1、課題１　標本化間隔と空間解像度

1，２　課題内容

画像をダウンサンプリングして（標本化間隔を大きくして）表示する。

1，３　結果

廃墟の画像を原画像とする。1600×1066 のディジタル画像である。

```
ORG=imread('haikyo.jpg'); % 原画像の入力
imagesc(ORG); axis image; % 画像の表示
```

によって、原画像を表示した結果を図 1 に示す。

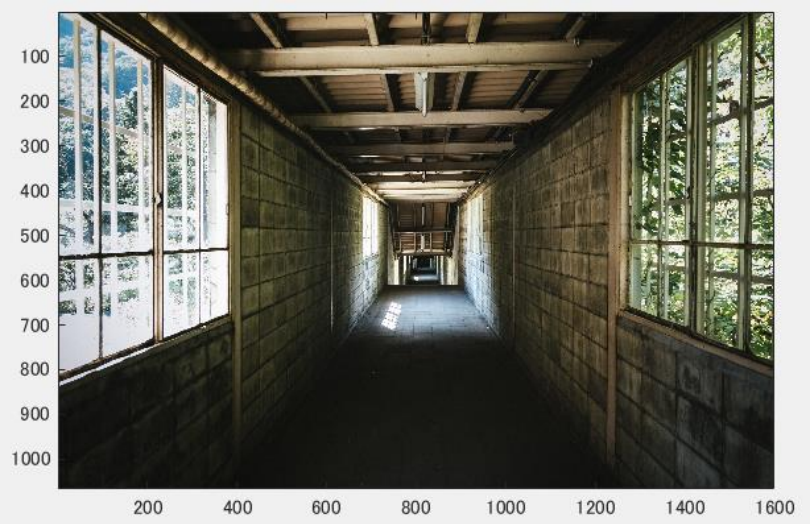

図 1、原画像

```
IMG = imresize(ORG,0.5); % 画像の縮小
IMG2 = imresize(IMG,2,'box'); % 画像の拡大
```

によって、原画像を 1/2 倍に縮小したあと、2 倍に拡大することで 1/2 サンプリングすることができる。拡大する際に、単純補間する「box」を設定する。1/2 サンプリングした結果を図 2 に示す。

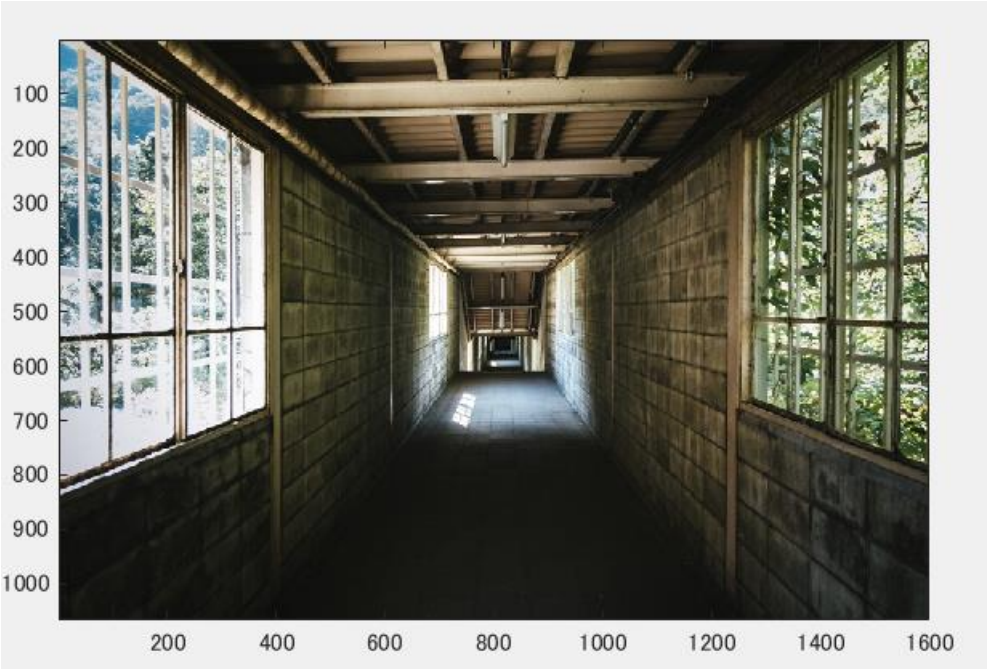

図 2、1/2 サンプリング

また、1/4 サンプリングは 1/2 倍に縮小したものをさらに 1/2 倍に縮小した後、4 倍に拡大すればよい。よって、以下に示すようになる。

```
IMG = imresize(ORG,0.5); % 画像の縮小
IMG2 = imresize(IMG,4,'box'); % 画像の拡大
```

また、1/8 から 1/32 も同様のため、それぞれ拡大する倍数を 8 から 32 に変更し繰り返す。サンプリングの結果を図 3〜6 に示す。

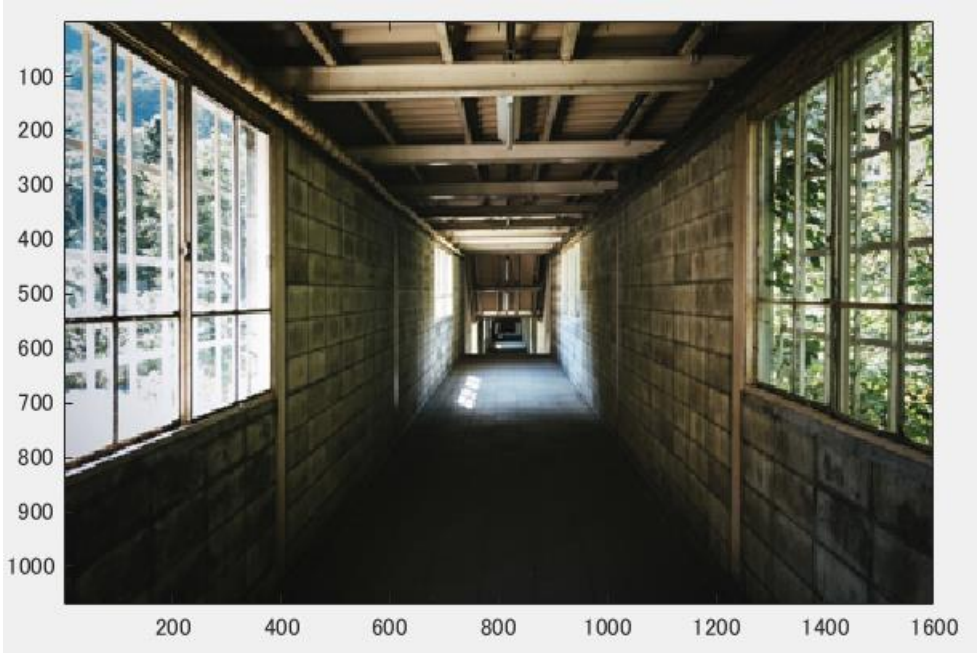

図 3、1/4 サンプリング

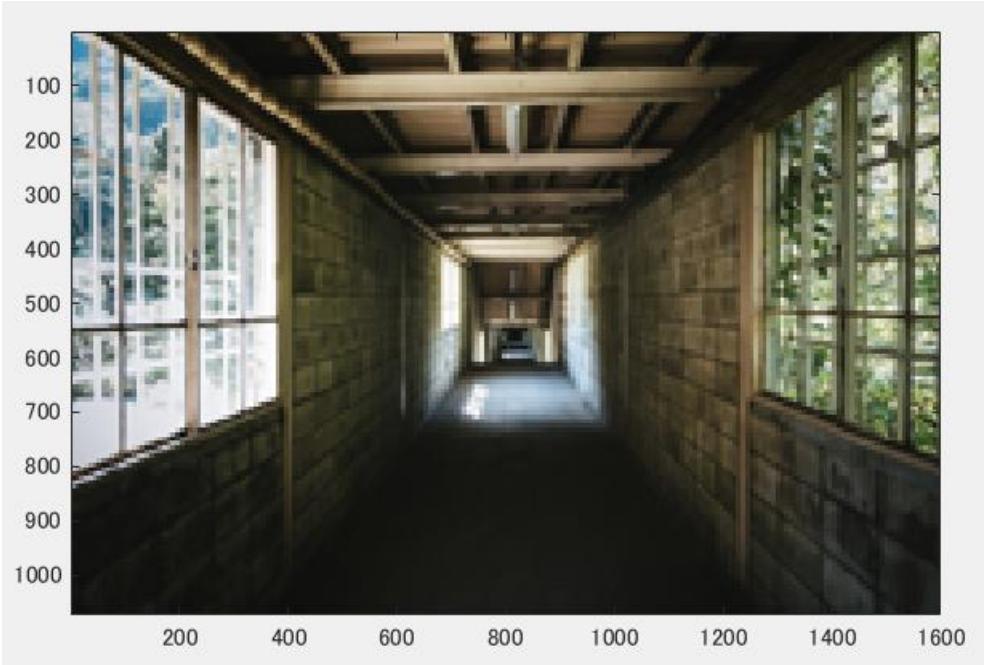

図 4、1/8 サンプリング

図 5、1/16 サンプリング

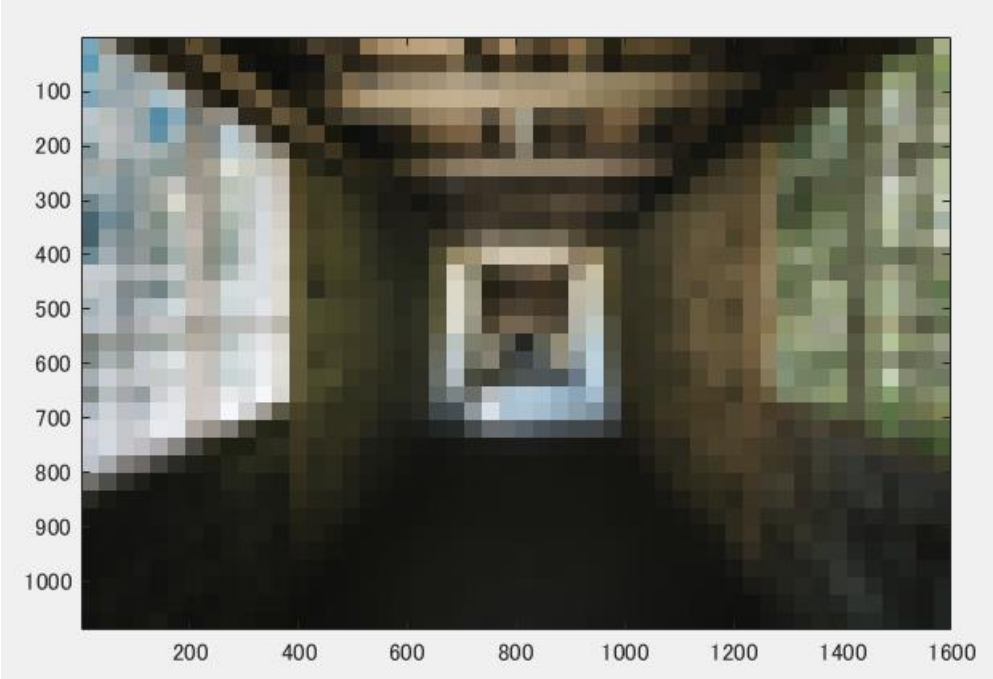

図 6、1/32 サンプリング

2、課題 2　階調数と疑似輪郭

2，2　課題内容

2 階調，4 階調，8 階調の画像を生成する。

2，3　結果

```
ORG=imread('haikyo.jpg'); % 原画像の入力
ORG = rgb2gray(ORG); colormap(gray); colorbar;
imagesc(ORG); axis image; % 画像の表示
```

によって、図 1 の原画像を白黒に変換し表示する。表示した結果を図 7 に示す。

図 7、白黒画像

2 階調の画像を生成するためには、256 階段の階調を 2 で割った 128 ごとに区切っていけばよい。よって、以下に示すようになる

```
IMG = ORG>128;
imagesc(IMG); colormap(gray); colorbar;　axis image;
```

2 階調画像を表示した結果を図 8 に示す。

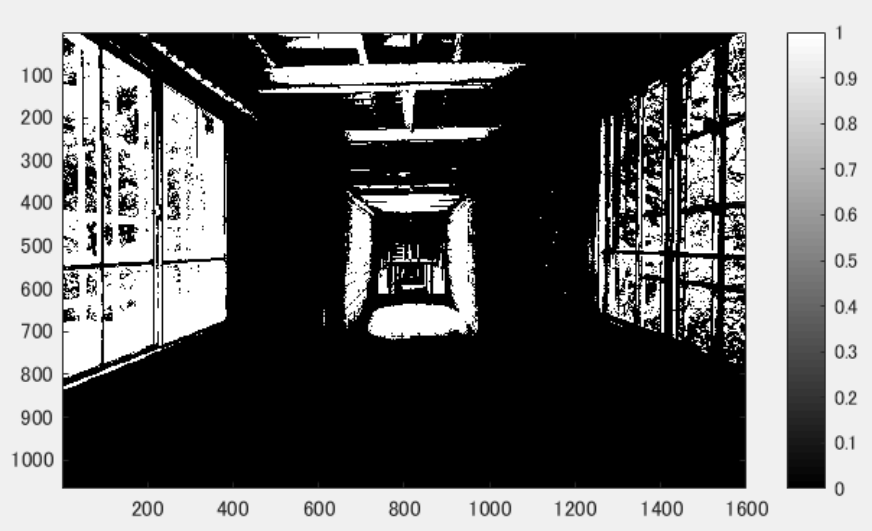

図 8、2 階調の画像

同様に 4 階調の画像を生成するためには、256 を 4 で割った 64 で区切っていけばよい。よって、以下に示すようになる。

```
IMG0 = ORG>64;
IMG1 = ORG>128;
IMG2 = ORG>192;
IMG = IMG0 + IMG1 + IMG2;
imagesc(IMG); colormap(gray); colorbar;    axis image;
```

4 階調画像を表示した結果は図 9 に示す。

図 9、4 階調の画像

同様に 8 階調の画像を生成するためには、256 を 8 で割った 32 で区切っていけばよい。よって、以下に示すようになる。

```
IMG0 = ORG>32;
IMG1 = ORG>64;
IMG2 = ORG>96;
IMG3 = ORG>128;
IMG4 = ORG>160;
IMG5 = ORG>192;
IMG6 = ORG>224;
IMG = IMG0 + IMG1 + IMG2 + IMG3 + IMG4 + IMG5 + IMG6;
imagesc(IMG); colormap(gray); colorbar;    axis image;
```

8 階調画像を表示した結果は図 10 に示す。

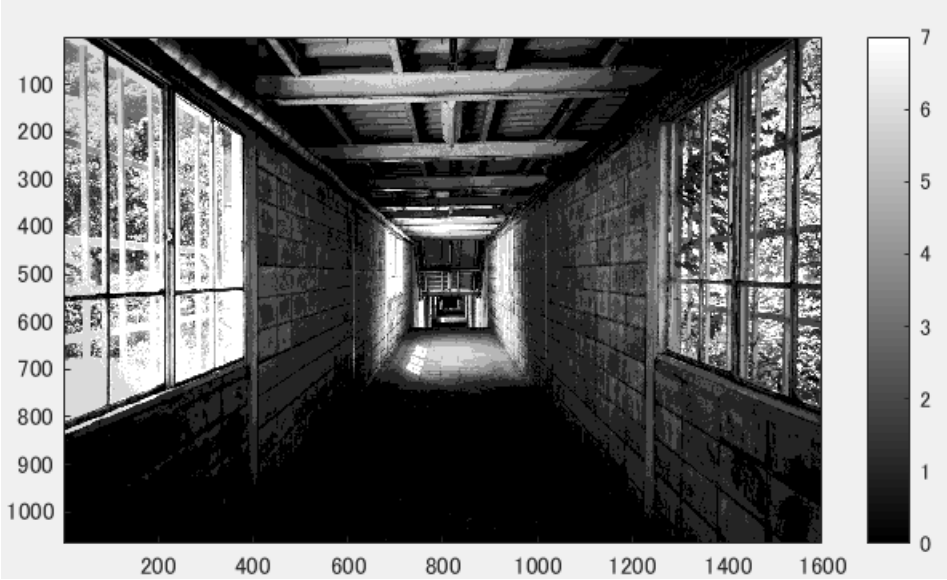

図 10、8 階調の画像

3、課題 3　閾値処理

3，2　課題内容

閾値を 4 パターン設定し、閾値処理した画像を示す。

3，3　結果

MG = ORG > 64; % 輝度値が64以上の画素を1，その他を0に変換
imagesc(IMG); colormap(gray); colorbar;

によって、図 7 の白黒画像を、輝度値が 64 以上の画素を 1，その他を 0 にした白黒画像を生成する。つまり、閾値を 64 とした画像である。表示した結果を図 11 に示す。

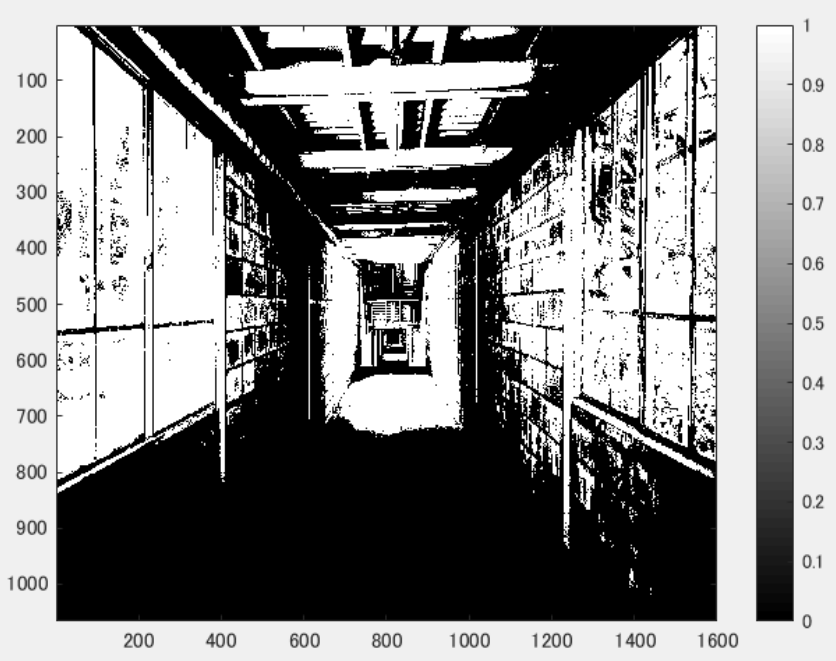

図 11、閾値を 64 とした画像

IMG = ORG > 96;
imagesc(IMG); colormap(gray); colorbar;

によって、図 7 の白黒画像を、輝度値が 96 以上の画素を 1，その他を 0 にした白黒画像を生成する。つまり、閾値を 96 とした画像である。表示した結果を図 12 に示す。

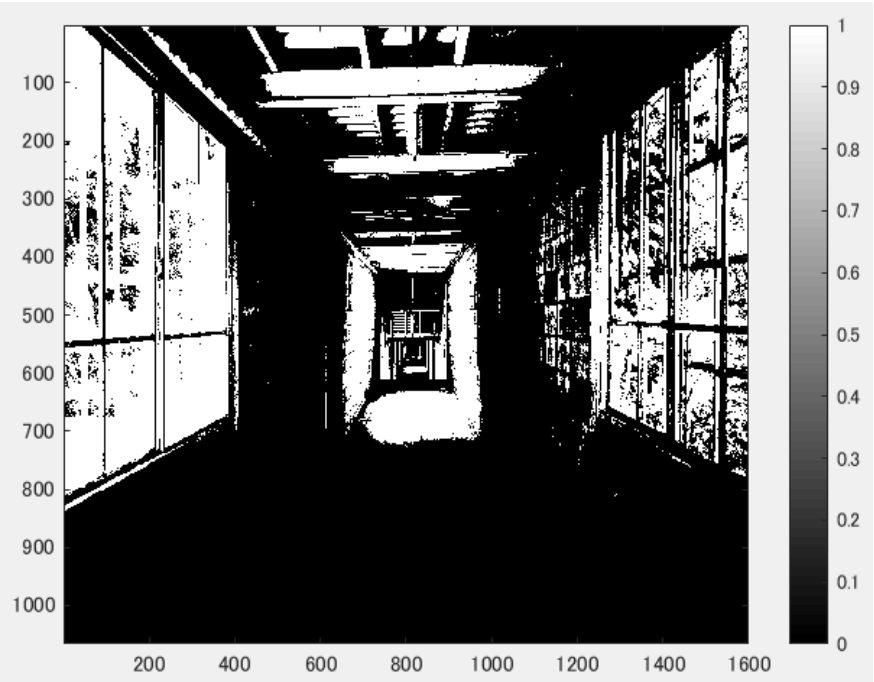

図 12、閾値を 96 とした画像

```
IMG = ORG > 128;
imagesc(IMG); colormap(gray); colorbar;
```

によって、図 7 の白黒画像を、輝度値が 128 以上の画素を 1，その他を 0 にした白黒画像を生成する。つまり、閾値を 128 とした画像である。表示した結果を図 13 に示す。

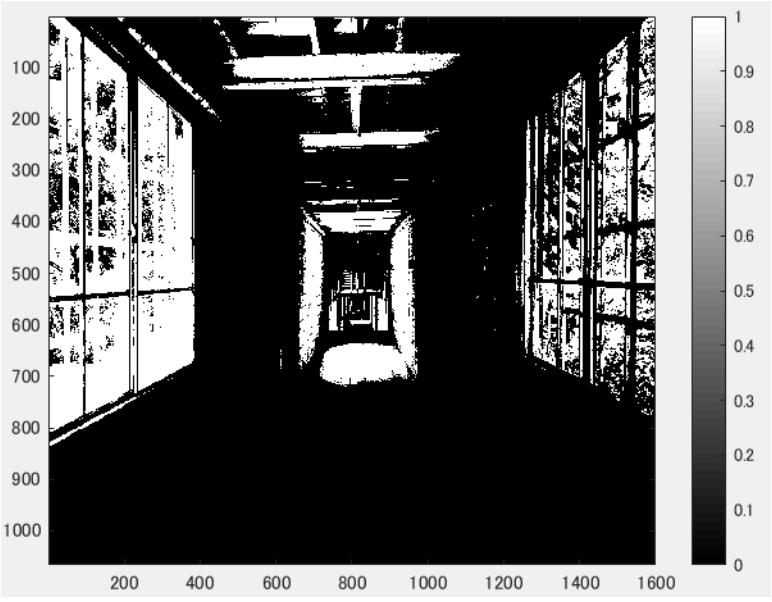

図 13、閾値を 128 とした画像

```
IMG = ORG > 196;
imagesc(IMG); colormap(gray); colorbar;
```

によって、図 7 の白黒画像を、輝度値が 192 以上の画素を 1，その他を 0 にした白黒画像を生成する。つまり、閾値を 192 とした画像である。表示した結果を図 14 に示す。

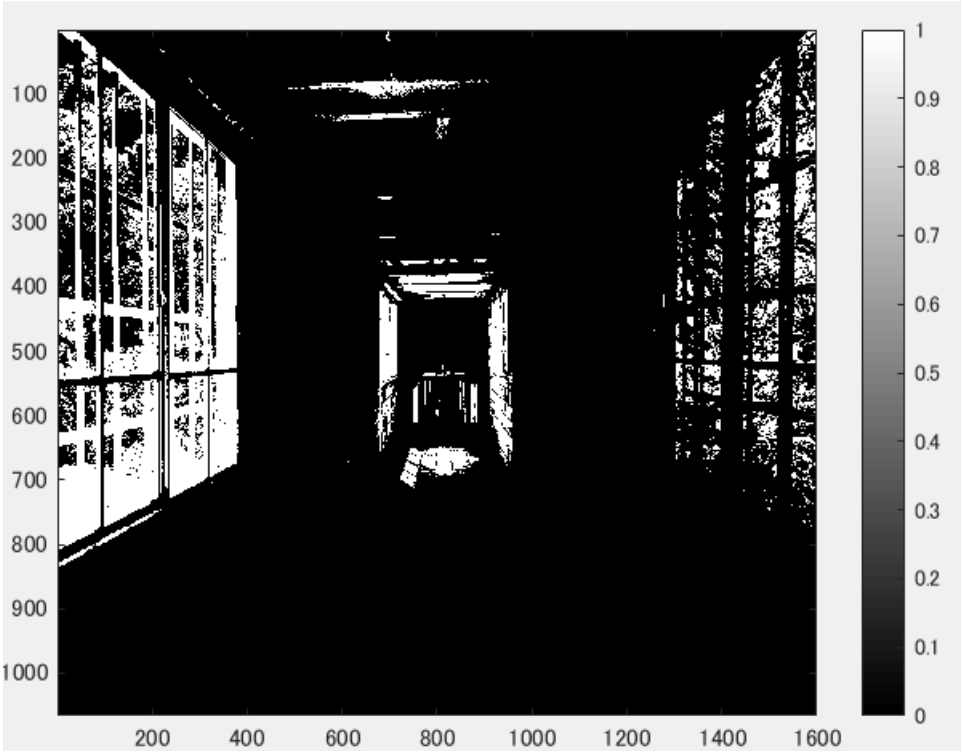

図 14、閾値を 192 とした画像

4、課題 4　画像のヒストグラム

4，２　課題内容

画像の濃度ヒストグラムを生成する。

4，３　結果

imhist(ORG); % ヒストグラムの表示

によって、図 7 の白黒画像をヒストグラム表示する。表示した結果を図 15 に示す。

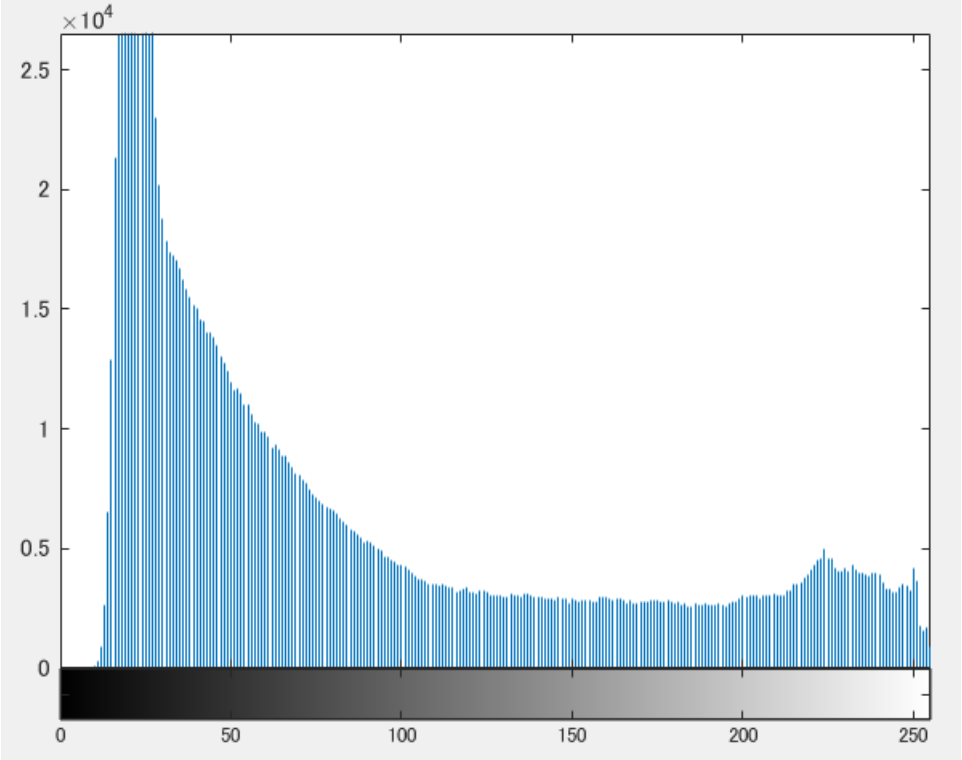

図 15、ヒストグラム表示

5、課題 5　判別分析法

5，2　課題内容

判別分析法を用いて画像 2 値化する

5，3　結果

```
C1 = H(1:i); %ヒストグラムを2つのクラスに分ける
C2 = H(i+1:256);
n1 = sum(C1); %画素数の算出
n2 = sum(C2);
myu1 = mean(C1); %平均値の算出
myu2 = mean(C2);
sigma1 = var(C1); %分散の算出
sigma2 = var(C2);
sigma_w = (n1 *sigma1+n2*sigma2)/(n1+n2); %クラス内分散の算出
sigma_B = (n1 *(myu1-myu_T)^2+n2*(myu2-myu_T)^2)/(n1+n2); %クラス間分散
の算出
```

によって、クラス内分散とクラス間分散の算出を行う。その後、

```
If max_val<sigma_B/sigma_w
max_val = sigma_B/sigma_w;
max_thres =i;
```

によって、クラス間分散/クラス内分散が最大となる値を求め、図 7 の白黒画像を 2 値化する。

2 値化した結果は図 16 に示す。

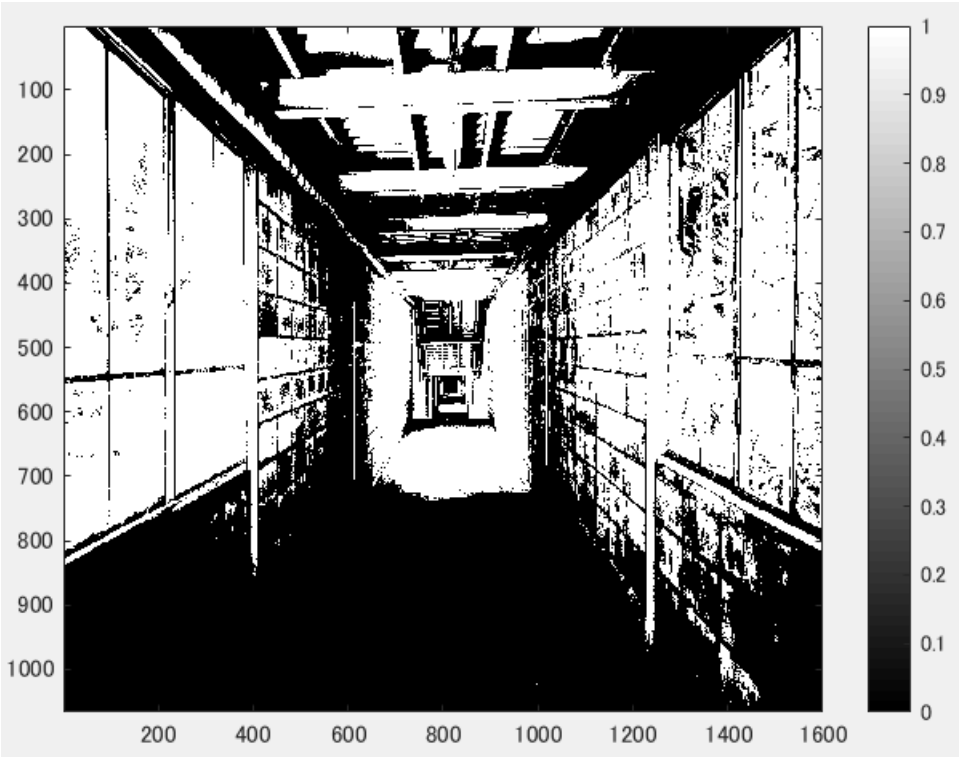

図 16、判別分析法によって 2 値化した画像

6、課題 6　画像の二値化

6，２　課題内容

画像を二値化する。

6，３　結果

IMG = ORG>128; % 128による二値化
imagesc(IMG); colormap(gray); colorbar; % 画像の表示

によって、図 7 の白黒画像を輝度値 128 で 2 値化し表示する。表示した結果を図 17 に示す。

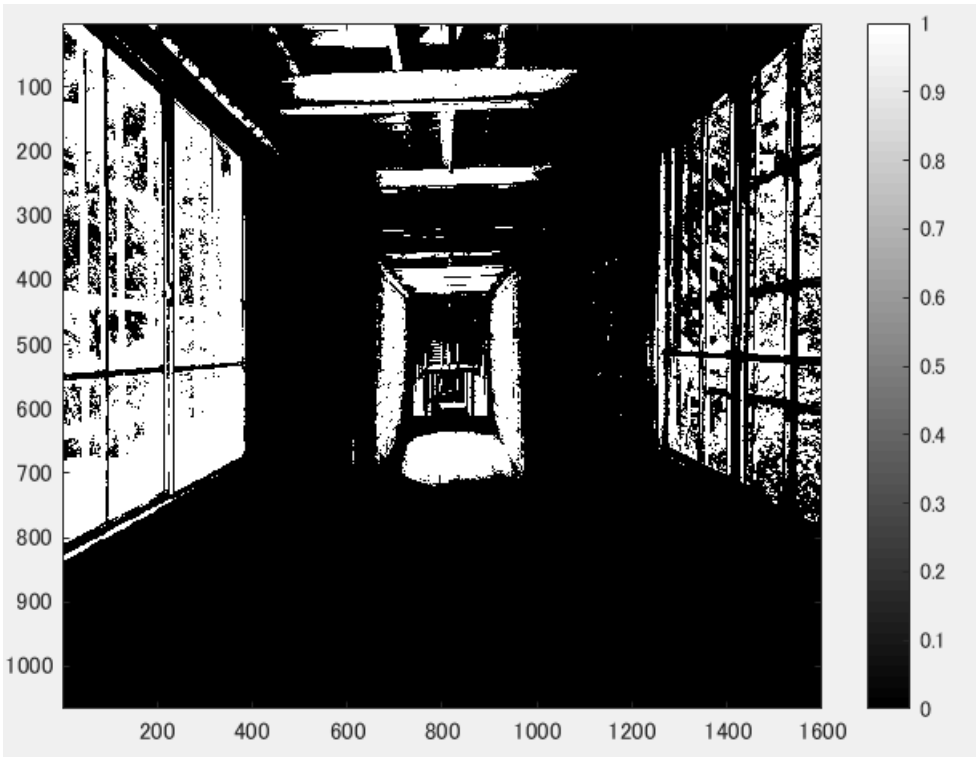

図 17、輝度値 128 での 2 値化

IMG = dither(ORG); % ディザ法による二値化
imagesc(IMG); colormap(gray); colorbar; % 画像の表示

によって、図 7 の白黒画像をディザ法で 2 値化し表示する。表示した結果を図 18 に示す。

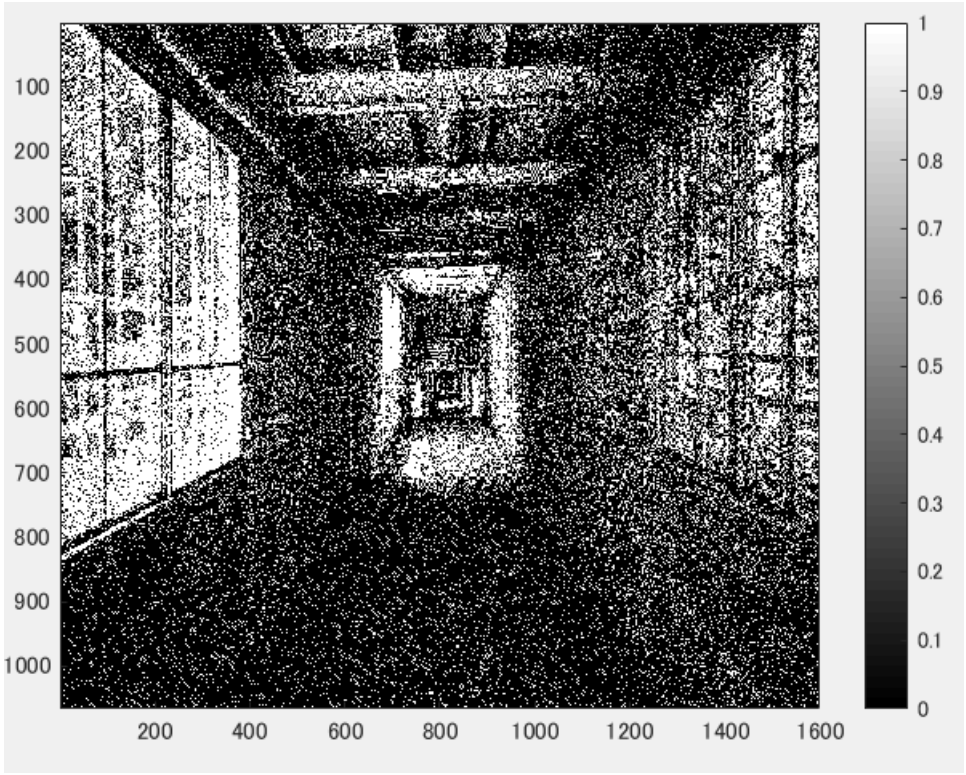

図 18、ディザ法による 2 値化

7、課題 5　ダイナミックレンジの拡大

5，2　課題内容

画素のダイナミックレンジを 0 から 255 にする。

5，3　結果

imhist(ORG); % 濃度ヒストグラムを生成、表示

によって、図 7 の白黒画像のヒストグラムを表示する。表示した結果、図 19 になった。

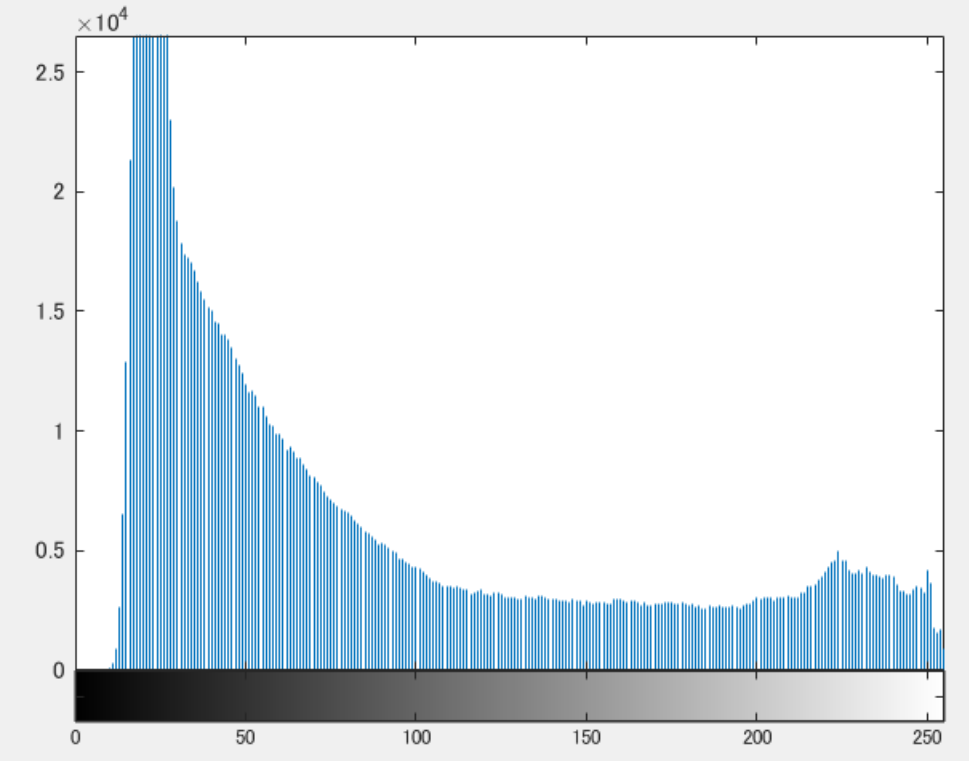

図 18、通常時のヒストグラム

```
mn = min(ORG(:)); % 濃度値の最小値を算出
mx = max(ORG(:)); % 濃度値の最大値を算出
ORG = (ORG-mn)/(mx-mn)*255;
imagesc(ORG); colormap(gray); colorbar; % 画像の表示
```

によって、ダイナミックレンジの拡大を行う。行った結果を図 19 に示す。

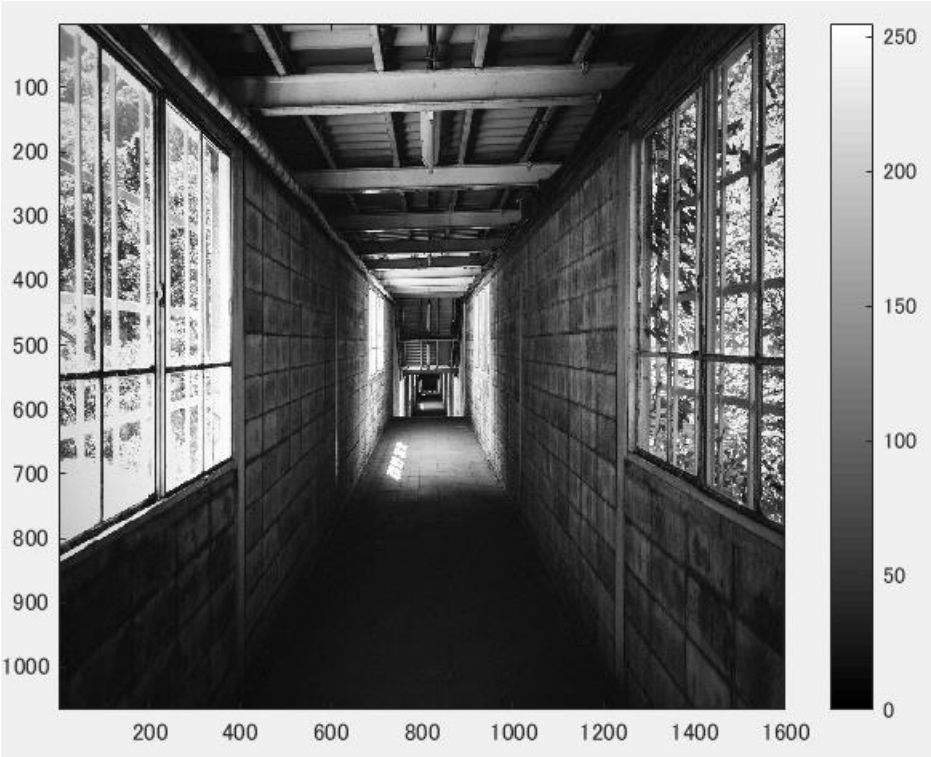

図 19、ダイナミックレンジの拡大をした画像

```
ORG = uint8(ORG); % この行について考察せよ
imhist(ORG); % 濃度ヒストグラムを生成、表示
```

によって、ダイナミックレンジの拡大を行った画像のヒストグラムを表示する。表示した結果を図 20 に示す。

また、ORG = uint8(ORG);とは、8 ビット符号なし整数列に変換するものである。今回、ダイナミックレンジの拡大で ORG が 8 ビット符号なし整数列ではなくなってしまっている。この状態ではヒストグラムを表示できないため、uint8 を使用していると考えられる。

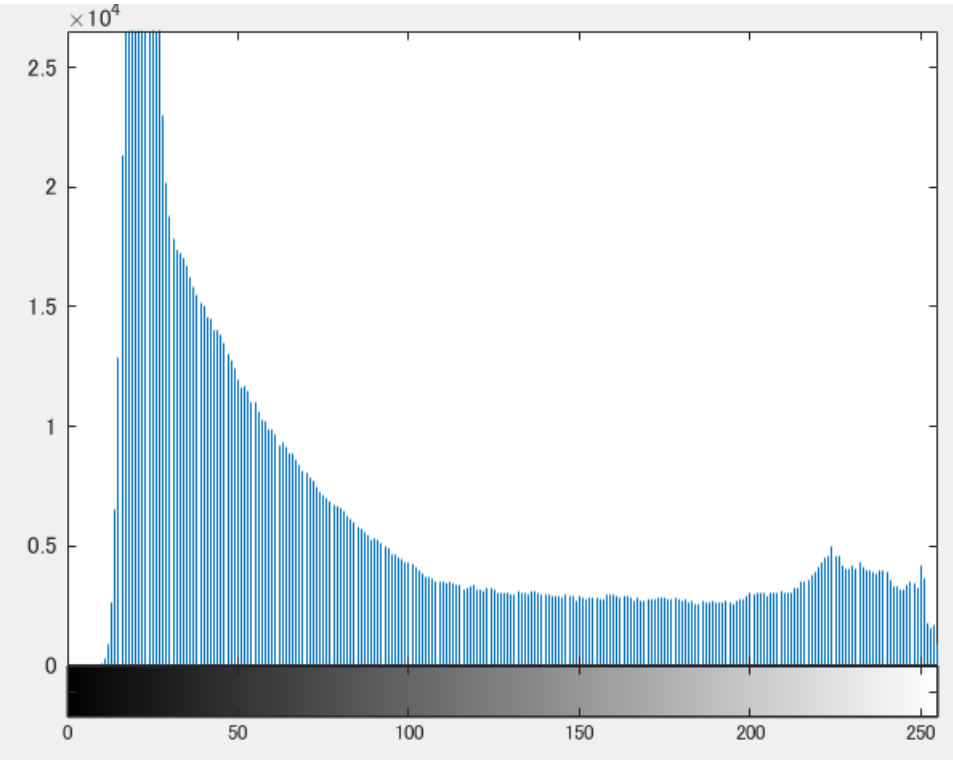

図 20、ダイナミックレンジの拡大をした画像のヒストグラム

8、課題 8　ラベリング

8，2　課題内容

二値化された画像の連結成分にラベルを付ける。

8，3　結果

IMG = ORG > 128; % 閾値128で二値化
imagesc(IMG); colormap(gray); colorbar; % 画像の表示

によって、図 7 の白黒画像を閾値 128 で二値化を行い、

IMG = bwlabeln(IMG);
imagesc(IMG); colormap(jet); colorbar; % 画像の表示

によって、ラベリングを行う。ラベリングした結果を図 21 に示す。

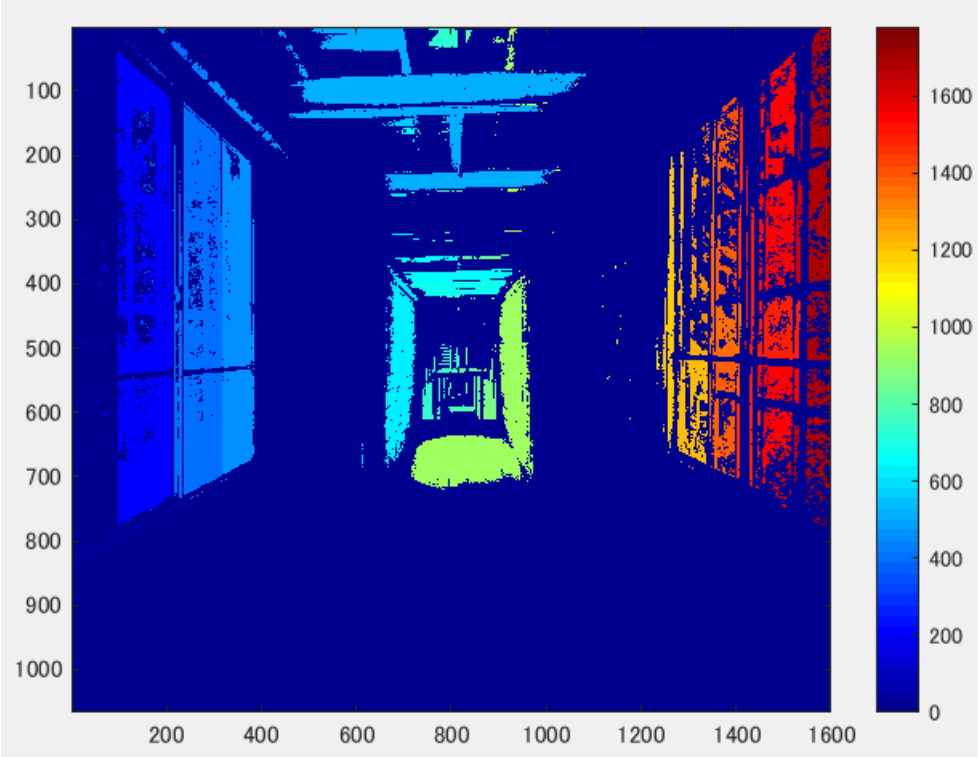

図 21、二値化画像をラベリングした画像

9、課題 9　メディアンフィルタと先鋭化

9，2　課題内容

メディアンフィルタを適用し、ノイズ除去を行う。

9，3　結果

```
ORG = imnoise(ORG,'salt & pepper',0.02); % ノイズ添付
imagesc(ORG); colormap(gray); colorbar; % 画像の表示
```

によって、ノイズを添付し表示する。ノイズを添付した画像を図 22 に示す。

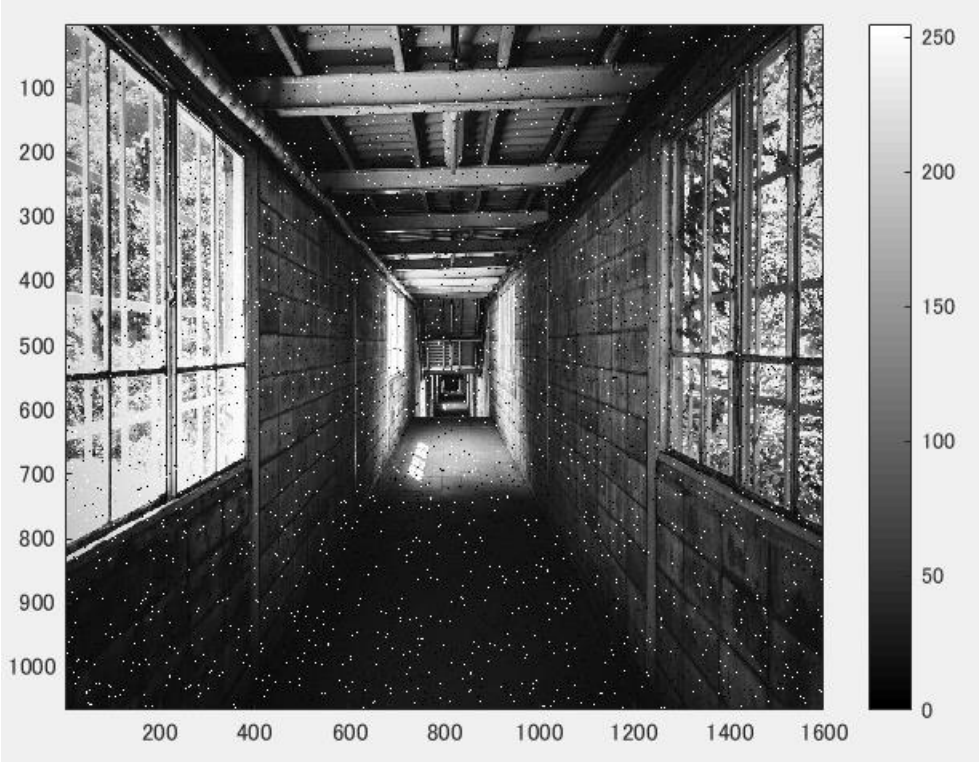

図 22、ノイズの添付

```
IMG = filter2(fspecial('average',3),ORG); % 平滑化フィルタで雑音除去
imagesc(IMG); colormap(gray); colorbar; % 画像の表示
```

によって、ノイズの添付した画像を、平滑化を用いて雑音除去する。平滑化による雑音除去の結果は図 23 に示す。

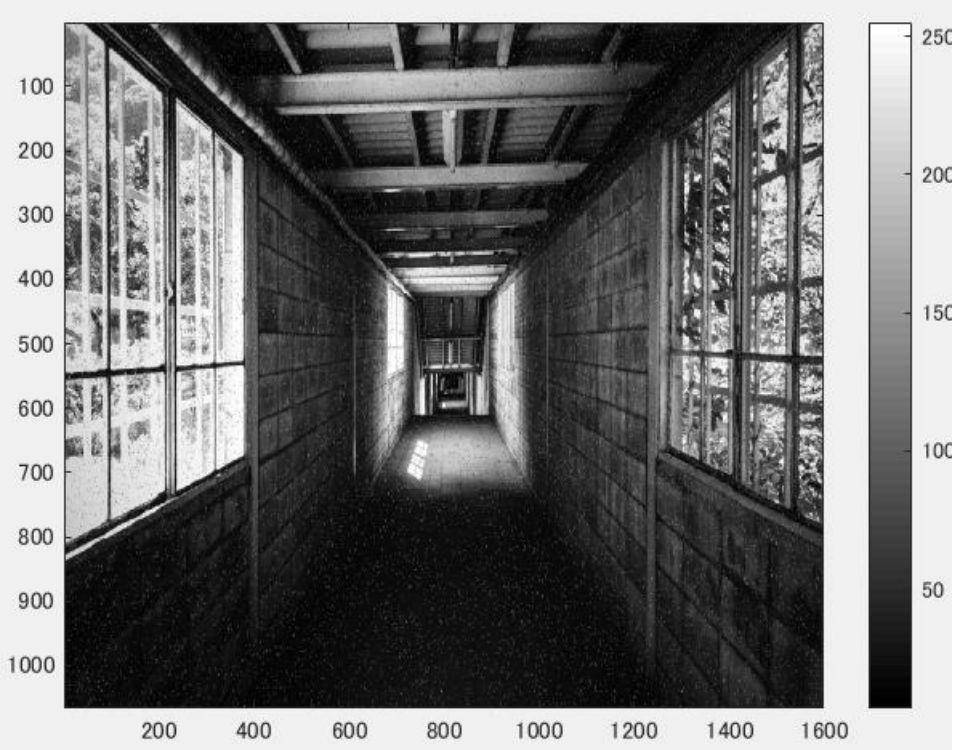

図 23、平滑化フィルタで雑音除去

```
IMG = medfilt2(ORG,[3 3]); % メディアンフィルタで雑音除去
imagesc(IMG); colormap(gray); colorbar; % 画像の表示
```

によって、ノイズの添付した画像を、メディアンフィルタを用いて雑音除去する。メディアンフィルタによる雑音除去の結果は図 24 に示す。

図 24、メディアンフィルタで雑音除去

```
f=[0,-1,0;-1,5,-1;0,-1,0]; % フィルタの設計
IMG = filter2(f,IMG,'same'); % フィルタの適用
imagesc(IMG); colormap(gray); colorbar; % 画像の表示
```

によって、フィルタの設計を行う。設計したフィルタを適用した結果を図 25 に示す。

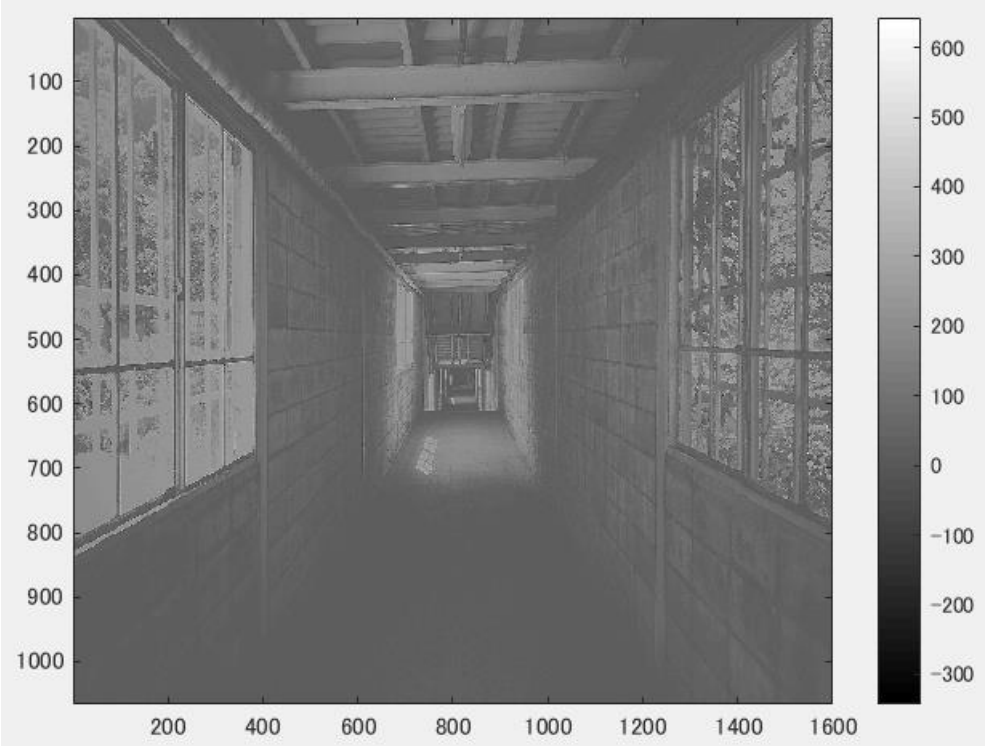
100
200
300
400
500
600
700
800
900
1000
200
400
600
800
1000
1200
1400
1600
600
500
400
300
200
100
0
−100
−200
−300

図 25、フィルタの設計

１０、課題 10　画像のエッジ抽出

１０，２　課題内容

エッジ抽出を行う。

１０，３　結果

```
IMG = edge(ORG,'prewitt'); %  エッジ抽出（プレウィット法）
imagesc(IMG); colormap('gray'); colorbar;%  画像表示
```

によって、プレウィット法を用いてエッジ抽出をする。結果を図 26 に示す。

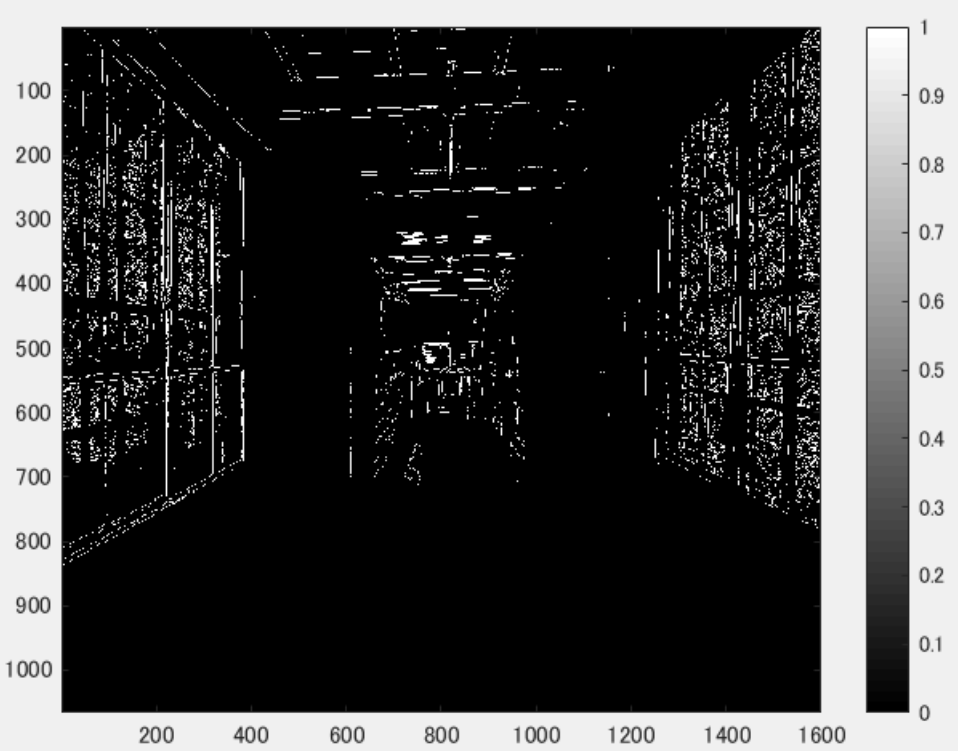

図 26、プレウィット法を用いてエッジ抽出

```
IMG = edge(ORG,'sobel'); %  エッジ抽出（ソベル法）
imagesc(IMG); colormap('gray'); colorbar;%  画像表示
```

によって、ソベル法を用いてエッジ抽出をする。結果を図 27 に示す。

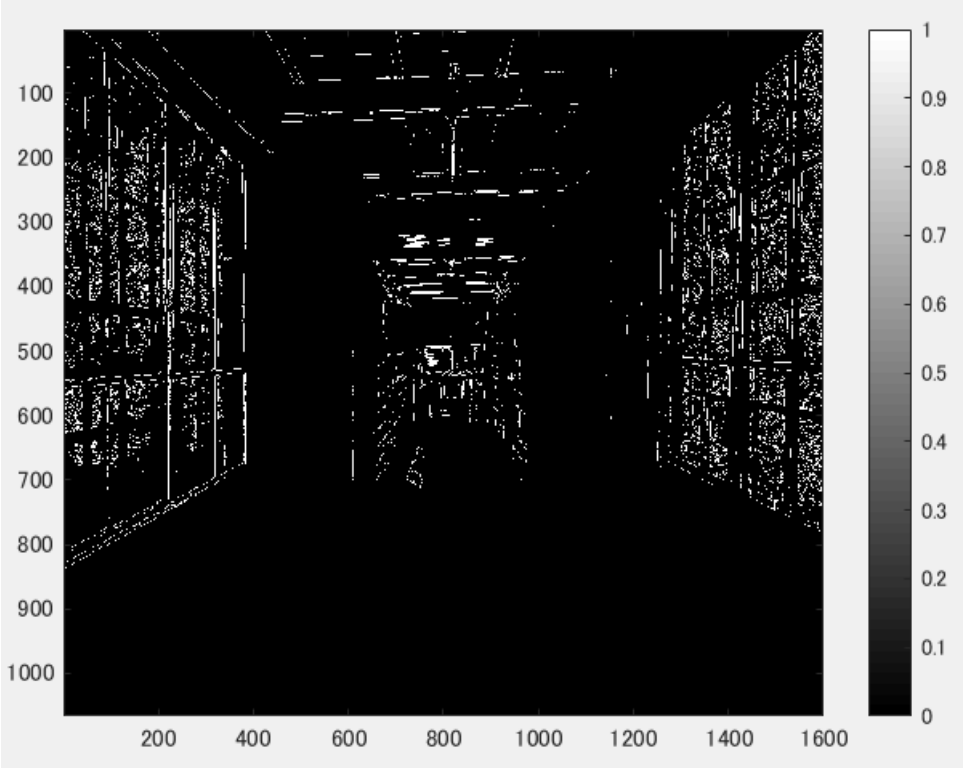

図 27、ソベル法を用いてエッジ抽出

```
IMG = edge(ORG,'canny'); %  エッジ抽出（キャニー法）
imagesc(IMG); colormap('gray'); colorbar;%  画像表示
```

によって、キャニー法を用いてエッジ抽出をする。結果を図 28 に示す。

この結果を比較すると、キャニー法が最もよくエッジ抽出されていると考えられる。

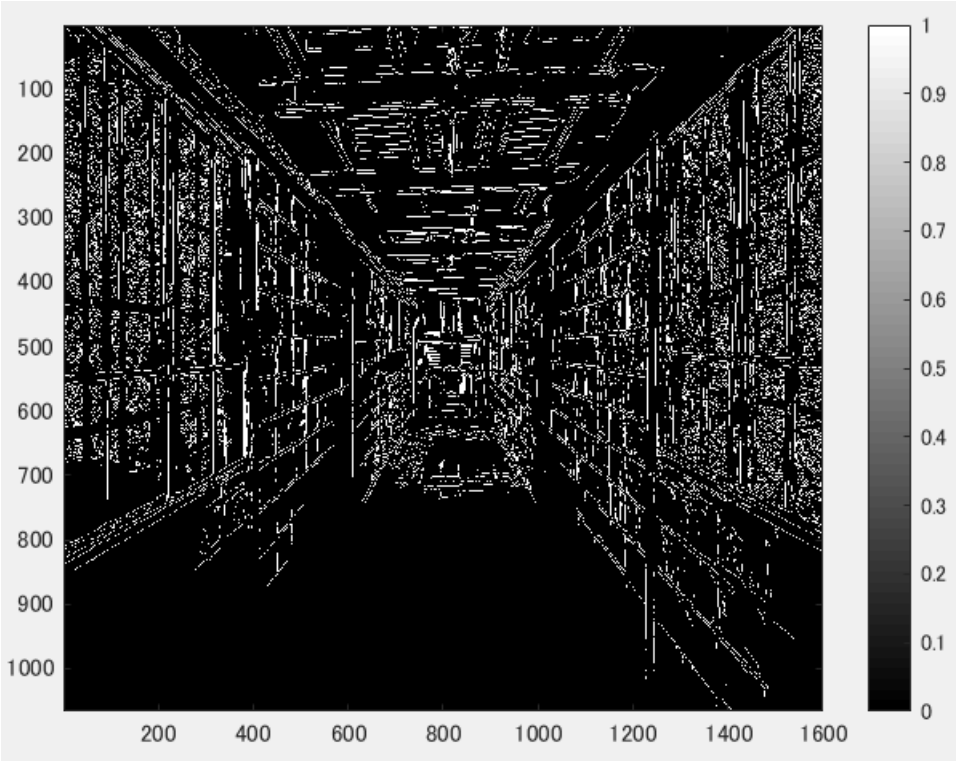

図 28、キャニー法を用いてエッジ抽出