PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO DE UN PROGRAMA DE ORTORECTIFICACIÓN Y GEOREFERENCIACIÓN DE IMÁGENES AÉREAS APLICADAS A CAMPOS DE CAÑA DE AZÚCAR

Tesis para optar el Título de Ingeniero Electrónico, que presenta el bachiller:

Gary Javier López de Paz

ASESOR: Donato Andrés Flores Espinoza

Lima, junio del 2011

Índice

Anexo1. Programa de Ortorectificación y Georeferenciación de Imágenes

El programa desarrollado se divide en dos etapas. La primera es la creación de un pequeño programa que realiza la lectura de una imagen satelital, para posteriormente representarla y guardarla en un formato de imagen BMP.
La segunda es la elaboración del programa principal de la tesis que consiste en ortorectificar y georeferenciar registros de imagen haciendo uso de una imagen satelital como referencia. En esta etapa se hace uso de la imagen generada por la primera.

1.1 Programa de Generación de Imagen Satelital en formato BMP

Comentario:

Este programa se encarga de la lectura de una imagen satelital y mostrarla en un formato de imagen BMP, para ello se establece el origen de la imagen satelital en coordenada UTM (metros), tanto para el eje y como para el eje x, que deseamos recortar. Esto debido a que la aeronave sobrevuela solo una zona especifica y como disponemos de dispositivos GPS sabemos la ubicación aproximada de la imagen que debemos analizar. Además se debe indicar los offsets de imagen, tantos pixeles a la derecha del origen establecido y pixeles debajo de este origen. Por último se debe indicar la banda del espectro que se desea visualizar, en nuestro caso estamos colocando el valor de 4 que indica banda infrarroja cercana.

Programa:

```
#include <gdal.h>
#include <cpl_conv.h>
#include <stdio.h>
#include <cv.h>
#include <highgui.h>
#include <cxcore.h>

// Definimos datos que vamos a usar en nuestro programa tanto el origen de la imagen recortada en el eje x,
// en el eje y, así como los offsets explicados anteriormente. Por último definimos la banda de frecuencia
// Se asigna de esta manera:
//  1: Banda Roja
```

```
// 2: Banda Azul
// 3: Banda Verde
// 4: Banda Infrarroja cercana


#define origenx_satelital 6128
#define origeny_satelital 7730
#define offsetx_satelital 3018
#define offsety_satelital 4063
#define factor_escala 0.25
#define banda 4



int main(int argc, char** argv)
{
    IplImage*  img0;
   GDALDatasetH  hDataset;
                const char *window1 = "Imagen";
   GDALAllRegister();
//Ingresamos el nombre de la imagen satelital para su lectura en formato TIF
   hDataset = GDALOpen( "10OCT30154925-S2AS-052421240010_01_P001.TIF", GA_ReadOnly );
   if( hDataset == NULL )
   {
      fprintf(stderr,"es nulo\n");
   }

   GDALDriverH   hDriver;
   double        adfGeoTransform[6];
   hDriver = GDALGetDatasetDriver( hDataset );
   printf( "Driver: %s/%s\n",
         GDALGetDriverShortName( hDriver ),
         GDALGetDriverLongName( hDriver ) );

//Mostramos en pantalla el tamaño en pixeles de la imagen satelital


   printf( "Size is %dx%dx%d\n",
         GDALGetRasterXSize( hDataset ),
         GDALGetRasterYSize( hDataset ),
         GDALGetRasterCount( hDataset ) );

//Mostramos algunos datos propios de la imagen satelital
//como proyecciones, origen en coordenadas UTM así como su
//resolución en pixeles/metros tanto para el eje vertical como para el
//horizontal


   if( GDALGetProjectionRef( hDataset ) != NULL )
      printf( "Projection is `%s'\n", GDALGetProjectionRef( hDataset ) );

   if( GDALGetGeoTransform( hDataset, adfGeoTransform ) == CE_None )
   {
      printf( "Origin = (%.6f,%.6f)\n",
            adfGeoTransform[0], adfGeoTransform[3] );

      printf( "Pixel Size = (%.6f,%.6f)\n",
            adfGeoTransform[1], adfGeoTransform[5] );
   }
      GDALRasterBandH hBand;
      int             nBlockXSize, nBlockYSize;
      int             bGotMin, bGotMax;
      double          adfMinMax[2];

//Declaramos la banda de frecuencia (banda) donde queremos trabajar
      hBand = GDALGetRasterBand( hDataset, banda );

     if( GDALGetRasterColorTable( hBand ) != NULL )
        printf( "Band has a color table with %d entries.\n",
              GDALGetColorEntryCount(
```

```
                GDALGetRasterColorTable( hBand ) ) );
```

//Declaramos un puntero (pafScanline) donde se almacenarán los datos de la imagen satelital recortada

```
    unsigned int *pafScanline;
    int   nXSize = GDALGetRasterBandXSize( hBand );

    pafScanline = (unsigned int *) CPLMalloc(sizeof(unsigned int)*offsetx_satelital*offsety_satelital);
    GDALRasterIO( hBand, GF_Read,origenx_satelital ,origeny_satelital, offsetx_satelital,offsety_satelital,
            pafScanline, offsetx_satelital,offsety_satelital, GDT_UInt16 ,
            0, 0 );

  CvMat mat = cvMat( offsety_satelital, offsetx_satelital, CV_16UC1,pafScanline );
  IplImage* Img8bits=cvCreateImage( cvGetSize(&mat), IPL_DEPTH_8U, 1 );
  cvConvertScale(&mat,Img8bits,factor_escala,0);
```

// Aquí guardamos la imagen satelital en formato BMP, en este caso con el nombre imaprueba1.bmp

```
  cvSaveImage("imaprueba1.bmp", Img8bits,0);
```

//Liberamos la memoria usada para almacenar estos datos

```
            CPLFree(pafScanline);
            cvReleaseImage( &Img8bits );

 return 0;

 }
```

1.2 Archivos de cabecera y variables globales en Programa Principal

Comentario:

Aquí se mencionarán las librerías y variables globales usadas en el programa Principal.

Programa:

```
#include <gdal.h>
#include <cpl_conv.h>
#include <stdio.h>
#include <cv.h>
#include <highgui.h>
#include <cxcore.h>
#include <math.h>
#include <gtk/gtk.h>
```

// Definimos los offset de corte que mencionamos en la etapa de generación de imagen satelital en formato
//BMP

```
#define origenx_satelital 6128
#define origeny_satelital 7730
```

//contadores de puntos de control tanto para imagen satelital, imagen a corregir e imagen ortorectificada

```
int contador_puntos_control_satelital=0,puntos=0,codigo=0;
int contador_puntos_control_imagen_sin_corregir=0,contador_puntos_control_imagen_resultado=0;
```

//matrices donde se almacenaran las coordenadas pixeles al realizar los clicks del mouse

```
CvPoint2D32f matriz_satelital [10]= {0}, matriz_zoom [1],matriz_corregir[10]={0},matriz_resultado[10];

//Tipos de datos propios de Opencv donde  almacenaremos tanto la imagen satelital, el registro de imagen
//y finalmente la imagen ortorectificada
IplImage* copia_roi,*copia_roi2,*copia_roi3;
float factor;

// Declaramos Cadenas de caracteres para los archivos tanto de entrada como de salida que vamos a usar
char imagen1[100],imagen2[100],nombre_salida[100],nombre_salida_1[100],comando[100],nombre_salida_2[100];
gint t = 0;
IplImage* opencvImage,*img1,*img2,*img3,*img_corregir;
GdkPixbuf* pix;
FILE *archivo;
FILE *archivo_salida;
```

1.3 Programa Principal

Comentario:

Este programa se encarga del proceso de ortorectificación y georeferenciación de registros de imágenes. Se divide en 10 subrutinas principales las cuales se mencionarán en la sección 1.4.

Se inicia la lectura de un archivo de texto imágenes.txt donde se encuentra los nombres, tantos de la imagen satelital como del registro de imagen. Estos se almacenarán en variables de imagen a la espera de un evento de menú (presión de una casilla). Al iniciar el programa solo se puede esperar el evento de presión de la casilla Imagen Satelital, si presionamos cualquier otra diferente a la mencionada simplemente no se ejecuta nada. Este proceso de presión de casillas le da un orden al programa y además una interfaz grafica sencilla de usar.

Previamente a la entrada de espera de un evento de menú se verifica la condición de la cantidad de puntos de control a usar. Este valor debe ser mayor o igual a 3 y menor a 10. Este último valor puede aumentar, para fines de asignar un máximo se decidió por este valor.

Programa:

```
int main(int argc, char*argv[])
{

int i;
CvMat *FTran;

archivo = fopen("imagenes.txt","r");
fgets(imagen1,100,archivo);
```

```
fgets(imagen2,100,archivo);

for (i = 0; i<100; i++){

  if  (imagen1[i] == 10){
                        imagen1[i] = 0;
              }
              if (imagen2[i] == '\n'){
                        imagen2[i] = 0;
              }
}
printf("Leido: %s\n", imagen1);
printf("Leido: %s\n", imagen2);

img1 = cvLoadImage(imagen1,1);
img2 = cvLoadImage(imagen2,3);
img_corregir=cvLoadImage(imagen2,3);

archivo_salida = fopen("salida_ortorectificacion.txt", "r");

fgets(nombre_salida,100,archivo_salida);
fgets(nombre_salida_1,100,archivo_salida);
fgets(nombre_salida_2,100,archivo_salida);
for (i = 0; i<100; i++){

              if (nombre_salida[i] == 10){
                        nombre_salida[i] = 0;
              }
              if (nombre_salida_1[i] == '\n'){
                        nombre_salida_1[i] = 0;
              }
              if (nombre_salida_2[i] == '\n'){
                        nombre_salida_2[i] = 0;
              }
}


double *datos;
double origen_georeferenciax,origen_georeferenciay,resox,resoy;

  printf( "Proceso de Ortorectificacion de Imagenes\n"
            "Click Derecho - Zoom en la imagen\n"
    "Click Izquierdo - Toma datos de puntos control\n"
    "Eventos del teclado :\n"
    "d ----> Salida de la ventana de imagen al tomarse los puntos adecuados\n"
    "R-r ----> Reset de toma de puntos de control o reset de zoom\n");

while((puntos==0)||(puntos<3)||(puntos>10))
            {
            printf("Tome puntos de control en el rango de 0 a 10\n");
            printf("Â¿cuantos puntos desea tomar?\n");
            scanf("%d",&puntos);
            }

    muestra_ambas(img1,img2);

    GtkWidget *window;
    GtkWidget *button;
    GtkWidget *table;
    GtkWidget *box1;
    GtkWidget *drawing_area;
    GtkWidget *label;

    gtk_init (&argc, &argv);
    box1 = gtk_vbox_new(FALSE, 0);

    opencvImage=muestra_ambas(img1,img2);
    cvSaveImage("muestra.bmp",opencvImage,0);
    opencvImage = cvLoadImage("muestra.bmp", 1);
    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW (window), "       Ortorectificacion Manual");
    g_signal_connect (G_OBJECT (window), "destroy",
    G_CALLBACK (gtk_main_quit), NULL);

    drawing_area = gtk_drawing_area_new();
    gtk_widget_set_size_request(drawing_area, opencvImage->width, opencvImage->height);
```

```
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title (GTK_WINDOW (window), "Ortorectificacion de Imagenes Manual");
    gtk_signal_connect (GTK_OBJECT (window), "delete_event",
                        GTK_SIGNAL_FUNC (cerrar_menu), NULL);
    gtk_container_border_width (GTK_CONTAINER (window), 10);
    table = gtk_table_new (3, 3, TRUE);
    gtk_container_add (GTK_CONTAINER (window), box1);


    button = gtk_button_new_with_label ("Imagen Satelital");
    gtk_signal_connect (GTK_OBJECT (button), "clicked",
    GTK_SIGNAL_FUNC (evento_menu), (gpointer) "button1");
    gtk_table_attach_defaults (GTK_TABLE(table), button, 1, 2, 0, 1);
    gtk_widget_show (button);

    button = gtk_button_new_with_label ("Imagen a corregir");
    gtk_signal_connect (GTK_OBJECT (button), "clicked",
            GTK_SIGNAL_FUNC (evento_menu), (gpointer) "button2");
    gtk_table_attach_defaults (GTK_TABLE(table), button, 1, 2, 1, 2);
    gtk_widget_show (button);


    button = gtk_button_new_with_label ("Imagen ortorectificada");
    gtk_signal_connect (GTK_OBJECT (button), "clicked",
    GTK_SIGNAL_FUNC (evento_menu), (gpointer) "button3");
    gtk_table_attach_defaults (GTK_TABLE(table), button, 1, 2, 2, 3);
    gtk_widget_show (button);
    button = gtk_button_new_with_label ("Genera Geotiff");
    gtk_signal_connect (GTK_OBJECT (button), "clicked",
                        GTK_SIGNAL_FUNC (evento_menu), "button4");
    gtk_table_attach_defaults (GTK_TABLE(table), button, 2, 3, 1, 2);
    gtk_signal_connect (GTK_OBJECT (button), "clicked",
    GTK_SIGNAL_FUNC (cerrar_geotiff), (gpointer) "button4");
    gtk_widget_show (button);

    label = gtk_label_new ("Tome puntos de control orden Imagen satelital\n Imagen a corregir y Imagen
ortorectificada.\n"
                                "Click derecho--->Zoom Imagen Satelital.\n"
                                "Click Izquierdo--->Toma Punto Control.\n"
                                "Enter---> Fin toma puntos de control.\n"
                                "R-r----> Reset de puntos de control");

    gtk_table_attach_defaults (GTK_TABLE(table), label, 0, 1, 0, 3);
    gtk_widget_show (label);
    gtk_box_pack_start(GTK_BOX(box1),table, FALSE, TRUE, 0);
    gtk_widget_show(box1);
    gtk_box_pack_start(GTK_BOX(box1),drawing_area,FALSE,TRUE, 0);
    gtk_widget_show (table);
    gtk_widget_show(drawing_area);
    gtk_widget_show (window);
    g_signal_connect (G_OBJECT (drawing_area), "expose_event",
    G_CALLBACK (expose_event_callback), NULL);

  gtk_main();

   fclose(archivo);
   fclose(archivo_salida);
cvWaitKey(0);
return 0;

 }
```

1.4 Subrutinas Programa Principal

1.4.1 Subrutina evento_menu

Comentario:

En esta subrutina se espera un evento de la ventana de menú, como por ejemplo presión de la casilla Imagen Satelital, Imagen a corregir, Imagen Ortorectificada o Genera Geotiff. De acuerdo a la casilla presionada se crea una nueva ventana para mostrar la imagen especificada, para luego asignarle puntos de control. En el caso del evento Imagen Ortorectificada, se aplica directamente la transformación al registro de imagen y se muestra la imagen transformada. Por último la casilla Genera Geotiff se usa para pasar por línea de comandos una instrucción que hace uso de la utilidad de GDAL (gdal_translate) para georeferenciar la imagen ortorectificada.

Programa:

```
void evento_menu (GtkWidget *widget, gpointer data)
{

  if ((data=="button1")&&(codigo==0))
   { cvNamedWindow("satelital", 0);
     toma_puntos_control(img1,"satelital",puntos);
     codigo++;

                    }

  if ((data=="button2")&&(codigo==1))
  {  cvNamedWindow("satelital", 0);
     cvResizeWindow("satelital",400,300);
     cvShowImage("satelital", copia_roi);
     cvNamedWindow("corregir", 0);
     toma_puntos_control(img2,"corregir",puntos);
     codigo++;
      printf("Elvalor de codigo es:%d \n",codigo);
                    }

  if ((data=="button3")&&(codigo==2))
   {
              IplImage *img_recortada;
              CvMat *FTran;
     cvNamedWindow("Imagen Ortorectificada", 0);
              FTran = hallaFT();

     Aplica_transformacion(img_corregir,FTran);

     IplImage *img3 = cvLoadImage(nombre_salida,3);
     toma_puntos_control(img3,"Imagen Ortorectificada",puntos);
     codigo++;
   }

  if ((data=="button4")&&(codigo==3))
   {
     double *datos;
     double origen_georeferenciax,origen_georeferenciay,resox,resoy;
     cvDestroyWindow("satelital");
     cvReleaseImage(&copia_roi);
     datos= genera_datos_TFW ();
     origen_georeferenciax=*datos;
     origen_georeferenciay=*(datos+1);
     resox=*(datos+2);
     resoy=*(datos+3);
```

```
    genera_archivo_tif(origen_georeferenciax,origen_georeferenciay,resox,resoy);
    sprintf(comando,"gdal_translate -of GTiff -a_srs EPSG:32718 -mo TFW=YES %s
%s",nombre_salida,nombre_salida_2);
    system(comando);
    system("rm muestra*");
    codigo++;
   }
}
```

1.4.2 Subrutina toma_puntos_control

Comentario:

Esta subrutina, como su nombre lo indica, se encarga de la toma de puntos de control a cada imagen de acuerdo al evento presionado en el menú. Aquí se encuentran las opciones de reset de puntos de control, reset de zoom en la imagen satelital y por último la presión de la tecla enter que indica la toma total de puntos y la salida de esta función.

Programa:

```
void toma_puntos_control(IplImage* img,const char* name,int puntos_c)
{

Inicio_Satelital:

if (name=="satelital")
{
   contador_puntos_control_satelital=0;
   copia_roi = cvCloneImage( img );
   cvResizeWindow(name,800,600);
   cvShowImage(name, img);
   while(1) {

     cvSetMouseCallback( name,evento_mouse,copia_roi);
       char c = cvWaitKey(33);
     if((c == 10)&&(contador_puntos_control_satelital==3)) break;

     if(( c == 'r' )||( c == 'R' ))
     {
        goto Inicio_Satelital;
     }
  }
 }
 Inicio_Corregir:

 if (name=="corregir")
{
   contador_puntos_control_imagen_sin_corregir=0;
   copia_roi2 = cvCloneImage( img );
   cvResizeWindow(name, 800,600);
   cvShowImage(name, img);
    while(1) {

       cvSetMouseCallback( name,evento_mouse,copia_roi2);
       char c = cvWaitKey(33);
     if(( c == '10)&&(contador_puntos_control_imagen_sin_corregir==3)) break;

     if(( c == 'r' )||( c == 'R' ))
     {
```

```
        goto Inicio_Corregir;
      }
    }
}

Inicio_Resultado:

if (name=="Imagen Ortorectificada")
{
  contador_puntos_control_imagen_resultado=0;
  copia_roi3 = cvCloneImage( img );
  cvResizeWindow(name, 800,600);
  cvShowImage(name, img);

  while(1) {

       cvSetMouseCallback( name,evento_mouse,copia_roi3);
       char e = cvWaitKey(33);
     if(( e == 10)&&(contador_puntos_control_imagen_resultado==3)) break;

     if(( e == 'r' )||( e == 'R' ))
     {
       goto Inicio_Resultado;
     }
   }

    cvDestroyWindow("satelital");
    cvReleaseImage(&copia_roi);
}


cvDestroyWindow(name);
cvReleaseImage(&img);
}
```

1.4.3 Subrutina evento_mouse

Comentario:

Esta subrutina se encarga de verificar que evento del mouse se está produciendo, para ello se declara dos tipos, uno de ellos es la presión del click izquierdo, el cual indica la toma de un punto de control. El otro es la presión del click derecho que indica el zoom en la imagen satelital. Esta función trabaja en conjunto con la función, mencionada anteriormente, toma_puntos_control.

Programa:

```
void evento_mouse(int event, int x, int y, int flags, void *img)
{

            if (img ==copia_roi)
            {
        int clave =1;
          switch(event)
         {
            case CV_EVENT_LBUTTONDOWN:
            {
        if(contador_puntos_control_satelital<puntos)
```

```
                {
                 dibujar_puntos(x,y,copia_roi,clave);
                 matriz_satelital[contador_puntos_control_satelital]= cvPoint2D32f(x,y);
                }
              contador_puntos_control_satelital++;
              }
             break;
            case CV_EVENT_RBUTTONDOWN:
                {
       if(contador_puntos_control_satelital==0)
              {
            cvSetImageROI(img,cvRect( x-800, y-800,1000,1000));
            matriz_zoom[0]= cvPoint2D32f(x,y);
            copia_roi = cvCloneImage( img );
            cvSaveImage("zoom.bmp",copia_roi,0);
            cvResetImageROI(img);
            cvResizeWindow("satelital",1024,768);
            cvShowImage("satelital", copia_roi);
              }
            break;
                 }
           }
    }
    if (img ==copia_roi2)
    {
                            int clave =2;
      char d = cvWaitKey(33);
           switch(event) {
             case CV_EVENT_LBUTTONDOWN:
              {
               if(contador_puntos_control_imagen_sin_corregir<puntos)
                 {
              dibujar_puntos(x,y,copia_roi2,clave);
              matriz_corregir[contador_puntos_control_imagen_sin_corregir]= cvPoint2D32f(x,y);
                 }
             contador_puntos_control_imagen_sin_corregir++;
                 }
            break;
                  }
    }
              if (img ==copia_roi3)
              {
                            int clave =3;
      switch(event)
        {
        case CV_EVENT_LBUTTONDOWN:
          {
           if(contador_puntos_control_imagen_resultado<puntos)
                {
            dibujar_puntos(x,y,copia_roi3,clave);
            matriz_resultado[contador_puntos_control_imagen_resultado]= cvPoint2D32f(x,y);
                }
           contador_puntos_control_imagen_resultado++;
           break;
          }
        }
                  }
}
```

1.4.4 Subrutina dibujar_puntos

Comentario:

Esta subrutina se encarga de graficar círculos blancos sobre cada punto de control. Una vez elegido un punto y presionar el click izquierdo del mouse en la posición asignada, se llama a esta función y se proporciona una imagen

con un punto blanco sobrepuesto en esa posición. Este proceso se realiza siempre y cuando no se haya cumplido con la asignación total de puntos control.

Programa:

```
void dibujar_puntos(int x,int y,IplImage* imagen, int clave)
{
  IplImage* screenBuffer;

   if((clave==1)||(clave==2)){
            cvCircle(imagen, cvPoint(x,y),6, CV_RGB(255,255,255), -1, CV_AA, 0);
    screenBuffer=cvCloneImage(imagen);
            cvCircle(screenBuffer, cvPoint(x,y),8, CV_RGB(0,0,0), 1, CV_AA, 0);
            }
    else {
    cvCircle(imagen, cvPoint(x,y),6, CV_RGB(255,255,255), -1, CV_AA, 0);
    screenBuffer=cvCloneImage(imagen);
            cvCircle(screenBuffer, cvPoint(x,y),8, CV_RGB(0,0,0), 1, CV_AA, 0);
            }
    switch (clave){
      case 1 :
                cvShowImage( "satelital", screenBuffer );
                break;
              case 2 :
                cvShowImage( "corregir", screenBuffer );
                break;
              case 3 :
                cvShowImage( "Imagen Ortorectificada", screenBuffer );
                break;}

            }
```

1.4.5 Subrutina hallaFT

Comentario:

Esta subrutina se encarga de encontrar la matriz de transformación, para ello se requiere de las matrices donde se encuentran almacenadas las coordenadas pixeles de los puntos de control, tanto para la imagen satelital como el registro de imagen. Aquí se halla tres tipos de matrices:

La matriz de transformación Afín para tres puntos de control

La matriz de transformación Perspectiva para cuatro puntos de control

La matriz de transformación Perspectiva Homográfica para más de cuatro puntos de control.

Programa:

```
CvMat *hallaFT()
```

```
{
  int i,z,auxiliar;
  float factor_correcion;
 CvMat*FT;
    if (puntos>4)
              {
                        CvMat     *src1 = cvCreateMat(2, puntos, CV_32FC1);
                        CvMat     *dst1 = cvCreateMat(2, puntos, CV_32FC1);

                  factor_correcion=obten_factor();

        for (i = 0;i<puntos; i++){
                        *( (float*)CV_MAT_ELEM_PTR( *dst1, 0, i ) ) = ((matriz_corregir+i)->x);
                        *( (float*)CV_MAT_ELEM_PTR( *dst1, 1, i ) ) = ((matriz_corregir+i)->y);

                        *( (float*)CV_MAT_ELEM_PTR( *src1, 0, i ) ) = ((matriz_satelital+i)->x);
                        *( (float*)CV_MAT_ELEM_PTR( *src1, 1, i ) ) = ((matriz_satelital+i)->y);
                        }

      FT=cvCreateMat(3,3,CV_64FC1);


              z = cvFindHomography( dst1,src1, FT, CV_RANSAC,5, NULL);
          float           h[9];

            h[0] = CV_MAT_ELEM( *FT, float, 0, 0 );
            h[1] = CV_MAT_ELEM( *FT, float, 0, 1 );
            h[2] = CV_MAT_ELEM( *FT, float, 0, 2 );
            h[3] = CV_MAT_ELEM( *FT, float, 1, 0 );
            h[4] = CV_MAT_ELEM( *FT, float, 1, 1 );
            h[5] = CV_MAT_ELEM( *FT, float, 1, 2 );
            h[6] = CV_MAT_ELEM( *FT, float, 2, 0 );
            h[7] = CV_MAT_ELEM( *FT, float, 2, 1 );
            h[8] = CV_MAT_ELEM( *FT, float, 2, 2 );
            printf("\n%f %f %f %f %f %f %f %f %f \n", h[0], h[1], h[2], h[3], h[4], h[5], h[6], h[7], h[8] );

                    }
 else
          {   CvPoint2D32f src1[puntos];
            CvPoint2D32f dst1[puntos];

            if (puntos==3){

            FT= cvCreateMat(2, 3, CV_64FC1);
            factor_correcion=obten_factor(); }
            else{

            FT= cvCreateMat(3, 3, CV_64FC1);
            factor_correcion=1; }

            for (i = 0;i<puntos; i++){

              src1[i]= matriz_satelital[i];
              dst1[i]= matriz_corregir[i];
                                }
            for (i = 0;i<puntos; i++){
                    (src1+i)->x=factor_correcion*((src1+i)->x);
                    (src1+i)->y=factor_correcion*((src1+i)->y);
                                }
          if(puntos==3)
           cvGetAffineTransform(dst1,src1,FT);
          else{

          cvGetPerspectiveTransform(dst1,src1,FT);
              }
        }
return FT;

}
```

1.4.6 Subrutina Aplica_transformacion

Comentario:

Esta subrutina se encarga de generar la imagen ortorectificada. Para ello hace uso de la matriz de transformación hallada en la función hallaFT y la aplica al registro de imagen. Si se obtuvo una matriz de transformación afín se aplica la función cvWarpAffine para el remuestreo de imagen, caso contrario la función cvWarpPerspective.En ambos se aplica un método de interpolación denominado por opencv CV_WARP_FILL_OUTLIERS.

Programa:

```
Aplica_transformacion(IplImage *img, CvMat *FT)
{

IplImage* copia=cvCreateImage(cvSize(img->width*2.5,img->height*2.5),
                              img->depth,img->nChannels);

IplImage* arreglo;
IplImage* imagen_recortada;

if (puntos==3){

    cvWarpAffine(img,copia,FT,CV_WARP_FILL_OUTLIERS,cvScalarAll(0));

    imagen_recortada=corta_imagen(copia);

    cvSaveImage(nombre_salida, imagen_recortada,0);
    cvReleaseImage(&img);
    cvReleaseImage(&copia);

          }
     else{

    cvWarpPerspective(img,copia,FT, /*CV_INTER_LINEAR +*/ CV_WARP_FILL_OUTLIERS, cvScalarAll(0));
    arreglo=agranda_imagen(copia);
    imagen_recortada=corta_imagen(arreglo);
    cvSaveImage(nombre_salida,imagen_recortada,0);
    cvReleaseImage(&img);
    cvReleaseImage(&copia);
    cvReleaseImage(&arreglo);
        }

}
```

1.4.7 Subrutina obtén_factor

Comentario:

Esta subrutina se encarga de obtener la división entre el lado mayor del polígono generado al unir las coordenadas pixeles en la imagen satelital con el lado mayor al realizar el mismo proceso en el registro de imagen. Esto debido a que una imagen satelital tiene una menor resolución que el registro, por ello dos objetos dentro de esta imagen tendrá como distancia en pixeles

una cantidad menor con respecto a la medida en el registro. En consecuencia se requiere de un factor que compense esto y se calculó mediante la división mencionada. Este valor procura no agregar información a la imagen ortorectificada y a su vez no quitársela. Es por esto que se elige a los lados mayores que contienen la mayor cantidad de información en pixeles

Programa:

```
float obten_factor()
{
int i;
float coor0x,coor0y,coor1x,coor1y,coor0cx,coor0cy,coor1cx,coor1cy;
float distancia1_sa,distancia1_co;
float coor_u_sax,coor_u_say,coor_s0x,coor_s0y,distancia1_sa_u;
float coor_u_cox,coor_u_coy,coor_c0x,coor_c0y,distancia1_co_u;
float factor;
float mayor_satelital=0;
float mayor_corregir=0;

  for (i = 0;i<puntos-1; i++){

                coor0x=(matriz_satelital+i)->x;
      coor0y=(matriz_satelital+i)->y;
      coor1x=(matriz_satelital+1+i)->x;
      coor1y=(matriz_satelital+1+i)->y;

              distancia1_sa=sqrt(pow((coor1x-coor0x),2)+pow((coor1y-coor0y),2));
              if(distancia1_sa>mayor_satelital)
    mayor_satelital=distancia1_sa;

      coor0cx=(matriz_corregir)->x;
      coor0cy=(matriz_corregir)->y;
      coor1cx=(matriz_corregir+1)->x;
      coor1cy=(matriz_corregir+1)->y;

    distancia1_co=sqrt(pow((coor1cx-coor0cx),2)+pow((coor1cy-coor0cy),2));
    if(distancia1_co>mayor_corregir)
    mayor_corregir=distancia1_co;

                  }

    coor_u_sax= (matriz_satelital+puntos-1)->x;
    coor_u_say= (matriz_satelital+puntos-1)->y;
    coor_s0x=(matriz_satelital)->x;
    coor_s0y=(matriz_satelital)->y;
    distancia1_sa_u=sqrt(pow((coor_u_sax-coor_s0x),2)+pow((coor_u_say-coor_s0y),2));

    if(distancia1_sa_u>mayor_satelital)
      mayor_satelital=distancia1_sa_u;

    coor_u_cox= (matriz_corregir+puntos-1)->x;
    coor_u_coy= (matriz_corregir+puntos-1)->y;
    coor_c0x=(matriz_corregir)->x;
    coor_c0y=(matriz_corregir)->y;
    distancia1_co_u=sqrt(pow((coor_u_cox-coor_c0x),2)+pow((coor_u_coy-coor_c0y),2));

    if(distancia1_co_u>mayor_corregir)
      mayor_corregir=distancia1_co_u;

factor= mayor_corregir/mayor_satelital;

printf("factor de correcion=%f\n",factor);

return factor;

}
```

1.4.8 Subrutina corta_imagen

Comentario:

Esta subrutina se encarga de recortar la imagen ortorectificada. Para ello se hace una búsqueda pixel por pixel tanto en filas como columnas hasta encontrar información. El valor de los índices (i, j) al encontrar pixeles propios de la imagen, indicará a partir de que posición debemos de recortar para no perder datos de la imagen.

Programa:

```
IplImage *corta_imagen(IplImage* img) {

 long        i, j;
 long filas_ceros=0;
 long contador1=0;
 long offset_col=0;
 long offset_col_i=0;
 long columnas_ceros=0;
 long contador2=0;
 long offset_fil=0;
 long offset_fil_i=0;

      CvScalar s;

      for(j=0;j<(img->width);j++)
      {
         for(i=0;i<(img->height);i++)
          {

      s=cvGet2D(img,i,j);

            if( (s.val[0] != 0.0) || (s.val[1] != 0.0) || (s.val[2] != 0.0))
       {
       filas_ceros++;
       contador1++;
            if((filas_ceros==1)&&(contador1==1))
              offset_col=j;
            if(filas_ceros==1)
              offset_col_i=j;
       }

          }
     filas_ceros=0;
      }


 for(i=0;i<(img->height);i++)
      {
         for(j=0;j<(img->width);j++)
          {

      s=cvGet2D(img,i,j);

            if( (s.val[0] != 0.0) || (s.val[1] != 0.0) || (s.val[2] != 0.0))
       {
       columnas_ceros++;
```

```
      contador2++;
           if((columnas_ceros==1)&&(contador2==1))
             offset_fil=i;
           if(columnas_ceros==1)
             offset_fil_i=i;
      }

     }
   columnas_ceros=0;
}

cvSetImageROI(img,cvRect(offset_col,offset_fil,offset_col_i-offset_col,offset_fil_i-offset_fil));

  return img;
}
```

1.4.9 Subrutina genera_archivo_tif

Comentario:

Esta subrutina se encarga de crear un archivo de texto TFW donde se indicarán datos como origen de coordenadas(en metros) de la imagen Geotiff, así como sus resoluciones en ambos ejes(x e y).

Programa:

```
void genera_archivo_tif (double origen_georeferenciax,double origen_georeferenciay,double resox,double resoy)
{
FILE *almacena;
 resoy=resoy*-1;
double texto[2];
      texto[0]=origen_georeferenciax;
      texto[1]=origen_georeferenciay;

almacena = fopen(nombre_salida_1,"w");
fprintf(almacena,"%f\n",resox);
fprintf(almacena,"%f\n",0.0000000000);
fprintf(almacena,"%f\n",0.0000000000);
fprintf(almacena,"%f\n",resoy);
fprintf(almacena,"%f\n",*texto);
fprintf(almacena,"%f\n",*(texto+1));
fclose(almacena);;

}
```

1.4.10 Subrutina genera_datos_TFW

Comentario:

Esta subrutina se encarga de crear los datos que se incluirán en el archivo de texto TFW, para ello se hace uso de los pixeles de la imagen satelital para obtener resoluciones en metros/pixeles reales. Finalmente se hace una

correspondencia con la imagen ortorectificada y se halla el valor de resolución y el origen de coordenadas (en metros) de esta última.

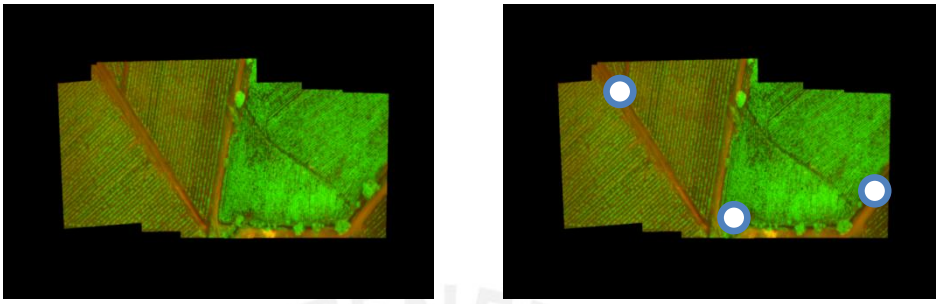
Programa:

```
 double* genera_datos_TFW ()

{   double matriz_datos[4];
    double* p;
            double nuevo_origenpx,nuevo_origenpy,resox,resoy,nuevo_origencx,nuevo_origency;
    double coordenada1_sax,coordenada1_say,coordenada2_sax,coordenada2_say;
    double coordenada1_resultadox,coordenada1_resultadoy,coordenada2_resultadox,coordenada2_resultadoy;
    double origen_cortadox,origen_cortadoy,origen_georeferenciax,origen_georeferenciay;

  origen_cortadox=204619.500000+(0.5*origenx_satelital);
  origen_cortadoy=8826391.000000-(0.5*origeny_satelital);
  nuevo_origenpx=((matriz_zoom)->x)-800;
  nuevo_origenpy=((matriz_zoom)->y)-800;
  nuevo_origencx= origen_cortadox+(0.500000*nuevo_origenpx);
  nuevo_origency= origen_cortadoy+((-0.500000)*nuevo_origenpy);
  coordenada1_sax=(((matriz_satelital)->x)*(0.500000))+nuevo_origencx;
  coordenada1_say=(((matriz_satelital)->y)*(-0.500000))+nuevo_origency;
  coordenada2_sax=(((matriz_satelital+1)->x)*(0.500000))+nuevo_origencx;
  coordenada2_say=(((matriz_satelital+1)->y)*(-0.500000))+nuevo_origency;

printf("La coordenada satelital 1x es: %f\n",coordenada1_sax);
printf("La coordenada satelital 1y es: %f\n",coordenada1_say);
printf("La coordenada satelital 2x es: %f\n",coordenada2_sax);
printf("La coordenada satelital 2y es: %f\n",coordenada2_say);

    coordenada1_resultadox=(matriz_resultado)->x;
    coordenada1_resultadoy=(matriz_resultado)->y;
    coordenada2_resultadox=(matriz_resultado+1)->x;
    coordenada2_resultadoy=(matriz_resultado+1)->y;

   resox=(coordenada2_sax-coordenada1_sax)/(coordenada2_resultadox-coordenada1_resultadox);
if (resox<0)
  resox=resox*-1;
resoy=(coordenada2_say-coordenada1_say)/(coordenada2_resultadoy-coordenada1_resultadoy);
if (resoy<0)
  resoy=resoy*-1;

origen_georeferenciax=coordenada1_sax-(resox*((matriz_resultado)->x));
origen_georeferenciay=coordenada1_say+(resoy*((matriz_resultado)->y));

      matriz_datos[0]=origen_georeferenciax;
      matriz_datos[1]=origen_georeferenciay;
      matriz_datos[2]=resox;
      matriz_datos[3]=resoy;

 p=matriz_datos;
 return p;

}
```

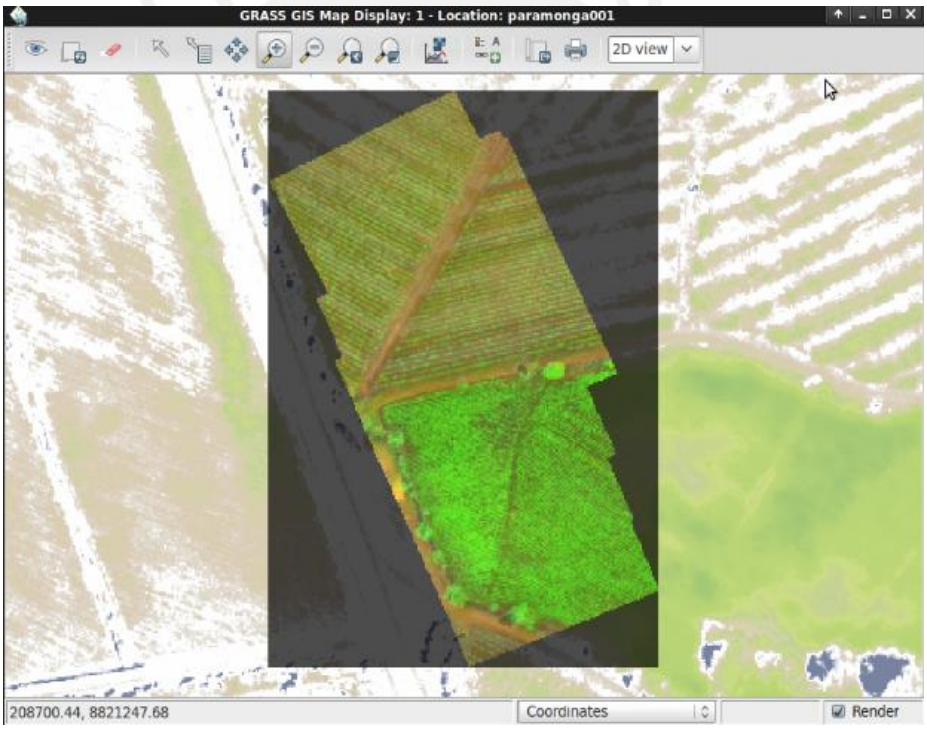
Anexo2: Pruebas Totales

Tal y como se presentó en el capítulo 4 de la memoria descriptiva, se presenta ahora todas la pruebas realizadas a los registros de imagen adquiridos desde la aeronave.

Imagen1: Registro 007



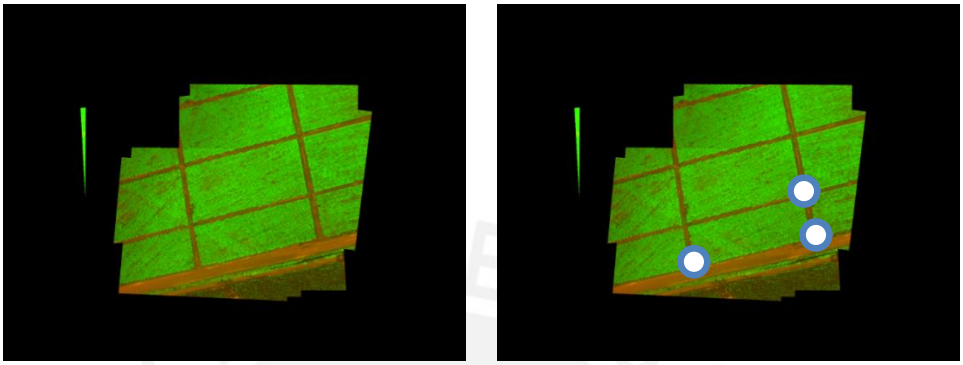
(a) (b)



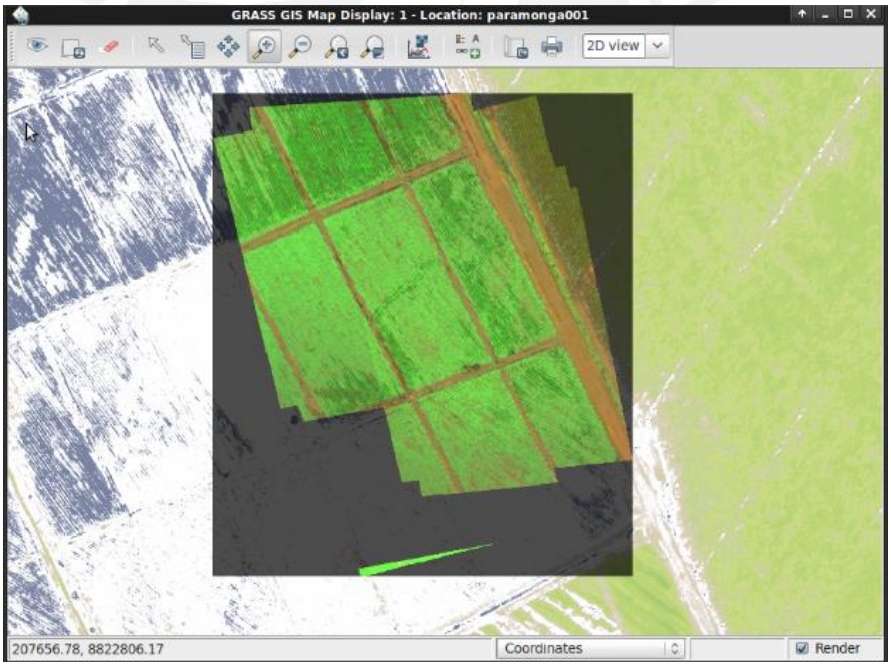
(c)

Figura1. (a) Registro de Imagen 007(b) Toma de puntos de control. (c) Superposición de Imagen Ortorectificada007 sobre Imagen Satelital

Imagen2: Registro 003



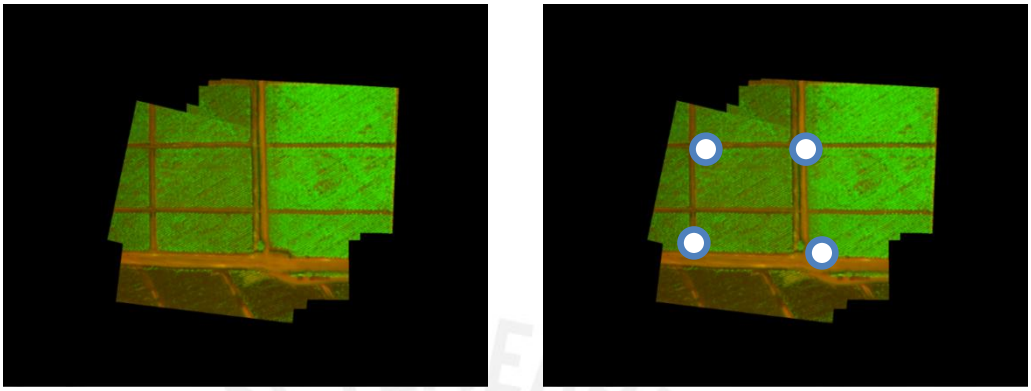
(a) (b)



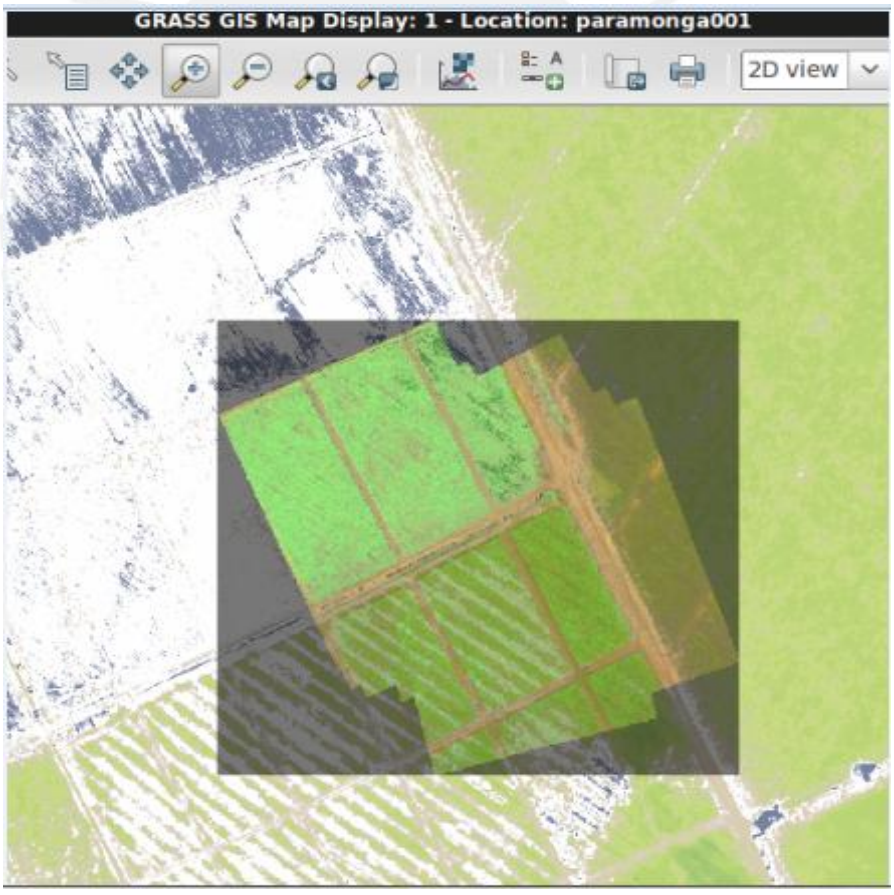
(c)

Figura2. (a) Registro de Imagen 003(b) Toma de puntos de control. (c) Superposición de Imagen Ortorectificada003 sobre Imagen Satelital

Imagen3: Registro 004



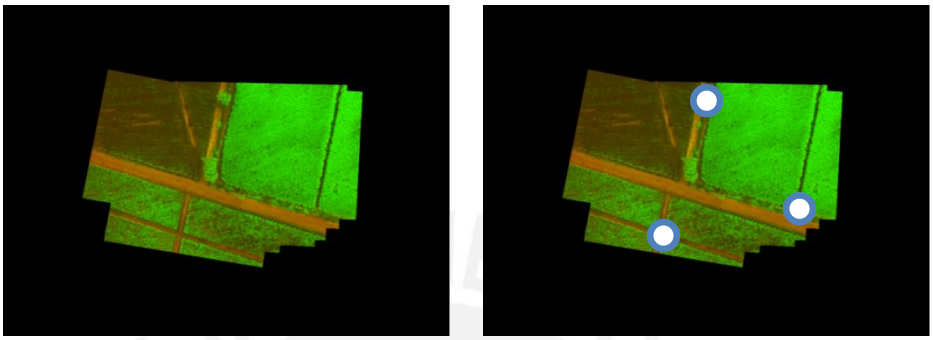
(a) (b)



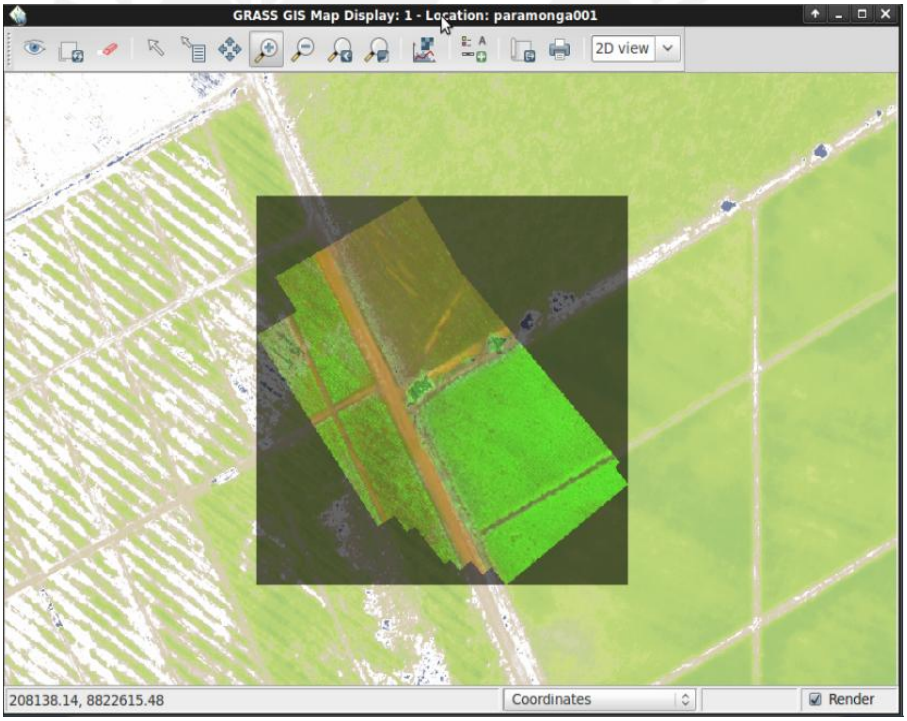
(c)

Figura3. (a) Registro de Imagen 004(b) Toma de puntos de control. (c) Superposición de Imagen Ortorectificada004 sobre Imagen Satelital

Imagen4: Registro 0016



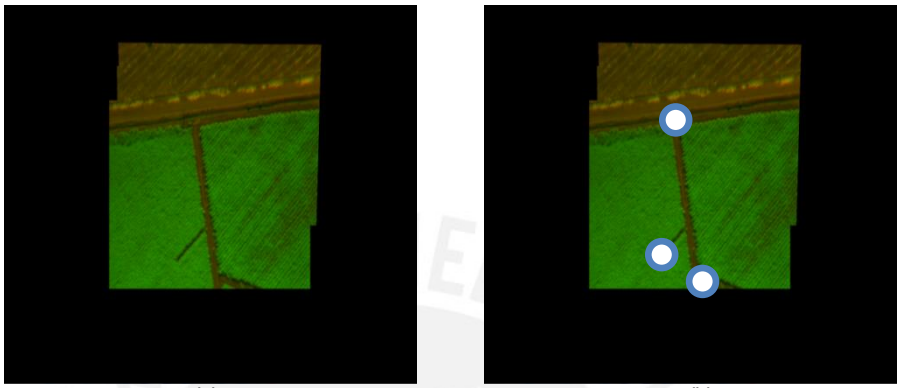
(a) (b)



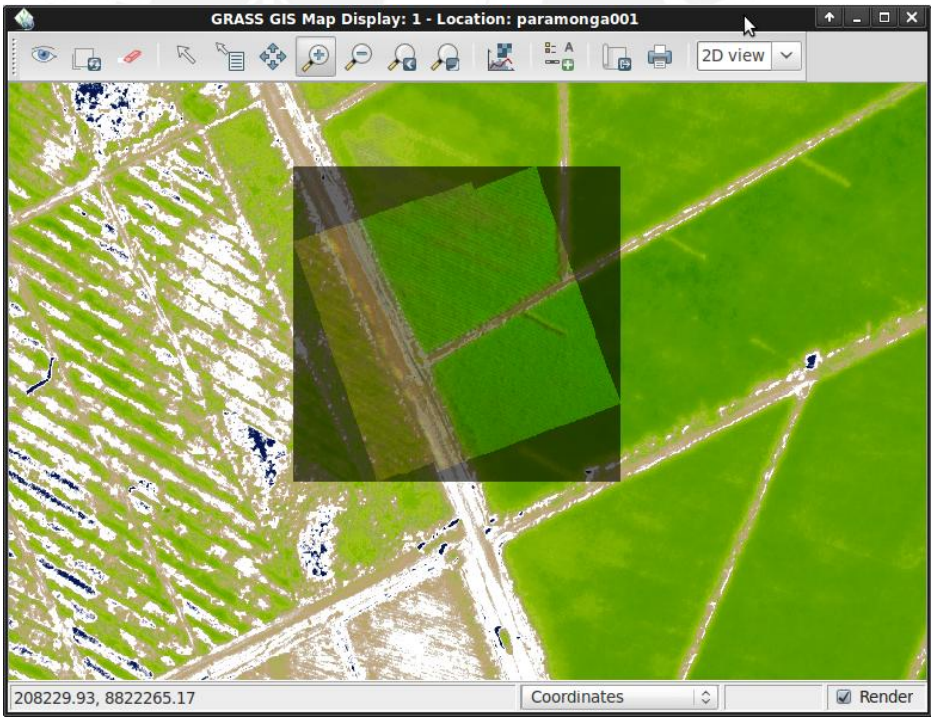
(c)

Figura4. (a) Registro de Imagen 0016(b) Toma de puntos de control. (c) Superposición de Imagen Ortorectificada0016 sobre Imagen Satelital

Imagen5: Registro 0025



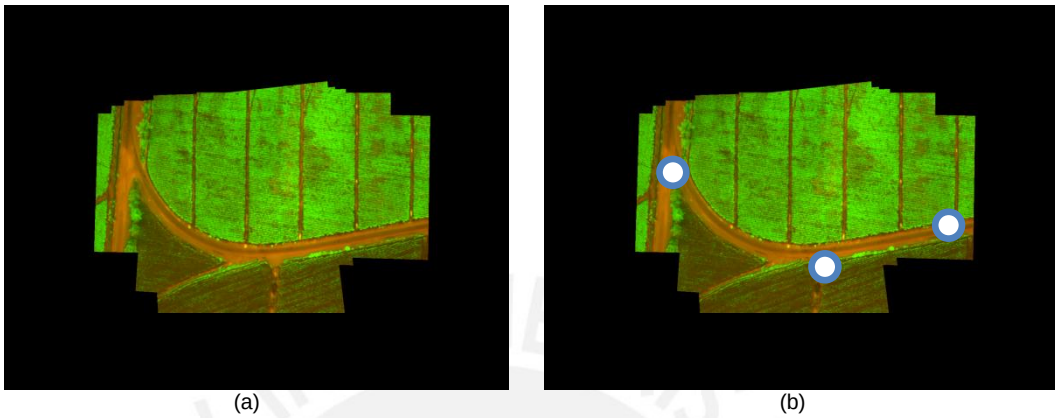
(a) (b)



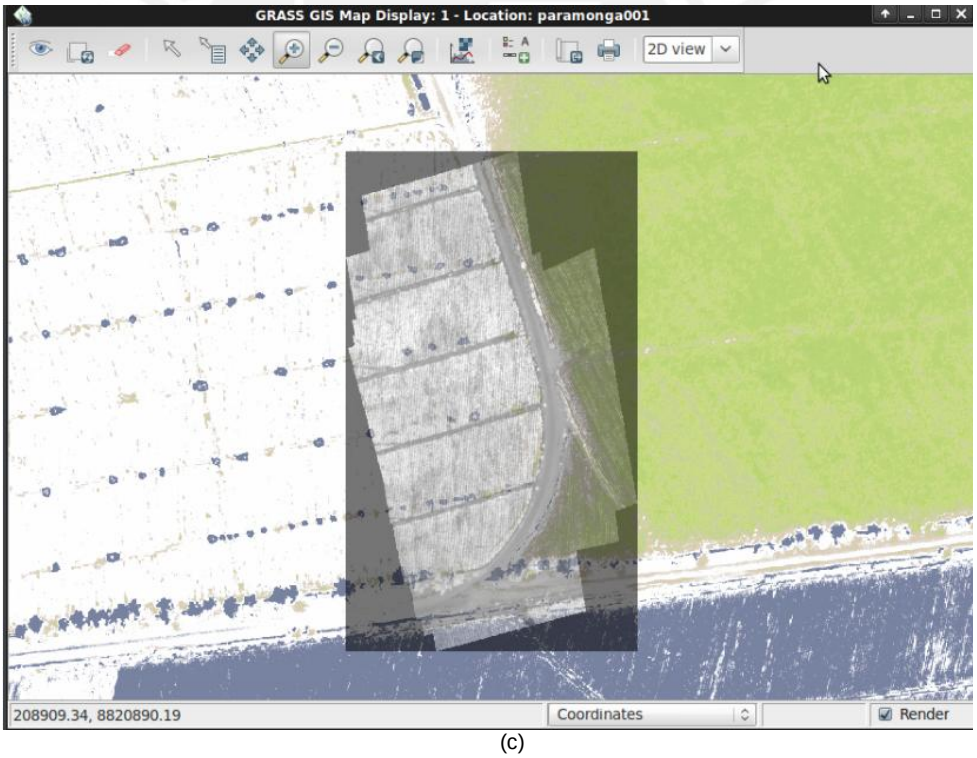
(c)

Figura5. (a) Registro de Imagen 0025(b) Toma de puntos de control. (c) Superposición de Imagen Ortorectificada0025 sobre Imagen Satelital.

Imagen6: Registro 0014



(a) (b)



(c)

Figura6. (a) Registro de Imagen 0014(b) Toma de puntos de control. (c) Superposición de Imagen Ortorectificada0014 sobre Imagen Satelital.

Imagen7: Registros 0016 y 004

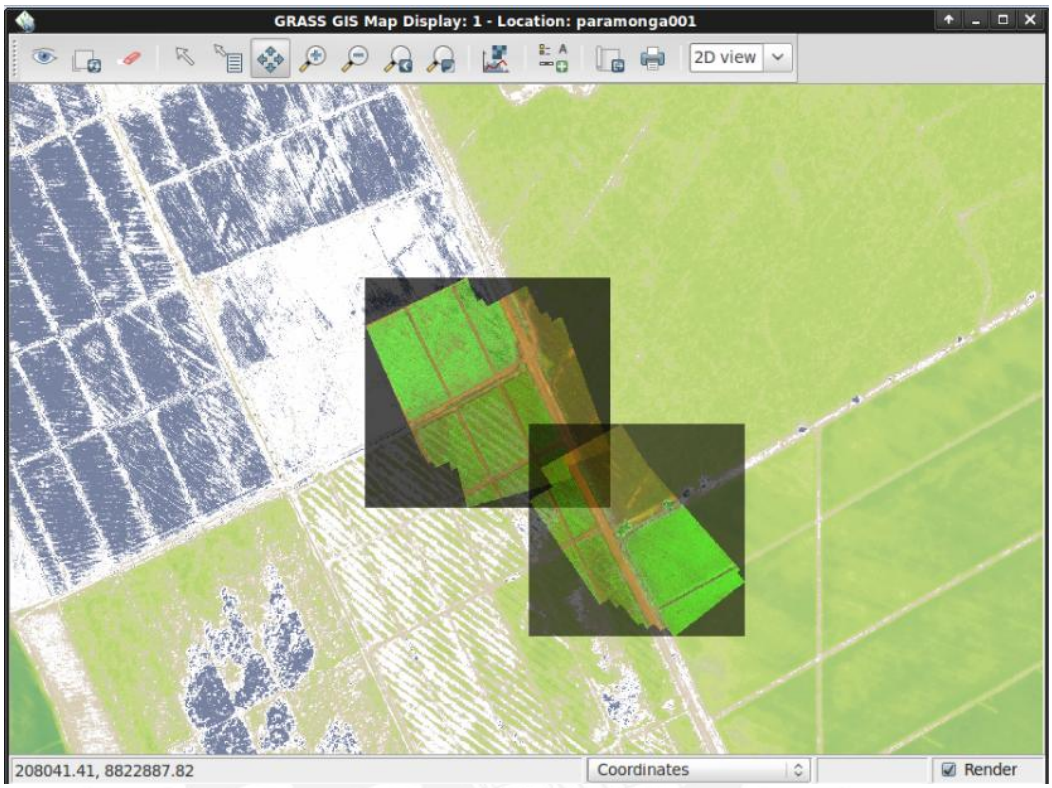


Figura7: Superposición de registros 0016 y 004 sobre Imagen Satelital

Imagen8: Registros Consecutivos

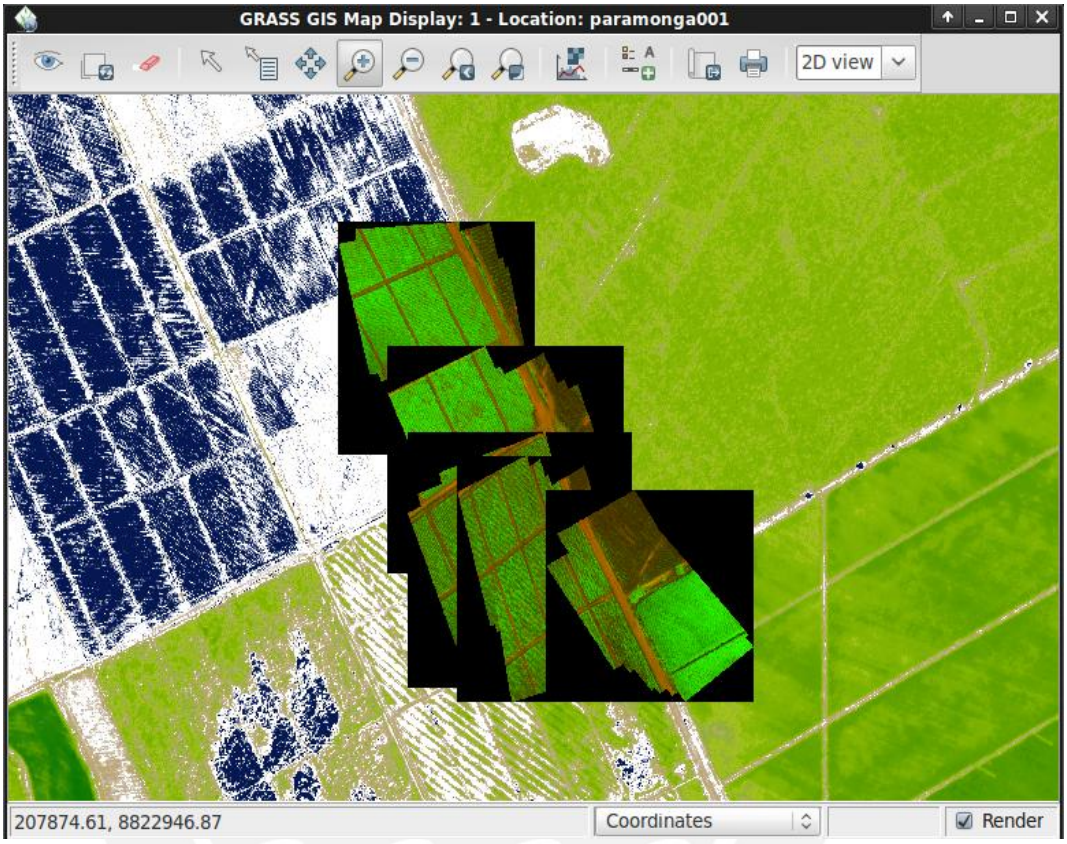


Figura7: Superposición de 6 registros sobre Imagen Satelital.