学生実験４：画像処理（DIP）
— 2 次元コードの生成と認識 —

担当：飯山将晃 （TA:藤村友貴）

2016 年度版 (Ver 1.0)

1　はじめに

本演習では，計算機によって画像を処理する基本技術を修得することを目的として，画像を生成したり，生成した画像やカメラで取得した画像を変換・認識したりするプログラムを作成し，その挙動を確認する．具体的には，社会に広く浸透した 2 次元コードを題材とし，その生成処理，認識処理を実現するプログラムの作成を通して，画像を扱う上で必要となるデータ構造，画像の可視化，変換，認識に必要な基礎技術を習得する．

2　QR コード

2 次元コードとして代表的な QR コード (図 1) は，多くのカメラ付き携帯電話を使って簡単に読み取ることができるようになっており，ポスター，製品パッケージ，名刺など，さまざまなところに印刷され，社会に広く浸透している．

図 1　QR コードの例

2.1　2 次元コード

2 次元コードとは，シンボルキャラクタ（正方形や長方形，点など）を縦横に配置したもので数値や文字などの情報をコード化したものである．1 次元コードに相当するバーコードも広く普及しているが，埋め込める情報量が小さいため，埋め込める情報を大きくすることを目的として

1980 年代半ばからいくつかの方式が提案されている [1]．2 次元コードは，従来のバーコードを縦に並べたスタック型と，小さな正方形を縦横に並べたマトリックス型に分けられる．スタック型では PDF417, Code49，マトリックス型では QR コード, DataMatrix, VeriCode などが提案されている．

2.2 QR コードの特徴

一般的に 2 次元コードはバーコードに比べて扱うデータが非常に多いため，処理が複雑で時間がかかる．そんな中，高速読み取りを重視したマトリックス型 2 次元コードとして，QR(Quick Response) コードが 1994 年にデンソーによって開発された．QR コードは以下の特徴を持ち，そのためのさまざまな工夫が施されている．

- 360 ° 全方向，高速に読取可能
- ひずみに強い
- 汚れや汚損に強い
- 白黒バランスを整えることにより，読取が安定化される

2.3 QR コードの主な仕様 [2]

数値や文字などの情報を QR コードで表現したものをシンボルと呼ぶ．シンボルを構成する主な要素を図 2 に示す．

■モジュール シンボルを構成する単位セル（正方形）．1 モジュールが 1 ビットに相当する．明モジュール（□）と暗モジュール（■）があり，明暗によって 1 ビットを表現する．

■サイズ QR コードは，色々な用途に使えるように 40 種類のサイズが用意されており，データ量や読み取り方法に応じて自由に選べるようになっている．シンボルサイズは 1 型：21 × 21 モジュール，2 型：25 × 25 モジュール，3 型：29 × 29 モジュール，…と 1 辺の大きさが 4 モジュールずつ増えていき，最大で 40 型：177 × 177 モジュールまでとされている．

■位置検出パターン 位置検出パターンは，図 2 に示すシンボルの左上，右上，左下の 3 つの隅に配置される 3 個のパターンから構成される，画像全体の中からシンボル位置を検出するためのパターンである．シンボルの 3 つの頂点に配置することにより，シンボルの位置だけでなく，大きさ，傾きも検出することができる．

図 3 に示すように，各位置検出パターンは (A) 暗の 3 × 3 モジュール，(B) 明の 5 × 5 モジュール，(C) 暗の 7 × 7 モジュールの 3 つの同心正方形が重なった形状である．各パターンのモジュール幅の比率は 1:1:3:1:1 とする．これに類似するパターンがシンボル内の他の位置に出現することを抑え，視野内でシンボルの認識を容易にすることができるように埋め込むデータを符号化する．シンボル内にある 3 つの位置検出パターンを識別することによって，視野内でのシンボルの位置及

図 2　QR コードのシンボルの構成

び方向を明確に認識できる．また，位置検出パターンの周りには分離パターンとして 1 モジュール幅で明モジュールが配置され，後述の符号化領域と分離される．

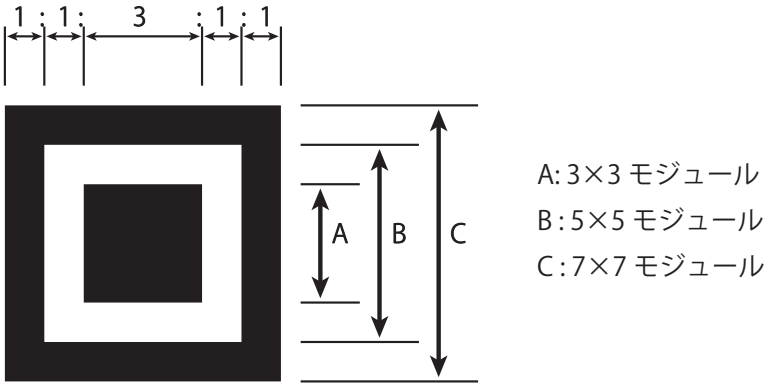

図 3　位置検出パターン

■クワイエットゾーン　シンボルの四辺の周囲を囲む，何も表示されない 4 モジュール幅の領域である．明モジュールと同じ値を持つ．上で定義されたサイズには，クワイエットゾーンは含まれ

ない．

■タイミングパターン　タイミングパターンは，1 モジュール幅で明モジュールと暗モジュールとが交互になっている．水平タイミングパターンは，シンボルの 7 行目，左上の位置検出パターンの分離パターンと右上のそれとの間にわたっており，暗モジュールで始まり暗モジュールで終わる．垂直タイミングパターンも同様に，シンボルの 7 列目，左上の位置検出パターンの分離パターンと左下のそれとの間にわたっている．これらによってシンボルのサイズを決めることができ，モジュール座標と画像の座標の対応が得られる．

■位置合せパターン　位置合せパターンは暗の 1 × 1 モジュール，明の 3 × 3 モジュール，暗 5 × 5 のモジュールの 3 つの同心正方形が重なった形状である．位置合せパターンの数はシンボルの型番によって決まる．2 型〜6 型は 1 個，7 型〜13 型は 6 個，14 型〜20 型は 13 個，21 型〜27 型は 22 個，28 型，35 型〜40 型は 46 個となる．シンボルの左上隅から右下隅への対角線の両側で対称に配置される．できるだけ均等な格子状に配置されるが，間隔が均等にできない場合は，タイミングパターンと最初の位置合せパターンとの間で調整される．画像にある程度のひずみが生じた場合でも，位置合せパターンを用いることでひずみを補正することができる．

■符号化領域　埋め込むデータをコード化したパターン，及びそれらに対する誤り訂正コードを含む領域である．QR コードでは，データを損失することなく，シンボルが損傷に耐えることができるよう，データのコードに付加する一連の誤り訂正コード語を生成するリードソロモン誤り訂正が採用されている．また，符号化領域内で位置検出パターンが現れることがないよう，また明暗をバランス良く配置できるよう符号化領域で特殊なマスク処理を行っている．この工夫が，読取時間を高速化し，また読取時に画像中で明モジュールと暗モジュールを区別する 2 値化処理の安定化に繋がっている．

3　演習の流れ

QR コードはデンソーウェーブによって特許が取得されているが，一般的に普及することを目的として開発されたため，下記のように表明されている [3].

> 2 次元コードが広く普及するためには、まずコードの仕様が明確に定義・公開される必要があります。さらに、ユーザが自由に使えるようなコードである必要があります。
> バーコード＝ 1 次元コードがこれほどまでに普及した背景の一つには、この仕様公開があります。今日、仕様の非公開、あるいは厳しい特許保護をもつバーコードはほとんどありません。
> QR コードは国家規格や国際規格で規格化されており、誰でも仕様を入手することができ、かつ規格化された QR コードについては、デンソーウェーブが保有する特許（特許第 2938338 号）の権利行使を行わないことを宣言しています。

そのため，例えば [2] を通して，詳細な仕様を誰でも入手でき，また復号アルゴリズムも記載されている．

前節でも一部紹介した通り，QR コードは頑健に動作するようさまざまな工夫が施されているすばらしいものであるが，本演習のような限られた時間の中では，全ての仕様に準拠したシステムを構築するのは容易ではない．そこで，本演習では必須機能のみを仕様から切り出し，この生成と認識を実現するシステムを実装する．

3.1 本演習で扱う 2 次元コードの仕様

■サイズ 本演習で扱う 2 次元コードシンボルは，2 型：25 × 25 モジュールとする．4 モジュール分の幅のクワイエットゾーンも加えると，33 × 33 モジュールのサイズとなる．ただし，演習の発展課題としてシンボルサイズの拡大も考えるため，容易に拡張できるよう実装すること．

■位置検出パターン 位置検出パターンについては QR コードに準拠する．図 3 に示すように，各パターンは (A) 暗の 3 × 3 モジュール，(B) 明の 5 × 5 モジュール，(C) 暗の 7 × 7 モジュールの 3 つの同心正方形が重なった形状である．さらに 1 モジュール幅で明モジュールが配置される分離パターンも (D) 明の 9 × 9 モジュールとして位置検出パターンに含めて扱うものとする．

■符号化領域 埋め込む文字のビット列を埋め込む．埋め込む方法については任意とする．ただし当然のことではあるが，各人でシンボル生成時の符号化とシンボル認識時の復号において処理方法が一貫していなければならない．

以上の仕様を図 4 に示す．タイミングパターンや位置合せパターンの配置，誤り訂正符号やマスク処理については，本演習では必須とはしない．ただし発展課題として，本節で示した仕様を各自で拡張し，より高速で安定した認識を実現するための機能を導入することは妨げない．

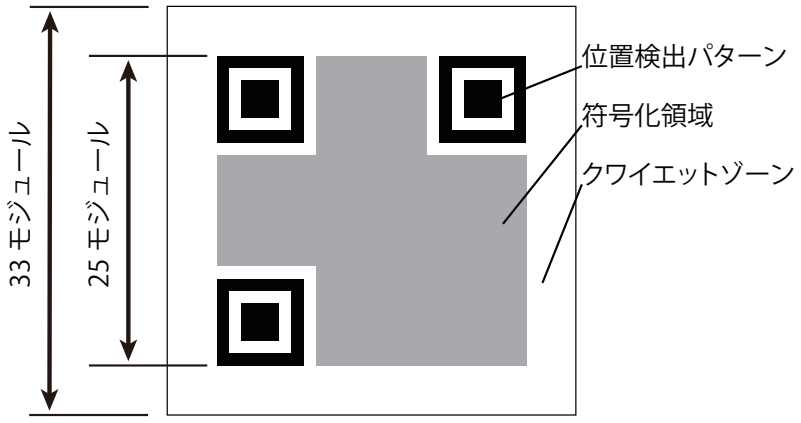

図 4 本演習で扱う 2 次元コードシンボル

3.2　2 次元コードシンボル生成の流れ (図 5)

(1) 生成する 2 次元コードシンボルを格納可能な画像のバッファを用意する
(2) 位置検出パターンやクワイエットゾーンなど，機能パターンをバッファにセットする
(3) 埋め込む文字列を符号化し，ビット列を得る
(4) ビット列を符号化領域に埋め込む

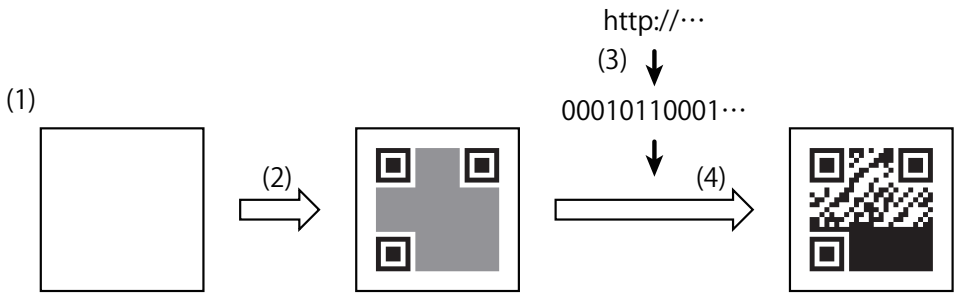

図 5　生成の流れ

3.3　2 次元コードシンボル認識の流れ (図 6)

(1) シンボルを撮影した画像を取得する（本演習では撮影を模した画像変換を実装し，生成した 2 次元コードシンボルに適用することで撮影画像をシミュレートする）
(2) 取得した画像を 2 値化し，明と暗に分類する
(3) 2 値画像から位置検出パターンを検出する
(4) 検出結果を基に傾きの補正やサイズの判定を行い，シンボルにおける元々の明/暗モジュールの配列 (2 次元コードシンボルとして生成した画像と同じもの：原シンボルと呼ぶ) を獲得する
(5) シンボルの符号化領域から埋め込まれたビット列を抽出する
(6) ビット列から文字列を復号する

4　2 次元コードシンボルの生成

4.1　計算機での画像の扱い

4.1.1　画像のデータ構造

ディジタル画像は画素の 2 次元配列として表すことができる．各画素は，その位置がどのような色や明るさで見えるかを値として保持する．計算機上ではこれを光の三原色である R（赤）G（緑）B（青）の成分を適当な明るさで合成することによって表す．本演習では，この RGB それぞれを 0

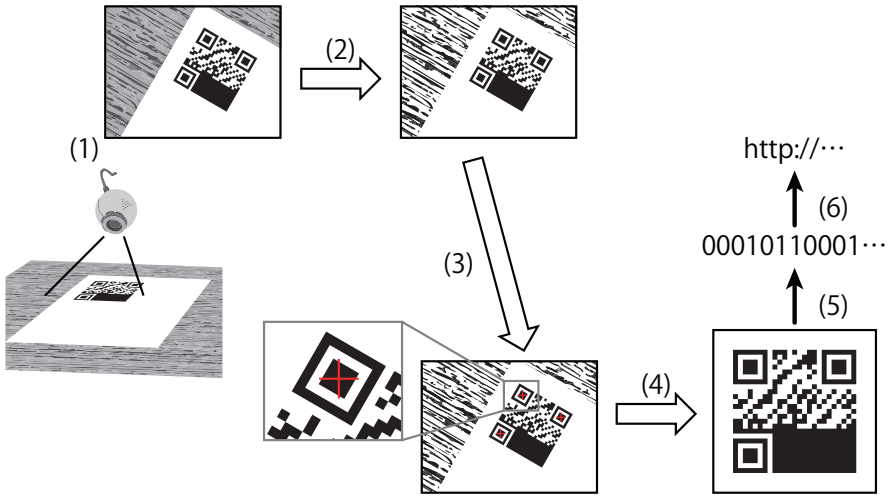

図 6　認識の流れ

から 255 の 256 段階の輝度で表す方法を採用する．これにより，2^{24} 通りの色（いわゆる 1670 万色）を表現できる．例えば，白は R=255, G=255, B=255 とし，黒は R=0, G=0, B=0 となる．

計算機で扱う画像のデータ構造にはさまざまなものがあるが，本演習では最も簡単なデータ構造を利用する．図 7 で示すように，幅が W，高さが H の画像に対し，画像の左上を原点とし，幅方向の右向きを X 軸，高さ方向の下向きを Y 軸とする座標系を考え，各画素を $I(x, y)(x = 0, \cdots, W-1, y = 0, \cdots, H-1)$ と表現する．各画素の RGB の値それぞれに 8bits = 1Byte を割当て，符号なし 8bit 整数 (unsigned char:$[0, 255]$) で表現する．画像の横 1 列 (走査線) で $W \times 3$ の長さの配列となる．これを全ての列について結合した 1 つの大きな配列によって画像全体を表す (ラスタ走査)．画像全体では $W \times H \times 3$ の長さを持つ unsigned char の 1 次元配列として扱う．配列の中では，$I(0,0)$ の R,G,B，$I(1,0)$ の R,G,B，…，$I(W-1,0)$ の R,G,B，$I(0,1)$ の R,G,B，$I(1,1)$ の R,G,B，…，$I(W-1,1)$ の R,G,B，……，$I(W-1,H-1)$ の R,G,B と割り当てられる．

4.1.2　画像の出力

このような画像データを表現する最も簡単なファイル形式に ppm 形式がある．この形式でファイルに出力すれば，画像表示ツールを用いて表示することができる．ファイルの拡張子は.ppm とするのが一般的である．ppm 形式は，ファイルの先頭部分に形式情報（P3：アスキー形式／ P6：バイナリ形式），画像の幅 W，高さ H，RGB の値の最大値を空白で区切って記述する．なお，#で始まる行は，その行末までをコメントとして扱い，無視する．続いて前述の画素の並びで，各画素の RGB の値を記述する．アスキー形式ならば数値を空白区切りで出力し，バイナリ形式であればバイト列を直接出力する．サンプルを図 8 に示す．ppm 形式についての詳細は man ページを参照されたい．

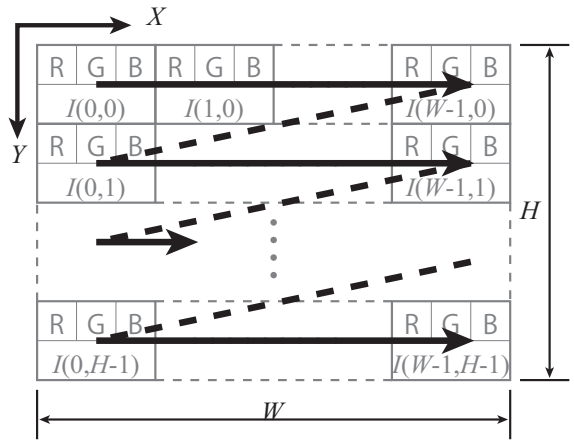

図 7　画像に対する画素配列の取り方

```
P3
# feep.ppm
4 4
255
  0   0   0     0   0   0     0   0   0   255   0 255
  0   0   0     0 255 127     0   0   0     0   0   0
  0   0   0     0   0   0     0 255 127     0   0   0
255   0 255     0   0   0     0   0   0     0   0   0
```

図 8　ppm ファイルのサンプル

4.2　シンボルの作成

4.2.1　バッファの用意

本演習で作成する 2 次元コードシンボルは，位置検出パターンとクワイエットゾーン，符号化領域も含めて 1 辺が $W_{sym} = 33$ モジュールの正方形となる．1 モジュールを 1 画素として扱うと，幅，高さ共に W_{sym} である画像を作成することになる．このような画像に対し，明モジュールを白（R=255, G=255, B=255），暗モジュールを黒（R=0, G=0, B=0）として各画素の値をセットしていく．

4.2.2　パターンのセット

用意したバッファに対し，位置検出パターンとクワイエットゾーンをセットする．

クワイエットゾーンについては，シンボルの周囲で 4 モジュール分の幅の持つ領域，つまり $x < 4, x \geq W_{sym} - 4, y < 4, y \geq W_{sym} - 4$ なる領域が明モジュールとなる．

位置検出パターンは左上，右上，左下の 3 つの隅に (A) 暗 3 × 3，(B) 明 5 × 5，(C) 暗 7 ×

7，(D) 明 9×9 の同心正方形を重ねて描画する．この正方形の中心は，それぞれのパターンで $(7,7),(W_{sym}-8,7),(7,W_{sym}-8)$ となる．

4.2.3 符号化領域への埋め込み

計算機が扱う文字には，それぞれビット列（数値）が割り当てられている．半角英数文字であれば 1 文字あたり 8bits=1Byte が割り当てられている．例えば ASCII コードでは，‘a’ には 01100001 が割り当てられている．シンボルに ‘a’ を埋め込む際，符号化領域のうちの 8 つのモジュールを使い，各モジュールを暗，明，明，暗，暗，暗，暗，明とすることになる．漢字など，多言語対応のために 2Byte 以上を使用して表現される文字もある．文字とビット列の対応付けには，例えば日本語であれば Shift-JIS や EUC，多言語に対応した Unicode でも UTF-8 などがあるが，どのような対応付けを行っていたとしても，「ビット列に変換し，その長さ分のモジュール列に，ビットに応じた明暗を割り当てる」，という処理に変わりはない．

埋め込む文字列の各文字のビット列から各ビットを抽出するには，例えばビット演算を用いる．‘a’ と 1 との&（ビット積）を取る，つまり 01100001&00000001 を計算すると，最下位ビットが 0 か 1 かを知ることができる．同様に，‘a’ と 2 や 4，つまり 01100001&00000010, 01100001&00000100 を計算すれば，それぞれ下から 2 番目，3 番目のビットを知ることができる．00000001, 00000010, 00000100 などの数値の生成にはビットシフト演算 (<<, >>) を活用するのがよい．

こうして埋め込む文字全てをビット列に変換し，それに対応する明暗パターンを符号化領域に埋め込む．この際，符号化領域内でのビットの順序については各人で任意とする．

5 2 次元コードシンボルの認識

5.1 撮像による画像変換

シンボルが印刷された平面をカメラで撮影し，取得した画像からシンボルに埋め込まれた文字列を抽出する．カメラで取得される画像は，元のシンボルに対して変換を施したものと考えることができる．この変換は幾何学的な変換と光学的な色の変換に分けることができる．

5.1.1 幾何変換

幾何学的な変換は，カメラとシンボルの距離に応じた拡大縮小や，カメラの向きに応じた回転などが考えられる．この変換の詳細については，カメラをピンホールカメラとしてモデル化する透視投影変換を考えることで議論できるが，本演習では簡単に画像平面上での拡大縮小，並進，回転のみを取り扱うものとする．ただし，拡大縮小や並進，回転だけでは対応できないような幾何変換はごく自然に起こるため，実際にはこれらの単純な変換だけでは十分に対応できず，例えば文字列の誤抽出などに繋がる．厳密な幾何変換への対応は発展課題とする．

5.1.2 色変換

シンボルが明モジュールと暗モジュールの 2 つから構成されていたとしても，シンボルが置かれた場所の照明環境，シンボルが印刷された材質（反射特性）などの影響を受け，カメラで取得される画像は多値を取る濃淡画像となる．例えば本演習で扱う画像では，0 から 255 の 256 階調を持つ画像となる．そのためカメラで取得した画像は濃淡を持ち，各画素を明モジュール／暗モジュールの 2 値に変換する，2 値化を行う必要がある．

各画素の濃淡は，明モジュールに対応する画素では大きな値，暗モジュールに対応する画素では小さな値を取る傾向にある．そのため，ある閾値 T を定め，これを境として各画素の濃淡値を 2 値化する閾値処理が最も基本的な処理としてよく用いられる．明モジュールと暗モジュールのコントラストが十分にある場合には，画像全体で一定値を適切に定めること（固定閾値処理）でうまく 2 値化することができる．

5.2 シンボルの検出

5.2.1 位置検出パターンの特徴を利用した検出処理

位置検出パターンは，先述の拡大縮小，並進，回転変換に対して不変な特徴を持っている．それは，パターンの中心を通る直線上では，どのような向きでも明暗が 1:1:3:1:1 になるという特徴である（図 9）．この特徴を利用することで，どのような大きさで，どのような向きになっているか分からないシンボルを画像中から検出することができる．

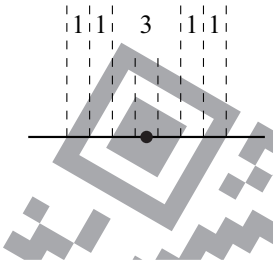

図 9　位置検出パターンの特徴

具体的には，画像の各行，各列に対し，明暗が 1:1:3:1:1 となっている箇所を検出する処理を行う．画像の何れかの行，列が位置検出パターンの中心を通っていれば，その部分が検出されることになる．シンボルが画像中である程度大きく写っていれば，明暗の比率が 1:1:3:1:1 から少しの誤差を許容するように検出することで，画像の行・列が多少パターンの中心からずれていたとしても，位置検出パターンをうまく検出できる．ただし，許容誤差を小さくし過ぎると検出されないし，大きくし過ぎると他の部分まで検出されてしまう．許容誤差をどの程度にすべきか，経験的に設定するのではなく，各自で考えて設定すること．

位置検出パターンの中心が画像中のどの位置にあるかを検出できただけの段階では，画像中での

シンボルの大きさや向きが分からない．そのため，明暗の比率は 1:1:3:1:1 であると分かっていても，パターンそのものの長さは分からない．このようなパターンの検出は難しそうに思えるが，ランレングス符号化を用いると比較的容易に実現できる．ランレングス符号化とは，明モジュールや暗モジュールが行方向（または列方向）で何画素分連続して並んでいるかを表現したものである．画像の各行，各列に対してランレングス符号化を施すと，明モジュールと暗モジュールの長さの配列が獲得される．そのような配列が得られれば，配列の連続した部分に対し，許容誤差を考慮しながら各要素の比が 1:1:3:1:1 となっているような部分を見つけることは困難ではない．

実際に埋め込む文字列にも依るが，明暗パターンの検出処理を行ってみると，各行，各列で 1:1:3:1:1 の比率となる部分は位置検出パターン以外でも検出されるだろう．しかし，同じ位置で，行方向にも列方向にも 1:1:3:1:1 の比率となるところは位置検出パターン以外ではほとんどない．文字列の埋め込みにより符号化領域に位置検出パターンと全く同じパターンが現れる場合は誤検出は避けられないが，オリジナルの QR コードの仕様では，そのようなパターンの発生はマスク処理を施すことで抑制するようになっている．このような特異なパターンを位置検出に利用していることが，大きさや向きの変化に頑健なシンボル検出を高速に実現することに貢献している．

5.2.2 位置検出パターンからのシンボルの大きさ，位置，向きの獲得

シンボルの 3 つの隅にある位置検出パターンを検出することができると，検出されたパターンの中心座標，行・列方向でのパターンの長さが得られる．こうして検出されたパターンのうち，左上，右上，左下となるパターンがそれぞれどれかを判定しなければならない．

ここで，左上，右上，左下のパターンの中心座標をそれぞれ P_{UL}，P_{UR}，P_{DL} とする．撮影による幾何変換で大きな歪みが発生していないと仮定すれば，$\overrightarrow{P_{UL}P_{UR}}$ と $\overrightarrow{P_{UL}P_{DL}}$ はほぼ直交すると言える．つまり，この 2 つのベクトルの内積はほぼ 0 になると言える．これに対し，$\overrightarrow{P_{DL}P_{UR}}$ と $\overrightarrow{P_{DL}P_{UL}}$，$\overrightarrow{P_{UR}P_{UL}}$ と $\overrightarrow{P_{UR}P_{DL}}$ はそれぞれ直交しない．そのため，検出されたパターンのうち，内積が 0 に最も近くなるような組み合わせを見つければ，P_{UL} の座標がどれかを得ることができる．あとは，P_{UR} と P_{DL} の区別が必要であるが，これについては同ベクトルの外積を判定に利用することができる．具体的な方法については各自で考えよ．

本演習の基本仕様では，シンボルのサイズ W_{sym} は既知としていた．しかし，W_{sym} が未知であったとしても，各パターンの中心座標 P_{UL}，P_{UR}，P_{DL}，及び左上のパターンを検出した際の行 (列) 方向でのパターンの長さ L_{UL} から，シンボルのサイズ W_{sym} を取得することができる (図 10)．例えば，中心座標 P_{UL}，P_{UR}，P_{DL} から，シンボルの傾きを計算することができ，それを基にパターンの長さ L_{UL} からパターンのサイズ W_{UL} を求めることができる．パターンのサイズ W_{UL} は，シンボルの 7 モジュール分であることは既知であるため，1 モジュールに対応する画像中でのサイズを知ることができる．パターン間の距離 $\left|\overrightarrow{P_{UL}P_{UR}}\right|$ は，シンボルのサイズ W_{sym} からクワイエットゾーンの幅と位置検出パターンの幅 1 つ分を合わせた，$(W_{sym} - 15)$ モジュール分のサイズと等しいことになる．以上を基にすれば，シンボルのサイズ W_{sym} を算出することが可能となる．

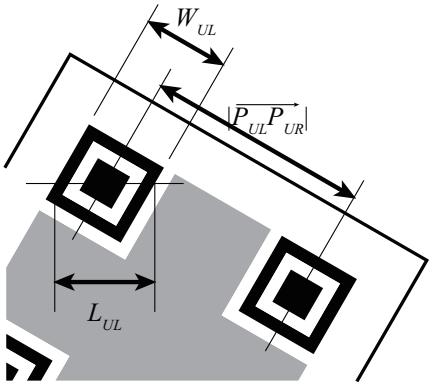

図 10　シンボルサイズの算出

5.3　符号化領域からの文字列の抽出

5.3.1　原シンボルの復元

カメラで取得した画像中から位置検出パターンの位置 P_{UL}，P_{UR}，P_{DL} やシンボルサイズ W_{sym} が得られたら，これを基に原シンボルを復元することができる．まず，シンボルを構成する各モジュールについて，その中心の画像上での座標を求める（図 11）．その座標における画素値を基に，そのモジュールが明モジュールであるか暗モジュールであるかを判定する．このような判定を全てのモジュールについて行い，これを並べれば，原シンボルを復元することができる．

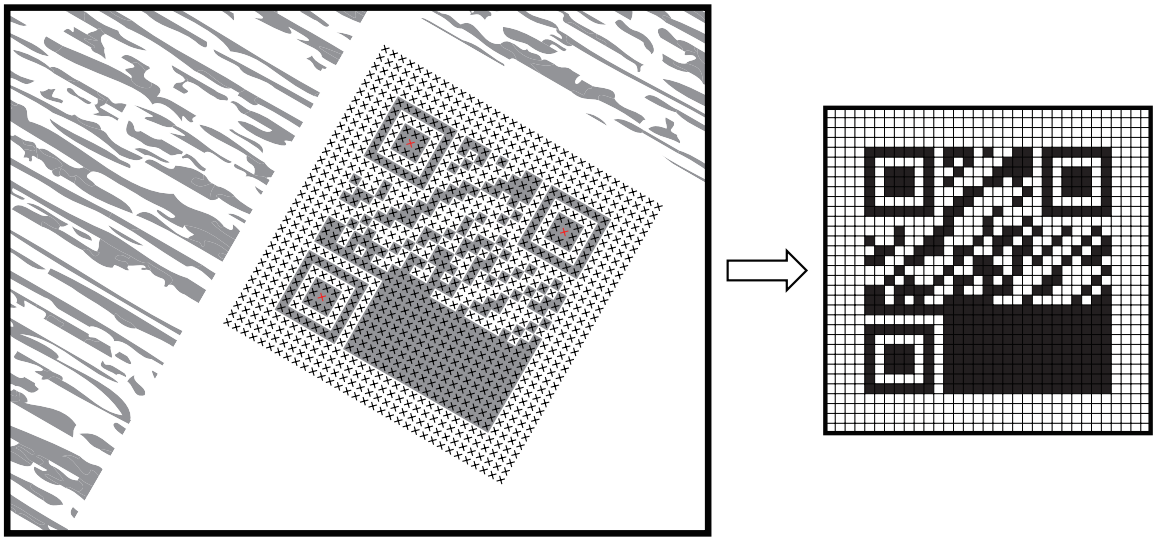

図 11　原シンボルの復元

5.3.2 文字列の抽出

原シンボルを復元することができたら，その中から符号化領域を特定することができる．符号化領域にビット列を埋め込んだときと逆の手順をとれば，符号化領域からビット列を抽出することができる．また同様に，埋め込む文字列をビット列に変換した時と逆の手順を取れば，抽出したビット列を文字列に変換することができる．

演習課題

【必修課題】2 次元コードの生成と認識

■**【必修課題 1】** 本演習の仕様に基づき，1 辺が W_{sym} モジュールの正方形シンボルを作成した時，クワイエットゾーンや位置検出パターンを除いた符号化領域に埋め込むことができる文字のバイト数を示せ．W_{sym} を用いた数式を示すとともに，$W_{sym} = 33$ である場合のバイト数も計算せよ．

■**【必修課題 2】** 学籍番号，氏名を埋め込んだシンボルを生成し，ppm 形式でファイルに出力するプログラムを作成せよ．

■**【必修課題 3】** 各自で生成した 2 次元コードシンボルに対し，任意の拡大・回転・並進の変換を施すプログラムを作成し，幅 320pix，高さ 240pix の画像を数パターン作成せよ．

■**【必修課題 4】** 幾何変換を施した画像を入力として，位置検出パターンを検出するプログラムを作成せよ．この際，許容誤差をどう設定したか，具体的なアルゴリズムを示すとともに，その根拠を述べよ．

■**【必修課題 5】** 検出した位置検出パターンを基に，文字列を取り出すプログラムを作成せよ．

■**【必修課題 6】** 印刷した 2 次元パターンをカメラで撮影し，その画像に対してシンボルの認識を行うプログラムを作成せよ．なお，カメラから撮影画像を取りこむプログラムはこちらで用意したものを利用してよい

【発展課題 A】異なるシンボルサイズへの対応

必修課題 1 で算出した通り，本演習の仕様で作成したシンボルには埋め込めるバイト数に限りがある．埋め込める容量を向上させるもっとも単純な方法は，埋め込むデータ量に応じてシンボルサイズ W_{sym} を大きくすることである．シンボルサイズを可変にするためには，先述の通り認識時に W_{sym} を算出するような工夫が必要となる．これを実現せよ．

【発展課題 B】2 値化処理の性能向上

カメラでシンボルを撮影する際，その環境によっては 2 値化がうまく働かない．シンボルを撮影する条件が常に一定と言えない場合には，取得された画像に応じて自動的に閾値を自動決定するアプローチも取られる．その例として，P タイル法，モード法，判別分析法などがある [4].

また，照明ムラなど，画像の場所によって濃淡レベルが異なる場合，画像全体に対して 1 つの閾値を定める固定閾値処理ではうまくいかない．例えばカメラ自体がシンボルに落とす影によって濃淡レベルが不均一になることが多く，これが原因でどのような固定閾値を設定したとしても明モジュールと暗モジュールをうまく判定できない場合がある．そのような場合，各画素で最適な閾値を計算する動的閾値処理もいくつか考案されている．

各手法の利点・欠点を考察した上で何れかの 2 値化処理を実装し，その手法の性質を実験によって示せ．

【発展課題 C】マスク処理

符号化領域内に位置検出パターンと同じ 1:1:3:1:1 パターンが現れないように，また 2 値化処理がうまく働くように，明モジュールと暗モジュールがバランス良く配置されるようにマスク処理が行われる．予めいくつかのマスクパターンを用意し，符号化領域の各モジュールと排他的論理和（XOR）演算を行う．その中で最適なものを最終的なシンボルとする（図 12）．このようなマスク処理を実装せよ．

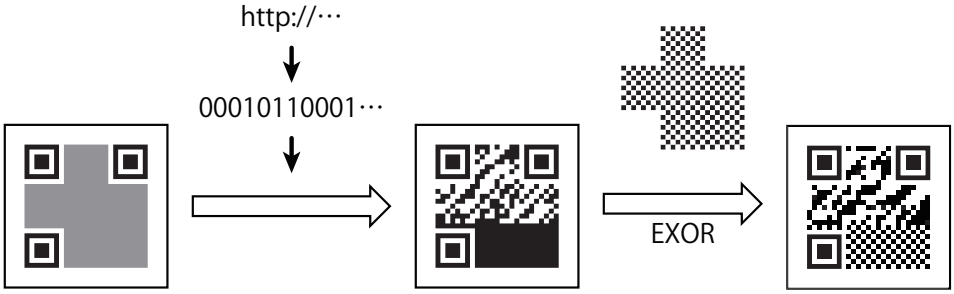

図 12　マスク処理

【発展課題 D】歪みへの対処

本演習では，撮像における幾何変換として簡単に拡大縮小，並進，回転のみを取り扱ったが，実際にはもう少し複雑である．以下の補足説明を参考に，あとに述べる課題に取り組め．

平面射影変換

カメラによる撮像過程は一般的に，ピンホールカメラをモデル化した透視投影モデルが用いられる．透視投影モデルは，3 次元空間中の点がカメラの画像平面上の 1 点に射影される過程を表す．

簡単のため，いまカメラのピンホールが点 $(0,0,0)$ にあり，カメラの方向は $(0,0,1)$ を向いているものとする (図 13)．また，画像平面 I が $z = z_I (z_I > 0)$ に設定されるものとする．実際のピンホールカメラでは画像平面は $z_I < 0$ となるが，計算理論上はピンホールのどちら側に画像平面 I があっても変わりはない．このようにカメラを基準として定められる座標系を，以後カメラ座標系と呼ぶ．

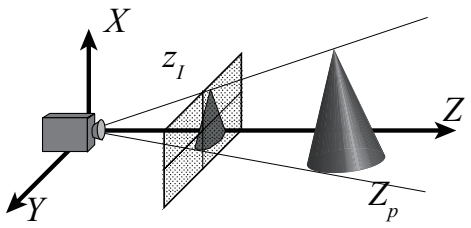

図 13　透視投影

このとき，空間内の一点 $P = (X_p, Y_p, Z_p)^T$ が画像平面上の 1 点 $p = (x_p, y_p, z_I)^T$ に投影されるとすると，これらの間には次のような関係式が成り立つ．

$$x_p = \frac{z_I}{Z_p} X_p, \quad y_p = \frac{z_I}{Z_p} Y_p \tag{1}$$

このような変換を行列の積で表現するため，同次 (斉次) 座標系を導入する．同次座標系は，空間中の座標を表す要素に 1 つの要素を加えたもので，3 次元座標であれば 4 次元ベクトル，2 次元座標であれば 3 次元ベクトルで表現するものである．3 次元空間中の 1 点 (X, Y, Z) は $(X, Y, Z, 1)^T$ として表現される．同次座標系では，各要素がスカラー倍されたベクトルは全て同じものとして扱われる．つまり，$(X', Y', Z', \gamma)^T$ は $(X'/\gamma, Y'/\gamma, Z'/\gamma, 1)^T$ と同じであるとして扱う．ここで，スカラー倍を s と表記することにし，次のように表わすこととする．

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ \gamma \end{pmatrix} = s \begin{pmatrix} \frac{X'}{\gamma} \\ \frac{Y'}{\gamma} \\ \frac{Z'}{\gamma} \\ 1 \end{pmatrix} \tag{2}$$

このような同次座標系を導入すれば，式 1 は以下のように行列の積で表される．

$$\begin{pmatrix} z_I & 0 & 0 & 0 \\ 0 & z_I & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix} = \begin{pmatrix} z_I X_p \\ z_I Y_p \\ Z_p \end{pmatrix} = s \begin{pmatrix} \frac{z_I}{Z_p} X_p \\ \frac{z_I}{Z_p} Y_p \\ 1 \end{pmatrix} = s \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix}$$

このとき，2 次元コードが 3 次元空間中で Z 軸に垂直な平面上にあるとき，撮影対象の全ての点において Z_p は定数とみなせるため，(X_p, Y_p) と (x_p, y_p) は単純に $\frac{z_I}{Z_p}$ 倍の拡大縮小の関係にあるとみなせることになる．

同次座標系を用いれば，並進や回転も行列の積として記述することができる．例えば，3 次元空間における回転を 3×3 行列 $\mathbf{R} = \{r_{ij}\}$，並進を 3 次元ベクトル $\mathbf{t} = \{t_i\}$ とすると，

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \mathbf{R} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \mathbf{t}$$

として表現できるが，同次座標系ではこれを

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

と表現できる．

先ほどはカメラのピンホールが点 $(0, 0, 0)$ にあり，カメラの方向は $(0, 0, 1)$ を向いているものとしたが，これを任意の位置から任意の向きを向いているとした場合には，先ほどの回転行列 $\mathbf{R}$，並進ベクトル $\mathbf{t}$ を用いて次式のように一般化することができる．

$$\begin{aligned} s \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} &= \begin{pmatrix} z_I & 0 & 0 & 0 \\ 0 & z_I & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} z_I r_{11} & z_I r_{12} & z_I r_{13} & z_I t_1 \\ z_I r_{21} & z_I r_{22} & z_I r_{23} & z_I t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix} \end{aligned}$$

3 次元空間中のシンボルが平面上 $(Z_p = 0)$ にあるとしたとき，そのシンボル (X_p, Y_p) がカメラ画像平面に射影される式は

$$s \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \begin{pmatrix} z_I r_{11} & z_I r_{12} & z_I t_1 \\ z_I r_{21} & z_I r_{22} & z_I t_2 \\ r_{31} & r_{32} & t_3 \end{pmatrix} \begin{pmatrix} X_p \\ Y_p \\ 1 \end{pmatrix}$$

となり，行列の各要素は任意の値を取ることができることが知られている [5]．この行列は平面射影変換と呼ばれており，スカラー倍 s を除くと，自由度が 8 となる．そのため，カメラ画像とシンボル上で 4 組以上の対応点を見つけることができれば，この行列を推定することが可能となる．

【発展課題 D-1】コードの拡張 - 位置合せパターン

位置検出パターンのみでは 3 組の対応しか得られないため，上記のような変換をうまく推定することができない．そこで，暗の 1×1 モジュールと明の 3×3 モジュールを重ねた位置合せパターンをシンボル中にいくつか配置すれば，その位置を検出することで平面射影変換を推定することが

できるようになる．このような位置合せパターンをシンボルに埋め込み，これを検出することでより複雑な幾何変換にも対応できるように工夫せよ．

■ヒント　この行列は，例えば特異値分解のような行列演算のみを用いて求める方法もあるが，前述の同次座標系の特徴を踏まえて連立方程式を立て，これを解く方が実装が容易であろう．

【発展課題 D-2】コードの拡張 - タイミングパターン

シンボルがひずんだり，モジュールサイズに誤差が生じたりしている場合，タイミングパターンをあらかじめ設定しておくことにより，各モジュールの中心座標を正しい位置に補正することができる．このようなタイミングパターンを採用し，中心座標を誤らずに推定できるように工夫せよ．

【発展課題 E】隠れへの対処

本演習では，埋め込む文字列を変換したビット列をそのまま符号化領域に埋め込んだ．そのため，符号化領域で隠れなどの画像の損傷が起きると，抽出される文字列は大きく異なるものとなる．そこで，ビット列を符号化領域に埋め込む際に誤り訂正符号化などを採用することで冗長化しておけば，隠れなどの多少の損傷があっても安定して文字列を抽出することができるようになる．誤り訂正符号化を導入し，隠れへの頑健性を強化せよ．

参考文献

[1] これで分かった 2 次元シンボル―バーコードのすべて，社団法人 日本自動認識システム協会 編，オーム社，2004.

[2] 二次元コードシンボル-QR コード-基本仕様，JIS-X-0510

[3] QR コードドットコム-QR コードのあゆみのご紹介，http://www.denso-wave.com/qrcode/qrstandard.html

[4] 新編 画像解析ハンドブック，高木幹雄, 下田陽久 監修，京大学出版会，2004.

[5] コンピュータビジョン，David A. Forsyth, Jean Ponce 著, 大北剛 訳，共立出版，2007.