広島市立大学 情報科学研究科 知能工学専攻
コンピュータグラフィックス研究室
[redacted] 中根 智絵
nakane@info.ime.hiroshima-cu.ac.jp
2014 年 1 月 13 日提出

画像応用数学特論
12月12日出題レポート

階層グラフカットでステレオ

開発環境

- OS : Windows7 Professional 32bit
- 開発環境 : Visual C++ 2010 Express
- プログラミング言語 : C/C++
- ライブラリ : OpenCV 2.4.5，MAXFLOW

アルゴリズム

階層グラフカットでステレオマッチングを行うアルゴリズムを図 1 に示す．データコスト D とスムーズコスト V の設定を図 2 に示す．データコスト D とスムーズコスト V の計算式を以下に示す．

$$D(f_{x,y}) = \sum_{W_{x,y}} \parallel I_l(x,y) - I_r(x - f_{x,y}, y) \parallel$$

$$V(f_p, f_q) = c \mid f_p - f_q \mid$$

ただし c は定数である．データコスト D の計算ではブロックマッチングを使用しており，ウィンドウサイズは奇数で設定する．

図 2 のノード同士を繋ぐ $e_{Ap}, e_{Aq}, e_{Bp}, e_{Bq}$ の設定を図 3 に示す．V(Bp,Bq)≤V(Ap,Aq) の場合と V(Bp,Bq)>V(Ap,Aq) の場合によって異なる値を設定する．

今回，階層グラフカットの性能を検証するため α 拡張でステレオマッチングを行うプログラムも作成した．α 拡張でステレオマッチングを行うプログラムのアルゴリズムを図 4 に示す．コストの設定を図 5 に示す．データコスト D とスムーズコスト V の計算式は階層グラフカットと同様である．

- ラベルの生成(A={{0},…,{…,n}})
- 2視点からの画像をそれぞれ読み込む
- 出力画像(B)を用意する
- E=とても大きな値
- for ループ=0～とても大きな値
  - success=0
  - for i=0~|A|
    - グラフの初期化
    - for 全画素
      - ノードの追加
      - A[i]のうちBpに最も近い値をApに設定
      - ソースとシンクとのデータコストを設定
    - end for
    - for 全画素隣接点
      - ノードの追加
      - A[i]のうちBpに最も近い値をApに設定
      - A[i]のうちBqに最も近い値をAqに設定
      - ソースとシンクとのスムーズコストを設定
      - 隣接する画素のノードとのスムーズコストを設定
    - end for
    - 最大流・最小カットアルゴリズムの適用
    - flow=求まったラベルで計算した総コスト関数
    - flow<Eのとき
      E=flow
      ノードが索子に属している画素=Ap
      success=1
    - グラフの消去
  - end for
  - success=0のときループ脱出
- end for
- 出力画像の濃度調整
- 出力画像の保存

図 1: 階層グラフカットのアルゴリズム

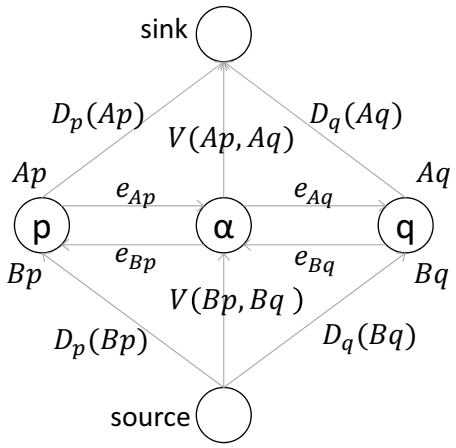
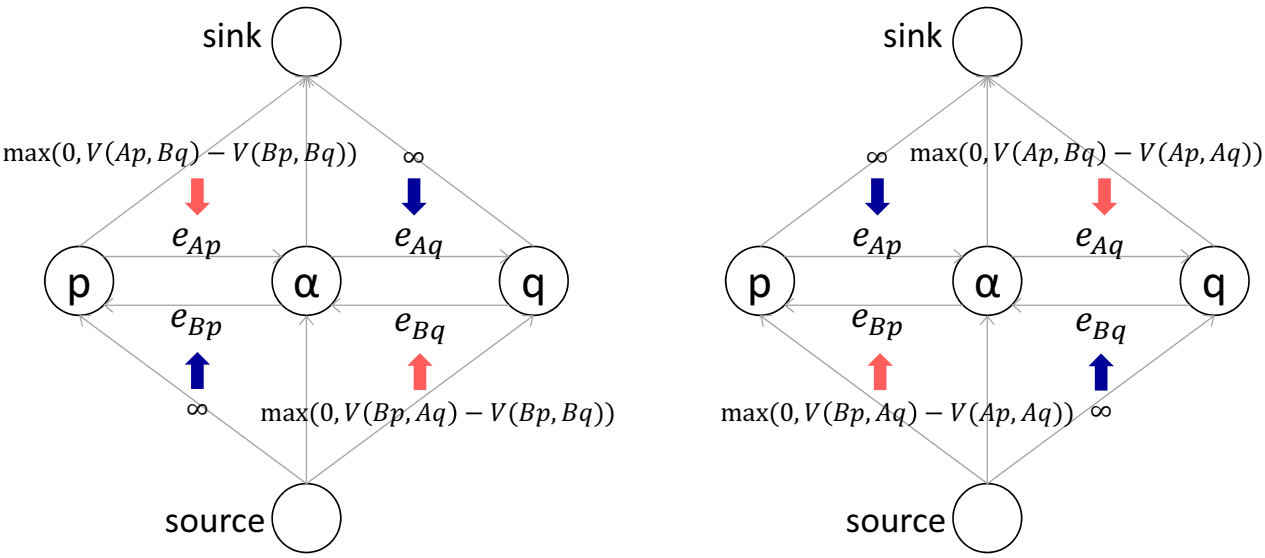


図 2: 階層グラフカットのコストの設定



(a)V(Bp,Bq)≤V(Ap,Aq) の場合　　(b) V(Bp,Bq)>V(Ap,Aq) の場合

図 3: ノード同士を繋ぐコストの設定

- 2視点からの画像をそれぞれ読み込む
- 出力画像を用意する
- E=とても大きな値
- for ループ=0～とても大きな値
  - success=0
  - for α=0～255
    - グラフの初期化
    - for 全画素
      - ノードの追加
      - ソースとシンクとのデータコストを設定
    - end for
    - for 全画素隣接点
      - ノードの追加
      - ソースとシンクとのスムーズコストを設定
      - 隣接する画素のノードとのスムーズコストを設定
    - end for
    - 最大流・最小カットアルゴリズムの適用
    - flow=求まったラベルで計算した総コスト関数
    - flow<Eのとき
      E=flow
      ノードが索子に属している画素=α
      success=1
    - グラフの消去
  - end for
  - success=0のときループ脱出
- end for
- 出力画像の保存

図 4: α 拡張のアルゴリズム

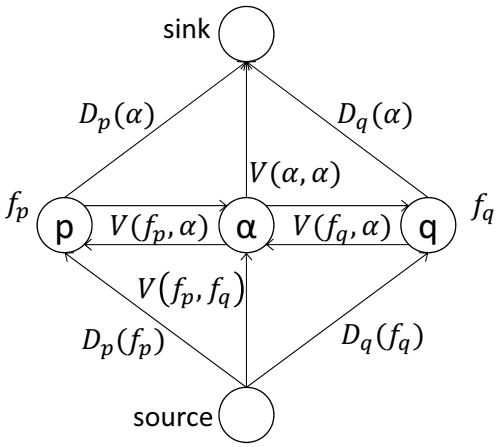


図 5: α 拡張のコストの設定

初期値の工夫

グラフカットでは出力画像の初期値を自由に設定できる．ここでは初期値に工夫を行うべきか検証を行う．図 6 に初期値を全て 0 にした結果を示す．図 7 に初期値を α 拡張によって得られたステレオ画像に設定した結果を示す．二つの結果には違いが見られない．よって今回は初期値には 0 を設定することする．

図 6: 初期値を 0 に設定

図 7: 初期値を α 拡張の結果に設定

データコストの工夫

データコストの設定について検証する．上述した式で示している通り，データコストの計算は SAD を用いている．これは SSD を用いた場合と比べ SAD を用いた方が結果が良かったためである．

スムーズコストの工夫

スムーズコストの設定について検証する．上述した式で示している通り，スムーズコストの計算には定数 c を用いている．この c を二画素間のエッジの強さによって違う値になるようプログラムを変更して結果を比較した．二画素間のエッジの強さ d は以下の式で求めた．

$$d = \| I_l(p) - I_l(q) \|$$

エッジの強さを考慮しなかったときの結果が図 8 である．エッジの強さを考慮したときの結果が図 9 である．エッジの強さを考慮したときの方が良い結果が得られていることがわかる．様々なウィンドウサイズで試した結果，今回作成したプログラムでの c の計算方法ではウィンドウサイズが大きいほどこの効果が大きく現れることがわかった．

図 8: エッジの強さを考慮しない

図 9: エッジの強さを考慮する

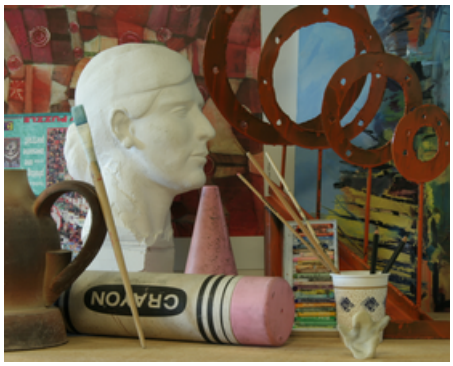
ステレオマッチング処理結果

α 拡張と階層グラフカットでのステレオマッチングの処理結果を比較する．

入力に用いる画像と出力画像の画像サイズは全て 278 × 222 である．

一つ目の対象の，左の視点からの画像と右の視点からの画像を図 10 に示す．これらの図を入力画像とし，ウィンドウサイズを変えてステレオマッチングを行った結果が図 11〜図 14 である．

(a) 左画像



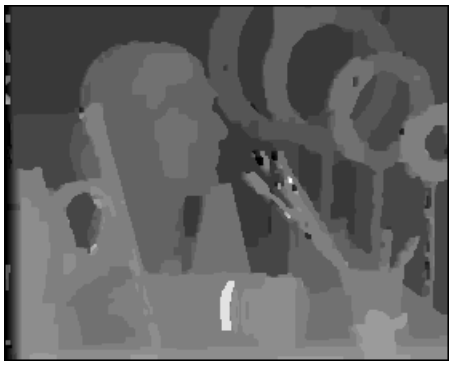
(b) 右画像

図 10: 画像 1:入力画像

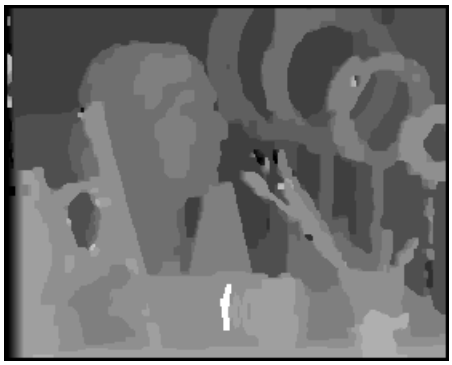
画像 1 の実行時間について以下に示す．ただし，W はウィンドウサイズである．

	α 拡張 [s]	階層グラフカット [s]
W = 1	36.530	15.141
W = 3	45.609	10.412
W = 5	39.419	13.462
W = 7	35.442	12.116

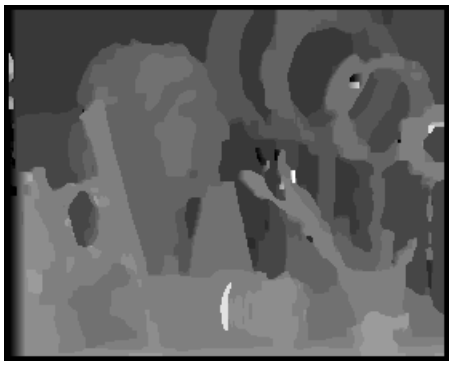

(a)α 拡張　　(b) 階層グラフカット
図 11: 画像 1:ウィンドウサイズ 1 処理結果




(a)α 拡張　　(b) 階層グラフカット
図 12: 画像 1:ウィンドウサイズ 3 処理結果




(a)α 拡張　　(b) 階層グラフカット
図 13: 画像 1:ウィンドウサイズ 5 処理結果




(a)α 拡張　　(b) 階層グラフカット
図 14: 画像 1:ウィンドウサイズ 7 処理結果

二つ目の対象の，左の視点からの画像と右の視点からの画像を図 15 に示す．これらの図を入力画像とし，ウィンドウサイズを変えてステレオマッチングを行った結果が図 16〜図 19 である．

(a) 左画像

(b) 右画像

図 15: 画像 2:入力画像

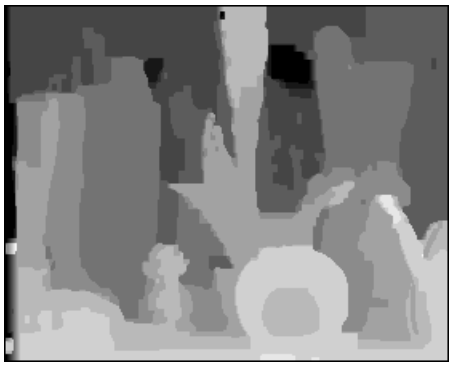
画像 2 の実行時間について以下に示す．

	α 拡張 [s]	階層グラフカット [s]
W = 1	45.236	13.652
W = 3	36.033	9.241
W = 5	36.033	12.301
W = 7	44.836	10.193

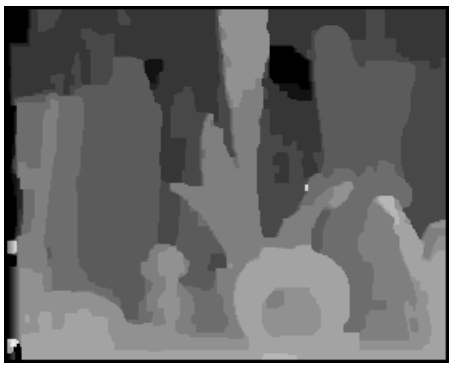
(a)α 拡張

(b) 階層グラフカット

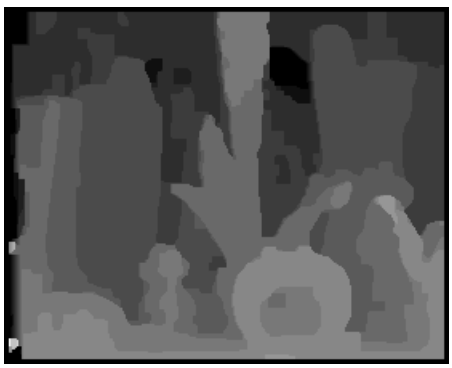
図 16: 画像 2:ウィンドウサイズ 1 処理結果




(a)α 拡張　　(b) 階層グラフカット
図 17: 画像 2:ウィンドウサイズ 3 処理結果




(a)α 拡張　　(b) 階層グラフカット
図 18: 画像 2:ウィンドウサイズ 5 処理結果




(a)α 拡張　　(b) 階層グラフカット
図 19: 画像 2:ウィンドウサイズ 7 処理結果

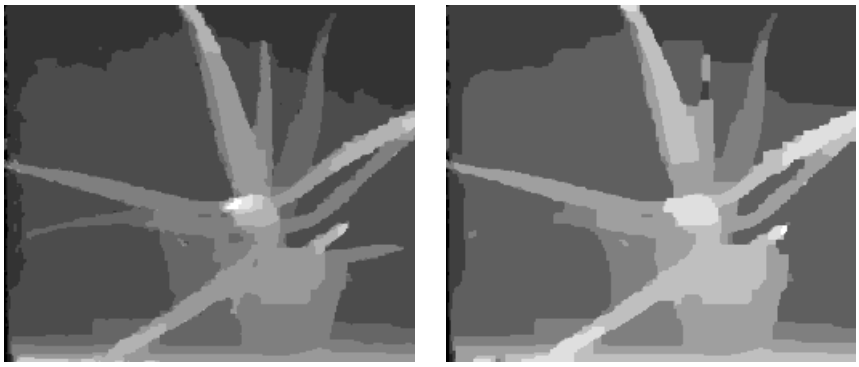
三つ目の対象の，左の視点からの画像と右の視点からの画像を図 20 に示す．これらの図を入力画像とし，ウィンドウサイズを変えてステレオマッチングを行った結果が図 21〜図 24 である．

(a) 左画像　　(b) 右画像

図 20: 画像 3:入力画像

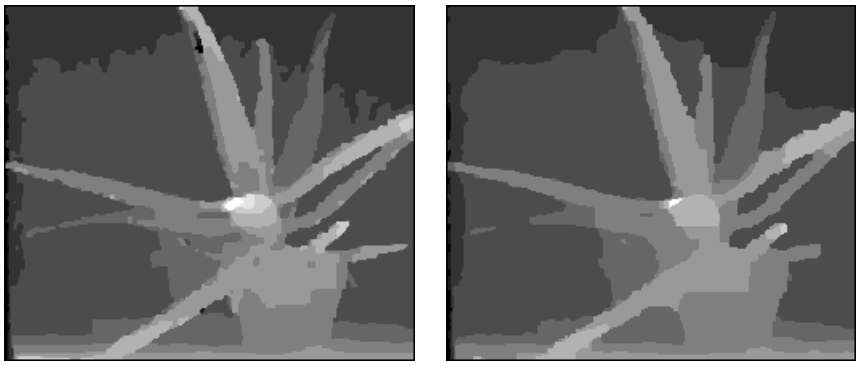
画像 3 の実行時間について以下に示す．

	α 拡張 [s]	階層グラフカット [s]
W = 1	32.303	10.829
W = 3	24.714	8.665
W = 5	21.382	7.562
W = 7	26.073	8.475



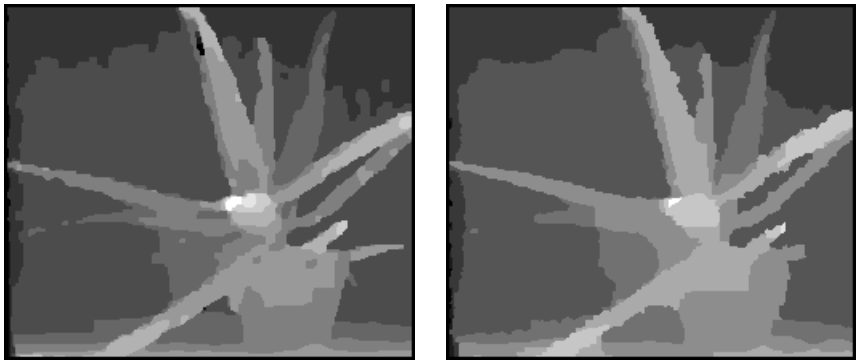
(a)α 拡張　　(b) 階層グラフカット

図 21: 画像 3:ウィンドウサイズ 1 処理結果



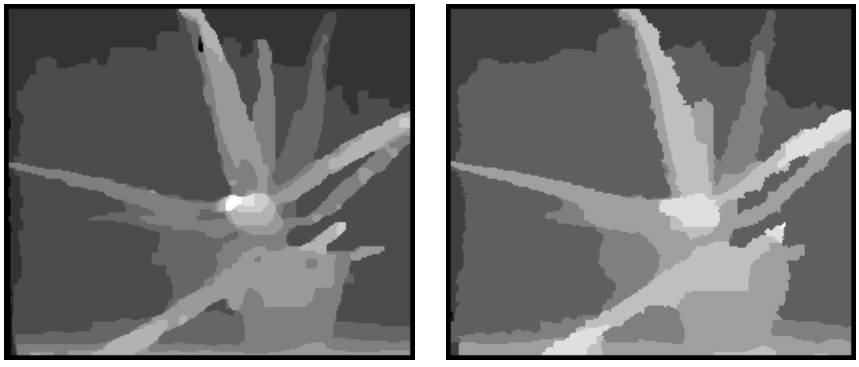
(a)α 拡張　　(b) 階層グラフカット

図 22: 画像 3:ウィンドウサイズ 3 処理結果



(a)α 拡張　　(b) 階層グラフカット

図 23: 画像 3:ウィンドウサイズ 5 処理結果



(a)α 拡張　　(b) 階層グラフカット

図 24: 画像 3:ウィンドウサイズ 7 処理結果

考察

どの画像のパターンでもα拡張より階層グラフカットの方が実行時間が短くなっている．大体3〜4倍階層グラフカットの方が速く処理を終えることができる．

処理結果については階層グラフカットの方が滑らかな結果画像となっている．α拡張では小さなノイズが乗りやすいが，階層グラフカットでは大きなノイズが乗りやすくなっている．

α拡張と階層グラフカットの両方で，ウィンドウサイズが小さい方が良い結果が得られた．

以上のことから，速く処理を行うには階層グラフカットの方が良い結果が得られるが，精度については更に工夫を加え改善していく必要があることと，今のプログラムではウィンドウサイズをなるべく小さくすれば精度が少し上がることがわかった．