# SISTEMA DE REALIDAD AUMENTADA PARA LA EVASIÓN DE OBSTÁCULOS UTILIZANDO EL MÉTODO DE LA TANGENTE: PARTE 1

*Israel Rivera Zárate, Jesús Pimentel Cruz, Patricia Pérez Romero*
*CIDETEC - IPN*

## 1. Introducción

La Realidad Aumentada es una técnica mediante la cual los usuarios pueden percibir la realidad superponiendo a los objetos reales modelos virtuales enriquecidos.

El observador puede trabajar y examinar objetos 3D reales mientras recibe información adicional sobre estos objetos o sobre la tarea que se está realizando. De este modo, la Realidad Aumentada permite al usuario permanecer en contacto con su entorno de trabajo, mientras su foco de atención no está en la computadora, sino en el mundo real. El papel que juega la computadora es el de asistir y mejorar las relaciones e interacciones entre las personas y el mundo real.

## 2. Herramienta ARToolkit

La Realidad Aumentada consiste en un conjunto de dispositivos que añaden información virtual a la información física ya existente. Esta es la principal diferencia con la realidad virtual, puesto que no sustituye la realidad física, sino que sobreimprime los datos informáticos al mundo real.

Este tipo de tecnología ha avanzado en forma constante, brindando a los usuarios una amplia aplicación de esta en diversas soluciones a problemas que se creían difíciles de atacar. Esta técnica de superposición de imágenes sirve como ayuda para la visualización de objetos que simulen uno real, como por ejemplo un auto; este al ser llevado a un ambiente de RA, por medio de unas gafas especiales el propio dueño verá su auto con una superposición de este mismo y así puede manipular cada parte para observar el proceso de armado de forma rápida, simple y en tiempo real.

Para esta nueva tecnología se tienen un conjunto de librerías y programas poderosos que brindan un mayor soporte al aplicar este tipo de proyecciones en tiempo real, permitiendo al usuario interactuar de forma más simple, haciendo de la RA algo fácil de emplear. Algunas herramientas de RA son: DART (Designers Augmented Reality Toolkit), osGAR y ARToolkit.

Una mejor forma de entender este funcionamiento de la superposición (Fig. 1.1) es tener el marcador físico y el objeto que se pretende superponer.

Figura 1: Muestra figura 3D y nuestro marcador físico.

Después al compilar y ejecutar el conjunto de librerías que las herramientas ya mencionadas nos proporcionan se tiene el reconocimiento del objeto físico y la superposición de la imagen (Fig. 1.2).

captionUnión de marcador y objeto 3D por medio de ARToolkit.

ARToolkit es una estructura simple para crear aplicaciones de Realidad Aumentada en tiempo real, mostrando su practicidad, facilidad y portabilidad, debido a que es una aplicación multiplataforma (Windows, Linux, Mac OS X, SGI). Logrando superponer modelos 3D en marcadores reales (basado en el algoritmo de visión por computadora), permite la captura de video contando con una biblioteca estándar de video, soporta múltiples fuentes de entrada (USB, Firewire, tarjeta de captura), soportando múltiples formatos (RGB/YUV420P, YUV), el múltiple seguimiento rápido y eficaz de la cámara además de una rutina de calibración sencilla, soporta gráficos VRML, cuenta con una rápida interpretación basada en OpenGL y además de tener una librería gráfica sencilla (basada en GLUT) [16].

ARToolkit utiliza OpenGL para la parte de representación, GLUT para las ventanas o controlador de eventos, bibliotecas de video de hardware dependiente (API) estándar en cada plataforma, en nuestro caso Win32 (Fig. 1.3).

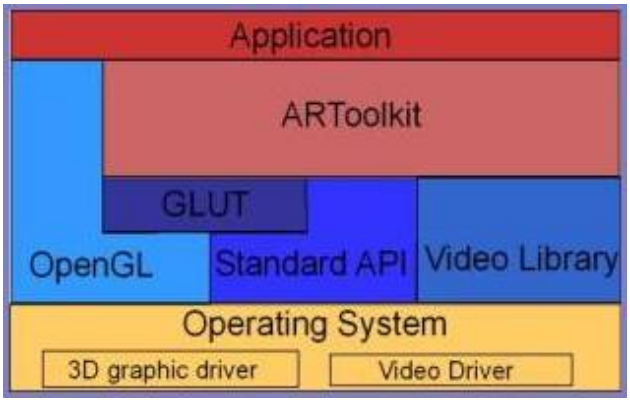


Figura 2: Arquitectura ARToolKit.

ARToolkit consta de cuatro módulos:

1. Módulo AR contiene las principales rutinas de seguimiento del marcador, la calibración y recolección de parámetros.
2. Módulo de video contiene el conjunto de rutinas para la captura de video de los marcos de entrada. Esta es una envoltura alrededor de la plataforma estándar de rutinas de captura de video SDK.
3. Módulo gsub (Fig. 1.4) es una colección de rutinas gráficas basadas en OpenGL y bibliotecas GLUT.
4. Módulo Gsub_lite (Fig. 1.5) reemplaza a gsub con rutinas más eficientes de gráficos, independiente de cualquier conjunto de herramientas.

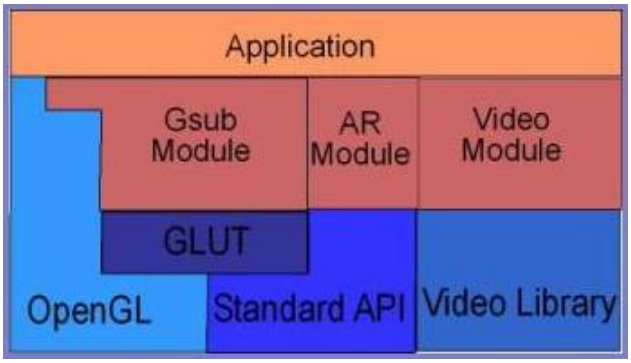


Figura 3: Estructura jerárquica de ARToolKit utilizando gsub.

Para una explicación más simple (Fig. 1.6) se muestra un diagrama a pasos simple del funcionamiento de ARToolkit:

ARToolkit maneja diversos formatos de imagen entre cada módulo como RGB24, RGB32, YUV, VUY para esto (Fig. 1.7) mostrando un diagrama de flujo de video.

## 3. Desarrollo

En esta parte se presentan a detalle el uso de la herramienta, los ejemplos para el recorrido del móvil, además de la visualización de su trayecto. Antes se mostrarán tres ejemplos: simpleTest, loadMultiple

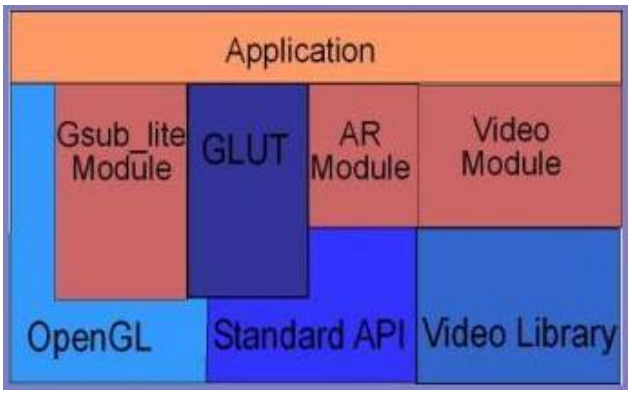


Figura 4: Estructura jerárquica de ARToolKit utilizando Gsub_Lite.

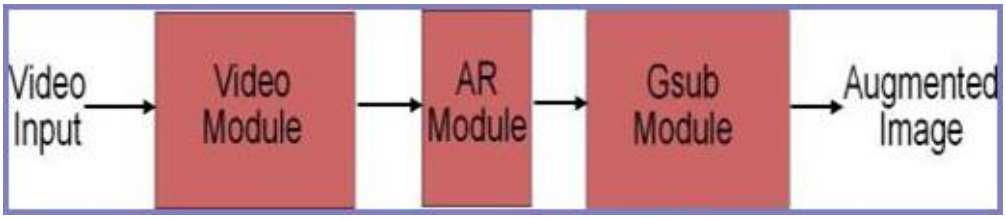


Figura 5: Diagrama de pasos.

y simpleVRML, donde se encuentran su ejecución y la parte del programa donde se manda llamar a cada uno. Cabe mencionar que estos tres fueron de vital importancia para observar el funcionamiento gráfico de este poderoso software de Realidad Aumentada ARToolkit.

Otro punto importante que se abordará es el uso de los marcadores, la colocación de estos en el área de trabajo, la superposición del móvil y edificios. Se muestran los casos sugeridos para el trayecto, también se observará el modelo del móvil a utilizar y el método que es sugerido para que se realice el trayecto.

### 3.1. simpleTest

Después de haber instalado exitosamente la herramienta y observar el funcionamiento veremos el primer ejemplo llamado SimpleTest, este se encuentra dentro de ARToolkit en la carpeta Example-s/simple, abrimos el código, lo compilamos y ejecutamos mostrando la configuración de la cámara como en Fig. 1.8.

Se validan los parámetros y si no se muestra enseguida la imagen de la cámara, es necesario ir a la carpeta bin dentro de ARToolkit y buscar SimpleTest.exe y ejecutar; se observará lo siguiente (Fig. 1.9).

Con esta aplicación nos podemos dar una idea general de cómo trabaja nuestra herramienta; además podemos observar que esta solamente permite trabajar con un solo marcador y una imagen.

En la Fig. 1.10 se detalla la parte del código que permite tomar el archivo de texto que contiene las características del marcador a reconocer.

En la Fig. 1.11 se observa un ciclo que maneja una variable 'j' que se inicializa en 0 para la comparación con marker_num (número de marcadores) y 'k' inicializada en -1 sirviendo como bandera para que el bucle continúe o se detenga.

En el círculo más grande se observa el ciclo for, que detecta el número de marcadores que se encuentran y en el círculo más chico la función draw que contiene las características principales del marcador para la colocación del gráfico.

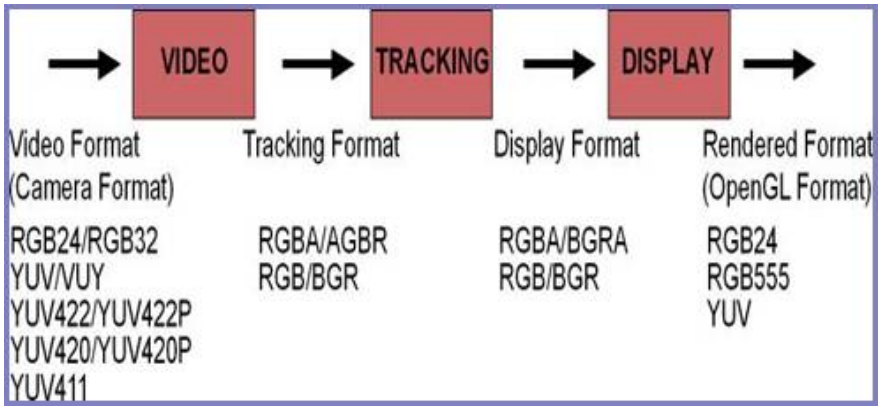


Figura 6: Diagrama de flujo de datos y formatos que acepta [17].

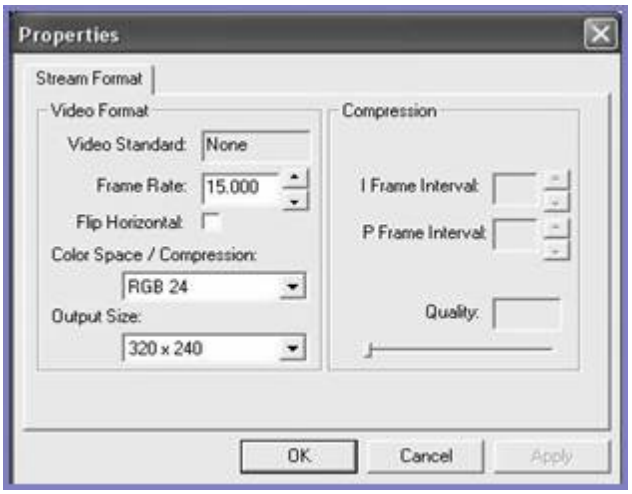


Figura 7: Configuración de la cámara.

### 3.2. loadMultiple

Para utilizar una amplia variedad de imágenes y marcadores se puede ejecutar loadMultiple, que permite tener no solo uno sino dos marcadores (ver Fig. 1.12) con su respectiva imagen.

Este ejemplo se encuentra dentro de bin al igual que el primer ejemplo. Se muestra en la Fig. 1.13 parte del programa que manda llamar el archivo que contiene los marcadores a reconocer además de los gráficos.

### 3.3. simpleVRML

Este programa es de suma importancia porque permite utilizar archivos con extensión VRML; estas son imágenes enriquecidas, ya no solo son esferas o cubos sólidos, sino que son capaces de manejar diversas texturas, formas, colores además de movimiento, como se muestra en Fig. 1.14.

En Fig. 1.15 se detalla gráficamente la parte del código en donde se mandan llamar los objetos VRML para que por medio del reconocimiento sean superpuestos en los marcadores detectados.

### 3.4. Escenario gráfico

A continuación se muestra un diagrama ejemplifico en Fig. 1.16 del modelo con la posible colocación de los marcadores y su respectiva imagen en Realidad Aumentada; este servirá de escenario para la visualización del recorrido y la evasión de los objetos por parte del móvil.

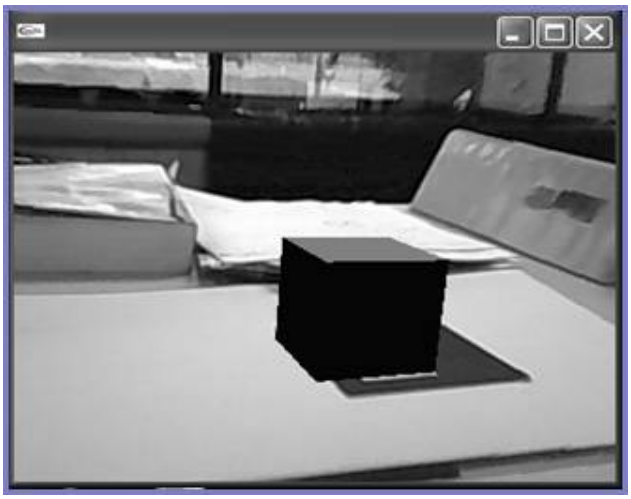

Figura 8: Funcionamiento real de simpleTest.

```
int             xsize, ysize;
int             thresh = 100;
int             count = 0;

char           *cparam_name    = "Data/camera_para.dat";
ARParam         cparam;

char           *patt_name      = "Data/patt.4x4_mio4";
int             patt_id;
double          patt_width     = 80.0;
double          patt_center[2] = {0.0, 0.0};
double          patt_trans[3][4];

static void   init(void);
static void   cleanup(void);
static void   keyEvent( unsigned char key, int x, int y);
static void   mainLoop(void);
static void   draw( void );
```

Figura 9: Archivo que sirve para saber qué marcador se está utilizando.

Cabe mencionar que para los marcadores se emplearon diseños propios así como algunos que la herramienta facilita. En la Fig. 1.17 se observa la imagen capturada por la cámara, que muestra una distribución de los marcadores sin superposición de imágenes como ejemplo.

Con esta distribución, se manda ejecutar el programa de reconocimiento de los marcadores para la superposición de las imágenes mostrando lo siguiente (Fig. 1.18).

### 3.5. Marcadores utilizados

En la Fig. 1.19 se observa el conjunto de marcadores o patts utilizados para el modelado del ambiente gráfico, así como para el móvil.

### 3.6. Móvil

En esta parte se observará el móvil empleado en el proyecto; este es un modelo virtual que cuenta con circuitos integrados, cuatro ruedas ejemplificando un todo terreno. Es importante mencionar que este puede cambiarse por cualquier otro, pero para dar más realidad al proyecto se manejó este.

```
if( arDetectMarker(dataPtr, thresh, &marker_info, &marker_num) < 0 ) {
    cleanup();
    exit(0);
}

arVideoCapNext();

/* check for object visibility */
k = -1;
for( j = 0; j < marker_num; j++ ) {
    if( patt_id == marker_info[j].id ) {
        if( k == -1 ) k = j;
        else if( marker_info[k].cf < marker_info[j].cf ) k = j;
    }
}
if( k == -1 ) {
    argSwapBuffers();
    return;
}

/* get the transformation between the marker and the real camera */
arGetTransMat(&marker_info[k], patt_center, patt_width, patt_trans);
//printf ( "% f% f% f \ n", patt_trans [0] [3], patt_trans [1] [3], patt_
draw();

argSwapBuffers();
```

Figura 10: Ciclo que muestra la cantidad de marcadores para asignar un gráfico.

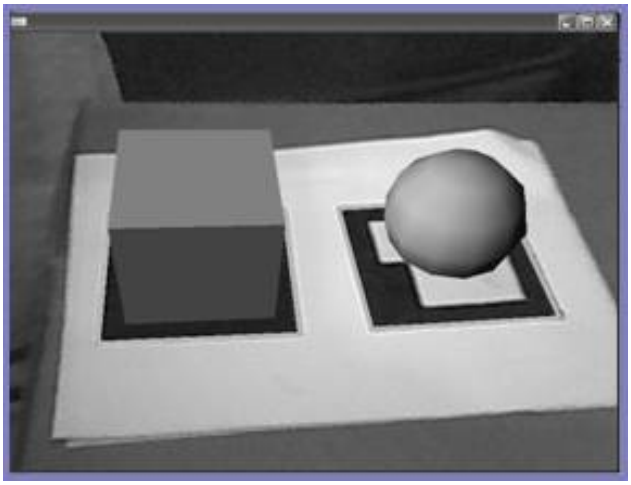

Figura 11: Ejemplo de loadMultiple.

### 3.7. Casos a resolver

Para este trabajo se tiene un móvil, el cual hace un recorrido, de tal forma que por medio de ciertas reglas evada los obstáculos (edificios) y llegue a su destino trazado.

#### 3.7.1. Caso 1: Movimiento en Diagonal

Para nuestro primer caso tendremos la distribución de los marcadores como se ve en Fig. 1.21 para que el móvil detecte los espacios ocupados y así moverse a los lugares libres de tal forma que su movimiento sea diagonal y llegue a su destino.

#### 3.7.2. Caso 2: Movimiento en línea recta

Este segundo es mucho más sencillo ya que la distribución de los marcadores cambiará de tal forma que el móvil solo detectará los obstáculos que se encuentren a su derecha e izquierda para trazar su camino en forma lineal y este logre el objetivo final, como en Fig. 1.22.

```
/* Object Data */
char            *model_name = "Data/object_data2";
ObjectData_T    *object;
int             objectnum;

int             xsize, ysize;
int             thresh = 100;
int             count = 0;

/* set up the video format globals */

#ifdef _WIN32
char            *vconf = "Data\\WDM_camera_flipV.xml";
#else
char            *vconf = "";
#endif

char           *cparam_name    = "Data/camera_para.dat";
```

Figura 12: Llamado a archivo que contiene los marcadores y gráficos.

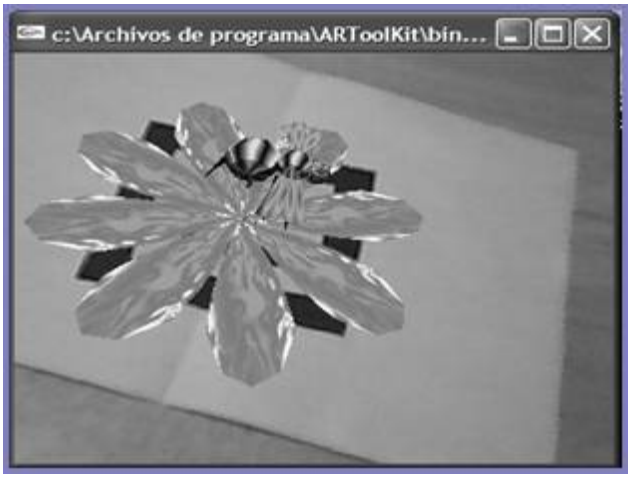


Figura 13: Este es un modelo que muestra la imagen en movimiento.

### 3.7.3. Caso 3: Sin salida

En este último el vehículo se encontrará situado en una situación muy compleja ya que no tendrá salida, pues sus caminos serán bloqueados por los marcadores (ver Fig. 1.23).

## 4. Método de la Tangente

En este apartado se expone el algoritmo basado en el método de la tangente para la evasión de obstáculos. Para el enrutamiento se traza una línea recta del centro del móvil al centro del destino llamada r; se trazan perpendiculares a la línea r cada una dirigida al centro de los obstáculos. A esta se le llamará $m_1$, $m_2$, ..., $m_n$ donde n es el número de obstáculos.

Si el tamaño de la línea $m_i$ (i = 1 hasta n) es mayor que el radio del móvil más el radio del objeto, se tendrá P como el camino directo al destino.

Si el tamaño del radio más el radio del móvil es menor, el móvil chocará con el obstáculo. Después se trazará una recta del centro del obstáculo que vaya en la misma dirección de $m_i$ pero cuya distancia sea igual al radio del móvil más el radio del obstáculo, llamándole a esta línea B. El camino partirá del móvil y terminará en el punto final de la línea B, teniendo en cuenta que el centro del móvil será el punto final de B.

```
int main(int argc, char** argv)
{
//  int i;
    char glutGamemode[32];
    const char *cparam_name = "Data/camera_para.dat";
#ifdef _WIN32
    char            *vconf = "Data\\WDM_camera_flipV.xml";
#else
    char            *vconf = "";
#endif
    char objectDataFilename[] = "Data/object_data_vrml";

    // -----------------------------------------------------------------------
    // Library inits.
    //

    glutInit(&argc, argv);

    // -----------------------------------------------------------------------
    // Hardware setup.
```

Figura 14: Llamado al archivo que contiene las figuras VRML.

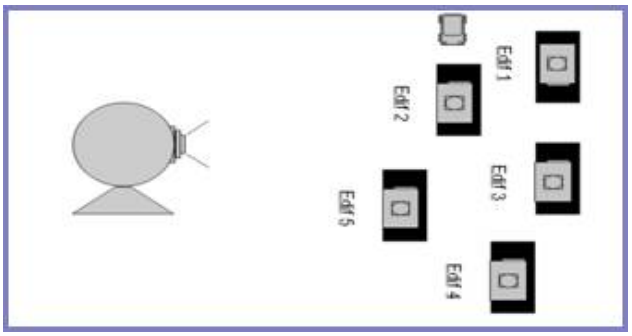


Figura 15: Esquema de nuestro modelo.

### 4.1. Implementación y visualización

En este apartado mostramos la interpretación del código empleado para mostrar los resultados en la visualización. Para esto se generaron tres condicionamientos para el recorrido del móvil; se tendrá una matriz de 16 la cual se definirá de la siguiente manera:

```
int Matriz[16] = – ’v’, ’v’, ’v’, ’v’, ’v’, ’e’, ’v’, ’e’ ″;
```

En donde:

- v = lugares vacíos
- e = refiriéndose a los edificios

Para hacer el recorrido se generaron las condiciones siguientes por medio de estructura if-else:

```
m = 0;
if (Matriz[m+5] == ’v’)
printf(”el móvil avanza”);
else if (Matriz[m+7] == ’v’)
printf(”el móvil avanza”);
else if (Matriz[m+6] == ’v’)
printf(”el móvil avanza”);
else
```


Figura 16: Captura de marcadores sin superposición.


Figura 17: Captura de marcadores con superposición de imagen.

```
printf("el coche se queda");
```

La letra m se ubica en la primera posición (ver Fig. 1.24) del arreglo refiriéndose al móvil:

Se puede observar que el camino de este móvil es en línea vertical, ya que al sumar la posición m+6 encontramos que ese lugar está vacío y por ende es el siguiente espacio donde colocar nuestro móvil. Al mirar este gráfico notamos que al lado derecho de nuestro esquema contamos con más espacios libres, pero una de las soluciones específicas es que el móvil se desplace diagonalmente.

Para la visualización del método se integró al código del ejemplo simpleVRML1 para mostrar con claridad la superposición del móvil y su movimiento en nuestros 3 ejemplos.

Tenemos un ciclo for que es el que permite definir el número de objetos que se tienen declarados en el archivo .object_data_vrml; dentro de este se encuentra un if con dos condiciones aquí mostradas:

```
if ((gObjectData[i].visible != 0) && (gObjectData[i].vrml˙id ¿= 0))
```

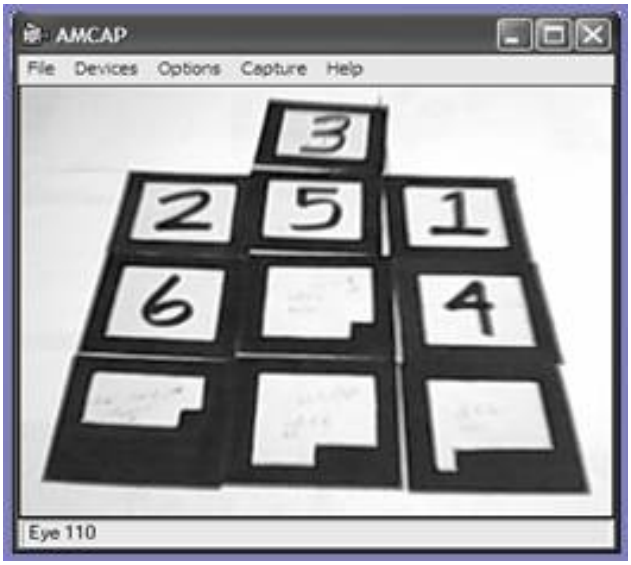


Figura 18: Marcadores.

Figura 19: Móvil.

La primera hace referencia a la detección del marcador y la segunda a la cantidad de los objetos por medio de su id; éstas al validarse permiten la detección del marcador y la superposición de la imagen.

La función arglCameraViewRH() dentro de nuestro programa se vuelve una parte esencial, porque es la que permite el avance del móvil: al bloquear el marcador donde se superpone la imagen del móvil, esta función quita el objeto virtual, apareciendo en el siguiente lugar vacío y así sucesivamente hasta llegar al destino (ver Figuras 1.25 a la 1.27).

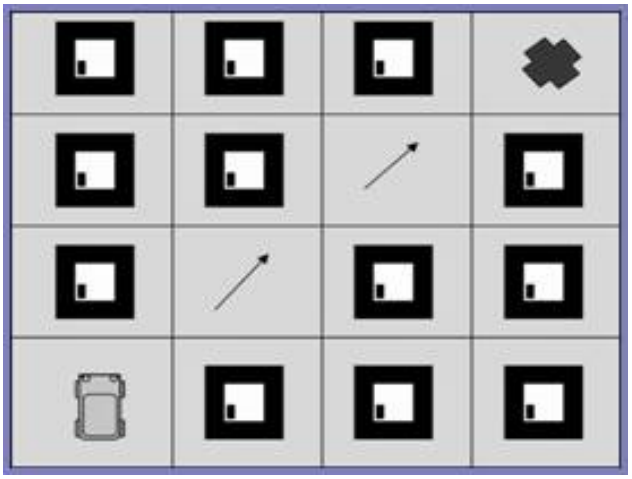

Figura 20: Caso 1: camino en diagonal.

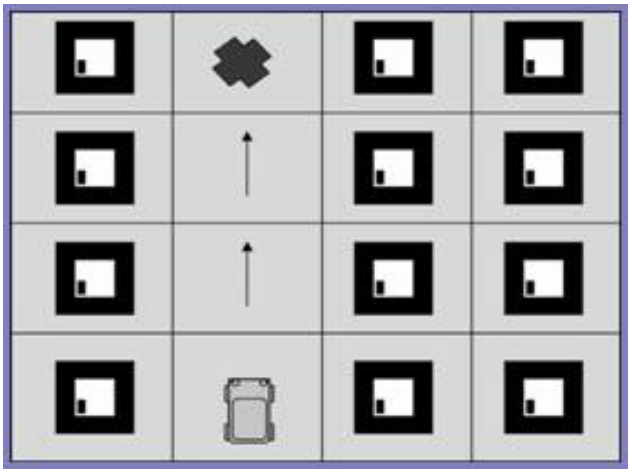

Figura 21: Caso 2: camino lineal.

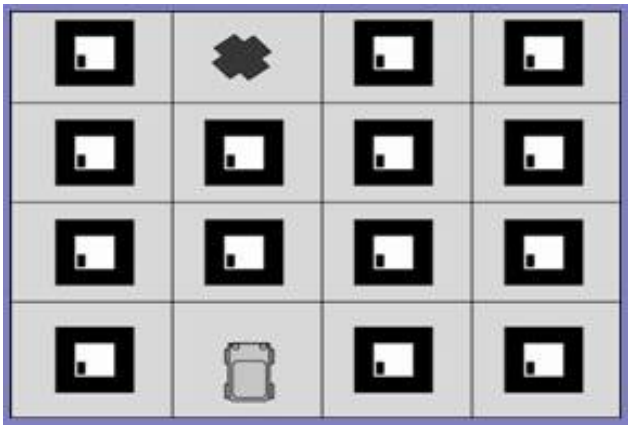

Figura 22: Caso 3: sin salida.

M V V V E V

Figura 23: Muestra gráfica de la comparación.


Figura 24: Imagen en nuestro primer marcador.


Figura 25: Imagen en nuestro segundo marcador.