コンピュータビジョン課題

05242628 三田村彰大

June 12, 2026

概要　SIFT を用いた対応点探索を高水準 API を使わずに実装した。具体的には DoG フィルタを用いた特徴点の検出・SIFT descriptor を用いた局所特徴の記述・最近傍探索 +Ratio Test を用いた対応点探索を実装した。その上で、SIFT descriptor が回転やスケール変化、照明の変化に対して頑健であることを実際の画像に対して対応点探索を行うことで確認した。また、DoG が特徴点の検出において万能でないことや、特徴のスケールが大きくなると計算量が大きくなること、似た局所パターンが複数箇所に現れるとミスマッチが生じやすくなることなどが分かった。

※ コード置き場：https://github.com/ramutami/computervision

※AI の利用について

今回対応点探索を実装するにあたって、コーディングにふんだんに ChatGPT を利用した。具体的には Lowe 2004 [2] の論文をコードに落とし込む方法の提案や、書いたコードのデバッグ等に利用した。[*1]

またレポートの執筆に際し誤字脱字や内容の誤りの検出に ChatGPT を利用している。

1　背景

はじめに対応点探索と SIFT (Scale-Invariant Feature Transform) の概要についてまとめる。(参考 [1])

対応点探索とは異なる視点から撮影された同一の物体の画像において物理的に同一の点をマッチングさせる技術であり、その技術は物体認識や画像マッチング、パノラマ合成などに用いられている。(Figure1参照。) 対応点探索は具体的には

1. 画像上の特徴点の検出
2. 特徴の記述
3. 二つの画像の特徴点のマッチング

といった手順によって実現される。

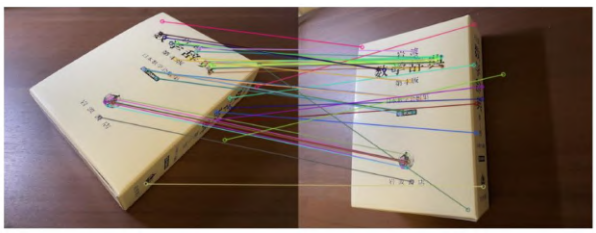

Figure1: 対応点探索。授業スライドより。

その上で、SIFT (Scale-Invariant Feature Transform) とは Lowe によって提案された特徴点の記述の手法の一つである。まずはじめ DoG フィルタによって特徴点を検出し、検出された特徴点を「特徴点周辺の画像輝度の勾配の分布」によって記述する。これは 128 次元ベクトルとして記述され、このベクトルの距離が最小となる点をマッチングさせることで対応点探索が実現される。

この SIFT においては特徴の記述の際、

- 代表的な輝度勾配の向きを基準に特徴点付近の画像を回転変換させる
- 特徴スケールを用いて特徴点付近の画像を縮小・拡大する
- 特徴ベクトルを正規化する

といった手順を踏むことによって「回転」「スケール変化」「照明変化」に頑健な対応点探索を実現している。

今回自分は、この一連の対応点探索を一から実装した。具体的には DoG フィルタを用いた特徴点の検出・SIFT descriptor を用いた局所特徴の記述・最近傍探索 +Ratio Test を Python を用いて実装した。またその際 SciPy の gaussian_filter 関数や maximum_filter 関数を利用したが、それ以外の Dog フィルタや特徴量記述、最近傍探索は既存の高水準 API を用いることなく実装した。[*2]

2　解法

以降においては Figure6で示されるような 2 枚の画像の対応点探索を例に考えることにする。

2.1　DoG フィルタを用いた特徴点の検出

画像を I で表す。このとき、大きさ $k^i \times \sigma_0$ の Gaussian フィルタ

$$G(k\sigma_0), G(k^2\sigma_0), G(k^3\sigma_0) \dots \tag{1}$$

に対しそれらによるフィルタリング

$$G(k\sigma_0) * I, G(k^2\sigma_0) * I, G(k^3\sigma_0) * I \dots \tag{2}$$

を考える。この時、あるスケール $k^n\sigma_0$ において (x, y) に特徴点があるならば、$G(k^n\sigma_0) * I - G(k^{n\pm1}\sigma_0) * I$ の極値として現れるはずである。よって畳み込みの線形性から、

[*1] 手法としてはややバイブコーディングに近いが、実際のコードは全て理解した上で自分で書いている。

[*2] ただし AI の利用については冒頭を参照。

$$\mathrm{DoG}(k^n\sigma_0) = G(k^{n+1}\sigma_0) - G(k^n\sigma_0) \tag{3}$$

というフィルタでフィルタリングを行えばよい[*3]。(今 Figure6左の画像に対し Gaussian フィルタ及び DoG フィルタをかけたものを Figure7及び Figure8に示した。)

その上で、DoG フィルタの出力が極値を取る位置とスケールを特徴点として採用する。具体的には DoG フィルタの出力を σ 方向に stack したときに注目画素 + その 26 近傍を探索し極値となっている点 (x, y, σ) を特徴点として検出した[*4]。(Figure2参照。)

また、Figure6の二つの画像に対して特徴点の検出を行ったものを Figure9に示す。

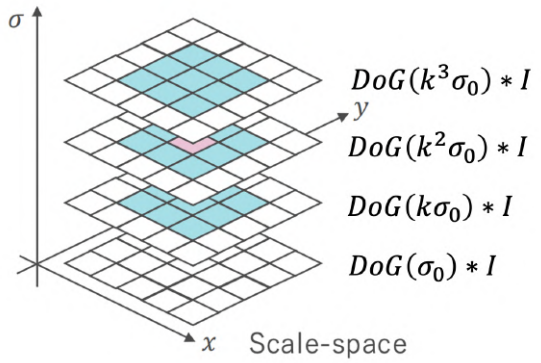

Figure2: DoG フィルタによる特徴点検出。授業スライドの画像を元に作成。

2.2 SIFT を用いた局所特徴の記述

特徴点が検出できたら次は特徴を記述する。今回は授業スライド及び Lowe 2004 [2] を参考に、SIFT descriptor の標準的な手法に従って局所領域内の輝度勾配強度の分布を (x, y, θ) の 3 変数についてヒストグラムで表現することでこれを行った。具体的には、局所領域内を空間 4×4 分割し、各空間上の点において輝度勾配の角度を輝度勾配強度・注目画素からの距離によって重み付けした上で 360° を 8 分割したヒストグラムに投票し、得られる $4 \times 4 \times 8 = 128$ 次元ベクトルを特徴ベクトル (SIFT descriptor) とした。

2.2.1 計算の流れ

今画像座標を X, Y で表すことにする。この時まず準備として、各スケール σ での平滑化画像 $L^\sigma(X, Y)$ に対し、

$$\begin{aligned} m(X,Y) &= ||\nabla L^\sigma(X,Y)||_2 \\ \theta(X,Y) &= \arg\left(\nabla L^\sigma(X,Y)\right) \end{aligned} \tag{4}$$

を計算しておく。

その上で、座標 X_0, Y_0 で検出された特徴を記述することを考える。この時、将来的に SIFT descriptor が回転に対して頑健であるためには特徴点付近の局所領域を代表的な方向 θ_0 を基準に回転させておく必要がある。

そこで、特徴点のスケール σ[*5]による平滑化画像 $L^\sigma(X, Y)$ を元に計算された輝度勾配から (X_0, Y_0) 近傍での代表的な輝度勾配の方向を算出し、それを θ_0 として採用することにする。Lowe 2004 によればこれは 2π を 36 分割したヒストグラムに対し $\theta(X, Y)$ を大きさ 1.5σ の Gaussian 重み及び輝度勾配の大きさ m によって重み付けした上で投票することによって決定される。

今、角度方向の離散化を θ_k で表し、投票に関与する特徴点近傍領域を Ω とする時、実際の実装においては投票を次のように行った[*6][*7][*8]。

$$\begin{aligned} v(k) = \sum_{(X,Y)\in\Omega} & G\left(X - X_0, Y - Y_0\,; 1.5\sigma\right) m\left(X,Y\right) \\ & \times 1_{\left[\frac{2\pi k}{36}, \frac{2\pi(k+1)}{36}\right)}\left(\theta\left(X,Y\right)\right) \end{aligned} \tag{5}$$

その上で、Lowe 2004 に

> "The highest peak in the histogram is detected, and then any other local peak that is within 80% of the highest peak is used to also create a keypoint with that orientation."

とあることを踏まえ、最も得票数の多かった角度と最大投票数に対して 80% 以上の得票数を集めた角度をその特徴点の代表的角度 θ_0 として採用した。(すなわち一つの特徴点に対し複数の方向 θ_0 が与えられることも起こりうる。)

さて、代表的方向が決定できたら次は特徴点近傍で輝度勾配の投票を行う。Lowe 2004 の論文で

> "In order to achieve orientation invariance, the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation"

とあるように、そのためには特徴点近傍で θ_0 が主軸となるよう座標系を回転させ、さらに $\theta \rightsquigarrow \theta - \theta_0$ のような変換を行う必要がある。また最終的には正規化された座標系 (授業のような正規化を考え、$x, y = -1.5, -0.5, 0.5, 1.5$ が bin 中心となるようにする) で考えたいので、座標のスケールも変換する必要がある。よって、

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{s} R(-\theta_0) \begin{pmatrix} X - X_0 \\ Y - Y_0 \end{pmatrix} \tag{6}$$

のように変換し、輝度勾配についても

$$\begin{aligned} m\left(x,y\right) &= m\left(X,Y\right) \\ \theta(x,y) &= \theta(X,Y) - \theta_0 \end{aligned} \tag{7}$$

のように変換しなければならない[*9]。(Figure3参照。)

ただし座標変換におけるスケール変換の大きさ s については、例えば授業スライドのような一つの空間方向の bin あたりの画素数が 4×4 となるような場合を考えると (Figure4参照)、$s = 4$ と必然的に定まる。その上で、SIFT descriptor がスケール変換に対しても頑健であるためにはは空間方向の bin あたりの画素数も特徴量のスケールに応じて変化させるべきであると思われ、今回は $s = 4\sigma$ とした。

[*3] ただし、σ_0 や k の大きさについては参考になるようなものがなかったので、今回は AI に相談した上で $\sigma_0 = 1\mathrm{pix}, k = 1.5$ と定めた。

[*4] ただし、極値を与えたとしたとしてもその値が閾値を超えないようなものは無視する処理を行った。この閾値の与え方については Lowe を参考に 0.03 と決定した。

[*5] 今回は $\mathrm{DoG}(k^n\sigma_0) = G(k^{n+1}\sigma_0) - G(k^n\sigma_0)$ で検出された特徴点のスケールをそのまま $\sigma = k^n\sigma_0$ としている。

[*6] ただし $1_A(x)$ は $x \in A$ の時に 1 を返す定義関数である。

[*7] Ω は十分大きな領域として $r = 3 * 1.5\sigma$、中心 X_0, Y_0 の円形領域を採用した。

[*8] ただし、実際の計算では分散投票を行っている。すなわち今 $\theta(X,Y) \in \left[\frac{2\pi k}{36}, \frac{2\pi(k+1)}{36}\right)$ となっている時、$\theta(X,Y) - \frac{2\pi k}{36}$ の距離で重み付けした上で $k, k+1$ の bin にそれぞれ投票している。この手法は (少なくとも自分が読んだ範囲では) Lowe 2004 には記載されていないが、AI の提案があったので実装した。

[*9] ここら辺は論文の行間を読んだり AI と相談したりしながら考えた。

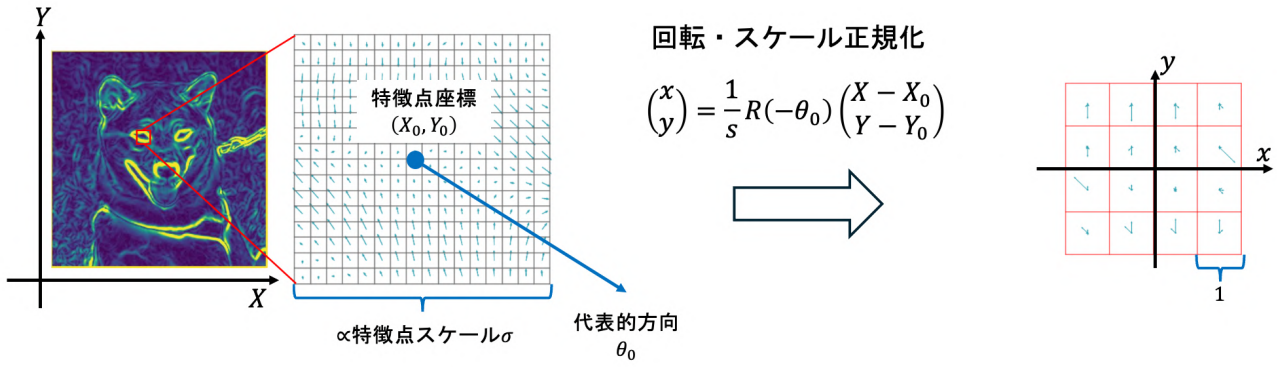

Figure3: 座標変換の模式図。授業スライドを元に作成。

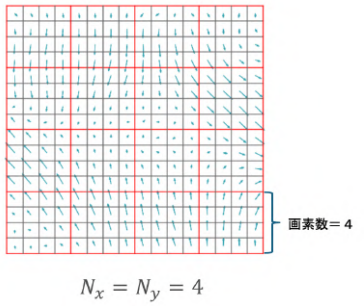

Figure4: 授業スライドより抜粋。一つの空間方向の bin あたりの画素数が 4×4 となっている。

こうして、正規化された座標系 (x, y) への変換さえ用意できれば、あとは授業で示された投票式に従って

$$v(i,j,k) = \sum_{(x,y)\in\mathbb{R}} G(x,y,\sigma_w) m(x,y) \times w_p(x - x_i) w_p(y - y_j) w_a(\theta(x,y) - \theta_k) \tag{8}$$

と投票すれば良い。ただしこのとき、$G(x, y, \sigma_w)$ の σ_w の取り方については Lowe 2004 が descriptor window の半分としているので、今回の場合 $\sigma_w = 2$ と取ることになる。

最後に、SIFT descriptor に照明変化に対する頑健性を持たせることを考える。これは descriptor ベクトルを正規化することで実現できると考えられるが Lowe 2004 はさらに非線形な照明変化への対策として、

> ”Therefore, we reduce the influence of large gradient magnitudes by thresholding the values in the unit feature vector to each be no larger than 0.2, and then renormalizing to unit length. ”

と述べている。よって照明変化の影響をなくすために先の投票で得られたヒストグラム（128 次元ベクトル）$\boldsymbol{d}$ を（一度正規化した上で）0.2 でクリッピングする。

2.3 最近傍探索 +Ratio Test を用いた対応点探索

二つの画像について 128 次元 descriptor$\boldsymbol{d}_i^1, \boldsymbol{d}_j^2$ 得られているとする。この時、最近傍探索においては各 $\boldsymbol{d}_i$ に対し $||\boldsymbol{d}_i^1 - \boldsymbol{d}_j^2||$ が最小となるような $\boldsymbol{d}_j$ をマッチングする。またその際、第二近傍も探索することで最近傍に対し Ratio test を行う。すなわち、$\mathrm{dist}(i,j) = ||\boldsymbol{d}_i^1 - \boldsymbol{d}_j^2||$ を最小とする $j = j_1$ と次点で最小値を与える $j = j_2$ を探し、

$$\frac{\mathrm{dist}(i, j_1)}{\mathrm{dist}(i, j_2)} < r \tag{9}$$

となったものをマッチングペアとして採用することにする。ただし Ratio test のパラメタ r についてはマッチングの挙動を見ながら適宜調整した。また、Figure6の 2 枚の画像に対し対応点探索を行った結果を Figure10に示す。

3 実験設定

今回、実装した対応点探索の性能を試すため、「スケール」「角度」「照明」を変化させた時のマッチング精度を観察することにする。

そのために、Figure5で示されているような被写体を

- 斜め横から撮影
- 遠くから撮影
- 照明を暗くして撮影

した上で、それぞれ正面から撮影した基準となる画像と対応店探索を行う。

Figure5: 今回の実験で用いる被写体。IKEA で 2 年ほど前に購入した。[*10]

[*10] 汚くて申し訳ないです。（ちょうど手元にあったので…）

Figure6: 同一の物体を別の角度から撮った 2 枚の画像。本郷キャンパス総合図書館にて撮影。

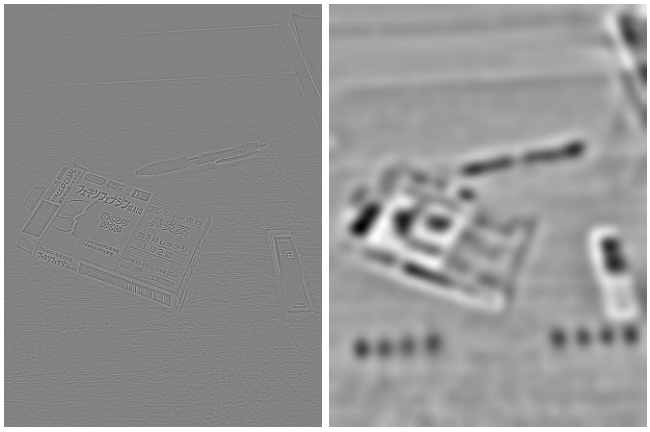

Figure7: 左図：Figure8下段の左から 4 番目の画像を拡大した図。右図：Figure8下段の一番右の画像を拡大した図。左はスケールの小さい DoG フィルタをかけた画像であり、シャープなエッジ・コーナーが捉えられていることがわかる。また右はスケールの大きい DoG フィルタをかけた画像であり、画像の大きな構造が捉えられていることがわかる。

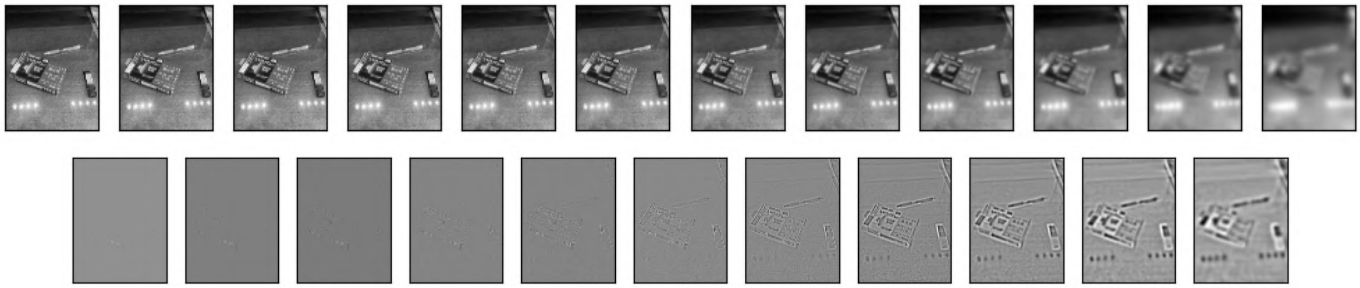

Figure8: Figure6左の画像に様々なスケールで gaussian フィルタをかけたもの（上図）と、その差分として得られる DoG フィルタ（下図）

Figure9: Figure6の二つの画像に対し DoG フィルタによる特徴点検出を行った図。

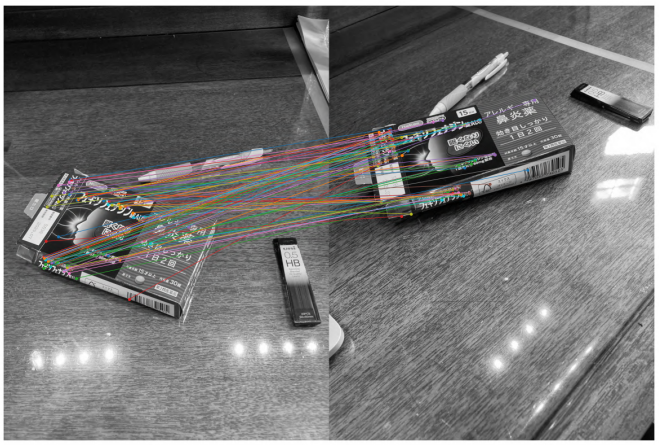

Figure10: Figure6の二つの画像の対応点探索の結果

4　実験結果

Figure11: 異なる距離から撮影した二つの画像の対応点探索の結果

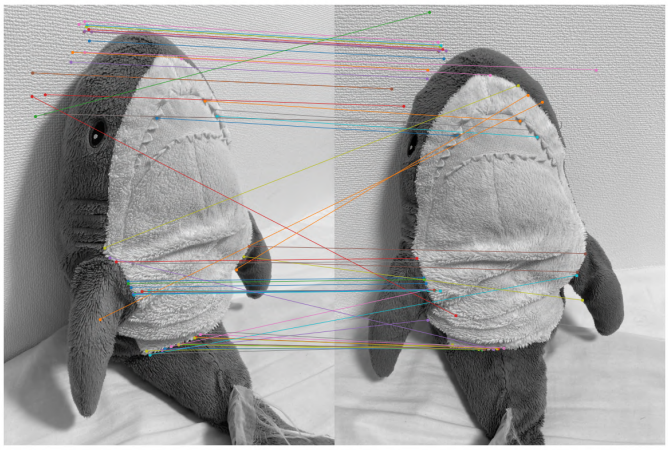

Figure12: 異なる角度から撮影した二つの画像の対応点探索の結果

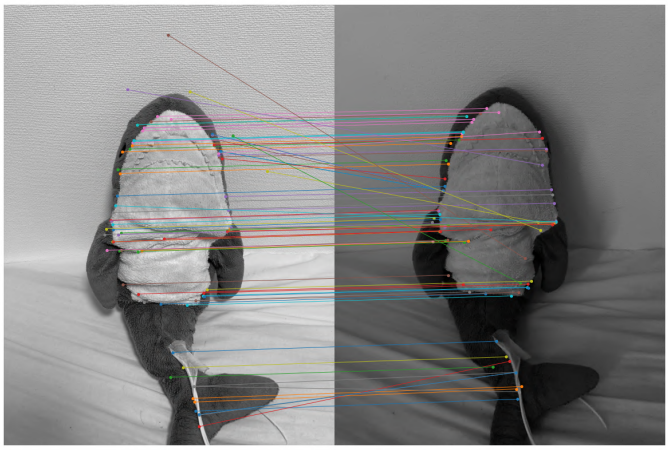

Figure13: 異なる明るさで撮影した二つの画像の対応点探索の結果

5　分析・考察

以下では実験で得られた Figure11〜Figure13の画像を観察しながら、DoG や SIFT などの性質について考察していく。

5.1　DoG による特徴点検出の難点

Figure14は今回の被写体に対して特徴点検出を行った結果を示している。この図を見ると、サメの肌の色が変わるエッジは非常によく捉えられていることがわかるが、サメの体全体の輪郭はあまり捉えられていないことがわかる。(例えばヒレの形など。)

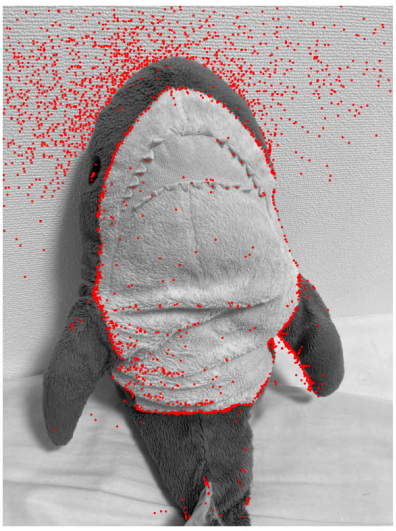

Figure14: 特徴点検出の結果

このような「シャープな特徴は捉えられるが物体の輪郭を捉え損ねる」といった現象は Figure9でも見られた現象であるが、これは検出される特徴点が膨大になりすぎないようフィルタリングしている影響であると考えられる。

すなわち、今回は画像に DoG フィルタをかけた後で極値として残った点を特徴点として検出しているわけであるが、そのままでは膨大な数の特徴点が検出されてしまい計算が重くなってしまう。そのため、仮に DoG フィルタ後の画像で極値となっていたとしても、その輝度の絶対値が小さければ特徴点として算出しないといった処理を行った。[*11]

これにより、画像内にシャープな特徴（今回でいうとサメの肌の色が変わるエッジ）が多数見られた場合検出される特徴点がそういった特徴に偏ってしまうため、あまりシャープではなく DoG フィルタ後の値が小さく出てしまうような特徴（今回でいうとサメの体の輪郭など）の検出が困難になってしまうと推察される。

さらにいうと、Figure14においては壁面の凸凹も特徴点として検出してしまっているが、これも同様の原因によって生じているであろうと推察できる。よって、DoG を用いた特徴点検出を行う際は、背景にシャープな特徴があると検出される特徴点がそちらに集まってしまい本来検出したい物体の特徴点を検出できない可能性がある。

また、Figure15には照明の明るさを変えた際に DoG フィルタにより検出される特徴点が変化する様子を示している。

今 Figure15右の画像（照明が暗い場合）を見ると、Figure15左の画像（照明が明るい場合）で特徴点として検出されていた壁の凹凸が照明が暗い場合では捉えられていないことがわかり、結果として対応点探索においてミスマッチが生じてしまっている。(Figure13参照。) これもまた、今回 DoG フィルタによる特徴点検出をする際に検出される特徴点の数を抑える処理をしてしまったことが原因だと

[*11] またその上でさらに、輝度の大きさについて上位 n 点のみを特徴点として検出する処理をしている。

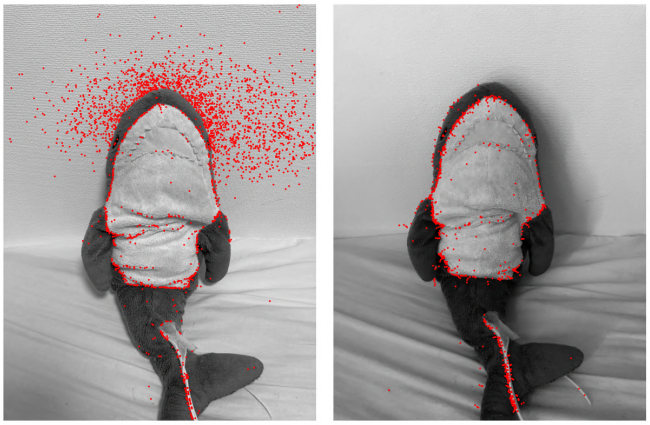

Figure15: 明るさを変えた二つの画像における特徴点検出の結果の違い。

思われる。[*12]

5.2 特徴点のスケールが大きい場合における SIFT descriptor の計算時間の問題

今回の実験では、Figure6の文房具[*13] の画像の SIFT 記述子を計算する為に要した時間が 5 分前後であったのに対し、Figure5のサメの画像の SIFT 記述子を計算するのに要した時間は 30 分程度であった。今回このような違いが生じた原因としては、特徴点のスケールが悪さをしているのではないかと考えた。

今改めて特徴点の代表的方向 θ_0 を計算する投票式を思い出すと、それは次のように書き表されるのであった。

$$v(k) = \sum_{(X,Y)\in\Omega} G\left(X - X_0, Y - Y_0\,; 1.5\sigma\right) m\left(X, Y\right) \times 1_{\left[\frac{2\pi k}{36}, \frac{2\pi(k+1)}{36}\right)}\left(\theta\left(X, Y\right)\right) \tag{10}$$

特に今、Ω は特徴点を中心とし、そのスケール σ に比例した広がりを持つ領域である。

また、最終的に 128 次元の記述子を得るための投票式を思い出すと、それは次のように書き表されるのであった。

$$v(i,j,k) = \sum_{(x,y)\in\mathbb{R}} G(x,y,\sigma_w) m(x,y) \times w_p(x - x_i) w_p(y - y_j) w_a(\theta(x,y) - \theta_k) \tag{11}$$

この時、式では便宜的に投票のために用いる画像領域を $\mathbb{R}$ としているが、実際に計算する上では先と同様に特徴点を中心とし、そのスケール σ に比例した広がりを持つ領域内から投票を行うことになる。[*14]

よって特徴のスケール σ が大きいと、記述子の作成のための投票が非常に広い画像範囲を参照することになり、計算が重くなってしまうことが予想される。このような考察から、今回サメの画像の記述子の作成が文房具の画像の記述子の作成より時間がかかったのは「サメの画像においてスケールの大きい特徴量が相対的に多く見つかったため」と考えることができる。すなわち、SIFT 記述子はスケールの大きな特徴を捉えることを（計算量的な意味で）苦手としている可能性がある。

5.3 似た特徴が異なる場所に存在することによるミスマッチ

Figure11〜Figure13や、Figure10のマッチングの結果を見ると、今回の手法は「画像の異なる箇所に似た特徴が現れる時ミスマッチが生じやすい」ということがわかる。これは特に同じ文字列が現れる Figure10の場合において顕著であり、Figure10を見ると、「フェキソフェナジン錠 ALG」という文字列が薬の箱のあちらこちらに書かれてしまっているせいでこれらの文字のマッチングが混戦してしまっている。

このような問題は、SIFT 記述子が輝度勾配という局所的な情報のみを用いて特徴を記述しているからであり、この問題を回避するには他の特徴との位置関係等を記述子に織り込むような工夫が必要になると思われる。

5.4 SIFT 記述子のスケール・角度・照明の変化に対する頑健性

Figure11、Figure12、Figure13はそれぞれ、「被写体からの距離」「被写体の角度」「照明の明るさ」を変化させた場合の対応点探索の結果を示している。まず Figure11を見ると、二つの画像で特徴は非常によくマッチングしており SIFT 記述子がスケール変化に対して非常に頑健であることが見て取れる。

その上で Figure12を見ると、画像が回転しているにも関わらずある程度特徴をマッチングできているが、同時にミスマッチもそれなりに生じてしまっていることがわかる。特に今回はサメの皮膚の色が変化する箇所でミスマッチが起きてしまっており、これは先に述べた「似た特徴が異なる場所に存在することによるミスマッチ」が画像を回転させたことによって特に強く出てしまっている構造になっている。

またスケールを変化させた場合に比べても、回転している場合はそもそものマッチングの数が少なくなっていることが分かる。今回の実装においてマッチングの数を減らす働きをしているのは Ratio-Test のみであることを考えると、これは単純に特徴ベクトルの最近傍の精度が悪くなっていることを意味し、「画像の回転はスケール変化に比べて特徴ベクトルの変化が大きい」と読み取ることができる。

最後に照明の明るさを変えた Figure13について考察する。この場合は Figure11のスケールを変えた場合と同様非常によく特徴がマッチングしており、ミスマッチが生じているのは主に明るい時に検出された壁の凹凸が暗い時に検出されてないことによる。これは先に述べたように SIFT の問題ではなく DoG で特徴点を検出する際に一定の閾値を設けていることに起因している。

6 まとめ

今回は SIFT を用いた特徴点探索を実装した。その結果、SIFT 記述子がスケール変化や回転、照明の変化に対して頑健であることがわかった。特に、実装を通じてそれぞれが「特徴のスケール・代表的方向を基準に記述子を作成する」「記述子を正規化する」といったことにより達成されていることがわかった。

また、SIFT 記述子はスケール変化や照明の変化に対しては非常に頑健であるが、回転に対しては若干精度が落ちることがわかった。

[*12] このような処理をしないと、DoG フィルターをかけた後に極値となっている点が特徴点として大量に検出されてしまいその後の計算が重くなってしまうため、この問題を避けることは難しいように思われる。

[*13] 文房具じゃないものも混じってますが…

[*14] 先に述べたように、投票を行う前の座標のスケール変換を決めるパラメタ s は $s = 4\sigma$ のように特徴量のスケール σ に依存しているため、投票に参加する画素は特徴量のスケールが大きいほど多くなる。

考察としては、DoG フィルタが特徴点の検出において万能ではなくシャープではない特徴を見落としてしまうことが分かったが、それは検出される特徴点の量を抑え計算量を削減することとトレードオフの関係にあると分かった。また、スケールが大きくなると SIFT 記述子の計算が重くなることがわかり、それは投票に関わる画素数が多くなることが原因であろうと推察された。他にも、同じ文字などの似た特徴が複数箇所に現れるとミスマッチが生じやすいことが分かった。

7 参考文献

[1] 電子情報通信学会誌　Vol.106 No.12 pp.1078-1081 2023 年 12 月 https://app.journal.ieice.org/trial/106_12/k106_12_1078/index.html

[2] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. doi: 10.1023/B:VISI.0000029664.99615.94