

SURF: Speeded Up Robust Features

中部大学 藤吉研究室

2009 年 8 月 10 日

1 はじめに

近年，物体認識や，特徴点追跡，マッチングにおいて，SIFT[1] が良く用いられている．SIFT は，画像の回転やスケール変化等に不変であるため，それら要素に適している．しかし，処理コストが高いことが問題となっており，SIFT を用いるアプリケーションではリアルタイム処理が難しい課題となっている．SIFT の高速化が重要なテーマである．SIFT を高速化したアルゴリズムである Grabner らが提案する FastApproximatedSIFT[2] は検出段階で Integral Image を活用し DoG 処理 DoM 処理に変更，記述段階では Integral Histogram を用いている．しかし，DoM 処理において DoG を近似していないため，ノイズによわく，スケール変化の頑健性も SIFT に比べ劣る．そこで，今回は，検出段階を SIFT と近似して高速化をはかった Herbert Bay らが提案する Speeded Up Robust Features(SURF)[4] を調査する．

2 検出段階

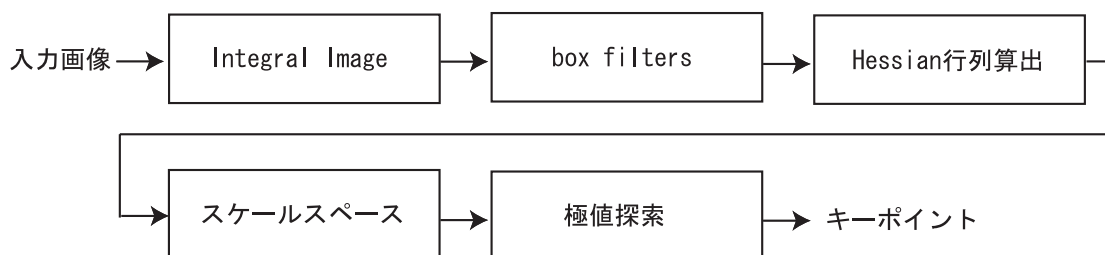


図 1: 手法の流れ

SIFT は DoG 画像を作成することで特徴点を検出しているが，SURF は Hessian 行列を算出することで特徴点を検出している．Hessian 行列を高速に算出するために，Integral Image を用いる．これは，Hessian 行列の算出に box filters を用いる box filters はある領域の画素の合計を求める処理が含まれるため，Integral Image を使うと高速に処理できるからである．キーポイント算出までの処理の流れを図 1 に示す．

2.1 特徴点とは

ハリスや SIFT の特徴点は基本的には輝度変化の大きなエッジ上が選ばれる．エッジ上が好まれる理由は，テクスチャのない所よりもテクスチャが多いところの方が，その場所を示す情報が多いためである．エッジは， x 方向と y 方向の片方に輝度差が大きい場所や xy 方向の両方の輝度差が大きい場所をいい，輝度差には正負があるため大きく 3 種類に分けられる．(全部で 6 種類) 図 2 に x と y 方向の輝度差について示す． xy 方向の両方の輝度差が大きく同一方向の時は図 2(左) のようにドーム型の曲面になる．図 2(中) は， xy 方向の両方の輝度差が大きい， x 方向と y 方向の輝度差の向きが異なっている場合，図 2(右) は， x 方向と y 方向の片方に輝度差が大きい場合を示している．エッジを見つける場合は，輝度差が図 2 のどのタイプになるのかを調べることにより検出できる．図 2 の分類は Hessian 行列を求めることにより判別することができる．

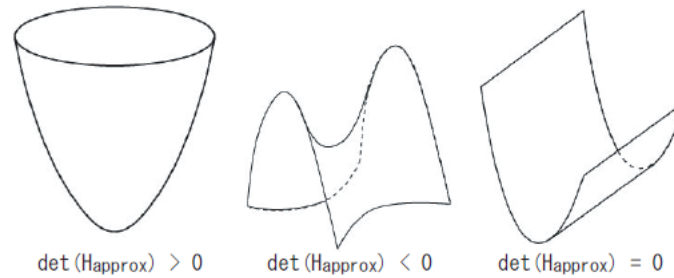


図 2: 曲面

2.2 Hessian 行列

$$H = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix} \quad (1)$$

Hessian 行列は，表面のエッジに関する情報を示している． $L_{xx}(x, y, \quad)$ は x 軸の 2 次微分結果であり， \quad はガウス関数の標準偏差である．

次式の $\det(H)$ を算出することでエッジの種類を判別することができる．

$$\det(H) = L_{xx}L_{yy} - (L_{xy})^2 \quad (2)$$

$\det(H)$ が正の値を持つ時は， x 方向と y 方向の輝度差が同一方向に大きいことになる． $\det(H)$ が負の値を持つ時は x 方向と y 方向の輝度差が大きい，向きが異なる場合である． $\det(H)$ が 0 のときは， x 方向と y 方向の片方に輝度差が大きい場合を示している． $\det(H)$ を調べることにより図 2 に示すようにエッジの種類を判別が可能となる．SURF は特徴点を各スケールの画像に対して $\det(H)$ を求めて極値探索により特徴点を検出する．Hessian 行列算出において， L_{yy} や L_{xy} を微分により求めると，各画素の値を参照して求める必要があるため，計算コストが高い．そこで，SURF では，Integral Image と box filters を用いることで高速に計算している．

box filters により算出された L_{yy}, L_{xx}, L_{xy} をそれぞれ D_{yy}, D_{xx}, D_{xy} とすると, $\det(H)$ は次式より算出される .

$$\det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \quad (3)$$

0.9 倍してあるのは, 近似のために生じる誤差を修正するためである . 0.9 倍は実験的に求めたパラメータである . SURF で特徴点とするのは図 2(左) のみである . 他の種類は特徴点として検出しない .

2.3 box filters

$\sigma=1.2$ の L_{xy}, L_{yy} の box filters を図 3 に示す . フィルタサイズ 9×9 の box filters は $\sigma=1.2$ のガウシアンと同等となる . L_{xy}, L_{yy} は, 図 3(左) になり box filters で近似した D_{xy}, D_{yy} は図 3(右) になる . box filters で近似することにより各画素ごとの演算が不要である領域の合計値を算出する処理に変わるため, Integral Image を用いた算出が可能となり高速化できる .

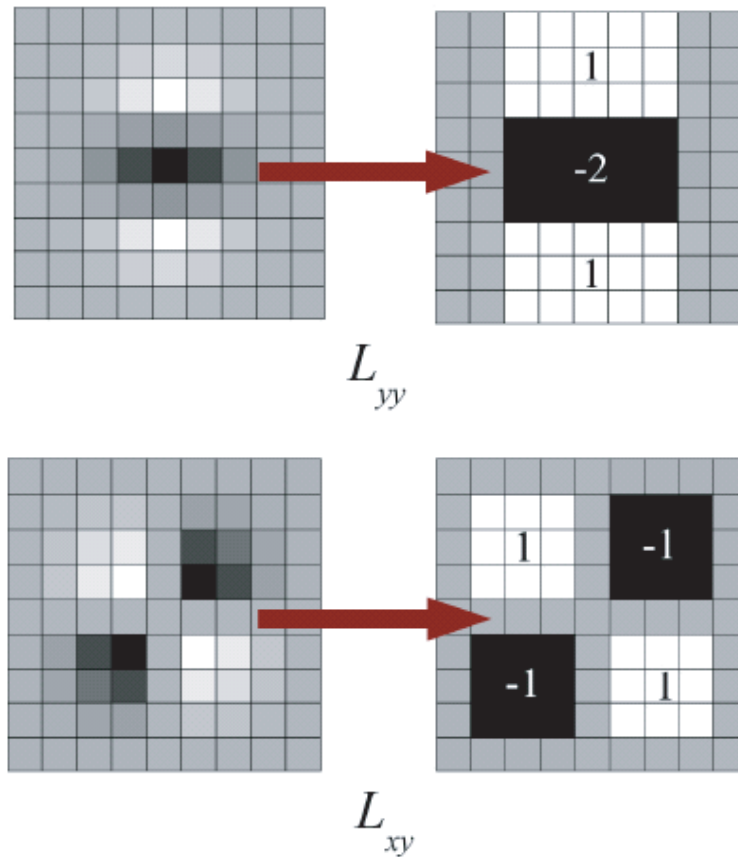


図 3: box filters

特徴点検出にはまず, $9 \times 9, 15 \times 15, 21 \times 21, 27 \times 27$ のフィルタサイズを用いる . それぞれ, 1.2, 2.0, 2.8, 3.6 のスケールに対応している .

2.4 Integral Image

Integral Image(積分画像) は, 2001 年に Viola と Jones により提案された手法である．矩形領域が重なり合う場合, 積分画像は高速に矩形領域の輝度値の和を算出することができる．画像 $I(x, y)$ から求められる積分画像 $ii(x, y)$ は次式より算出できる．

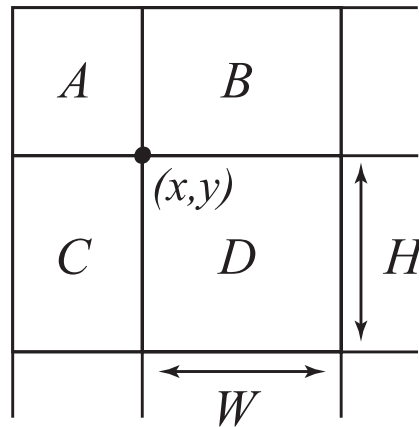
$$ii(x, y) = ii(x - 1, y) + s(x, y) \quad (4)$$

$$s(x, y) = s(x, y - 1) + I(x, y) \quad (5)$$

ここで, s は y 軸方向の累積画像, $s(x, -1) = ii(-1, y) = 0$ とする．一度, 積分画像を算出することにより, 図 4(a) の領域 D の輝度値の合計を式 (6) より高速に算出することができる．

$$D = (ii(x, y) + ii(x + W, y + H)) - (ii(x + W, y) + ii(x, y + H)) \quad (6)$$

積分画像を用いない場合, 矩形領域内の画素数だけ加算が必要である．しかし, 積分画像を用いた場合, ラスタスキャンを 2 回行い, 矩形領域の数の度に 3 回の加減算を行うだけでよい．



積分画像による矩形領域の輝度和

図 4: 領域の輝度和 ([?])

2.5 スケールスペース

SIFT では, 画像サイズを変えることで, スケールスペースを求めている．しかし, 画像のスケールリングは処理コストが大きい．SURF では, box filters と Integral Image を用いてキーポイント候補を求めているため, box filters のフィルタサイズを大きくしても処理コストが増加しない．そのため, SIFT のスケールスペースより高速に求めることができる．

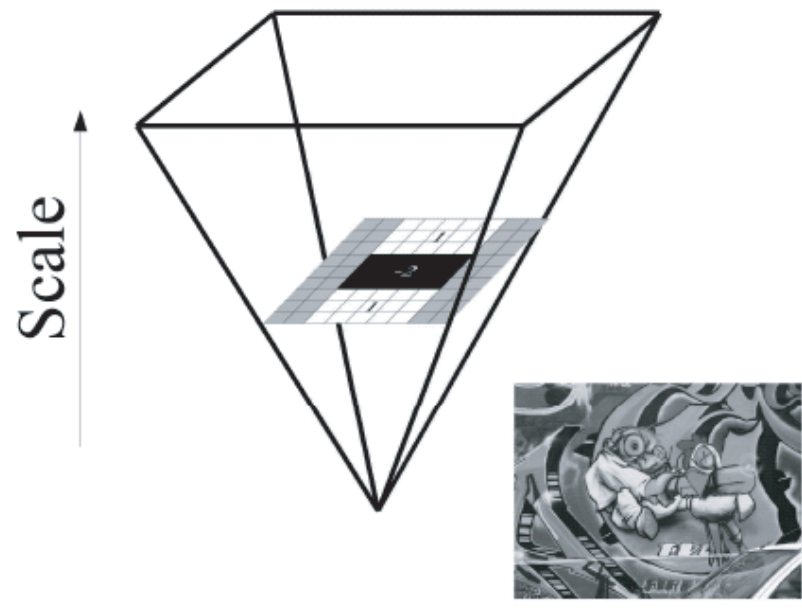


図 5: スケールスペース

2.6 極値探索

キーポイント検出の極値探索は SIFT と同様に行われる．異なるスケールのフィルタリング結果を 3 枚 1 組で 26 近傍探索を行う．26 近傍で極値ならキーポイントとして検出する．検出結果を図 7 に示す．

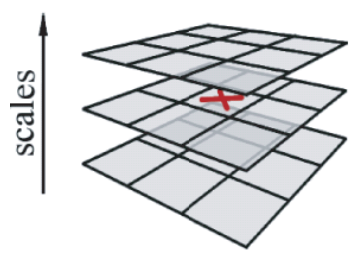


図 6: 極値探索

3 記述段階

回転の不変性を得るためのオリエンテーションと特徴量記述方法を下記に示す．

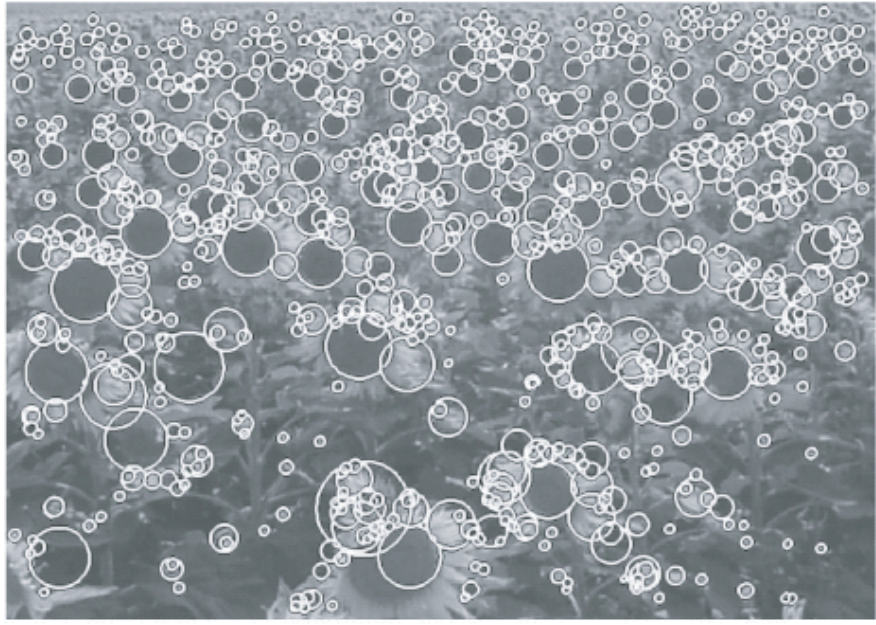


図 7: キーポイント検出結果

3.1 オリエンテーション

特徴点を中心とした半径 $6s$ (s : スケール) の範囲から Haar-Wavelet ($4s$) を用いて勾配方向を求める。そのとき, SIFT と同様に勾配強度を算出し, 勾配強度が最も大きい方向をオリエンテーションとしている。SIFT では 10 度ごとにもっているが, SURF では 60 度ごとである。U-SURF はこの工程をスキップしている。そのため, 回転変化に不変ではない。

3.2 特徴量記述

特徴点を中心とした $20s \times 20s$ (s : スケール) の正方形領域を 4×4 のサブ領域に分割し Haar-Wavelet ($2s \times 2s$) を算出することにより, 勾配ベクトルを求める。勾配ベクトルの x 方向, y 方向をそれぞれ d_x, d_y とする。 d_x, d_y より 4 次元ベクトル ($\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|$) を求める。 x 方向の勾配が大きく, その絶対値も大きいときは正の勾配方向となり, 逆に負の勾配方向のときは, x 方向の勾配が小さく, その絶対値は大きくなる。このように, 方向とその絶対値の情報により強度変化の極性の情報が得られる。16 のサブ領域から 4 次元ベクトルを求めて 64 個の特徴量を算出することができる。より精度を求めるために 128 次元の特徴量も用意されている。これは, 同様の方法により 4×4 のサブ領域に分割する。64 次元では 4 次元ベクトル ($\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|$) を求めているが, 128 次元では, $\sum d_x, \sum |d_x|$ を求める時に $d_y < 0$ と $d_y \geq 0$ の場合に分けて $d_y < 0$ の時の合計と $d_y \geq 0$ の時の合計を求めている。同様に, $\sum d_y, \sum |d_y|$ を求めている。これは, 強度変化 (向き, 強さ) の情報が詳細になるため精度の向上が期待できる。また, 比較用として 36 次元の特徴量も用意された。これは, 64 次元の特徴量算出のときに 4×4 のサブ領域に分割するのを 3×3 のサブ領域に変更する。これにより 9×4 の 36 次元の特徴量を記述する。

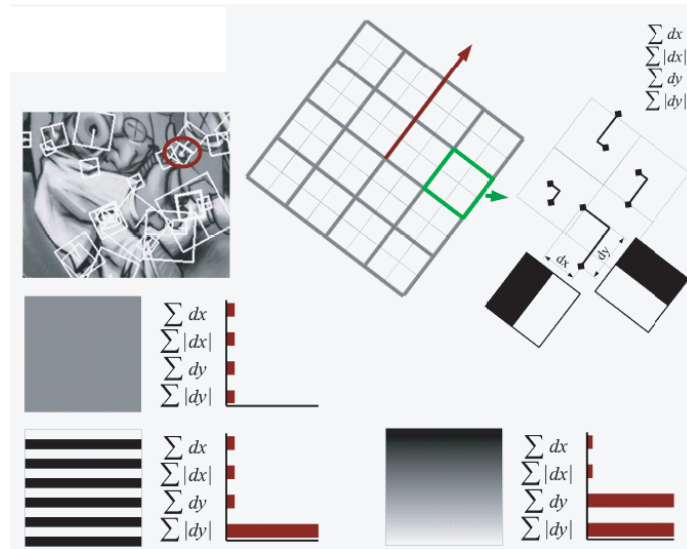


図 8: 特徴量記述

4 評価実験

テストデータには, mikolajczyk(<http://www.robots.ox.ac.uk/~vgg/research/affine/>) が公開している画像を使用．特徴量の精度の比較, Detector の速度比較, 回転, 照明, スケール変化による特徴点の再現性の比較結果を示す．図 9 に, descriptor の精度の比較を示す．精度の比較では, 2 種類の 30 度の視点変化の画像に対して検出率と誤検出率を求めて評価している．比較対象は, SIFT, PCA-SIFT, GLOH, SURF, SURF-128, SURF-36 である．SURF の後に付く数字は算出する特徴量の次元数を示している．

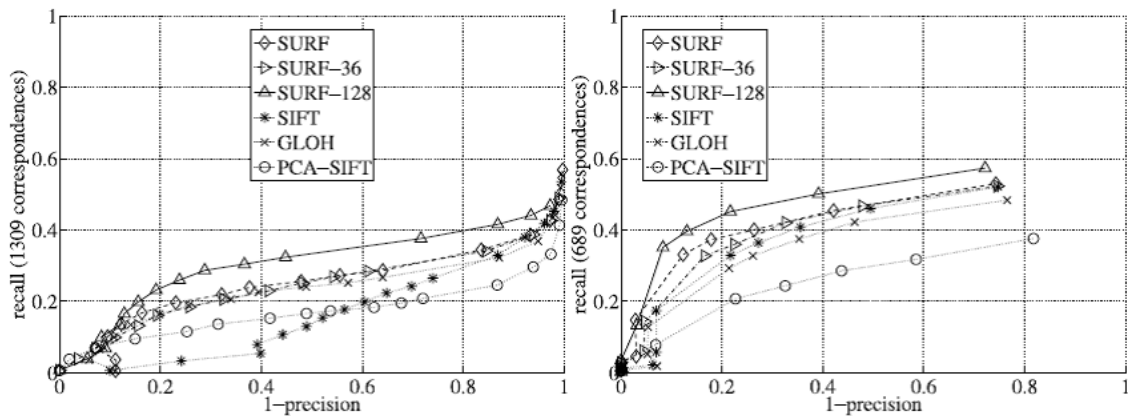


図 9: descriptor の精度の比較

図 9 では, グラフ左上にあるほど性能が良いことになる．図 9(左)SIFT と PCA-SIFT が最も悪い結果となり, 良い結果を出したのは SURF-128 である．SURF と SURF-36 に大きな差がない．図 9(右) では, PCA-SIFT が最も悪い結果となり, ここでも, SURF-128 が良い結果となった．次に, detector の処理時間の比較を表 1 に示す．比較対象は, SURF で用いる Fast-Hessian, Hessian-Laplace, Harris-Laplace, DoG である．特徴点数は, Fast-Hessian と DoG

に大きな違いはないが，処理速度は約 $1/3$ 倍ほど短縮されているのがわかる．Hessian-Laplace と Harris-Laplace は処理時間が大きい．

表 1: detector 処理時間

detector	しきい値	特徴点数	処理時間 [ms]
Fast-Hessian	600	1418	120
Hessian-Laplace	1000	1979	650
Harris-Laplace	2500	1664	1800
DoG	default	1520	400

次に，descriptor の処理時間の比較を表 2 に示す．比較対象は，オリエンテーションが除かれた U-SURF と SURF，SURF-128，SIFT である．SIFT では 1 秒かかるのが SURF では約 $1/3$ ほど短縮されている．最も高速なのが，オリエンテーションを除いた U-SURF である．SURF-128 は SURF に比べて次元数が倍になっているため 10 % ほど処理時間がかかる事がわかる．次

表 2: descriptor 処理時間

	U-SURF	SURF	SURF-128	SIFT
time[ms]	255	354	391	1036

に，対応点マッチングの正解率の比較を表 3 に示す．比較対象は，SIFT，PCA-SIFT，GLOH，

表 3: 対応点マッチング率

	U-SURF	SURF	SURF-128	SIFT	GLOH	PCA-SIFT
正解率	83.8 %	82.6 %	85.7 %	78.1 %	78.3 %	72.3 %

SURF，SURF-128 である．最もよい性能は SURF-128 となり，続いて U-SURF と SURF の順番となり，一番悪いのは PCA-SIFT となる．SURF を用いると SIFT にくらべ高速になる他に精度も向上することがわかる．次に，特徴点の再現性の比較を図 11 に示す．

比較対象は，Fast-Hessian，Hessian-Laplace，Harris-Laplace，DoG である．図 11 上の左と右は，視点変化による比較であり，下の左は照明変化，右はスケール変化の比較である．全てにおいて，Fast-Hessian は DoG と同等な性能であることがわかる．次に，画像変化に対する descriptor の性能比較を図 12 に示す．比較対象は，SIFT，PCA-SIFT，GLOH，SURF，SURF-128 である．図 12 上の左は，視点変化（50 度），右はスケール変化，中の左は自動車画像の平滑化，右は木の画像の平滑化，下の左は照明変化，右は JPEG 圧縮の結果である．SURF は，ほとんどすべての比較で他の記述子より優れた結果となる．全ての結果をまとめると，SURF 記述子は他の記述子より優れており，計算するのは速い．SURF-128 は，通常の SURF よりわずかに良い結果を示した．



図 10: 照明変化と視点変化のテストデータ

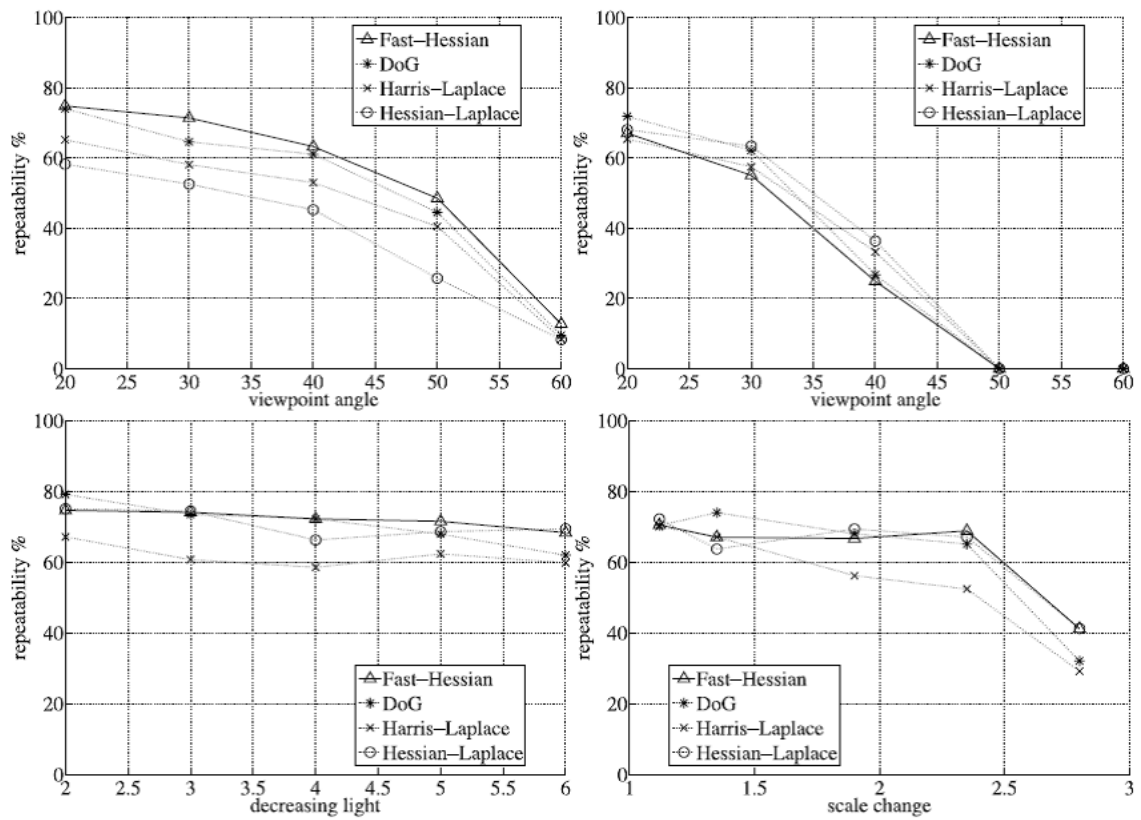


図 11: 特徴点の再現性の比較

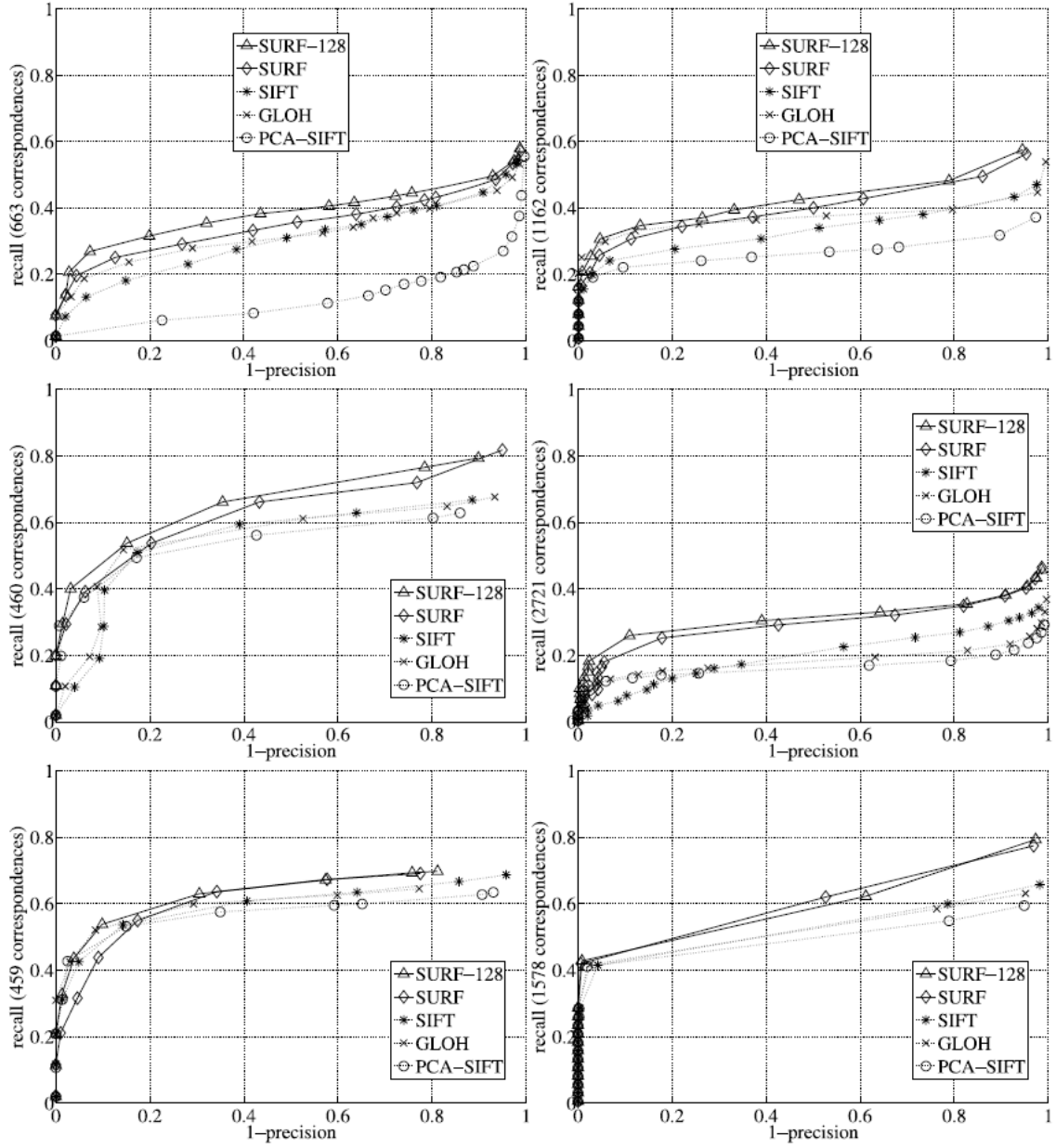


図 12: 画像変化に対する descriptor の性能比較

5 ソースコード

SURF は OpenCV1.1 に実装されており，OpenCV1.1 をインストールすることで利用することができる．ここでは，OpenCV1.1 に実装されているプログラムと別のプログラムの SURF を比較する．OpenCV1.1 に実装された SURF のマッチング結果を図 13 に示し，別のプログラムの SURF の結果を図 14 に示す．

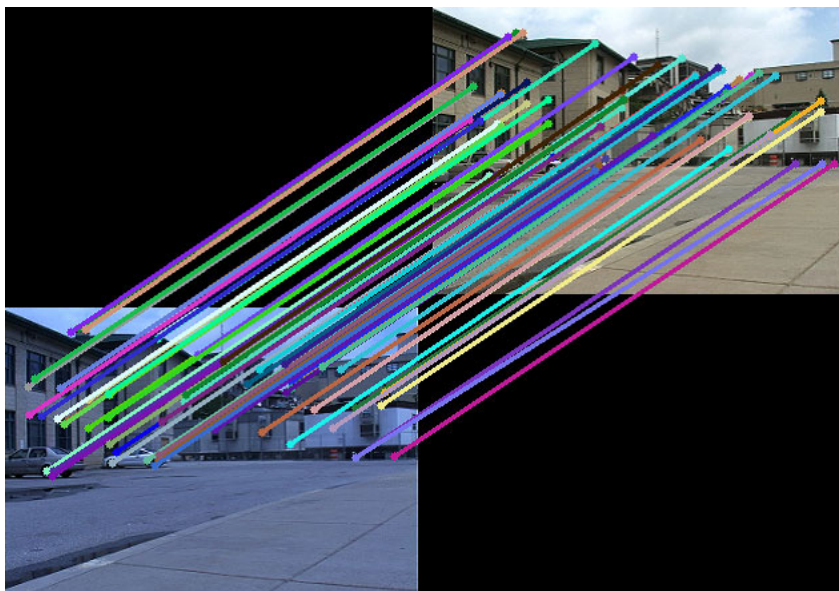


図 13: OpenCV1.1 に実装された SURF (マッチング数 63 個)

両者を比較すると，マッチングした位置とマッチング数が異なる結果となった．

6 おわりに

SURF は SIFT の精度を維持したまま高速にマッチングが可能であることがわかった．これは，Integral Image の利用と Hessian 行列算出に box filters を利用したことで可能になった．

参考文献

- [1] David G.Lowe, “Distinctive image features from scale-invariant keypoints”, Int.Journal of Computer Vision, Vol.60, No.2, pp.91-110, 2004 .
- [2] Grabner Michael, Grabner Helmut, Bischof Horst, “ Fast approximated SIFT ”, Proc. of ACCV, pp.918-927, 2006.
- [3] P. Viola and M. Jones, “Rapid Object Detection Using a Boosted Cascade of Simple Features”, Proc. of IEEE Compute Vision and Pattern Recognition, pp. 511–518, 2001.

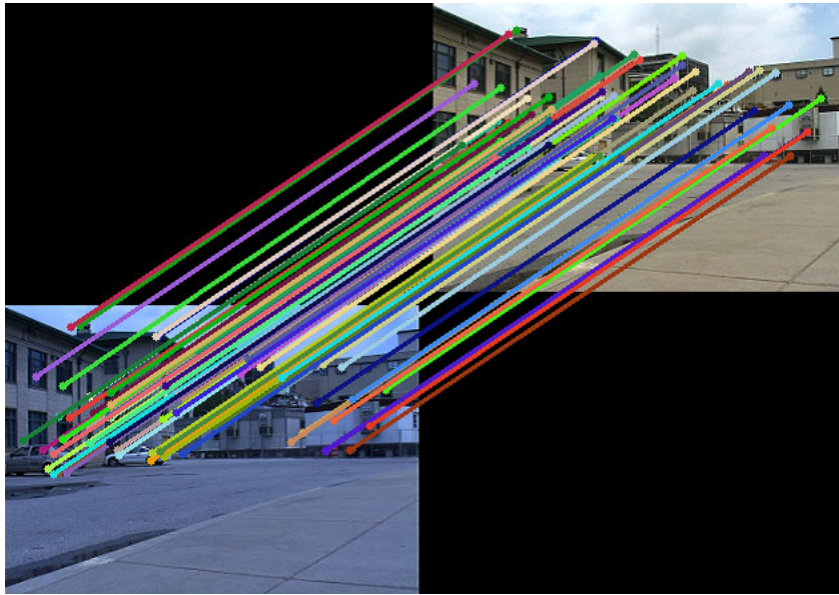


図 14: 別のプログラムの SURF (マッチング数 72 個)

- [4] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded Up Robust. Features”, In ECCV , pp.404-417, 2006.