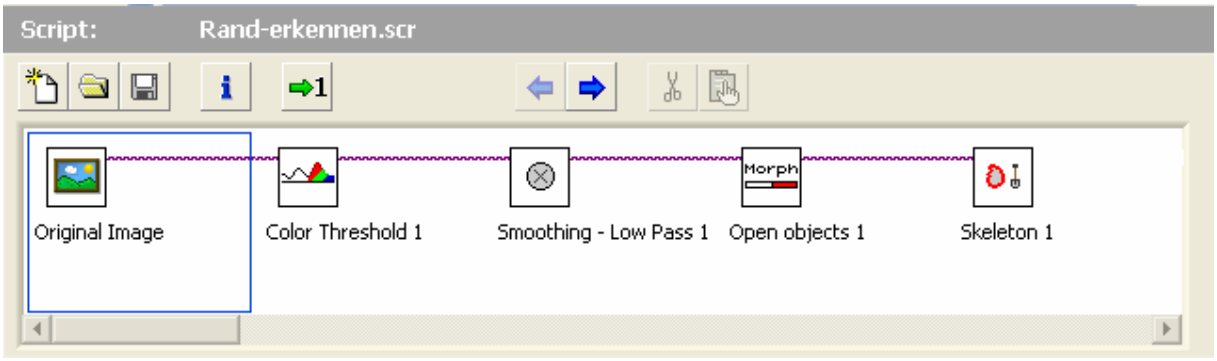
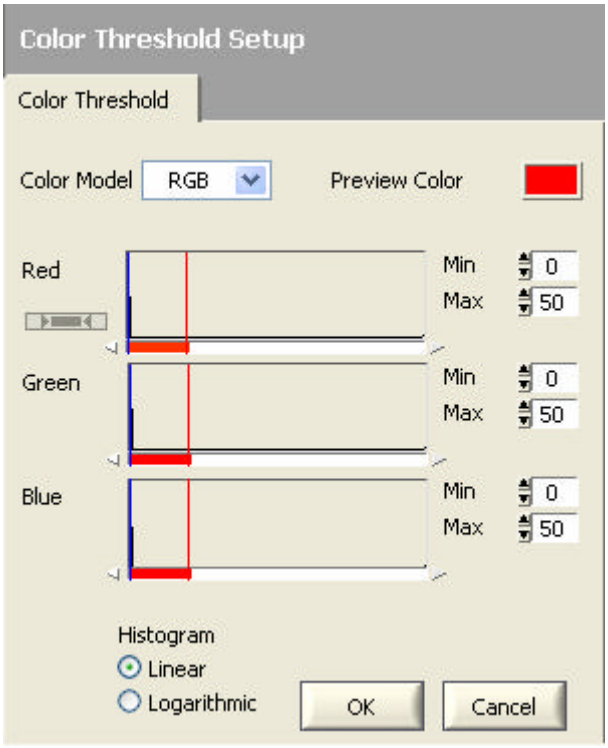
Mit dem Vison-Bulider habe ich folgendes Skript erstelle.

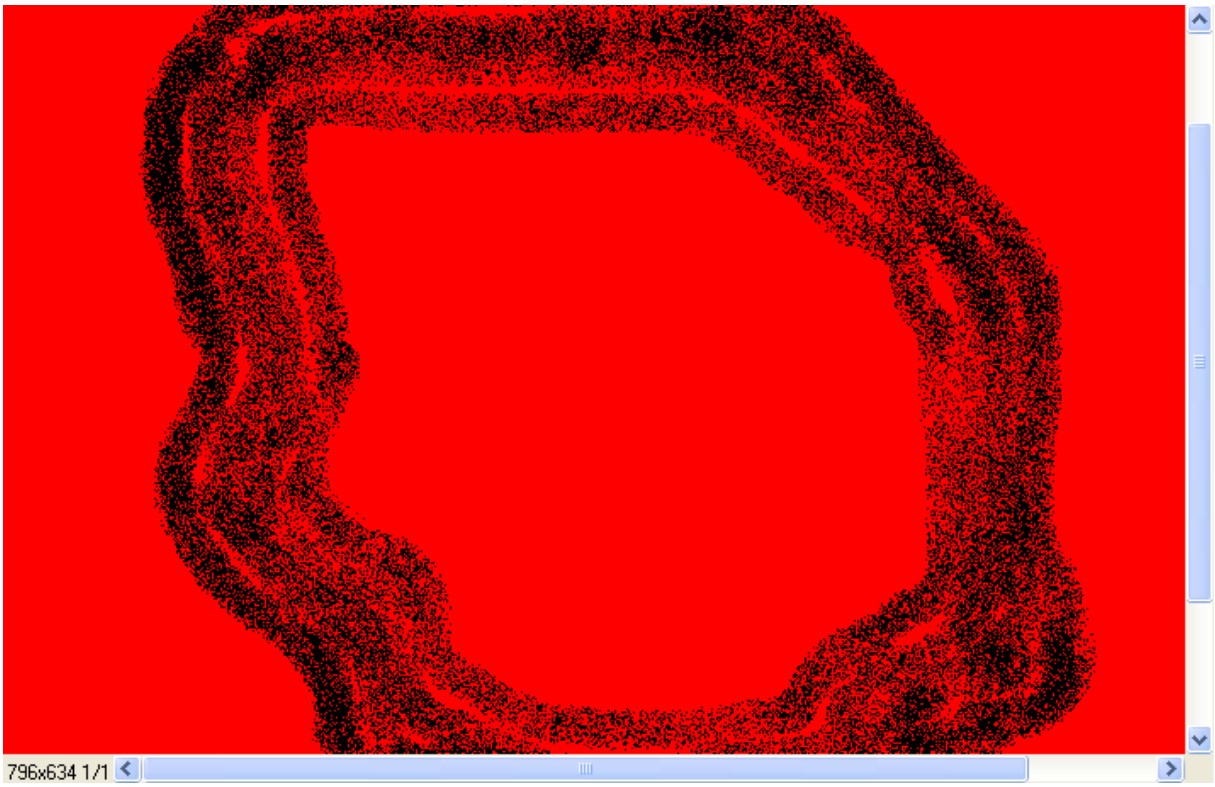


Dieses Skript ist als .skr – Datei abgespeichert und kann direkt in den „NI Vision Assistent“ geladen werden.

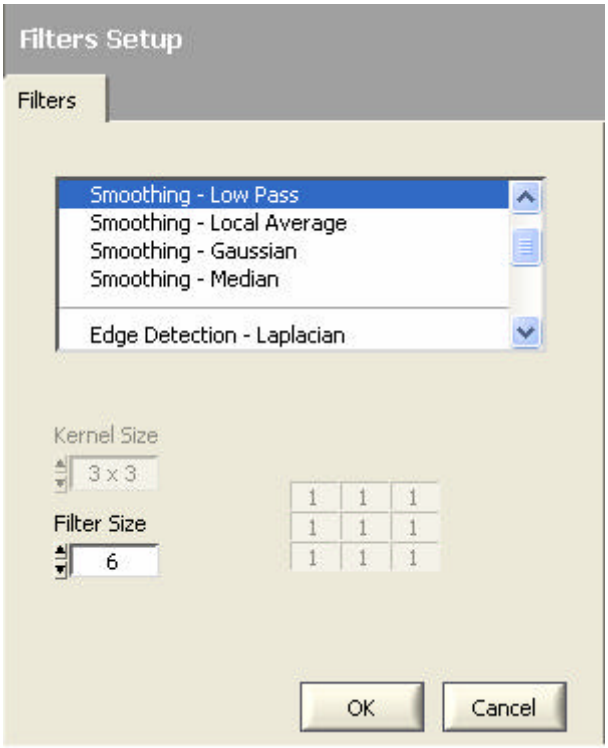
Color Threshold hat folgende Einstellungen



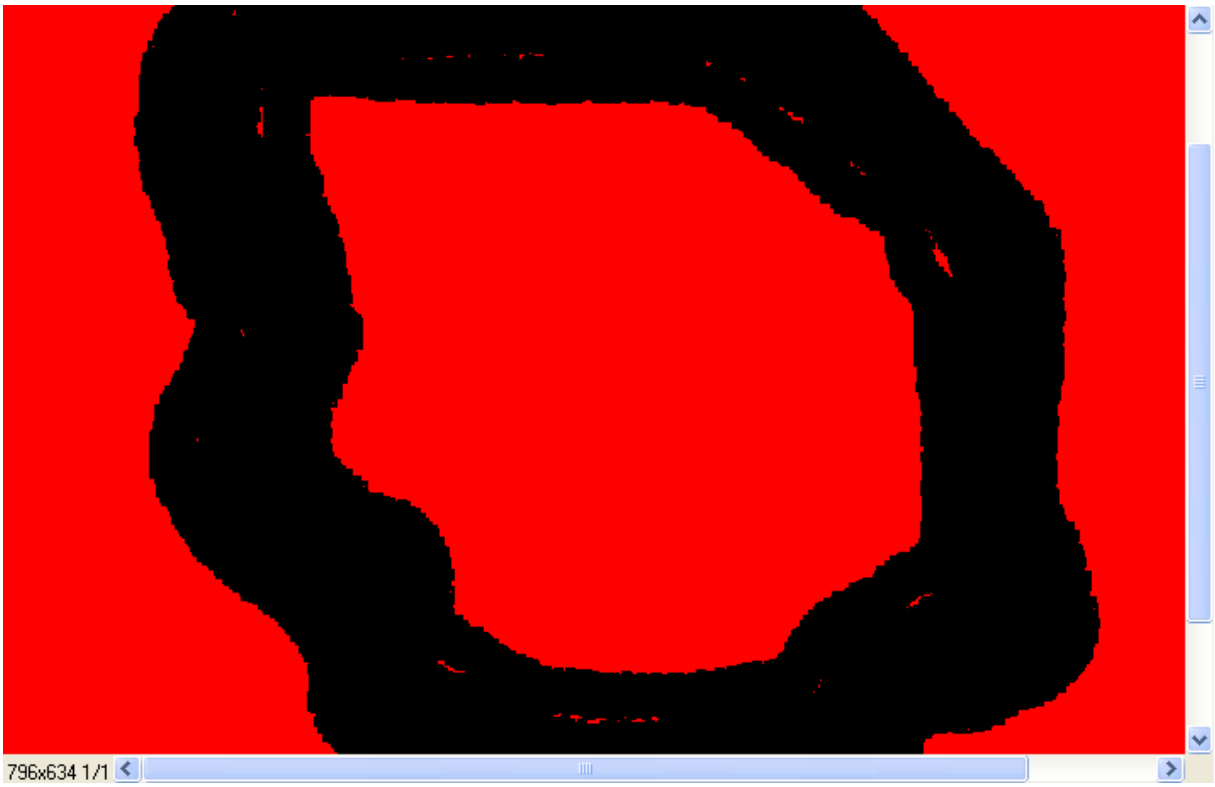
Danach ergibt sich folgendes Bild



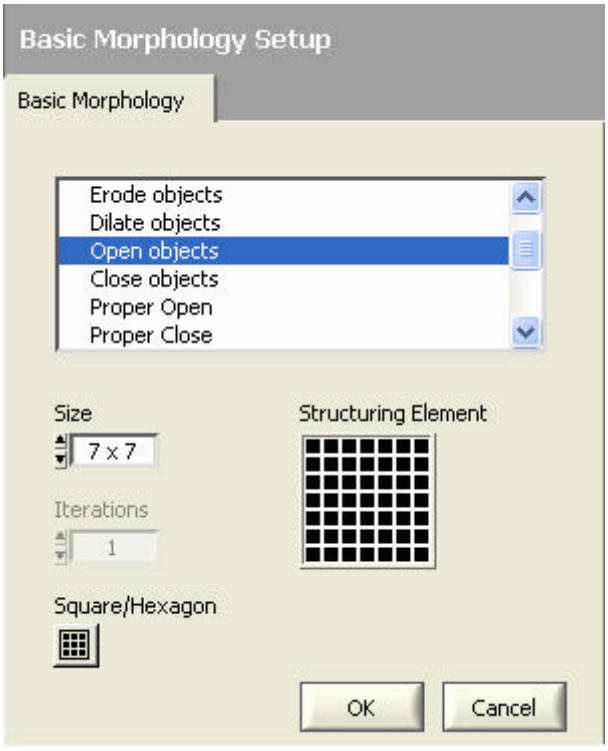
Smooting Lowpass 1 hat folgenden Einstellungen



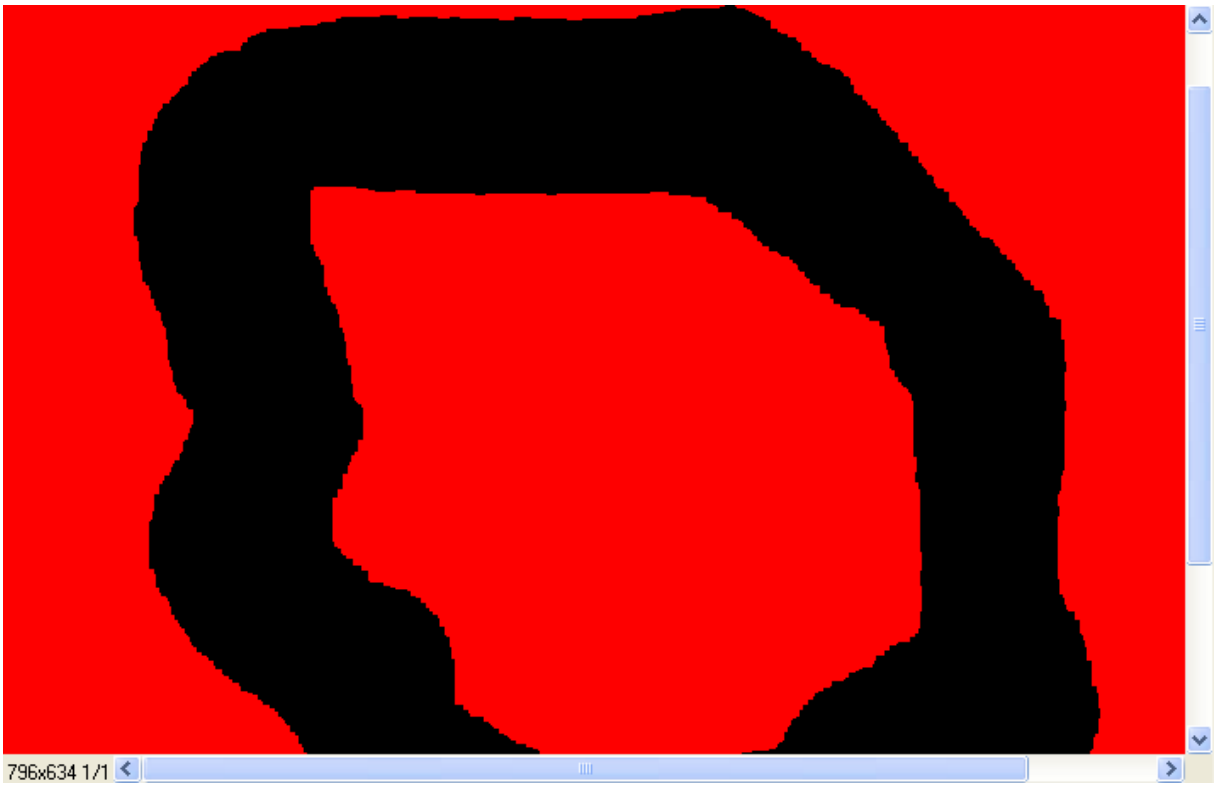
Danach ergibt sich folgendes Bild



Open objekt 1 hat folgenden Einstellungen

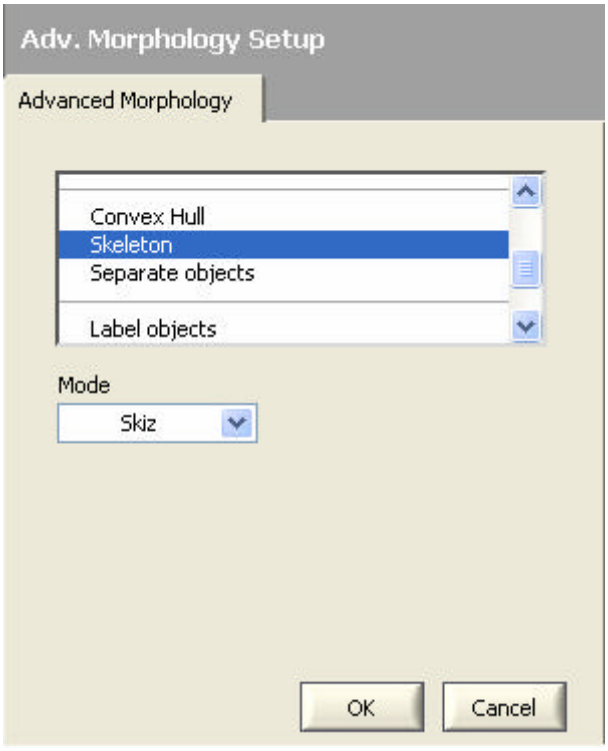


Danach ergibt sich folgendes Bild



Es sind die kleinen roten Punkete geschlossen.

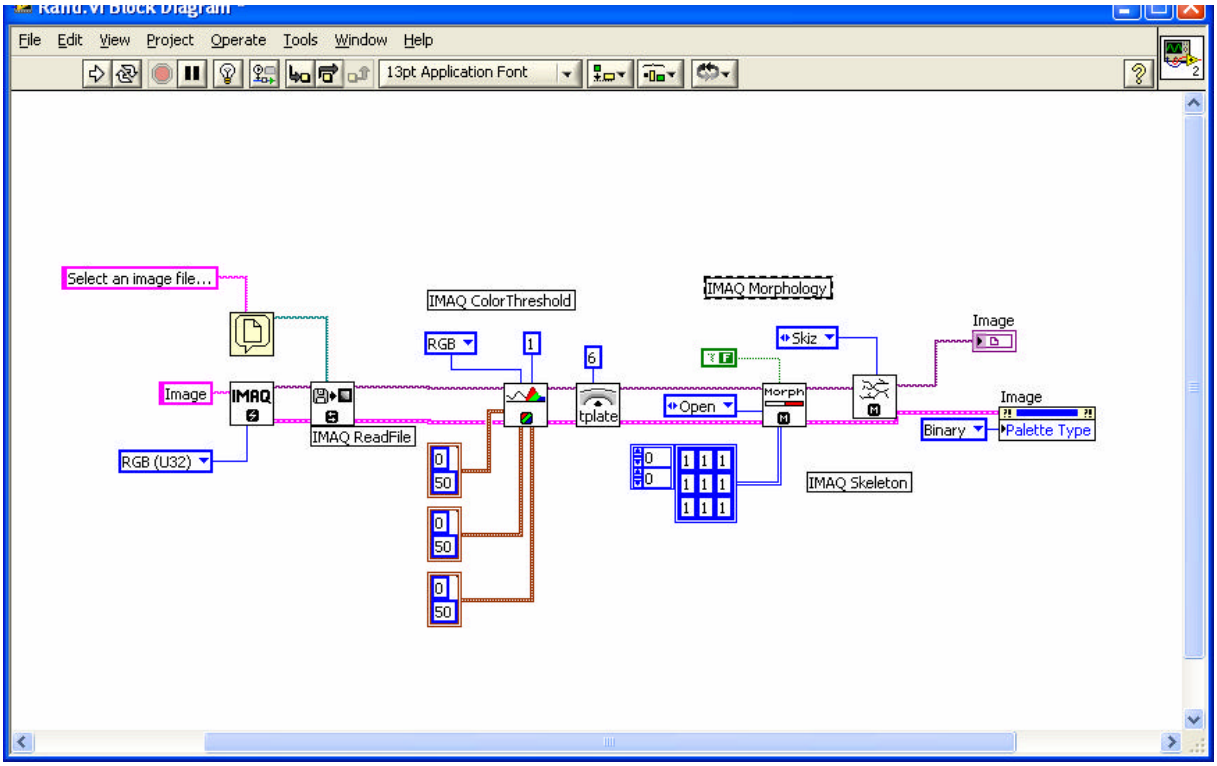
Skelektion 1 hat folgende Einstellungen



Danach ergibt sich folgendes Bild

Man kann mit dem Vision Assistent aus dem Skript ein LabVIEW oder C Programm erstellen.

Vom Vison Assistent erstelltes LabVIEW Programm



Erstelltes C- Programm

xxx.h Header

```
//*****************************************************************************
*
//* WARNING: This file was automatically generated.  Any changes you make
*
//*          to this file will be lost if you generate the file again.
*
//*****************************************************************************
*
#ifndef IMAGEPROCESSING_TASK_INCLUDE
#define IMAGEPROCESSING_TASK_INCLUDE
#include <nivision.h>
#include <nimachinevision.h>
#ifdef __cplusplus
      extern "C" {
#endif

int IVA_ProcessImage(Image *image);

#ifdef __cplusplus
      }
#endif

#endif // ifndef IMAGEPROCESSING_TASK_INCLUDE
```

xxx.C

```
//*********************************************************************
*
//* WARNING: This file was automatically generated.  Any changes you make
*
//*          to this file will be lost if you generate the file again.
*
//*********************************************************************
*
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <nivision.h>
#include <nimachinevision.h>
#include <windows.h>

// If you call Machine Vision functions in your script, add
NIMachineVision.c to the project.

#define MAX_BUFFERS 10
#define INIT_POINTS_ARRAY_ELEMENTS 100

#define VisionErrChk(Function) {if (!(Function)) {success = 0; goto
Error;}}

typedef struct IVA_Data_Struct
{
    Image* buffers[MAX_BUFFERS];    // Vision Assistant Image Buffers
    PointFloat* pointsResults;      // Array of points used for Caliper
Measurements
    int numPoints;                  // Number of points allocated in the
pointsResults array
    int pointIndex;                 // Insertion Point
} IVA_Data;


static IVA_Data* IVA_InitData(void);
static int IVA_DisposeData(IVA_Data* ivaData);
static int IVA_CLRThreshold(Image* image, int min1, int max1, int min2, int
max2, int min3, int max3, int colorMode);

int IVA_ProcessImage(Image *image)
{
      int success = 1;
    IVA_Data *ivaData;
    int pKernel[49] = {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
        1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
    StructuringElement structElem;

    // Initializes internal data (buffers and array of points for caliper
measurements)
    VisionErrChk(ivaData = IVA_InitData());

      VisionErrChk(IVA_CLRThreshold(image, 0, 50, 0, 50, 0, 50, IMAQ_RGB));

    //--------------------------------------------------------------//
    //                         Filters: Low Pass                    //
    //--------------------------------------------------------------//
```

```
    // Filters an image using a non-linear filter.
    VisionErrChk(imaqLowPass(image, image, 6, 6, 50.0, NULL));

    //-------------------------------------------------------------------//
    //                          Basic Morphology                         //
    //-------------------------------------------------------------------//

    // Sets the structuring element.
    structElem.matrixCols = 7;
    structElem.matrixRows = 7;
    structElem.hexa = FALSE;
    structElem.kernel = pKernel;

    // Applies a morphological transformation to the binary image.
    VisionErrChk(imaqMorphology(image, image, IMAQ_OPEN, &structElem));

    //-------------------------------------------------------------------//
    //                    Advanced Morphology: Skeleton                  //
    //-------------------------------------------------------------------//

    // Calculates the skeleton of the particles inside the image.
    VisionErrChk(imaqSkeleton(image, image, IMAQ_SKELETON_INVERSE));

    // Releases the memory allocated in the IVA_Data structure.
    IVA_DisposeData(ivaData);

Error:
      return success;
}

////////////////////////////////////////////////////////////////////////////
/////
//
// Function Name: IVA_CLRThreshold
//
// Description  : Thresholds a color image.
//
// Parameters   : image      -  Input image
//                min1       -  Minimum range for the first plane
//                max1       -  Maximum range for the first plane
//                min2       -  Minimum range for the second plane
//                max2       -  Maximum range for the second plane
//                min3       -  Minimum range for the third plane
//                max3       -  Maximum range for the third plane
//                colorMode  -  Color space in which to perform the
threshold
//
// Return Value : success
//
////////////////////////////////////////////////////////////////////////////
/////
static int IVA_CLRThreshold(Image* image, int min1, int max1, int min2, int
max2, int min3, int max3, int colorMode)
{
    int success = 1;
    Image* thresholdImage;
    Range plane1Range;
    Range plane2Range;
    Range plane3Range;


    //-------------------------------------------------------------------//
```

```
    //                          Color Threshold                          //
    //-------------------------------------------------------------------//

    // Creates an 8 bit image for the thresholded image.
    VisionErrChk(thresholdImage = imaqCreateImage(IMAQ_IMAGE_U8, 7));

    // Set the threshold range for the 3 planes.
    plane1Range.minValue = min1;
    plane1Range.maxValue = max1;
    plane2Range.minValue = min2;
    plane2Range.maxValue = max2;
    plane3Range.minValue = min3;
    plane3Range.maxValue = max3;

    // Thresholds the color image.
    VisionErrChk(imaqColorThreshold(thresholdImage, image, 1, c olorMode,
&plane1Range, &plane2Range, &plane3Range));

    // Copies the threshold image in the souce image.
    VisionErrChk(imaqDuplicate(image, thresholdImage));

Error:
    imaqDispose(thresholdImage);

    return success;
}


////////////////////////////////////////////////////////////////////////////
/////
//
// Function Name: IVA_InitData
//
// Description  : Initializes data for buffer management and caliper
points.
//
// Parameters   : None
//
// Return Value : success
//
////////////////////////////////////////////////////////////////////////////
/////
static IVA_Data* IVA_InitData(void)
{
    int success = 1;
    IVA_Data* ivaData = NULL;
    int i;


    // Allocate the data structure.
    VisionErrChk(ivaData = (IVA_Data*)malloc(sizeof (IVA_Data)));

    // Initializes the image pointers to NULL.
    for (i = 0 ; i < MAX_BUFFERS ; i++)
        ivaData->buffers[i] = NULL;

    // Initializes the array of points to INIT_POINTS_ARRAY_ELEMENTS
elements.
    ivaData->numPoints = INIT_POINTS_ARRAY_ELEMENTS;

    ivaData->pointsResults = (PointFloat*)malloc(ivaData ->numPoints *
sizeof(PointFloat));
    ivaData->pointIndex = -1;
```

```
Error:
    return ivaData;
}


//////////////////////////////////////////////////////////////////////////
/////
//
// Function Name: IVA_DisposeData
//
// Description  : Releases the memory allocated in the IVA_Data structure
//
// Parameters   : ivaData  -  Internal data structure
//
// Return Value : success
//
////////////////////////////////////////////////////////////////////////// ///
/////
static int IVA_DisposeData(IVA_Data* ivaData)
{
    int i;


    // Releases the memory allocated for the image buffers.
    for (i = 0 ; i < MAX_BUFFERS ; i++)
        imaqDispose(ivaData->buffers[i]);

    // Releases the memory allocated for the array of points.
    free(ivaData->pointsResults);
    ivaData->numPoints = 0;
    ivaData->pointIndex = -1;

    free(ivaData);

    return 1;
}
```