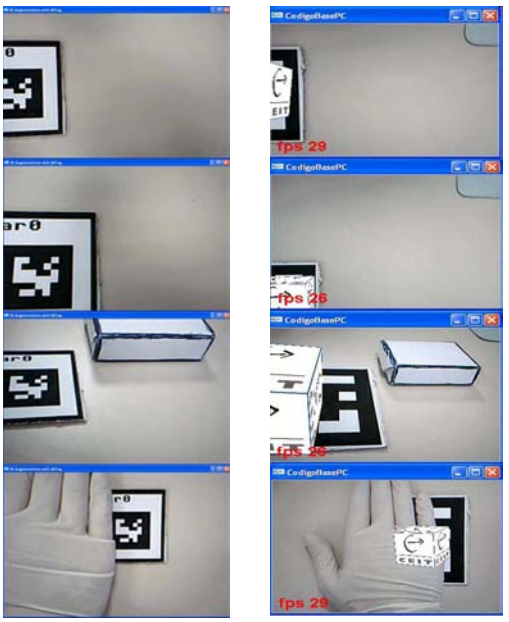
ARTag es un software comercial de Realidad Aumentada basado en marcadores. Una de las principales diferencias con ARToolkitPlus es la forma en la que se detectan cuadriláteros en la imagen, que son considerados como potenciales marcadores. ARTag utilizada métodos de extracción de bordes (a varias escalas), mientras que ARToolkitPlus usa segmentación basada en un umbral, que es calculado automáticamente en cada fotograma. Esto otorga a ARTag una mayor robustez en la detección de cuadriláteros en la imagen, y por consiguiente, también tiene mayor robustez ante oclusiones.

Puesto que no se dispone de la licencia de ARTag, la comparativa frente a nuestro método se presenta de forma visual. En la página web de ARTag (*www.artag.net*) permiten la descarga de una demo del sistema, que es la que se ha utilizado en este artículo.

Tal y como muestra la Figura 16, nuestro método permite oclusiones más extremas en todos los casos. Sin embargo, hay casos en los que la pose de la cámara no es del todo precisa.

Debido a que ARTag no realiza ninguna estimación, las oclusiones que permite (cualquier traslación y rotación) son más precisas que las ofrecidas por nuestro sistema, que son en base a mediciones y heurísticos. Por tanto, podríamos resumir diciendo que ARTag es menos robusto y preciso ante oclusiones que nuestro método.

Otro aspecto a considerar a favor de ARToolkitPlus y nuestro sistema, es que se puede ejecutar en entornos móviles, ventaja de la que carece ARTag.



(a) (b)

Figura 16: *Respuesta ante oclusión parcial del marcador de ARTag (a) y ARToolkitPlus (b). De arriba abajo: oclusión lateral, oclusión con una esquina, oclusión donde el marcador y el plano de la cámara no son paralelos, oclusión con la mano del usuario.*

5. Trabajos futuros

En las secciones anteriores se ha expuesto el método implementado para mejorar la robustez de un sistema de Realidad Aumentada basado en marcadores frente a oclusiones. Además, muchos de los cálculos han quedado limitados debido a que se ha querido ofrecer una solución tanto para PC como para entornos móviles. Sin embargo, todavía presente algunas limitaciones, como las rotaciones en los ejes X e Y. Además, hay casos en los que la pose de la cámara obtenida no es lo suficientemente precisa. Por tanto, creemos que mejorando la parte de tratamiento de imagen, que hasta ahora se utilizado la información extraída por ARToolkitPlus, y como se ha expuesto en la Sección 5 es peor que la de otros sistemas existentes, nuestro sistema mejoraría. Además, se realizará un estudio para comprobar la viabilidad del uso modelos estocásticos, como puede ser el filtro de Kalman.

6. Conclusión

En este artículo se ha presentado un nuevo método para el cálculo de la traslación en cualquiera de los 3 ejes (X, Y, Z), así como la rotación en el eje Z (perpendicular al plano de imagen), cuando un marcador está parcialmente ocluido. Necesita un coste computacional despreciable, ya que sólo requiere un análisis de una pequeña área de la imagen, lo

cual lo convierte en una solución ideal para entornos móviles, como pueden ser las PDAs.

En comparación con las soluciones propuestas por otros autores para abordar el problema de las oclusiones de marcadores, nuestro método no requiere ningún marcador extra y permite oclusiones más extremas. A cambio necesita un proceso *offline* para poder obtener los valores de las proporciones que son utilizadas por las estimaciones que generan la nueva pose de la cámara. Este proceso sólo es necesario hacerlo una vez para cada cámara y resolución de imagen, a modo de calibración. Por tanto, es un proceso de poco coste y no dependiente del entorno.

Aunque se ha utilizado ARToolkitPlus para la implantación de nuestro método en una aplicación real, no tiene dependencia de este paquete de software. La información que es calculada por ARToolkitPlus y es utilizada por nuestro método son datos que se pueden encontrar fácilmente en otras librerías haciendo pequeños ajustes, o bien se podría añadir una capa superior a nuestro método que calculase esta información.

La posibilidad de cambiar la configuración de determinados parámetros, como el valor de *k* o *VENTANA*, favorece su exportación a otros contextos con un pequeño esfuerzo.

El método se ha abordado como solución a la oclusión de los marcadores, consiguiendo un sistema más robusto sin tener que intervenir el entorno adicionalmente. Sin embargo, en otros trabajos como [Yua06b, Sim00], donde utilizan planos para calcular la nueva pose de la cámara, también se podrían aplicar estas mismas ideas cuando el plano se encuentre parcialmente ocluido. Puesto que nuestro método no ha hecho uso de la información que aparece en el interior de los marcadores (lo cual le permite no limitarse sólo a un tipo de marcador) el marcador se ha tratado como un plano, y de ahí que se pueda aplicar en contextos en los que se utilicen planos.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Gobierno Vasco, a través de una ayuda del Programa de Formación de Investigadores del Departamento de Educación, Universidades e Investigación.

Referencias

[Azu97] R. Azuma, "A survey of augmented reality," *ACM SIGGRAPH*, 6-11 August 1995.

[ABB*01] R. Azuma, Y. Baillot, R. Behringer, Feiner, S., Julier, S. and B. MacIntyre, "Recent Advances in Augmented Reality," *IEEE Comput. Graph.*, pp. 34-47, 2001.

[DF00] J.J. Dolado and L. Fernandez, *Medición Para La Gestión En La Ingeniería Del Software.* RA-MA, 2000, pp. 296.

[Fia05] M. Fiala, "ARTag, a Fiducial Marker System Using Digital Techniques," *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision Pattern Recognition (CVPR'05)*, pp. 590-596, 2005.

[KB99] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, 1999, pp. 85.

[RBG01] Jannick P. Rolland, Yohan Baillot, and Alexei A. Goon, "A Survey of tracking technology for virtual environments," *Fundamentals of Wearable Computers and Augmented Reality. (Chapter 3) Ed. Barfield and Caudell*, 2001

[SW07] D. Schmalstieg and D. Wagner, "Experiences with Handheld Augmented Reality," *The Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 3-15, 2007.

[Sim00] G. Simon, "Markerless tracking using planar structures in the scene," *Augmented Reality, 2000. (ISAR 2000). Proceedings. IEEE and ACM International Symposium on*, pp. 120-128, 2000.

[Tat07] K. Tateno, "A Nested Marker for Augmented Reality," *Virtual Reality Conference, 2007. VR '07. IEEE*, pp. 259-262, 2007.

[Uml02] E. J. Umlauf, "ARLib: the augmented library," *Augmented Reality Toolkit, the First IEEE International Workshop*, pp. 2, 2002.

[WS07] D. Wagner and D. Schmalstieg, "ARToolkitPlus for Pose Tracking on Mobile Devices," Proceedings of 12th Computer Vision Winter Workshop (CVWW'07), February. 2007

[Yua06a] C. Yuan, "A tracking by detection approach for robust markerless tracking," *Industrial Augmented Reality Workshop; at the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Oct. 22 to Oct 25. 2006.

[Yua06b] C. Yuan, "Markerless pose tracking for augmented reality," *Advances in Visual Computing. Second International Symposium*, pp. 721-730, November 6-8. 2006.