デジタルメディア処理2

担当: 井尻 敬

デジタルメディア処理２、2017（前期）

~~4/13　デジタル画像とは　：イントロダクション~~
~~4/20　フィルタ処理1　：画素ごとの濃淡変換、線形フィルタ，非線形フィルタ~~
~~4/27　フィルタ処理2　：フーリエ変換，ローパスフィルタ，ハイパスフィルタ~~
~~5/11　画像の幾何変換１　：アファイン変換~~
~~5/18　画像の幾何変換２　：画像の補間，イメージモザイキング~~
~~5/25　画像領域分割　：領域拡張法，動的輪郭モデル，グラフカット法，~~
~~6/01　**前半のまとめ (約30分)と中間試験（約70分）**~~
6/08　特徴検出1　：テンプレートマッチング、コーナー・エッジ検出
6/15　特徴検出2　：DoG、SIFT特徴量、ハフ変換
6/22　画像認識1　：パターン認識概論，サポートベクタマシン
6/29　画像認識2　：ニューラルネットワーク、深層学習
7/06　画像処理演習　：ImageJを使った画像処理
7/13　画像処理演習　：Pythonプログラミング
7/20　**後半のまとめ (約30分)と期末試験（約70分）**

特徴検出 と パターン認識

11章 – パターン・図形・特徴の検出とマッチング

画像の中から，特定のパターン，コーナー，直線，円，などの特徴点を検出するアルゴリズムを紹介する

12章 -- パターン認識

既存のデータセットから暮らす分類を学習し，未知画像がどのクラス属すかを推測する手法を紹介する

上記の特徴点を利用し，クラス分類を記述する（手法も多い）

深層学習にも触れる

Contents 画像内の特定パターンを発見する手法

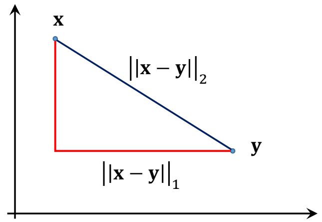
- テンプレートマッチング
- 特徴点検出
  - コーナー検出（Harris corner detector/FAST）
  - エッジ検出（Canny edge detector）
  - その他有名な特徴点（SIFT/BRIEF/ORB）
- 特徴点の対応付け
- Hough変換

準備: ノルム(norm)

d次元空間のベクトル $\mathbf{x} = (x_1, x_2, \ldots, x_n)$の p -ノルムは以下の通り定義される

$$||\mathbf{x}||_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{\frac{1}{p}}$$

例 d=2のとき



p=2なら…
$||\mathbf{x}-\mathbf{y}||_2 = \sqrt{(x_1-y_1)^2 + (x_2-y_2)^2}$
これはよく知っているユークリッド空間の距離

p=1なら…
$||\mathbf{x}-\mathbf{y}||_1 = |x_1-y_1| + |x_2-y_2|$
点**x**から点**y**へ，軸に沿った方向のみで移動した際の距離
市街地における移動距離になぞらえて**市街地距離**や**マンハッタンノルム**と呼ばれる

左の画像から右の画像を探せ

※地味な例ですみません。。。

左の画像から右の画像を探せ

※地味な例ですみません。。。

templateMaching.py

テンプレート マッチング

入力画像

テンプレート
画像

- 入力画像を**ラスタスキャン**し，入力画像と**テンプレート**の**類似度**を比較
- 類似度が閾値より高い部分を出力する

※**テンプレート**：検索対象を表す標準画像
※**ラスタスキャン**：画像を左から右に，上から下に，一画素ずつ走査すること

類似度（相違度）の定義

入力画像
$I(i,j)$

テンプレート
$T(i,j)$

Grayscale化
されている

- 相違度: **S**um of **S**quare **D**istance

$$R_{SSD} = \sum_{i,j} \left(I(i,j) - T(i,j) \right)^2$$

- 相違度: **S**um of **A**bsolute **D**istance

$$R_{SAD} = \sum_{i,j} |I(i,j) - T(i,j)|$$

- 類似度: **N**ormalized **C**ross **C**orrelation(正規化相互相関)

$$R_{NCC} = \frac{\sum_{i,j} I(i,j)T(i,j)}{\sqrt{\sum_{i,j} I(i,j)^2 \sum_{i,j} T(i,j)^2}}$$

templateMaching.py

テンプレートマッチングの結果

テンプレート

入力画像

SAD

$\sum_{i,j} |I(i,j) - T(i,j)|$

SSD

$\sum_{i,j} \left(I(i,j) - T(i,j) \right)^2$

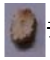
NCC

$\frac{\sum_{i,j} I(i,j)T(i,j)}{\sqrt{\sum_{i,j} I(i,j)^2 \sum_{i,j} T(i,j)^2}}$

SAD/SSDは相違度なので，近いところほど値が小さくなる
NCCは類似度なので近いところほど値が大きくなる
例えば，閾値以下の局所最小部を検出対象とすればよい

templateMaching.py

テンプレートマッチングの結果



テンプレート

入力画像

SAD

$\sum_{i,j} |I(i,j) - T(i,j)|$

SSD

$\sum_{i,j} \left(I(i,j) - T(i,j) \right)^2$

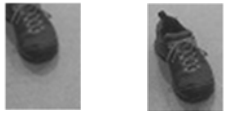
NCC

$\frac{\sum_{i,j} I(i,j)T(i,j)}{\sqrt{\sum_{i,j} I(i,j)^2 \sum_{i,j} T(i,j)^2}}$

SAD/SSDは相違度なので，近いところほど値が小さくなる
NCCは類似度なので近いところほど値が大きくなる
例えば，閾値以下の局所最小部を検出対象とすればよい

類似度・相違度の意味的理解

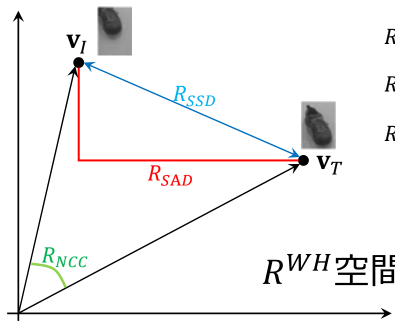
- 入力画像・テンプレートは W x H グレースケール画像
- これを (WH)-次元ベクトルと考える



入力画像
$I(i,j)$

テンプレート
$T(i,j)$

$\mathbf{v}_I, \mathbf{v}_T \in R^{WH}$



R_{SSD}は$\mathbf{v}_I, \mathbf{v}_T$のユークリッド距離

R_{SAD}は$\mathbf{v}_I, \mathbf{v}_T$の市街地距離

R_{NCC}は$\mathbf{v}_I, \mathbf{v}_T$の角度

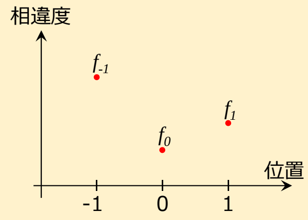
サブピクセル精度のテンプレートマッチング

- テンプレートマッチングは目的画像にテンプレート画像を重ね差分を評価するため発見できる位置は**ピクセル単位（離散値）**
- **サブピクセル（連続値）**精度で位置検出を行いたい

- 局所的に関数をフィッティングし，最小値を求める

→ 等角直線フィッテイング

→ パラボラフィッティング



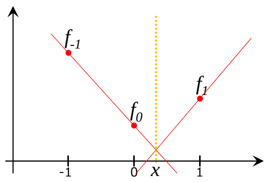
問題

- 相違度が最小の画素を原点(x=0)にとる
- x=±1 の相違度も既知
- 最小値を与える位置xを実数精度で求める

※画像に適用する際は縦横を独立に扱えば良い

等角直線フィッティング

下図の通り傾きが-1倍の2本の直線の交点を利用

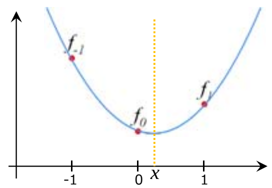


$x =$

$f_{-1} > f_1$ のときは、f_{-1} と f_0 を通る直線と，f_1を通る直線を考える

パラボラフィッティング

二次関数で相違度を補間し相違度の最小位置を求める



$x =$

テンプレートマッチングの高速化

W×H

w×h

対象画像全領域にテンプレートを重ね合わせて差分を計算する計算複雑度は…

残差逐次検定：目標画像をラスタスキャンしテンプレートとの差分計算をする際，現在の最小値よりも差分が大きくなったら計算を打ち切る

粗密探査法：ガウシアンピラミッドを生成し低解像度画像からマッチングを計算．ひとレベル高解像度画像に移動し，発見した画素に関係する数画素のみにたいしてマッチングを計算する

教科書 図11.5

復習: Steepest descent - 最急降下法

最小化問題

関数$f(\mathbf{x})$を最小化する$\mathbf{x}$を求めよ

$$\arg\min_{\mathbf{x}} f(\mathbf{x})$$

※ 関数$f(\mathbf{x})$の形が分かっていて$\nabla f(\mathbf{x}) = \mathbf{0}$が解けるならそれでよいが，そうでない場合に使える手法の一つが最急降下法

最急降下法

1. $\mathbf{x}^0$を初期解とする（何らかの方法で発見する）
2. 変化が十分少なくなるまで以下を繰り返す

$$\mathbf{x}^{t+1} = \mathbf{x}^t - h\nabla f(\mathbf{x})$$

Chamfer Matching

1. 入力画像 I のエッジ画像 $I_E(x,y)$を生成しそれをエッジからの距離画像I^{DT}に変換
2. テンプレート画像Tをエッジ画像に変換$T_E(u,v)$
3. 相違度を以下の通り定義する

$$S(x,y) = \sum_{v=0}^{H}\sum_{u=0}^{H} T_E(u,v) I_{DT}(x+u, y+v)$$

※ エッジ画素の場所で距離画像をサンプリング

※ テンプレート全体を見ないので高速

教科書図 11.7

Chamfer Matching

3. 相違度を以下の通り定義する

$$S(x,y) = \sum_{v=0}^{H}\sum_{u=0}^{W} T_E(u,v) I_{DT}(x+u, y+v)$$

4. 初期位置(x^0, y^0)から最急降下法により相違度が最小となる位置を探索する

$$(x^{t+1}, y^{t+1}) = (x^t, y^t) - \nabla S(x,y)$$

※相違度をエッジ画像の距離場を用いて定義したのでその勾配エッジ同士がフィットする方向を向く

※勾配の式は以下の通り

$$\nabla S(x,y) = \begin{pmatrix} \sum_v \sum_u T_E(u,v) \frac{\partial}{\partial x} I_{DT}(x+u, y+v) \\ \sum_v \sum_u T_E(u,v) \frac{\partial}{\partial y} I_{DT}(x+u, y+v) \end{pmatrix}$$

教科書図 11.7

まとめ：テンプレートマッチング

入力画像

テンプレート

入力画像から物体を検出するための手法

検出対象の画像（テンプレート）を用意し，入力画像をラスタスキャンし相違度を評価

相違度が閾値以下の領域を出力する

相違(類似)度：SAD, SSD, NCCなど

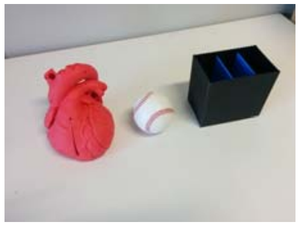
サブピクセル精度で検出するための関数フィッティング

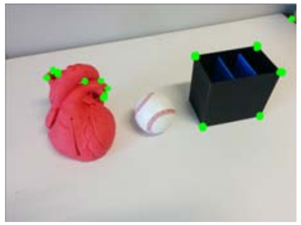
高速化のための残差逐次検定・粗密(coarse to fine)探索・chamfer matching

コーナー、輪郭線の検出

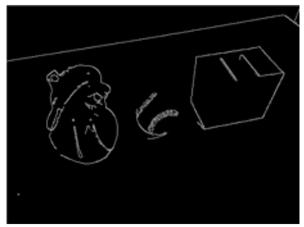
HarrisCorner.py
CannyEdge.py

物体認識・物体追跡・位置あわせなど，より高度な画像処理に利用するため画像から『コーナー』や『輪郭線』といった特徴的な点・曲線を検出する





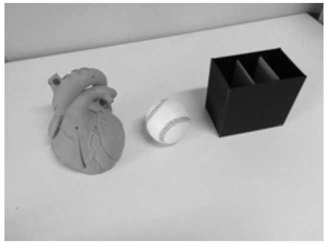
コーナー検出
(Harris Corner Detector)

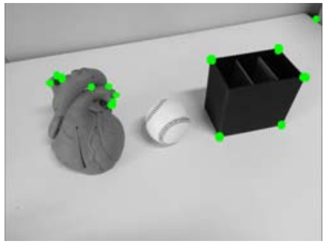


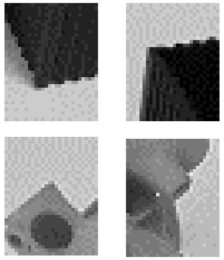
輪郭検出
(Canny Edge detector)

Harrisのコーナー検出アルゴリズム

[C. Harris & M. Stephens (1988). "A Combined Corner and Edge Detector". Proc. of the 4th ALVEY Vision Conference. pp. 147–151.]







- **入力**：グレースケール画像
- **出力**：コーナー画素群
- **手法の概要**

  Harris行列（又はStructure tensor matrixと呼ばれる）を定義し，この固有値固有ベクトルを用いて，局所領域の輝度変化方向と変化量を検出する

  局所領域の輝度変化が，直交する2方向について大きくなる部分をコーナーと定義

Structure tensor matrix (1/3)

画像上の点(*x*,*y*)の輝度値を$I(x,y)$と表す

点(*x*,*y*)における**Structure tensor matrix**は以下の通り定義される

$$\mathbf{A}(x,y) = \sum_{u,v} G(u,v) \begin{pmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{pmatrix}$$

ただし、$I_y = I_y(x+u, y+v)$, $I_x = I_x(x+u, y+v)$ と省略したもの

I_xとI_yは画像の微分（sobel filter）

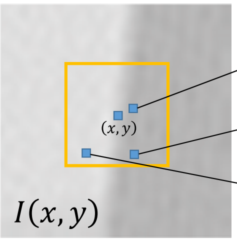
また、$G(u,v)$は重み関数（ガウシアンを用いる）

※教科書の式11.6 ~ 11.9に対応する

Structure tensor matrix (2/3)

実際の計算手順

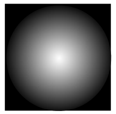
$$\mathbf{A}(x,y) = \sum_{u,v} G(u,v) \begin{pmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{pmatrix}$$



$\begin{pmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{pmatrix}$

$\begin{pmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{pmatrix}$

$\begin{pmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{pmatrix}$



$G(u,v)$

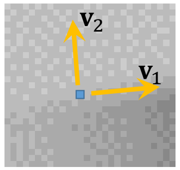
中心からの距離に応じて
重み付けして足し合わせる

Structure Tensorの性質

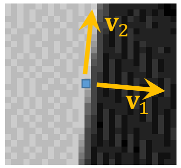
- 固有値をλ_1, λ_2 とする ($\lambda_1 > \lambda_2$)
- 固有ベクトルを $\mathbf{v}_1, \mathbf{v}_2$とする
- 対象行列 → 固有値は実数
- 対象行列 → 固有ベクトルは直交
- 半正定置 → $\lambda_1, \geq 0$, $\lambda_2 \geq 0$
  - 半正定置行列の和なので.

- $\mathbf{v}_1$は輝度値変化の最も大きな方向
- λ_1は$\mathbf{v}_1$方向の輝度値変化の大きさ
- λ_2は$\mathbf{v}_2$方向の輝度値変化の大きさ

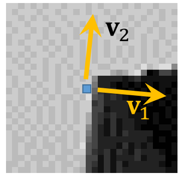
Structure tensor matrix (3/3)



λ_1:小
λ_2:小



λ_1:大
λ_2:小



λ_1:大
λ_2:大

- Structure Tensor Matrixの二つの固有値は，局所領域の輝度値変化の様子に依存して大小が変化する

  (小, 小)　→ 全体的に変化がすくない　：フラット

  (大, 小)　→ ある方向にのみ大きく変化：エッジ

  (大, 大)　→ 2方向に大きく変化　　　：**コーナー**！

Harrisのコーナー検出アルゴリズム

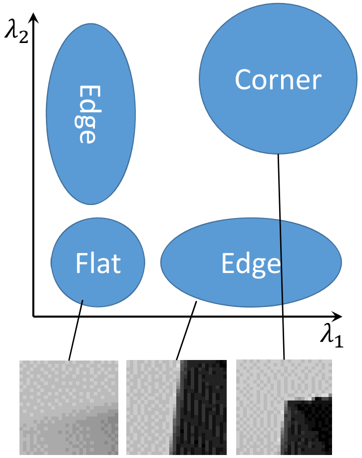
グレースケール画像からコーナーを検出

1. 各画素(x,y)におけるStructure Tensor $\mathbf{A}$ と固有値λ_1, λ_2を計算
2. 各画素(x,y)において$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$を計算
3. Rが極大かつ閾値以上の点をコーナーとして出力する

※ただし，kはユーザが指定するパラメタ (0.04~0.06)

※$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$ は，コーナーらしさを現す関数：
λ_1とλ_2が大きくかつ近いときに大きな値を返す

評価式Rの3Dプロット →
http://www.wolframalpha.com/input/?i=z%3Dx*y+-+0.02*(x%2By)%5E2



Harrisのコーナー検出アルゴリズム

グレースケール画像からコーナーを検出

1. 各画素(x,y)におけるStructure Tensor $\mathbf{A}$ と固有値λ_1, λ_2を計算
2. 各画素(x,y)において$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$を計算
3. Rが極大かつ閾値以上の点をコーナーとして出力する

⇩

グレースケール画像からコーナーを検出 **new**

1. 各画素(x,y)におけるStructure Tensor $\mathbf{A}$ を計算
2. 各画素(x,y)において$R = \det\mathbf{A} - k(\operatorname{tr}\mathbf{A})^2$を計算
3. Rが極大かつ閾値以上の点をコーナーとして出力する

固有値の計算時間が無駄

$\det\mathbf{A} = \lambda_1 \times \lambda_2$

$\operatorname{tr}\mathbf{A} = \lambda_1 + \lambda_2$

という関係を利用すると計算を効率化できる

※練習）上記の関係を証明せよ

Harrisのコーナー検出アルゴリズム（実装例）

Cannyの輪郭線検出アルゴリズム(1/2)

※井尻はキャニーと呼んでますが、教科書はケニーですね。。。

1. ガウシアンフィルタをかける：$I \rightarrow G * I$

例) 5x5, σ＝1.4 のガウシアンなどが利用される

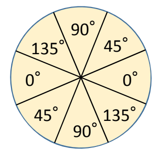
2. 勾配強度・勾配方向計算

Sobel filterにより縦横方向の微分を計算：$I \rightarrow I_x, I_y$

勾配強度：$g(x,y) = \sqrt{I_x(x,y)^2 + I_y(x,y)^2}$

勾配方向：$d(x,y) = \tan^{-1}\frac{I_y(x,y)}{I_x(x,y)}$

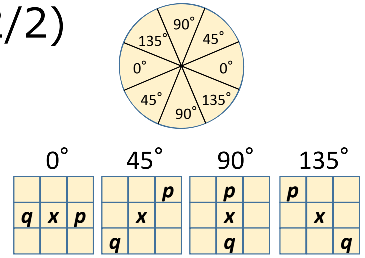
(0°/45°/90°/135°の4通りに量子化)



参考: OpenCV http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

原著論文: Canny, J., *A Computational Approach To Edge Detection*, IEEE PAMI, 1986.

Cannyの輪郭線検出アルゴリズム(2/2)



3. non-maximum suppression

細い輪郭線抽出のため，勾配強度が極大となる画素のみを残す

各画素*x*に対して…

勾配方向に隣接する2画素*p,q*と*x*の勾配強度を比較

画素*x*の勾配強度が*p,q*と比べて最大でないなら*x*の勾配強度を*0*に

4. 閾値処理

二つの閾値T_{max}とT_{min}を用意

画素xの勾配強度が…

- T_{max}より大きい → Strong edge: 画素xは輪郭線である
- T_{min}より小さい → not edge：画素xは輪郭線でない
- それ以外 → week edge: もしstrong edgeに隣接していれば輪郭線とする

※紹介したものは実装の一例です.

Cannyの輪郭線検出アルゴリズム（実装例）

まとめ：コーナー・輪郭検出

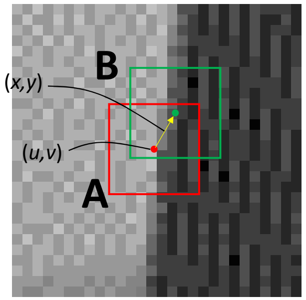
コーナー検出：画像中の『角』形状を検出

- **Harris Corner detection**
  → Structure Tensorの固有値により角らしさを定義
- 様々な手法が知られる(FAST/SUSAN/ヘッセ行列)

輪郭検出：画像中の物体と物体の境界を検出

- **Canny Edge Detection**
  - 微分フィルタによる勾配画像取得
  - 勾配方向を考慮した細線化
  - 二つの閾値処理
- 様々な手法が知られる(Sobel/Hough変換…)

補足資料



Structure Tensor Matrix（導出）

画像上で『点(u,v)を中心とする領域A』と『微少量(x,y)だけ動かした領域B』を考える

領域AとBの差分を，重みを考慮して定義する;

$$D(x,y)=\sum_{x,y}G(x,y)\big(I(u+x,v+y)-I(x,y)\big)^2 \qquad \dots(1)$$

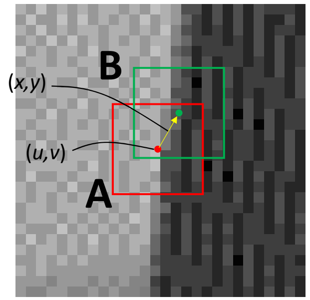
※ 重み関数G(x,y)には，ガウシアンがよく用いられる．

テーラー展開し二次以降の項を無視すると，以下の変形が得られる

$$I(u+x,v+y)\approx I(u,v)+xI_x(u,v)+yI_y(u,v)$$

これを(1)に代入すると, 以下の通りStructure Tensor Matrixが現れる

$$D(x,y)=(x,y)\mathbf{A}\begin{pmatrix}x\\y\end{pmatrix},\qquad \mathbf{A}=\sum_{x,y}G(x,y)\begin{pmatrix}I_xI_x & I_xI_y\\ I_xI_y & I_yI_y\end{pmatrix}$$



Structure Tensor Matrix（導出）

画像上で『点(u,v)を中心とする領域A』と『微少量(x,y)だけ動かした領域B』の差は以下の通り

$$D(x,y)=(x,y)\mathbf{A}\begin{pmatrix}x\\y\end{pmatrix},\qquad \mathbf{A}=\sum_{x,y}G(x,y)\begin{pmatrix}I_xI_x & I_xI_y\\ I_xI_y & I_yI_y\end{pmatrix}$$

今知りたいのは，どの方向(x,y)に動かすと差分が最大になるか？つまり，画像の変化が大きいか？である．そのため以下の最大化問題を考える．

$$argmax\,\frac{(x,y)\mathbf{A}\begin{pmatrix}x\\y\end{pmatrix}}{(x,y)\begin{pmatrix}x\\y\end{pmatrix}}$$

この目的関数はレイリー商と呼ばれ，(x,y)が行列Aの固有ベクトルに一致するとき，最大値（最小値）をとり，最大値・最小値は固有値と一致することが知られている(証明省略).

つまり，Structure Tensor matrixの固有値固有ベクトルをλ_1,λ_2 ($\lambda_1>\lambda_2$) $\mathbf{v}_1,\mathbf{v}_2$とすると，(x,y)が$\mathbf{v}_1$に一致するときに画像は最も大きく変化する．また(x,y)が$\mathbf{v}_2$に一致するとき画像の変化は最小になる．