

KCF Tracker による高速物体追跡*

内海 佑麻[†] (61602452)

2019/07/23

概要

モダンな物体追跡手法として, KCF Tracker (Kernelized Correlation Filter Tracker) を紹介し, 実装をおこなった. KCF Tracker は, データ行列に巡回性を持たせることにより, DCT による対角化が可能となり, 処理にかかるストレージおよび計算量を大幅に削減することができる. さらに, カーネルトリックにより

目次

1	はじめに	1
1.1	Optical flow	1
2	理論	3
2.1	Ridge 回帰	3
2.2	巡回行列と DFT	3
2.3	巡回行列を使った Ridge 回帰	4
2.4	Kernel Ridge 回帰	5
2.5	巡回行列を使った Kernel Ridge 回帰	6
3	実装	6
4	考察	8
5	結論	8

1 はじめに

1.1 Optical flow

異なる時点の 2 画像間で, 各画素値の移動量を表現したベクトルを Optical Flow という. Optical Flow により特徴点を検出することで, たとえば動画内の特定の物体を追跡す

*慶應義塾大学 理工学研究科 2019 春学期 "Computer Vision 特論" 最終レポート

[†]情報工学科 4 年, Email: uchiumi@ailab.ics.keio.ac.jp

ることができる。まず、一般論として、勾配法 (Gradient-based method) による Optical Flow の求解手順を定式化する。勾配法では、Taylor 展開による近似を用いるため、「連続する 2 画像での対象物の移動が微小であること」を前提としている。

勾配法による Optical Flow の求解 画像 I における画素 (x, y) の時刻 t における画素値を $I(x, y, t)$ 、時間 Δt に対する画素値の移動量を $(\Delta x, \Delta y)$ とおく。移動前後で対象画素の画素値が不変であると仮定すると、

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, \Delta t) \quad (1)$$

が成り立つ。さらに、画素値の変化が滑らかであると仮定すると、1 次までの Taylor 展開により、以下の近似式を得る。

$$I(x, y, t) \approx I(x, y, t) + \frac{\partial I}{\partial x} \Delta x + \frac{\partial I}{\partial y} \Delta y + \frac{\partial I}{\partial t} \Delta t \quad (2)$$

両辺を Δt で割って、整理すると、

$$\frac{\partial I}{\partial x} \frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y} \frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t} = 0 \quad (3)$$

$\Delta t \rightarrow 0$ とすれば、

$$\frac{\partial I}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial I}{\partial y} \frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0 \quad (4)$$

ここで、画素 (x, y) の Optical Flow: $(u, v) = \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t} \right)$ に着目する。 $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$, $I_t = \frac{\partial I}{\partial t}$ 、とおけば、 (u, v) がみたすべき方程式¹は、次式のようにになる。

$$I_x u + I_y v + I_t = 0 \quad (5)$$

しかし、上式は 2 つの未知数 u, v に対して方程式は 1 つであり、解が定まらない²。[1] そのため、上式に加えて、いくつかの仮定 (制約条件) を導入して Optical Flow: (u, v) を球解する手法が提案されている。古典的で重要な手法として、Lucas-Kanade 法や Horn-Schunck 法がある。[2][3][4][5]

- Lucas-Kanade 法

「各画素の近傍では、移動方向が相関する」という仮定をおく。画像中の特定の点 (領域) に絞って追跡を行うような用途に適しているため、sparse 型と呼ばれる。

- Horn-Schunck 法

「各画素は、変化率が最小となる方向へ移動する」という仮定をおく。画像中の画素全体の動きを解析するような用途に適しているため、dense 型と呼ばれる。

¹これを、OpticalFlow の拘束式と呼ぶ。

²これを、Aperture Problem(窓問題) という。

2 理論

Kernelized Correlation Filters[6] について解説する.

2.1 Ridge 回帰

SVM などの洗練されたモデルに近い性能をもち, かつ単純な閉形式解³をもつ Ridge 回帰を用いる. n 個の訓練データ $(\mathbf{x}_i, y_i)_{i=1 \dots n}$ に対する Ridge 回帰は, 以下のように定式化される.

$$\min_{\mathbf{w}} \sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2 + \lambda \|\mathbf{w}\|^2, \quad f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i \quad (6)$$

データ行列

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \\ \vdots \\ x_N^T \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nn} \end{bmatrix} \quad (7)$$

を用いると, Ridge 回帰の閉形式解は, 以下ようになる.

$$\mathbf{w} = (X^T X + \lambda I)^{-1} X^T \mathbf{y} \quad (8)$$

さらに, 転置行列 X^T を, エルミート転置 $X^H := (X^*)^T$ によって拡張すると,

$$\mathbf{w} = (X^H X + \lambda I)^{-1} X^H \mathbf{y} \quad (9)$$

となる.⁴

2.2 巡回行列と DFT

注目点の移動パスをベクトル $\mathbf{x} = (x_1, \dots, x_n)^T \in \mathbb{R}^n$ とおく. このとき, cyclic shift operator として, $n \times n$ 行列

$$P = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \end{bmatrix} \quad (10)$$

³四則演算と初等関数の合成関数によって表せる解を, 閉形式解 (closed-form solution) という.

⁴ A^* は, 行列 A の複素共役を表す.

を定義すると、ベクトル \mathbf{x} の巡回置換は、

$$\{P^u \mathbf{x} \mid u = 0, \dots, n-1\} \quad (11)$$

と得られる。すなわち、あるベクトル \mathbf{x} から、 n 個の仮想サンプルを作成することができる。そこで、ある信号 \mathbf{x} を、 P を用いて拡張すると、

$$C(\mathbf{x}) = \begin{bmatrix} (P^0 \mathbf{x})^T \\ (P^1 \mathbf{x})^T \\ (P^2 \mathbf{x})^T \\ \vdots \\ (P^{n-1} \mathbf{x})^T \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_n & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \cdots & x_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \cdots & x_1 \end{bmatrix} \quad (12)$$

を得られる。ここで、 $C(\mathbf{x})$ は巡回行列 (Circulant matrix) だから、これは DFT (Discrete Fourier Transform) によって対角化可能である。すなわち、

$$\forall \mathbf{z} \in \mathbb{R}^n, \quad F(\mathbf{z}) = \sqrt{n} F \mathbf{z} \quad (13)$$

をみたす、ある定まった DFT 行列 F と、 \mathbf{x} の周波数成分

$$\hat{\mathbf{x}} = F(\mathbf{x}) = \sqrt{n} F \mathbf{x} \quad (14)$$

を用いると、 $C(\mathbf{x})$ は次式のように固有値分解できる。

$$C(\mathbf{x}) = F \text{diag}(\hat{\mathbf{x}}) F^H \quad (15)$$

2.3 巡回行列を使った Ridge 回帰

$C(\mathbf{x})$ をデータ行列として、Ridge 回帰を行う、中心化していない共分散行列は、DCT 行列 F を用いて次式のように表せる。

$$C(\mathbf{x})^H C(\mathbf{x}) = F \text{diag}(\hat{\mathbf{x}}^*) F^H F \text{diag}(\hat{\mathbf{x}}) F^H \quad (16)$$

さらに、要素積 \odot を用いて

$$F^H F = (F^*)^T F = I \quad (17)$$

$$\text{diag}(\hat{\mathbf{x}}^*) \cdot \text{diag}(\hat{\mathbf{x}}) = \text{diag}(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}) \quad (18)$$

が成り立つから、

$$C(\mathbf{x})^H C(\mathbf{x}) = F \text{diag}(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}) F^H \quad (19)$$

となる。ここで、 $\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}$ は、信号 \mathbf{x} の自己相関 (auto correlation) またはパワースペクトル (power spectrum) と呼ばれる量である。すなわち、 P によって空間方向にシフトさせ

た信号どうしの共分散を表している．以上から，データ行列として $C(\mathbf{x})$ を用いた Ridge 回帰の閉形式解は，次のようになる．

$$\hat{\mathbf{w}} = (C(\mathbf{x})^H C(\mathbf{x}) + \lambda I)^{-1} C(\mathbf{x})^H \hat{\mathbf{y}} \quad (20)$$

$$= \text{diag} \left(\frac{\hat{\mathbf{x}}^*}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda} \right) \hat{\mathbf{y}} \quad (21)$$

$$= \frac{\hat{\mathbf{x}}^* \odot \hat{\mathbf{y}}}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda} \quad (22)$$

$$(23)$$

よって，逆 DFT F^{-1} を用いて， w が求められる．

$$\mathbf{w} = F^{-1}(\hat{\mathbf{w}}) \quad (24)$$

2.4 Kernel Ridge 回帰

任意の正定値関数 $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ に対して，

$$\mathbf{w} = \sum_{i=1}^n \alpha_i \phi(\mathbf{x}_i) \quad (25)$$

とおけば，グラム行列

$$K = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & k(\mathbf{x}_1, \mathbf{x}_3) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & k(\mathbf{x}_2, \mathbf{x}_3) & \cdots & k(\mathbf{x}_2, \mathbf{x}_n) \\ k(\mathbf{x}_3, \mathbf{x}_1) & k(\mathbf{x}_3, \mathbf{x}_2) & k(\mathbf{x}_3, \mathbf{x}_3) & \cdots & k(\mathbf{x}_3, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & k(\mathbf{x}_n, \mathbf{x}_3) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \quad (26)$$

を用いて，Representer 定理より，

$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i = \sum_{j=1}^n \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \quad (27)$$

$$\|\mathbf{w}\|^2 = \alpha^T K \alpha \quad (28)$$

が成り立つ．すなわち， n 個の訓練データ $(\mathbf{x}_i, y_i)_{i=1 \dots n}$ に対する Kernel Ridge 回帰は，以下のように定式化される．

$$\min_{\alpha} \sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2 + \lambda \alpha^T K \alpha, \quad f(\mathbf{x}_i) = \sum_{j=1}^n \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \quad (29)$$

Kernel Ridge 回帰の閉形式解は，以下のようになる．

$$\alpha = (K + \lambda I)^{-1} \mathbf{y} \quad (30)$$

2.5 巡回行列を使った Kernel Ridge 回帰

Ridge 回帰と同様に、ベクトル \mathbf{x} とその巡回シフトに対するカーネルは、 P と $k(\cdot, \cdot)$ を用いて

$$\{k(\mathbf{x}, P^u \mathbf{x}) \mid u = 0, \dots, n-1\} \quad (31)$$

となる。よって、ベクトル $\mathbf{k}^{\mathbf{xx}}$ を、

$$\mathbf{k}_i^{\mathbf{xx}} = k(\mathbf{x}, P^{i-1} \mathbf{x}) \quad (i = 1, \dots, n) \quad (32)$$

と定義し、これを用いた巡回行列

$$C(\mathbf{k}^{\mathbf{xx}}) = \begin{bmatrix} (P^0 \mathbf{k}^{\mathbf{xx}})^T \\ (P^1 \mathbf{k}^{\mathbf{xx}})^T \\ \vdots \\ (P^{n-1} \mathbf{k}^{\mathbf{xx}})^T \end{bmatrix} = \begin{bmatrix} k(\mathbf{x}, P^0 \mathbf{x}) & k(\mathbf{x}, P^1 \mathbf{x}) & \dots & k(\mathbf{x}, P^{n-1} \mathbf{x}) \\ k(\mathbf{x}, P^{n-1} \mathbf{x}) & k(\mathbf{x}, P^0 \mathbf{x}) & \dots & k(\mathbf{x}, P^{n-2} \mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}, P^1 \mathbf{x}) & k(\mathbf{x}, P^2 \mathbf{x}) & \dots & k(\mathbf{x}, P^0 \mathbf{x}) \end{bmatrix} \quad (33)$$

を得られる。さらに、中心化していない共分散行列は、DCT 行列 F を用いて次式のよう表せる。

$$C(\mathbf{k}^{\mathbf{xx}})^H C(\mathbf{k}^{\mathbf{xx}}) = F \text{diag} \left(\hat{\mathbf{k}}^{*\mathbf{xx}} \odot \hat{\mathbf{k}}^{\mathbf{xx}} \right) F^H \quad (34)$$

以上から、データ行列として $C(\mathbf{k}^{\mathbf{xx}})$ を用いた Kernel Ridge 回帰の閉形式解は、次のようになる。

$$\hat{\alpha} = (C(\mathbf{k}^{\mathbf{xx}}) + \lambda I)^{-1} \mathbf{y} \quad (35)$$

$$= \frac{\hat{\mathbf{y}}}{\hat{\mathbf{k}}^{\mathbf{xx}} + \lambda} \quad (36)$$

よって、逆 DFT F^{-1} を用いて、 w が求められる。

$$\alpha = F^{-1}(\hat{\alpha}) \quad (37)$$

3 実装

RBF kernel を用いる。

$$\mathbf{k}^{\mathbf{xx}'} = \exp \left\{ -\frac{1}{\sigma^2} \left(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - 2F^{-1}(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}}') \right) \right\} \quad (38)$$

実行環境 以下の環境でプログラムを実装した. 主要なスクリプトの記述には Python を採用した. 使用したコードは, https://github.com/yumaloop/CV_report2019 で公開した.

- OS: macOS Mojave 10.14.4
- App: Apple QuickTime Player 10.5 (935.3)
- Lang: Python 3.7.4
- Lib: Numpy 1.16.1, Opencv-python 4.1.0

なお, 今回は, KCF Tracker アルゴリズムのみを実装したため, 動画の入出力やユーザーインターフェースは, OpenCV の定義済み関数を利用した. 具体的に, KCF Tracker のオブジェクトをインポートできる OpenCV の `cv2.TrackerKCF.create()` を使って, Object Tracking を行うコードは次のようになる. このうち, コード中のオブジェクト `tracker` と関連する処理を実装した.

```
import sys
import cv2

# Set up tracker as KCF.
tracker = cv2.TrackerKCF_create()

# cap: captured images from the video
input_file = "*****.mp4"
cap = cv2.VideoCapture(input_file)
cap_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
cap_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
cap_size = (cap_width, cap_height)

# out: result video (m4v)
# see also https://gist.github.com/takuma7/44f9ecb028ff00e2132e
output_file = "*****.m4v"
fourcc = cv2.VideoWriter_fourcc('m', 'p', '4', 'v')
out = cv2.VideoWriter(output_file, fourcc, 20.0, cap_size)

# Exit if video not opened.
if not cap.isOpened():
    print("Cannot open the video file.")
    sys.exit()

# Read first frame.
ret, frame = cap.read()
if not ret:
    print('Cannot read the video file.')
    sys.exit()

# Initialize bounding box
bbox = (287, 23, 86, 320)
# Uncomment the line below to select a different bounding box
bbox = cv2.selectROI(frame, False)

# Initialize tracker
ret = tracker.init(frame, bbox)
```

```

while True:
    # Read a new frame
    ret, frame = cap.read()
    if not ret: break

    # =====
    ''' Update tracker (Tracking process in OpenCV) '''
    ret, bbox = tracker.update(frame)
    # =====

    # Draw bounding box
    if ret:
        # Tracking succeeded
        p1 = (int(bbox[0]), int(bbox[1]))
        p2 = (int(bbox[0] + bbox[2]), int(bbox[1] + bbox[3]))
        cv2.rectangle(frame, p1, p2, (255,0,0), 2, 1)
    else :
        # Tracking failed
        cv2.putText(frame, "Tracking failure detected",
            (100, 80), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0,0,255) ,2)

    # Display result
    cv2.imshow("Tracking", frame)

    # Save result as mp4 file
    out.write(frame)

    # Exit if ESC pressed
    k = cv2.waitKey(1) & 0xff
    if k == 27:
        break

```

4 考察

5 結論

- foo
- bar
- buz

参考文献

- [1] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.
- [2] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. Technical report, Cambridge, MA, USA, 1980.

- [3] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pp. 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.
- [4] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pp. 147–151, 1988.
- [5] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.
- [6] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *CoRR*, Vol. abs/1404.7584, , 2014.