

## 報告

### 試したドメイン

- アニメの顔



- カートゥーンキャラクター（ポケモン）



- 建物



- 服



- 商品



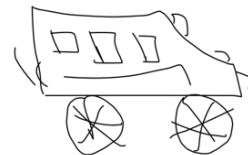
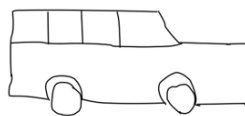
- 顔画像



- 動物 (犬)



- スケッチ



## 試した方法




### 1. Image-similarity-measures ライブラリ


- Root mean square error (RMSE) : 各ピクセルの差を計算する。入力画像とターゲットが一致すれば 0 になります。

- **Peak signal-to-noise ratio (PSNR)** : 信号の最大パワーと、その表現の忠実性に影響を与えるノイズのパワーとの比を測定するものです。
- **Structural similarity index (SSIM)** : データ圧縮などの処理やデータ送信時の損失による画質の劣化を定量化するものです。(値範囲 -1 ~ 1、完全一致は 1)
- **Feature-based similarity index (FSIM)** : 復元した画像とオリジナル画像との間の構造的および特徴的な類似度を比較する目的で開発されました。この手法は、位相整合性と勾配の大きさに基づいています。(値範囲 0 ~ 1、完全一致は 1)
- **Information theoretic-based Statistic similarity measure (ISSM)** : 情報理論を統計量で補う（情報理論は画像の強度の関係を予測する能力が高いから）。この手法は情報理論（シャノンエントロピー）と統計量（SSIM）に加え、エッジ検出（Canny）による明確な構造的特徴を取り入れたものです。
- **Signal to reconstruction error ratio (SRE)** : 信号のパワーに対する誤差を測定します。
- **Spectral angle mapper (SAM)** : 物理的なスペクトル分類である。2つのスペクトル間の角度を計算し、それらをバンド数に等しい次元を持つ空間のベクトルとして扱うことで、2つのスペクトルの類似度を判定します。角度が小さければ小さいほど、参照スペクトルに近いということです。
- **Universal image quality index (UIQ)** : 相関性の低下と輝度の歪みとコントラストの歪みの組み合わせのモデルです。

## 結果

### アニメ顔画像





		
	rmse : 0.020675059407949448 psnr : 33.6910653465076 ssim : 0.7281224303348339 fsm : 0.4322899623954397 issm : 0.0 sre : 41.343862384119454 sam : 89.40120382249957	rmse : 0.021304087713360786 psnr : 33.430740868358185 ssim : 0.7136584127996318 fsm : 0.41514416331887743 issm : 0.0 sre : 41.368153515792194 sam : 89.44335917392834

	uiq : 0.027918297247531604	uiq : 0.08930537745604614
		rmse : 0.019919002428650856 psnr : 34.01464942625003 ssim : 0.7617373907810149 fsim : 0.4150735512459705 issm : 0.0 sre : 43.382923985169995 sam : 89.64622858057893 uiq : 0.012321575896868638




## 建物

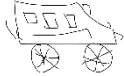
		
	rmse : 0.018374629318714142 psnr : 34.71562865903812 ssim : 0.6743795410764828 fsim : 0.29672690286438663 issm : 0.0 sre : 41.37553458309034 sam : 76.54604858810661 uiq : - 0.002497651269310938	rmse : 0.018113093450665474 psnr : 34.84014753034549 ssim : 0.7235885948139211 fsim : 0.3319530761017757 issm : 0.0 sre : 41.519503992121834 sam : 77.1294541215253 uiq : 0.0021460567337208696
		rmse : 0.01816035993397236 psnr : 34.81751207044559 ssim : 0.7160427435768467 fsim : 0.2989146967117015 issm : 0.0 sre : 42.425778711506325 sam : 61.12600211195615 uiq : 0.004670954577797454

## ポケモン





		
	rmse : 0.019698679447174072 psnr : 34.111259018687775 ssim : 0.7017652798296018 fsim : 0.34869190340594264 issm : 0.0 sre : 32.4405488848232 sam : 13.837584862537048 uiq : 0.003743752559536912	rmse : 0.01772238314151764 psnr : 35.029556786326935 ssim : 0.7303531453504606 fsim : 0.3720487816873266 issm : 0.0 sre : 32.88100019988674 sam : 16.16372031981588 uiq : 0.025012508087624594
		rmse : 0.016827432438731194 psnr : 35.479643379954865 ssim : 0.7477918332294449 fsim : 0.3540630063946535 issm : 0.0 sre : 31.68229339065055 sam : 14.710323432528448 uiq : 0.016922893761030327

## スケッチ

		
	rmse : 0.011393965221941471 psnr : 38.86650327375369 ssim : 0.9532574901060742 fsim : 0.3850250288110557 issm : 0.0 sre : 61.77038361372672 sam : 87.44315407232679	rmse : 0.011312618851661682 psnr : 38.928737675456574 ssim : 0.9537641957791653 fsim : 0.4016968026995658 issm : 0.0 sre : 61.801500266158186





	uiq : - 0.0008654510705927502	sam : 87.4742058782244 uiq : - 0.006932648160433431
		rmse : 0.012510589323937893 psnr : 38.05444482496219 ssim : 0.943468863679522 fsim : 0.3803873692194533 issm : 0.0 sre : 61.28400829187885 sam : 86.90503932304905 uiq : 0.0012486085759363507

## 顔画像

		
	rmse : 0.03229045495390892 psnr : 29.818515812663605 ssim : 0.6353289287162114 fsim : 0.30361895672240097 issm : 0.0 sre : 48.00711813185458 sam : 86.57886521558528 uiq : 0.031515843473597496	rmse : 0.02706577070057392 psnr : 31.35159019163089 ssim : 0.7538280771811906 fsim : 0.2894873397262056 issm : 0.0 sre : 48.81494085188078 sam : 89.18760755498279 uiq : 0.025471720877726874
		rmse : 0.02136894129216671 psnr : 33.40434077909427 ssim : 0.6924388175404156 fsim : 0.3324216093187277 issm : 0.0 sre : 44.22560722933511 sam : 86.0559141402825 uiq : 0.03682614969318347


## 服







		
	rmse : 0.026928389444947243 psnr : 31.39579299619632 ssim : 0.625951178324375 fsim : 0.31588314583361776 issm : 0.0 sre : 44.47933163241781 sam : 87.84338623661722 uiq : 0.003421956869395914	rmse : 0.024254804477095604 psnr : 32.30404380491214 ssim : 0.7170704163359866 fsim : 0.28341789474047463 issm : 0.0 sre : 44.93213974741731 sam : 88.85473172656313 uiq : - 0.003624180453251951
		rmse : 0.019912833347916603 psnr : 34.0173387878223 ssim : 0.7100469084531126 fsim : 0.27123587417379075 issm : 0.0 sre : 40.747231839935225 sam : 86.85145100655498 uiq : - 0.01797919595957364

商品

		
	rmse : 0.014999504201114178 psnr : 36.478462254313996 ssim : 0.8594856300708559 fsim : 0.34642160873476796 issm : 0.0 sre : 45.114039059288515 sam : 88.43047931071769	rmse : 0.01861298643052578 psnr : 34.60367921522098 ssim : 0.8161664627664907 fsim : 0.28161027995853444 issm : 0.0 sre : 44.17689454159523 sam : 88.38634979154088

	uiq : 0.14655688071266293	uiq : 0.05069281206534761
		rmse : 0.01992190070450306 psnr : 34.01338580778079 ssim : 0.7962842781945606 fsim : 0.2455104524859398 issm : 0.0 sre : 44.45958930751579 sam : 88.5482648553704 uiq : - 0.006107056907367911

犬

		
	rmse : 0.018867891281843185 psnr : 34.48553327253299 ssim : 0.7529883413163931 fsim : 0.317097493143379 issm : 0.0 sre : 43.86113465434869 sam : 86.90498783091748 uiq : 0.017648275051535458	rmse : 0.027722828090190887 psnr : 31.14325050046878 ssim : 0.6199088632251359 fsim : 0.2970361090428075 issm : 0.0 sre : 42.428215541827825 sam : 88.71869921689358 uiq : - 0.018208353869550283
		rmse : 0.022800209000706673 psnr : 32.84122470548587 ssim : 0.7469475721113015 fsim : 0.2772577141844957 issm : 0.0 sre : 43.613743736660695 sam : 88.99739063656227 uiq : 0.005192279612816253

考察



- **Root mean square error (RMSE)**  
この方法は各ピクセルの差を計算するので、入力画像と処理された全く同じ画像の間の差を求めるのが得意だと思います。全体または物として比較していないため、比較しているものは同じ種類かどうか判明できないと思います。上の建物例を見ると、違う建物より、同じ建物同士の方が差が大きい。
- **Peak signal-to-noise ratio (PSNR)**  
この手法は2つの画像の画質的な類似度を計算すると思います。例えば、圧縮された画像とオリジナル画像間の類似度など。
- **Structural similarity index (SSIM)**  
この手法は、データ圧縮やデータ送信といった処理によって下がった画質の画像とオリジナル画像の類似度を求めるものです。画像のオブジェクトとして計算するのではなくて、画像の構造として計算します。物や材質の視点で類似度の判明はできません。
- **Feature-based similarity index (FSIM)**  
この方法はオリジナル画像と復元画像の構造と特徴両方の類似度を計算する方法なので、似ている物を分類できると思いますが、上の建物の結果を見ると同じ建物の画像はある程度似ていると思いますが、**FSIM** による類似度は極めて低いと考えています。それに、違う建物との類似度の方が高かった。
- **Information theoretic-based Statistic similarity measure (ISSM)**  
この手法による類似度はなぜか全部 0.0 になっています。恐らく、こういう画像の組み合わせに対応ではないかと考えています。この手法を提案した論文を調べたところ、評価のセクションには、全く違う画像のペアではなくオリジナル画像とぼかした同じ画像間の類似度を求めています。
- **Signal to reconstruction error ratio (SRE)**  
論文によりますと、この方法はいろんな違う明るさの画像間の類似度を求めるのが得意そうです。
- **Spectral angle mapper (SAM)**  
この技術は、校正された反射率データに使用した場合、照明やアルベドの影響を比較的受けにくいそうです。
- **Universal image quality index (UIQ)**  
この手法は類似度の求めるのにいくつかの視点から計算していますが、物としての特徴や情報は使っていないです。

いろんな画像ペアを上記のアルゴリズムで類似度を計算した結果、これらの手法は違う画像ペアを使いますとうまくいかないと分かった。

## 2. Key points matching

- SIFT

Scale-Invariant Feature Transform (SIFT)は4つの処理に大別されます。

1. スケール空間における極値検出：Difference of Gaussian (DoG)を使います。
2. キーポイントの位置同定：2つの固有値の差が閾値より大きければ、そのキーポイントは候補から除外されます。
3. 回転角の計算：回転不変性を実現するため。
4. 特徴量の記述：画像勾配の大きさと向きに基づいて各キーポイントの特徴量記述子を計算する。

この方法の検出器はスケール不変です。

参考：[https://docs.opencv.org/master/da/df5/tutorial\\_py\\_sift\\_intro.html](https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html)

- BRIEF

Binary Robust Independent Elementary Features (BRIEF)は特徴量記述子を使うことなく直接2値ベクトルを計算します。平滑化した画像パッチに対して $nd$ 個の画素 $(x,y)$ のペアを構築します。次に、各ペアに対して画素値を比較します。

SIFTは128次元の実ベクトル(浮動小数)を計算します。このような特徴点が数千個もあると想像してください。マッチングの際にメモリ使用量が増大し計算時間がかかってしまいます。高速化のためにSIFT特徴量を圧縮できます。それでも、まず初めにSIFT特徴量を計算しなければいけません。ここではBRIEFという省メモリかつ高速なマッチングが可能な二値ベクトルを計算する特徴量記述子を使います。

参考：[https://docs.opencv.org/master/dc/d7d/tutorial\\_py\\_brief.html](https://docs.opencv.org/master/dc/d7d/tutorial_py_brief.html)

- ORB

Oriented FAST and Rotated BRIEF (ORB)は基本的にFASTによる特徴点検出とBRIEFによる特徴量記述子を組合わせたものです。まず始めにFASTによって特徴点を検出し、Harrisのコーナー評価により上位 $N$ 点を選びます。また、マルチスケールの特徴を得るため、ピラミッドを使います。

計算コスト、マッチング精度、特許を考慮するとSIFTとSURFの良い代替と言えます。

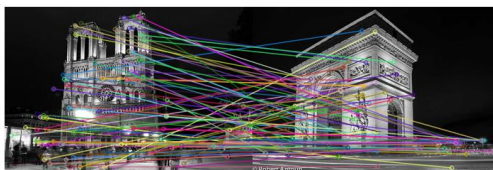
参考：[https://docs.opencv.org/master/d1/d89/tutorial\\_py\\_orb.html](https://docs.opencv.org/master/d1/d89/tutorial_py_orb.html)

## 結果

- 建物



1



2

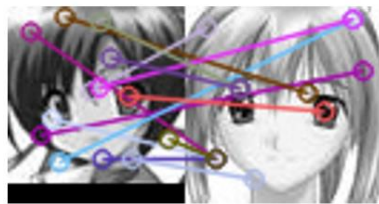


3

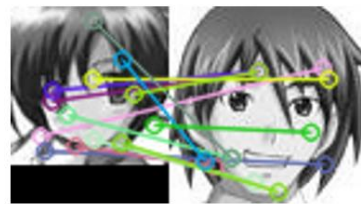
	1					2					3				
	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)
SIFT	0.072	739	898	238	29	0.05	739	547	100	15.5	0.06	898	547	147	20.3
BRIEF	0.008	114	226	27	15.8	0.006	114	113	31	27.3	0.006	226	113	62	36.5
ORB	0.017	500	500	98	19.6	0.017	500	495	87	17.4	0.019	500	495	73	14.6

- BRIEF 手法が一番は早い。
- 図 1 が一番 Rate が高いはずなので、BRIEF の結果は良くないと考えられます。

- アニメ顔画像



1



2

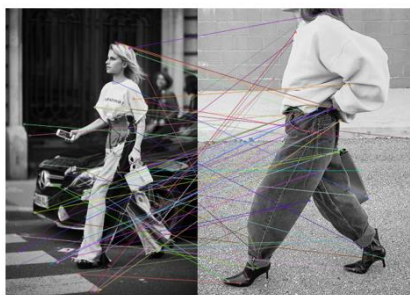


3

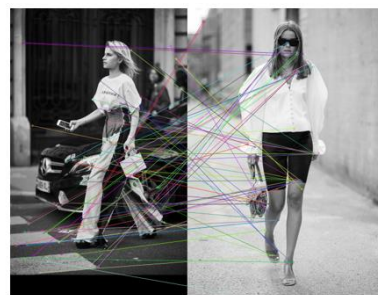
	1					2					3				
	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)
SIFT	0.004	59	71	13	20	0.004	59	39	13	26.5	0.003	71	39	15	27.2
BRIEF															
ORB	0.001	0	5	0	0	0.00	0	1	0	0					

○ BRIEF と ORB 手法はうまくいかなかった。画像質が低いまたは画像サイズが小さいのが原因かもしれません。キーポイントをうまく検出できなかった。

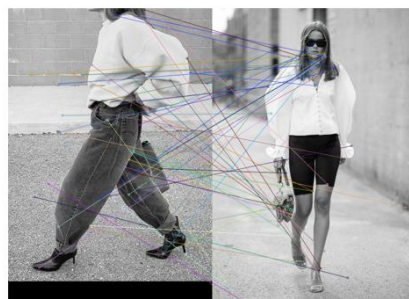
# ● 服



1



2

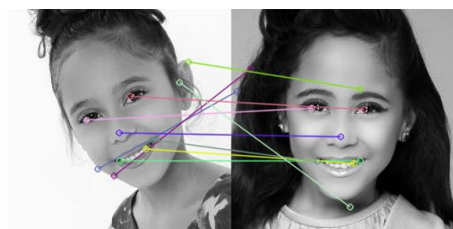


3

	1					2					3				
	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)
SIFT	0.323	1810	4145	69	2.31	0.24	1810	922	74	5.41	0.27	4145	922	62	2.44
BRIEF	0.04	497	309	135	33.4	0.04	497	194	150	43.4	0.05	309	194	73	29.0
ORB	0.05	500	500	82	16.4	0.04	500	500	82	16.4	0.05	500	500	89	17.8

- 図 1 の似ているジーンズを検出できなかった。単色で処理するからだと思えます。

- 顔画像



1



2

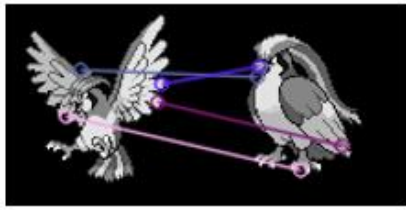


3

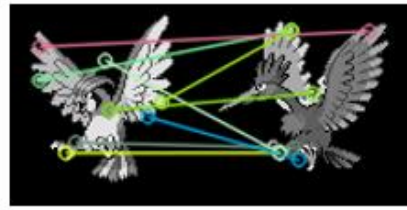
	1					2					3				
	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)
SIFT	0.02	167	154	10	6.23	0.023	167	187	10	5.64	0.024	154	187	8	4.69
BRIEF	0.00	16	25	5	24.3	0.00	16	22	1	5.26	0.00	25	22	1	4.25
ORB	0.00	371	390	21	5.51	0.01	371	342	13	3.64	0.00	390	342	25	6.83

- 図 1 が同じ人物の画像なので、Rate が一番高いはずなので、ORB の結果だけが間違った。
- BRIEF 手法は顔の特徴を検出するのが得意と考えられます。

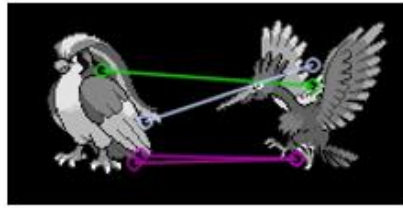
- ポケモン



1



2

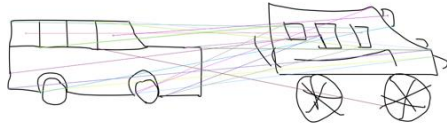


3

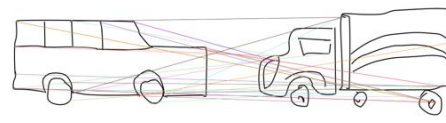
	1					2					3				
	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)
SIFT	0.009	68	46	4	7.01	0.006	68	57	8	12.8	0.009	46	57	4	7.76
BRIEF	0.000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ORB	0.002	61	31	1	2.17	0.001	61	35	10	20.8	0.001	31	35	1	3.03

○ BRIEF がキーポイントを検出できなかった。

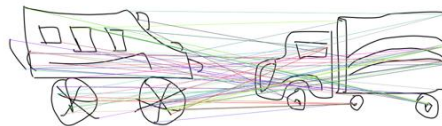
# ● スケッチ



1



2



3

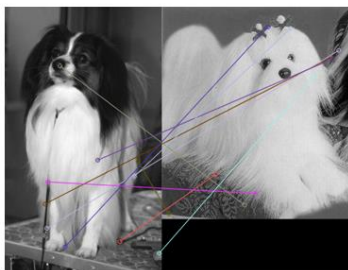
	1					2					3				
	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)
SIFT	0.42	109	392	27	10.77	0.36	109	217	31	19.01	0.35	392	217	77	25.28
BRIEF	0.14	139	376	27	10.48	0.142	139	251	34	17.43	0.14	376	251	66	21.05
ORB	0.067	500	500	71	14.2	0.06	500	500	65	13.0	0.066	500	500	50	10.0

○ 図 1 は同類の車なので、ORB 方法だけ正しい結果得られた。

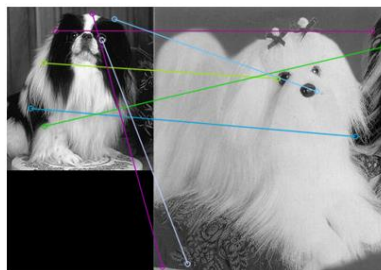


- 図3は違う種類ですが、構造的には似ているので SIFT と BRIEF の Rate が高かった。

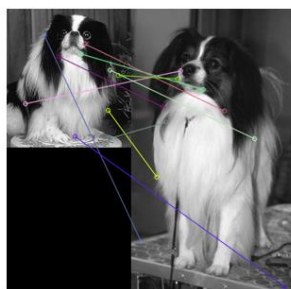
- 犬



1



2



3

	1					2					3				
	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)
SIFT	0.038	158	313	10	4.246	0.032	158	281	7	3.189	0.042	313	281	10	3.367
BRIEF	0.002	1	25	1	7.69	0.002	1	8	0	0	0.003	25	8	4	24.24
ORB	0.010	301	444	12	3.221	0.009	301	340	11	3.432	0.011	444	340	9	2.29

- 商品



1



2



3

	1					2					3				
	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)	Time (sec)	Kp1	Kp2	Match	Rate (%)
SIFT	0.077	328	174	26	10.35	0.010	328	700	16	3.112	0.099	174	700	14	3.203
BRIEF	0.010	29	20	1	4.081	0.010	29	42	3	8.450	0.009	20	42	6	19.35
ORB	0.021	384	298	16	4.692	0.028	384	392	7	1.804	0.015	298	392	6	1.739

## 考察

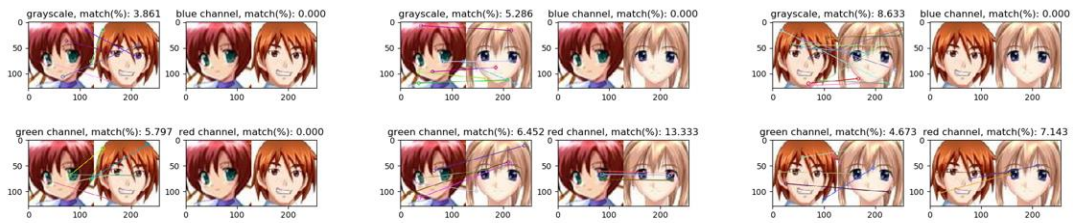
- 色の情報を使っていない。
- ほとんどのキーポイントは背景にある。
- 背景にオブジェクトがない画像はもっとうまくキーポイントを検出することができる。
- マッチポイントはあまりありません。

## 3. Key points matching + 色の情報

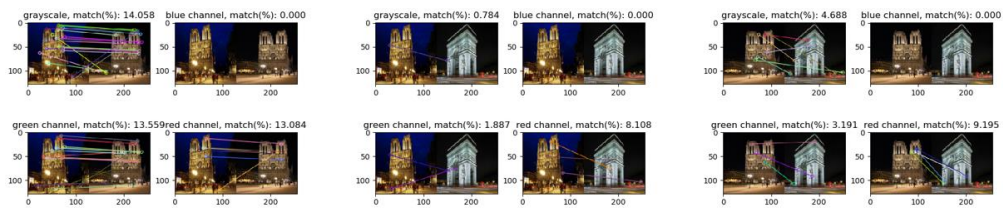
色情報を使うために、RGB チャンネルに key points matching アルゴリズムを適用した。

## 結果

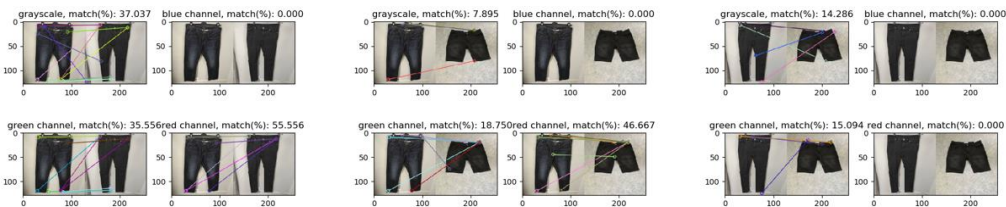
- アニメ顔



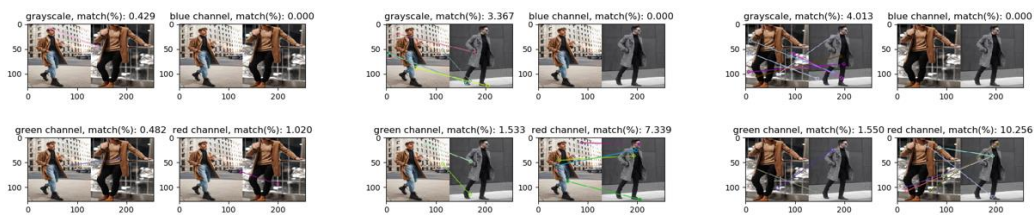
## • 建物



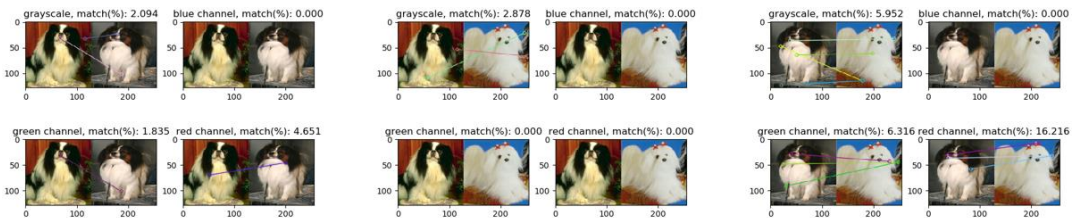
## • 商品



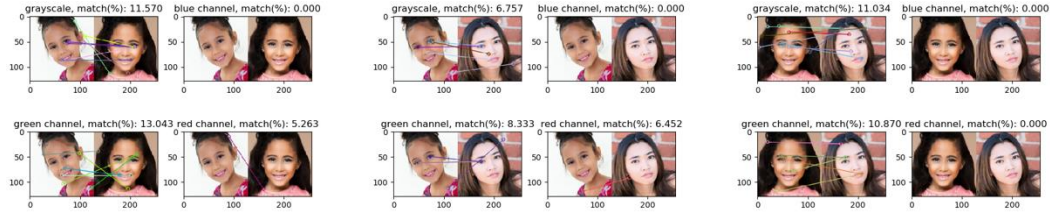
## • 服



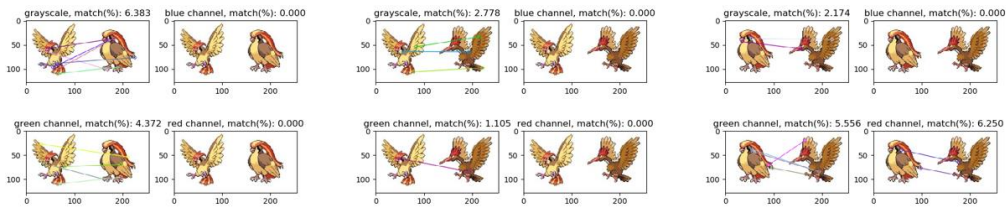
## • 犬



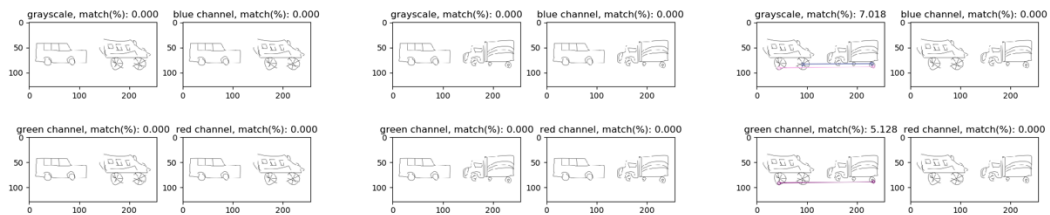
## • 顔



## ● ポケモン



## ● スケッチ



## 考察

- blue channel だけ keypoint がなかった（全部の画像ペア）
- 残り green と red channels で検出したキーポイントは grayscale のとあまり変わらないという印象でした。

## 4. Key points matching + Histogram

- マッチポイントを検出する
- ポイントのサイズと位置をとる（両方の画像）
- ヒストグラムのコンテナを作る（例えば、4つのコンテナだと、0~63, 64~127, 128~191, 192~255 で分けられる）
- ポイントのサイズをループして各チャンネルのヒストグラムを作る（サイズが4だったら、4x4）
- ヒストグラムをノーマライズする
- 両方の画像のヒストグラムを作れたら、各ヒストグラム差の総和を求める

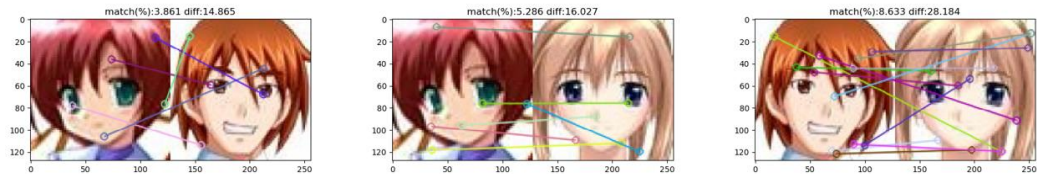


g. 各マッチポイントを step 3 ~ step 6 をする

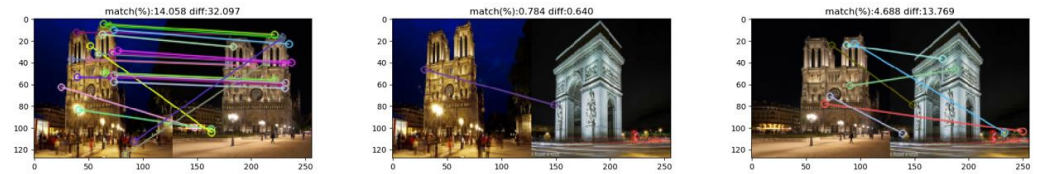
h. 最後に全部のマッチポイントのヒストグラムによる色の差を求めることができた

## 結果

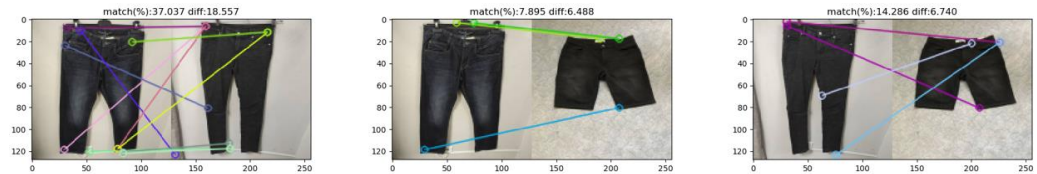
- アニメ顔



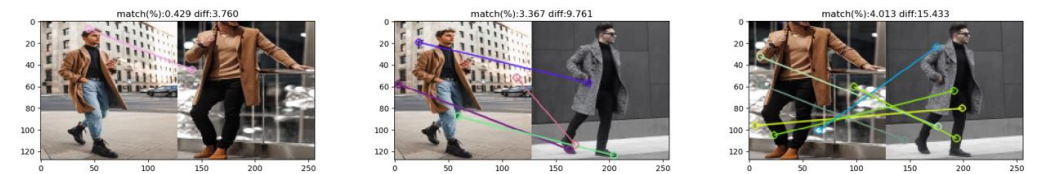
- 建物



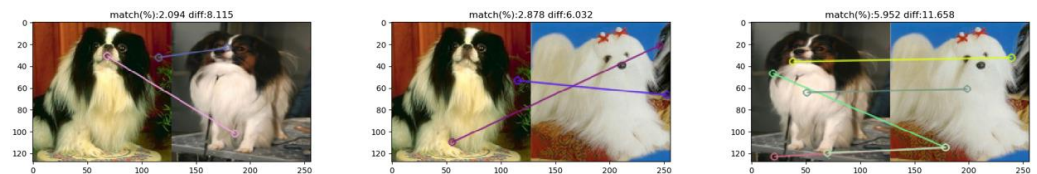
- 商品



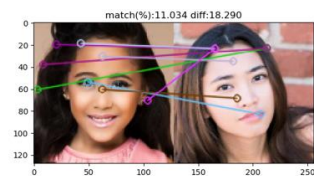
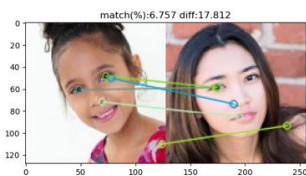
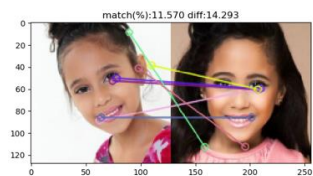
- 服



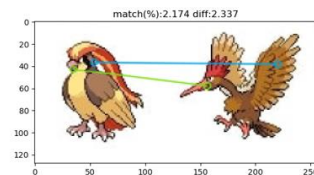
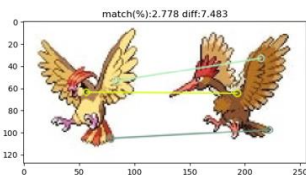
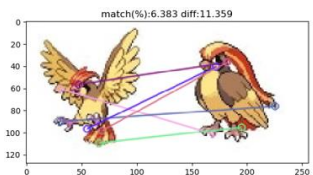
- 犬



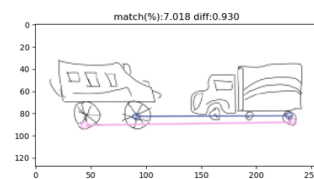
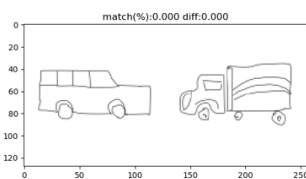
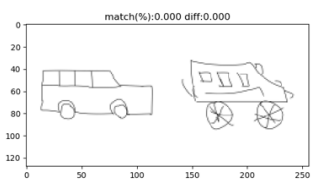
- 顔



- ポケモン



- スケッチ



## 考察

- マッチポイントがあつていれば、色の視点で類似度を求めることができると考えています。

## 5. Frechet Inception Distance (FID)

Frechet Inception Distance (FID)は GAN による生成した画像と正解画像の特徴距離を求めるものです。

## 結果

- アニメ顔



FID: 253.7056957788412



FID: 224.2854593335234

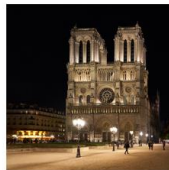


FID: 160.66370662748628



- 建物

FID: 53.974199199443085



FID: 220.05133518829516



FID: 218.92166371326888



- 商品

FID: 42.14264778095393



FID: 132.5724415919746



FID: 134.68251211512913



- 服

FID: 334.5038475012299



FID: 244.2945145481018



FID: 372.8600689504675



- 犬

FID: 140.6064183759991



FID: 252.3508249727599



FID: 239.97898347107963



- 顔

FID: 152.21808526886193



FID: 233.21661575323736



FID: 179.59502300474045



- ポケモン

FID: 152.52421988537154



FID: 294.73483864482586



FID: 330.56629629473446



- スケッチ

FID: 282.71339890519226



FID: 231.8122011095084



FID: 274.172209142467



### 考察

- 全体的にいい結果得られました。
- 形と色の視点では類似度を求めれたと思います。
- 失敗したドメインは服とスケッチだけでした。

### まとめ

うまく類似度を出せなかった

- Image-similarity-measures ライブラリ
- Key points matching アルゴリズムのみ
- Key points matching + RGB チャンネル

うまく類似度を出せた

- Frechet Inception Distance (形と色またはスタイルの視点)
- Key points matching + histogram
- Key points matching + Gaussian filter + histogram

Key points matching(SIFT)の結果は検出されたポイントとマッチングによりますので、不安なところは十分あります。ただし、背景に何も映っていない画像やガウシアンフィルタをかけることによって、精度を少し上げることができると考えた。