SEGUIMIENTO EN TIEMPO REAL DE OBJETOS SOBRE SECUENCIA DE IMÁGENES EMPLEANDO DISPOSITIVOS DE LÓGICA PROGRAMABLE

MARLON STEEPHEN NARVÁEZ RODRIGUEZ
MIGUEL ANDRÉS SARRIA FERNÁNDEZ

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN
PROGRAMA DE INGENIERÍA ELECTRÓNICA
PEREIRA, COLOMBIA
2010

SEGUIMIENTO EN TIEMPO REAL DE OBJETOS SOBRE SECUENCIA DE IMÁGENES EMPLEANDO DISPOSITIVOS DE LÓGICA PROGRAMABLE

MARLON STEEPHEN NARVÁEZ RODRIGUEZ
MIGUEL ANDRÉS SARRIA FERNÁNDEZ

Proyecto de grado para optar al título de
Ingeniero Electrónico

Director
MSc. Paula Jimena Ramos Giraldo

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN
PROGRAMA DE INGENIERÍA ELECTRÓNICA
PEREIRA, COLOMBIA
2010

Nota de aceptación:

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

Pereira, Octubre de 2010

Damos gracias a Dios por darnos la vida y permitirnos a través de su luz y guía y de nuestra inteligencia y entendimiento poder terminar este trabajo; a nuestros padres y a toda nuestra familia por creer en nosotros y por estar siempre a nuestro lado dandonos aliento, apoyo y fuerza.

AGRADECIMIENTOS

Agradecemos muy especialmente a nuestra directora, la ingeniera Paula Jimena Ramos Giraldo y al ingeniero Julián David Echeverry para que a través de sus conocimientos sigan con la formación de todos los estudiantes de Ingeniería Electrónica.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

En este proyecto de grado se plantea una metodología para determinar, en tiempo real, la ubicación y orientación de un objeto en movimiento, sobre una superficie plana, durante un intervalo de tiempo. Se utilizará para ello una cámara y una tarjeta de video para la adquisición de las imágenes, el procesamiento se realizará empleando dispositivos de lógica programable con visualización en pantalla. Los principales problemas al realizar este tipo de aplicaciones es el alto costo computacional y económico de los elementos que se utilizan para tal fin, como también de la velocidad de respuesta del sistema a implementar. Se propone una solución de compromiso entre la precisión en la determinación de la información obtenida, el costo computacional y la velocidad de la técnica. Con la técnica a desarrollar se pretende en primer lugar realizar la adquisión de la imagen, el preprocesamiento y procesamiento de la imagen, la identificación del objeto para luego abordar el problema de seguimiento del mismo y finalmente su visualización. En la parte de identificación se utiliza segmentación por color y una técnica para el seguimiento del objeto.

INTRODUCCIÓN

La estimación de la ubicación y orientación de un objeto en movimiento a partir de la percepción del entorno constituye uno de los problemas fundamentales en cualquier sistema de visión por computador. Las técnicas para determinar la posición de un objeto y estimar su movimiento se han aplicado y se están aplicando con éxito en campos tan diversos como el reconocimiento de objetos, navegación de sistemas autónomos en robótica móvil, seguimiento en sistemas de video-vigilancia y de seguridad, monitoreo de tráfico, navegación autónomas de robots y vehículos, estimación de movimiento en robótica de manipuladores, análisis de imágenes médicas, biomecánica, compresión y descompresión de imágenes, animación por computadora (captura de movimiento), ergonomía, video-conferencia, desarrollo de modelos de representación autónomos o activos, que son empleados principalmente en el tratamiento de movimiento no-rígido y en sistemas de seguimiento celular en biología para la determinación de dinámica celular.

Por lo general, cuando se requiere estimar posición y determinar movimiento a partir de la percepción del entorno se emplean cámaras, sonares, sensores láser, etc. En los mecanismos de sensorización basados en cámaras se requiere seleccionar un conjunto determinado de características, ya sean marcas naturales, puntos de interés, información de color o cualquier otro tipo de información basada en apariencia y que esté presente en el entorno, de modo que sea proyectada y registrada en imágenes bidimensionales.

Con este trabajo de grado buscamos a través de una técnica, una cámara y una tarjeta de adquisión de video realizar la adquisición de la imagen, el preprocesamiento seguido del procesamiento de la imagen y la identificación del objeto para luego abordar el problema de seguimiento del mismo y finalmente su visualización. En la parte de identificación se utiliza segmentación por color y una técnica para el seguimiento del objeto.

1. PRELIMINARES

1.1. PLANTEAMIENTO DEL PROBLEMA

El seguimiento de objetos es uno de los problemas críticos dentro de los procesos de visión artificial donde la exactitud y el tiempo de respuesta son de gran importancia dentro de la respuesta esperada, por ejemplo en los sistemas de vigilancia, en la robótica y en la industria cuando se necesita hacer seguimiento a un proceso.

Los sistemas de visión por computador se basaron inicialmente en imágenes estáticas, sin embargo el mundo es dinámico, por tanto el esfuerzo invertido en mejorar la habilidad de seguir objetos en movimiento queda justificado. Actualmente existen diversas aplicaciones, siendo la robótica y la medicina dos de las más importantes [NLO06].

En la mayoría de las tareas de localización de objetos mediante visión artificial los objetos son estáticos. Existen diversos factores que determinan la dificultad en la solución del problema de seguimiento de objetos y un aspecto fundamental al desarrollar estructuras de representación de objetos es su forma. Los objetos pueden estar constituidos por puntos, líneas, aristas, curvas o poseer una morfología libre, incrementando el grado de dificultad al identificar el objeto [CDS09]. También se debe tener en cuenta los cambios morfológicos del objeto en un momento dado, los cambios fotométricos como la modificación en parámetros como la iluminación, la presencia de sombras, los desplazamientos y los giros. Dentro del análisis o reconocimiento de objetos existen dos vertientes principales: el análisis de objetos en movimiento basado en el reconocimiento y el reconocimiento basado en el movimiento.
En la actualidad, el seguimiento de objetos, busca como objetivo fundamental disponer de resultados con la máxima exactitud posible enmarcados en el desarrollo del diseño del sistema. Es por ello que se ha recurrido a la lógica reconfigurable que es por el momento la forma mas apropiada de abordar las aplicaciones de seguimiento con el procesamiento en tiempo real de una secuencia de imágenes justificando el desarrollo del trabajo de grado en el avance del estado de la técnica para la Universidad Tecnológica de Pereira y para la región, teniendo en cuenta que se busca solucionar el problema de seguimiento de objetos con dispositivos de lógica programable.

1.2. JUSTIFICACIÓN

El seguimiento de objetos por medio de visión artificial proporciona al sistema la información básica del ambiente por medio de imágenes. El proceso de captura del sistema visual a través de una cámara recibe a su entrada una secuencia de imágenes dinámica tomada del mundo real cada una de ellas en un instante de tiempo diferente.

Con el procesamiento de imágenes se hace el proceso de seguimiento el cual busca conocer el cambio de la localización u orientación de un punto en una secuencia de imágenes a partir de dos o más imágenes. Los cambios entre cada imagen pueden ser debidos al movimiento de desplazamiento y de rotación de los objetos y a transiciones de los niveles de iluminación y sombras siendo estos los más difíciles de controlar generando pérdida de la información en las características visuales. Por tanto el problema genérico en un sistema de visión artificial, puede clasificarse según sean los objetos que se muevan y del ambiente en donde se encuentren.
Se busca con este trabajo abordar el problema de seguimiento de objetos cuando están en movimiento utilizando una secuencia de imágenes, haciendo uso de dispositivos de lógica programable y algoritmos de procesamiento de imágenes y video. Al implementar sistemas de seguimiento de objetos a través de un dispositivo de lógica programable se tiene un costo menor que al hacerlo con un computador de alto desempeño. Además de poder contar también con una alta velocidad de procesamiento y un consumo de potencia mínimo [LO05].
Finalmente también debemos tener en cuenta que los diferentes aplicativos o sistemas desarrollados para el seguimiento de objetos a través de otros dispositivos puedan ser desarrollados a través de lógica reconfigurable para mejorar su rendimiento.

1.3. OBJETIVOS

Los objetivos de la investigación se conforman de un solo objetivo general, que se logrará una vez alcanzados los tres objetivos específicos que se exponen a continuación.

1.3.1. OBJETIVO GENERAL

Desarrollar e implementar el seguimiento en tiempo real de objetos sobre secuencia de imágenes utilizando dispositivos de lógica programable en condiciones de laboratorio.

1.3.2. OBJETIVOS ESPECIFICOS

- Adquirir la secuencia de imágenes en el dispositivo de lógica programable.
- Desarrollar el algoritmo para seguimiento de objetos.
- Validar el algoritmo desarrollado con visualización en pantalla de las imágenes procesadas del seguimiento del objeto.

1.4. DISEÑO METODOLÓGICO

1.4.1. HIPÓTESIS

El algoritmo de seguimiento de objetos implementado en la FPGA tiene una eficiencia 2 veces mayor a la eficiencia del algoritmo implementado en

un PC (Personal Computer).

1.4.2. VARIABLES En la tabla 1.1 se muestran las variables:

Variable	**Indicador**	**Valoración**
Iluminación	Luxómetro	Lux
Posición del Centroide del Objeto	Píxel Central	Coordenadas de los píxeles
Elasticidad	Eficacia de Seguimiento	Permanencia del Valor de las Coordenadas
Velocidad de respuesta de la implementación del sistema	Tiempo de Respuesta	Seguimiento del objeto, sin perder la concentridad de la imagen de salida
Porcentaje de Reconocimiento del Objeto	Píxeles	Número de Píxeles

Tabla 1.1: Variables

1.4.3. POBLACIÓN Y MUESTRA Uno de los componentes de los que se depende para lograr la representación de los objetos es su forma. La representación se puede llevar a cabo con puntos, líneas, curvas o figuras básicas, y aumenta su complejidad cuanto más complicada sea su morfología. La población son formas geométricas definidas como círculos en la cual se probaran dos tipos diferentes de color definido para el seguimiento, para así tener una muestra de dos objetos a reconocer. Los cambios sobre el objeto también influyen en el proceso. Variaciones producidas por movimientos de desplazamiento y de rotación llevan a cambios aparentes de tamaño o forma. Transiciones de los niveles lumínicos, las sombras y los tonos de color generan pérdida de la información en las características visuales o incertidumbre con modelos corruptos que se tienen del ambiente y los objetos. Las muestras (video) se tomaran en un escenario, en el cual se tendrán en cuenta estos factores. El lugar en el cual se realizarán las pruebas debe poseer una iluminación en lo posible controlada.

2. MARCO DE REFERENCIA

2.1. ESTADO DEL ARTE

2.1.1. APLICACIONES DE LOS DISPOSITIVOS DE LÓGICA PROGRAMABLE

PROGRAMABLE Los dispositivos de lógica programable son utilizados en diversos campos. En el mundo se han realizado muchos trabajos acerca del seguimiento de objetos los cuales han sido utilizados en diversas aplicaciones o sistemas pues una de las principales ventajas de la implementación con dispositivos de lógica programable es su relación de un buen desempeño, bajo costo y además de un bajo consumo de potencia [ZAC05]. A continuación se describen algunas de estas aplicaciones:

- La implementación del protocolo de enrutamiento OSPF (primero la ruta libre más corta) con un dispositivo lógico programable está enmarcado dentro de la línea de investigación en telecomunicaciones del grupo SIRIUS de la Universidad Tecnológica de Pereira [LM06]. Se pretende aprovechar al máximo la tecnología de los PLD (Dispositivos Lógicos Programables), en particular la tarjeta FPGA Spartan-3, para diseñar una función lógica que seleccione la mejor ruta para un paquete de datos según el estado de la red.

- En cuanto al seguimiento de imágenes se realizó un trabajo de registros de transeuntes en tiempo real utilizando un sistema de visión artificial sobre un ambiente controlado [Lop06]. Con este sistema libre de parámetros y que trabaja en o fuera de línea se busca contar y localizar transeuntes en un ambiente controlado. Para realizar esta tarea se hizo uso de la segmentación aplicando la técnica de umbralización la cual se adapta muy fácilmente a cambios suaves de iluminación. Después se hace un seguimiento de las coordenadas del centroide para cada transeúnte por medio de la predicción de movimiento que identifica durante todo el recorrido a cada uno de los objetos de interés con la misma etiqueta en la secuencia de imágenes. A partir de este seguimiento se identifica la dirección en que se desplazan los elementos de interés (de arriba a abajo y viceversa), permitiendo así obtener un registro general del número de transeúntes que han circulado por el área de conteo.

- Con estos dispositivos se han realizado implementaciones como compresión de video y descompresión de video [YJJK08], procesamiento de imágenes en la secuencia de video (zoom), espacios de color, vistas panorámicas, identificación de bordes, identificación de rostros, reconocimiento de gestos, seguimiento de objetos [YNS08], codificación y decodificación de video, y aplicaciones en

multimedia.

- En varias investigaciones se ha trabajado la optimización de uso de memoria para sistemas de video [BTO05].

- Con la utilización de un dispositivo de lógica programable también podemos realizar la adquisición de imágenes, procesarlas y dar un resultado [YLZH91], ya que estos dispositivos poseen altas velocidades de procesamiento de datos y el desarrollo de algoritmos en concurrencia.

- Los sistemas de reconstrucción en 3D han desarrollado algoritmos de gran velocidad. El principal factor para su evolución, se presenta en el rápido desarrollo tecnológico en procesadores y tarjetas procesadoras de imágenes. En velocidad de procesamiento se hace uso de diseños para la estimación del movimiento y se ha utilizado la implementación de arquitecturas VLSI (Very Large Scale Integration) en el desarrollo eficiente de tarjetas programables para el procesamiento de imágenes como la FPGA (Field Programmable Gate Arrays), la cual es la más expandida en los sistemas de alta velocidad y en tiempo real. En la FPGA, todos los sistemas han sido desarrollados y evaluados para ser dedicados exclusivamente al procesamiento de imágenes [Tir07].

- Los sistemas reconfigurables dinámicamente tienen la capacidad de modificar su estructura de hardware en tiempo de ejecución, no para fines de actualización del sistema, sino para optimizar recursos [FRVRV08]. Se puede hacer una analogía entre la reconfiguración dinámica y el funcionamiento de los programas orientados a objetos. Cuando un programa orientado a objetos se está ejecutando, puede solicitar memoria al sistema para crear un objeto en dicha memoria, y entonces hacer uso del objeto. Cuando el objeto ya no se ocupa, puede ser destruido y el espacio de memoria que ocupaba puede ser reutilizado para crear algún otro objeto que se requiera.

- Un fuerte trabajo de investigación se está llevando a cabo en los siguientes aspectos: Arquitectura de Dispositivos de Lógica Programable, Sistemas de Control y de Video-Vigilancia internos y externos, Dispositivos de Domótica en los diagnósticos médicos, Aceleradores por Hardware, Sistemas en un Chip (SoC), Sistemas RTR y Herramientas de Desarrollo y Verificación para Sistemas Digitales [FRVRV08].
  En particular, los algoritmos de procesamiento de video usando FPGA para realizar el reconocimiento de imagen y detección de movimiento hace posible la automatización de las aplicaciones de control como son ejemplos de conteo y

reconocimiento de las personas que entran o salen de un edificio, reconocimiento de placas de vehículos y de la posterior identificación de las personas usando un algoritmo de OCR (Optical Character Recognition) en la misma placa, esto se puede hacer de manera totalmente autónoma, sin la presencia de una persona en el lugar [Oli07].

Para el seguimiento de objetos existen diversas técnicas como son el Flujo Óptico [KBR09], los Filtros Mached, la Segmentación por Color, los Filtros de Partículas [JUCJ07], las Redes Neuronales [MKK02] y la Correlación, entre otros.
A continuación haremos enfásis de los trabajos realizados sobre seguimiento de objetos con algunas de las técnicas mencionadas anteriormente:

Redes Neuronales Artificiales (ANN): Los trabajos con una red neuronal no resuelven por completo la detección y el movimiento de objetos pues el resultado binario de la imagen de la red ideal tal como se utiliza en las simulaciones basadas en software incluye errores con respecto a la imagen de muestra de aprendizaje. Los errores se pueden eliminar con las operaciones morfológicas como la erosión y la dilatación los cuales se llevan a cabo muy rápidamente, ya que son aplicados a las imágenes binarias como se puede observar en la figura 2.1 [MKK02].

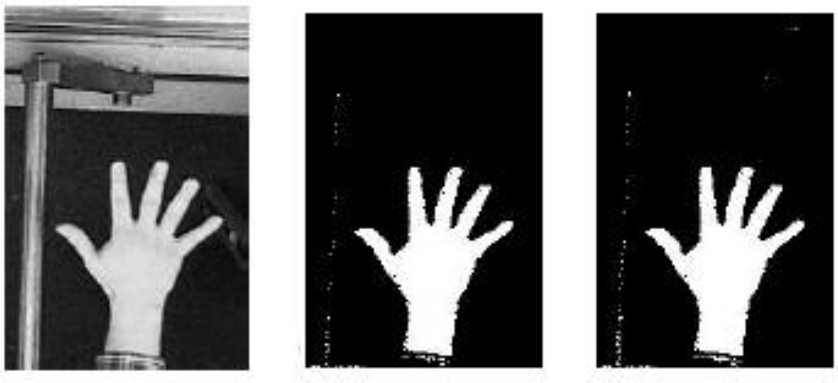


Figura 2.1: Ejemplo de una imagen con una Red Neuronal

Esta técnica fue utilizada en el trabajo **'FPGA Implementation of a Neural Network for a Real-Time Hand Tracking System'**. Los resultados de la Redes Neuronales Artificiales (ANN) aplicados son ligeramente inferiores a las redes originales. Las modificaciones aplicadas después de la fase de aprendizaje no afectan el desempeño de las redes. El resultado presentado en la figura 2.1 muestra una mano bien detectada y algunos píxeles clasificados erróneamente. Sin embargo, las partes erróneamente detectadas son del tamaño de 1 a 4 píxeles de diámetro. Mediante esta clasificación todos los errores fuera de la mano se pueden eliminar sin afectar el resultado de detección de la mano.

Los resultados proporcionados por la realización de la FPGA de la red neuronal artificial y las ofrecidas por la solución original basada en MATLAB sin ningún tipo de modificación aplicada, difieren en menos del 0,02 % para una imagen de tamaño 288 x 384 píxeles que se utiliza en la figura. Las redes son importantes en sistemas de detección que pueden ser utilizados como un componente en una instalación de seguridad basada en video. La prueba no sólo es importante durante el diseño, sino también durante el funcionamiento normal del dispositivo para asegurar la detección exacta y el seguimiento para toda la vida útil del sistema.

Filtro de Partículas: El porcentaje de utilización de las características del dispositivo FPGA para el circuito del algoritmo de seguimiento de objetos utilizando el filtro de partículas es bastante bajo. Esta técnica fue utilizada en el trabajo **'Multiple Objects Tracking Circuit using Particle Filters with Multiple Features'**. La figura 2.2 muestra un sistema de visión usando una FPGA XC2V8000-4CFF1152 [JUCJ07]. Esta FPGA puede adquirir imágenes de una cámara y mostrar tanto las imágenes en bruto y las procesadas y los datos de la trayectoria de los objetos generados por el filtro de partículas al monitor o LCD directamente presentando una alta velocidad de procesamiento de tramas y una baja latencia. Cuando el sistema de visión con el circuito propuesto de seguimiento de objetos a través de filtros de partículas se aplica a una cámara RS-170, produce imágenes que consisten de 640 x 480 píxeles, el cual puede procesar las imágenes a velocidades de hasta 56,38 fps (fotogramas por segundo) para realizar seguimiento de objetos en movimiento. El rendimiento de este sistema se mide mediante la evaluación de la duración máxima de retraso medio.
La figura 2.2 muestra el resultado obtenido en el seguimiento de objetos en movimiento en una secuencia de video mediante el seguimiento de varios objetos del sistema.

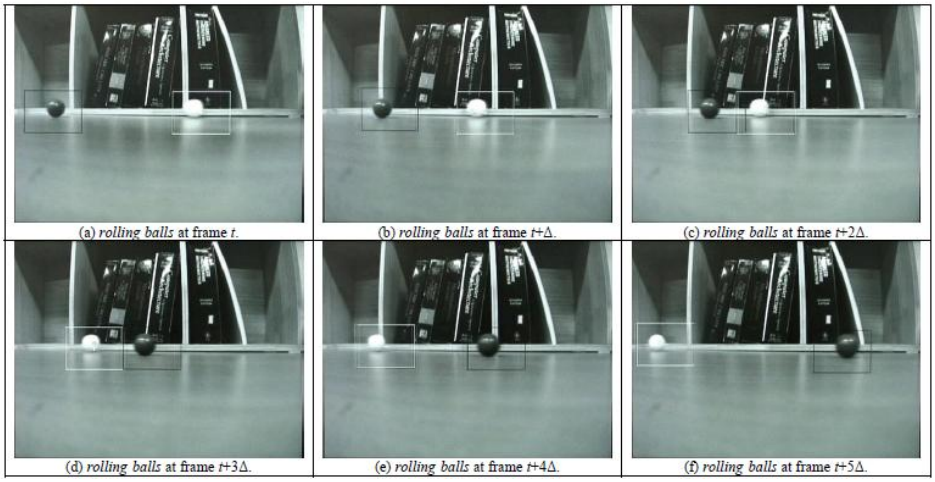


Figura 2.2: Seguimiento de Objetos con el Filtro de Partículas

En la figura 2.2 el cuadrado blanco con la cruz blanca en el centro representa la trayectoria del objeto de destino. En el seguimiento de objetos se usa solamente

la característica IFD (Inter-Frame Difference) para el seguimiento de un objeto en movimiento. En el seguimiento de múltiples objetos se utilizan múltiples características como la IFD y el nivel de grises. En la figura 2.2 el cuadrado negro con la cruz negra presenta la trayectoria de destino del objeto que especifica el nivel de grises incluido en el rango [0, 127].
El cuadrado blanco con la cruz blanca representa la trayectoria del objeto de destino que tiene el nivel de grises incluido en el rango [128, 255]. Incluso aunque hay una oclusión, el objeto especifico de seguimiento no se ve afectado por otro en la figura. Un conjunto de partículas se compone de 64 partículas. La trayectoria del destino del objeto se calcula por la media de la posición de partículas en un conjunto de partículas.

Flujo Óptico: El Flujo Óptico es importante en la estimación y descripción del movimiento y por este motivo se utiliza en la detección, segmentación y seguimiento de objetos móviles a partir de un conjunto de imágenes. Esta técnica fue utilizada en el trabajo **'Detección de Objetos Móviles en una Escena Utilizando Flujo Óptico'** como se muestra en la figura 2.3. El movimiento de objetos implica movimiento de patrones de intensidad en el plano de la imagen por ejemplo en el flujo vehicular. Existen 4 técnicas para estimar el flujo óptico: las basadas en gradientes espacio-temporales, las basadas en comparación de regiones, las basadas en fase y las basadas en energía [DMS09].

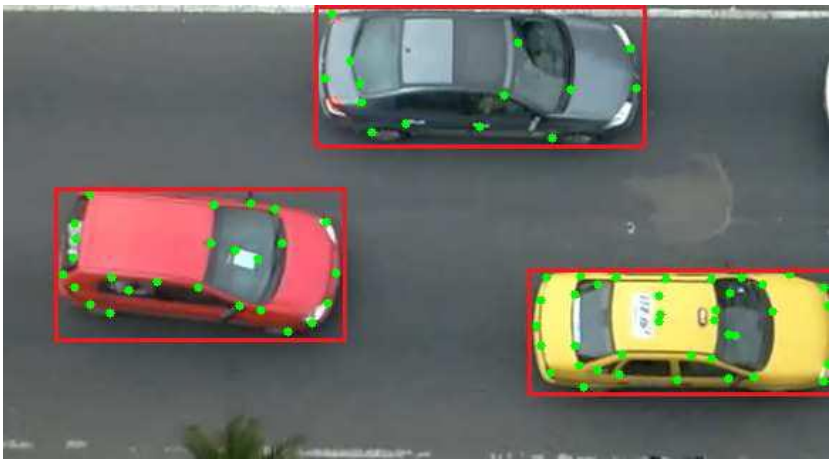


Figura 2.3: Seguimiento de Objetos con el Flujo Óptico

Siempre se parte de que los niveles de gris permanecen constantes ante movimientos espaciales en un tiempo dado. También se puede asumir que el flujo óptico es constante en una determinada región mediante diferentes métodos como por ejemplo el de Lucas y Kanade. Hay ciertos factores que pueden provocar errores en la estimación como son la variación temporal de los niveles de gris sobre la región, desplazamientos grandes de la región entre las imágenes consecutivas e incoherencia de movimiento, pero se pueden controlar seleccionando un tamaño apropiado de la región. La magnitud y dirección del

flujo óptico es encontrada en función del desplazamiento horizontal y vertical de los puntos seleccionados. Los puntos obtenidos son agrupados por distancia, velocidad e información a priori del objeto de interés, utilizando un algoritmo de clustering, el cual es muy eficiente en términos de costo computacional.

2.1.2. PROCESAMIENTO DIGITAL DE IMÁGENES La vista es nuestro sentido más avanzado, y no es sorprendente que las imágenes jueguen el papel más importante en la percepción humana. Aunque los seres humanos estemos limitados a la banda visible del espectro electromagnético (EM), las máquinas pueden percibir casi el espectro completo, desde los rayos gamma, a las ondas de radio. Las máquinas también pueden procesar imágenes generadas por fuentes que los humanos no asociamos con imágenes, como es el caso del ultrasonido, la microscopía de electrones, etc. Se explicarán algunos conceptos básicos de las imágenes digitales:
El píxel: Las imágenes digitales están formadas por un arreglo o matriz de puntos, denominados píxeles, los cuales son la unidad fundamental en las imágenes digitales [YVV98].
Dimensiones: La composición del número total de columnas y filas de píxeles de la matriz de una imagen determinan las dimensiones, la cual se denota de la siguiente manera:

$$M * N \text{ píxeles}$$

donde M es el número de filas y N es el número de columnas de la imagen digital.

Tipos de imágenes: Las imágenes se pueden clasificar en 3 tipos: Imágenes Binarias, Imágenes en Escala de Grises e Imágenes a Color. La distinción entre estas lo hace posible la cantidad de colores que pueden ser representados por cada píxel, obteniendo una imagen binaria cuando a cada uno de sus píxeles se le puede asignar solamente 2 colores: blanco o negro. Para las imágenes en escala de grises, cada píxel toma un valor dentro de una gama de grises que van desde el blanco hasta el negro. La cantidad de valores puede variar de acuerdo al número de bits asignado para el almacenamiento del valor del píxel. Para las imágenes a color, cada píxel varia en una gama más amplia de tonos, producto de la combinación de ciertos valores. Cada píxel de una imagen a color es definido como un vector, donde cada elemento del vector indica el aporte de una tonalidad. Por ejemplo, en el modelo RGB los elementos del vector corresponden a los aportes de las tonalidades de rojo, verde y azul en la posición del píxel. Estas imágenes transmiten mayor información, sin embargo, ocupan mayor espacio de memoria para ser almacenadas en medios digitales.
Hay 3 clases de operaciones en el procesamiento de las imágenes digitales [YVV98]:

Basadas en puntos de la imagen: En esta técnica, cada píxel de la imagen de salida es producto de una operación realizada sobre el píxel respectivo de la imagen de entrada. Para la determinación de cada uno de los píxeles de la imagen procesada, solo es necesario conocer el valor del píxel original y efectuar la operación deseada. Un ejemplo

es la obtención del negativo de una imagen cuya expresión es la siguiente para imágenes con grises de 256 valores:

$Pout = 255$ - Pin

donde Pin y $Pout$ son los píxeles de entrada y de salida, respectivamente.

Basadas en la imagen global: La operación realizada para la determinación de cada píxel de salida necesitará el valor de todos los píxeles de la imagen. Cada píxel de la imagen procesada es función de todos los valores de los píxeles de la imagen original.

Basadas en región de la imagen: En esta técnica se necesita conocer el valor de los puntos de la región alrededor del píxel en estudio, para luego aplicar la operación determinada y obtener el valor píxel de salida. Cada punto de la imagen de salida es función de su valor en la imagen original y de la región circundante a él. A estos píxeles se les denomina vecinos y el tamaño de la región (ventana) que los contiene varia dependiendo de la operación que se desea llevar a cabo. En aplicaciones de video con dispositivos de lógica programable, es fundamental el procesamiento de las imágenes, técnicas que se aplican a las imágenes digitales para mejorar la calidad o facilitar la búsqueda de información. Hay diversas técnicas como la reducción de ruido, mejora del contraste, segmentación, cambios en los espacios de colores, filtrado de la señal [YJJK08], umbralización, realce [RA07], muestreo y cuantificación, entre otros. El proceso de adquisición de la imagen, el preprocesamiento, la extracción (segmentación) de carácteres individuales, la descripción de los carácteres de una forma aceptable para el procesamiento computacional, y el reconocimiento de cada carácter individual entra en el campo de lo que se denomina Procesamiento Digital de Imágenes.

2.1.3. PROCESAMIENTO DIGITAL DE VIDEO Dentro del análisis del movimiento se encuentran muchas aplicaciones como la navegación y el seguimiento y también en la consecución de información sobre objetos estáticos y en movimiento de una escena. La información para resolver un problema relacionado con el movimiento son una secuencia de imágenes. El análisis de movimiento se utiliza también para resolver problemas de restauración, compresión, obtención de imágenes y en secuencia de imágenes de alta resolución a partir de una de baja resolución.
En el procesamiento visual es muy importante la dimensión temporal ya que con esta se puede entender la estructura y el movimiento 3-D del movimiento aparente de los objetos en el plano de la imagen y también porque se utiliza el movimiento visual para poder extraer características de 3-D [Sor07]. Una solución a problemas de movimiento es a través de la estimación del campo de movimiento. Hay 3 clases de técnicas para la estimación del campo de movimiento en el procesamiento digital de video:

Basadas en la ecuación del flujo óptico (técnicas diferenciales): Lucas y Kanade suponen que el vector de movimiento no cambia en el bloque de píxeles que se está estudiando

y estima el vector de movimiento en un bloque para cada píxel x,y.

Basadas en el desplazamiento de bloques: Método usado principalmente para la compensación del movimiento en casi la mayoría de los estándares de compresión de video y también en la aplicación del filtrado y restauración de secuencias de video. La forma más sencilla de movimiento es considerar que un bloque de tamaño L x L en una imagen m centrado en el punto (x0,y0) coincide con un bloque del mismo tamaño en la imagen m+1.

Basadas en el acoplamiento de rasgos entre imágenes: Las imágenes tienen una resolución dada y para realizar los cálculos de los desplazamientos se debe utilizar esta resolución, aplicando la interpolación de la imagen. También se hace uso de la homografía que relaciona dos imágenes en perspectiva en donde puntos o líneas de un plano de la escena en una imagen están relacionados con puntos o líneas de la otra imagen. Esta relación es válida si la escena es plana o si el desplazamiento con el recurso que capta la imagen es pequeño.

2.2. MARCO TEÓRICO

Una imagen puede ser definida como una función bidimensional, $f(x, y)$, donde x y y son coordenadas espaciales (en un plano), y f en cualquier par de coordenadas es la intensidad o nivel de gris de la imagen en esa coordenada.

Cuando x, y , y los valores de f son todas cantidades finitas, discretas, decimos que la imagen es una imagen digital. Una imagen digital se compone de un número finito de elementos, cada uno con un lugar y valor específicos. Estos elementos son llamados pels o píxeles.

El campo del procesamiento digital de imágenes está construido sobre bases matemáticas y probabilísticas, pero la intuición y análisis humanos juegan un importante papel al momento de escoger una técnica u otra. Esta elección se basa usualmente en juicios visuales subjetivos. La función $f(x, y)$ se caracteriza por dos componentes:

1) Iluminación: la cantidad de luz incidente procedente de la fuente sobre la escena.

2) Reflectancia: la cantidad de luz reflejada por los objetos de la escena.

La combinación de las dos componentes indica que la reflectancia está acotada entre absorción total y reflexión total. Las características de los objetos está determinada por la fuente de iluminación.
Esto se aplica también al caso en que las imágenes se forman por la transmisión de la 'iluminación' por el medio, como en los rayos X. En ese caso la segunda componente sería de capacidad de transmisión y no reflectancia. A la intensidad de una imagen

monocromática f en las coordenadas (x, y) se le denomina nivel de gris (l) de la imagen en ese punto.

Una imagen puede ser continua tanto respecto a sus coordenadas x y y, como a su amplitud. Para convertirla a forma digital, hay que digitalizarla en los dos aspectos (espacialmente y en amplitud). La digitalización de las coordenadas espaciales (x, y) se denomina muestreo de la imagen y la digitalización de su amplitud se conoce como cuantificación [GE92].

La función bidimensional (en dos ejes coordenados) mostrada en la segunda figura es una gráfica de los valores de amplitud (el nivel de gris) de la imagen continua en la primera figura a lo largo del segmento de línea AB. Las variaciones aleatorias se deben a ruido de la imagen.

Para muestrear esta función, tomamos muestras a espacios iguales a lo largo de AB, indicadas por los cuadritos blancos. El conjunto de estos cuadritos nos da la función muestreada.

Sin embargo los valores de las muestras aún se encuentran en un rango continuo de valores de niveles de gris. Para obtener una función digital, debemos convertir (cuantificar) los valores de gris a cantidades discretas. Esto se hace simplemente asignando uno de los ocho valores de la figura a cada muestra.

En la figura2.4 se pueden observar las muestras digitales que resultan del muestreo y cuantificación.

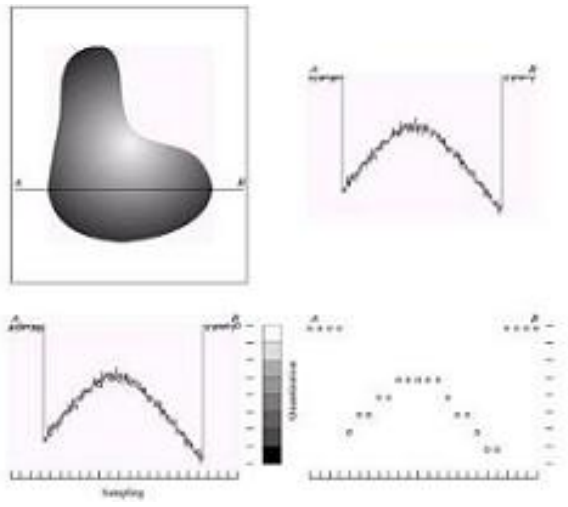


Figura 2.4: Muestras digitales que resultan del muestreo y la cuantificación.

En la práctica, el método de muestreo está determinado por el orden de los sensores utilizados para generar la imagen. Por ejemplo, si se utiliza una banda de sensores, el

número de sensores en la banda establece las limitaciones de muestreo en una de las direcciones. Si se usa un arreglo de sensores, el número de sensores del arreglo establece las limitaciones de muestreo en ambas direcciones [GE92].

Claramente, la calidad de una imagen digital se determina en gran manera por el número de muestras y niveles de gris utilizados en el muestreo y cuantificación.

La resolución de nivel de gris se refiere al más pequeño cambio discernible en nivel de gris, aunque como ya lo habíamos señalado, medir los cambios discernibles en niveles de intensidad es un proceso altamente subjetivo. La potencia de dos que determina el número de niveles de gris es usualmente 8 bits, es decir, 256 diferentes niveles de gris. Algunas aplicaciones especializadas utilizan 16 bits.

Usualmente decimos que una imagen digital de tamaño $M * N$ con L niveles de gris tiene una resolución espacial de $M * N$ píxeles y una resolución de nivel de gris de L niveles. El principal objetivo de la mejora es procesar una imagen para que el resultado sea más conveniente que la imagen original para una aplicación especifica.

La mejora de la imagen se divide en 2 categorías: métodos del dominio espacial y métodos del dominio de la frecuencia. Los métodos del dominio espacial trabajan sobre el plano de la imagen, y en éste se manipulan directamente los píxeles de una imagen. En los métodos del dominio de la frecuencia se modifica la transformada de Fourier de una imagen. Existen técnicas que se basan en combinaciones de métodos de ambas categorías.

No hay una teoría general de mejora de la imagen. Cuando la imagen se procesa para interpretación visual, el observador es el que juzga qué tan bueno es un método: la evaluación visual de una imagen es un proceso altamente subjetivo. Cuando la imagen se procesa para ser percibida por una máquina, la evaluación es más fácil: el mejor procesamiento de la imagen es aquel que provoca un mejor reconocimiento por parte de la máquina.

Uno de los algoritmos de seguimiento de objetos es la representación B-Spline de curvas y espacios de formas.
Una etapa crucial en el seguimiento de objetos es la segmentación, la cual se ocupa de detectar el objeto en cada fotograma de la secuencia. Los métodos de Level Set son una potente herramienta para la segmentación de imágenes [Cle08].

Estos métodos se encuentran basados en una deformación continua e iterativa de una superficie, la función de Level Set, la cual proporciona el contorno del objeto a segmentar. A pesar de la facilidad de adaptación de estos métodos al contorno del objeto, se requiere de un gran costo computacional para conseguir una exitosa convergencia en la siguiente imagen. Además de que si el objeto sufre cambios radicales

entre fotogramas, existe una gran probabilidad de que la segmentación del objeto no suceda. A través de un movimiento brusco del objeto, la segmentación del Level Set no converge hasta el contorno del objeto. En este caso, la segmentación es deficiente porque no se adapta al contorno del objeto, además de que se segmentan partes que no pertenecen a él.

Los algoritmos de predicción se pueden clasificar en dos grupos: los algoritmos de medición de una transformación afín y los adaptados a modelos de transformación no rígida. Estos últimos se encuentran representados por el flujo óptico y un método probabilístico de detección de movimiento [NYS04]. En el primer grupo, se destacan el acoplamiento de siluetas o formas, la búsqueda de patrones y el flujo óptico con una parametrización afín.

El sistema de segmentación se integra con un sistema bayesiano de seguimiento de objetos previamente existente. Este módulo de detección segmenta el objeto de interés en cada fotograma, permitiendo a su vez que la máscara obtenida de él sirva como plantilla. De esta plantilla se extraen las pistas que caracterizan a un objeto como fase de preprocesamiento, antes de comenzar la ejecución del algoritmo de seguimiento.

Hay dos aproximaciones para resolver el problema en aplicaciones complejas de visión por computador. La primera aproximación se basa en la utilización de la información del contexto donde tiene lugar el seguimiento. Como resultado se presenta una aplicación de anotación de video: la reconstrucción 3D de jugadas de un partido de fútbol.

Escogiendo un esquema Bayesiano de seguimiento visual, la segunda aproximación es un algoritmo que utiliza como observaciones los valores de apariencia de los píxeles de la imagen. Este algoritmo, denominado iTrack, se basa en la construcción y ajuste de un modelo estadístico de la apariencia del objeto que se desea seguir.
Algunos algoritmos usados en la actualidad son los siguientes:

- Coherencia del camino.
- Meanshift. Basado en una técnica no paramétrica para aumentar la densidad de los gradientes hasta encontrar la cima de distribuciones de probabilidad. Permite encontrar agrupamiento de datos en una muestra. De este modo, si se dispone de una imagen con puntos clasificados, por ejemplo, como piel humana, estos se encontrarán agrupados en una misma parte de la misma y, usando este algoritmo podemos encontrar este agrupamiento.
- Camshift (Continously Adaptive Mean Shift) para realizar el seguimiento de un objeto en la secuencia de imágenes. Este algoritmo es una adaptación del algoritmo Mean Shift para poder tratar las distribuciones de probabilidad dinámicas (con

cambios en su tamaño y su posición) que representan los objetos en movimiento. Este algoritmo se basa en estadística robusta, es decir, se tiende a desechar los datos atípicos de la muestra o los que se alejan demasiado del grupo, con lo que se tiende a eliminar el ruido.

- ABCshift describe un modelo de seguimiento adaptativo basado en Camshift.

- Filtro predictivo de Kalman. El filtro de Kalman es un conjunto de ecuaciones matemáticas que proveen una solución recursiva eficiente del método de mínimos cuadrados. Esta solución permite calcular un estimador lineal, insesgado y óptimo del estado de un proceso en cada momento del tiempo con base en la información disponible en el momento t-1, y actualizar, con la información adicional disponible en el momento t, dichas estimaciones. Este filtro es el principal algoritmo para estimar sistemas dinámicos especificados en la forma de estado-espacio (state-space).

- Filtro de Partículas. Los filtros de partículas son estimadores secuenciales Monte Carlo. El propósito general de estos algoritmos es representar funciones de densidad de probabilidad y su evolución en el tiempo. Estas representaciones están basadas en muestras discretas de las funciones que son modeladas. Estas muestras discretas se llaman partículas, y están formadas por un estado (x) y un peso (π).

2.3. MODELO DE COLOR Y ESTÁNDAR DE VIDEO

2.3.1. MODELO DE COLOR El color de un objeto es dado físicamente por la reflexión de longitudes de ondas en el espectro electromagnético que comprende la luz. El espectro de la luz visible por el ojo humano se compone de longitudes de onda entre 400 nm (violeta) y 700 nm (rojo) como se muestra en la figura 2.5.

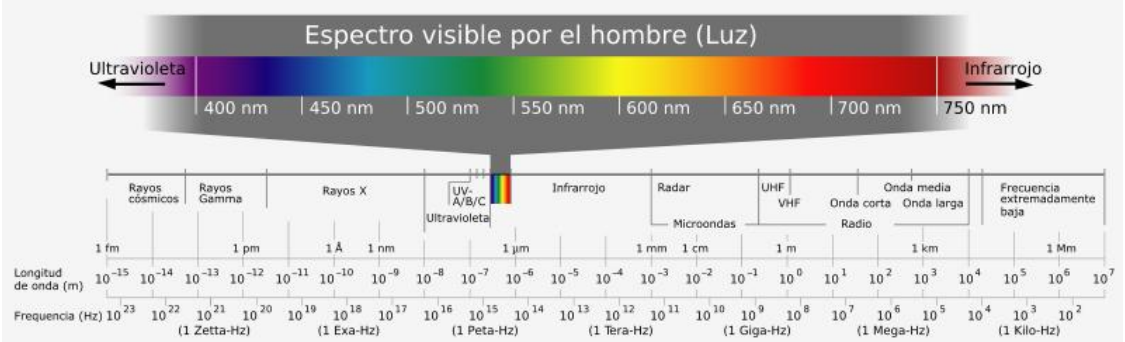


Figura 2.5: Espectro Electromagnético de Luz Visible

Un tono puede ser representado mediante diversos modelos, por ejemplo, mezclando los tres colores primarios que actúan sobre el brillo y la saturación.

2.3.2. MODELO RGB El RGB (rojo, verde, azul) es un modelo aditivo que permite la representación de cualquier color mediante la combinación de tres colores principales: rojo, verde y azul. Aditivo se define como la luz que se agrega cada vez a la 'oscuridad' inicial para dar origen a una imagen. Fue creado con el desarrollo de la televisión en color y todavía se utiliza en la pantalla de rayos catódicos, plasma o LCD, cada píxel de la pantalla viene dado por la superposición, con diferentes pesos de los tres colores principales. En el caso de la tecnología de rayos catódicos, cada píxel se compone de tres fósforos de diferentes colores (rojo, verde y azul) dispuestos en un modo muy estrecho, lo que no hace posible distinguirlos por separado y sólo se pueden percibir en conjunto. Al variar la intensidad de cada fósforo que se ilumina, es posible obtener el tono deseado para cada píxel [Oli07].
El modelo RGB puede ser representado por un cubo, donde los tres ejes corresponden a tres colores, cada color se identifica por tres números, por lo general de 8 bits cada uno (en el dominio digital), cubriendo así el rango de valores decimales entre 0 y 255; además de este formato (24 bits) hay representaciones de 16, 32 o 48 bits (16 bits por componente) como se muestra en la figura 2.6.

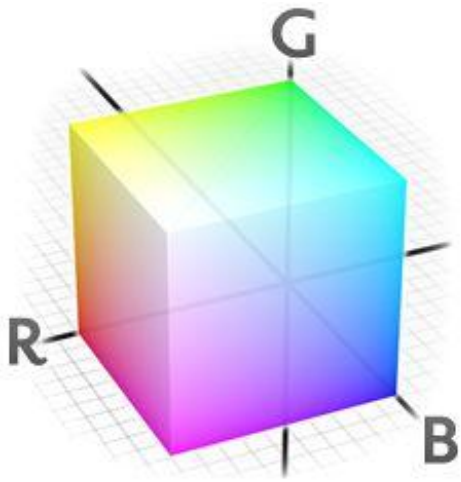


Figura 2.6: Representación del Modelo RGB

RGB no proporciona una definición espectroscópica del rojo, verde y azul, es decir, qué no está asociado a cada uno de estos tonos de longitud de onda, y cuando esto sucede, el modelo de color se convierte en un espacio de color absoluto. Se define toda la gama de tonos de la representación una vez elegido un determinado espacio de color el cual es un subconjunto de colores dispuesto en el CIE (Comisión Internacional de la Luz) como se muestra en la figura 2.7.

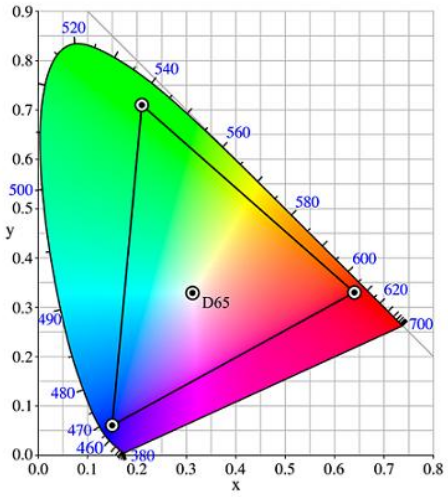


Figura 2.7: Gama de Espacio de Color RGB

Hay que tener en cuenta que los extremos del triángulo son los colores principales del modelo RGB, los cuales fueron elegidos como la maximización de la diferencia en la respuesta de los conos presentes en el ojo humano.

2.3.3. MODELO YUV El YUV se utiliza en el estándar de video compuesto NTSC, PAL y SECAM, y es un modelo que representa un color por una componente de luminancia (Y) y dos componentes de crominancia (U y V). La luminancia contiene información relativa sobre el brillo, es decir, la parte acromática o blanco y negro de la imagen, lo contrario de la crominancia que se refiere a sus componentes de color. Los valores de Y, U y V se derivan directamente de los de R, G y luminancia B que vienen dados por una suma ponderada, U se obtiene restando la Y de la señal azul y un factor de escala adecuado, V es la diferencia entre R y Y, subiendo a un factor diferente [Oli07].
El espacio de color YUV no es un absoluto, sino que se limita a la codificación del formato RGB: los colores efectivamente representados dependen por lo tanto de la superficie utilizada por los tres componentes de rojo, verde y azul. Otros dos formatos que se basan en la luminancia y la crominancia son YPrPb y YCrCb, la primera utilizada para señales analógicas y el segundo en un entorno digital. En comparación al YUV, hay pequeñas diferencias en los componentes de crominancia: en este caso porque Cb/Pb y Cr/Pr representan desviaciones a lo largo del eje gris respectivamente del amarillo-azul y rojo-azul (Cian).

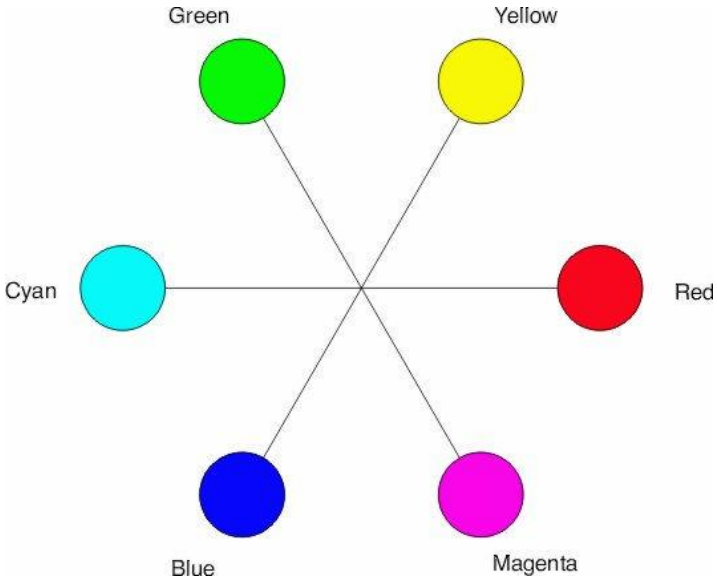


Figura 2.8: Representación de los Colores Principales y sus Complementarios

La codificación de luminancia/crominancia es ampliamente utilizado para las emisiones de televisión. Su ventaja principal es garantizar la compatibilidad con aparatos antiguos en blanco y negro, ya que utiliza dos canales separados para los niveles de grises y colores. Otra ventaja del YUV es la posibilidad de descartar la información, reduciendo así el ancho de banda utilizado, sin comprometer la correcta visualización de las imágenes.
En cuanto a mantener la exactitud de la representación de video es mucho más importante en precisión el componente de luminancia y la señal puede ser comprimida mediante la eliminación de las porciones de los datos de crominancia: el ojo humano es capaz de interpretar los colores por igual correctamente. Este enfoque se utiliza generalmente en YCrCb 4:2:2, que está cubierto por dos muestras de crominancia por cada cuatro de luminancia. Además de las emisiones de televisión, una amplia gama de aplicaciones importantes para la YCrCb es la compresión de video digital: un ejemplo es el algoritmo MPEG-2.

2.4. ESTÁNDAR ANALÓGICO Y DIGITAL

2.4.1. ESTÁNDAR ANALÓGICO NTSC NTSC, PAL y SECAM TV son tres estándares de televisión analógica que se utilizan actualmente en diferentes regiones del planeta. Son incompatibles entre sí, ya que difieren en el número de líneas y cuadros, frecuencia horizontal y vertical, la frecuencia portadora de color ancho de banda de video y el transporte de audio. Pero todo el uso de 'entrelazado', es decir, para cada fotograma (imagen única que comprende una secuencia de video) se transmiten sólo las líneas pares o las impares, o alternativamente, que recibirán durante el desentrelazado de líneas: la imagen es reconstruida con base a las líneas pares del cuadro que se acaba de recibir del cuadro anterior y las líneas impares (o viceversa).

El entrelazado permite un importante ahorro de ancho de banda sin comprometer la correcta visualización del video, ya que el ojo humano no se da cuenta de la superposición de dos marcos contiguos, a frecuencias suficientemente altas (50-60 Hz). La técnica de enviar primero las líneas pares y luego las impares (o viceversa) permite, de hecho, la mejora de la calidad de video. Visualizando todo el marco, con el uso de CRT (Tubo de Rayos Catódicos) se creó un inconveniente, ya que el haz de electrones dibuja el desplazamiento de la imagen por la pantalla de izquierda a derecha y de arriba hacia abajo: esto significa que la parte superior del marco es más estable, mientras que la inferior se percibe en continua vibración, ya que permanece visible durante un tiempo demasiado corto. Con el entrelazado se resuelve este problema [Oli07].
NTSC significa Comisión Nacional de Sistemas de Televisión. El organismo de estandarización estadounidense lo creó en 1953. Se transmite principalmente en América y Japón. Un cuadro completo NTSC se compone de 525 líneas, de los cuales 480 contienen la información de video y el restante son necesarios para la sincronización o para transmisión de datos auxiliares. La frecuencia de envío del marco entrelazado (campo) es de 60 Hz, por lo que cualquier fotograma completo se visualiza cada 1/30 segundos (en realidad 1/29.97). A partir de estos valores podemos deducir que la frecuencia con que se envía cada línea es de alrededor de 15734 kHz. El ancho de banda total de transmisión de video es de 6 MHz: la señal de video en sí es modulada en amplitud con una portadora de 1,25 MHz y ocupa un ancho de banda de 4,2 MHz a la derecha de tal frecuencia y 750 kHz a su izquierda. La información sobre el color (en formato YIQ, variante del YUV) es modulada en cuadratura, y posee una frecuencia portadora de 3,579545 MHz más alta que la de video. La señal de audio es modulada en frecuencia con portadora de 5,75 MHz (4,5 MHz por encima de la frecuencia portadora de video).

Caracteristicas	**NTSC**
Lineas/Campos	525/60
Frecuencia Horizontal	15.734 kHz
Frecuencia Vertical	60 Hz
Frecuencia de la Subportadora de Color	3.579545 MHz
Ancho de Banda del Video	4.2 MHz
Portadora de Sonido	4.2 MHz

Tabla 2.1: Caracteristicas del Estandar

2.4.2. VGA VIDEO GRAPHICS ARRAY Sistema gráfico de pantallas para PCs desarrollado en 1987 por IBM. VGA se convirtió en un estándar que la mayoría de fabricantes decidió seguir. Las tarjetas gráficas VGA estándares traían 256 KB de memoria de video. En modo texto, el sistema VGA provee una resolución de 720 x 400 píxeles. En modo gráfico permite 640 x 480 (con 16 colores) ó 320 x 200 (con 256

colores). El número total de colores de la paleta es 262.144 y tasa de refresco de hasta 70 Hz. A diferencia de estándares para PC más viejos (MDA, CGA y EGA), VGA utiliza señales analógicas. Por esta razón un monitor diseñado para estándares anteriores no puede utilizar VGA [Oli07]. Desde que se introdujo en 1987, muchos otros estándares han sido desarrollados, ofreciendo mayores resoluciones y más colores. Oficialmente VGA fue reemplazado por XGA. Sin embargo han sido lanzados otros sistemas clones superiores que han sido conocidos como SVGA.
Resoluciones gráficas soportadas por VGA:

- 640x480 en 16 colores.
- 640x350 en 16 colores.
- 320x200 en 16 colores.
- 320x200 en 256 colores.

Resoluciones de texto soportadas por VGA:

- 80x25.
- 40x25.

Ambas con 16 colores disponibles para el primer plano y 8 colores para el fondo. En los colores de video VGA estándar para cada píxel envía el valor analógico de los tres colores básicos (rojo, verde, azul), en tres cables separados. Un video en formato VGA es una secuencia de cuadros: cada uno se compone de una serie de líneas horizontales, y cada una de ellas se compone de un número de píxeles. Las filas se transmiten de arriba hacia abajo (a VGA estándar no entrelazada), mientras que los píxeles son enviados de izquierda a derecha, por lo que se necesita para definir el final de cada línea y cada marco a través de las señales de sincronización adecuada respectivamente HSync y VSync. Estas señales se transmiten por cables separados, también en el canal dedicado a verde y el código de una señal de sincronismo compuesto, obtenida por el XOR entre HSync y VSync.
Cada línea de un marco comienza con una región de video activa, que se transmiten en valores RGB para cada píxel, seguida por una zona de corte en el que se hallan píxeles negros. Dentro de esta región también se transmite un pulso de sincronización horizontal Canal HSync: el intervalo de corte antes del impulso se llama parte frontal, mientras que el siguiente es llamado nuevo Poch. Lo mismo es cierto para los marcos dentro de una secuencia de video y para la señal de sincronización vertical VSync.

2.4.3. ESTÁNDAR ITU-R BT.656 El ITU-R BT.656 es un estándar de codificación de un formato de video digital YCbCr 4:2:2, utilizado para datos de salida del VDEC1. Es así como el transporte de información de video, encapsula el flujo de bits, códigos especiales necesarios para la sincronización en lugar de o además de las habituales

señales HSync, VSync y Campo. Para describir el estándar consideramos una señal de tipo NTSC, que consta de dos campos, uno para las filas pares (242 líneas) y uno para los impares (243). Para cada uno de ellos hay una zona vertical blanking superior (19/20 líneas) en el que se transmiten datos adicionales, una región de video activa (252 líneas) que contienen información de video y una zona adicional vertical blanking de fondo mucho menor (2 líneas) [Oli07].
El ITU-R BT.656 determina que cada línea se codifica de la siguiente manera:

- Código EAV (Fin de Video Activo), 4 bytes.
- Blanking Horizontal, 280 bytes.
- Código SAV (Inicio de Video Activo), 4 bytes.
- Video Activo, 1440 bytes.

EAV y SAV son respectivamente, quienes indican el comienzo y el final de la zona Activa de Video, por línea: ambos se componen de un byte que contiene todos los '1', dos bytes '0' y un byte especial que tiene una estructura como esta: ((1 - F - V - H) - xor (H - F)) xor (H - F) xor (V - F) xor (H) xor (V). El bit F indica en qué campo está (F=0 es Campo 1, F=1 es Campo 2), mientras que el bit V especifica si la línea en cuestión pertenece a la zona de borrado vertical (V = 1) o el video activo (V = 0). H es el único elemento distintivo de SAV (H=0) y EAV (H=1). Los restantes cuatro bits de XOR se utilizan para detectar y corregir cualquier error de uno o dos bits que pueden estar presentes en F, V, H.
En la región de Video Activo se transmiten los valores de Y,Cb y Cr (un byte cada uno) relativos de cada píxel, especialmente de cada cuatro muestras de luminancia se envían dos azules y dos rojas de crominancia, de acuerdo con el formato 4:2:2, que proporciona algunas variaciones. Se puede por ejemplo transmitir una componente de crominancia para todos los píxeles iguales y otro para los impares para obtener (Cb0,Y0),(CR1,Y1), ..., o hacer un promedio entre los valores de Cb y Cr para un píxel dado, y el píxel correspondiente siguiente, con el resultado de que la codificación conjunta de dos píxeles adyacentes, es decir, (CB01, Y0, Cr01, Y1),(CB23, Y2, CR23, Y3).
Se pueden transmitir los componentes de crominancia sólo cada dos píxeles, mientras que la luminancia se transmite regularmente sin submuestreos: la secuencia digital en este caso es Cb0, Y0, CR0, Y1, CB2, Y2, Cr2. Hay que tener en cuenta que en cualquier variante se utiliza, el segundo byte como el cuarto, sexto, etc, y siempre representa una muestra de luminancia.

Field 1 – First Vertical Blanking (Top) - Repeat for 22 (19) lines																
EAV Code				Blanking Video				SAV Code				Active Video				
255	0	0	182	128	16	128	16	255	0	0	171	128	16	128	16	
Repeat 1 (1) time				Repeat 70 (67) times				Repeat 1 (1) time				Repeat 360 (360) times				

Field 1 - Active Video - Repeat for 288 (240) lines																
EAV Code				Blanking Video				SAV Code				Active Video				
255	0	0	157	128	16	128	16	255	0	0	128	240	41	110	41	
Repeat 1 (1) time				Repeat 70 (67) times				Repeat 1 (1) time				Repeat 360 (360) times				

Field 1 - Second Vertical Blanking (Bottom) - Repeat for 2 (3) lines																
EAV Code				Blanking Video				SAV Code				Active Video				
255	0	0	182	128	16	128	16	255	0	0	171	128	16	128	16	
Repeat 1 (1) time				Repeat 70 (67) times				Repeat 1 (1) time				Repeat 360 (360) times				

Field 2 – First Vertical Blanking (Top) - Repeat for 23 (20) lines																
EAV Code				Blanking Video				SAV Code				Active Video				
255	0	0	241	128	16	128	16	255	0	0	236	128	16	128	16	
Repeat 1 (1) time				Repeat 70 (67) times				Repeat 1 (1) time				Repeat 360 (360) times				

Field 2 - Active Video - Repeat for 288 (240) lines																
EAV Code				Blanking Video				SAV Code				Active Video				
255	0	0	218	128	16	128	16	255	0	0	199	240	41	110	41	
Repeat 1 (1) time				Repeat 70 (67) times				Repeat 1 (1) time				Repeat 360 (360) times				

Field 2 - Second Vertical Blanking (Bottom) - Repeat for 2 (3) lines																
EAV Code				Blanking Video				SAV Code				Active Video				
255	0	0	241	128	16	128	16	255	0	0	236	128	16	128	16	
Repeat 1 (1) time				Repeat 70 (67) times				Repeat 1 (1) time				Repeat 360 (360) times				

Figura 2.9: Ejemplo de un fotograma de video codificado en formato ITU-R BT.656, los valores entre paréntesis se refieren a una señal NTSC

2.5. COMPONENTE COMPUESTO Y S-VIDEO

Una señal de video analógica puede ser transmitida por un solo cable o dividirse en dos o más componentes y el envío en líneas independientes: en el primer caso hablamos de video compuesto y en el segundo de video por componentes. Un ejemplo de video por componentes está dado por una señal que se transmite a través de los valores analógicos de rojo, verde y azul: se caracteriza por la alta calidad, sino que requiere una banda amplia y contiene muchos datos redundantes, en el momento en que cada canal lleva información sobre la misma imagen en blanco y negro. Las señales de sincronización horizontal y vertical se pueden enviar en dos hilos separados, como en el estándar VGA o mezclados entre sí y enviados a través de un canal dedicado, como sucede en la Comunidad Europea de TV SCART, y un tercer método, menos común, implica la inclusión de la señal de sincronización en el hilo dedicado a la componente verde [Oli07].

Hablamos de video por componentes cuando hacemos referencia a una señal YPbPr. En este caso se resuelve el problema de la redundancia GBR típico, como la imagen en blanco y negro y se transporta sólo en el canal de luminancia, también en los componentes de crominancia y se puede realizar un submuestreo para reducir el ancho de banda requerido.

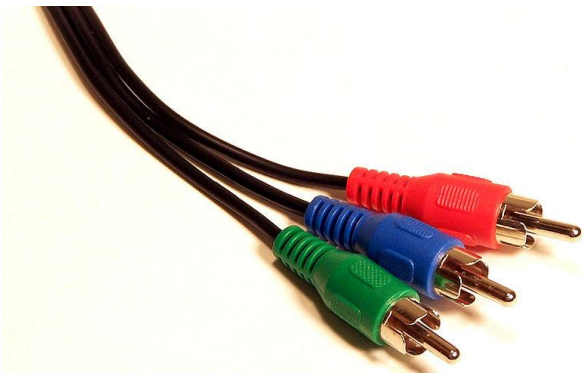


Figura 2.10: Conectores RCA para Señales de Tipo Componente

Un ejemplo particular de video por componentes está representado por una señal descompuesta en una porción de luminancia y en otra de crominancia, transmitida en dos hilos separados: hablamos en este caso de Video Independiente (S-Video). Utilizando sólo dos canales en lugar de tres, hay una reducción del ancho de banda disponible que conduce a una reducción en la calidad de video. El S-Video se usa principalmente en EE.UU., Canadá, Australia y Japón, por ejemplo, en aparatos de televisión y reproductores de DVD, mientras que en Europa prefieren hacer uso del RGB.

Figura 2.11: Conector S-Video Estándar

En una señal de tipo compuesto se utiliza un solo cable para transmitir toda la información relativa a una imagen y la luminancia es una señal en banda base y que contiene también el impulso de sincronización, y el componente de color es modulado con cierta frecuencia portadora. Este es el formato típico de las señales de TV analógica (NTSC, PAL, SECAM); el conector más común es un conector RCA, generalmente de color amarillo, a menudo acompañado de dos cables (rojo y blanco) para señales de audio respectivamente de izquierda y derecha como el que se muestra en la figura 2.12:

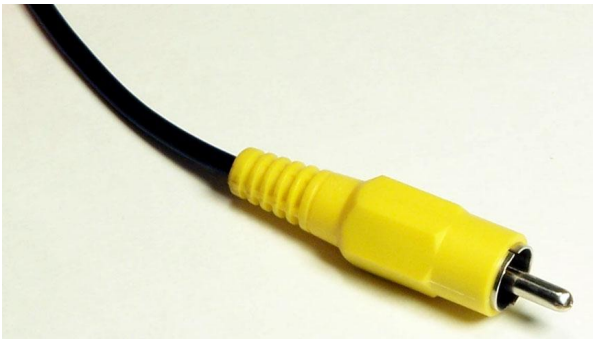


Figura 2.12: Conector de Video Compuesto

2.6. ARQUITECTURA DE LA FPGA SPARTAN 3E

Los dispositivos FPGA están conformados principalmente por Bloques Lógicos Configurables (CLB: Configurable Logic Blocks) y por Bloques de Entrada-Salida (IOB: Input/Output Blocks). De manera más precisa, los CLBs son las unidades que ejecutan las operaciones combinacionales, aritméticas y de memoria necesarias para la implementación de la aplicación descrita en el lenguaje de configuración de la FPGA. Por su parte, los IOBs son módulos de interconexión entre los pines del circuito integrado FPGA y la lógica interna del dispositivo.

Además de estos elementos fundamentales, las FPGA suelen tener otros elementos que dependerán del fabricante y del modelo del integrado. Estos recursos van desde memorias, multiplicadores, hasta arreglos lógicos más complejos como lo son los administradores de reloj. Todos estos recursos añadidos permiten realizar diversas tareas en los distintos diseños basados en FPGA, y al estar dedicados para funciones específicas, permiten implementar aplicaciones con alta eficiencia. Esto finalmente redunda en un mejor aprovechamiento de los recursos de uso global, especialmente en la minimización de la cantidad de CLBs utilizados.
Los recursos de uso global (CLBs) ocupan el área más extensa del circuito integrado. Sin embargo, algunos recursos dedicados para operaciones específicas también se encuentran presentes. De manera más precisa, los recursos adicionales son: los administradores digitales de reloj (DCMs: Digital Clock Managers), las memorias RAM y los multiplicadores dedicados de 18x18 bits con signo.
Para el caso particular de la FPGA Spartan 3E, los CLBs se encuentran constituidos por 4 paquetes o divisiones denominados SLICEs [Inc09b].

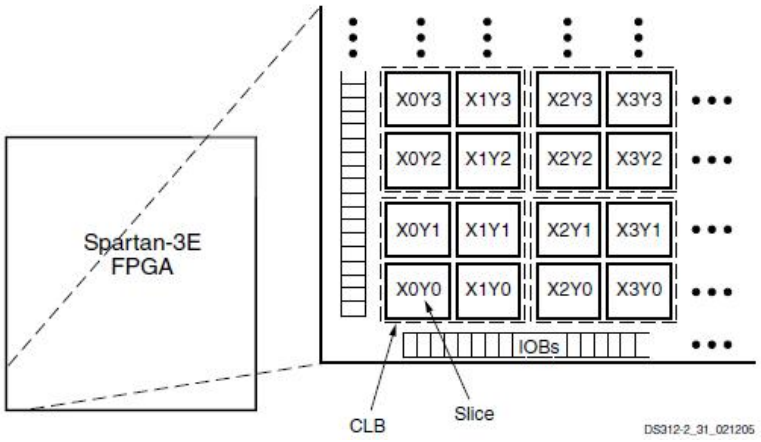


Figura 2.13: CLBs constituidos por SLICEs

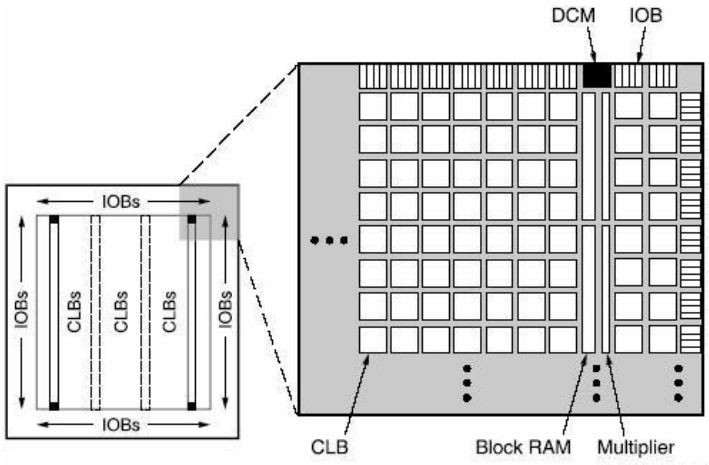


Figura 2.14: Arquitectura de la Plataforma Spartan 3E

Cada SLICE contiene 2 tablas de verdad o LUTs (Look Up Table) que permiten la implementación de cualquier función lógica de cuatro entradas y una salida, por lo que las LUTs no tienen limitaciones en cuanto a la complejidad de la función a implementar y su retardo será constante independientemente de la función combinacional que realice. Por otro lado, habrá limitación si la función excede las 4 entradas.
Debido a la existencia de la restricción en el número de entradas de los LUTs, los SLICEs también poseen una lógica aritmética y de acarreo en las salidas de los LUTs, que se activan en aquellos casos en los cuales las funciones lógicas posean más de cuatro entradas. Además de esto, la lógica de acarreo también se encuentra interconectada entre LUTs de varios SLICEs, por lo que se pueden configurar funciones lógicas con una cantidad de entradas aún mayor. Finalmente, cada SLICE posee un registro por cada salida. Estos registros, permiten almacenar el resultado de toda la operación lógica durante un tiempo equivalente a un ciclo de reloj. Los bloques de entradas-salidas (IOBs) son las interfaces entre los pines del dispositivo FPGA y el circuito lógico configurable interno.
Específicamente, en la FPGA Spartan 3E los IOBs pueden ser configurados de manera unidireccional o bidireccional según los requerimientos del diseño. Además, los IOBs tienen la ventaja de estar diseñados para trabajar con diversos estándares digitales de tensión diferencial (LVDS, BLVDS, LDT, LVPECL) y de tensión referenciada (LVTTL, LVCMOS, PCI-X, PCI, GTL, GTLP) [Inc09b].

2.7. TARJETA DE DESARROLLO SPARTAN-3E STARTER KIT

Para la implementación del sistema en la FPGA, se utilizó la tarjeta de desarrollo Spartan-3E Starter Kit. Esta tarjeta es fabricada por la empresa Diligent Inc, y está diseñada con la finalidad de facilitar la implementación de los diseños en la FPGA Spartan 3E. La tarjeta de desarrollo utilizada en este trabajo posee una FPGA Spartan-3E modelo XC3S500E fabricado por Xilinx, Inc. Este dispositivo electrónico configurable contiene 500000 compuertas lógicas distribuidas en 1164 CLBs [Inc09b].

Adicionalmente, la tarjeta de desarrollo está configurada de modo que la FPGA está conectada a una serie de recursos periféricos que permiten la comunicación entre la FPGA y los dispositivos electrónicos externos. A continuación se listan los recursos periféricos más importantes de la tarjeta de desarrollo Spartan-3E Starter Kit:

- Puertos para comunicación serial DB-9.
- Puerto VGA.
- Conector USB para la configuración de la FPGA.
- Interruptores deslizables, pulsadores, leds y pantalla LCD.
- Conector de alimentación 5VDC.
- Puerto ethernet.
- Conversor ADC y DAC.
- Puerto PS2

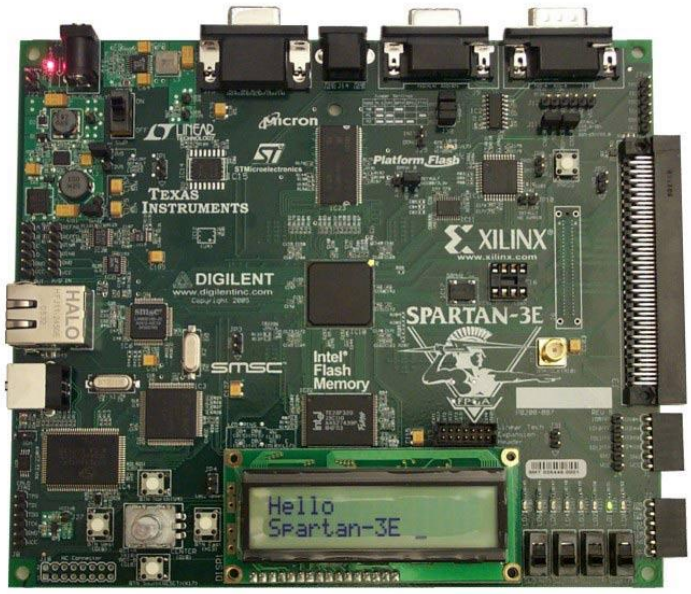


Figura 2.15: FPGA y Recursos Periféricos de la Tarjeta Spartan 3E

2.8. TARJETA DECODIFICADORA DE VIDEO VDEC1

La Tarjeta VDEC1 es una tarjeta que se conecta a la Tarjeta Spartan 3E a través de un conector de expansión de alta velocidad. Su función es la de digitalizar una señal de video analógico PAL, NTSC o SECAM al formato YCbCr 4:2:2 (8 ó 16 bits, de acuerdo a la UIT-R BT.656). Además del conector de 100 pines para interfaz con la Tarjeta

Spartan 3E, hay cinco pines que permiten la adquisición de la señal de video en modo S-Video, Video Compuesto o por Componentes.

Figura 2.16: Tarjeta Digilent VDEC1

La Tarjeta VDEC1 posee un decodificador de video integrado ADV7183B. Se trata de un decodificador de video que detecta automáticamente las señales de TV en normas diferentes (PAL, NTSC,SECAM) y las digitaliza a través de tres codificadores analógico-digitales (ADC) de 10 bits que funcionan a una frecuencia de 54 MHz. La señal de entrada es multiplexada y, antes de convertirla, es adaptada utilizando un circuito especial que tiene los niveles de tensión requeridos para el buen funcionamiento de los ADC.
Los convertidores de señal digital de salida, se preprocesan y dividen en dos componentes (de 10 bits cada uno), uno para la luminancia y otro para la crominancia, los cuales entrarán en un SDP (Procesador de Definición Estándar) que maneja todos los filtrados y remuestreos de operaciones necesarias para ajustar la señal al formato deseado para la salida. Los diversos cálculos se realizan en paralelo en dos canales distintos (luminancia y crominancia) los cuales son similares, y el componente de luminancia extrae las señales de sincronización vertical y horizontal. El SDP también contiene un bloque específico a la entrada analógica de auto-formato.
La interfaz de salida digital avanzada y altamente flexible permite la decodificación de video y el rendimiento de conversión en los sistemas basados en línea de bloqueo de reloj. Esto hace que el dispositivo sea ideal para una amplia gama de aplicaciones con diversas características de video analógico, incluyendo las fuentes basadas en cinta, las fuentes de emisión, de seguridad o cámaras de vigilancia y sistemas profesionales. [Inc05a]
A continuación se listan las características más importantes de la tarjeta decodificadora:

- Decodificador de video multiformato soporta NTSC-(J,M,4.43), PAL(B/D/G/H/I/M/N), SECAM.

- Tres Integrados de 54 MHz, 10-bit ADC.
- Velocidad de reloj de un solo cristal de 27 MHz.
- Línea de bloqueo de reloj compatible (LLC).
- 5-filtros de línea adaptativos.
- Mejora de Chroma transitoria (CTI).
- Reducción de ruido digital (DNR).
- Múltiples formatos de entrada analógica programable de video compuesto (CVBS).
- S-Video (Y/C)
- Componente YPrPb (VESA, MII, SMPTE, y Betacam)

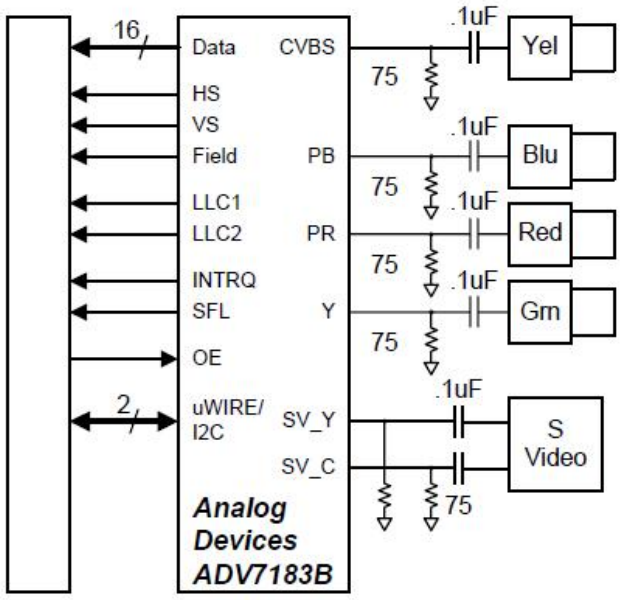


Figura 2.17: Entradas y Salidas Principales del ADV7183B

En la figura 2.17 se muestran las principales entradas y salidas del decodificador de video. En la parte derecha se pueden observar las entradas analógicas:

- CVBS (Blanking Sync y Video Compuesto).
- Y, PR y el PB son los tres componentes del formato de Video por Componentes.

- SV-Y y SV-C son los componentes de luminancia y crominancia del S-Video.

En la parte izquierda se puede observar la interfaz a la Tarjeta Spartan 3E. En particular hay las siguientes señales [Inc05b]:

- Data [15:0]: 16 puertos por donde se envía la información de los píxeles. Si utilizamos el formato YCbCr de 8 bits, sólo se utilizan los bits más significativos [15:8].
- HS (Sincronización horizontal), VS (Sincronización vertical) y FIELD (señales de campo en la sincronización del fotograma de Video).
- LLC1 (Línea de bloqueo de reloj). La señal de reloj que marca el flujo de bytes de datos cuando Data es de 8 bits. La salida del decodificador nominalmente tiene una frecuencia de 27 MHz, aunque puede haber ligeras variaciones según la longitud de las líneas de video.
- LLC2 reloj de frecuencia de la mitad de la LLC1. La señal de reloj que marca el flujo de bytes de datos cuando Data es de 16 bits.
- INTRQ (Petición de interrupción). La señal de solicitud de interrupción habilita al pin INTRQ del ADV7183B si detecta una entrada de video en particular.
- SFL (Subportadoras de Bloqueo de Frecuencias). Este pin contiene una secuencia serial de salida que puede ser usada para bloquear la frecuencia de la subportadora cuando el decodificador no esté conectado a algún codificador de video.
- OE (Output Enable): Cuando es uno, las salidas DATA, SA, SV y SFL se colocan en estado de alta impedancia, y si el bit es cero, quedan habilitadas.
- I2C (puerto MPU) consta de dos conexiones, el SDA (Serial Data,I/O) y el SCLK (Reloj de Entrada Serial,I) necesarios para la interfaz con el bus I2C, lo que proporciona un canal para datos y uno para el reloj.

2.8.1. REGISTROS DEL ADV7183B El ADV7183B cuenta con 249 registros de 8 bits, que permite al usuario configurar el decodificador para que sea utilizado para diversas aplicaciones y tomar el control del estado. En general, hay tres clases de registros:

- Registros Globales de Control: Son utilizados por ejemplo para apagar o reiniciar el sistema, elegir qué y cómo utilizar el ADC y la configuración de los pines del chip (activar la señal de temporización y la polaridad del reloj del LLC).
- Registro de Estado: Permite conocer el estado del sistema, facilitando, por ejemplo, el resultado del autoreconocimiento del formato de video de entrada.

- Registros de configuración del SDP: En particular se ocupa de la elección del formato de video a utilizar (PAL, NTSC, SECAM) o la activación de los controles del autoreconocimiento de color (brillo, contraste, saturación), de la gestión del procesamiento de la señal en su componente de luminancia y crominancia (sujeción, filtrado, control de ganancia, reducción de ruido), de la salida codificación del formato digital y de la temporización de las señales de sincronización (HS, VS, FIELD).

Entre los registros más importantes a citar están: el Registro de Control de Entrada (dirección 0x00), que le indica al decodificador que la señal de entrada de video está presente (S-video, compuesto ó por componentes), y el Registro de Salida de control (0x03), que le permite seleccionar el formato de salida YCbCr de 8 ó 16 bits. Los registros son accesibles a través de un bus I2C, gracias a la interfaz MPU del ADV7183B. [Inc05a]

2.8.2. PROTOCOLO I2C Y LA CONFIGURACIÓN DE REGISTROS El bus I2C consiste simplemente de dos líneas bidireccionales, dato en serie (SDA) y reloj en serie (SCL), que transmiten la información entre los dispositivos conectados al bus. Cada uno de estos se identifica por una dirección única y pueden actuar como transmisores del receptor los cuales se pueden configurar como maestro o esclavo, es decir, el dispositivo maestro inicia la transmisión de datos y genera la señal de reloj. De momento el maestro puede compartir el mismo bus, el protocolo establece un procedimiento de arbitraje para evitar conflictos y pérdida de datos.
Las transmisiones comienzan con una condición de START y terminan con un estado de STOP, que se caracteriza por una transición, respectivamente, de arriba abajo y de abajo hacia arriba, la señal SDA a un alto nivel de SCL. Para evitar ambigüedad durante la transmisión normal de datos SDA sólo puede cambiar cuando el reloj es cero. Cada paquete consta de un byte, se transmiten del bit más significativo al menos significativo, seguido por un bit de reconocimiento: el noveno ciclo de reloj (del primer bit enviado) del dispositivo receptor envía a la línea SDA a un nivel bajo, indicando que el transmisor recibe los datos correctos, y si esto no ocurre se interrumpe la transmisión. El estándar de comunicación maestro-esclavo se compone de cuatro fases: START, el envío del direccionamiento del dispositivo esclavo, envio de datos, STOP. El protocolo prevé dos tipos de direccionamiento, de 7 a 10 bits. La primera consiste en que hemos utilizado, en la transmisión de ocho bits, que representan los siete primeros y la última dirección indica si la operación que se realiza en la dirección indicada será de escritura (bit 0) o de lectura (bits a 1). Sólo el dispositivo esclavo con el direccionamiento correspondiente con el enviado por el maestro responde con el bit de acuse de recibo, y una vez se establece la conexión, se puede iniciar la transferencia de datos, un byte a la vez no se genera hasta la condición de STOP.

El ADV7183B, la interfaz con un bus I2C (implementado como un núcleo de software de VHDL dentro de la FPGA) se extiende a lo largo del puerto MPU como un

dispositivo esclavo, que no es capaz de iniciar una comunicación, y sólo puede responder a las peticiones de lectura o escritura de un maestro, en nuestro caso del bloque de comunicación I2C implementado en la Tarjeta Spartan 3E. El decodificador de video, así como una dirección adecuada para la conexión del dispositivo esclavo I2C, cuenta con 249 sub-direcciones, cada una asociada a un registro interno. Para acceder a uno de estos registros debe ser el dispositivo maestro, después de enviar la dirección al ADV7183B, incluido, en su primer byte de datos, la sub-dirección correspondiente a la ubicación donde usted quiere ir con las operaciones de escritura/lectura.

La subdirección se almacena en un registro (Registro Select, RS), que se incrementa automáticamente en cualquier cambio dado, para que la comunicación se mantenga de forma continua, a través de todos los registros hasta una condición de STOP. En la práctica, en cada operación, la interfaz MPU lee el Registro de Selección para saber a que registro acceder. [Inc05a]

2.8.3. CÁMARA LB1000 En nuestro trabajo, la señal de entrada al VDEC1 es suministrada por una cámara (modelo LB1000), que cumple con las especificaciones requeridas para la aplicación. Para ello se requiere un cable de alimentación y uno de S-Video por donde se transmite la información. La cámara está diseñada para pequeños sistemas de vigilancia, mini portátiles, móviles, así como para lugares de residencia donde el tamaño pequeño, la alta resolución y la construcción con el micrófono son factores importantes. Características técnicas:

- Base Tecnología Óptica: Lente 6mm.
- Sensor: CMOS a Color 1/3".
- Resolución: 380 Líneas.
- Iluminación Mínima: 1 Lux.
- Radio S/N: Mayor a 45 dB.
- Salidas: RCA con fono de video (amarillo) y línea de audio (blanco).
- Balance Blanco/Ganancia/Control AE: Auto.
- Salida de Audio: Si.
- Temperatura de Operación: -10 to 45 grados °C.
- Peso: 3 oz.
- Voltaje de Alimentación: DC 9V-12V a 50mA.

Esta cámara maneja un estándar de video NTSC el cuál se explicó en la sección 3.4.1.

Figura 2.18: Cámara LB1000

2.9. EL LENGUAJE DE DESCRIPCIÓN DE HARDWARE VHDL

VHDL es un Lenguaje de Descripción de Hardware para Circuitos Integrados de Muy Alta Velocidad (VHDL: VHSIC Hardware Description Languaje, donde a su vez VHSIC es Very High Speed Integrated Circuit). Este fue un lenguaje patrocinado por el Departamento de Defensa de los Estados Unidos y el Instituto de Ingenieros Electricistas y Electrónicos (IEEE) en el año 1980. Las características principales de este lenguaje de programación son:

- Estructuración del programa en forma jerárquica. Esta característica permite la descomposición del código y la reutilización de sus módulos en el mismo proyecto o en otros proyectos distintos.
- Capacidad de especificar comportamientos en forma de algoritmos o en forma de estructura de hardware real, permitiendo realizar operaciones con alto grado de paralelismo.

Todos los módulos de un programa realizado en VHDL están conformados por una declaración de entidad y una definición de arquitectura. En la Declaración de Entidad están especificados los puertos de entradas y salidas del circuito lógico, así como los tipos de datos que son válidos para tales entradas o salidas. Cada puerto tendrá a su vez un nombre o etiqueta asignada por el usuario, que permite la identificación y la interconexión con los otros módulos VHDL.
La programación estructural de VHDL permite que estos módulos puedan ser vistos como 'cajas negras' ó 'envolturas' que poseen una interfaz que permite la interacción entre la estructura interna del dispositivo lógico configurable y el circuito electrónico externo. Las 'cajas negras ó envolturas' son la representación de las entidades de los módulos VHDL y las interfaces constituyen los puertos declarados en la entidad

correspondiente.
Por su parte, en la definición de la Arquitectura, se destaca el nombre de la arquitectura así como la entidad a la cual se encuentra asociada. También posee una zona para declaración de señales internas que facilitan la elaboración del código. Posteriormente se encuentra el lugar donde son declaradas las instrucciones a ser ejecutadas, de manera algorítmica (instrucciones secuenciales) o paralela (instrucciones concurrentes).
El contenido de las 'cajas negras ó envolturas' está representado en la Definición de la Arquitectura. En ella está definida la función lógica que permite ejecutar la tarea deseada [Wak01].

2.10. FLUJO DE DISEÑO

Para realizar implementaciones en FPGA, es necesario seguir una serie de pasos definidos. A este conjunto de procedimientos se le denomina comúnmente como Flujo de Diseño. Para la realización de este trabajo, se siguió un procedimiento previamente esquematizado tal como se muestra en el diagrama de la figura 2.19, donde cada una de sus etapas serán descritas a continuación.
La primera fase corresponde al análisis del problema planteado en forma de diagrama de bloques y Análisis Jerárquico. Desde luego, este tipo de análisis es factible gracias a la característica de modularidad del lenguaje VHDL [Wak01].

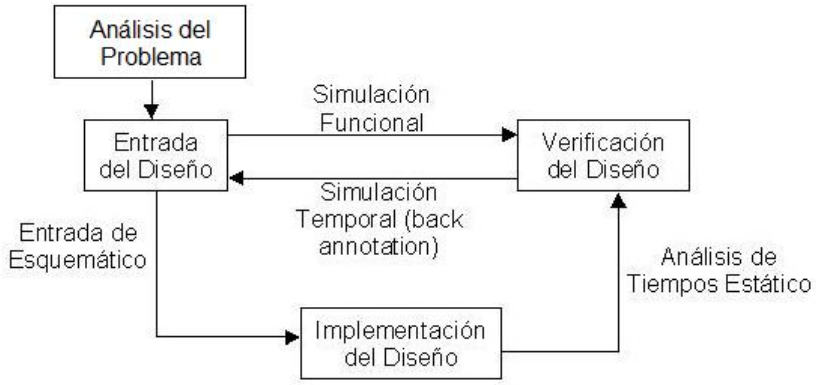


Figura 2.19: Flujo de Diseño

En la siguiente etapa se estructura el código en VHDL o se modifican los códigos previamente diseñados para sistemas similares al que se esté desarrollando. Cuando los módulos previamente diseñados tienen especificaciones idénticas al módulo en desarrollo, en el código solamente es necesario redefinir las conexiones de los puertos asociados al módulo.
El paso que se ejecuta a continuación es la Simulación HDL (Hardware Description Languaje) de los módulos estructurados en el proyecto. Para ello se construye un conjunto de señales de entrada denominado señales de prueba, que aplicadas al código, producen un comportamiento determinado en el puerto de salida. Por la observación

de dichas salidas se puede comprobar si el código realiza o no el proceso deseado. Las señales de prueba pueden ser diseñadas mediante una interfaz gráfica o programándolas en VHDL.
Posteriormente, se procede a ejecutar la etapa de Síntesis. En esta etapa, la descripción del hardware que se encuentra en el código se traslada a una función lógica expresada en forma de una 'lista de red'. Esto quiere decir que el programa escrito en VHDL se traslada a un archivo de datos comprensible por el software con el que se ejecuta el flujo de diseño. Esta función lógica o 'lista de red' posee las características propias de la tecnología a implementar (ya sea un ASIC, CPLD o una FPGA). Esto resulta de esta manera debido a que cada tecnología tiene estructuras con características propias. Por lo tanto un mismo código en VHDL generará funciones lógicas estructuradas de manera distinta según la tecnología a implementar al realizarse la Síntesis.
El paso realizado a continuación es la implementación de la lista de red. En este punto del proceso, se configura la arquitectura de la FPGA con base a sus elementos constitutivos fundamentales (CLBs e IOBs). En la FPGA la lista de red está estructurada para ser implementada con LTUs. La implementación se subdivide a su vez en 3 sub-etapas:

- Traducción
- Ajuste
- Ubicación y Enrutamiento

En la sub-etapa de Traducción se procede a obtener una lista de red, ahora con cierta información añadida. Tal información es suministrada a partir de varios archivos, en diferentes formatos. Entre los datos que se agregan están, por ejemplo: la asignación de las entradas y salidas a los pines del chip, restricciones referidas al tiempo máximo para la ejecución de ciertos procesos, restricciones en cuanto al uso de un área específica del dispositivo, etc.

La sub-etapa de Ajuste segmenta la función lógica en varias partes, donde cada parte de la función es asignada a los LUTs dependiendo de sus capacidades. De esta manera, cada LUT ejecuta sólo una parte de toda la función lógica. Seguidamente de la sub-etapa de Ubicación y Enrutamiento coloca los LUTs ya configurados, en los CLBs más adecuados para su implementación y conecta a los CLBs entre sí.

Como última etapa se realiza la Configuración. En ella, la arquitectura de la FPGA se traslada a un archivo binario que es utilizado para la configuración del Hardware. Su transmisión al chip, en este trabajo es específico, y se llevó a cabo por medio de una interfaz entre el puerto USB del PC y una entrada de la tarjeta asignada específicamente para el proceso de configuración.

2.11. CARACTERÍSTICAS BÁSICAS DEL SOFTWARE UTILIZADO PARA LA EJECUCIÓN DEL FLUJO DE DISEÑO

El flujo de diseño expuesto en la sección anterior fue aplicado con la ayuda del software que suministra la compañía que fabricó la FPGA Spartan-3E de Xilinx, Inc. El paquete tiene por nombre ISE y la versión utilizada fue la WEB PACK 12.1i para el sistema operativo Windows 7. La ejecución del Flujo de Diseño se lleva a cabo en un ambiente de trabajo que se observa en la figura 2.20

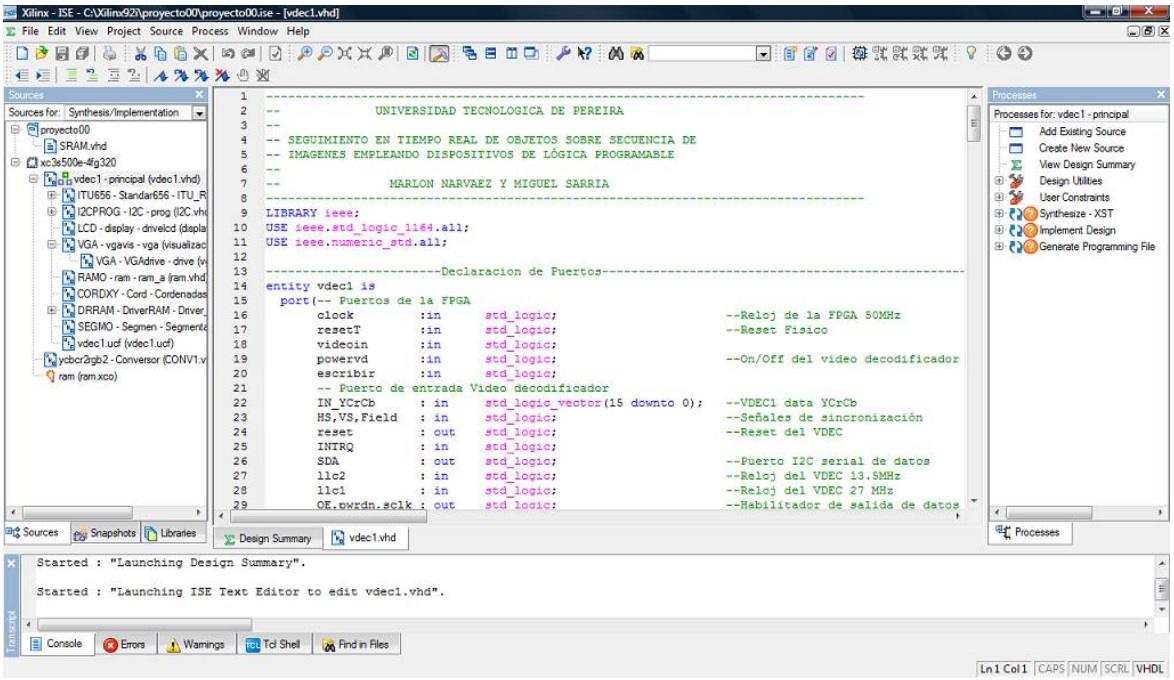


Figura 2.20: Ambiente de Trabajo del Software ISE

La creación de los códigos se realizan y modifican en una región denominada área de trabajo. En esta zona, aparte de manipularse los archivos fuentes en VHDL, también se pueden modificar archivos de diferentes formatos y fuentes. Por lo tanto es un área Multipropósito. La selección del archivo fuente con el que se trabaja en cada momento en específico, se hace en la denominada ventana de fuentes. Los procesos aplicados a los archivos seleccionados en la ventana de fuentes antes nombrada, se eligen y se configuran en una lista ubicada en la ventana de procesos. En esta ventana es donde se decide realizar las etapas de Simulación, Síntesis, Implementación (con sus respectivas sub-etapas)y finalmente la Configuración.
El software ofrece para cada una de las etapas del flujo de diseño, herramientas adicionales que pueden utilizarse para una adecuada ejecución según las necesidades:

- Dentro de la fase de Síntesis, se puede realizar, como un proceso individual, un chequeo de sintaxis del código a todo el programa ó solo a un subprograma.

- También es posible la visualización del código en forma gráfica (como esquema lógico de entidades inteconectadas). Además, se puede visualizar un esquema lógico de los recursos utilizados por la FPGA (LUT, Flip-Flops, etc).
- Implementación del diseño y generación del archivo de programación.

Finalmente en la última ventana se visualiza el estado de los procedimientos en ejecución mediante una consola. También se visualizan los errores y alertas que son emitidos en las situaciones en que existan.

3. MATERIALES Y MÉTODOS

En este capítulo se nombran los elementos utilizados en la tarea de validación de la metodología especificando el hardware utilizado para la implementación del sistema de seguimiento, y posteriormente se enuncian todas las etapas que conforman la metodología planteada para alcanzar los objetivos planteados en el proyecto.

3.1. MATERIALES

La lista de los componentes físicos usados en el proyecto son los siguientes:

- Tarjeta de Desarrollo Spartan-3E Starter Kit.
- Tarjeta Decodificadora de Video VDEC1.
- Cámara LB1000.
- Cable S-Video.
- Un computador personal marca Hewlett Packard modelo dv4-2116la.
- Monitor VGA marca QBEX.
- RAM interna de la FPGA.
- Pantalla LCD de la Tarjeta de Desarrollo.
- Switches deslizables de la Tarjeta de Desarrollo.
- Interfaz USB para programación de la Tarjeta de Desarrollo.
- Modulo DCM (Digital Clock Manager).
- Background (Superficie de Pruebas y Desarrollo).
- Objetos de Primer Plano para seguimiento.
- Computador de Mesa con un procesador Intel(R) Core(TM)2 Duo, CPU E450 de 2.20GHz con un 1GB de memoria RAM.
- Tarjeta Aceleradora de Video NVIDIA GeForce 7300 GS para PC.
- Tarjeta de Video LR37 rev:B con una entrada S-Video y dos de Video Compuesto (CVBS).

- Cámara Panasonic DMC-FS6.

El software utilizado en el proyecto es el siguiente:

- Xilinx ISE Design Suite 12.1 en lenguaje VHDL
- Matlab 7.9.0.529 (R2009B)
- Software de Procesamiento de Video.
- Software de Estadística para Análisis de Datos.

3.2. MÉTODOS

El proyecto se dividió en 9 etapas:

- Revisión, recopilación y análisis de literatura relacionada con el tema de estudio.
- Verificación de los estándares.
- Pruebas con los estándares.
- Como realizar asignación a memoria.
- Solución de problemas de visualización.
- Mejora de la imagen.
- Diseño del Sistema de iluminación para control de variables a nivel de laboratorio. Inmunidad a la luz ambiente y control de distancia de la cámara.
- Pruebas de Laboratorio.
- Análisis de Resultados.

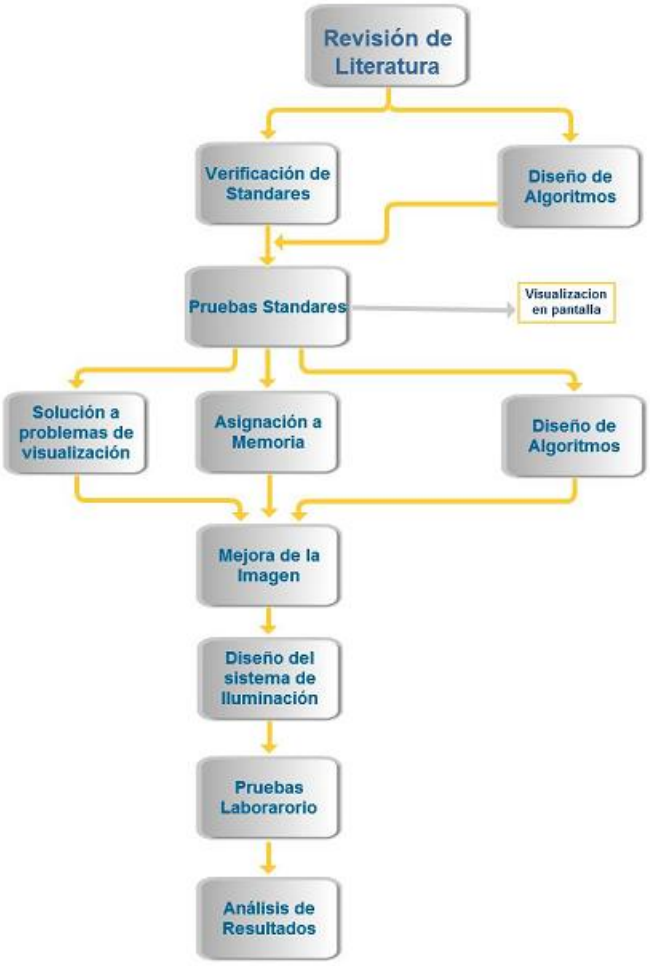


Figura 3.1: Etapas del Procedimiento Metodológico

A continuación se muestran los procesos que hacen parte de la metodología desarrollada:

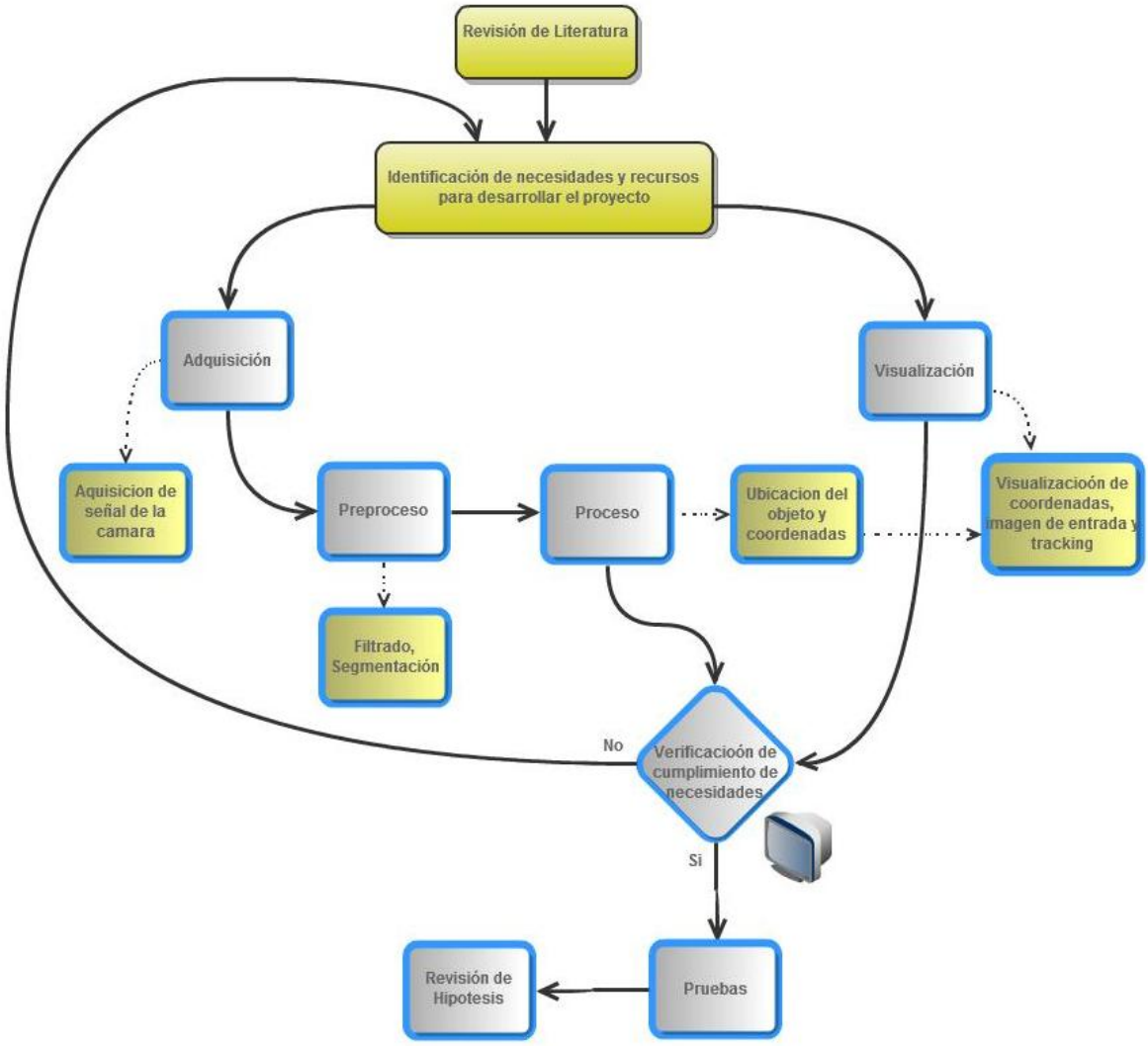


Figura 3.2: Procesos de la Metodología

El tipo de investigación realizada es Experimental. El desarrollo e implementación del seguimiento en tiempo real de objetos sobre secuencia de imágenes utilizando dispositivos de lógica programable en condiciones de laboratorio se llevará a cabo con los siguientes procesos que permitirán alcanzar los objetivos planteados:
Se hizo una recopilación y un análisis de la literatura relacionada con el tema de sistemas digitales, procesamiento de imágenes y video lo cual nos brindaría las herramientas necesarias para la identificación de las necesidades y recursos para el desarrollo el proyecto. Luego de identificar las necesidades se subdivide el desarrollo del proyecto en cuatro bloques de tareas específicas (Adquisición, Preproceso, Procesamiento y Visualización en Pantalla). Para el proceso de adquisición, con el cual se captura una señal de video se debe realizar un estudio de interfaces que nos permitan manejar el estándar de video entrante en un espacio de color que pueda ser procesado en

el dispositivo lógico programable. En el preproceso se deberán aplicar técnicas de pre-procesamiento de la señal de video adquirida, con el fin de realizar el filtrado y la segmentación de la señal de video adquirido para su posterior procesamiento.En el procesamiento se debe aplicar una técnica de seguimiento de objetos para encontrar la ubicación del objeto realizando el tracking (seguimiento) del mismo proporcionando las coordenadas de ubicación en un espacio bidimensional, teniendo en cuenta que la técnica desarrollada no requiera un alto costo computacional y ofrezca una velocidad de respuesta apropiada por parte del sistema.

Después de terminar el desarrollo de los bloques de tareas específicas descritos anteriormente se validará el algoritmo implementado con visualización en pantalla de las imágenes procesadas del seguimiento del objeto y sus coordenadas se realizará una prueba de la metodología planteada en la cual si no se cumplen los objetivos se deberá ejecutar una realimentación a la metodología para encontrar mejores soluciones a los problemas planteados en el proyecto. En el desarrollo del proyecto se pueden realizar bloques de tareas de manera concurrente para optimizar el tiempo de desarrollo del mismo, ejemplo de esto es realizar el proceso de adquisición y al mismo tiempo el proceso de visualización en pantalla.

3.3. DESARROLLO DEL PROYECTO

Para realizar las pruebas en condiciones de laboratorio se construyó un background de color negro en el cual se colocaron objetos de primer plano de varios colores que se desplazaban por esta superficie con imanes para a través de una cámara instalada en un tripode realizar el seguimiento de los mismos como se puede observar en la figura 3.3.

Figura 3.3: Background y Cámara en Tripode

Al principio se presentaron muchos inconvenientes con la imagen que se adquiría a través de la Tarjeta Decodificadora de Video VDEC1 pues no conociamos bien su funcionamiento, y por lo tanto se necesitó profundizar más en el estudio y forma de implementación de los estándares con lo cual se desarrollaron los algoritmos para el manejo del estándar ITU-R BT.656. En la figura 3.4 se pueden observar algunas pruebas de la adquisición de líneas de video que obteníamos a través de la cámara y la Tarjeta Decodificadora determinando el número de líneas y píxeles del estándar de video antes mencionado.

Figura 3.4: Pruebas del Estándar de Video

En la anterior figura también se muestra parte de la visualización, teniendo un espacio de pantalla para mostrar la imagen, un espacio para el tracking y se implementaron los algoritmos para el reconocimiento de coordenadas en pantalla que inicialmente fueron usadas para establecer el número de líneas y píxeles máximos dados por el estándar de

video. También se hizo uso de un código de Xilinx para la validación del formato de video entregado por el video decodificador de donde se usan las señales de sincronización VS, HS y Field para realizar un contador determinando el video entrante NTSC ó PAL. El código implementado es el siguiente:

```
126  -- NTSC tiene 19 lineas, PAL tiene 24.
127  process(PB_R, llc, V_rising, H_rising, V_falling, format_count_max)
128  begin
129     if PB_R = '0' then
130        format_count      <= 0;
131        format_count_max  <= 19;
132     elsif llc'event and llc = '1' then
133        if V_rising = '1' and H_rising = '1' then
134           format_count      <= 1;
135           format_count_max  <= format_count_max;
136        elsif V_falling = '1' and H_rising = '1' then
137           format_count      <= 1;
138           format_count_max  <= format_count;
139        elsif (not(V_rising))= '1' and H_rising = '1' then
140           format_count      <= format_count + 1;
141           format_count_max   <= format_count_max;
142        else
143           format_count      <= format_count;
144           format_count_max  <= format_count_max;
145        end if;
146     end if;
147  if format_count_max = 19 then
148  NTSC_in <= '1';
149  else
150  NTSC_in <= '0';
151  end if;
152  end process;
```

Figura 3.5: Validación del Formato de Video

Para poder realizar el preprocesamiento fue necesario primero identificar la forma como se iba a almacenar la imagen. En primer lugar se intentó implementar una memoria RAM internamente en la FPGA, pero de esta forma la imagen adquirida representaba un porcentaje alto de silicios de la FPGA.

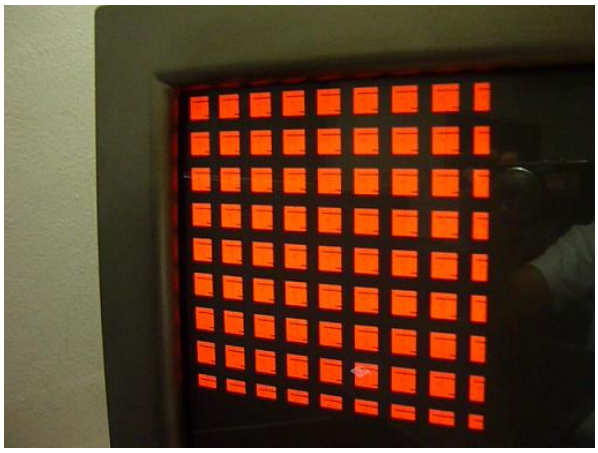


Figura 3.6: Implementación con RAM Interna

Luego estudiamos la viabilidad de realizarlo en una memoria DDR SDRAM (MT46V32M16) ó en la memoria Intel StrataFlash PROM incluidas dentro de la Tarjeta de Desarrollo pero su capacidad de almacenamiento era limitado. Dentro del estudio de viabilidad se encontró una herramienta que está incluida en el Xilinx ISE Design Suite 12.1 llamada Block Memory Generator v3.3 con la cual se puede implementar una memoria RAM de puerto dual por lo que seleccionamos esta ya que nos ofrecía un espacio de memoria para la imagen y el tracking. En la figura 3.7 se puede observar la prueba de la asignación de datos a memoria.

Figura 3.7: Asignación de Datos a Memoria con RAM Interna

Posteriormente se realiza la adquisición de píxeles y líneas de video para probar una imagen de entrada pero se adquiere con interferencia y para solucionarlo se cambió dentro de la implementación del algoritmo el código matemático a implementaciones lógico matemáticas mejorando el proceso de adquisición. Esto se muestra en la figura 3.8.

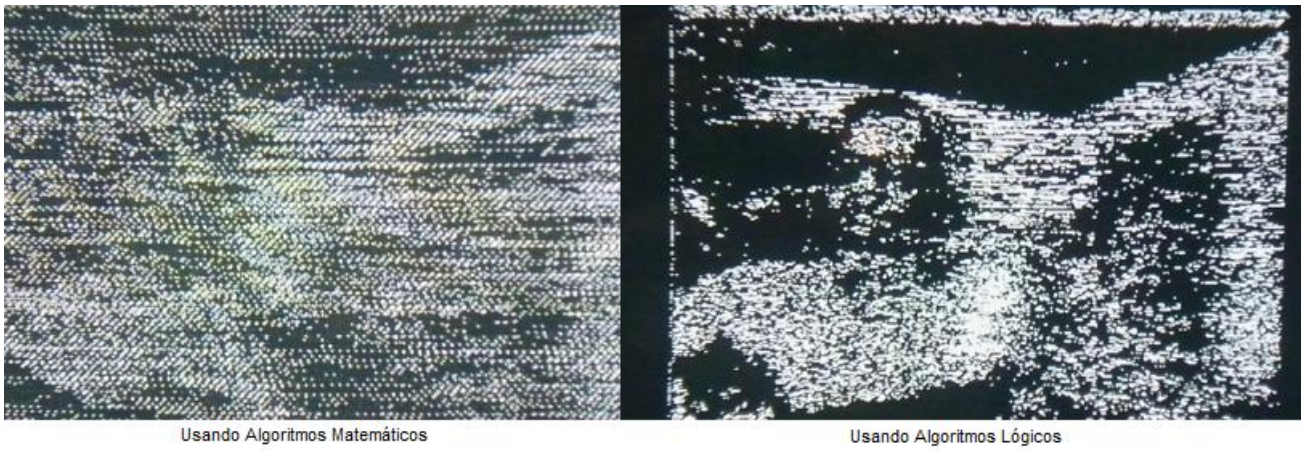


Figura 3.8: Problema con Algoritmos

Se ve necesario mejorar la segmentación por lo que se utilizaron filtros para la imagen de entrada con el fin de eliminar ruido como se muestra en la figura 3.9:

Figura 3.9: Filtro de Video Entrante

Se puede observar en la anterior figura que aún con el filtrado se presentaba ruido en la imagen que deseábamos segmentar por lo que se tuvo que realizar modificaciones a los algoritmos implementados intentando realizar la mayor cantidad de procesos secuenciales con máquinas de estados y reorganizando los algoritmos en procesos independientes unidos mediante componentes, señales y banderas. Se usaron señales de habilitación o punteros para procesos no concurrentes. Con estos cambios se logro mejorar el preprocesamiento de la imagen como se muestra en la figura 3.10:

Figura 3.10: Mejora de la Imagen por cambios en algoritmos

También se tuvo problemas debido a factores como el ruido que genera la luz ambiente y la luz artificial de los bombillos pues no lograbamos controlar su incidencia en la cámara LB1000 utilizada para el seguimiento de los objetos. Para contrarrestar el inconveniente del efecto del ruido se fue modificando el código implementado, y después de varios cambios y mejoras no se logro quitar todo el ruido por lo que se opto por mejorar las condiciones ambientales aislando el sistema utilizando una fuente de luz controlable y de esta forma mejorando no solo el ruido sino también evitando la pérdida de la información en las características visuales por variables como el nivel de iluminación y las sombras.
Para aislar la condiciones exteriores y tener control sobre la iluminación se procedió a construir una caja ó cubo con la iluminación de manera paralela al background. El color interno del sistema de aislamiento debe ser blanco para difuminar la iluminación sobre todo el Background. Con el fin de generar la iluminación se instalaron 70 leds de luz blanca de alto brillo de 5mm colocados de forma paralela al background distribuidos alrededor de su périmetro (rectángulo). Para el sistema de alimentación se utilizo una fuente de poder DC de 12 voltios de salida nominal a 6 Amperios. En la figura 3.11 se puede observar el montaje realizado del sistema de iluminación:

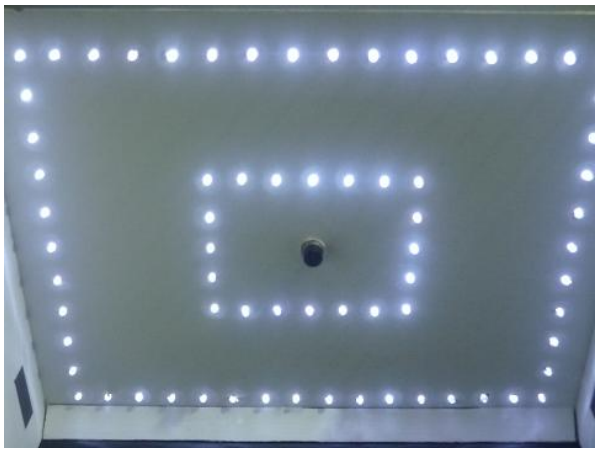


Figura 3.11: Iluminación con Leds

Al realizar pruebas con los leds se encontró el problema de que la luz generada por estos elementos no era suficiente ni la más apropiada para lo que se necesitaba pues se estaban formando sombras en la superficie de pruebas como se observa en los recuadros rojos de la figura 3.12:

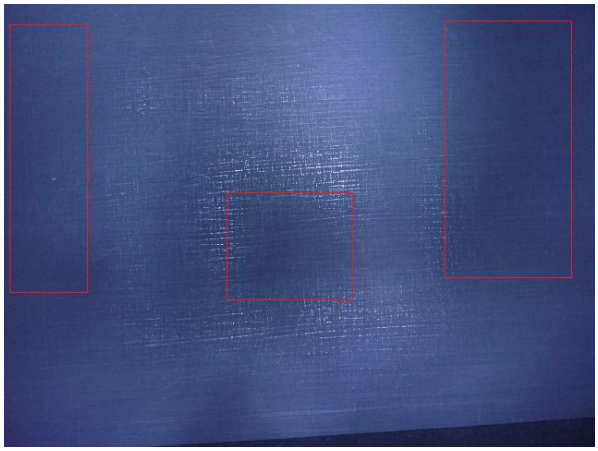


Figura 3.12: Iluminación del Background con Leds

Para corregir este problema se utilizó una superficie refrectiva interna, pero el inconveniente que se tuvo fue la acumulación de brillo en el centro del Background. Al conectar todos los elementos del sistema, descargar el algoritmo a la tarjeta FPGA y colocar la cubierta de la caja se observaban partes blancas que no eran parte de la segmentación de los objetos. Debido a que esta fuente de luz no sirvió después utilizamos un par de lámparas de luz blanca de 8W con una alimentación de 110-130V las cuales instalamos también en el fondo donde se instalaron los leds. Las Lámparas se instalaron de manera vertical pues se probaron de manera horizontal y se presentaban los mismos inconvenientes que con los leds.

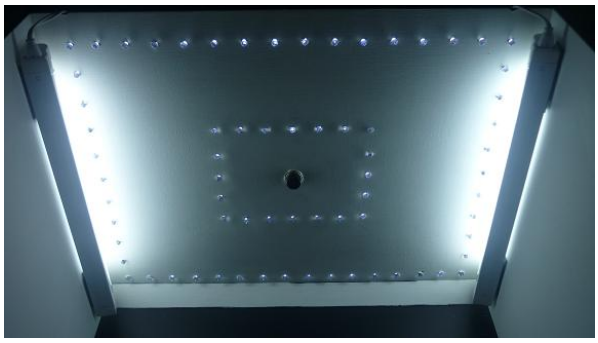


Figura 3.13: Iluminación con Leds y Lámparas

Al realizar este tipo de iluminación y para que el sistema de iluminación fuera homogeneo se usaron superficies de color blanco paralelas a la pantalla y de color negro para las superficies perpendiculares como se observa en la figura 3.13. Para obtener un video de salida más definido se modificó la posición de los lentes de la cámara LB1000 como se muestra en la figura 3.14.

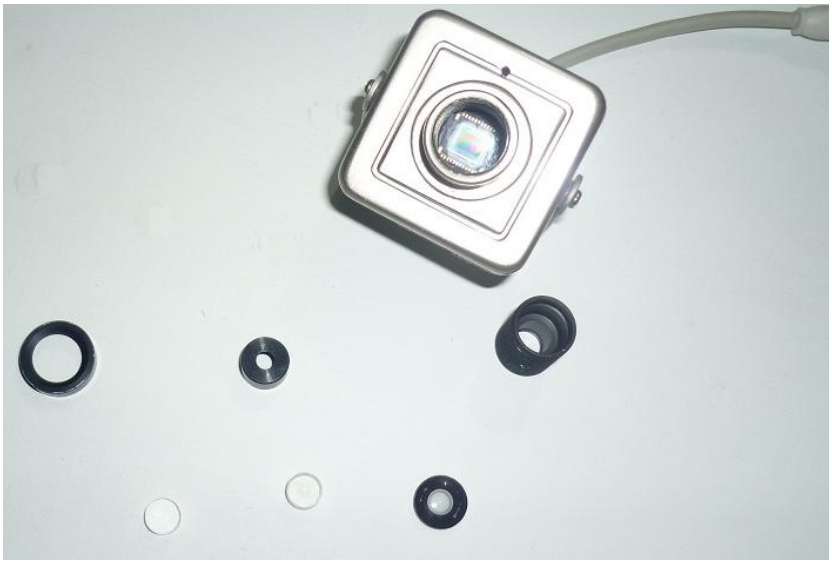


Figura 3.14: Modificación Lentes de la Cámara

3.3.1. IMPLEMENTACIÓN DEL PROGRAMA EN VHDL El punto de partida de nuestro trabajo fue la investigación acerca de los estándares de video analógico, especialmente el estándar NTSC el cual define la señal de video con una taza de refresco de 60 medio-cuadros (entrelazados) por segundo. Cada cuadro contiene 525 líneas y pueden contener más de 16 millones de colores. También se investigó como adquirir y procesar el estándar NTSC usando una FPGA. Inicialmente tratamos de implementar un decodificador de video el cual extrajera la información de la imagen contenida en la señal NTSC por medio de un conversor análogo digital LTC1407A

(incluido en la Tarjeta Spartan-3E) y la FPGA. Debido a que una gran parte de los recursos de la FPGA tenían que ser destinados solo para realizar esta implementación decidimos buscar otra alternativa que pudiera realizar la decodificación sin tener que usar la tarjeta. La forma de solucionarlo fue utilizando una Tarjeta Decodificadora de Video VDEC1 la cual utiliza un chip ADV7183B que puede digitalizar una señal de video NTSC, PAL y SECAM. Este chip detecta automáticamente el estándar analógico bandabase de señales de televisión y las digitaliza con tres ADC´s de 10 bits de 54 MHz. El video decodificador se conecta a la FPGA a través del puerto de expansión Hirose de 100 pines.

La implementación del proyecto fue ejecutado a través del programa Xilinx ISE Design Suite 12.1 en lenguaje VHDL y su implementación física con una tarjeta Spartan-3E Starter Kit modelo XC3S500E, una tarjeta decodificadora de video VDEC1, una cámara LB1000 y con visualización en un monitor VGA. La descripción global del proyecto es adquirir una secuencia de imágenes a través de una cámara, la cual controla la cantidad y el tiempo de luz del entorno la cual es sensada por un sensor fotosensible CMOS que trasforma la luz en señales eléctricas las cuales son codificadas en un estándar de video de televisión. La tarjeta VDEC1 recibe esta señal y la digitaliza en el estándar ITU-R BT.656 que es un estándar de codificación de un formato de video digital YCbCr 4:2:2, extraemos la información de video, realizamos el procesamiento de la imagen encontrando el objeto con su respectiva posición (Coordenada X,Y) y por último lo mostramos en pantalla.

La lista de los módulos de programa desarrollados en el proyecto son los siguientes:

- Módulo Estándar ITU-R BT.656.
- Módulo Programación I2C.
- Módulo Manejo DCM y LCD.
- Módulo Visualización VGA.
- Módulo Inicialización RAM.
- Módulo Preprocesamiento y Procesamiento.
- Módulo Tracking.
- Módulo de Segmentación.

Los procesos son concurrentes entre sí, y en los que se requiera esperar otro proceso se utilizan banderas de espera de programa. A continuación se describen los algoritmos usados en cada Módulo para el desarrollo del proyecto.

MÓDULO ESTÁNDAR ITU-R BT.656 Como lo describimos en la sección 3.4 este estándar de codificación de video digital consta de una sección de códigos AV (Active Video), Borrado de Datos durante el intervalo Blanco Vertical (VBI) y los datos de información. En nuestra implementación se utilizó una interfaz de 8 bits de salida (Cb/Y/Cr/Y) intercalados, el código de AV se define como FF/00/00/AV con AV como la palabra de trasmisión que contiene la información acerca de H, V y F [Inc05a].

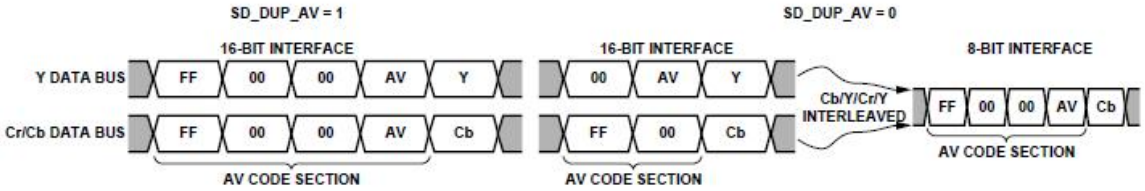


Figura 3.15: Formato de Salida Digital.

El código AV se usa para poder determinar la ubicación de video que se está transmitiendo. Además se usa para la sincronización, y a través de este podemos determinar el inicio o el fin de una línea del video, el inicio del video activo y la terminación del video. La forma de usar este código empieza con el inicio de video activo, después se trasmite una línea de video, seguido se termina esta línea de video y por último se transmite una trama de audio, EDH (Error Detection and Handling) u otros datos que se requieran transmitir. Después se vuelve a repetir el mismo ciclo. El código AV se compone de 4 palabras que son FF/00/00/AV, donde AV es F/V/H/P3/P2/P1/P0/0/0 siendo F el bit más significativo. Para encontrar este código se implementó una serie de registros los cuales se actualizan cuando se presenta un evento en la señal de reloj (LLC1) del videodecodificador y cuando este es igual a uno. Las señales de sincronización HS,VS y Field también son entregadas directamente por el videodecodificador por lo que no es necesario extraerlas del código AV. El siguiente código representa los registros en VHDL [Inc05a]:

```
process(llc,PB_R) --Registros de memoria de 5 pixel
begin
    if llc'event and llc = '1' then
      if(PB_R = '0') then
            reg1 <= "0000000000";
            reg2 <= "0000000000";
            reg3 <= "0000000000";
            reg4 <= "0000000000";
            reg5 <= "0000000000";
         else
            reg1 <= CYCrCb422;      -- 1 clock delay
            reg2 <= reg1;           -- 2 clock delay
            reg3 <= reg2;           -- 3 clock delay
            reg4 <= reg3;           -- 4 clock delay
            reg5 <= reg4;           -- 5 clock delay
       end if;
    end if;
end process;
```

Figura 3.16: Registros de Memoria.

Teniendo 5 datos podemos determinar cuando se presenta un código EAV (Final de Video Activo) ó SAV (Inicio de Video Activo). A este evento lo llamaremos TRS cada vez que reg2 = 00, reg3 = 00 y reg4 = FF entonces TRS será igual a uno. Esto se puede hacer ya que 00 y FF son valores reservados. La imagen de salida está limitada al siguiente rango para la luminancia de $1 \leq Y \leq 254$ y para la crominancia de $1 \leq Cr \leq 254$ ó $1 \leq Cb \leq 254$.

```
TRSV <= not(reg2(9 downto 2)) and (not(reg3(9 downto 2))) and (reg4(9 downto 2));

process(TRSV) --Comienzo de video activo
begin
if (TRSV = "11111111") then
TRS <= '1';
else
TRS <= '0';
end if;
end process;
```

Figura 3.17: Código de Palabra EAV y SAV.

Hasta el momento se ha determinado cuando se presenta un código de sincronización ya sea para EAV ó SAV.

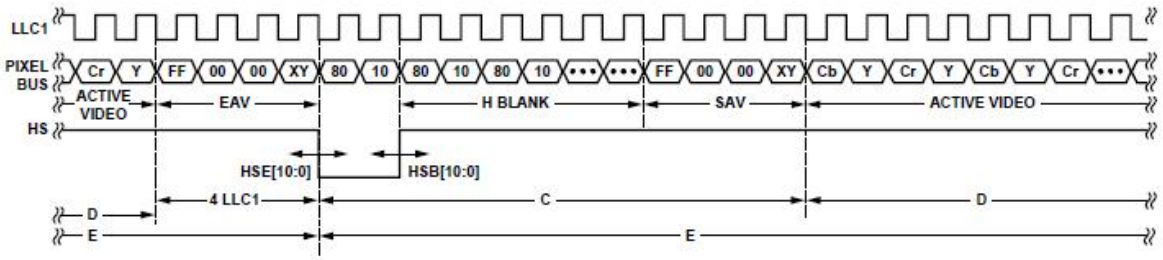


Figura 3.18: Sincronización Horizontal.

Como se observa en la figura 3.18 la transmisión de video activo ocurre cuando la señal de sincronización horizontal es igual a uno, HS es igual a cero luego del código EAV y vuelve a ser igual a uno luego de dos pulsos de reloj para dar comienzo al blanqueo horizontal. La siguiente tabla nos muestra el número de datos por sincronización horizontal para cada estándar.

	Characteristic				
Standard	**HS Begin Adjust (HSB[10:0]) (Default)**	**HS End Adjust (HSE[10:0]) (Default)**	**HS to Active Video (LLC1 Clock Cycles) (C in Figure 20) (Default)**	**Active Video Samples/Line (D in Figure 20)**	**Total LLC1 Clock Cycles (E in Figure 20)**
NTSC	00000000010b	00000000000b	272	720Y + 720C = 1440	1716
NTSC Square Pixel	00000000010b	00000000000b	276	640Y + 640C = 1280	1560
PAL	00000000010b	00000000000b	284	720Y + 720C = 1440	1728

Figura 3.19: Datos para Sincronización Horizontal.

Lo anterior es solo para un Sincronismo Horizontal. Para la transmisión total de vídeo tendríamos 525 líneas de video, como se puede apreciar en la figura 3.20:

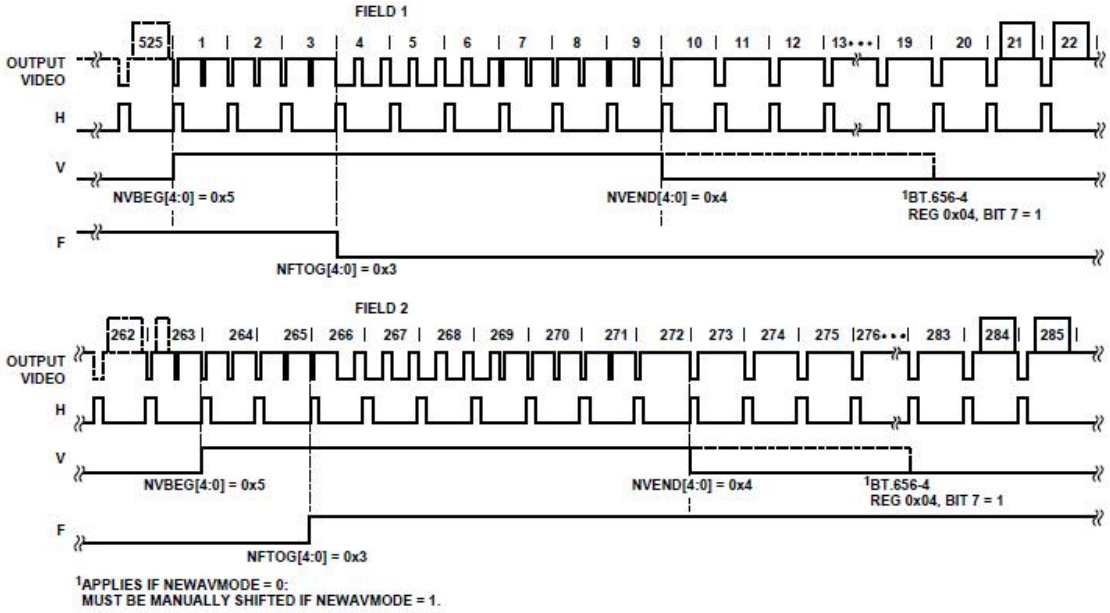
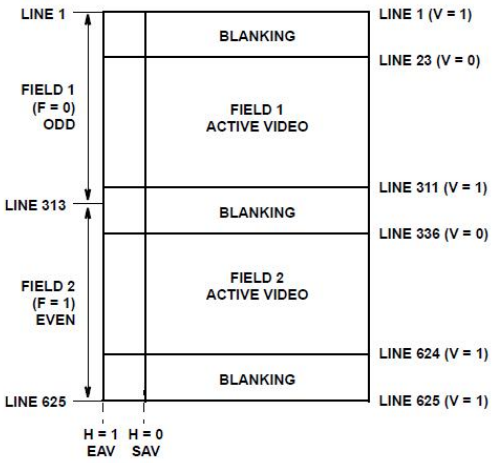


Figura 3.20: Transmisión de Video YCrCb a una Entrada de Video NTSC.

En la tabla 3.21 se describe de una manera más fácil el comportamiento de HS, VS y Field.



LINE NUMBER	F	V	H (EAV)	H (SAV)
1-22	0	1	1	0
23-310	0	0	1	0
311-312	0	1	1	0
313-335	1	1	1	0
336-623	1	0	1	0
624-625	1	1	1	0

Figura 3.21: Comportamiento de HS, VS y Field.

La forma más sencilla de implementar un sistema que determine cada una de las transiciones de video es una máquina de estados.

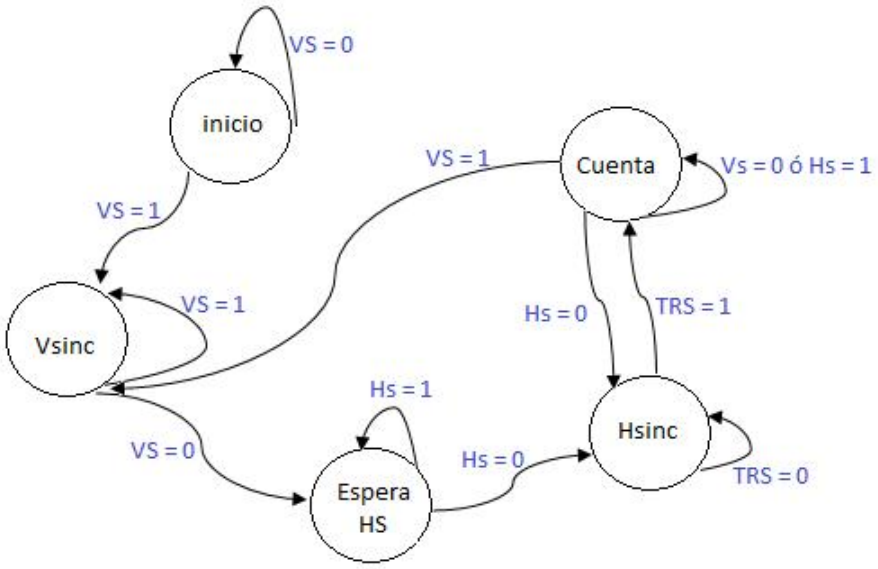


Figura 3.22: Máquina de Estados para Proceso de Sincronización del Video.

En la acción de los estados tendremos dos variables auxiliares reset-lineas e iniciar-cuenta. De esta forma podemos definir el inicio de la imagen y el inicio de una línea de video. Cuando estamos en el estado de cuenta estamos recibiendo una nueva línea de video por lo que iniciar-cuenta tendrá un valor lógico de uno y en otro estado tendrá un valor lógico de cero. Para reset-lineas esta tiene un valor lógico de uno cuando está en el estado Vsinc y para los demás será cero.
Para recibir un pixel de video dentro de esta máquina de estado se tiene el siguiente programa:

```
22  process(llc,PB_R,StartCount)
23  begin
24     if(PB_R = '0') then
25          value <= "00";
26     elsif(rising_edge(llc)) then
27           if(StartCount = '1') then
28            value <= value + 1;
29           else
30            value <= "00";
31           end if;
32     end if;
33  end process;
34
35  load <= value(1) and value(0);
```

Figura 3.23: Contador para un Píxel de Video.

De esta forma load es una bandera con la que podemos adquirir la información de un píxel (Y/Cr/Y/Cb) y esto ocurre cuando iniciar-cuenta que en este caso lo llamamos StartCount es igual a uno. La información de los datos de entrada está dada en formato YCrCb 4:2:2 por lo que antes de convertir los datos de YCrCb a RGB, el YCrCb 4:2:2 debe ser convertido a YCrCb 4:4:4.
La señal de video se comprime si se pasa de 4:4:4 a 4:2:2, a lo cual llamaremos submuestreo, que consiste en reducir la información de color preservando intacta la luminosidad, y esto se puede realizar ya que el ojo humano, es más sensible a la luminosidad (cantidad de luz por píxel) que al color. Si centramos la compresión en el color preservando la luminosidad, conseguiremos en primer lugar la reducción del tamaño de la captura sin afectar la calidad, reducir la cantidad de datos al realizar el almacenamiento ya que se reduce a la mitad el número de componentes diferentes, Cr y Cb, y reducir el ancho de banda en la transmisión.

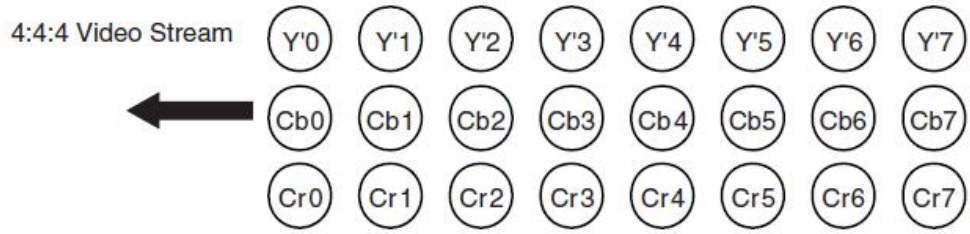


Figura 3.24: Transmisión de Video Digital en 4:4:4.

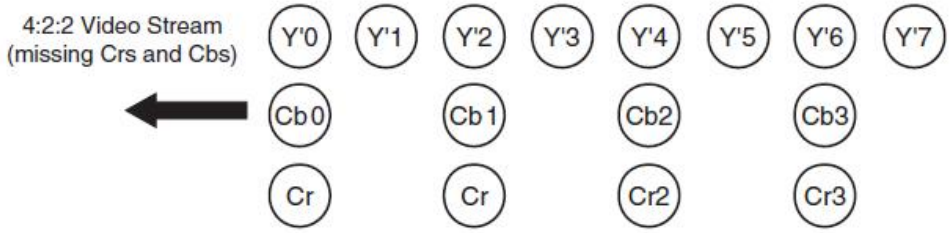


Figura 3.25: Transmisión de Video Digital en 4:2:2.

Ahora para realizar la decodificación de video y obtener los valores faltantes de crominancia, es decir convertir de nuevo al formato 4:4:4, se deben interpolar los valores disponibles. Uno de los objetivos de la conversión a un formato análogo es calcular los valores faltantes sin producir efectos indeseables en este proceso. Debemos tener en cuenta que las palabras de datos de video se transmiten a una frecuencia de 27 MHz en el siguiente orden: Cb0, Y'0, Cr0, Y'1, CB1, Y'2, CR1, Y'3, CB2, Y'4, Cr2, Y'5...
A una frecuencia de muestreo de 13,5 MHz un videodecodificador es compatible con señales de video NTSC y PAL. A esta frecuencia, las tramas de video NTSC contiene 858 muestras (720 activas) y con 525 líneas en total (485 líneas activas) y las tramas

PAL contiene 864 muestras (720 activas) y con 625 líneas en total (576 líneas activas). Dentro de los métodos para realizar la conversión del muestreo hablaremos de tres métodos:
Primer Método: Replicación de Cr y Cb:
Si el objetivo de diseño no demanda una alta calidad de video, los datos desaparecidos Cr's y Cb's se puede replicar de las anteriores, suponiendo que no hay datos inexactos de muestreo en la entrada. La salida no debe superar nunca los límites de frecuencia. Se recomienda para ejecutar este método realizar un promedio entre dos datos de Cr y Cb disponibles. La ecuación para realizar este promedio es:

$$Cb[i] = \frac{Cb[i-1] + Cb[i+1]}{2}$$

Esta ecuación es implementada como se muestra en la figura 3.26:

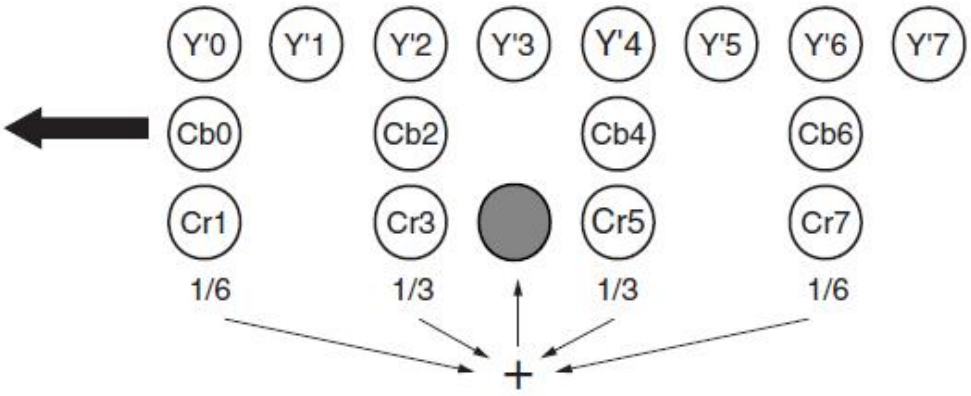


Figura 3.26: Replicación de Cr y Cb

Segundo Método: Filtro FIR Paralelo:
Cuando es necesario una calidad de video superior se requiere utilizar un filtro FIR. Se debe tener en cuenta que la respuesta del filtro FIR debe ser unitaria, sino entonces el resultado debe ampliarse en forma adecuada. La siguiente ecuación tiene coeficientes que suman 256 por lo que el resultado debe dividirse por 256 para ser unitario.

$$Cb[i] = (160(Cb[i-1] + Cb[i+1]) - 48(Cb[i-3] + Cb[i+3]) + 24(Cb[i-5] + Cb[i+5])$$
$$-12(Cb[i-7] + Cb[i+7]) + 6(Cb[i-9] + Cb[i+9]) - 42(Cb[i-11] + Cb[i+11]))/256$$

Tercer Método: Filtro FIR Paralelo de Xilinx:
El filtro realizado por los laboratorios de Xilinx cumple con los requisitos de especificación en calidad de video. Se aprovecha la simetría, donde los términos 'similares' en las ecuaciones se resumen antes de la multiplicación correspondiente [Inc01].

$$Cb[i] = (-4(Cb[i-23]+Cb[i+23])+6(Cb[i-21]+Cb[i+21])-12(Cb[i-19]+Cb[i+19])$$
$$+20(Cb[i-17]+Cb[i+17])-32(Cb[i-15]+Cb[i+15])+48(Cb[i-13]+Cb[i+13])$$
$$-70(Cb[i-11]+Cb[i+11])+104(Cb[i-9]+Cb[i+9])-152(Cb[i-7]+Cb[i+7])$$
$$+236(Cb[i-5]+Cb[i+5])-420(Cb[i-3]+Cb[i+3])+1300(Cb[i-1]+Cb[i+1]))/2048$$

Esta ecuación es implementada como se muestra en la figura 3.27:

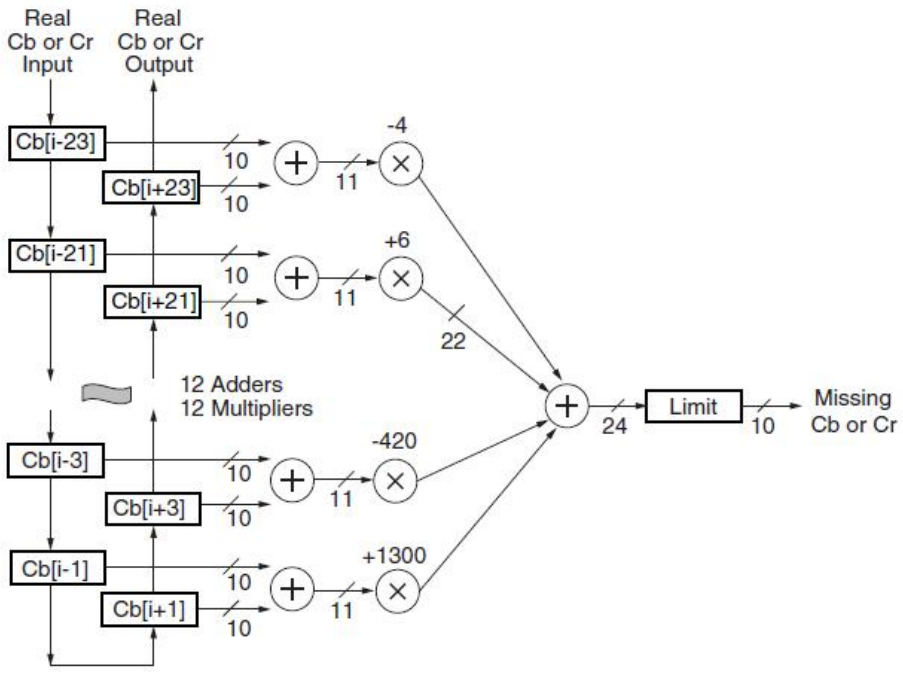


Figura 3.27: Implementación Filtro FIR de 4:2:2 a 4:4:4

En nuestro trabajo se implementó el Método de Replicación de Cr y Cb. Para aplicaciones donde se deba tener una alta fidelidad de video es recomendable realizarlo con filtros FIR paralelos.
Teniendo la señal de video YCrCb 4:4:4 podemos realizar la conversión de video a RGB. El espacio de color RGB es una definición simple y robusta de color utilizado en los

sistemas informáticos y de Internet para ayudar a asegurar que un color está asignado correctamente de una plataforma a otra sin pérdida significativa de información de color. RGB utiliza tres componentes numéricas para representar un color. Este espacio de color puede ser pensado como un sistema de coordenadas tridimensionales del sistema cuyos ejes corresponden a los tres componentes, R (rojo), G (verde), y B (azul). RGB es un sistema de color aditivo. Los tres colores primarios rojo, verde y azul se agregan para formar el color deseado. Cada componente tiene un rango de 0 a 255, donde 000 es color negro y 111 es color blanco.
Como habíamos visto YCrCb es un espacio de color que fue desarrollado como parte de la recomendación ITU-R BT.656 en todo el mundo como un estándar de video digital por componentes y utilizado en las transmisiones de video de televisión. YCrCb es una escala y offset de la versión del espacio de color YUV, donde Y representa la luminancia (o brillo), U representa el color, y V representa el valor de saturación. En el espacio de color RGB se divide en una parte de luminancia (Y) y dos partes de crominancia (Cb y Cr).
Se ha determinado que del 60 al 70 por ciento de la luminancia o brillo se encuentra en el color 'verde'. En la crominancia la parte Cb y Cr, la información de brillo se puede extraer de los colores azul y rojo. El resultado es que Cb y Cr proporciona el tono y la saturación de la información del color y la 'Y' proporciona la información de brillo del color. Y está definido que para tener un rango de 16 a 235 y Cb y Cr se tiene un rango de 16 a 240 con 128 igual a cero. Debido a que el ojo es menos sensible a Cb y Cr, no se tendrá que transmitir Cb y Cr a la misma velocidad como Y. Menos almacenamiento y ancho de banda son necesitados, lo que resulta en costos de diseño bien reducidos. El espacio de color YCrCb se convierte en el espacio de color RGB con las siguientes ecuaciones cuando se usan 8 bits:

$$R' = 1{,}164(Y' - 16) + 1{,}596(Cr - 128)$$
$$G' = 1{,}164(Y' - 16) - 0{,}813(Cr - 128) - 0{,}392(Cb - 128)$$
$$B' = 1{,}164(Y' - 16) + 2{,}017(Cb - 128)$$

Para las entradas de 10-bits, las ecuaciones son:

$$R' = 1{,}164(Y' - 64) + 1{,}596(Cr - 512)$$
$$G' = 1{,}164(Y' - 64) - 0{,}813(Cr - 512) - 0{,}392(Cb - 512)$$
$$B' = 1{,}164(Y' - 64) + 2{,}017(Cb - 512)$$

En nuestro caso usamos una entrada de 10 bits. En el código implementado tenemos:

```
27  P1:process (clk,rst)
28    begin
29     if (rst = '0') then
30        Y_reg <= "0000000000"; CR_reg <= "0000000000";CB_reg <= "0000000000";
31     elsif (rising_edge (clk)) then
32        Y_reg  <= Y;
33          CR_reg <= Cr;
34          CB_reg <= Cb;
35     end if;
36  end process P1;
37
38
39   Y_int1 <= (('0' & Y_reg)  - "00001000000"); -- (Y_reg  - 64 )
40   CR_int1<= (('0' & CR_reg) - "01000000000"); -- (Cr_reg - 512)
41   CB_int1<= (('0' & CB_reg) - "01000000000"); -- (Cb_reg - 512)
42
43  P2:process (clk,rst)
44    begin
45     if (rst = '0') then
46        X_int  <= (others => '0');
47        A_int  <= (others => '0');
48        B1_int <= (others => '0');
49        B2_int <= (others => '0');
50        C_int  <= (others => '0');
51     elsif  (rising_edge (clk)) then
52        X_int  <= ("00100101001" * (Y_int1));     --1.164(Y_reg  - 64 )
53        A_int  <= ("00110011000" * (CR_int1));        --1.596(Cr_reg - 512)
54        B1_int <= ("00011000000" * (CR_int1));        --0.813(Cr_reg - 512)
55        B2_int <= ("00011000001" * (CB_int1));        --0.392(Cb_reg - 512)
56        C_int  <= ("01000000100" * (CB_int1));        --2.017(Cb_reg - 512)
57     end if;
58  end process P2;
60  P3:process (clk,rst)
61    begin
62     if (rst = '0') then
63        R_int <= (others => '0');
64        G_int <= (others => '0');
65        B_int <= (others => '0');
66     elsif  (rising_edge (clk)) then
67        R_int <= X_int + A_int;
68        G_int <= X_int - B1_int - B2_int;
69        B_int <= X_int + C_int;
70     end if;
71  end process P3;
```

Figura 3.28: Conversión YCrCb 4:4:4 a RGB.

MÓDULO PROGRAMACIÓN I2C La comunicación I2C se utilizo para programar el videodecodificador ADV7183B. El protocolo de comunicación serial I2C diseñado por Phillips, es un bus de comunicaciones en serie. Su nombre viene de Inter-Integrated Circuit (Circuitos Inter-Integrados). La versión 1.0 data del año 1992 y la versión 2.1 del año 2000. Es un bus muy usado en la industria, principalmente para comunicar circuitos integrados entre sí que normalmente residen en un mismo circuito impreso. Las características del bus I2C son:

- Se necesitan solamente dos líneas, una de datos (SDA) y una de reloj (SCLK).

- Cada dispositivo conectado al bus tiene un código de dirección seleccionable mediante software. Hay permanentemente una relación master/slave entre el videodecodificador y los dispositivos conectados.
- El bus permite la conexión de varios masters, ya que incluye un detector de colisiones.
- El protocolo de transferencia de datos y direcciones posibilita diseñar sistemas completamente definidos por software.
- Los datos y direcciones se transmiten con palabras de 8 bits.

Dentro de la comunicación I2C se da una condición de inicio (START) y otra de parada (STOP) dentro de la cual se da la transferencia de datos.

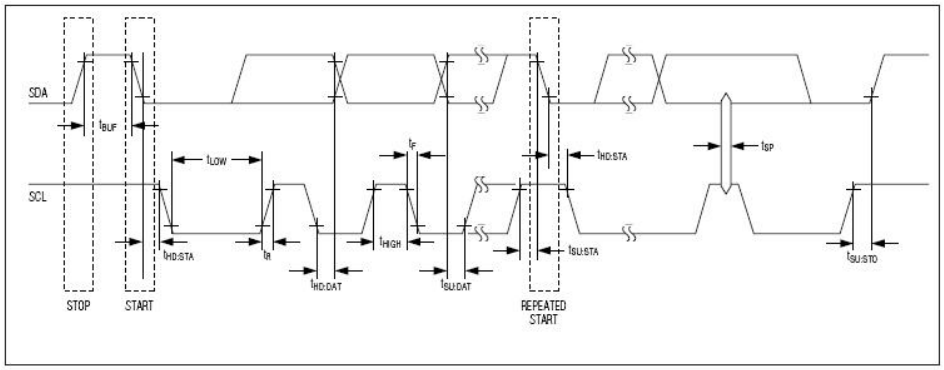


Figura 3.29: Transferencia de Datos I2C.

En la figura 3.29 se puede apreciar que, la condición START está definida, cuando la señal de reloj SCLK permanece estable en ALTO. El nivel de la señal de 'no reconocimiento', debe ser también ALTO, si en ese preciso instante se produce un flanco de bajada en la señal de datos, automáticamente se produce la condición de START (inicio) en la transmisión y el dispositivo que la produjo se convierte en MAESTRO, dando comienzo a la transmisión. El pulso de reconocer o reconocimiento, conocido como ACK (del inglés Acknowledge), se logra colocando la línea de datos a un nivel lógico bajo, durante el transcurso del noveno pulso de reloj [Inc05a].

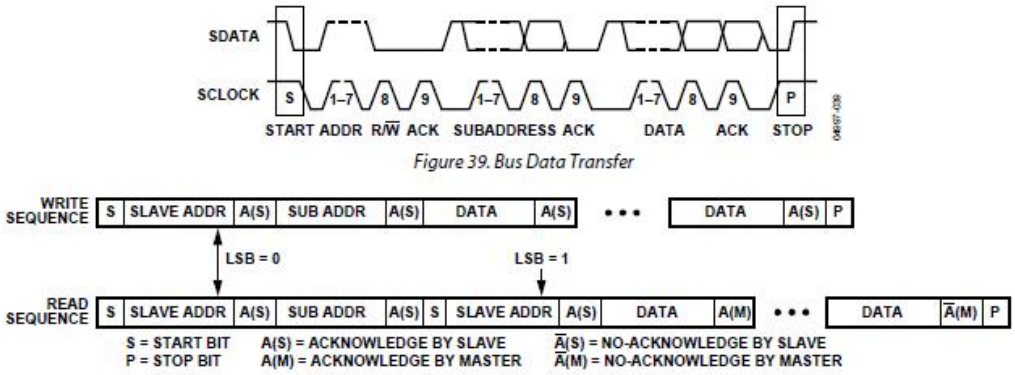


Figura 3.30: Secuencia de Lectura y Escritura.

Para la transferencia de los datos como se muestra en la figura 3.30, el maestro genera la condición de START. Cada palabra puesta en el bus SDA debe tener 8 bits, y la primera palabra transferida contiene la dirección del Esclavo seleccionado. Luego el maestro lee el estado de la línea SDA, si vale cero (impuesto por el esclavo), el proceso de transferencia continúa. Si vale uno, indica que el circuito direccionado no valida la comunicación, entonces, el maestro genera un bit de stop para liberar el bus I2C.
Este acuse de recibo se denomina ACKL y es una parte importante del protocolo I2C. Al final de la transmisión, el maestro genera la condición de Stop y libera el bus I2C, y las líneas SDA y SCL pasan a estado alto. Para implementar la comunicación I2C se realizaron 4 programas, con el fin de poder tener la facilidad de cambiar un programa sin afectar el protocolo de comunicación. El programa I2Cprogram es el que vamos a enviar al videodecodificador.

```
process(pair_num) --Valores en los registros a programar
begin
    case pair_num is
        when "00000" => Sreg_byte <= x"00"; Sdata_byte <= x"04"; --Entrada de video
        when "00001" => Sreg_byte <= x"15"; Sdata_byte <= x"00"; --Digital Clamp Control 1
        when "00010" => Sreg_byte <= x"17"; Sdata_byte <= x"41"; --Shaping Filter Control
        when "00011" => Sreg_byte <= x"27"; Sdata_byte <= x"58"; --ADC Control ADC1 and ADC 2 Off
        when "00100" => Sreg_byte <= x"3A"; Sdata_byte <= x"16"; --CTI DNR Ctrl 4
        when "00101" => Sreg_byte <= x"50"; Sdata_byte <= x"04"; --ADI Control
        when "00110" => Sreg_byte <= x"0E"; Sdata_byte <= x"80"; --CTI DNR Ctrl 4
        when "00111" => Sreg_byte <= x"50"; Sdata_byte <= x"20"; --
        when "01000" => Sreg_byte <= x"52"; Sdata_byte <= x"18"; --
        when "01001" => Sreg_byte <= x"58"; Sdata_byte <= x"ED"; --
        when "01010" => Sreg_byte <= x"77"; Sdata_byte <= x"C5"; --
        when "01011" => Sreg_byte <= x"7C"; Sdata_byte <= x"93"; --
        when "01100" => Sreg_byte <= x"7D"; Sdata_byte <= x"00"; --
        when "01101" => Sreg_byte <= x"D0"; Sdata_byte <= x"48"; --
        when "01110" => Sreg_byte <= x"D5"; Sdata_byte <= x"A0"; --
        when "01111" => Sreg_byte <= x"D7"; Sdata_byte <= x"EA"; --SD Saturation Cr
          when "10000" => Sreg_byte <= x"E4"; Sdata_byte <= x"3E"; --Recommended Setting
        when "10001" => Sreg_byte <= x"EA"; Sdata_byte <= x"0F"; --PAL F Bit Toggle
        when "10010" => Sreg_byte <= x"0E"; Sdata_byte <= x"00"; --ADI Control
        when others => null;
    end case;
end process;
```

Figura 3.31: Programación de los Registros.

El programa I2Ccontrol incrementa el puntero pairnum, el cual se presenta luego de toda la transmisión de una línea de programa. El programa I2Cconfig, es el protocolo de comunicación I2C que implementamos a través de una máquina de estados. El programa ClkI2C, es un divisor de frecuencia para generar una señal de reloj a 1.56 MHz necesaria para la transmisión de los datos a los registros del videodecodificador.

MÓDULO DCM Y LCD La tarjeta de desarrollo FPGA Spartan-3E Starter Kit cuenta con un cristal líquido (LCD) de 2 líneas por 16 carácteres. La FPGA controla el LCD a través de los datos de 4-bit de la interfaz cómo se muestra en la figura 3.32 [Inc09a].

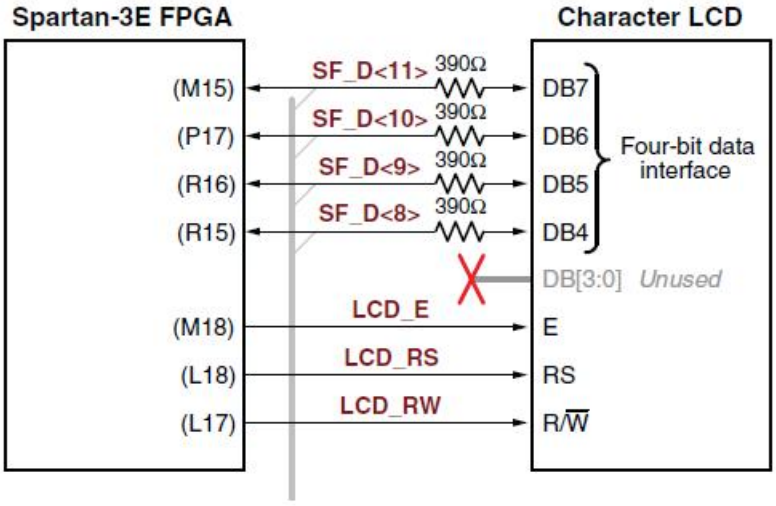


Figura 3.32: Interfaz LCD.

Aunque la pantalla LCD es compatible con una interfaz de 8-bit de datos, la interfaz de comunicación con la FPGA es de 4-bit para reducir al mínimo el número de pines en total [Inc09a].
Las señales de interfaz son:

- Data (D[11], D[10], D[9], D[8]): Bus de 4 bit de datos.
- LCD_E: Pin de habilitación. Cuando es 0: Desactivado, 1: Operación de lectura/escritura activada.
- LCD_RS: Registro de Selección. Cuando es 0: Registro de instrucción durante las operaciones de escritura. Coloca la Flash ocupada durante las operaciones de lectura. Cuando es 1: Datos para operaciones de lectura y escritura.
- LCD_RW: Control de Lectura y Escritura. Cuando es 0: ESCRITURA. LCD acepta datos. Cuando es 1: LECTURA. LCD entrega datos.

En este proyecto se hizo uso del Display para el manejo de la frecuencia del DCM.

El DCM (Digital Clock Managers) multiplica o divide la frecuencia del reloj de entrada para sintetizar una nueva frecuencia de reloj, especialmente en implementaciones de alto rendimiento y aplicaciones de alta frecuencia. Las principales características son:

- Mejorar el rendimiento general del sistema y eliminar los retrasos de distribución de reloj.
- Señal de reloj, ya sea para una fracción fija de un período de reloj o para incrementos de cantidades.

- Multiplicar o Dividir una frecuencia de reloj o sintetizar una frecuencia completamente nueva.
- Garantizar un ciclo de trabajo del 50 % en las señales de reloj.
- Espejo o búfer de una señal de reloj, para alineación entre señales de entrada y salida estándar.

La función de este módulo es ofrecer una mayor velocidad de procesamiento y de manejo en la RAM.

MÓDULO VISUALIZACIÓN VGA Para la programación de la visualización en pantalla hicimos uso de un código utilizado en la materia Sistemas Digitales III. El código para el manejo de la VGA consta de dos programas, el primero es el VGAdrive que es un driver que da los tiempos para la sincronización vertical y horizontal al monitor. El segundo código es vgavis el cual es una interface con la cual podemos manejar la información que se envía a pantalla. En este se implementaron los algoritmos de visualización para mostrar la imagen en pantalla, la imagen del tracking y las coordenadas. Para el desarrollo de estos algoritmos fue necesario implementar varios procesos como matrices de números y letras para visualizar las coordenadas del objeto, procesos de desplazamiento para la imagen y tracking.
Para visualizar las coordenadas en pantalla se reciben dos señales std_logic_vector(10 downto 0) de 11 bits (cmax y fmax) que corresponden a la coordenada del centroide del objeto entregada del procesamiento del video. Con estas señales es posible representar 2046 números, las cuales se convierten en tipo entero y se comienza a separar el número en unidades, decenas, centenas y milésimas, con el siguiente código:

```
182   ---- Posicion del objeto coordenadas
183   process(clock)
184   variable aux    : integer;
185   begin
186   if clock = '1' and clock'event then
187   aux:=cordx;     -- Coordenada X
188   dat1<=div(aux,1000);
189   aux:=aux-(dat1*1000);
190   dat2<=div(aux,100);
191   aux:=aux-(dat2*100);
192   dat3<=div(aux,10);
193   aux:=aux-(dat3*10);
194   dat4<=aux;
195   aux:=cordy;     -- Coordenada Y
196   dat5<=div(aux,1000);
197   aux:=aux-(dat5*1000);
198   dat6<=div(aux,100);
199   aux:=aux-(dat6*100);
200   dat7<=div(aux,10);
201   aux:=aux-(dat7*10);
202   dat8<=aux;
203   end if;
204   end process;
```

Figura 3.33: Identificación de Dígitos.

Se puede observar que cada dígito queda guardado en variables dat. Luego de tener cada dígito separado se compara con las matrices de números guardadas como se observa en la siguiente parte del código:

```
111  letX<=(('1','0','1'),('1','0','1'),('0','1','0'),('0','1','0'),
112         ('0','1','0'),('1','0','1'),('1','0','1'));
113  letY<=(('1','0','1'),('1','0','1'),('1','0','1'),('1','1','1'),
114         ('0','0','1'),('0','0','1'),('1','1','1'));
115  num1<=(('0','0','1'),('0','0','1'),('0','0','1'),('0','0','1'),
116        ('0','0','1'),('0','0','1'),('0','0','1'));
117  num2<=(('1','1','1'),('0','0','1'),('0','0','1'),('1','1','1'),
118        ('1','0','0'),('1','0','0'),('1','1','1'));
119  num3<=(('1','1','1'),('0','0','1'),('0','0','1'),('1','1','1'),
120        ('0','0','1'),('0','0','1'),('1','1','1'));
121  num4<=(('1','0','1'),('1','0','1'),('1','0','1'),('1','1','1'),
122        ('0','0','1'),('0','0','1'),('0','0','1'));
123  num5<=(('1','1','1'),('1','0','0'),('1','0','0'),('1','1','1'),
124        ('0','0','1'),('0','0','1'),('1','1','1'));
125  num6<=(('1','0','0'),('1','0','0'),('1','0','0'),('1','1','1'),
126        ('1','0','1'),('1','0','1'),('1','1','1'));
127  num7<=(('1','1','1'),('0','0','1'),('0','0','1'),('0','0','1'),
128        ('0','0','1'),('0','0','1'),('0','0','1'));
129  num8<=(('1','1','1'),('1','0','1'),('1','0','1'),('1','1','1'),
130        ('1','0','1'),('1','0','1'),('1','1','1'));
131  num9<=(('1','1','1'),('1','0','1'),('1','0','1'),('1','1','1'),
132        ('0','0','1'),('0','0','1'),('0','0','1'));
133  num0<=(('1','1','1'),('1','0','1'),('1','0','1'),('1','0','1'),
134        ('1','0','1'),('1','0','1'),('1','1','1'));
```

Figura 3.34: Caracteres Representados en Matrices.

De esta forma podemos representar cualquier carácter para posteriormente ser mostrado

en pantalla. El tamaño de los arreglos de los carácteres es de (1 a 7, 1 a 3) de señales std_logic. Se asigna un espacio en pantalla llamada pos y con el siguiente algoritmo comparamos el digito guardado y se asigna a la posición de la matriz correspondiente:

```
188   ---------------------------------Coordenada a Pantalla----------------------
189   process(dat1,dat2,dat3,dat4,dat5,dat6,dat7,dat8)
190   begin
191   case dat1 is
192       when 0 => pos1 <= num0; when 1 => pos1 <= num1; when 2 => pos1 <= num2;
193       when 3 => pos1 <= num3; when 4 => pos1 <= num4; when 5 => pos1 <= num5;
194       when 6 => pos1 <= num6; when 7 => pos1 <= num7; when 8 => pos1 <= num8;
195       when 9 => pos1 <= num9; when others => pos1 <= letX;
196   end case;
197   case dat2 is
198       when 0 => pos2 <= num0; when 1 => pos2 <= num1; when 2 => pos2 <= num2;
199       when 3 => pos2 <= num3; when 4 => pos2 <= num4; when 5 => pos2 <= num5;
200       when 6 => pos2 <= num6; when 7 => pos2 <= num7; when 8 => pos2 <= num8;
201       when 9 => pos2 <= num9; when others => pos2 <= letX;
202   end case;
203   case dat3 is
204       when 0 => pos3 <= num0; when 1 => pos3 <= num1; when 2 => pos3 <= num2;
205       when 3 => pos3 <= num3; when 4 => pos3 <= num4; when 5 => pos3 <= num5;
206       when 6 => pos3 <= num6; when 7 => pos3 <= num7; when 8 => pos3 <= num8;
207       when 9 => pos3 <= num9; when others => pos3 <= letX;
208   end case;
209   case dat4 is
210       when 0 => pos4 <= num0; when 1 => pos4 <= num1; when 2 => pos4 <= num2;
211       when 3 => pos4 <= num3; when 4 => pos4 <= num4; when 5 => pos4 <= num5;
212       when 6 => pos4 <= num6; when 7 => pos4 <= num7; when 8 => pos4 <= num8;
213       when 9 => pos4 <= num9; when others => pos4 <= letX;
214   end case;
```

Figura 3.35: Asignación de Coordenada en Pantalla.

Teniendo los datos (arreglos de vectores) y la asignación (posición en pantalla) otro proceso envía los datos asignados en las posiciones a la VGA. El código para realizar esta tarea es el siguiente:

```
Coordenadas de Posición
if  row > posimay and row < 250+posimay and column > posimax and column < 380 +posimax
      and reading = '1' then
   green <= dataRGB(0);
elsif row > postray  and row < 60 + postray and column > postrax and column < 60 + postrax then
   green <= colorR(0);
--x y
elsif row >= h and row <= h+7 and column >= k and column <= k+4 then
   datoY <= vector_to_entero(column-k); datoX <= vector_to_entero(row-h);
   numdato <= letX(datoX,datoY);
   if numdato = '1' then   green<='1';
   else                    green<='0';
   end if;
elsif  row >= h and row <= h+7 and column >= k+5 and column <= k+9 then
   datoY <= vector_to_entero(column-5-k); datoX <= vector_to_entero(row-h);
   numdato <= letY(datoX,datoY);
   if numdato = '1' then   green<='1';
   else                    green<='0';
   end if;
-- Coordenada x
elsif  row >= h and row <= h+7 and column >= k+15 and column <= k+19 then
   datoY <= vector_to_entero(column-15-k); datoX <= vector_to_entero(row-h);
   numdato <= pos1(datoX,datoY);
   if numdato = '1' then   green<='1';
   else                    green<='0';
   end if;
elsif  row >= h and row <= h+7 and column >= k+20 and column <= k+24 then
```

Figura 3.36: Asignación de Dígitos en Pantalla.

Para mostrar la imagen en pantalla y el tracking lo que hacemos es leer directamente la memoria RAM donde está guardada la imagen (RAM0) y leer la RAM donde está el tracking de la imagen (RAM1). Para leer la RAM se tienen las siguientes señales: L_C y L_R las cuales representan a las filas y columnas generadas por el driver de la VGA, usaremos estas señales para dar comienzo a la lectura de la RAM cuando L_C=0 y L_R=0, es decir el comienzo de una imagen de video. Las señales C_num, R_num y C_n, R_n leen la direcciones de la RAM para la lectura. C_num y R_num son señales para leer la imagen de entrada desde la RAM0. C_n y R_n son señales para leer la imagen del tracking del objeto al cual se le está haciendo el seguimiento. En la pantalla estas imágenes son mostradas en diferentes segmentos de la pantalla. En el centro se mostrará la imagen procesada de entrada, y en la parte derecha inferior en un recuadro el tracking. Para realizar esto se implementó un código de offset para mover fácilmente los objetos, lo cual se muestra en el siguiente código:

```
297     elsif row > posimay and row < 250+posimay
298           and column > posimax and column < 380 + posimax
299           and reading = '1' then
300        blue <= dataRGB(0);
301     elsif row > postray  and row < 60 + postray
302           and column > postrax and column < 60 + postrax then
303        blue <= colorR(0);
304     else
305        blue<= '0';
306     end if;
```

Figura 3.37: Visualización de Objeto y Tracking.

DataRGB es la información del pixel extraída de RAM0 donde está la imagen guardada, y color es la información extraída de RAM1 del tracking del objeto. El Offset se puede ver como (imagen x + offset x, imagen y + offset y) en un sistema bidimensional. Como se puede ver en el anterior código el offset para la imagen para 'y' es posimay, y para 'x' es posimax. Para el tracking postrax y postray. Para leer la RAM desde la posición 0 se debe quitar el offset, y para esto se implementó el siguiente código:

```
150   L_C <= column;
151   L_R <= row;
152   posimax <= 100;
153   posimay <= 100;
154   postrax <= 540;
155   postray <= 410;
156
157   process(column,row)
158   begin
159   if column >= posimax and row >= posimay then
160   C_num <= column- posimax;
161   R_num <= row    - posimay;
162   else
163   C_num <= "00000000000";
164   R_num <= "00000000000";
165   end if;
166   end process;
167
168   process(column,row)
169   begin
170   if column >= postrax and row >= postray then
171   C_n <= column- postrax;
172   R_n <= row  - postray;
173   else
174   C_n <= "00000000000";
175   R_n <= "00000000000";
176   end if;
177   end process;
```

Figura 3.38: Posición de la Imagen sin Offset.

Con las anteriores implementaciones se creó la interfaz gráfica para mostrar las coordenadas del objeto, la imagen captada y el tracking de la imagen. La visualización en el monitor se puede observar en la figura 3.39:

Figura 3.39: Imagen de Prueba con Umbral de Luminancia.

MÓDULO INICIALIZACIÓN MEMORIA RAM La implementación de la memoria RAM para el tracking y para la imagen se realizó a través del Block Memory Generator v3.3 de Xilinx Inc. Con esta herramienta los usuarios pueden crear rápidamente memorias optimizadas para aprovechar el rendimiento y las características de la RAM en el bloque de FPGAs de Xilinx [Inc09a]. Las principales características son:

- Genera un solo puerto de RAM, memoria RAM simple de dos puertos, memoria RAM de doble puerto verdadero, ROM de un solo puerto, y ROM de doble puerto.
- Rendimiento de hasta 450 MHz.
- Algoritmos optimizados para la utilización de los recursos mínimos del bloque RAM o la utilización de bajo consumo de energía.
- La Inicialización de la memoria es configurable.
- Optimización de los tiempos de simulación y simulación precisa de las conductas de memoria.

El bloque generador de memoria tiene dos puertos totalmente independientes que acceden a un espacio de memoria compartida. Tanto A como B tienen puertos de escritura y una interfaz de lectura. Los tipos de memoria generadas utilizan una memoria RAM en bloque incrustado para generar cinco tipos de memorias:

- RAM de un solo puerto.

- RAM de doble puerto.
- RAM de dos puertos.
- ROM de un solo puerto.
- ROM de doble puerto.

En nuestro trabajo se uso la RAM de dos puertos como se muestra en la figura 3.40:

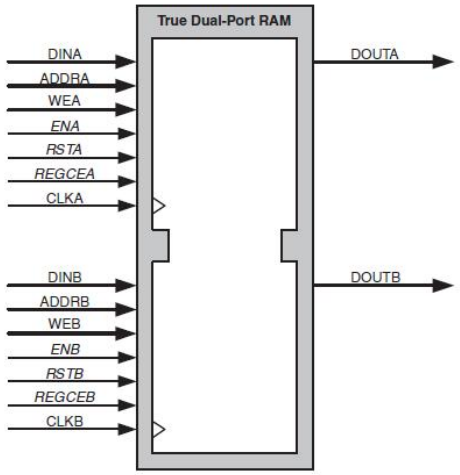


Figura 3.40: Memoria RAM de dos puertos.

Para la Tarjeta Spartan 3E es posible configurar la memoria RAM de las siguientes formas: 16Kx1, 8Kx2, 4Kx4, 2Kx9, 1Kx18, 512kx36 y para configuraciones de puerto simple 256Kx72. En nuestro caso tenemos una dirección de memoria de 17bits (131070 Posiciones de memoria) y un tamaño de datos de palabra de 1 bit para la imagen y para el tracking una memoria de 12 bits (4094 Posiciones de memoria) y un tamaño de palabra de 1 bit. Para leer y guardar datos en la memoria RAM se debe leer y escribir durante dos ciclos de reloj. No es posible leer y escribir al mismo tiempo y cada una de estas acciones deben ser realizadas en espacios diferentes de tiempo. En la figura 3.41 se puede observar:

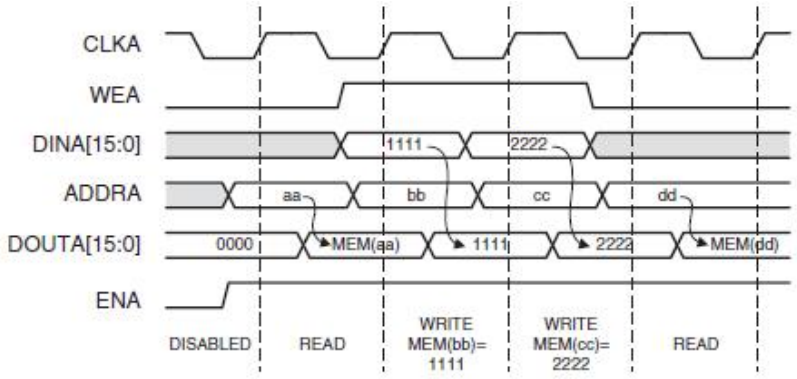


Figura 3.41: Ciclos de Reloj Memoria RAM.

Como se había mencionado antes para nuestro proyecto se implementaron dos memorias RAM, una para guardar la imagen y otra para guardar el tracking de la imagen. Los pasos para la implementación es la siguiente:
Seleccionamos la opción Project y damos click en New Source donde escogemos la opción IP (Coregen & Architecture Wizard) como se muestra en la figura 3.42:

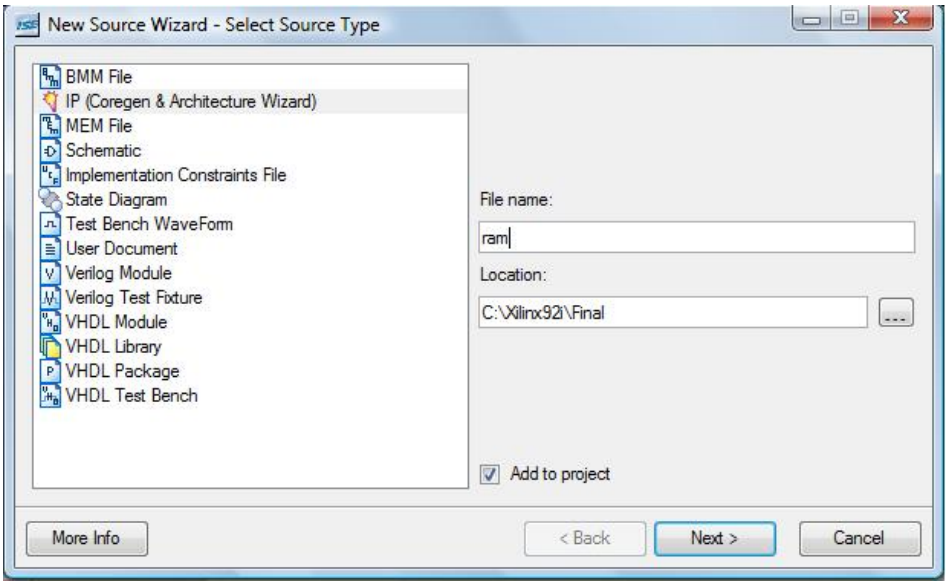


Figura 3.42: Paso 1 Generación de RAM.

Después seleccionamos la opción Memories & Storage Elements, después la opción RAMs & ROMs y seguido seleccionamos Block Memory Generator el cual nos proporciona las opciones de selección del tipo de memoria:

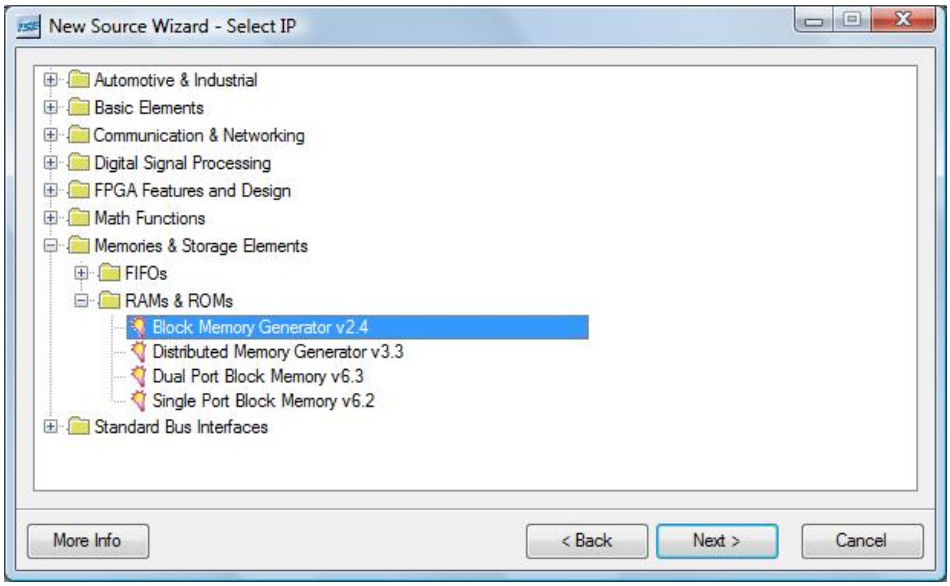


Figura 3.43: Paso 2 Generación de RAM.

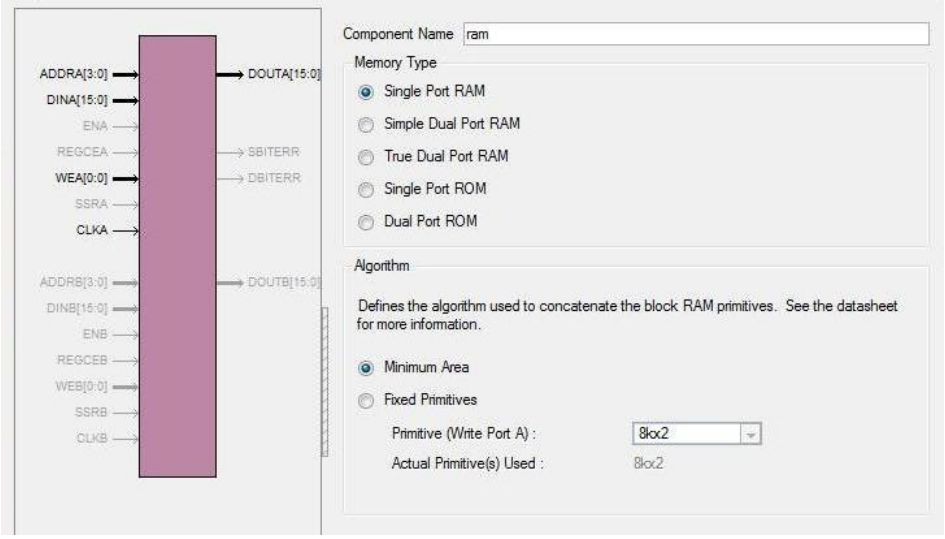


Figura 3.44: Paso 3 Generación de RAM.

Se escogió implementar dos memorias RAM simple con área mínima. En la siguiente ventana se selecciona el tamaño de la memoria.

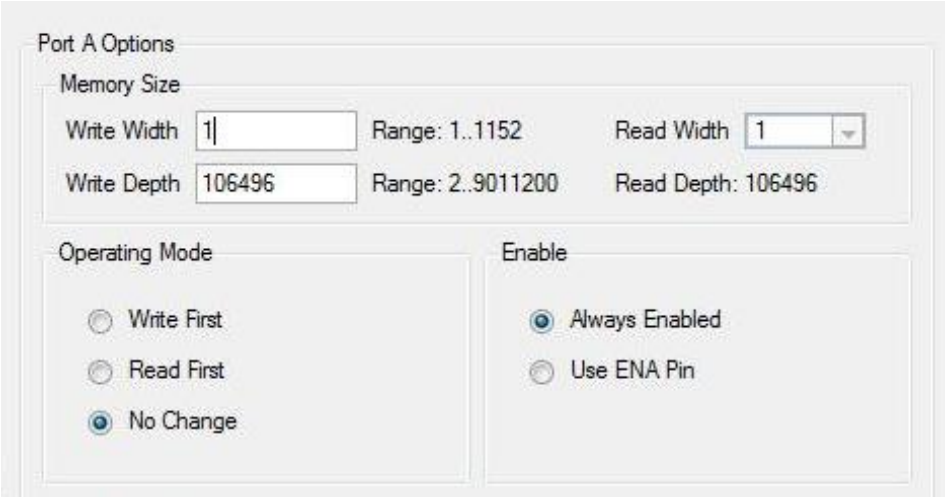


Figura 3.45: Paso 4 Generación de RAM.

Write Width hace referencia al tamaño de la palabra que vamos a utilizar. Para una tarjeta con un espacio de memoria y salida SVGA un tamaño de memoria adecuado sería 24 para guardar los tres espacios de color RGB a 8 bits por la segmentación de la imagen. La imagen de salida es una imagen binaria y por este motivo es suficiente asignarle un bit en el tamaño de palabra.
Write Depth hace referencia al número de posiciones de memoria en la cual se va a guardar la imagen. Después se muestran las opciones de los registros de salida. Para aplicaciones en las cuales no es importante la velocidad de procesamiento se pueden usar estas opciones y mejorar el uso de la RAM.

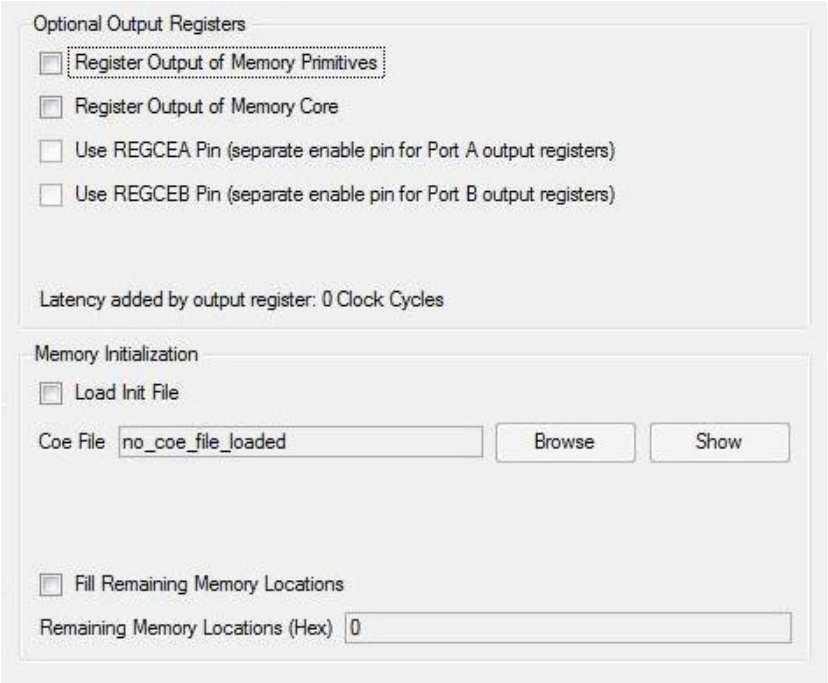


Figura 3.46: Paso 5 Generación de RAM.

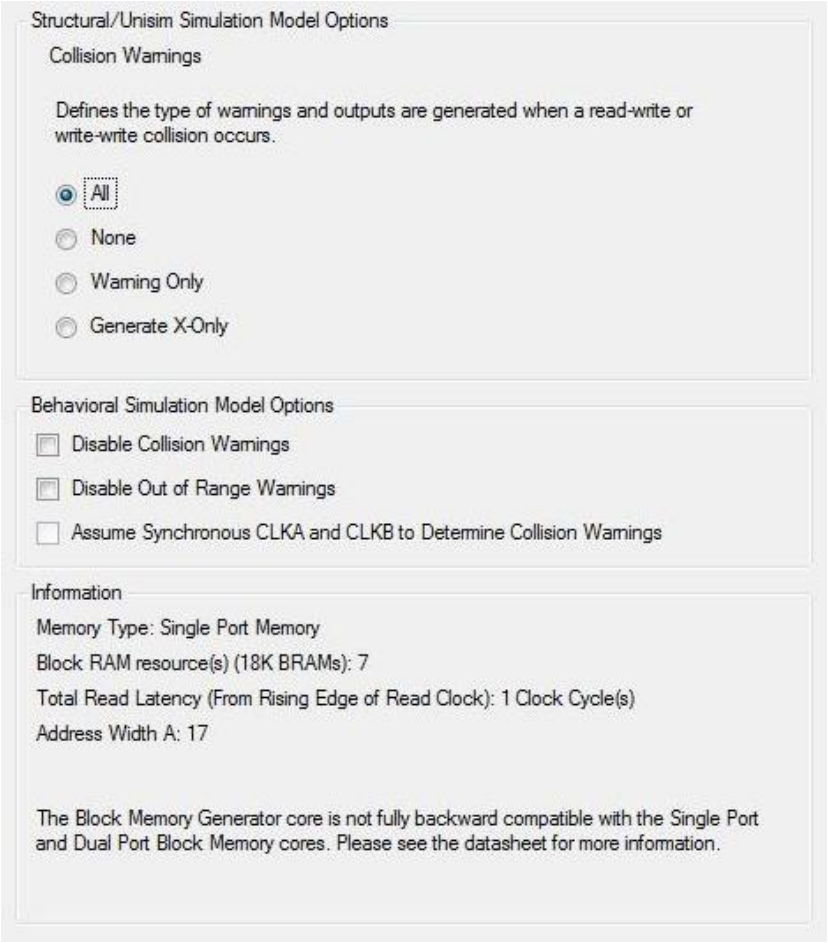


Figura 3.47: Paso 6 Generación de RAM.

Luego de realizar estos pasos se obtiene la implementación de la memoria RAM donde se guardara la imagen adquirida como se observa en la figura 3.48:

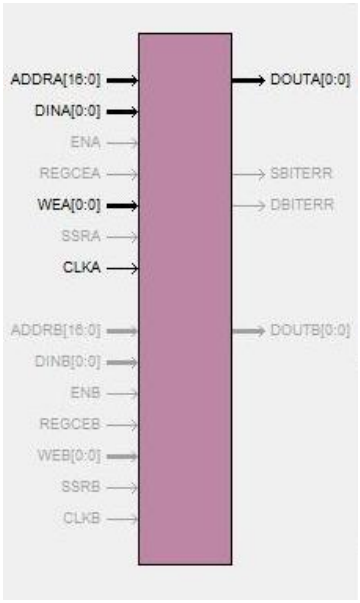


Figura 3.48: Memoria RAM.

MÓDULO PRE-PROCESAMIENTO Y PROCESAMIENTO En este módulo se implementaron los algoritmos para realizar el pre-procesamiento y el procesamiento del video, y adicionalmente se realiza el manejo de la memoria RAM para realizar escritura, procesos con los datos almacenados y lectura. Este módulo está compuesto de 6 programas cada uno con una tarea específica: El primero hace parte del pre-procesamiento en donde se guarda la imagen de entrada a la RAM y del procesamiento al crear un espacio para la manipulación de los datos previamente guardados. La principal función es organizar que los procesos de escritura, procesamiento y lectura puedan acceder a la memoria RAM en diferentes instantes de tiempo de una manera organizada y para lograr esto se hizo uso de una bandera a la que llamamos estado. El algoritmo para acceder de forma apropiada a la RAM para cada proceso se muestra a continuación:

```
154  process(clky, dirW, dirR, dirP, din, douta, estado)
155  begin
156  if    estado = "10" then --Grabando imagen
157     if clky = '1' then
158        addra <= dirW;
159        reading <= '0';
160        dina    <= din;
161        dataRGB_Read <= "0";
162     else
163        addra <= dirR;
164        dataRGB_Read <= douta;
165        dataRGB_Pro  <= "0";
166        reading  <= '1';
167        dina  <= "0";
168     end if;
169  elsif estado = "01" then --Procesando Imagen
170     if clky = '1' then
171        addra <= "00000000000000000";
172        reading  <= '0';
173        dina  <= din;
174        dataRGB_Read <= "0";
175     else
176        addra <= dirP;
177        dataRGB_Read <= "0";
178        dataRGB_Pro  <= douta;
179        reading  <= '0';
180        dina  <= "0";
181     end if;
182  elsif estado = "00" then -- Leyendo imagen
183     if clky = '1' then
184        addra <= "00000000000000000";
185        reading  <= '0';
186        dina  <= din;
187        dataRGB_Read <= "0";
188     else
189        addra <= dirR;
190        dataRGB_Pro  <= "0";
191        dataRGB_Read <= douta;
192        reading  <= '1';
193        dina  <= "0";
194     end if;
```

Figura 3.49: Control de la Memoria RAM.

A los puertos de la memoria RAM se le asignan diferentes señales dependiendo del proceso que se esté ejecutando. El segundo programa del pre-procesamiento está encargado de organizar las posiciones de los píxeles y líneas en un espacio bidimensional de acuerdo al estándar ITU-R BT.656, para lo cual se hace uso de las señales de StartCount, RstrtLines, Field y Load de este módulo. Los píxeles y líneas se asignan en vectores de 11 bits lo que nos permite guardar una imagen de hasta 2046x2046 píxeles aunque el tamaño posible de la imagen en la memoria estará dado por el tamaño de la memoria RAM. El código implementado realiza aumentos de líneas horizontales y verticales como se observa en la figura 3.50.

Figura 3.50: Organización de los Datos del Bloque ITU-R BT.656.

La implementación del programa para el aumento de las líneas horizontales (píxeles) y verticales (líneas) es el siguiente :

```
29  process(llc) -- Aumento Para las Lineas Horizontal
30  begin
31     if llc'event and llc = '1' then --if load'event and load = '1' then
32        if StartCount = '1' and Field ='0' then
33              if load = '1' then
34              pixel_n <= pixel_n + 1;
35              else
36              pixel_n <= pixel_n;
37              end if;
38        else
39              pixel_n  <= "00000000000";
40        end if;
41     end if;
42  end process;
43
44  process(RstrtLines, StartCount,PB_R) -- Aumento Para las Lineas Vertical
45  begin
46  -- if RstrtLines = '1' or PB_R = '0' then
47  --          linea_n <= "00000000000";
48  -- else
49        if  StartCount'event and StartCount = '1' then
50           if Field = '0' then
51              linea_n <= linea_n + 1;
52           else
53              linea_n <= "00000000000";
54           end if;
55        end if;
56  -- end if;
57  end process;
58
```

Figura 3.51: Aumento de Píxeles y Líneas.

El tercer programa se encarga de recibir las señales bidimensionales y organizarlas de manera unidimensional partiendo la dirección de la memoria RAM en dos partes (alta y baja), es decir de un tamaño de 17 bits para posiciones de memoria repartiendose de tal

forma que los píxeles son la parte baja con 9 bits y las líneas son la parte alta con 8 bits lo que nos permitiría almacenar una imagen de 512x256 píxeles. Las señales se reciben del módulo ITU-R BT.656, así como también la posición de líneas y píxeles recibidas del procesamiento y posición de filas y columnas recibidas del módulo de visualización. La dirección de salida a memoria se guarda en las señales dEscritura, dLectura y dProceso cuyo algoritmo se muestra a continuación:

```
43  --Escritura
44  tempdirW <= linea_num(7 downto 0)&"000000000";
45  temppixW <= "00000000"& pixel_num(8 downto 0);
46  process(linea_num, pixel_num, temppixW, tempdirW, dEscritura)
47  begin
48  if (linea_num <= 253) then
49     if(pixel_num <= 381) then
50        dEscritura <= tempdirW + temppixW;
51     else
52        dEscritura <= dEscritura;
53     end if;
54  else
55        dEscritura <= dEscritura;
56  end if;
57  end process;
58
59  --Lectura
60  tempdirR <= R_num(7 downto 0)&"000000000";
61  tempColR <= "00000000"& C_num(8 downto 0);
62  process(R_num, C_num, tempColR, tempdirR, dEscritura)
63  begin
64  if(R_num <= 253) then
65     if(C_num <= 381) then
66        dLectura <= tempdirR + tempColR;
67     else
68        dLectura <= dLectura;
69     end if;
70  else
71        dLectura <= dLectura;
72  end if;
73  end process;
74
75  --Procesamiento
76  tempdirP <= linea(7 downto 0)&"000000000";
77  temppixP <= "00000000"& pixel(8 downto 0);
78  process(linea, pixel, tempdirP, temppixP, dProceso)
79  begin
80  if (linea <= 253) then
81     if(pixel <= 381) then
82        dProceso <= tempdirP + temppixP;
83     else
84        dProceso <= dProceso;
85     end if;
86  else
87        dProceso <= dProceso;
88  end if;
89  end process;
90
91
92  end ReadWrite;
```

Figura 3.52: Rutina para Lectura y Escritura en la Memoria RAM.

El cuarto programa realiza el control de lectura y escritura de la memoria RAM, creando la bandera 'estado' para el adecuado uso de tiempos en la memoria RAM al guardar,

procesar y leer. Este programa se implementó a través de las señales L_R entregada por el Módulo de Visualización VGA, linea_n del Módulo Estándar ITU-R BT.656 y linea del programa de ubicación del objeto. La implementación se hizo a través de la siguiente máquina de estados:

```
38  process (estado_act)    -- Accion de los estados
39  begin
40     case estado_act is
41        when EspW => estado <= "10";-- Esperando W
42        when Wima => estado <= "10";-- Grabando imagen
43        when Pima => estado <= "01";-- Procesando imagen
44        when EspR => estado <= "00";-- Esperando R
45        when Rima => estado <= "00";-- Leyendo imagen
46        when others    => null;
47        end case;
48  end process;
49  process (estado_act, Linea_n, linea, L_R)
50  begin
51     case estado_act is
52        when EspW => if Linea_n =  0  then  estado_sig <= Wima;
53                     else                   estado_sig <= EspW;
54                     end if;
55        when Wima => if Linea_n = 253 then  estado_sig <= Pima;
56                     else                   estado_sig <= Wima;
57                     end if;
58        when Pima => if linea   = 253 then  estado_sig <= EspR;
59                     else                   estado_sig <= Pima;
60                     end if;
61        when EspR => if  L_R    = 0   then  estado_sig <= Rima;
62                     else                   estado_sig <= EspR;
63                     end if;
64        when Rima => if  L_R    = 479 then  estado_sig <= EspW;
65                     else                   estado_sig <= Rima;
66                     end if;
67        when others =>                      estado_sig <= EspW;
68        end case;
69  end process;
70
71
72  end RAMcontrolRW;
```

Figura 3.53: Control de Lectura y Escritura en la Memoria RAM.

Según el anterior código, se espera el inicio de la primera línea del estándar ITU-R BT.656 para escribir la imagen en la memoria RAM. Cuando se termina de realizar la escritura del frame de video se realiza el procesamiento y cuando este termina se espera la primera fila dada por la VGA para realizar la lectura de la imagen procesada desde la memoria RAM.

El quinto programa hace parte del pre-procesamiento y su función es filtrar la imagen con motivo de eliminar el ruido y poder realizar un buen procesamiento el cual se implementó con un filtro de mediana. Se debe tener en cuenta que los datos en el proceso de escritura ya se encuentran umbralizados por el Módulo de Segmentación.
El sexto y último programa de este Módulo hace parte del proceso de procesamiento del video y tiene como función la ubicación del objeto. Como se mencionó anteriormente la

imagen guardada se encuentra pre-procesada lo que quiere decir que previamente se le realizó el proceso de filtrado y segmentación. La ubicación del objeto se realiza con la ubicación del centroide. Para realizar la ubicación de la existencia de objeto se realiza una validación con una matriz de 3x3 como se muestra a continuación:

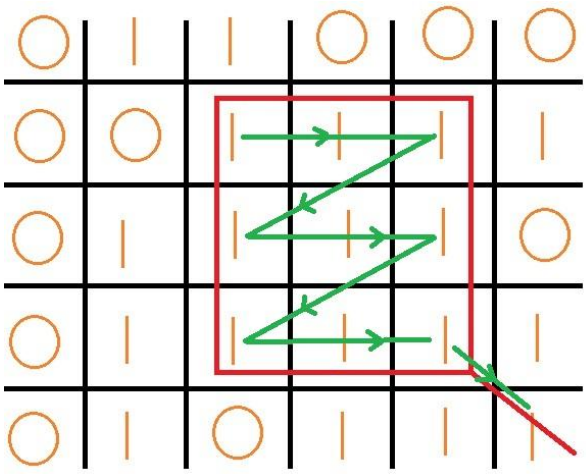


Figura 3.54: Validación en la Matriz 3x3.

Para asegurar que esto solo se realice cuando exista en una fila de 3 bits continuos se tienen tres registros para guardar los datos leídos de la memoria RAM como se muestra en el siguiente código:

```
46  ----------------------------------------------------------------
47  --------------------------Registros-----------------------------
48  ----------------------------------------------------------------
49
50  process(clockRAM,PB_R, reset) --Registros de memoria de 2 pixel
51  begin
52     if clockRAM'event and clockRAM = '1' then
53        if(PB_R = '0' or  reset = '1') then
54           reg1 <= "0";
55           reg2 <= "0";
56        else
57           reg1 <= DatoRAM;      -- 1 clock delay
58           reg2 <= reg1;         -- 2 clock delay
59        end if;
60     end if;
61  end process;
```

Figura 3.55: Registros de Memoria.

Hacemos uso de una bandera la cual nos indica el estado del programa de acuerdo a que proceso se está realizando (escritura, procesamiento, lectura) y que parte de la validación y ubicación del centroide se está realizando. Cuando se encuentra el centroide del objeto la bandera toma el valor de '1000':

```
196  process(bandera,Spixel,Slinea)
197  begin
198  if bandera = "1000" then
199  piximax <= Spixel;
200  linimax <= Slinea;
201  end if;
202  end process;
```

Figura 3.56: Bandera del Estado del Programa.

MÓDULO TRACKING Este módulo hace parte del procesamiento y se compone de dos programas. El primero se encarga del manejo de lectura y escritura en la memoria RAM el cual se muestra a continuación:

```
23  --------Control de la RAM
24  process(Dw, Dr, wenab, colorW, dout)
25  begin
26  if wenab = "1" then
27  wea  <= "1";
28  addr <= Dw;
29  din  <= colorW;
30  else
31  wea <= "0";
32  addr <= Dr;
33  colorR <= dout;
34  end if;
35  end process;
```

Figura 3.57: Lectura y Escritura en la Memoria RAM.

El segundo código tiene como función asignar la dirección de escritura de acuerdo a las coordenadas entregadas de la ubicación del objeto y asignar la dirección de lectura para el módulo de visualización. Para asegurar un cambio de coordenada por frame y el almacenamiento de señales para realizar el proceso de escritura se implementó el siguiente algoritmo:

```
61  process(linea_num ,Spixel_track, Slinea_track )--Asegura una coordenada por cambio de frame
62  begin
63  if linea_num = 1 or linea_num = 249 then
64  px <= vector_to_entero(Spixel_track);
65  Py <= vector_to_entero(Slinea_track);
66  end if;
67  end process;
68
69  Spixel <= pixel_num - (Spixel_track - "00000011110");-- X - (CorX -30)
70  Slinea <= linea_num - (Slinea_track - "00000011110");-- Y - (CorY -30)
```

Figura 3.58: Acondicionamiento de Señales.

Para asignar la dirección de la posición de escritura y lectura se implementó el siguiente algoritmo:

```
 96  --Escritura
 97  tempdW <= linea(5 downto 0)&"000000";
 98  process(linea_num, pixel_num, px, py, tempdW, pixel)
 99  begin
100  if ( linea_num >= py - 30) and (linea_num <= py + 30)  then
101     if ( pixel_num >= px - 30) and (pixel_num <= px + 30)  then
102        SDw <= tempdW + pixel;
103     else
104        SDw <= SDw;
105     end if;
106  else
107        SDw <= SDw;
108  end if;
109  end process;
110
111  --Lectura
112  tempdR <= fila(5 downto 0)&"000000";
113  process(fila, colum, tempdR)
114  begin
115  if(fila < 60) then
116     if(colum < 60) then
117        SDr <= tempdR + colum;
118     else
119        SDr <= SDr;
120     end if;
121  else
122        SDr <= SDr;
123  end if;
124  end process;
```

Figura 3.59: Control de Grabación y Lectura.

Se hace un recuadro de 60x60 para el objeto encontrado por medio de las señales pixel_track y linea_track del Módulo Tracking. La imagen del tracking es guardada en la memoria RAM sin realizar el filtrado del Módulo explicado anteriormente.

MÓDULO DE SEGMENTACIÓN Para realizar la segmentación se recibe la información del píxel entregada del Módulo ITU-R BT.656 en el espacio de color RGB. La segmentación se realiza umbralizando como se observa en el siguiente código:

```
23  process(SignalR, llc)
24  begin
25  if llc = '1' and llc'event then
26     if SignalR > 180  and SignalB > 180 then  SiR <= '1'; SiG <= '1';  SiB <= '1';
27     else                    SiR <= '0'; SiG <= '0';  SiB <= '0';
28     end if;
29  end if;
30  end process;
```

Figura 3.60: Segmentación.

Luego de hacer este proceso se realiza la asignación a las señales de salida del módulo para que pueda ser procesada. dataRGB_Write es la señal que será procesada en el módulo de pre-procesamiento y procesamiento y colorW es para la imagen del tracking directamente en pantalla.

```
23  segment <= SiR and SiG and SiB;
24
25  process(SiR, SiG, SiB, segment, llc)
26  begin
27  if llc = '1' and llc'event then
28     if segment = '1' then   dataRGB_Write <= "1";   colorW <= "1";
29     else                    dataRGB_Write <= "0";   colorW <= "0";
30     end if;
31  end if;
32  end process;
```

Figura 3.61: Segmentación a RGB.

3.3.2. IMPLEMENTACIÓN DEL PROGRAMA EN PC La implementación en PC se realizó a través de una Tarjeta de Video LR37 Rev:B adquiriendo la imagen por el puerto de S-Video.

Figura 3.62: Tarjeta de Video LR37 para PC

Para instalar la tarjeta de video en el sistema fue necesario ejecutar el programa con modo de compatibilidad para windows 2000 y reemplazar el archivo de configuración original del driver por otro para hacer compatible el driver con el sistema operativo windows XP Profesional. El procesamiento de video se realizó con el programa MatLab R2009b, dentro del cual se implementó un algoritmo para la configuración con la tarjeta de video el cual se muestra a continuación:

```
%% Inicialización de Variables
toma=1;
camara = 0;

%% Adquisición de Video y Configuración
canalVideo=videoinput('winvideo',1);              % Crea un canal de Video
set(canalVideo,'TriggerRepeat',Inf);
set(canalVideo, 'ReturnedColorSpace','rgb');      % Cambia formato de la camara
tic
```

Figura 3.63: Configuración Tarjeta de Video.

La bandera 'toma' del anterior código nos indica la duración máxima del video ya que como se está guardando datos de la posición del centroide si no se coloca una bandera se llega a un volcado de memoria en MatLab presentándose errores en la ejecución del programa. La bandera camara nos indica el estado de la cámara, si esta prendida (1) ó apagada (0). Para iniciar el canal de video anteriormente configurado se tiene:

```
%% Programa Principal                          % Crea Visor
start(canalVideo);                            % Iniciar canal de Video
camara =1;

while toma<=30
data = getsnapshot(canalVideo);               % Selecciona un frame
```

Figura 3.64: Inicialización Canal de Video.

Se realiza la segmentación por color del objeto, y ubicación del objeto encontrando el centroide a través de un barrido de la imagen, inicialmente se realizó definiendo cada espacio de color>180 pero por problemas de ubicación del objeto se definió como se muestra a continuación:

```
%% Detección de color
ImaR = (data(:,:,1))> 210;
ImaG = (data(:,:,2))> 180;
ImaB = (data(:,:,3))> 190;
IR = (data(:,:,1));
IG = (data(:,:,2));
IB = (data(:,:,3));
seg= and(ImaR, ImaB);
seg(100,100)= 1;
```

Figura 3.65: Segmentación por Color y Ubicación del Objeto.

Para el tracking (seguimiento) luego de tener las coordenadas del centroide, se guardan en otro espacio de memoria como se muestra en el siguiente código:

```
c1 = round(c(1));
c2 = round(c(2));

track1(1,2)= 1;
track2(1,2)= 1;
track3(1,2)= 1;

for m = 1:240;
    for n = 1:320;
        if m> c2 - 25 && m < c2 +25
            if n> c1 -25 & n< c1 + 25
                xx = m - (c2 -25);
                yy = n - (c1 -25);
                x(m,n) = 1;
                dato1 = IR(m,n);
                dato2 = IG(m,n);
                dato3 = IB(m,n);
                track1(xx,yy) = dato1;
                track2(xx,yy) = dato2;
                track3(xx,yy) = dato3;
            else
                x(m,n) = 0;
            end
        else
                x(m,n)= 0;
        end
    end
end
```

Figura 3.66: Seguimiento del Objeto.

Donde c1 y c2 son variables adquiridas de la ubicación del centroide. Se adecuan los datos para poder ser mostrados en pantalla como se muestra a continuación, y luego se incrementa el contador toma para procesar una nueva frame.

```
ImaTrack(:,:,1)=track1;
ImaTrack(:,:,2)=track2;
ImaTrack(:,:,3)=track3;

T8b = uint8(ImaTrack);
ax(1) = subplot(2,2,1);
image(data); title('Imagen de Entrada')
ax(2) = subplot(2,2,2);
imshow(seg); title('Segmentación')
ax3(3)= subplot(2,2,3);
image(T8b); title('Tracking')
linkaxes(ax,'XY')
axis(ax,'image')

coordx(toma) = c1;
coordy(toma) = c2;

toma=toma+1;

end
```

Figura 3.67: Ubicación del Centroide.

Cuando se acaban de procesar los números de frame que se determinó, se cierra el canal de video. El centroide de un objeto está definido como la parte del cuerpo donde se distribuye la masa y el centro de gravedad, el cual se puede ver como su punto de equilibrio. El centroide de un cuerpo es un concepto totalmente geométrico. Su posición solo depende de la geometría del cuerpo, y no de sus propiedades físicas (densidad, homogeneidad, peso específico, entre otras). En nuestro caso la ubicación del centroide del objeto está definida como:

$$\overline{X}=\frac{\sum \widetilde{X}_i * A_i}{\sum A_i}$$

$$\overline{Y}=\frac{\sum \widetilde{Y}_i * A_i}{\sum A_i}$$

donde A = Area. Un algoritmo para ejecutar las operaciones anteriores es:

```
im = Data;
[rows,cols] = size(im);
x = ones(rows,1)*[1:cols];
y = [1:rows]'*ones(1,cols);
area = sum(sum(im));
meanx = sum(sum(double(im).*x))/area;
meany = sum(sum(double(im).*y))/area;
```

Figura 3.68: Algoritmo Ubicación del Centroide.

La ubicación del centroide también esta implementada en la función 'regionprops' (Centroid). Para una implementación más práctica de ubicación del centroide teniendo en cuenta que la imagen a identificar es geométrica y el centroide de un círculo coinciden con su centro se implementó un algoritmo de centroide sin tener que procesar toda la imagen sino a lo que se va recibiendo o leyendo la imagen se determina si hay objeto o no.
El algoritmo se implementó mediante una matriz N x N con validación vertical, esto quiere decir que se ubica un objeto con la matriz y para este caso se implementó una matriz de 3 x 3 [111 ; 111 ; 111], donde se crea una bandera Auxlin cuya función es encontrar una línea = '111'. Cuando esto sucede pasamos a la siguiente fila y se realiza nuevamente la misma validación línea = '111' y si se valida se incrementa nuevamente la fila y se realiza la última validación. Si cualquiera de las validaciones son cero se devuelve la fila y se incrementa un píxel horizontal para seguir leyendo la imagen. Si se valida solo la matriz se determina su centro trazando una línea vertical desde el primer punto donde se encontró el objeto. Cuando sea cero, el centroide estará definido como la media de las coordenadas dadas de la línea vertical como se observa en la figura 3.69 donde los puntos rojos representan la matriz de ubicación del objeto y la línea roja para encontrar el centroide.

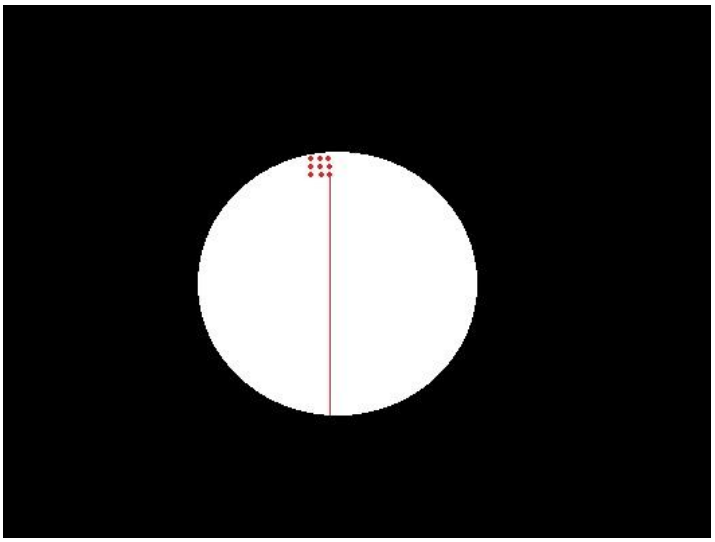


Figura 3.69: Ubicación del centroide en PC.

El código implementado para encontrar el centroide es el siguiente:

```
function [c1,c2] = centroide(M)

[nfilasM, ncolsM]=size(M);
i = 1;
j = 1;
bandera = 1;
while i < (ncolsM +1) && (j < nfilasM +1)
    if M(j,i)&&M(j,i+1)&&M(j,i+2)
        Auxlin = 1;
    else
        Auxlin = 0;
    end;
    if bandera == 1
        if Auxlin == 1
            bandera = 3;
            i = i;       j = j+1;
        else
            if i < nfilasM
                bandera = 1;
                i = i+1;    j = j;
            else
                bandera = 2;
            end
        end
    elseif bandera == 2
        if j < ncolsM
            bandera = 1;
            j = j+1;  i = 1;
        else
            bandera = 9;
        end
    elseif bandera == 3
        if Auxlin == 1
            bandera = 4;
            j = j+1;   i = i;
        else
            bandera = 1;
            i = i+1;  j = j-1;
        end
    elseif bandera == 4
       if Auxlin == 1
            bandera = 5;
            C2i = j;    C1i = i;
       else
            bandera = 1;
            i = i+1;    j = j-2;
       end
    elseif bandera == 5
            if M(j,i) == 1
                bandera = 5;
                j = j+1;
            else
                bandera = 9;
                C2f = j;  C1f = i;
            end
    elseif bandera == 9
        i = nfilasM;    j = ncolsM;
    else
        i = nfilasM+10; j = ncolsM+10;
    end
end
C1 = C1f+4;
C2 = round((C2i+C2f)/2)-2;
c2 = C2;
c1 = C1;
```

Figura 3.70: Función Implementada del centroide en PC.

4. RESULTADOS

En este capítulo se muestran los resultados obtenidos al realizar las cinco pruebas descritas en el numeral 1.4.2.

4.1. PRUEBAS DE ILUMINACIÓN

Estas pruebas se realizaron con un luxómetro Extech Instruments Light Meter con un escala de iluminación de 0 a 1999 en modo Fast como se puede observar en la figura 4.1:

Figura 4.1: Luxómetro

Se realizaron tres pruebas al sistema de iluminación dividiendo el background en 9 puntos verticales y 11 puntos horizontales obteniendo 99 puntos de toma de datos como se observa en la figura 4.2:

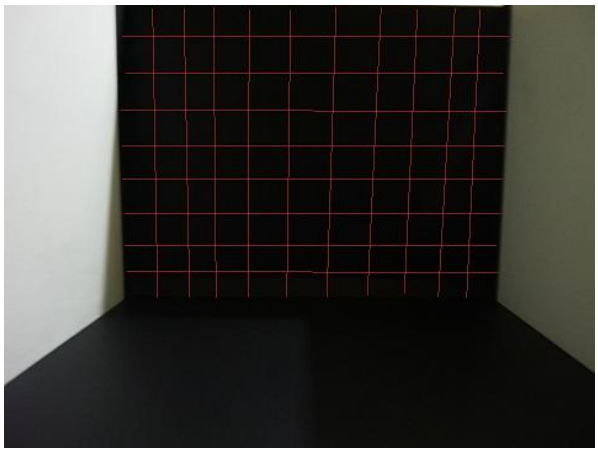


Figura 4.2: División del Background para Pruebas

Estas pruebas se realizaron de la siguiente manera:

- Conexión con Leds
- Conexión con Lámparas
- Conexión con Leds y Lámparas.

En la tabla 4.1 se muestran los datos obtenidos de la conexión de iluminación con los Leds:

	V1	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**	**V8**	**V9**	**V10**	**V11**
H1	20	26	28	27	29	27	28	29	30	31	25
H2	22	27	29	31	32	34	32	32	33	32	23
H3	22	27	24	20	22	23	22	24	24	31	24
H4	22	28	24	21	31	33	32	22	25	31	24
H5	23	27	24	20	32	17	31	23	24	31	23
H6	22	27	25	21	30	30	31	22	24	32	24
H7	22	27	24	22	23	24	23	23	26	31	24
H8	20	26	28	30	31	32	31	30	29	32	24
H9	20	26	27	26	28	27	28	28	29	32	24

Tabla 4.1: Iluminacion con Leds

En la tabla 4.2 se muestran los datos obtenidos de la conexión con las Lámparas:

	V1	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**	**V8**	**V9**	**V10**	**V11**
H1	195	206	213	222	227	235	231	224	213	205	194
H2	218	234	241	245	247	250	247	242	236	225	210
H3	223	238	248	255	256	261	255	255	248	236	218
H4	235	238	249	256	259	262	257	255	248	237	219
H5	236	239	249	257	259	262	258	256	249	237	218
H6	253	261	263	268	274	279	273	270	263	256	251
H7	255	257	265	269	275	279	277	271	266	260	254
H8	257	263	267	270	274	277	275	271	268	263	257
H9	259	267	270	271	273	275	274	272	271	268	260

Tabla 4.2: Iluminacion con Lamparas

En la tabla 4.3 se muestran los datos obtenidos de la conexión con los Leds y las Lámparas:

	V1	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**	**V8**	**V9**	**V10**	**V11**
H1	210	219	226	234	247	252	248	236	226	218	210
H2	226	232	243	252	258	269	259	255	245	234	224
H3	234	238	246	257	262	272	263	259	247	241	238
H4	239	243	252	262	268	275	269	263	254	246	240
H5	241	249	255	265	273	278	274	267	255	250	243
H6	224	238	248	256	257	260	256	254	247	236	219
H7	216	232	239	243	245	248	245	241	235	230	214
H8	198	216	229	231	236	238	235	229	219	213	196
H9	196	214	227	229	235	237	233	227	218	211	194

Tabla 4.3: Iluminacion con Leds y Lamparas

donde H es Punto Horizontal y V es Punto Vertical.
Realizando un análisis gráfico de manera tridimensional podemos observar los diferentes comportamientos de iluminación. Este análisis se muestra en las figuras 4.3 y 4.4:

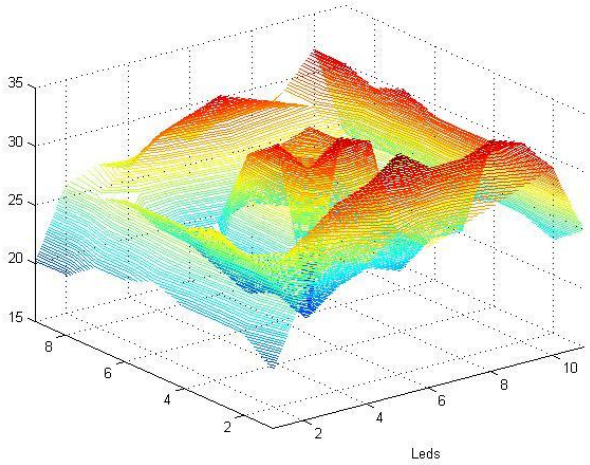


Figura 4.3: Comportamiento con Leds

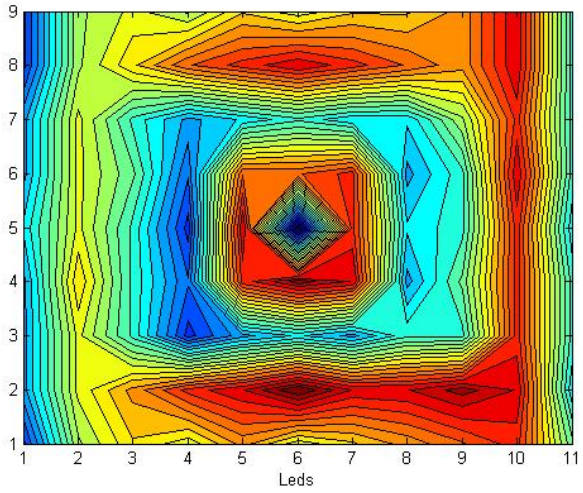


Figura 4.4: Flujo Luminoso con Leds

De la figura anterior se puede concluir que el flujo luminoso (medida de la potencia luminosa percibida) presenta picos en ciertas zonas del background (bordes y alrededores del centro) y en otras zonas cercanas hay niveles bajos formando sombras como se observa en el centro. La potencia luminosa además es variada en toda la superficie. Este sistema de iluminación no fue utilizado ya que los niveles de iluminación son bajos presentando un máximo de 33 lumen. Además se debe tener un mejor control en las variables como sombras y los tonos de color los cuales generan pérdida de la información en las características visuales. El valor promedio de luminicidad utilizando los leds es de 26.4949 lumens. Realizando un análisis de la desviación estándar por fila obtenemos

los siguientes valores: [2,9695 4,0519 2,9480 4,4782 4,7749 3,9954 2,6594 3,7513 3,0271] y la matriz de correlación es:

$$\begin{bmatrix} 1 & 0{,}8295 & 0{,}3800 & 0{,}3841 & 0{,}3667 & 0{,}4421 & 0{,}4892 & 0{,}8674 & 0{,}9738 \\ 0{,}8295 & 1 & 0{,}0395 & 0{,}4844 & 0{,}0982 & 0{,}4419 & 0{,}1889 & 0{,}9497 & 0{,}8189 \\ 0,3800 & 0,03957 & 1 & 0,3078 & 0,3836 & 0,4345 & 0,9496 & 0,1307 & 0,5134 \\ 0,3841 & 0,4844 & 0,3078 & 1 & 0,4536 & 0,9709 & 0,3931 & 0,5644 & 0,5036 \\ 0,3667 & 0,0982 & 0,3836 & 0,4536 & 1 & 0,5870 & 0,3858 & 0,2009 & 0,4289 \\ 0,4421 & 0,4419 & 0,4345 & 0,9709 & 0,5870 & 1 & 0,4996 & 0,5610 & 0,5734 \\ 0,4892 & 0,1889 & 0,9496 & 0,3931 & 0,3858 & 0,4996 & 1 & 0,2578 & 0,6199 \\ 0,8674 & 0,9497 & 0,1307 & 0,5644 & 0,2009 & 0,5610 & 0,2578 & 1 & 0,8709 \\ 0,9738 & 0,8189 & 0,5134 & 0,5036 & 0,4289 & 0,5734 & 0,6199 & 0,8709 & 1 \end{bmatrix}$$

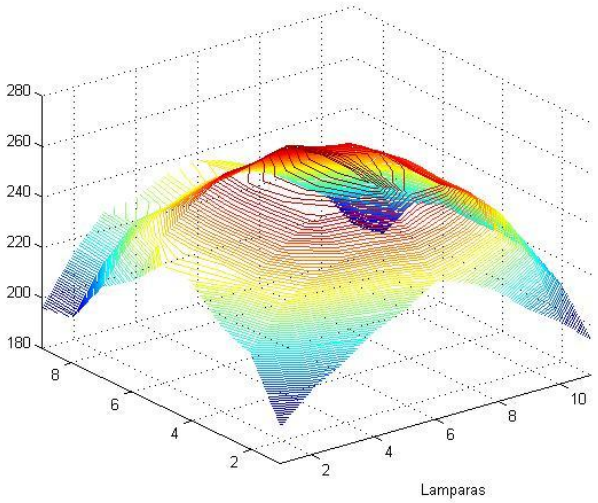


Figura 4.5: Comportamiento con Lámparas

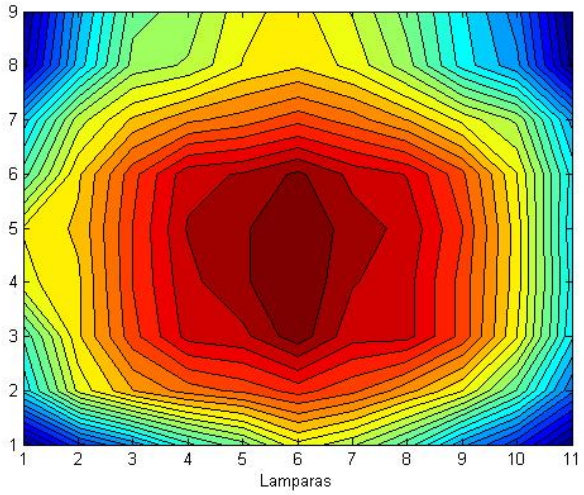


Figura 4.6: Flujo Luminoso con Lámparas

El comportamiento al realizar el sistema de iluminación con lámparas es más uniforme, los valores obtenidos se concentran formando un paraboloide elíptico, y esto se puede explicar pues las lámparas están paralelas al background en dos de sus extremos como se describió anteriormente en la sección 3.2 y cada una de ellas aporta su nivel de iluminación en la superficie como se muestra en la figura 4.7 :

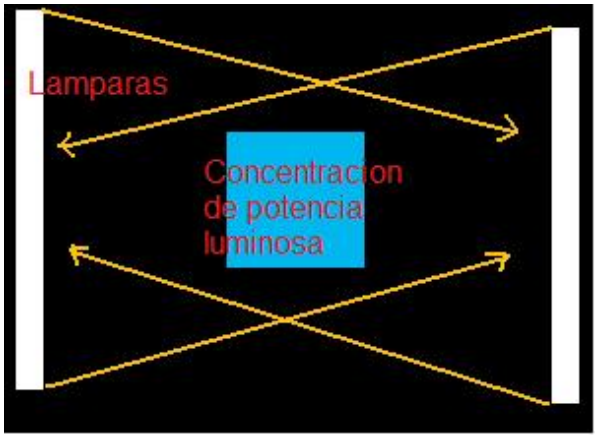


Figura 4.7: Iluminación en el Background

concentrando de esta forma la potencia luminosa en el centro del background, el nivel de iluminación es aceptable presentando un valor mínimo de 194 y un máximo de 275 lumen. El valor promedio de luminicidad utilizando las lámparas es de 234.6667 lumens. Realizando un análisis de la desviación estándar por fila obtenemos los siguientes valores: [14 13,0572 14,2745 13,1439 13,4319 13,9713 11,4899 14,7568 14,9496] y la matriz de correlación es:

$$\begin{bmatrix} 1 & 0,9458 & 0,9607 & 0,9466 & 0,9380 & 0,9647 & 0,9573 & 0,9603 & 0,9622 \\ 0,9458 & 1 & 0,9822 & 0,9688 & 0,9666 & 0,9855 & 0,9860 & 0,9849 & 0,9851 \\ 0,9607 & 0,9822 & 1 & 0,9719 & 0,9677 & 0,9983 & 0,9886 & 0,9801 & 0,9818 \\ 0,9466 & 0,9688 & 0,9719 & 1 & 0,9988 & 0,9752 & 0,9512 & 0,9515 & 0,9537 \\ 0,9380 & 0,9666 & 0,9677 & 0,9988 & 1 & 0,9708 & 0,9448 & 0,9426 & 0,9445 \\ 0,9647 & 0,9855 & 0,9983 & 0,9752 & 0,9708 & 1 & 0,9904 & 0,9850 & 0,9862 \\ 0,9573 & 0,9860 & 0,9886 & 0,9512 & 0,9448 & 0,9904 & 1 & 0,9941 & 0,9941 \\ 0,9603 & 0,9849 & 0,9801 & 0,9515 & 0,9426 & 0,9850 & 0,9941 & 1 & 0{,}9995 \\ 0,9622 & 0,9851 & 0,9818 & 0,9537 & 0,9445 & 0,9862 & 0,9941 & 0{,}9995 & 1 \end{bmatrix}$$

Cuando utilizamos el sistema de iluminación con Leds y Lámparas el flujo luminoso se hace más uniforme como se observa en la figura 4.8 y 4.9:

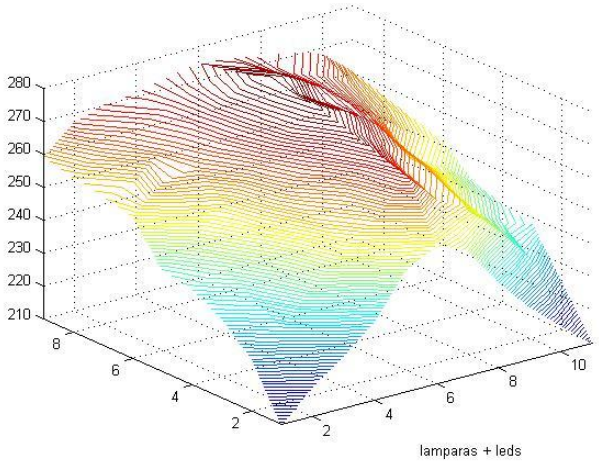


Figura 4.8: Comportamiento con Leds y Lámparas

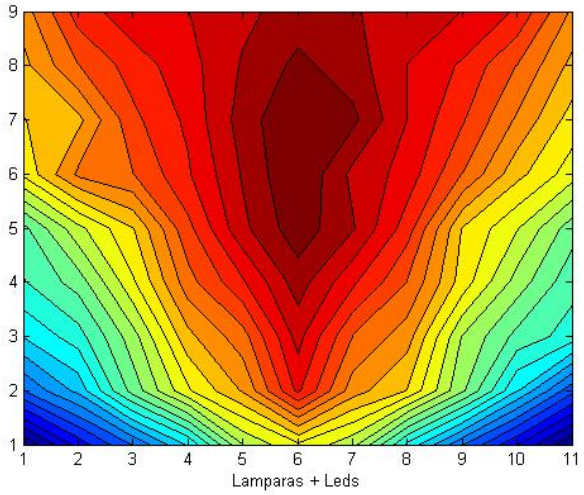


Figura 4.9: Flujo Luminoso con Leds y Lámparas

La potencia luminosa se concentra en la parte superior del background y se difumina en las partes inferiores hacia los bordes exteriores. El valor promedio de luminicidad utilizando las lámparas es de 234.6667 lumens. Realizando un análisis de la desviación estándar por fila obtenemos los siguientes valores: [14,9818 14,7838 12,5401 12,6124 12,9418 9,0473 8,8748 6,8463 5,3000] y la matriz de correlación es:

$$\begin{bmatrix} 1 & 0,9809 & 0,9764 & 0,9881 & 0,9945 & 0,9838 & 0,9888 & 0,9835 & 0,9046 \\ 0,9809 & 1 & 0,9831 & 0,9937 & 0,9851 & 0,9821 & 0,9905 & 0,9870 & 0,9237 \\ 0,9881 & 0,9937 & 0,9934 & 1 & 0,9939 & 0,9737 & 0,9933 & 0,9800 & 0,9012 \\ 0,9466 & 0,9688 & 0,9719 & 1 & 0,9988 & 0,9752 & 0,9512 & 0,9515 & 0,9537 \\ 0,9945 & 0,9851 & 0,9879 & 0,9939 & 1 & 0,9799 & 0,9871 & 0,9813 & 0,9066 \\ 0,9838 & 0,9821 & 0,9620 & 0,9737 & 0,9799 & 1 & 0,9685 & 0,9812 & 0,9141 \\ 0,9888 & 0,9905 & 0,9773 & 0,9933 & 0,9871 & 0,9685 & 1 & 0,9859 & 0,9159 \\ 0,9835 & 0,9870 & 0,9583 & 0,9800 & 0,9813 & 0,9812 & 0,9859 & 1 & 0,9632 \\ 0,9046 & 0,9237 & 0,8671 & 0,9012 & 0,9066 & 0,9141 & 0,9159 & 0,9632 & 1 \end{bmatrix}$$

Después realizamos un análisis de los datos obtenidos en las matrices donde se tienen en cuenta el espacio, el promedio de las medias, los límites superior e inferior de control y la desviación estándar de los datos. Cuando se presentan valores fuera de los límites de control se marcan estos datos como violaciones las cuales se muestran en círculos rojos en las gráficas 4.10, 4.11 y 4.12:

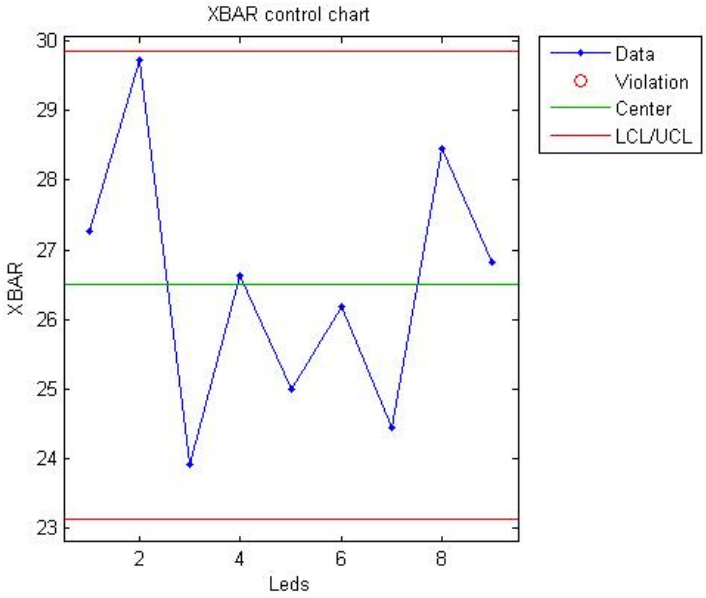


Figura 4.10: Análisis de Datos de la Matriz con Leds

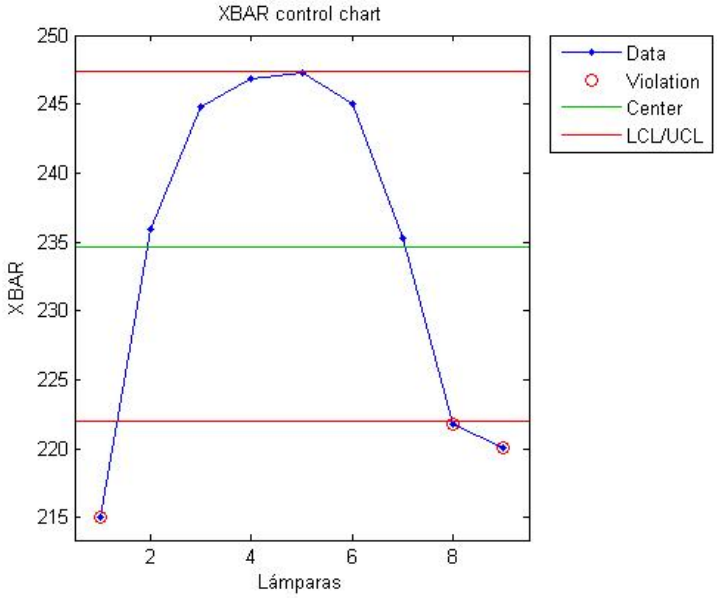


Figura 4.11: Análisis de Datos de la Matriz con Lámparas

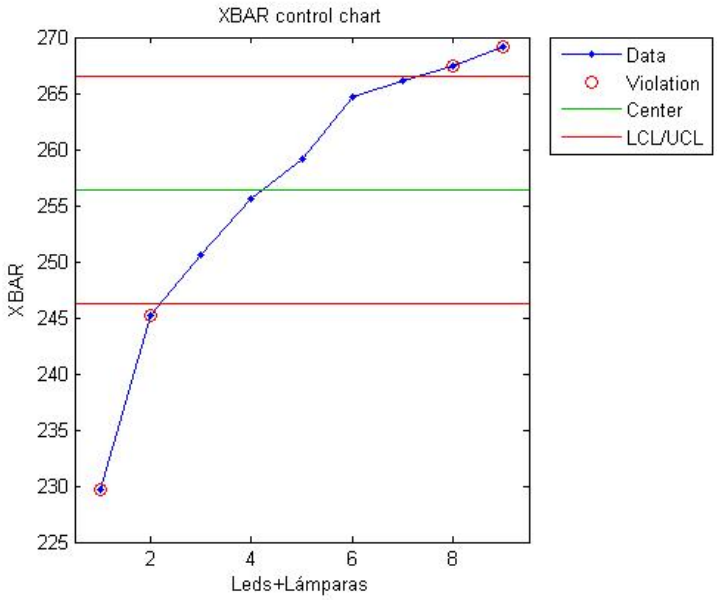


Figura 4.12: Análisis de Datos de la Matriz con Leds y Lámparas

Con los análisis de correlación y de datos realizado a los tres tipos de sistemas de iluminación podemos concluir que el más adecuado es con Lámparas, teniendo en cuenta de no usar los bordes exteriores laterales donde se observa violación de datos.

4.2. PRUEBAS ESTÁTICAS DE SEGUIMIENTO

4.2.1. PRUEBAS ESTÁTICAS EN EL PC Las pruebas estáticas nos ayudan a medir la variable de la posición del centroide cuya valoración está dada por la permanencia de las coordenadas es decir el cambio de las coordenadas al permanecer el objeto estático en un solo punto. Con estas pruebas se mide la velocidad de procesamiento y la eficiencia del seguimiento estando el objeto estático en los diferentes puntos de prueba del background. Se contará para esto con 9 puntos de pruebas distribuidos en toda el área del background. En la figura 4.13 podemos observar el video de entrada luego de realizar la adquisición con la tarjeta de video LR37 REV:B a través de la entrada S-Video y en la cual marcamos los puntos de prueba que se tuvieron en cuenta al realizar las pruebas estáticas:

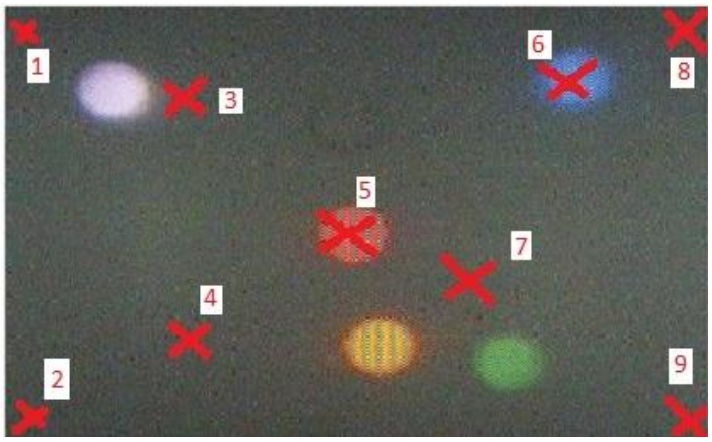


Figura 4.13: Adquisición de Imágenes con Tarjeta de Video LR37

Para cada punto se realizaron 5 pruebas en las que se tomaron 30 frames de video midiendo el tiempo de procesamiento total y la posición con las coordenadas del centroide en cada frame. El sistema realizó el seguimiento en los puntos 1, 2, 3, 4, 5, 6 y 7 como se muestra en la figura 4.14:

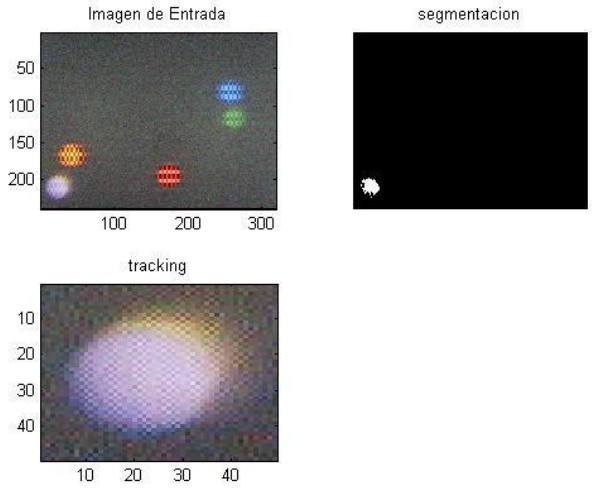


Figura 4.14: Seguimiento para el Punto de Prueba No. 2

Para el tracking en los extremos del background se presentan problemas de sobreposición pues lo que hace es un seguimiento recortando el objeto de la matriz donde se guardo el frame procesado lo cual podemos observar en la figura: 4.15:

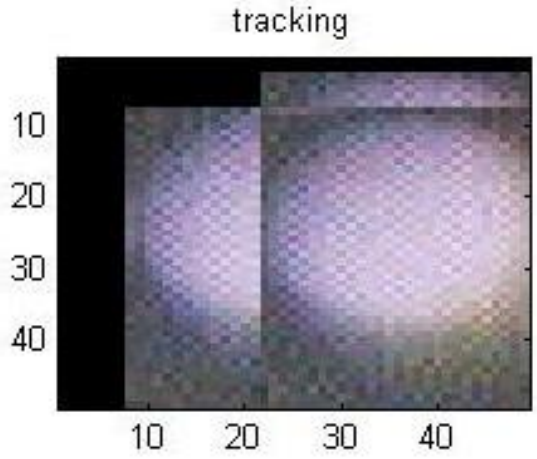


Figura 4.15: Error en el Seguimiento Punto de Prueba No. 1

Este inconveniente solo se presenta para los Puntos de Prueba No. 1 y 2. Los niveles de segmentación para el sistema de seguimiento implementado en PC y en FPGA fueron iguales con valores de R>180, B>180 y G>150 pero para el sistema en PC se alcanza a identificar puntos en imágenes rojas que se definen como puntos de alto brillo en el color rojo, por lo que se vió necesario para una mejor identificación subir los niveles de segmentación del sistema en el espacio rojo a R>210 y el espacio azul a B>190. En la gráfica de probabilidad normal podemos observar este error.

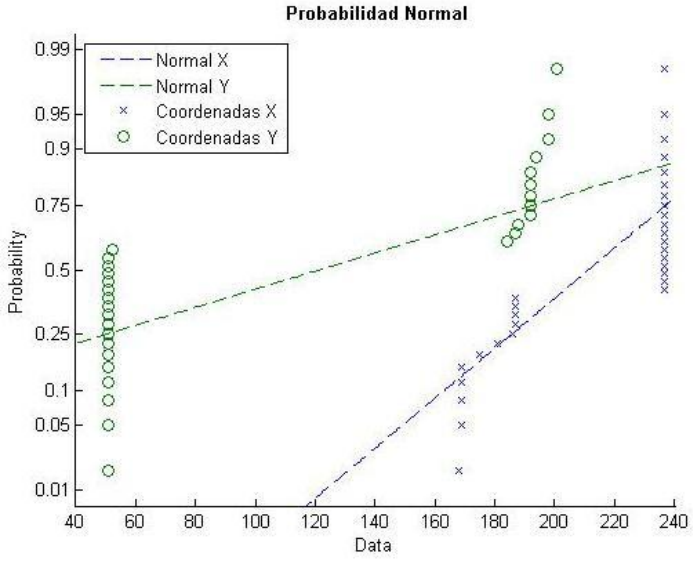


Figura 4.16: Punto de Prueba No. 6 sin mejorar segmentación.

En la figura 4.16 se observa que no se realiza una ubicación de coordenadas de forma eficiente y se presentan errores al identificar el objeto por otro. Al umbralizar el espacio

de color como se menciono anteriormente se obtuvo el siguiente mejoramiento:

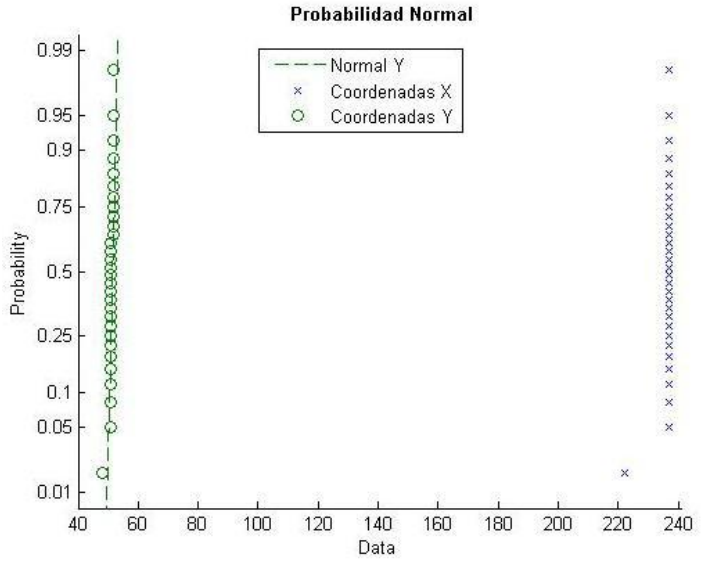


Figura 4.17: Punto de Prueba No. 6 mejorando segmentación

En la figura 4.17 se puede observar que los datos se mantienen sobre una misma línea de tendencia o normal, lo que nos indica que se realizó la ubicación del objeto y se definió de forma adecuada el centroide del objeto al cual se le está realizando el seguimiento. Para analizar el tipo de distribución de los datos se realizaron histogramas para definir la frecuencia de los datos, un ejemplo de esto se observa en la figura 4.18:

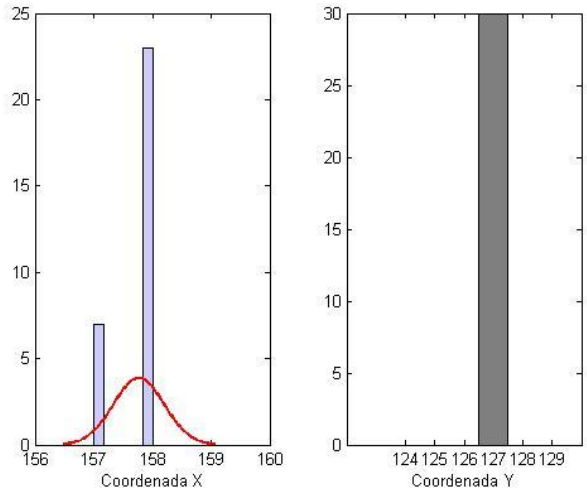


Figura 4.18: Histograma para el Punto de Prueba No. 5

De lo anterior podemos construir una gráfica de distribución de probabilidad de los

resultados teniendo en cuenta que por el tipo de datos tenemos una distribución de variable discreta ya que los datos sólo toma valores positivos en un conjunto de valores de X finito definido por el espacio máximo de la memoria donde se guarda las frame de video para su procesamiento. En la siguiente gráfica podemos observar la gráfica de densidad estimada:

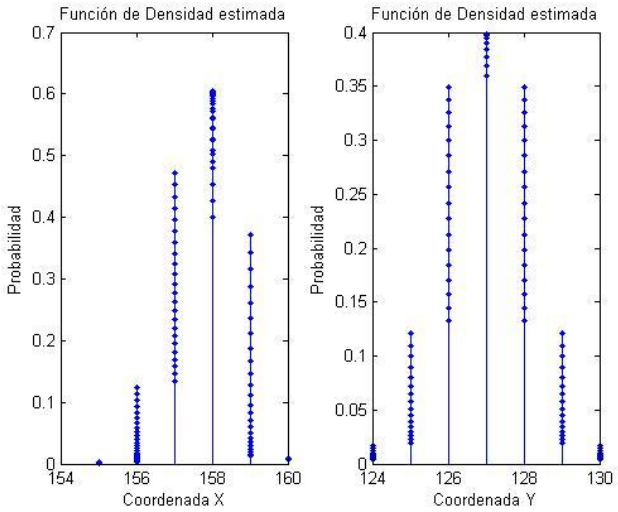


Figura 4.19: Función de Densidad Estimada para el Punto de Prueba No. 5.

De la figura 4.19 podemos decir que la ubicación del centroide varía y esto es debido en gran parte a la segmentación, en los 7 puntos de prueba. El incoveniente de no tener una segmentación uniforme se da por problemas visuales con la cámara, cambios entre frames y a la variación de la iluminación con el tiempo entre otros.

Debido a la variabilidad de la ubicación del centroide para poder realizar medición de la variable de la posición del centroide del objeto se adopto la medición de la eficiencia del seguimiento haciendo uso de un intervalo de confianza para poder procesar la frecuencia de los datos y realizar un análisis de los mismos.

En los puntos 8 y 9 del sistema en PC no se realizó seguimiento, esto debido principalmente a que en este borde se tiene un defecto provocado entre la cámara y su estándar NTSC de transmisión. En este borde del background se tiende a difuminar el video, presentándose cambio de color y opacidad en el video por problemas en el lente de la cámara. Todos estos problemas inciden en el proceso de segmentación por lo que no se puede realizar el seguimiento al no poder validar la matriz de comparación realizada en el algoritmo del centroide descrito en la sección 3.3.2.

Figura 4.20: Video de entrada en PC Punto de Prueba No. 8

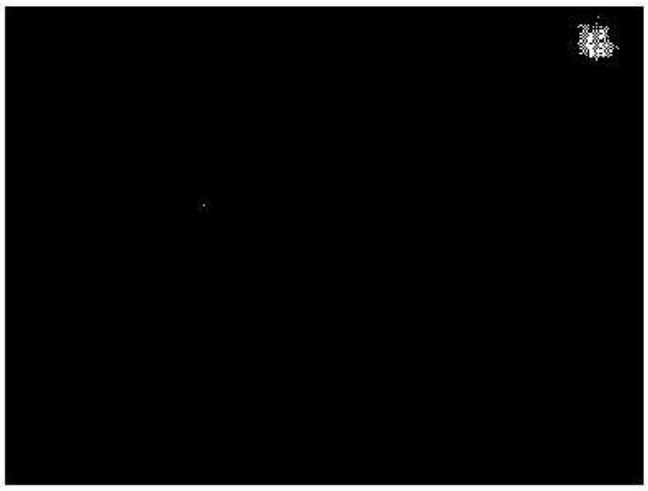


Figura 4.21: Problema de Segmentación en PC Punto de Prueba No. 8

Figura 4.22: Video de entrada en PC Punto de Prueba No. 9.

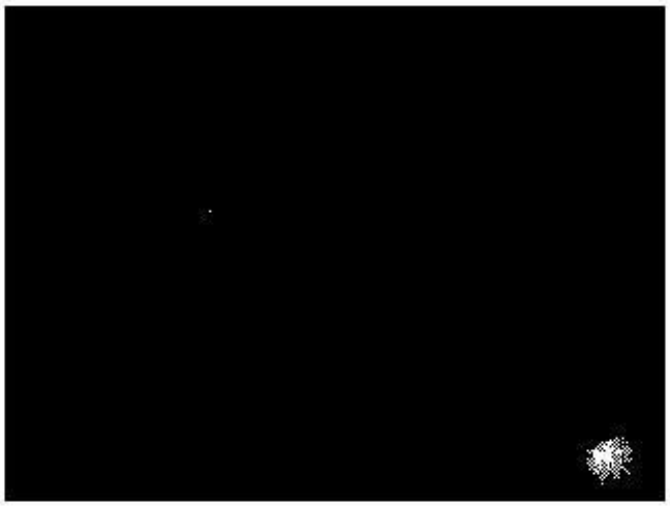


Figura 4.23: Problema de Segmentación en PC Punto de Prueba No. 9.

En general el tiempo promedio total de 1 y de 30 frame en cada prueba se muestran en la tabla 4.4:

	Tiempo Total Promedio (s)	Tiempo Frame Promedio (ms)
Punto Prueba 1	5.519,60	189,9866
Punto Prueba 2	5.501,20	183,3733
Punto Prueba 3	5.532,4	184,4133
Punto Prueba 4	5.570,80	185,6933
Punto Prueba 5	5.540,60	184,6866
Punto Prueba 6	5.510,80	183,6933
Punto Prueba 7	5.531,40	184,38
Punto Prueba 8	5.561,40	185,38
Punto Prueba 9	5.627,20	187,5733

Tabla 4.4: Tiempo Total y Frame Promedio para PC

La tasa de adquisición para un video NTSC es de 30 frames por segundo y de acuerdo con los datos obtenidos del sistema de PC solo se logra procesar teniendo el mínimo de tiempo promedio por frame de 183,693333 ms y solo 5,4644 frames por segundo y con el máximo tiempo promedio el tiempo para procesar un frame es de 189,986 ms y en un segundo solo podemos procesar 5,26 frames por lo que el sistema implementado en PC solo procesa aproximadamente 5 frames por segundo. Para determinar la posición central del centroide que está realizando el seguimiento se saco la moda (valor que hace máxima la función de probabilidad) en cada una de las 5 pruebas en cada punto de prueba. Teniendo la coordenada que representa el centroide del objeto se saco el porcentaje promedio de reconocimiento en cada prueba.

	Coord. X encontrada	Coord. Y encontrada	% Permanencia Coord. X (Moda)	% Permanencia Coord. Y (Moda)
Punto Prueba 1	16	15	68,66 %	87,33 %
Punto Prueba 2	25	210	47,33 %	51,33 %
Punto Prueba 3	91	47	96,66 %	89,33 %
Punto Prueba 4	107	175	99,33 %	67,33 %
Punto Prueba 5	158	127	86,00 %	100,00 %
Punto Prueba 6	237	51	91,33 %	66,00 %
Punto Prueba 7	209	150	90,66 %	77,33 %
Punto Prueba 8	0	0	0,00 %	0,00 %
Punto Prueba 9	0	0	0,00 %	0,00 %

Tabla 4.5: Permanencia de Coordenadas con Moda para PC

Con los anteriores datos se saco un intervalo de confianza para cada prueba con un nivel de confianza del 95 %, para poder medir la identificación de la posición del centroide y el reconocimiento del mismo.
El intervalo de confianza se define como un número entre los cuales se estima que estará más o menos cierto valor desconocido con una determinada probabilidad de acierto. Formalmente, estos números determinan un intervalo, que se calcula a partir de datos de las muestras, y la desviación estándar de la prueba para cada coordenada. La probabilidad de éxito en la estimación se representa por $1-\alpha$ y se denomina nivel de confianza. En estas circunstancias, α es el llamado error aleatorio o nivel de significación. El nivel de confianza y la amplitud del intervalo varían conjuntamente, de forma que un intervalo más amplio tendrá más posibilidades de acierto (mayor nivel de confianza), mientras que para un intervalo más pequeño, que ofrece una estimación más precisa aumenta la probabilidad de error.

La tabla de los porcentajes de estimación del centroide se muestra en la tabla 4.6:

	Coord. X Promedio encontrada	**Coord. Y Promedio encontrada**	**% Permanencia Coord. X (Confianza)**	**% Permanencia Coord. Y (Confianza)**
Punto Prueba 1	16	15	86,67 %	86,67 %
Punto Prueba 2	25	210	46,67 %	60,00 %
Punto Prueba 3	91	47	96,67 %	90,00 %
Punto Prueba 4	107	175	100,00 %	66,67 %
Punto Prueba 5	158	127	86,67 %	100,00 %
Punto Prueba 6	237	51	90,00 %	66,67 %
Punto Prueba 7	209	150	90,00 %	93,33 %
Punto Prueba 8	0	0	0,00 %	0,00 %
Punto Prueba 9	0	0	0,00 %	0,00 %

Tabla 4.6: Permanencia de Coordenadas con Intervalo de Confianza para PC

Se realizó adicionalmente un coeficiente de variación promedio también conocido como coeficiente de Pearson para cada punto de prueba, la cual es una medida de dispersión útil para comparar dispersiones a escalas distintas. En este caso para cada punto de prueba debido a que es una medida invariante ante cambios de escala útil para comparar variables que están en distintas escalas pero que están correlacionadas estadísticamente y sustantivamente con un factor en común. Es decir, ambas variables tienen una relación causal con ese factor. Su fórmula expresa la desviación estándar como porcentaje de la media aritmética.

$$C_v = \frac{\sigma}{\overline{x}}$$

Los coeficientes de variación promedio para cada punto de prueba se muestran en la tabla 4.7:

	Coeficiente de Variación Promedio Coordenada CX	Coeficiente de Variación Promedio Coordenada CY
Punto Prueba 1	13,7072	6,6134
Punto Prueba 2	39,4203	0,9825
Punto Prueba 3	0,1208	0,6373
Punto Prueba 4	0,0332	0,2244
Punto Prueba 5	0,2119	0,0
Punto Prueba 6	2,9894	13,712
Punto Prueba 7	2,6491	5,6372
Punto Prueba 8	No hay seguimiento	No hay seguimiento
Punto Prueba 9	No hay seguimiento	No hay seguimiento

Tabla 4.7: Coeficientes de Variacion para PC

4.2.2. PRUEBAS ESTÁTICAS EN EL DISPOSITIVO DE LÓGICA PROGRAMABLE

De forma teórica tenemos que para adquirir una línea de video se requiere de 1716 ciclos de reloj LLC (27 MHz) para el estándar NTSC y son 525 líneas de video para un total de 900900 ciclos de reloj, esto quiere decir que cada adquisición de un frame se realiza en 33,4 ms por lo tanto en un segundo se realizan 1/0,0334 muestras de video esto quiere decir 29,94 ms aproximadamente 30 frames por segundo, luego de la adquisición se realiza el procesamiento del video de dimensiones 380 x 252 para un total de 95760 píxeles, la frecuencia del reloj para el procesamiento se implementó a través del DCM. La lectura del video se realiza en 992 ciclos de reloj para una línea y 525 líneas totales para un total de 520800 ciclos de reloj. La frecuencia de lectura es de 50MHz por lo que un frame de video es leído en 10,4 (ms).
Para poder realizar el procesamiento se debe primero leer el video, luego hacer el procesamiento y finalmente enviarla a pantalla, realizando un análisis de tiempos se tiene que para las condiciones antes mencionadas el tiempo mínimo del sistema es de 33,4 ms + tiempo de procesamiento + 10,414 ms. El tiempo de procesamiento es dependiente de los procesos de segmentación, filtrado y ubicación del objeto, de lo que podemos concluir que el tiempo mínimo para estas condiciones es:

Tiempo de Procesamiento Total = Tiempo de Procesamiento + 43,814 ms.

De forma práctica para mejorar el tiempo de procesamiento total en la lectura del video de cada frame se tomó un solo campo de los dos sacrificando la definición del video pero realizando el proceso de lectura en 1716 ciclos de reloj por línea con 252 líneas para un total de 432432 ciclos de reloj. Como la lectura se realiza a 27MHz el video es leída en 16 ms de los 33,4 ms del total de la lectura de los dos campos, ya adquirido un frame el tiempo del segundo frame se usa para realizar procesamiento y parte del proceso de lectura. El reloj para el módulo de procesamiento se implementó por medio del DCM mejorando el tiempo de procesamiento del video, para una frecuencia de 100MHz

teniendo que el video guardado en la RAM es de 380 x 252 tenemos un tiempo de procesamiento de 0.957 ms y el tiempo de lectura se realiza en 10,4 ms. Para garantizar la escritura, procesamiento y lectura de un frame completo se creó banderas de espera cuya función es realizar todo el proceso del sistema de forma secuencial y el tiempo máximo para realizar todo el proceso será:

2(16ms) + 0,957ms + 2(10,4ms) = 53,757 ms suponiendo el peor de los casos en el que nos tocara esperar hasta la primera línea y fila para cada proceso de escritura y lectura.

El tiempo mínimo es: 16ms + 0,957ms + 10,4ms = 27,357 ms suponiendo el mejor de los casos en el que un proceso termina antes de la primera línea del otro.
De lo anterior se puede decir que los tiempos obtenidos en las pruebas deben de estar en el rango de [27,357 a 53,757] ms.

En la tabla: 4.8 se tienen los tiempos obtenidos con el sistema en el dispositivo lógico programable de los puntos de prueba definidos en la figura 4.13.

	Tiempo Total Promedio (ms)	**Tiempo Frame Promedio (ms)**
Punto Prueba 1	1122,240	37,408
Punto Prueba 2	1269,200	42,30666667
Punto Prueba 3	1222,240	40,74133333
Punto Prueba 4	1175,680	39,18933333
Punto Prueba 5	1175,680	39,18933333
Punto Prueba 6	1122,240	37,408
Punto Prueba 7	1189,040	39,63466667
Punto Prueba 8	1155,640	38,52133333
Punto Prueba 9	1329,320	44,31066667

Tabla 4.8: Tiempo Total y Frame Promedio para FPGA

Podemos observar que todos los tiempos obtenidos estuvieron dentro del umbral definido anteriormente como se esperaba y que el sistema responde en tiempos por debajo del umbral máximo. De acuerdo con los datos obtenidos el sistema en la FPGA solo logra procesar 26,73 frames por segundo de 30 frames de adquisición en un segundo teniendo el mínimo de tiempo promedio por frame de 37,408 ms, y con el máximo tiempo promedio el tiempo para procesar un frame obtenido es de 44,31 ms y en un segundo solo es posible procesar 5,26 frames por lo que el sistema implementado en FPGA procesa aproximadamente 22,5 frames por segundo, y considerando que se presente el tiempo máximo de procesamiento tendriamos 22 frames procesadas de 30.
Para obtener los anteriores tiempos se implementó un temporizador el cual cuenta cada dos tiempos totales de procesamiento de frames. Este se observa como un punto visual

de donde se puede extraer la información de tiempo midiendo el tiempo de inicio y de fin en la herramienta de análisis de cada video de prueba obtenido. También se obtuvieron las coordenadas para definir la eficiencia de la variable de la posición del centroide cuya valoración está dada como definimos anteriormente como la permanencia de las coordenadas.
De igual forma que para el sistema de seguimiento en PC para cada punto de prueba se realizaron 5 pruebas en las que se tomo 30 frames procesadas, midiendo el tiempo de procesamiento con el contador definido anteriormente y tomando las coordenadas del centroide en cada frame. En la figura 4.24 podemos ver el formato de prueba que se realizo para la toma de datos:

FORMATO PRUEBAS ESTATICAS

Punto Prueba	2	Numero de Prueba	1	PC ___	FPGA__x_	100 MHz	Seguimiento		
Contador XX proceso (1)	Tiempo Evento (s:ms)	Frame Evento	Numero de Frames	Tiempo por 2 Frames	Coordenada X	Coordenada Y	Centro	Desplazado	Nulo
0	0	0	1	66,8	26	209	x		
0	33,4	1			26	209	x		
1	66,8	2	3	100,2	26	209	x		
1	133,6	4			26	209	x		
0	167	5	3	100,2	26	209	x		
0	233,8	7			25	200	x		
1	267,2	8	3	100,2	25	200	x		
1	334	10			23	212	x		
0	367,4	11	3	100,2	32	200	x		
0	434,2	13			32	200	x		
1	467,6	14	2	66,8	32	200	x		
1	501	15			32	200	x		
0	534,4	16	2	66,8	27	211	x		
0	567,8	17			27	211	x		
1	601,2	18	4	133,6	27	211	x		
1	701,4	21			34	200	x		
0	734,8	22	2	66,8	34	200	x		
0	768,2	23			34	200	x		
1	801,6	24	2	66,8	34	200	x		
1	835	25			25	200	x		
0	868,4	26	3	100,2	26	200	x		
0	935,2	28			29	200	x		
1	968,6	29	3	100,2	29	200	x		
1	1035,4	31			29	211	x		
0	1068,8	32	2	66,8	29	211	x		
0	1102,2	33			34	200	x		
1	1135,6	34	2	66,8	24	200	x		
1	1169	35			24	209	x		
0	1202,4	36	3	100,2	24	209	x		
0	1269,2	38			24	209	x		
Tiempo Total 30 Frames				1302,6					
Tiempo Promedio (ms)				43,42					

Figura 4.24: Formato de Pruebas para Datos en FPGA

El sistema realizó el seguimiento en todos los puntos de prueba, en la figura 4.25 se muestra como se realizó el seguimiento estático del objeto en el dispositivo lógico programable obteniendo el siguiente resultado:

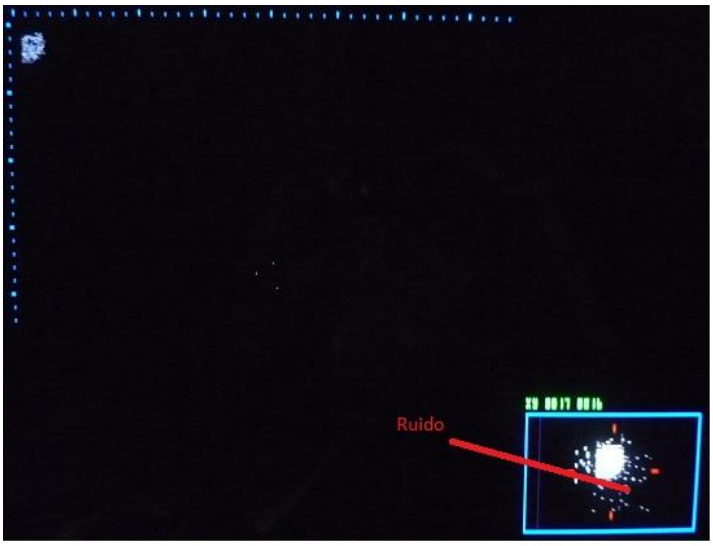


Figura 4.25: Seguimiento Estático en el Punto de Prueba No. 1

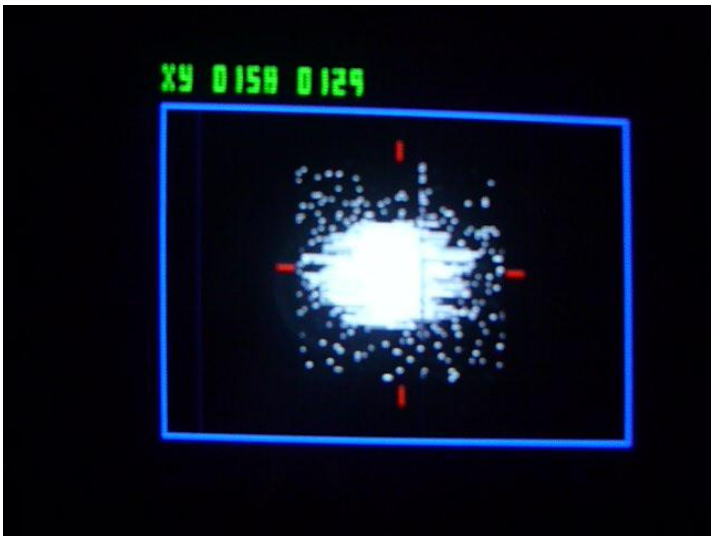


Figura 4.26: Tracking en el Punto de Prueba No. 5

En la anterior figura podemos observar el ruido en la adquisición del video en el tracking ya que es la visualización del video adquirido de la cámara sin procesar, la cual es filtrada en el procesamiento en el contorno de visualización del video de la figura 4.25. Se hizo uso de análisis gráfico para observar posibles fallas en el seguimiento del objeto, de lo que se identifico que en los puntos 1, 2, 3, 4, 5, 6, 7 el seguimiento se realizó sin presentarse una considerable variación en las coordenadas, como se muestra en la figura 4.27

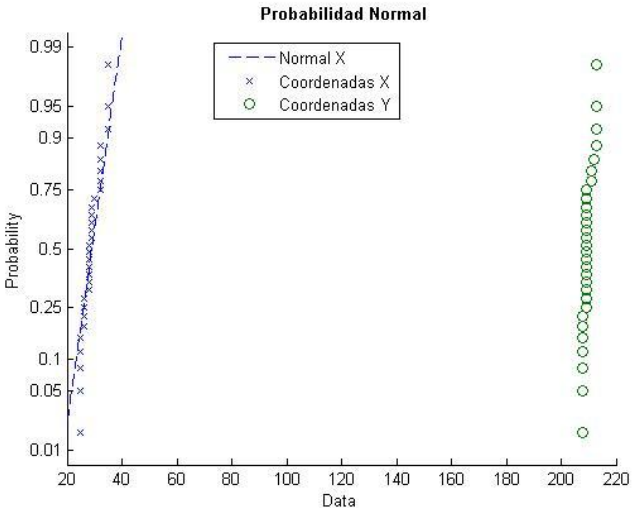


Figura 4.27: Probabilidad Normal en el Punto de Prueba No. 2

En los puntos 8 y 9 se presentó que la posición del objeto tiende a variar en la coordenada X pero realizando el seguimiento definiendo el centroide del objeto en el centro del recuadro del tracking como se observa en la figura 4.28:

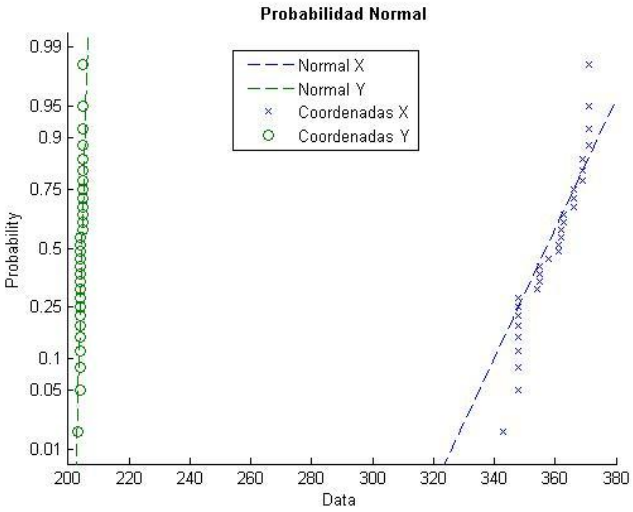


Figura 4.28: Probabilidad Normal en el Punto de Prueba No. 9

Esta variación es debida a que el objeto en el borde derecho de la cámara presenta problemas con el lente opacando el objeto como se muestra en la figura 4.29:

Figura 4.29: Tracking en el Punto de Prueba No. 9

En esta figura no es tan definido como se observa en la figura 4.26. También se obtuvieron gráficas de densidad de procesamiento como se muestra en la figura 4.30

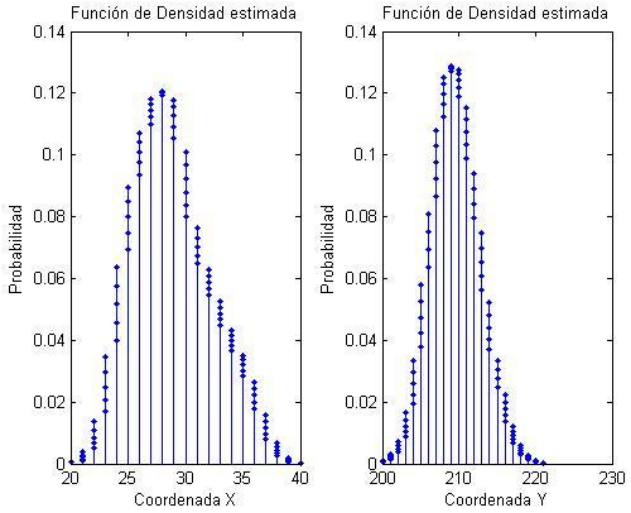


Figura 4.30: Tracking en el Punto de Prueba No. 9

De la posición central del centroide del objeto que está realizando el seguimiento se saco la moda (valor que hace máxima la función de probabilidad) en cada una de las 5 pruebas de cada punto de prueba como se observa en la tabla 4.9, Para tener una aproximación y una mayor confiabilidad de los datos se definieron intervalos de confianza.

	Coord. X encontrada	Coord. Y encontrada	% Permanencia Coord. X (Moda)	% Permanencia Coord. Y (Moda)
Punto Prueba 1	16	14	50,66 %	44,66 %
Punto Prueba 2	27	209	26,66 %	46,00 %
Punto Prueba 3	95	48	27,33 %	57,33 %
Punto Prueba 4	103	174	28,66 %	52,00 %
Punto Prueba 5	152	129	30,66 %	92,00 %
Punto Prueba 6	225	51	34,00 %	72,66 %
Punto Prueba 7	210	140	30,00 %	94,66 %
Punto Prueba 8	365	16	24,66 %	30,66 %
Punto Prueba 9	358	205	30,66 %	64,66 %

Tabla 4.9: Permanencia de Coordenadas con Moda para FPGA

El intervalo de confianza se tomo con un nivel de confianza del 95 %, para poder medir la identificación de la posición del centroide, los datos obtenidos se muestran en la tabla 4.10:

	Coord. X Promedio encontrada	Coord. Y Promedio encontrada	% Permanencia Coord. X (Confianza)	% Permanencia Coord. Y (Confianza)
Punto Prueba 1	16	14	63,33 %	70,00 %
Punto Prueba 2	27	209	43,33 %	66,67 %
Punto Prueba 3	95	48	43,33 %	66,67 %
Punto Prueba 4	103	174	40,00 %	53,33 %
Punto Prueba 5	152	149	43,33 %	93,33 %
Punto Prueba 6	225	51	40,00 %	73,33 %
Punto Prueba 7	210	140	43,33 %	96,67 %
Punto Prueba 8	365	16	36,67 %	46,67 %
Punto Prueba 9	358	205	40,00 %	63,33 %

Tabla 4.10: Permanencia de Coordenadas con Intervalo de Confianza para FPGA

Una medida de dispersión útil para comparar dispersiones a escalas distintas es el coeficiente de variación promedio, en cada punto de prueba para cada coordenada se saco el coeficiente de variación promedio:

	Coeficiente de Variación Promedio Coordenada CX	Coeficiente de Variación Promedio Coordenada CY
Punto Prueba 1	15,7537	14,2459
Punto Prueba 2	10,8796	1,2663
Punto Prueba 3	3,3823	3,1974
Punto Prueba 4	4,3331	0,5072
Punto Prueba 5	2,3412	0,1941
Punto Prueba 6	1,6275	0,8794
Punto Prueba 7	1,7683	1,6069
Punto Prueba 8	1,5967	10,4974
Punto Prueba 9	2,3421	0,2445

Tabla 4.11: Coeficiente de Variacion para FPGA

4.3. COMPARACIÓN DE RESULTADOS DE PRUEBAS ESTÁTICAS

Realizando una comparación de los dos sistemas tenemos que el sistema en FPGA es más rápido que en PC como se observa en la tablas 4.4 y 4.8. Para comparar la precisión entre los sistemas FPGA y PC se recurrió a evaluar la diferencia entre los valores de las coordenadas y la media para cada par de coordenadas entregadas por cada sistema lo que se conoce como el error promedio absoluto. Los valores obtenidos fueron los siguientes:

Para los valores de Coordenadas X:

	Error Promedio Absoluto X	
	Sistema con PC	**Sistema con FPGA**
Punto Prueba 1	0,8333	1,6933
Punto Prueba 2	7,3333	2,44
Punto Prueba 3	0,0333	2,86
Punto Prueba 4	0,0066	3,86
Punto Prueba 5	0,14	3,3066
Punto Prueba 6	0,1250	3,0866
Punto Prueba 7	1,6466	3,3330
Punto Prueba 8	0	4,7533
Punto Prueba 9	0	7,4466

Tabla 4.12: Error Promedio Absoluto Coordenadas X

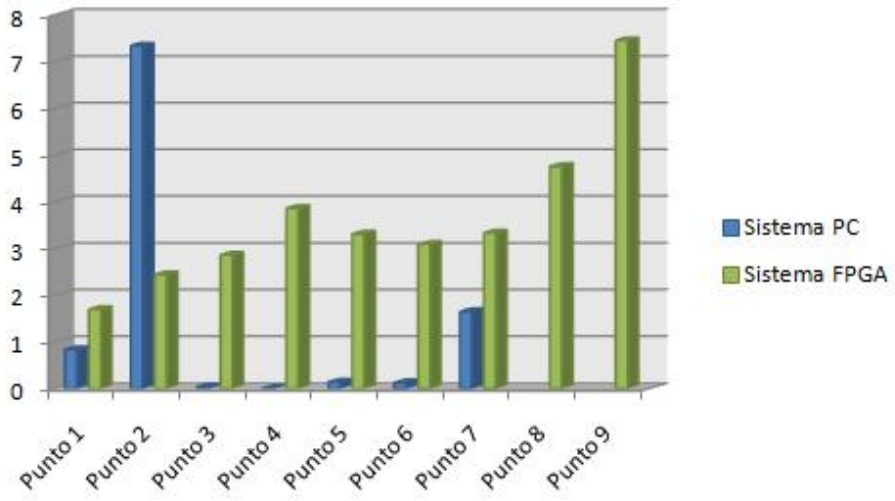


Figura 4.31: Comparación del Error Absoluto entre Sistemas Coordenadas X

Para los valores de las Coordenadas Y:

	Error Promedio Absoluto Y	
	Sistema con PC	**Sistema con FPGA**
Punto Prueba 1	0,3133	1,466
Punto Prueba 2	1,6133	1,9333
Punto Prueba 3	0,1133	0,8466
Punto Prueba 4	0,3266	0,6266
Punto Prueba 5	0	0,08
Punto Prueba 6	0,3333	0,2866
Punto Prueba 7	2,7	0,5133
Punto Prueba 8		1,7533
Punto Prueba 9		0,3533

Tabla 4.13: Error Promedio Absoluto Coordenadas Y

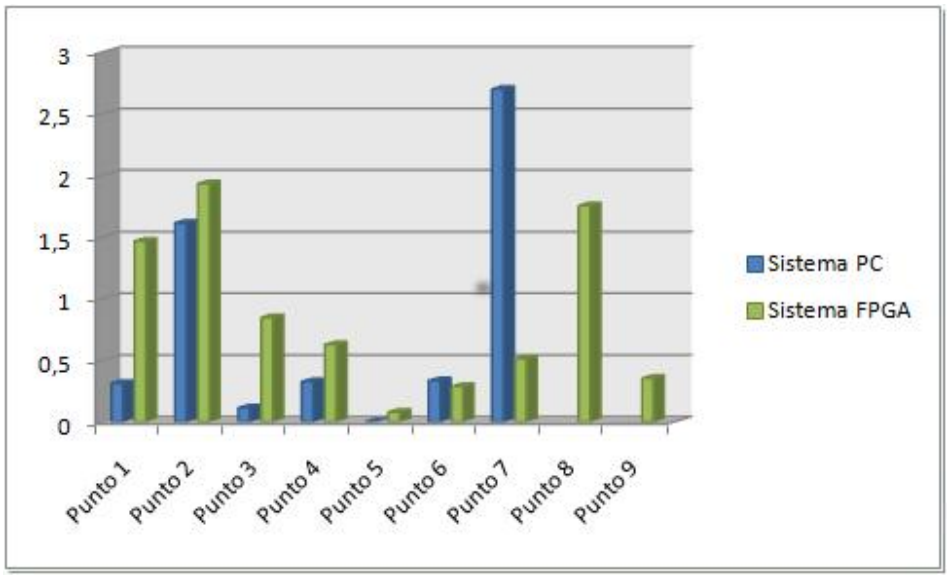


Figura 4.32: Comparación del Error Absoluto entre Sistemas Coordenadas Y

Con el error promedio absoluto se determinó el error porcentual para tener una idea más general de la precisión para cada sistema el error porcentual está definido como el cociente entre el error absoluto y el valor medio multiplicado por 100. Los datos obtenidos se muestran en las tablas 4.14 y 4.15 para cada coordenada X y Y.

	Error Porcentual Coordenadas X	
	Sistema con PC	**Sistema con FPGA**
Punto Prueba 1	5,12 %	10,33 %
Punto Prueba 2	29,33 %	8,87 %
Punto Prueba 3	0,040 %	3,00 %
Punto Prueba 4	0,010 %	3,75 %
Punto Prueba 5	0,090 %	2,18 %
Punto Prueba 6	0,050 %	1,37 %
Punto Prueba 7	0,790 %	1,58 %
Punto Prueba 8		1,30 %
Punto Prueba 9		

Tabla 4.14: Error Porcentual para Coordenadas X

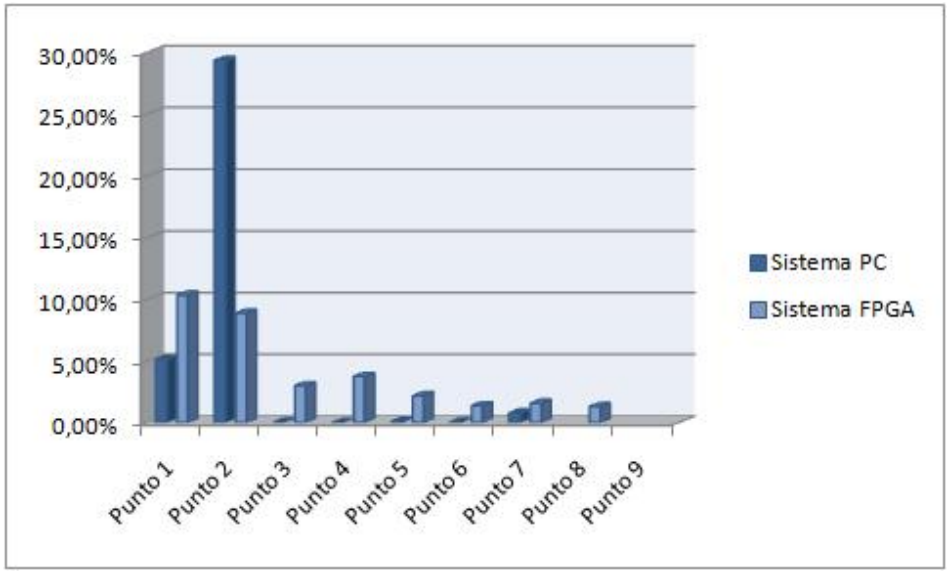


Figura 4.33: Representación Error Porcentual Coordenadas X

Para las coordenadas X el sistema con PC presenta errores en los bordes inferiores del background, y no realiza el seguimiento del objeto en el borde del extremo izquierdo del background. En los demás puntos de prueba el error del sistema con FPGA es mayor al del PC.

Para los valores de las Coordenadas Y:

	Error Porcentual Coordenadas Y	
	Sistema con PC	**Sistema con FPGA**
Punto Prueba 1	2,02 %	10,33 %
Punto Prueba 2	0,77 %	0,94 %
Punto Prueba 3	0,24 %	1,76 %
Punto Prueba 4	0,19 %	0,36 %
Punto Prueba 5	0,00 %	0,06 %
Punto Prueba 6	0,65 %	0,56 %
Punto Prueba 7	1,80 %	0,37 %
Punto Prueba 8		10,89 %
Punto Prueba 9		0,17 %

Tabla 4.15: Error Porcentual para Coordenadas Y

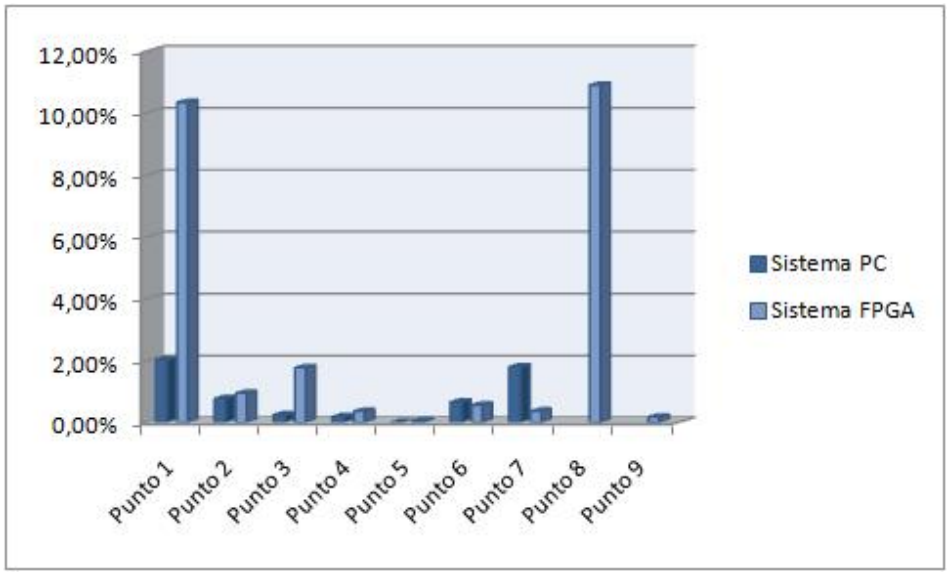


Figura 4.34: Representación Error Porcentual Coordenada Y

De la figura 4.34 podemos concluir que para el sistema con FPGA el máximo error para la posición de los puntos en Y está en los bordes superiores del background, el sistema con FPGA realiza el seguimiento en toda el área del background a diferencia del sistema con PC, y ambos sistemas tienen la tendencia a presentar menor error en el centro del background y más especificamente en los puntos de prueba 4, 5, y 6. El error para las coordenadas en el sistema con PC es muy similar y en el sistema con FPGA el mayor error está en las coordenadas X y el error en las coordenadas Y es menor.

Realizando un análisis con el error promedio relativo y con un intervalo de confianza del 95 % obtenemos las gráficas para cada coordenada donde las líneas rojas representan al

sistema con FPGA y las azules al sistema con PC:

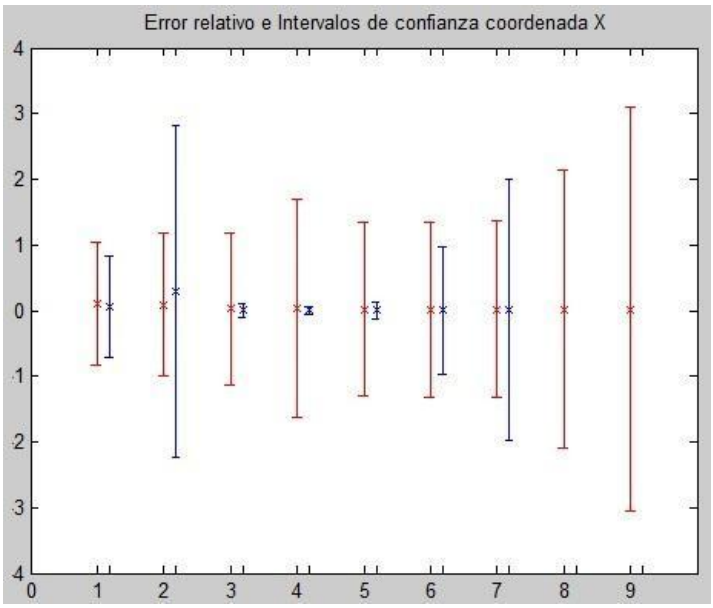


Figura 4.35: Representación Error Relativo e Intervalos de Confianza Coordenada X

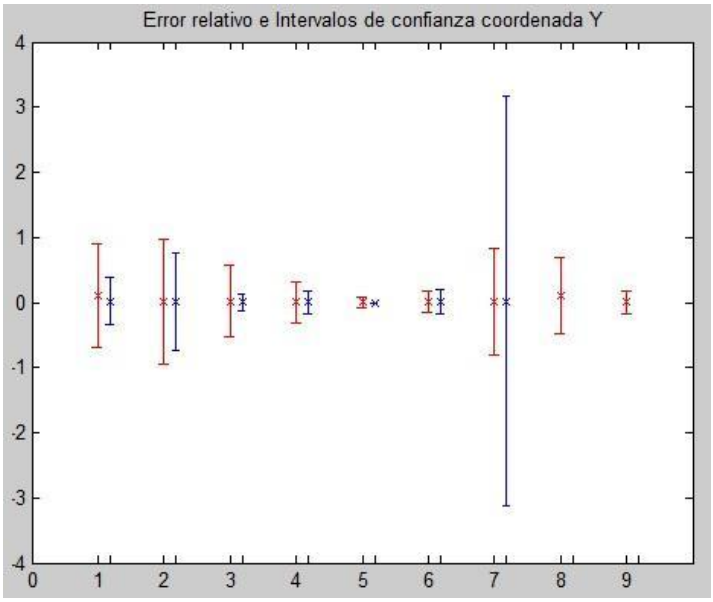


Figura 4.36: Representación Error Relativo e Intervalos de Confianza Coordenada Y

De la figura 4.35 y 4.36 ya que se cruzan los intervalos estadisticamente el error relativo es igual y dependiente del sistema y del punto de prueba.

4.4. PRUEBAS DINÁMICAS DE SEGUIMIENTO

4.4.1. PRUEBAS DINÁMICAS CON EL PC Con estas pruebas se determina la validez del seguimiento (evento de realizar el seguimiento del objeto) a diferentes velocidades de respuesta, para lo que se realizaron 3 pruebas como se observa en la figura 4.37:

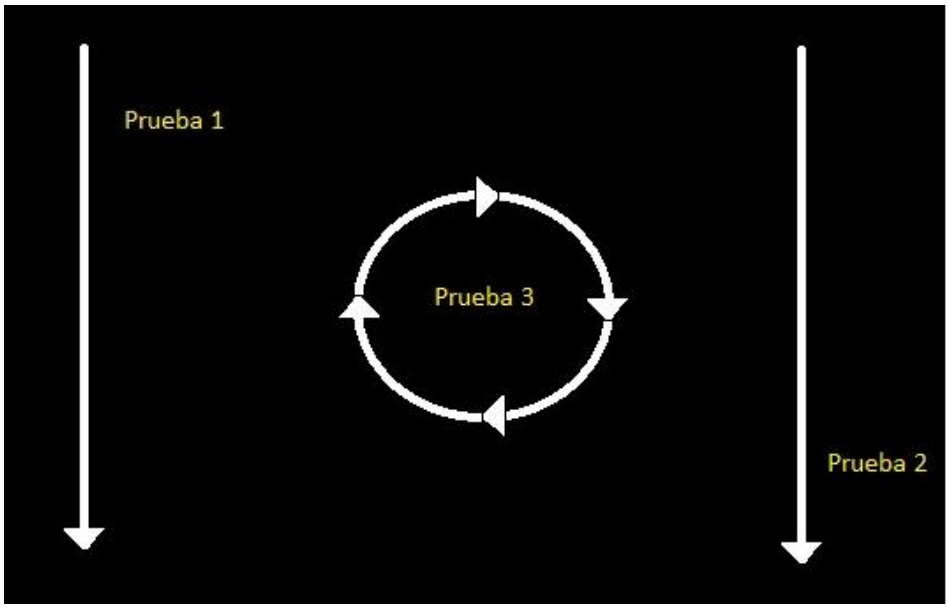


Figura 4.37: Puntos de Prueba con PC.

En las pruebas estáticas se determinó que el tiempo de procesamiento es variante con el tiempo y las pruebas dinámicas están sujetas a la misma naturaleza de variación del tiempo de procesamiento por lo que se vuelve necesario tener una base de tiempo promedio fija para poder estandarizar estas pruebas.
De las pruebas estáticas se determinó que en PC el tiempo promedio del sistema para procesar un frame es de 185,5 ms esto quiere decir que se procesan aproximadamente 5,4 frames por segundo. Se define la velocidad como la magnitud física que muestra y expresa la variación en cuanto a posición de un objeto y en función del tiempo, es decir la distancia recorrida por un objeto en la unidad de tiempo. La trayectoria del recorrido del objeto está definida por las coordenadas del centroide retornadas del procesamiento del video.
De las pruebas realizadas se obtuvo:
Para el Punto de Prueba No. 1 se realizó el seguimiento en diferentes tiempos, la distancia recorrida en unidades (posición de los puntos entregados por el sistema) es de 220 en sentido vertical hacia abajo, donde podríamos establecer en forma teórica que se da un óptimo seguimiento del objeto si el cambio de coordenadas corresponde a una tasa de cambio menor a 5 coordenadas por muestra de lo que tendríamos 44 frames procesadas de la trayectoria en 8,162 s y para cumplir con las anteriores condiciones la

velocidad umbral es de 26,95 unidades/t.
De forma práctica se realizó un seguimiento a una velocidad de 31,21 [unidades/s] en el que se procesaron 38 frames de video. Con esto la tasa de cambio es de 5,78 coordenadas por muestra, y el seguimiento se puede observar en la figura 4.38:

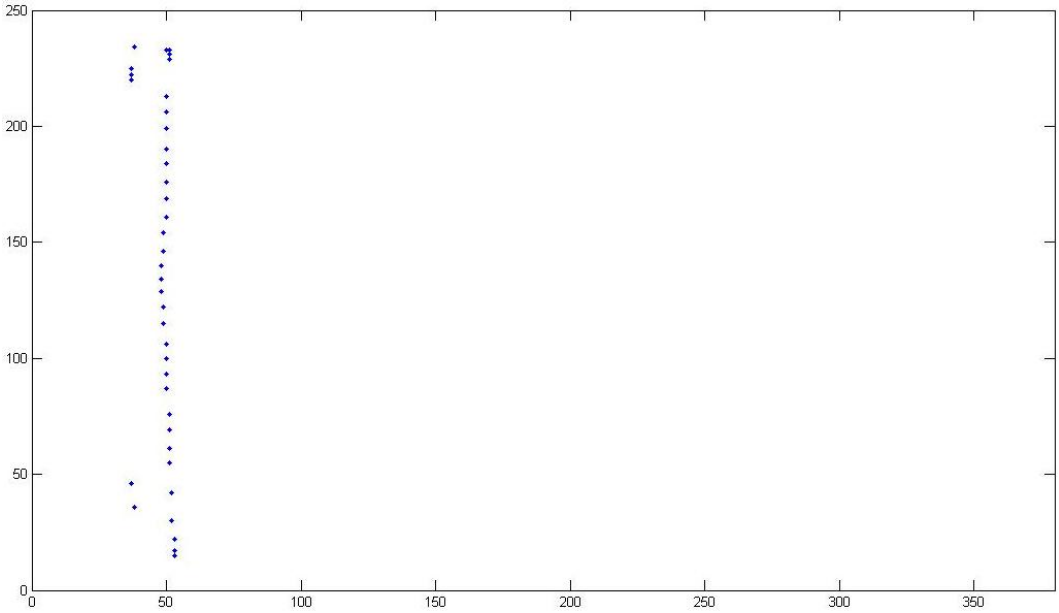


Figura 4.38: Prueba Rectilínea a 31,21 unidades/s (7,049s)

Para velocidades por debajo de este valor el seguimiento se considera correcto. Para velocidades por encima el seguimiento se hace muy espaciado aumentando la incertidumbre del recorrido del objeto es decir no se tendrá una idea clara de la trayectoria del objeto, en la figura 4.39 se muestra el objeto moviéndose a una velocidad de 148,24 unidades/t en las que se procesaron 8 frames.

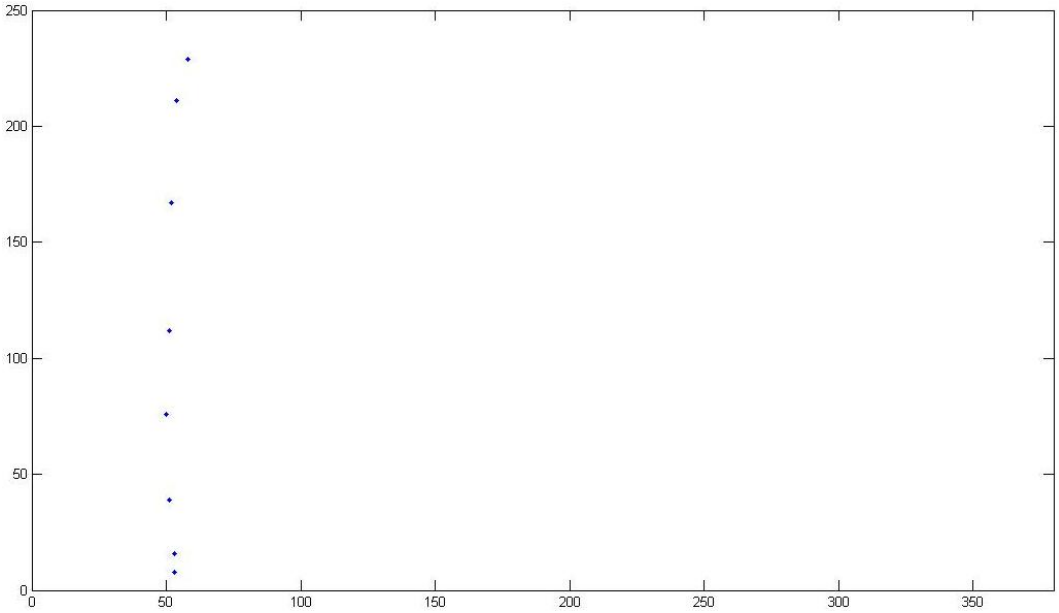


Figura 4.39: Prueba Rectilínea a 148,24 unidades/t (1,484s)

Para el Punto de Prueba No. 2 se obtuvo un comportamiento similar que para el punto 1, con la diferencia que las coordenadas presentaban mas variación en las coordenadas X durante su trayectoria. La figura 4.40 muestra el seguimiento a 11,627 unidades/t en las que se alcanzaron a procesar 102 frames.

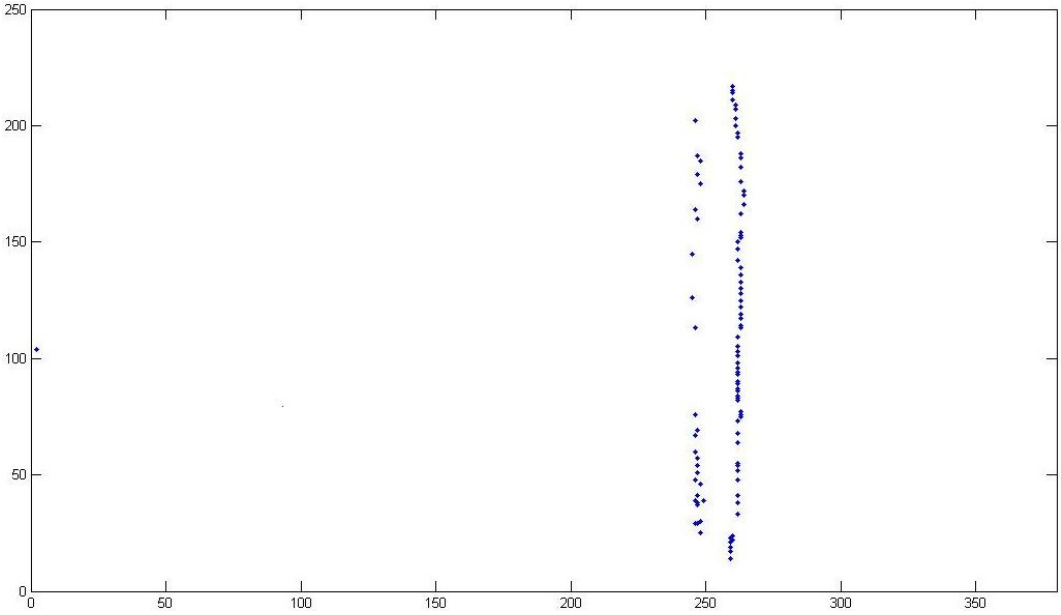


Figura 4.40: Prueba Rectilínea a 11,627 unidades/t (18,921 s)

En las anteriores pruebas el tiempo era variable y con trayectoria rectilínea. En el

Punto de Prueba No. 3 se realizó una trayectoria circular y se fijo el tiempo de prueba a aproximadamente 11 segundos para evaluar el seguimiento a diferentes velocidades con movimientos circulares. Se estableció hacer el seguimiento en un círculo de diámetro de 90 unidades, esto quiere decir que la trayectoria total para hacer un recorrido es aproximadamente de 283 unidades si se desea un cambio por coordenada de solo 5 unidades quiere decir que se deben tener al menos (283/5) frames procesadas lo que es aproximadamente 56 frames. La base de tiempo se definió de las pruebas estáticas como 185,5ms por lo que el tiempo mínimo total para realizar el trayecto circular en estas condiciones es de 10,38 s, aproximadamente una velocidad de 27,24 unidades/t.

Realizando la prueba a una velocidad de 27,73 unidades/s se realiza el trayecto de la circunferencia en 10,2 s, en la figura 4.41 se puede apreciar el resultado:

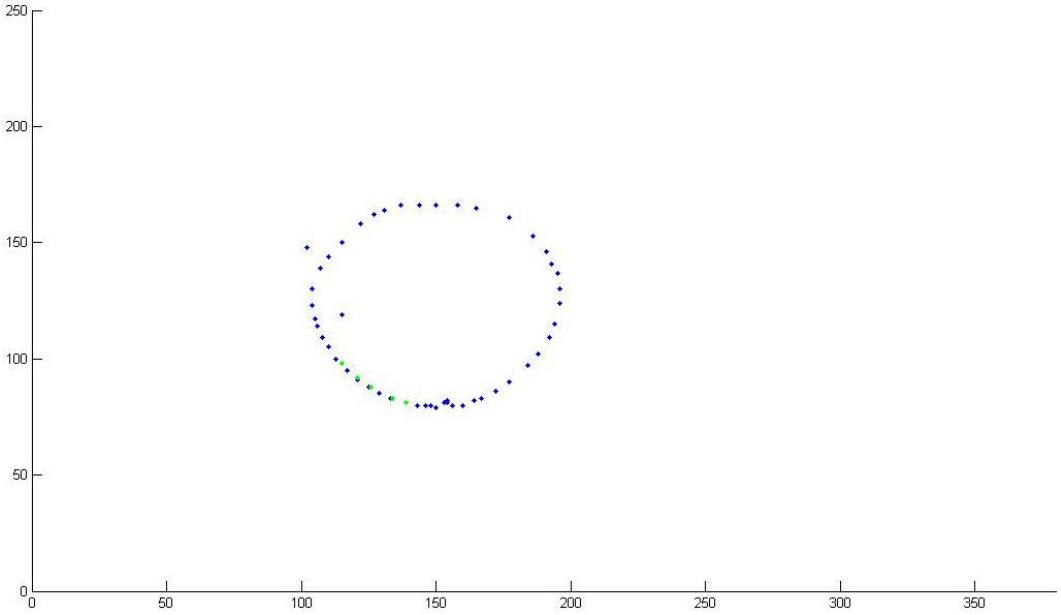


Figura 4.41: Prueba Curvilínea a 27,73 unidades/t (11 s)

La trayectoria azul corresponde a los 10,2 segundos en realizar el primer trayecto. La trayectoria verde corresponde al resto del trayecto para completar los 11 segundos. Se hace de esta forma para poder evaluar por trayectos independientes. Aumentando la velocidad a 37,23 unidades/t se obtuvo la siguiente trayectoria:

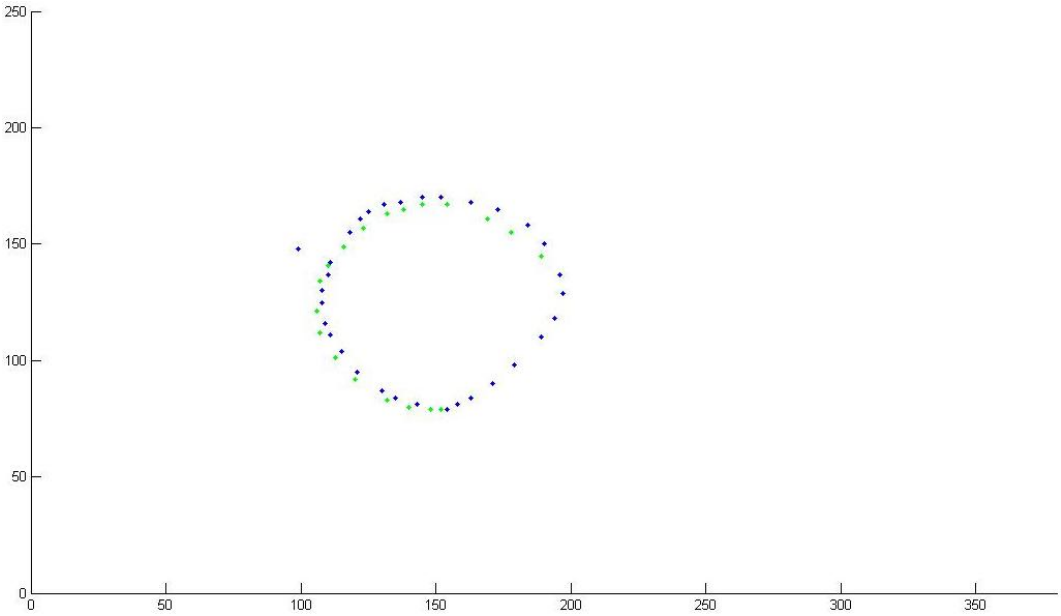


Figura 4.42: Prueba Curvilínea a 37,23 unidades/t (11 s)

Si se aumenta considerablemente la velocidad se puede ver que el seguimiento se realiza pero no se tiene definida la forma en que se realizó la trayectoria.

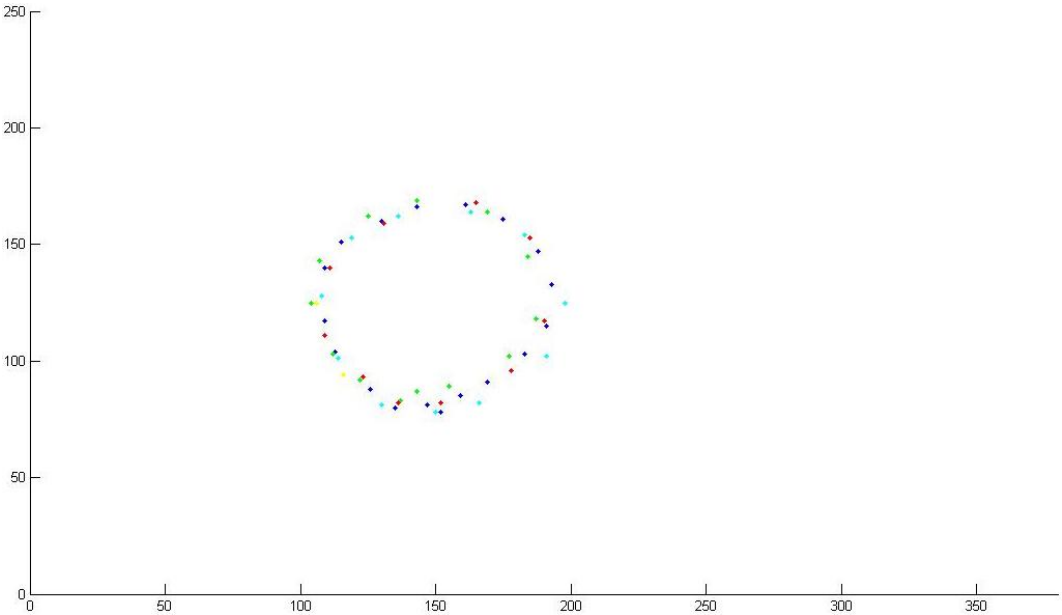


Figura 4.43: Prueba Curvilínea a 106,5 unidades/t (11 s)

En la anterior prueba se alcanzan a realizar 4 trayectorias.

4.4.2. PRUEBAS DINÁMICAS CON EL DISPOSITIVO DE LÓGICA PROGRAMABLE

Se realizaron las pruebas dinámicas figura 4.37 en el dispositivo lógico programable de forma similar que en PC. Se determinaron tres puntos de pruebas donde se realizaron varias pruebas a diferentes velocidades. De las pruebas estáticas se determinó que en la FPGA el tiempo promedio para realizar el proceso de adquisición, procesamiento y lectura de un frame es de 39,85 ms lo que quiere decir que se procesan aproximadamente 25 frames por segundo.

De las pruebas realizadas se obtuvo: Podríamos establecer en forma teórica que se da un óptimo seguimiento al objeto si el cambio de coordenadas corresponde a una tasa de cambio menor a 5 coordenadas por muestra (220/5) de lo que tendríamos 44 frames procesadas de la trayectoria en 1,7534 s y para cumplir con las anteriores condiciones la velocidad umbral es de 125,47 unidades/t. De forma práctica se realizó un seguimiento a una velocidad de 14,045 [unidades/s] en el que se procesaron 394 frames de video, en la figura 4.44 se puede observar la trayectoria del seguimiento :

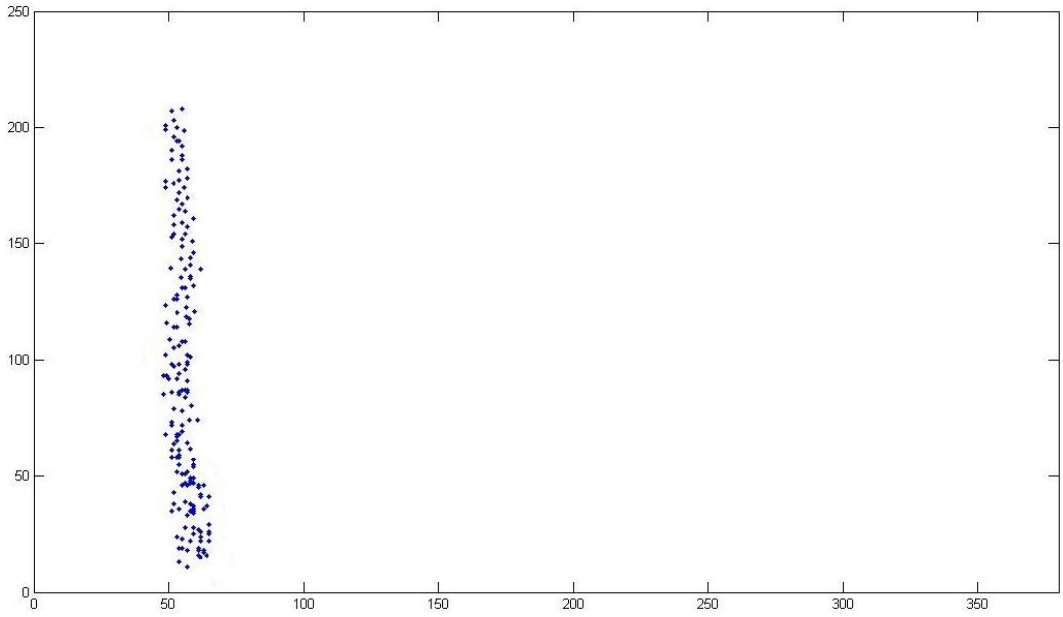


Figura 4.44: Prueba Rectilínea a 14,045 unidades/t (15,661 s)

Comparando esta trayectoria con la de PC de la figura 4.40 en la que se tiene un tiempo de 18,921 s se puede observar que se presenta más densidad de puntos en la FPGA debido a que el sistema en FPGA es más rápido que en PC lo que se determinó en las pruebas estáticas. Los datos en el sistema con FPGA presentan repetitividad de coordenadas. De la anterior figura podemos concluir que el sistema con FPGA no entrega el punto del centroide sino una tendencia del centroide.

Aumentando la velocidad a 38,07 unidades/s obtenemos el siguiente trayecto:

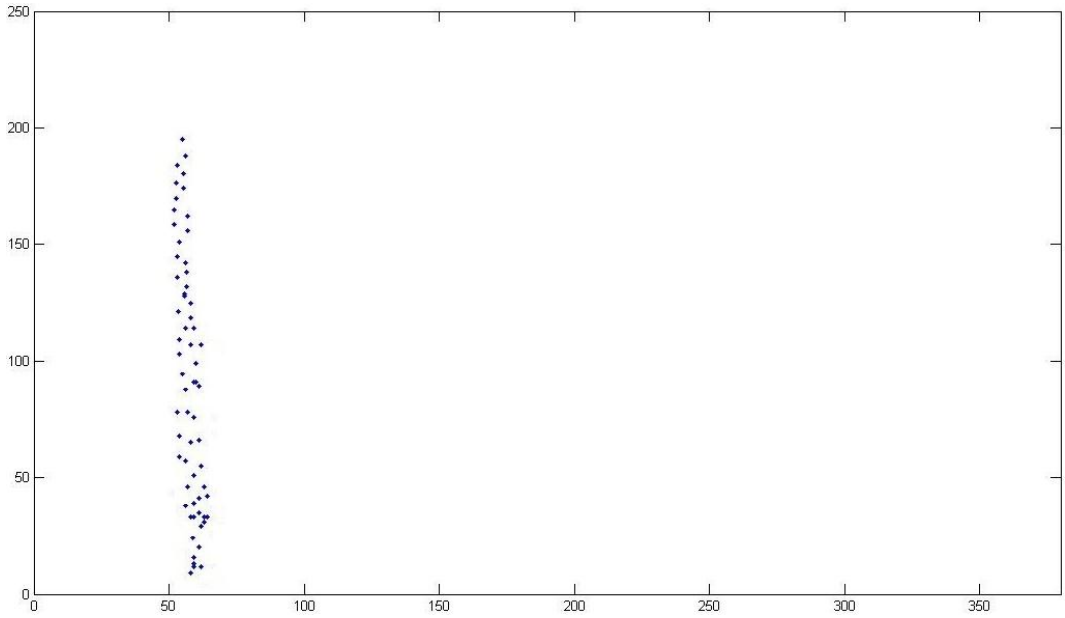


Figura 4.45: Prueba Rectilínea a 38,07 unidades/t (5,778 s)

Y a una velocidad de 110,44 unidades/s tenemos:

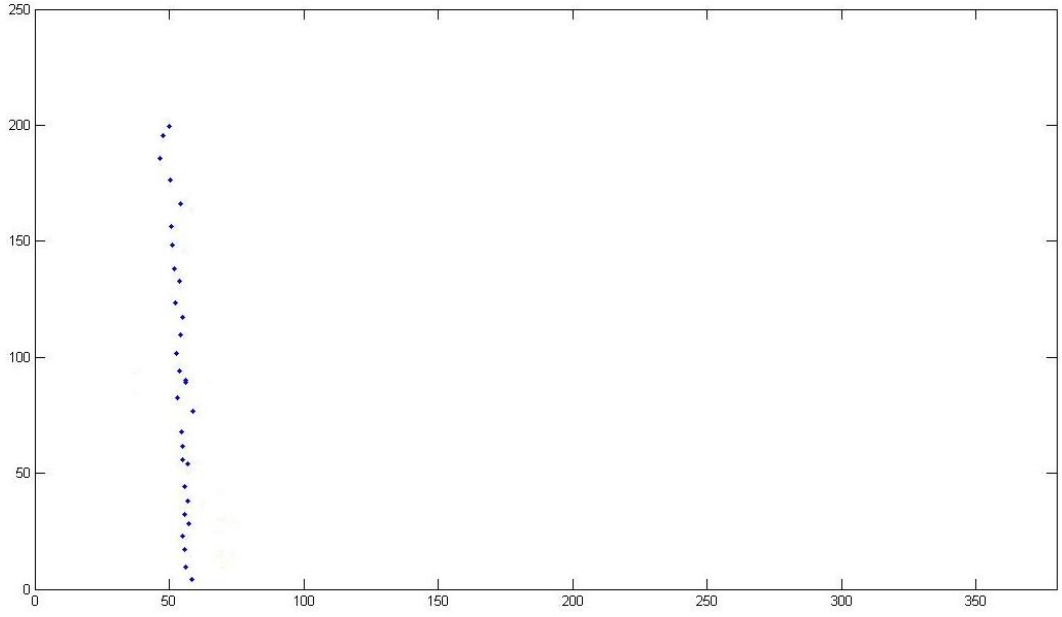


Figura 4.46: Prueba Rectilínea a 110,44 unidades/t (1,992 s)

De la figura 4.46 en la trayectoria se sigue realizando el seguimiento.

Para el Punto de Prueba No. 2 se presentó un comportamiento similar al del Punto

de Prueba No. 1 para una velocidad de 120,02 unidades/s se obtiene el siguiente seguimiento:

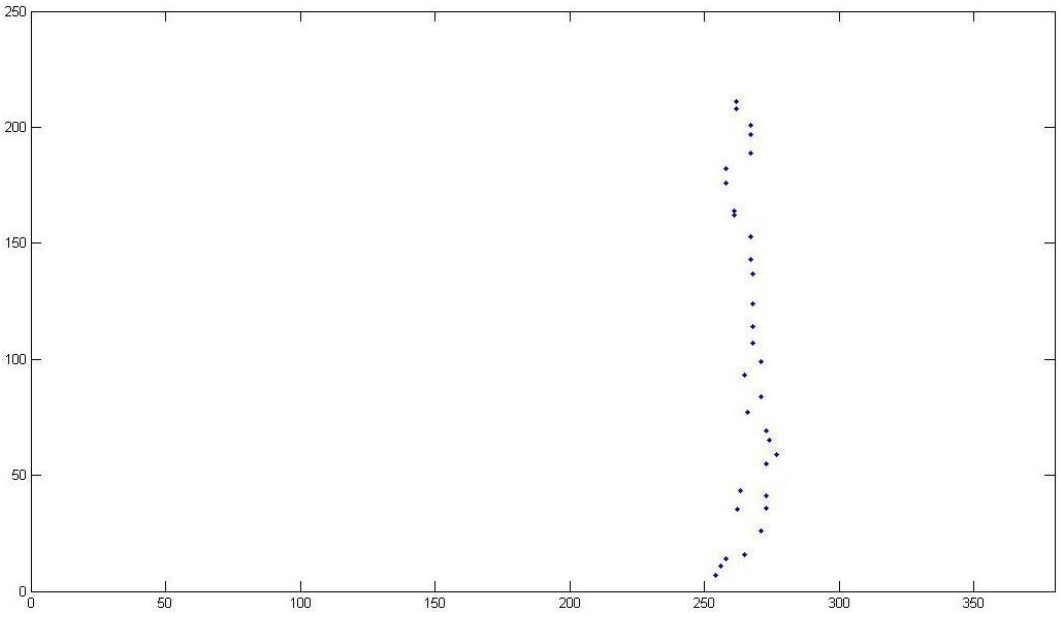


Figura 4.47: Prueba Rectilínea a 120,02 unidades/t (1,833 s)

Aumentando la velocidad 614,52 unidades/s, velocidad sobre el umbral definido se obtiene una trayectoria como se observa en la figura 4.48:

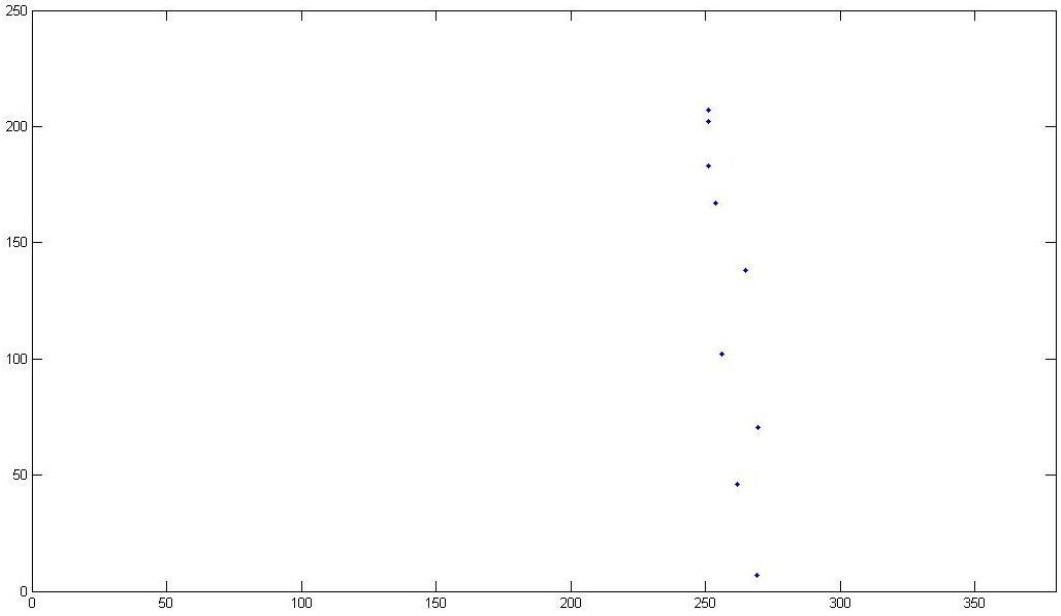


Figura 4.48: Prueba Rectilínea a 614,52 unidades/t (0,358 s)

No se observa en forma clara la trayectoria del objeto.

Para el Punto de Prueba No. 3 se realizó una prueba similar a la de PC fijando el tiempo de prueba a 11 segundos para evaluar el seguimiento a diferentes velocidades con movimientos curvilíneos. Se estableció hacer el seguimiento en un círculo con radio de 45 unidades. Esto quiere decir que la trayectoria total para hacer un recorrido es aproximadamente de 283 unidades. Si se desea un cambio por coordenada de solo 5 unidades se deben tener al menos (283/5) frames procesadas lo que es aproximadamente 56 frames. La base de tiempo se definió de las pruebas estáticas como 39,85 ms por lo que el tiempo mínimo total para realizar el trayecto circular en estas condiciones es de 2,23 s, aproximadamente una velocidad 126,90 unidades/t.
Realizando la prueba a una velocidad de 21,85 unidades/s se realiza el trayecto de la circunferencia en 12,95 s, en la figura 4.49 se puede apreciar el resultado de la trayectoria:

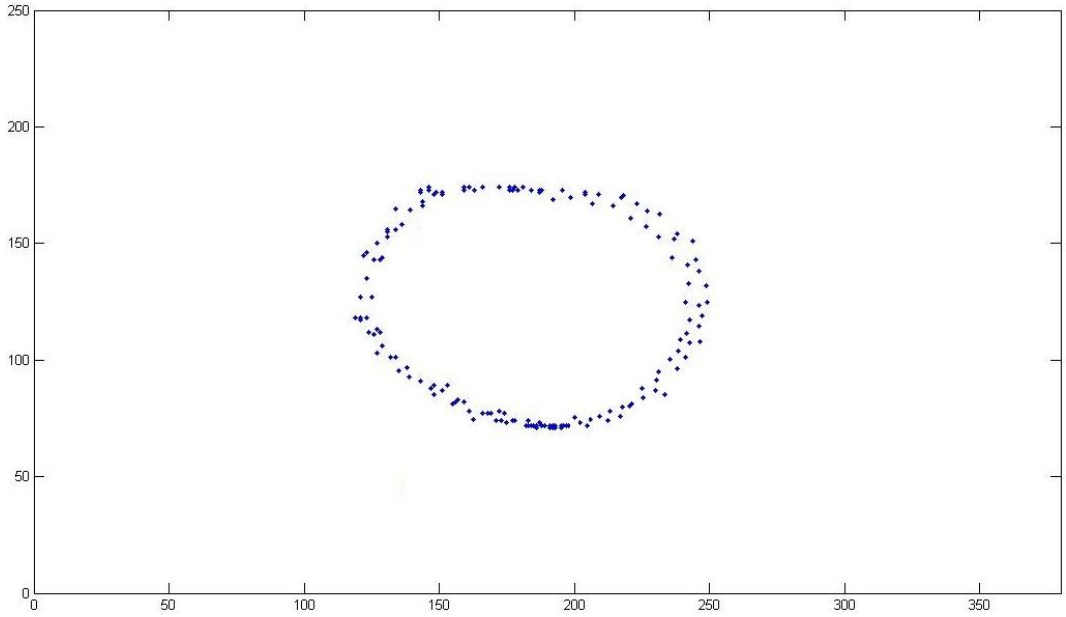


Figura 4.49: Prueba Curvilínea a 27,73 unidades/t (11 s)

Aumentando la velocidad a 35,68 unidades/t el sistema con FPGA realiza una trayectoria en 7,93 s lo que se muestra en la figura 4.50:

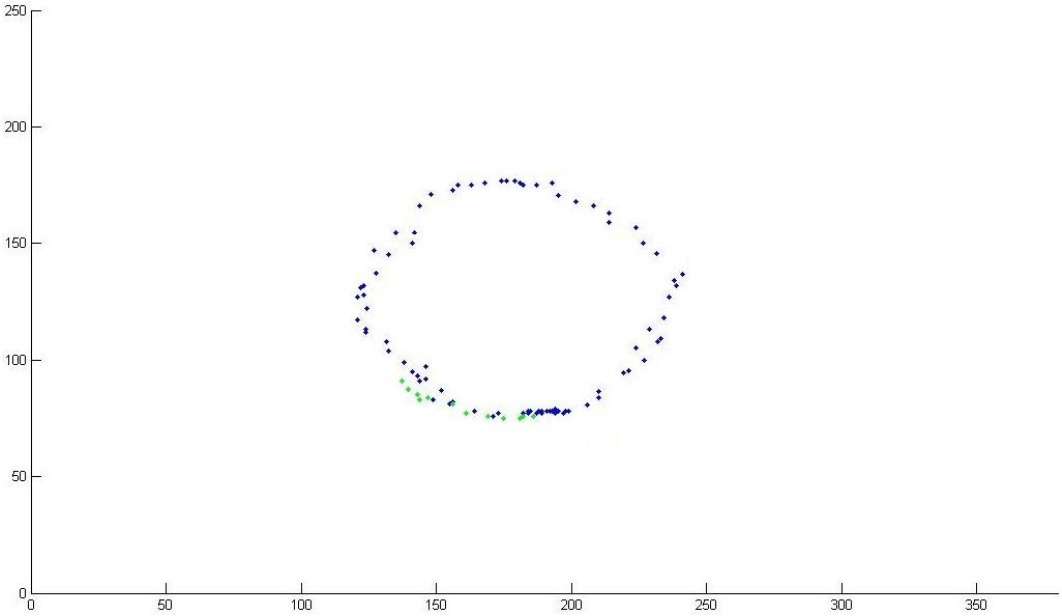


Figura 4.50: Prueba Curvilínea a 35,68 unidades/t (11 s)

5. CONCLUSIONES

- De los análisis en las pruebas de iluminación consignadas en la sección 4.1 al ejecutar el análisis de correlación y de datos realizado a los tres tipos de sistemas de iluminación podemos concluir que el más adecuado es con Lámparas, teniendo en cuenta de no usar los bordes exteriores laterales donde se observa violación de datos como se muestra en la figura 4.11.

- La rápidez de la transmisión de datos en la FPGA influye directamente en la velocidad de procesamiento del video por lo que hacer uso de una conexión más rápida con modelos de FPGA producidos en la actualidad por Xilinx Inc., como es el caso por ejemplo de Virtex II, Virtex IV, Virtex V, Virtex VI y Virtex VII contienen una mayor cantidad de recursos comparados con la Spartan 3E lo que puede contribuir considerablemente a mejorar los procesos implementados en este sistema.

- En la adquisición de video en el sistema con FPGA para realizar la sincronización de video se debe utilizar las secuencias EAV y SAV y no debe hacerse sobre el número de ciclos de reloj por línea o intervalo de borrado horizontal ya que la adquisición de video puede no realizarse.

- La velocidad de procesamiento en PC está ligado a recursos físicos como la memoria, el procesador, el sistema operativo entre otros.

- El problema de la segmentación en el sistema con FPGA se debe en parte a que solo se utiliza un frame de video para hacer el procesamiento y también a inconvenientes con el lente de la cámara en los bordes y este mismo problema del lente se presentó con el sistema en PC.

- Para el sistema en el dispositivo lógico programable los datos YCbCr adquiridos del videodecodificador no pueden utilizar los valores de 8 bits de 00H y FFH ya que los valores se utilizan para la información de la sincronización.

- En la implementación de la visualización en pantalla del sistema con el dispositivo lógico programable hay limitaciones por la Tarjeta Spartan 3E pues solo es posible obtener una salida de un bit por color (RGB) pudiendo solo visualizar 8 colores. La imagen del videodecodificador nos entrega por color 8 bits por lo que con una mejor tarjeta es posible mostrar el video de salida con una amplia gama de colores. Por este inconveniente en la memoria RAM solo se manejo un bit de 24 procesados (8 bits por color R-G-B) mostrando la imagen adquirida y el tracking en color blanco.

- La definición y fidelidad del video es debida en parte al video entregado por la cámara. Utilizamos la misma cámara para comparar los dos sistemas pero para cada uno se implementó el proceso de adquisición con un video decodificador diferente, el LR37 (REV:B) para PC y VDEC1 (ADV7183B) para FPGA de lo que se concluye que para mejorar el sistema en PC se puede realizar la implementación con un mejor video decodificador y para mejorar los tiempos de procesamiento se puede implementar la aplicación con un software más eficiente como es OpenCV librería de visión artificial originalmente desarrollada por Intel. OpenCV abarca una gran gama de áreas en el proceso de visión, como es el reconocimiento de objetos, reconocimiento facial, calibración de cámaras, visión estereoscópica y visión robótica, entre otros.

- De las tablas 4.4 y 4.8 podemos decir que la velocidad de procesamiento del video para el seguimiento de objetos en FPGA es mayor, en PC se obtuvo que el tiempo promedio de procesamiento de un frame se realiza en 185,5 ms y para el dispositivo de lógica programable se realiza en 39,8 ms por frame de lo que podemos concluir que el sistema en FPGA es 4,6608 veces mas rápido que el sistema en PC.

- Teniendo en cuenta los tiempos máximos de procesamiento en las pruebas estáticas para el sistema se tiene que: para PC con el máximo tiempo promedio el tiempo para procesar un frame es de 189,986 ms y en un segundo solo podemos procesar 5,26 frames y con el sistema con FPGA el máximo tiempo promedio el tiempo para procesar un frame es de 44,31 ms procesando aproximadamente 22,5 frames en un segundo. De igual forma con el tiempo mínimo promedio se pueden procesar 5,4644 frames por segundo para el sistema con PC y 26,73 frames por segundo para el sistema con FPGA de lo que podemos concluir que en PC se procesan de 5,26 a 5,4644 frames por segundo y en FPGA de 22,5 a 26,73 frames por segundo.

- En el sistema con FPGA para mejorar el tiempo de procesamiento total en la lectura del video se tomó un solo frame de los dos sacrificando definición del video pero realizando el proceso de lectura en 1716 ciclos por línea con 252 líneas para un total de 432432 ciclos de reloj. Como la lectura se realiza a 27 Mhz la imagen es leida en 16 ms. Como se está adquiriendo un frame el tiempo del segundo frame se usa para realizar procesamiento y parte del proceso de lectura.

- De las tablas 4.6 y 4.10 el seguimiento de los objetos con FPGA se realizó en toda el area del background mientras que con el PC se tuvieron inconvenientes en los bordes debido a problemas con la cámara que afectaban la segmentación, lo que se puede observar en las figuras 4.21 y 4.23. Además la implementación en PC tiene menos definición de video como podemos observar las anteriores figuras en las que el video se observa pixelado comparado con el sistema en FPGA como se observa en la figura 4.29.

- Para el sistema en FPGA se realizaron las pruebas teniendo como base de tiempo el reloj de la FPGA de 50 MHz y el tiempo de procesamiento del video de 100MHz pero con el DCM implementado es posible graduar frecuencias superiores aumentando la velocidad del sistema de la FPGA al fijar la base de tiempo del sistema como el reloj del DCM. La única consideración es que el bloque de lectura siempre debe estar a 50 MHz para video con VGA.

- En las Pruebas Estáticas para determinar la precisión de los sistemas es decir la identificación del centroide obteniendo el mismo resultado en mediciones diferentes realizadas en las mismas condiciones a cada sistema se llevaron a cabo varios análisis estadísticos, entre los cuales se determinó la tendencia de la coordenada en cada punto de prueba con la moda (Tabla 4.5 y 4.9). Para tener una idea general de la permanencia del centroide se sacó un intervalo de confianza de 95 % para estimar el número de repetitividad de la coordenada (Tabla 4.6 y 4.10) en donde se observa que el sistema en PC tiene menos variabilidad y para definirla se determinaron los coeficientes de variación promedio (Tabla 4.7 y 4.11) lo cual indica que para PC las coordenadas 'X' y 'Y' tienden a tener poca variabilidad y ambas varían de forma similar a diferencia del sistema en FPGA en el que la coordenada X presenta más variabilidad que la coordenada Y. De los sistemas se puede definir el error de cada implementación para lo cual se realizaron gráficas de error porcentual comparando para cada coordenada en ambos sistemas. De la figura 4.33 podemos concluir que para las coordenadas X el sistema con PC presenta mayor error en los bordes inferiores del background, y no realiza el seguimiento del objeto en el borde del extremo izquierdo. En los demás puntos de prueba el error del sistema con FPGA es mayor al del PC. De la figura 4.34 podemos concluir de las coordenadas Y que para el sistema con FPGA el máximo error para la posición de los puntos está en los bordes superiores del background. El sistema con FPGA realiza el seguimiento en toda el área del background a diferencia del sistema con PC, y ambos sistemas tienen la tendencia a presentar menor error en el centro del background y más específicamente en los puntos de prueba 4, 5 y 6. En general el error para las coordenadas en el sistema con PC es pequeño y similar para el sistema con FPGA y el mayor error esta en las coordenadas X y las coordenadas Y son más estables.

- En las pruebas dinámicas se determina la validez del seguimiento (evento de realizar el seguimiento del objeto) a diferentes velocidades de respuesta. En las pruebas se fijó que un seguimiento es óptimo si se encuentra que sus coordenadas tienen una tasa de cambio igual o menor a 5 unidades siendo estas la representación espacial de los píxeles de la adquisición de la imagen. De la prueba 1 y 2 donde la distancia es de 220 unidades y la velocidad promedio de procesamiento es de 185,5 ms se determinó que la velocidad umbral es 26,95 unidades/s. Se realizaron pruebas de los puntos donde se confirmó que velocidades por debajo de este umbral realizan un seguimiento con las condiciones establecidas como se observa en la figura 4.40. Para velocidades muy por encima del umbral

como se esperaba no se tiene una idea clara de la trayectoria (figura 4.39) y de igual forma que para el sistema en FPGA con las mismas condiciones la velocidad umbral es de 125,47 unidades/t. Para velocidades por debajo se realiza un buen seguimiento (figura 4.46) y por encima del umbral a 614,52 unidades/s se realiza seguimiento en 9 puntos del background pero no sucede igual que el sistema en PC donde no se tiene claridad de la trayectoria. Con las velocidades umbral podemos confirmar que el sistema en FPGA es 4,65 veces más rápido que el sistema en PC. De todas las pruebas se confirmó que el error encontrado en las pruebas estáticas es que el sistema en PC nos retorna la posición del centroide del objeto y el sistema en FPGA nos retorna una tendencia lo que se puede observar en las figuras del Punto de Prueba No. 3, figura 4.41 para PC y figura 4.49 para FPGA.

- De la figura 4.35 y 4.36 ya que se cruzan los intervalos estadisticamente el error relativo es igual y dependiente del sistema y del punto de prueba de lo que se puede concluir que ambos sistemas son igualmente efectivos.

- Para este trabajo implementamos algoritmos con estructuras sencillas, pues no simpre el utilizar algoritmos robustos nos proporciona la mejor solución buscando opciones que mejoren el costo computacional y tiempo de procesamiento.

6. DISCUSIÓN Y RECOMENDACIONES

Con este trabajo hemos desarrollado un bloque VHDL capaz de muestrar una señal de vídeo digital en un formato YCbCr y almacenar los valores resultantes en una memoria RAM para su procesamiento. Este bloque después lo integramos al programa principal. Después de integrar y realizar el procesamiento de los datos se envia información del video en este caso de las coordenadas de orientación de un objeto al que se le realizó seguimiento para su visualización en pantalla con tracking a través de un puerto VGA. El almacenamiento de las imágenes puede ser usado por ejemplo para probar algoritmos de reconocimiento de objetos basado en SMV (Support Vector Machine), que puede ser fácilmente adaptable a aplicaciones específicas y plataforma de hardware de interés.

En particular, creamos un sistema integrado en una tarjeta de desarrollo basada en FPGA que ejecuta en tiempo real el 'reconocimiento de objetos' en el video adquirido por una cámara que puede ser usado por ejemplo, para contar el número de personas que entran o salen de un laboratorio, el reconocimiento de objetos, la navegación de sistemas autónomos en robótica móvil, el seguimiento en sistemas de vigilancia y la estimación de movimiento en robótica de manipuladores. Para trabajos futuros se plantea estudiar la escogencia de la cámara a utilizar. Las cámaras pueden estar basadas en dos tipos de tecnología, CCD (Charged Couple Device) ó CMOS (Complementary Metal Oxide Semiconductor). Los sensores CCD tienen mayor sensibilidad a la luz, más calidad y también precio más alto, en tanto que los de tipos CMOS son menos sensibles y de menor calidad, pero al ser fáciles de fabricar son más baratos. Para aplicaciones que requieran alta calidad de video es recomendable usar cámaras con sensores CCD.

BIBLIOGRAFÍA

[BTO05] Leif Olsson Benny Thörnberg and Mattias O'Nils, *Optimization of memory allocation for real-time video processing on fpga*, Mid-Sweden University **87V** (2005).

[CDS09] Erdal Oruklu Christophe Desmouliers and Jafar Saniie, *Fpga-based design of a high-performance and modular video processing platform*, Information of Technology **47** (2009).

[Cle08] Irene Ayllón Clemente, *Investigación e implementación de un sistema de seguimiento de objetos basado en métodos de segmentación a través de level sets*, Honda Research Institute Europe y TU Darmstadt **8865a495** (2008).

[DMS09] Andrés Páez David Mora and Julián Quiroga Sepúlveda, *Detección de objetos móviles en una escena utilizando flujo Óptico*, XIV Simposio de Tratamiento de Señales, Imágenes y Visión Artificial – STSIVA 2009 **Doc** (2009).

[FRVRV08] Jaime Fernando Gómez Rodríguez Francisco Raúl Valdiviezo Román and Leticia Mojica Vera, *Dispositivos electrónicos de lógica programable*, Revista Cientitech **27** (2008), 7–20.

[GE92] Rafael C. Gonzalez and Richard E.Woods, *Procesamiento digital de imágenes*, Addison-Wesley Publishing Company, 1992.

[Inc01] Xilinx Inc., *Digital component video conversion 4:2:2 to 4:4:4*, Xilinx XAPP294, 2001.

[Inc05a] Analog Devices Inc, *Multiformat sdtv video decoder*, ADV7183B, 2005.

[Inc05b] Diligent Inc, *Diligent video decoder board*, VDEC1 Reference Manual, 2005.

[Inc09a] Xilinx Inc., *Block memory generator v3.3*, Xilinx DS512, 2009.

[Inc09b] Xilinx Inc, *Data sheet spartan-3e fpga family*, Xilinx DS312 v 3.8, 2009.

[JUCJ07] Xuan Dai Pham Jung Uk Cho, Seung Hun Jin and Jae Wook Jeon, *Multiple objects tracking circuit using particle filters with multiple features*, IEEE International Conference on Robotics and Automation **10T** (2007).

[KBR09] Mahalingam Venkataraman Koustav Bhattacharya and Nagarajan Ranganathan, *A vlsi system architecture for optical flow computation*, Department of Computer Science and Engineering University of South Florida **27V** (2009).

[LM06] Andrea Betancourt López and Catalina Zuluaga Muñoz, *Implementación del protocolo de enrutamiento ospf en un dispositivo fpga*, Master's thesis, Universidad Tecnológica de Pereira, 2006.

[LO05] Najeem Lawal and Mattias O'Nils, *Embedded fpga memory requirements for real-time video processing applications*, Information Technology and Media **08** (2005).

[Lop06] Joan Mauricio Lopez, *Registro de transeuntes en tiempo real utilizando un sistema de visión artificial sobre un ambiente controlado*, Master's thesis, Universidad Tecnológica de Pereira, 2006.

[MKK02] Thomas Lammert Marco Krips and Anton Kummert, *Fpga implementation of a neural network for a real-time hand tracking system*, Department of Electrical and Information Engineering University of Wuppertal **33T** (2002).

[NLO06] Benny Thornberg Najeem Lawal and Mattias O'Nils, *Address generation for fpga rams for efficient implementation of real-time video processing systems*, Revista IEEE **01** (2006).

[NYS04] Kichul Kim Nan Yu and Zoran Salcic, *A new motion estimation algorithm for mobile real-time video and its fpga implementation*, Revista IEEE **20** (2004), 383–386.

[Oli07] Alberto Oliveri, *Signal processing algorithms for limited resources digital architectures*, Master's thesis, Universidad de Génova, 2007.

[RA07] Kumara Ratnayake and Aishy Amer, *Secuential, irregular and complex object contour tracing on fpga*, Concordia University, Electrical and Computer Engineering, Montreal, Quebec, Canadá **18** (2007).

[Sor07] Rafael Molina Soriano, *Estimación del campo de movimiento en el plano de la imagen*, Master's thesis, Universidad de Granada, 2007.

[Tir07] Wilmer Alvarez Tirado, *Seguimiento de objetos para la reconstrucción de ambientes en 3d*, Maestría en Ingeniería de Sistemas y Computación, Universidad Nacional de Colombia **07** (2007).

[Wak01] J. Wakerly, *Diseño digital:principios y prácticas*, Prentice Hall, 2001.

[YJJK08] Kwang-Seok Moon Yong-Jae Jeong and Jong-Nam Kim, *Implementation of real time video watermark embedder based on haar wavelet transform using fpga*, Dept. of Electronic Computer Communication Engineering, Pukyong Natational University **12V** (2008).

[YLZH91] Dar-chang Juang Yee-Lu Zhaog and Chun-Hsein Horng, *A color video camera using fpga video peocessor*, Electronics Research Service Organization, Hsinchu, Taiwan **25V** (1991).

[YNS08] Takao Kawamura Yukinori Nangase, Takahiko Yamamoto and Kazunori Sugahara, *Hardware realization of panoramic camera with speaker-oriented face extraction for teleconferencing*, Faculty of Engineering, Tottori University **88V** (2008).

[YVV98] Gerbrands J. Young, I. and L. Van Vliet, *Fundamentals of image processing*, The Netherlands: Delf University of Technology, 1998.

[ZAC05] Wei Wang Z. Abid and Yaobin Chen, *Low-power fpga implementation for da-based video processing*, Department of Electrical and Computer Engineering Indiana University **19V** (2005).