画像応用数学特論最終課題レポート

研究室　知的メディア工学研究室
学籍番号 [redacted]
氏名　小新田 亮人
メール [redacted]at]e.hiroshima-cu.ac.jp([at]を@に変換)
提出日　2016 年 1 月 19 日

1, 課題概要

階層グラフカットによるステレオマッチングで，サイト[1]からダウンロードした以下の 2 つの画像の視差画像を出力した．なお，今回使用した画像は「縮小専用。AIR[2]」というフリーソフトを用いて，元の画像から 1/6 にサイズを縮小した画像を使用した．

図 1：入力画像として使用した左画像

図 2：入力画像として使用した右画像

2, アルゴリズム説明

今回，階層グラフカットによるステレオマッチングのアルゴリズムは，次ページのアルゴリズム 1 のように実装した．

[1] http://vision.middlebury.edu/stereo/data/
[2] http://labo.i-section.net/shukusen-air/

アルゴリズム 1：階層グラフカットによるステレオマッチングを行うプログラムのアルゴリズム

1, ラベル配列 A[i]を計算
2, 視差記録配列 data の初期化(今回は初期値として，0 を設定)
3, 画素数に応じて，グラフを初期化
4, 画素に対応するノードに対して，
　　SOURCE 側: 視差記録配列 data を基に計算したデータコスト
　　SINK 側: 視差記録配列 data の値に最も近いラベル配列 A[i]を基に計算したデータコスト
　を設定
5, 画素ノードを結ぶエッジに対応するノードを追加し，それらのノードに対して以下のコストを追加
　　SOURCE 側: 視差記録配列 data に関するスムーズコストを設定
　　SINK 側: 視差記録配列 data の値に最も近いラベル配列 A[i]に関するスムーズコストを設定
　また，画素ノードとエッジノード間のエッジコストは以下のように定義
　　配列 data に関するスムーズコスト <= 配列 A[i]に関するスムーズコストである時
　　　エッジノード→画素ノードのコスト: 1 万などの大きい値に設定
　　　エッジノード←画素ノードのコスト: 0 と |配列 data, A[i]に関するコスト – data のコスト|
　　　　のうち，**大きい値をコストとして設定**
　　配列 data に関するスムーズコスト > 配列 A[i]に関するスムーズコストである時
　　　エッジノード→画素ノードのコスト: 0 と |配列 data, A[i]に関するコスト – data のコスト|
　　　　のうち，**大きい値をコストとして設定**
　　　エッジノード←画素ノードのコスト: 1 万などの大きい値に設定
6, MAXFLOW 関数を用いてグラフカットを行い，コストを計算
7, 6 で計算したコストが現状のコストよりも低ければ，計算したコストに更新し，
　視差記録配列 data のラベルを更新
8, コストが収束するまで 3～7 を繰り返す

また，アルゴリズム内のデータコストは以下の式(1)のように設定した．

$$D(f_{x,y}) = \sum_{x,y} \| I_l(x,y) - I_r(x - f_{x,y}, y) \| \qquad (1)$$

式(1)における$I_l(x,y)$は，(x,y)座標における左画像の画素の明るさを示し，$f_{x,y}$は視差を示す．また，アルゴリズム内のスムーズコストは以下の式(2)のように設定した．

$$V(f_p, f_q) = c * |f_p - f_q| \qquad (2)$$

式(2)のf_pは画素 p の明るさを，f_qは画素 p の隣の画素 q の明るさを示す．また，c は定数であり，今回作成したプログラムでは式(3)のように設定した．

$$c = \begin{cases} c_0 & if \ \|I_l(p) - I_l(q)\| > T \\ c_1 & if \ \|I_l(p) - I_l(q)\| \le T \end{cases} \qquad (3)$$

式(3)の$I_l(x)$は左画像の画素 x における明るさを示す．c_0, c_1, Tは適当な値の定数である．なお，c_0とc_1は$c_0 < c_1$の関係になるような値に設定した．

3, 実行結果

2 のアルゴリズムに沿って作成プログラムを$\boldsymbol{c_0 = 1, c_1 = 5, T = 5}$とし，**ウインドウサイズを 5，最大視差を 32** とした時の実行結果は，図 3 のようになった．また，この時の**実行時間は 5.74 秒**であった．

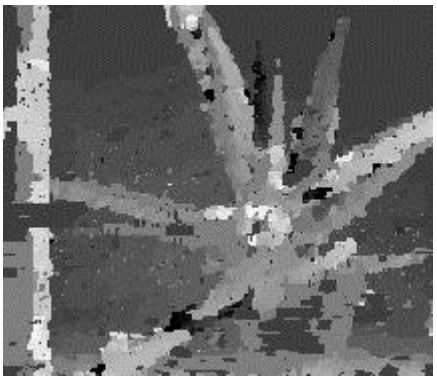


図 3：出力された視差画像

この図 3 の結果を見ると，視点に近いほど，画素が白くなっているため，妥当な結果であると考えられる．

4, 考察

3 章の実行結果を出力するために，設定した値が妥当かどうかを検証するため，ウインドウサイズなどの値を変更し，妥当な値の検証を行った．その結果を表 1, 2 に示す．

表 1：各値を変更した時の出力結果

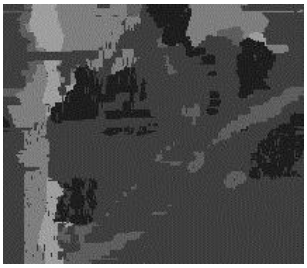
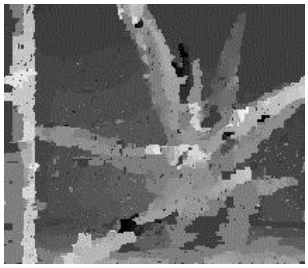
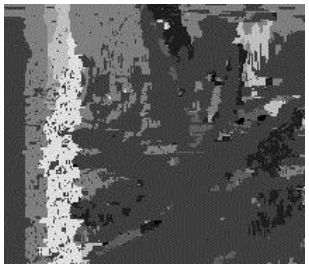
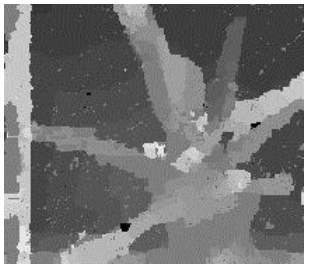
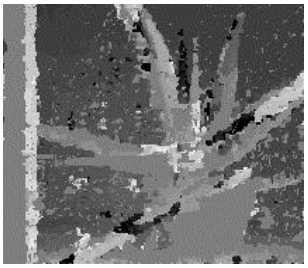
出力画像				
c_0	10	10	2	1
c_1	50	50	10	5
T	100	100	15	5
ウインドウサイズ	5	5	5	7
最大視差	64	32	64	32
実行時間[秒]	13.72	11.80	10.53	6.21

表 2：各値を変更した時の出力結果

出力画像			
c_0	1	1	1
c_1	5	5	5
T	5	5	5
ウインドウサイズ	5	13	3
最大視差	64	32	32
実行時間[秒]	10.42	10.83	7.30

3 章や表 1,2 の結果から，実行時間，出力結果を考慮すると，定数の値は**$c_0 = 1, c_1 = 5, T = 5$が適切**で，**最大視差は 32 が適切**，**ウインドウサイズは 5～9 の値に収めることが妥当**であると考えられる．故に，3 章の実行結果は実行時間，出力結果の観点から妥当な結果であると考えられる．

5, 開発環境

今回開発した環境及び実行時間の計測に用いた環境は，以下の表 3 のような環境で行った.

表 3：今回使用した開発環境

OS	Windows7 Professional 64bit
CPU	Intel Core i7-3770(3.40GHz)
メモリ	12.0GB
GPU	AMD Radeon HD 7800 Series
使用言語	C++
統合開発環境	Visual Studio Express 2013 for Windows Desktop
ライブラリ	OpenCV ver2.4.11