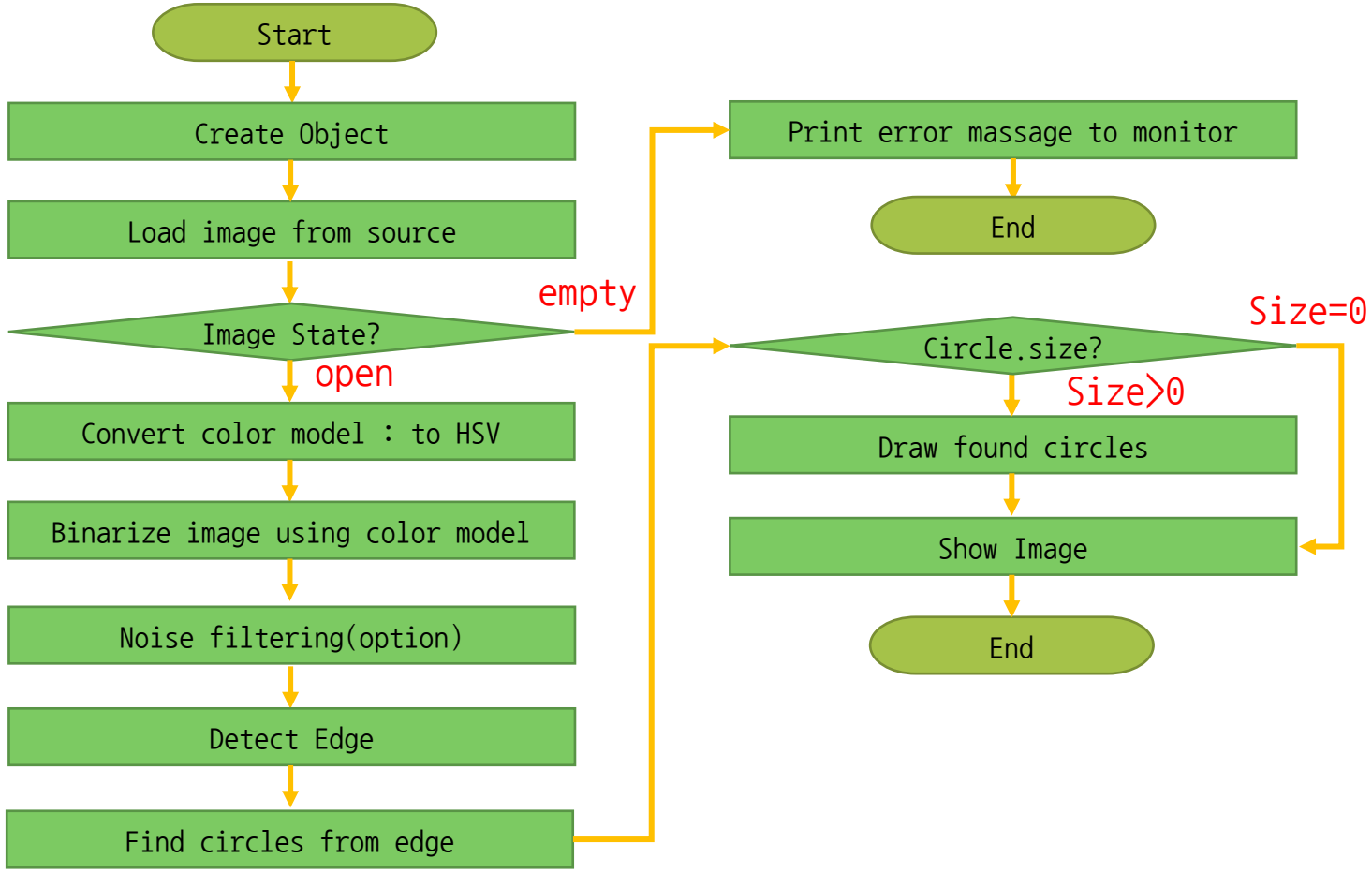
연구실 세미나 : 프로그래밍 교육 (OpenCV)

모서리 검출 및 타원 찾기
Edge Detecting, Circle Fitting

Amylose

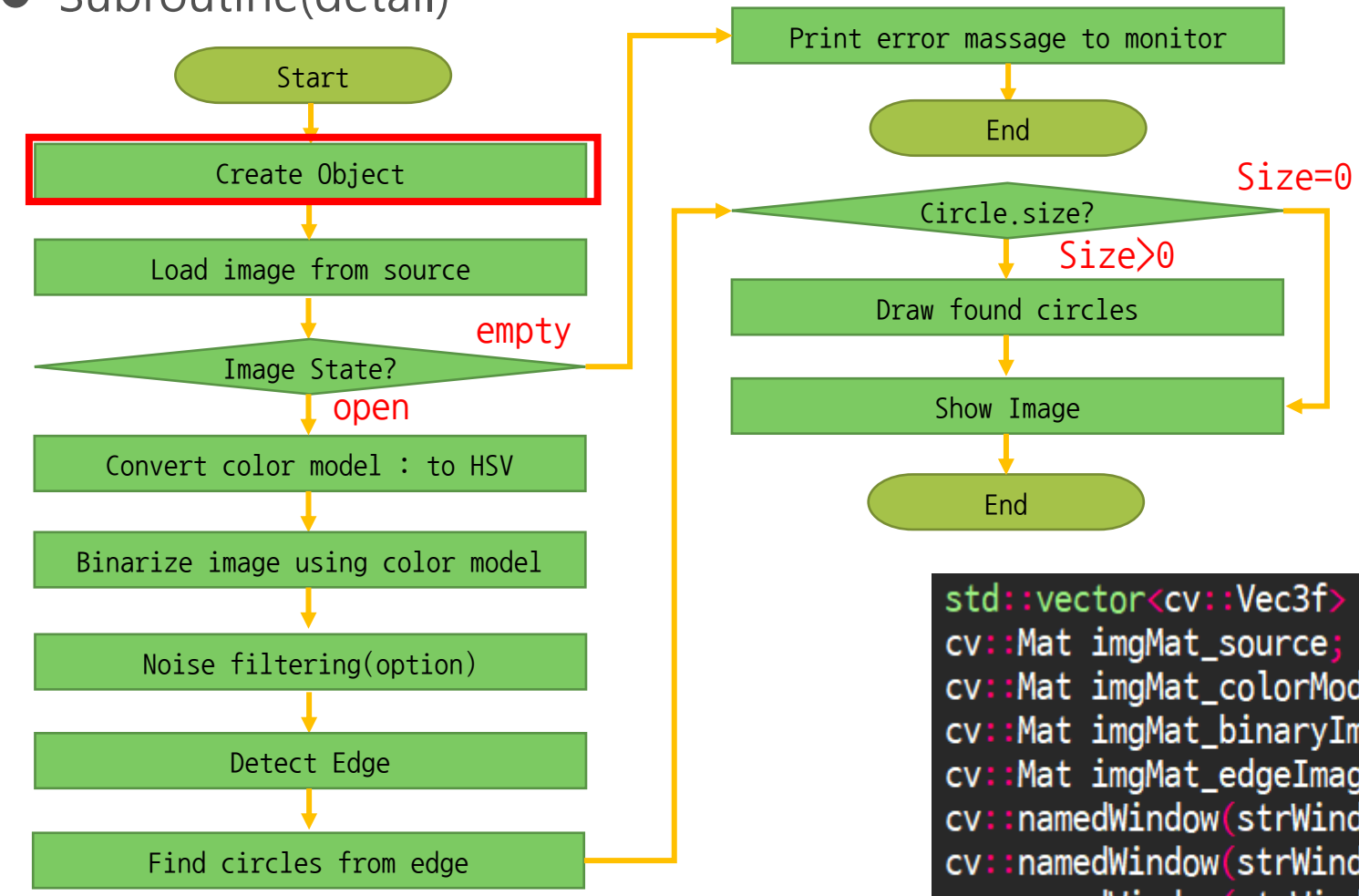
DATE : 2019-08-22 TUE

- Subroutine


Start
Create Object
Load image from source
Image State?
empty
open
Convert color model : to HSV
Binarize image using color model
Noise filtering(option)
Detect Edge
Find circles from edge
Print error massage to monitor
End
Circle.size?
Size=0
Size>0
Draw found circles
Show Image
End

Programming Material : OpenCV Edge Detecting, Circle Fitting

- Subroutine(detail)

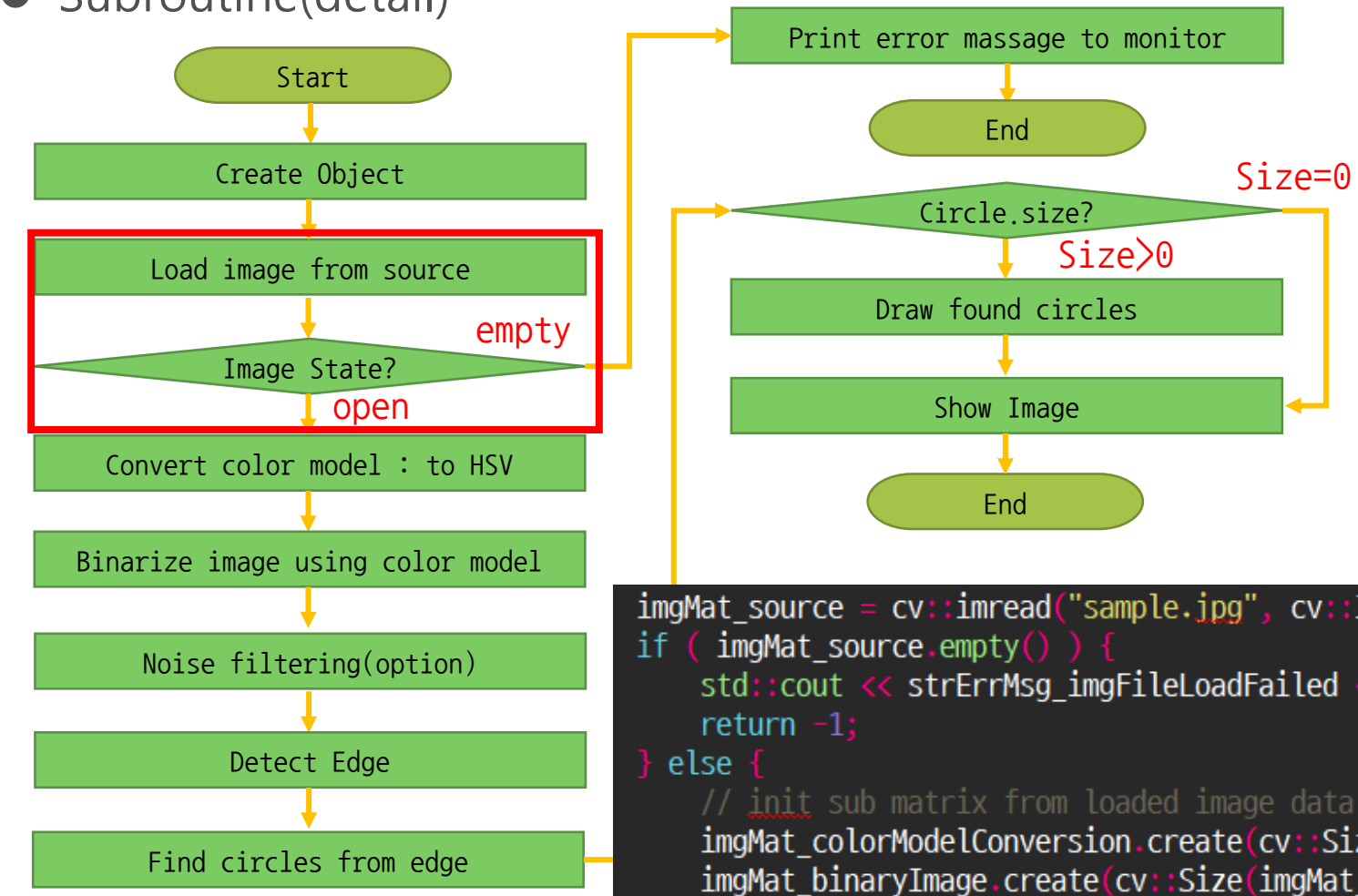



```
std::vector<cv::Vec3f> circlesData;
cv::Mat imgMat_source;
cv::Mat imgMat_colorModelConversion;
cv::Mat imgMat_binaryImage;
cv::Mat imgMat_edgeImage;
cv::namedWindow(strWindowTitle_source, cv::WINDOW_AUTOSIZE);
cv::namedWindow(strWindowTitle_binaryImage, cv::WINDOW_AUTOSIZE);
cv::namedWindow(strWindowTitle_edgeImage, cv::WINDOW_AUTOSIZE);
```

연구실 세미나 : 프로그래밍 교육

Programming Material : OpenCV Edge Detecting, Circle Fitting

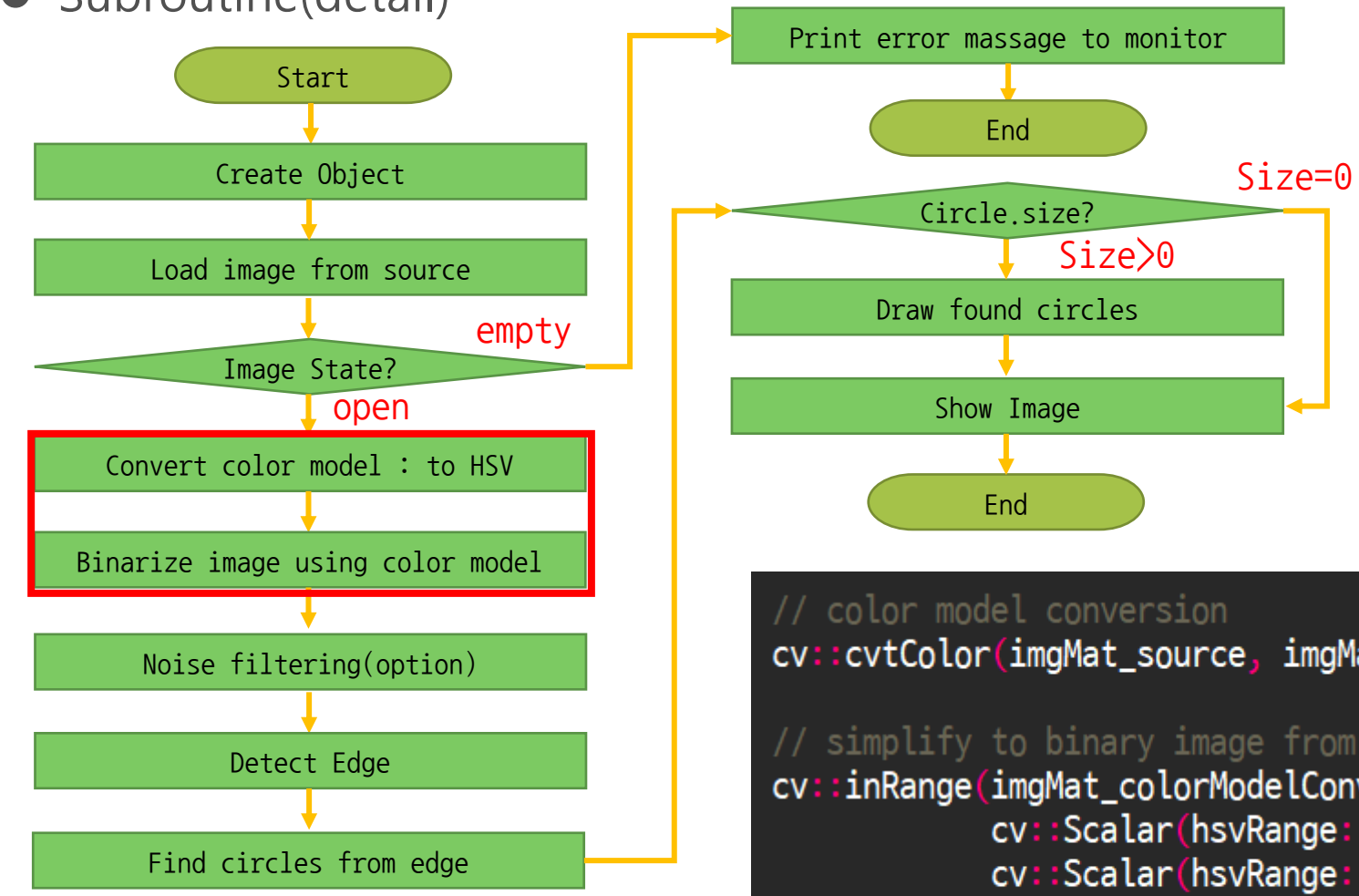
- Subroutine(detail)




```
imgMat_source = cv::imread("sample.jpg", cv::IMREAD_COLOR);
if ( imgMat_source.empty() ) {
    std::cout << strErrMsg_imgFileLoadFailed << std::endl;
    return -1;
} else {
    // init sub matrix from loaded image data size
    imgMat_colorModelConversion.create(cv::Size(imgMat_source.cols, imgMat_source.rows), CV_8UC3);
    imgMat_binaryImage.create(cv::Size(imgMat_source.cols, imgMat_source.rows), CV_8UC1);
    imgMat_edgeImage.create(cv::Size(imgMat_source.cols, imgMat_source.rows), CV_8UC1);
```

Programming Material : OpenCV Edge Detecting, Circle Fitting

- Subroutine(detail)

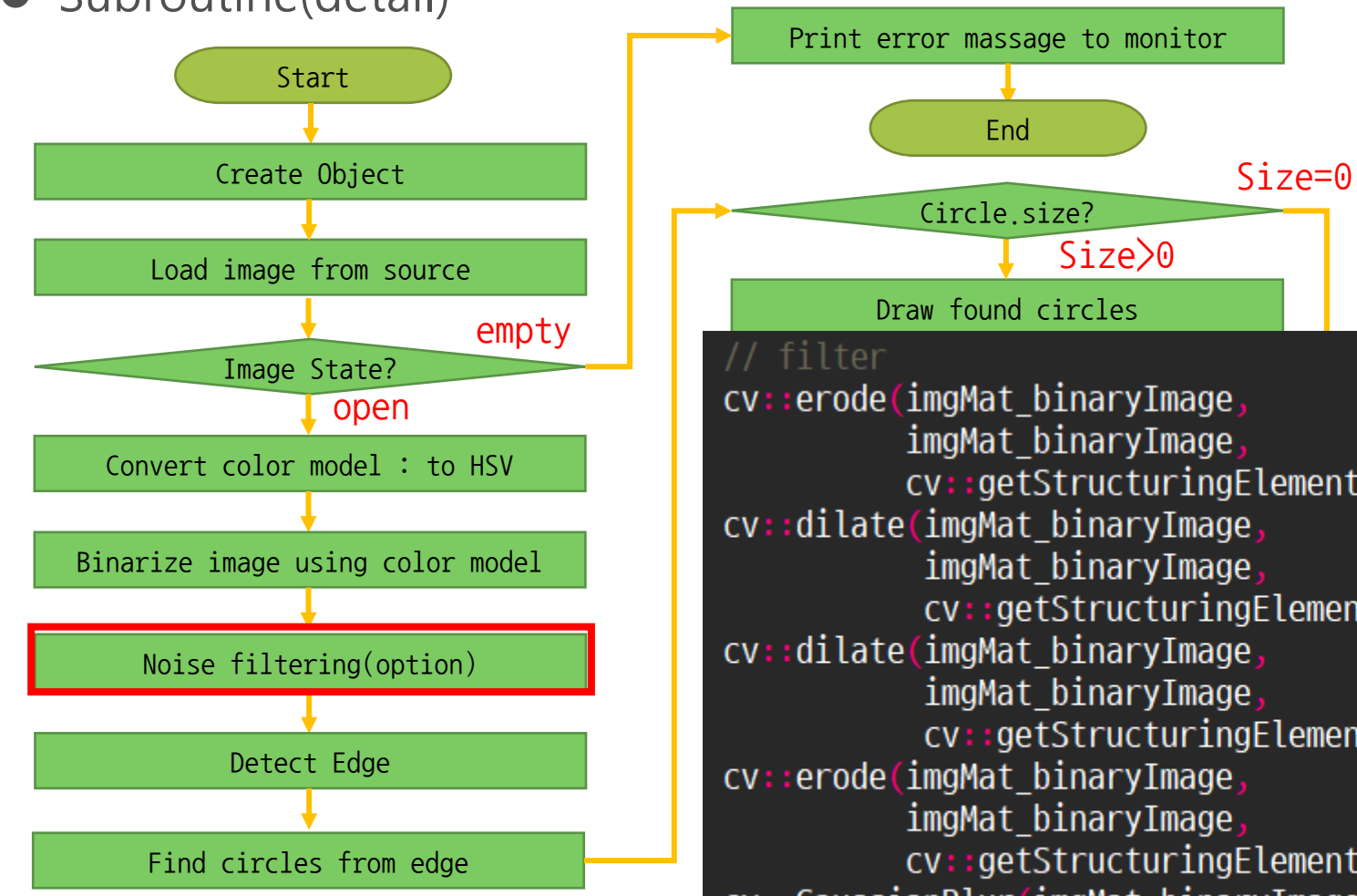



```
// color model conversion
cv::cvtColor(imgMat_source, imgMat_colorModelConversion, CV_BGR2HSV);

// simplify to binary image from hsvType image
cv::inRange(imgMat_colorModelConversion,
            cv::Scalar(hsvRange::minHue, hsvRange::minSat, hsvRange::minVal),
            cv::Scalar(hsvRange::maxHue, hsvRange::maxSat, hsvRange::maxVal),
            imgMat_binaryImage);
```

연구실 세미나 : 프로그래밍 교육

Programming Material : OpenCV Edge Detecting, Circle Fitting

- Subroutine(detail)




```
// filter
cv::erode(imgMat_binaryImage,
          imgMat_binaryImage,
          cv::getStructuringElement(cv::MORPH_ELLIPSE, cv::Size(1,1)));
cv::dilate(imgMat_binaryImage,
           imgMat_binaryImage,
           cv::getStructuringElement(cv::MORPH_ELLIPSE, cv::Size(1,1)));
cv::dilate(imgMat_binaryImage,
           imgMat_binaryImage,
           cv::getStructuringElement(cv::MORPH_ELLIPSE, cv::Size(1,1)));
cv::erode(imgMat_binaryImage,
          imgMat_binaryImage,
          cv::getStructuringElement(cv::MORPH_ELLIPSE, cv::Size(1,1)));
cv::GaussianBlur(imgMat_binaryImage, imgMat_binaryImage, cv::Size(5, 5), 2, 2);
cv::threshold(imgMat_binaryImage, imgMat_binaryImage, 127, 255, CV_THRESH_BINARY);
```

- Functions

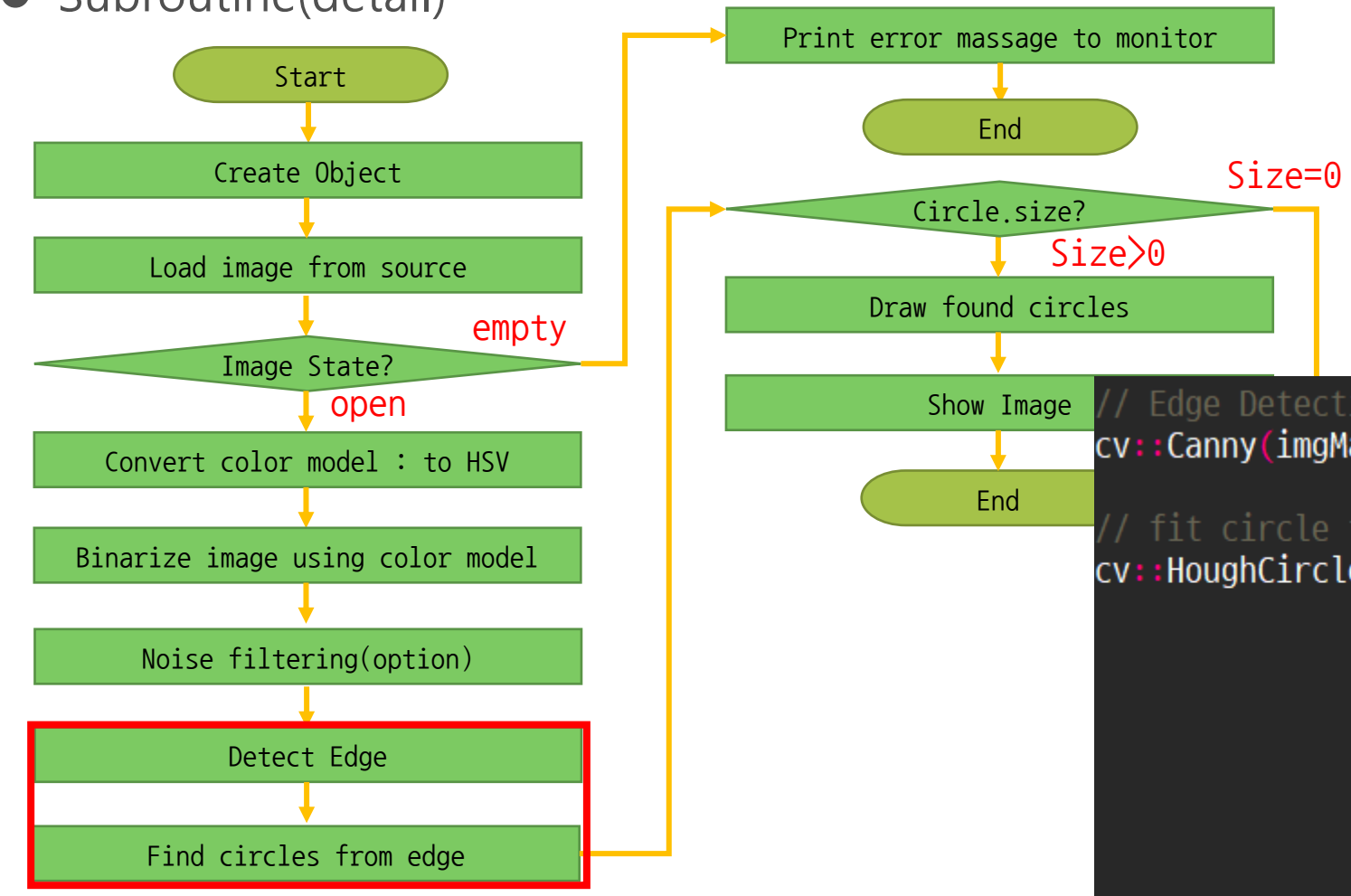
```
// Erodes an image by using a specific structuring element.
void cv::erode (
    cv::Mat &src,                                   // input image; the number of channels can be arbitrary,
                                                    // but the depth should be one of CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.
    cv::Mat &dst,                                   // output image of the same size and type as src.
    InputArray kernel,                              // structuring element used for erosion.
    cv::Point anchor = cv::Point(-1, -1);           // position of the anchor within the element.
    int iter = 1,                                   // number of times erosion is applied.
    int borderType = cv::BORDER_CONSTANT,           // pixel extrapolation method, see BorderTypes.
    const cv::Scalar &borderValue
            = morphologyDefaultBorderValue()        // border value in case of a constant border.
);

// Dilates an image by using a specific structuring element.
void dilate (
    cv::Mat &src,                                   // input image; the number of channels can be arbitrary,
                                                    // but the depth should be one of CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.
    cv::Mat &dst,                                   // output image of the same size and type as src.
    InputArray kernel,                              // structuring element used for erosion.
    cv::Point anchor = cv::Point(-1, -1);           // position of the anchor within the element.
    int iter = 1,                                   // number of times erosion is applied.
    int borderType = cv::BORDER_CONSTANT,           // pixel extrapolation method, see BorderTypes.
    const cv::Scalar &borderValue
            = morphologyDefaultBorderValue()        // border value in case of a constant border.
);
```

● Functions

```
// Blurs an image using a Gaussian filter.
void cv::GaussianBlur (
    cv::Mat &src,                       // input image
    cv::Mat &dst,                       // output image of the same size and type as src.
    cv::Size ksize,                     // Gaussian kernel size.
    double sigmaX,                      // Gaussian kernel standard deviation in X direction.
    double sigmaY = 0,                  // Gaussian kernel standard deviation in Y direction;
    int borderType = BORDER_DEFAULT     // pixel extrapolation method
);

// Applies a fixed-level threshold to each array element.
double cv::threshold(
    cv::Mat &src,               // input array (multiple-channel, 8-bit or 32-bit floating point)
    cv::Mat &dst,               // output array of the same size and type and the same number of channels as src
    double thresh,              // threshold value
    double maxval,              // maximum value to use with the THRESH_BINARY and THRESH_BINARY_INV thresholding
    types.
    int type                    // thresholding type
);                              // return val : the computed threshold value if Otsu's or Triangle methods used.
```

Programming Material : OpenCV Edge Detecting, Circle Fitting

- Subroutine(detail)




```
// Edge Detection
cv::Canny(imgMat_binaryImage, imgMat_edgeImage, 400, 800);

// fit circle from edge detected image
cv::HoughCircles(imgMat_edgeImage,
                 circlesData,
                 CV_HOUGH_GRADIENT,
                 2,
                 imgMat_edgeImage.rows/4,
                 100,
                 80,
                 75,
                 0
                 );
```

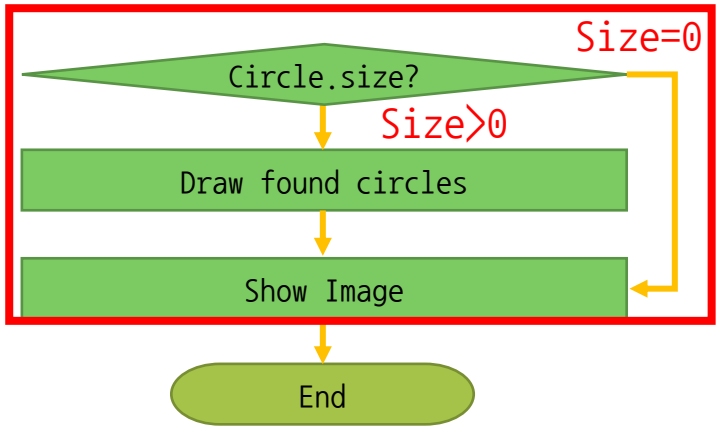
- Functions

```
// Finds edges in an image using the Canny algorithm
void cv::Canny(
    cv::Mat &src,               // (8bit) input image.
    cv::Mat &dst,               // output edge map; single channels 8-bit image,
                                // which has the same size as image.
    double threshold1,          // first threshold for the hysteresis procedure.
    double threshold2,          // second threshold for the hysteresis procedure.
    int apertureSize = 3,       // aperture size for the Sobel operator.
    bool L2gradient = false     // reference :
    https://docs.opencv.org/3.4.7/dd/d1a/group__imgproc__feature.html#ga04723e007ed888ddf11d9ba04e2232de
);
```

- Functions

```
// Finds circles in a grayscale image using the Hough transform.
// The function finds circles in a grayscale image using a modification of the Hough transform.
void cv::HoughCircles(
    cv::Mat &src,               // 8-bit, single-channel, grayscale input image.
    std::vector &circles,       // Output vector of found circles.
                                // Each vector is encoded as 3 or 4 element floating-point vector
    int method,                 // Detection method, see HoughModes.
                                // Currently, the only implemented method is HOUGH_GRADIENT

    double dp,                  // Inverse ratio of the accumulator resolution to the image resolution.

    double minDist,             // Minimum distance between the centers of the detected circles.

    double param1 = 100,        // First method-specific parameter.

    double param2 = 100,        // Second method-specific parameter.

    int minRadius = 0,          // Minimum circle radius.
    int maxRadius = 0           // Maximum circle radius. If <= 0, uses the maximum image dimension.
                                // If < 0, returns centers without finding the radius.
);
```

- Subroutine(detail)




```
if ( circlesData.size() != 0 ) {
    for ( int iter = 0; iter < circlesData.size(); iter++ ) {
        cv::circle(imgMat_source,
                   cv::Point((int)circlesData[iter][0], (int)circlesData[iter][1]),
                   (int)circlesData[iter][2],
                   cv::Scalar(0, 0, 255),
                   2);
    }
} else {
    cv::putText(imgMat_source,
                strMsg_noDetection,
                cv::Point(15, 15),
                1,
                1.5,
                cv::Scalar(0, 0, 255),
                1);
}
cv::imshow(strWindowTitle_source, imgMat_source);
cv::imshow(strWindowTitle_binaryImage, imgMat_binaryImage);
cv::imshow(strWindowTitle_edgeImage, imgMat_edgeImage);
cv::waitKey();
```

연구실 세미나 : 프로그래밍 교육

Programming Material : OpenCV Edge Detecting, Circle Fitting

- Subroutine(detail)