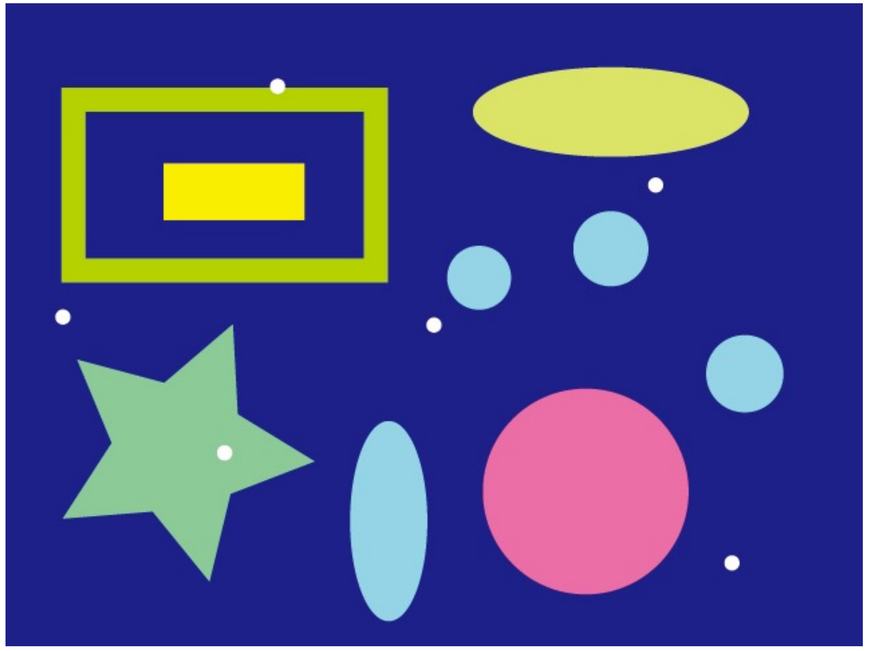
二値画像処理 (2)

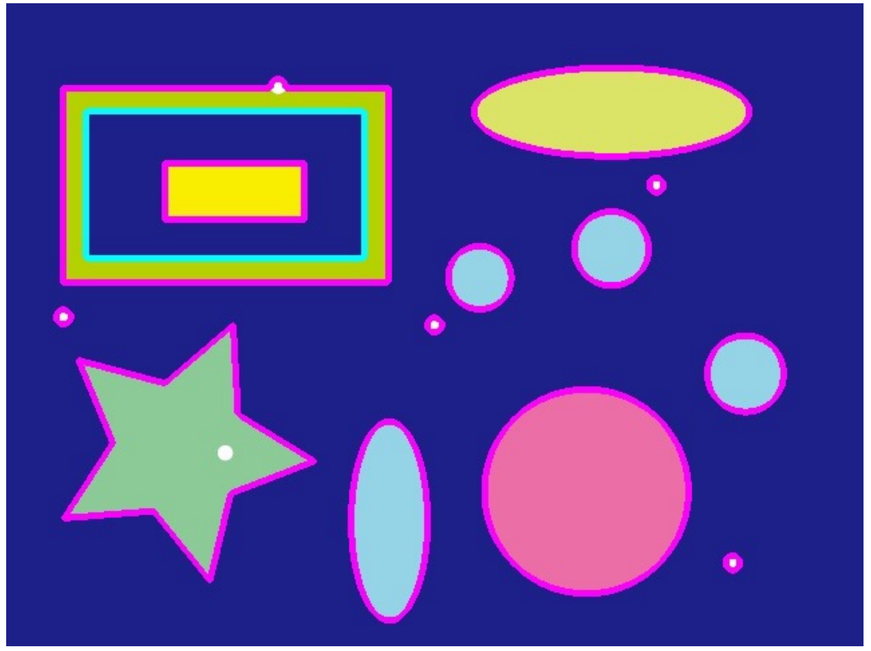
末永，森本，塚田，澤野

演習

- teamsからcontour4student.cppとsample.jpgの取得
- プロジェクト名: 08_01_contour4student
- カラー画像を入力し，輪郭を抽出



入力画像



輪郭画像例 (ピンクの輪郭)

アルゴリズム

1. 画像の宣言
   (入力画像，グレースケール画像，二値画像，
   一時的な画像，出力画像)
2. 輪郭の座標リストの宣言
3. 画像の入力 (**カラー**で入力)
4. 入力画像を結果画像にコピー

New!

5. グレースケール化
6. 二値化 (固定閾値で実装．閾値: 100)
7. 輪郭追跡
8. 輪郭の描画

New!

9. 表示

まずは二値化まで
実装してください

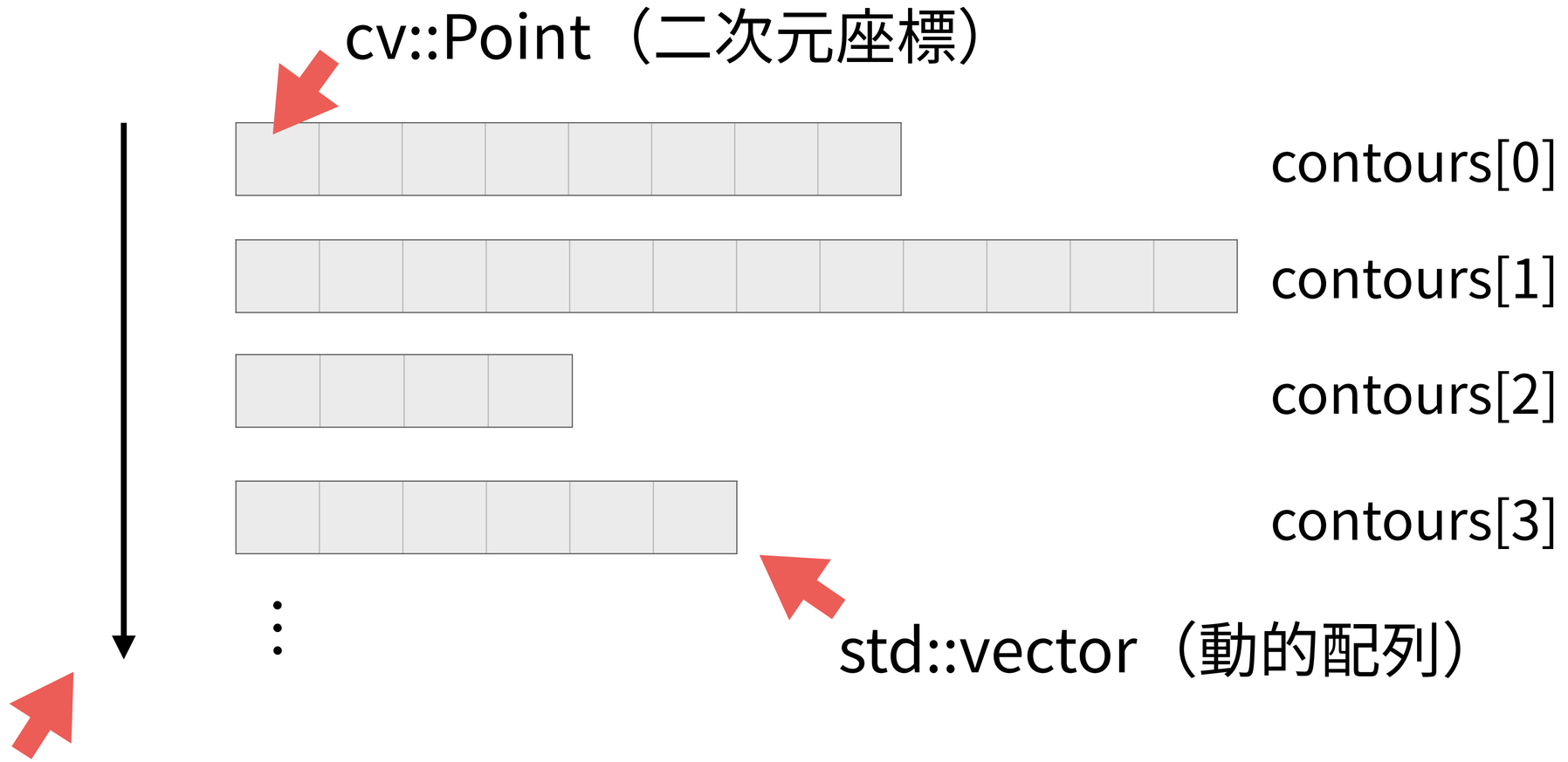
ノート

輪郭の座標リストの宣言

```
std::vector< std::vector< cv::Point > > contours;
```

- std::vector: 動的配列
  - 配列の大きさが固定されない（可変長）
- cv::Point: 座標
  - 二次元の座標
- 画素位置の 並び（＝輪郭）をリスト化
  (cv::Point) の (std::vector) の (std::vector)

contoursの構造



二値化まで

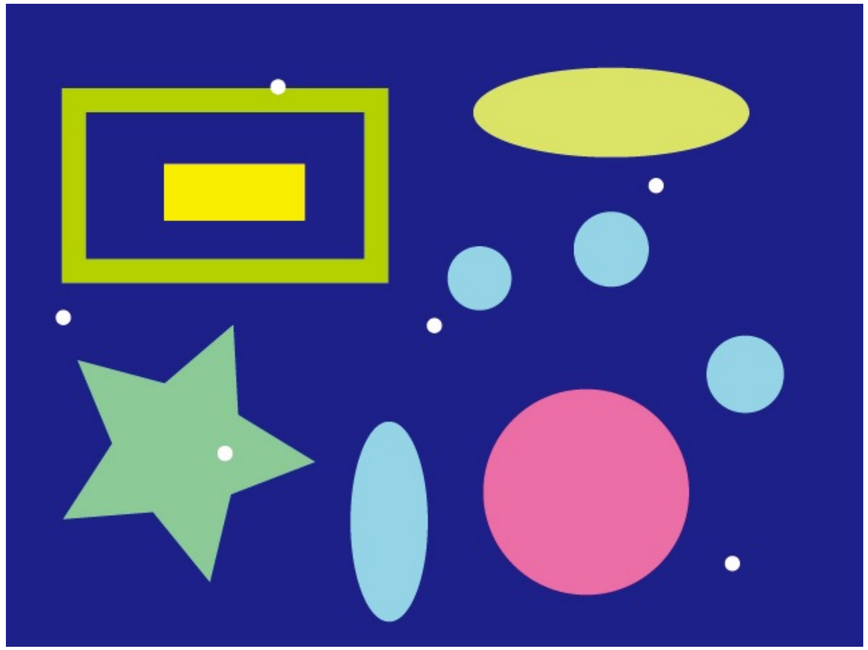
```
//4. 入力画像を結果画像にコピー（New!)
dst_img = src_img.clone();

//5. グレースケール化
cv::cvtColor(src_img, gray_img, cv::COLOR_BGR2GRAY);

//6. 二値化（固定閾値で実装. 閾値: 100)
cv::threshold(gray_img, bin_img, BIN_TH, 255,
cv::THRESH_BINARY);
//二値画像コピー（New!)
tmp_img = bin_img.clone();
```

- BIN_TH：二値化の閾値 (100)
- .clone()：画像のコピーを作成

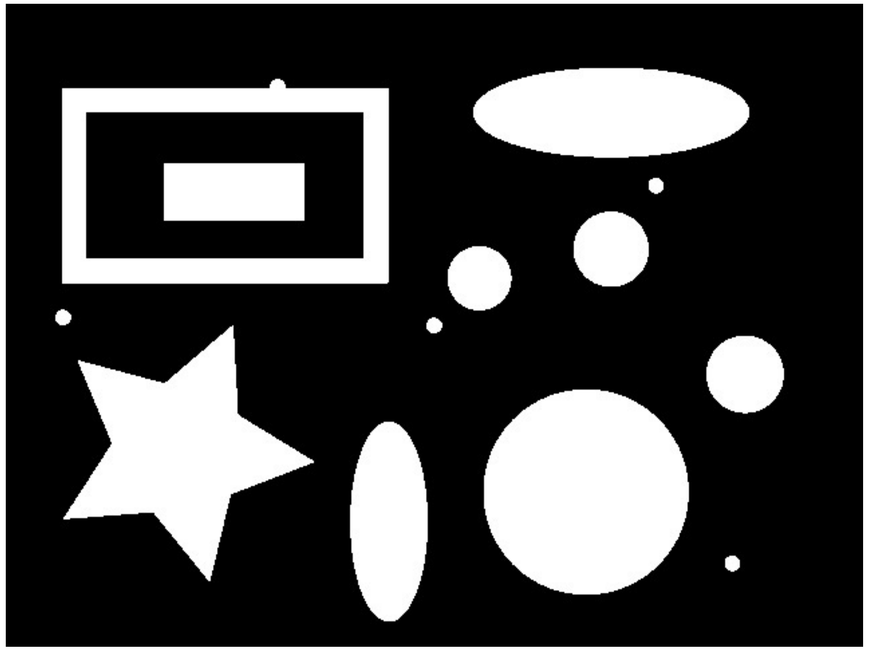
輪郭追跡で画像が加工されてしまうため,

予めbin_imgをtmp_imgにコピーしておく

二値画像



入力画像



二値画像

ノート

輪郭追跡関数の説明

- 関数紹介

```
//7. 輪郭追跡(New!)
 cv::findContours(tmp_img, contours, cv::RETR_LIST, cv::CHAIN_APPROX_NONE);
```

```
cv::findContours(二値画像, 輪郭, 追跡モード, 輪郭近似手法);
```

- cv::RETR_LIST: すべての輪郭追跡、リスト出力
- cv::CHAIN_APPROX_NONE: 8近傍、近似なし

各輪郭へのアクセス方法

- for文で各輪郭にアクセスしていく

```
//8. 輪郭の描画(New!)
for (int i=0; i<contours.size(); i++) {


}
```

contours.size(): 動的配列contoursのサイズ (輪郭の数)

ノート

輪郭の描画

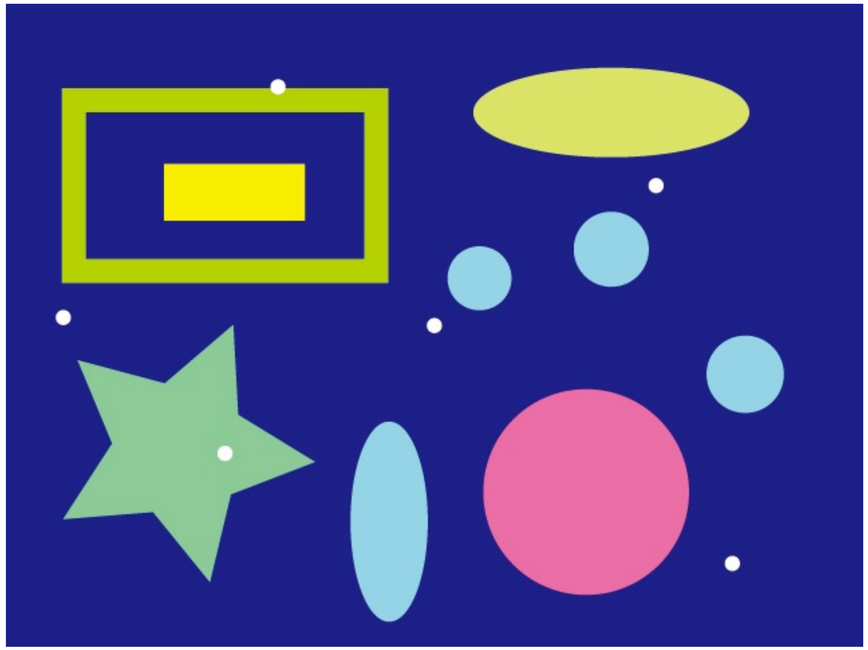
```
//輪郭の描画
cv::drawContours(dst_img, contours, i, CV_RGB(255, 0, 255), 3);
```

※B=R=255でマゼンタとしている

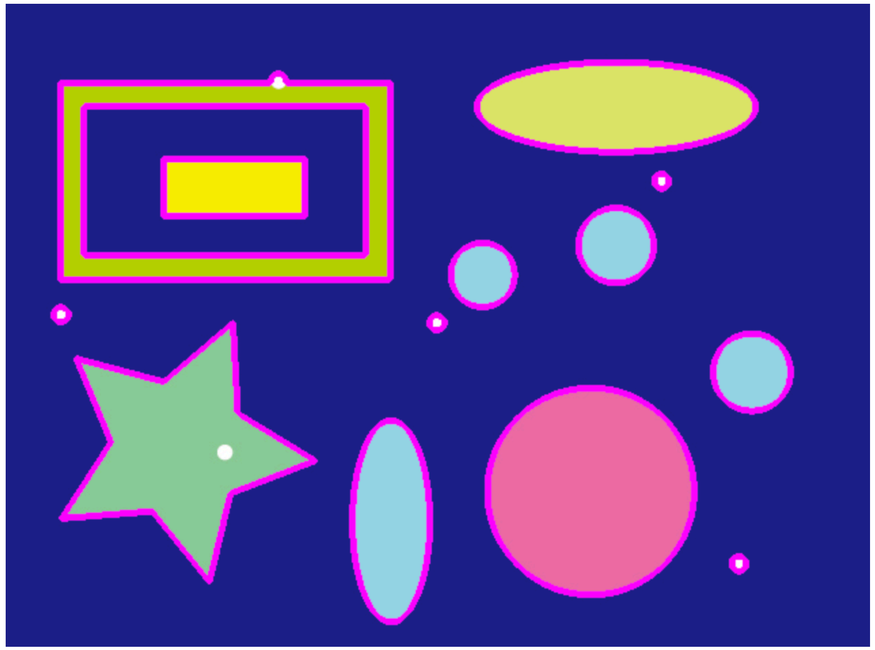
```
cv::drawContours(出力画像, 輪郭情報, 輪郭番号,
                                    輪郭の色, 描画用の線幅);
```

- for文の中に入れる
  - 輪郭番号を指定する（この場合 i ）
- 線幅を負にすると塗り潰しができる

描画された輪郭



入力画像



輪郭画像
（外輪郭・内輪郭全て）

ノート

領域特徴量（p.118）

- 輪郭追跡で閉輪郭が得られる

  ➡領域が定まる（領域抽出）

  ➡領域を表す特徴量を計算できる

- 領域特徴量

  - 重心
  - 面積
  - 周囲長（輪郭の長さ）
  - 円形度（どれだけ円に近いか）
  - 外接長方形（領域に接する最小の長方形）(バウンディングボックス)

ノート

周囲長と面積

- i番目の輪郭に対する周囲長と面積を求める

```
double L, S;


//周囲長（輪郭の長さ）
L = cv::arcLength(contours[i], true);
//面積
S = cv::contourArea(contours[i]);
```

円形度

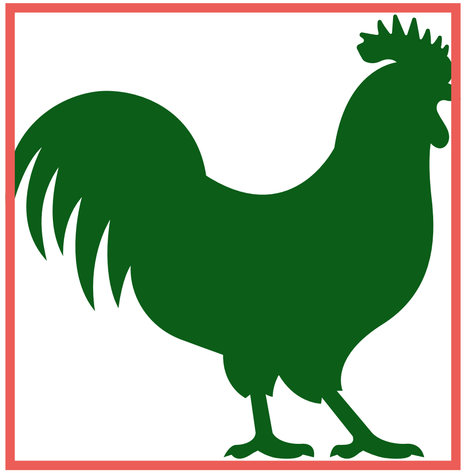
- ある図形がどれだけ円に近いかを算出
- ある図形の面積をＳ，周長をＬとするときの円形度Ｒ

$$R = \frac{4\pi S}{L^2}$$

- 理想的な円のとき，R=1
- プログラムではM_PIでπを表現される

外接長方形 (バウンディングボックス)

- 輪郭の外接長方形

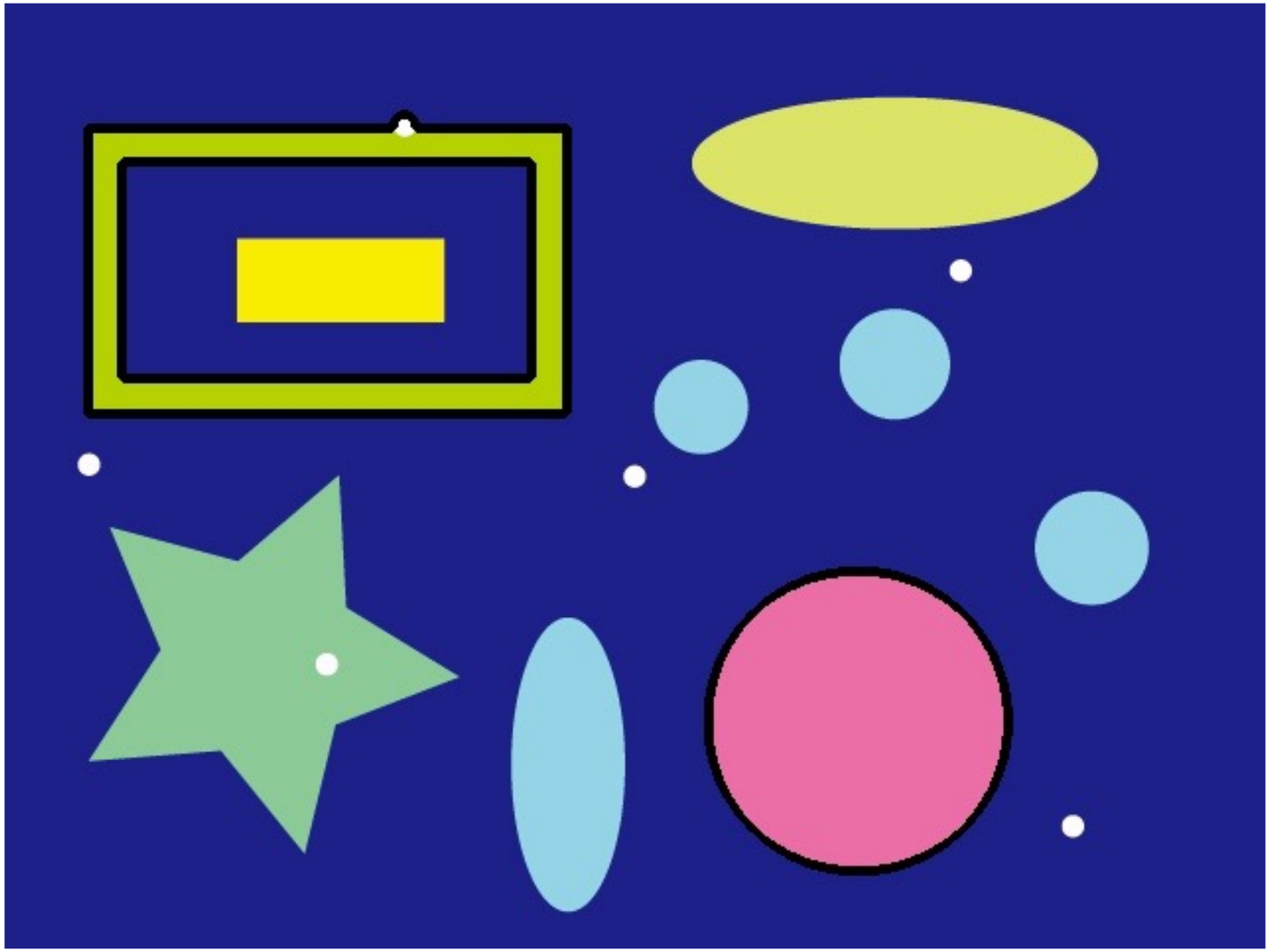


```
for (int i=0; i<contours.size(); i++) {
    //バウンディングボックスの計算
    cv::Rect bounding_box = cv::boundingRect(contours[i]);
    std::cerr << "width=" << bounding_box.width << ", height=" << bounding_box.height << std::endl;
    cv::rectangle(dst_img, bounding_box, CV_RGB(255, 255, 0), 5); //描画 (確認用)
}
```

課題1

- sample.jpgに描画された図形のうち，
  面積が18,000以上の領域を抽出し，
  輪郭を**黒色**で描画せよ
- 提出ファイル名:
  - ソースファイル: 08_01_学籍番号.cpp
  - 結果画像: 08_01_result_学籍番号.jpg (カラー画像, tiffでも良い)

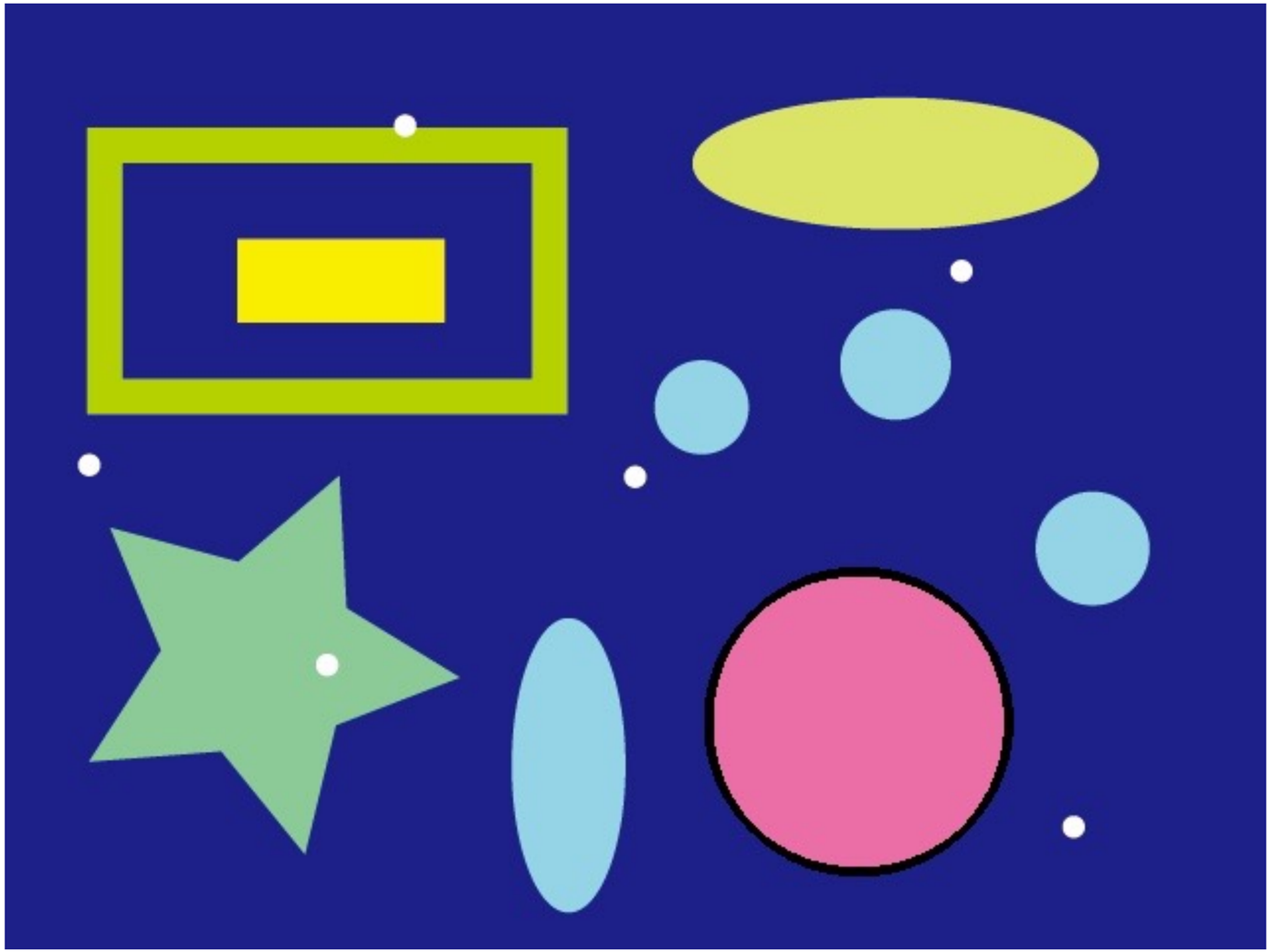
課題1の回答例



課題2

- sample.jpgに描画されたいる図形を，
  円形度を利用して，
  面積18,000以上の円の輪郭を黒線で描画せよ
- ヒント: 円形度は0.89以上とする．
- 提出ファイル名:
  - ソース: 08_02_学籍番号.cpp
  - 結果画像: 08_02_result_学籍番号.jpg (カラー画像, tiffでも良い)

課題2の結果画像



課題3 (加点対象，提出自由)

- figure.jpgで描かれている四つの図形を
  それぞれの色で塗りつぶすプログラムを書け
  - 正方形: (255, 0, 0)
  - 長方形: (0, 0, 255)
  - 円: (255 , 0, 255)
  - 星: (128, 0, 128)
- 提出ファイル名:
  - プログラム: 08_03_学籍番号.jpg
  - 結果画像: 08_03_result_学籍番号.jpg

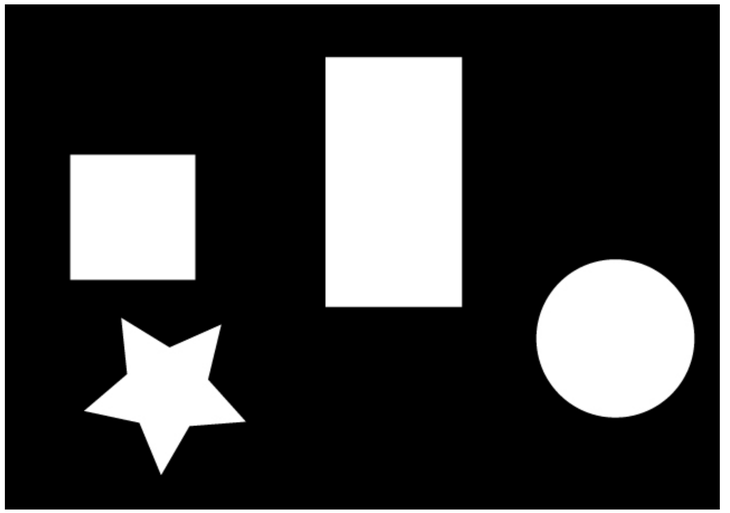
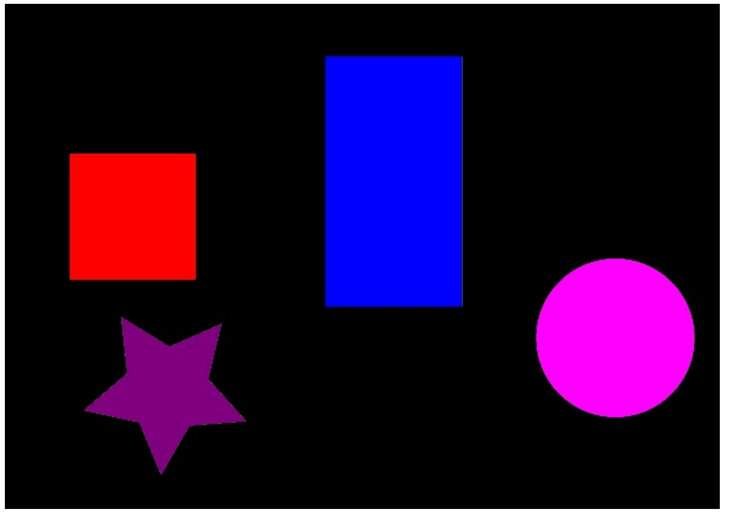


figure.jpg



結果画像

課題3のヒント

- 傾きのない四角形の場合，
  輪郭とバウンディングボックスの面積はほぼ同じとなる．
  - 課題3の場合，400画素未満の誤差がありました
- 戦略としては以下がおすすめです．

1. 四角形の判定
   1. 正方形の判定: true → 描画，continue;
   2. 長方形の判定: true → 描画，continue;
2. 円の判定: true → continue
3. そのほか