広島市立大学 情報科学研究科 知能工学専攻
医用画像工学研究室
[redacted] 松本 尚
takashi@info.ime.hiroshima-cu.ac.jp
2015 年 1 月日提出

画像応用数学特論
12月11日出題レポート

階層グラフカットでステレオ

環境について

- OS：Linux Cent OS 5.5
- CPU：Intel Core i7 @ 3.20GHz
- メモリ：24GB
- コンパイラ：gcc バージョン 4.1.2
- プログラミング言語：C/C++

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．また，用いたデータコスト D とスムーズコスト V の計算式を以下の通りである．

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

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

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 W 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．$V(Bp, Bq) \leq V(Ap, Aq)$ 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

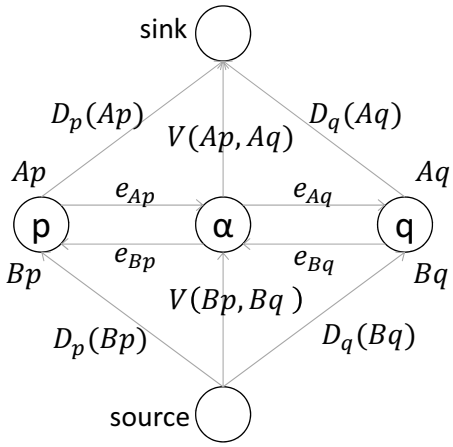
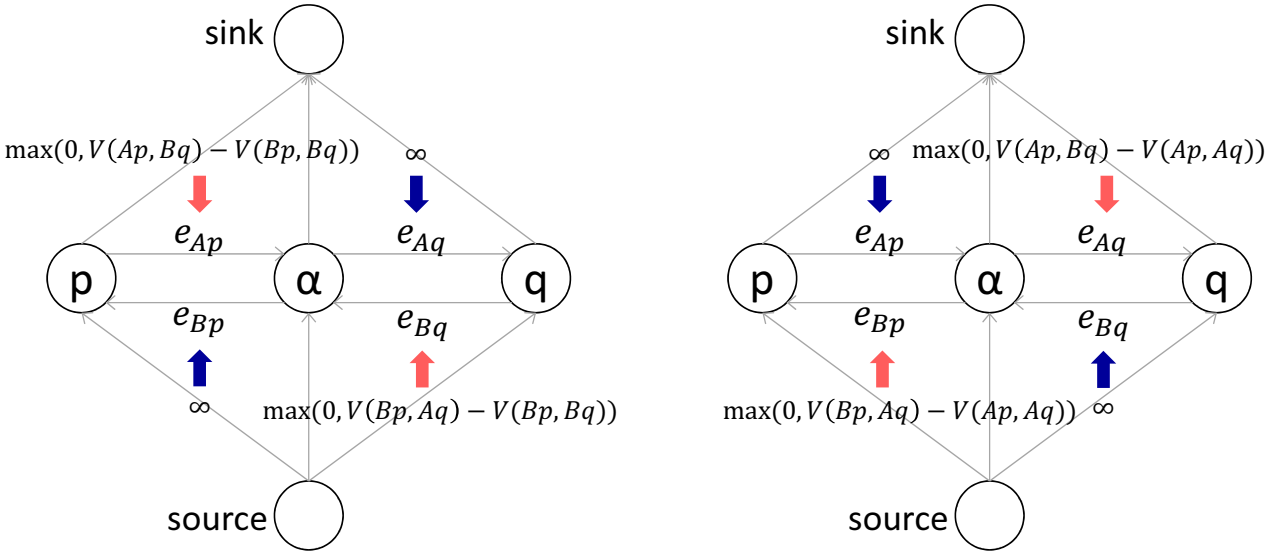


図 2: 階層グラフカットのコストの設定



(a)$V(Bp, Bq) \leq V(Ap, Aq)$ 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

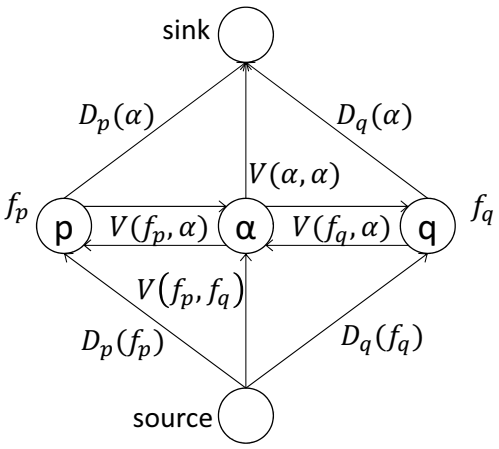


図 5: α 拡張のコストの設定

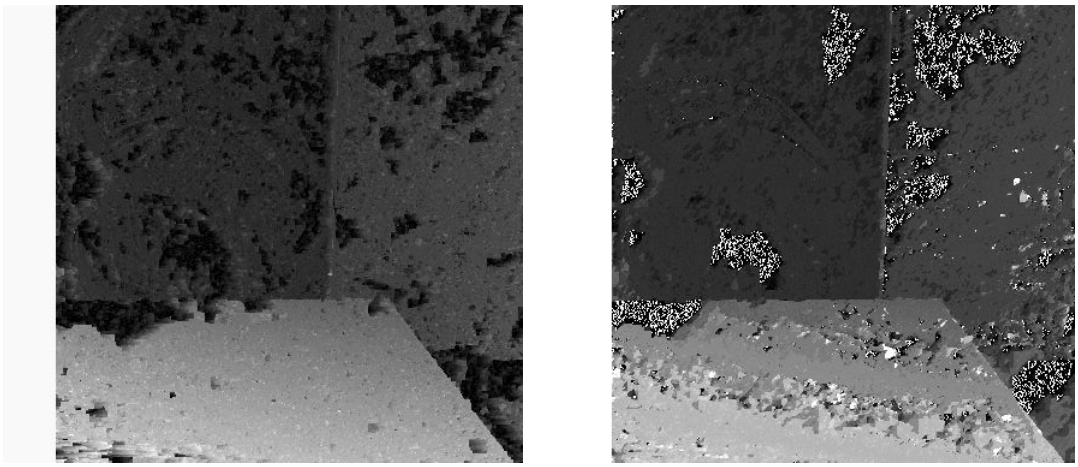
ステレオマッチング処理結果

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．2枚の入力画像と出力画像の画像サイズは全て 433×381 である．入力画像の左の視点からの画像と右の視点からの画像を図6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7　図11である．今回ウィンドウサイズ W は 1,3,5,7,9 の5種類である．

また，処理時間の比較も行った．その結果についても以下に示す．ただし，W はウィンドウサイズである．

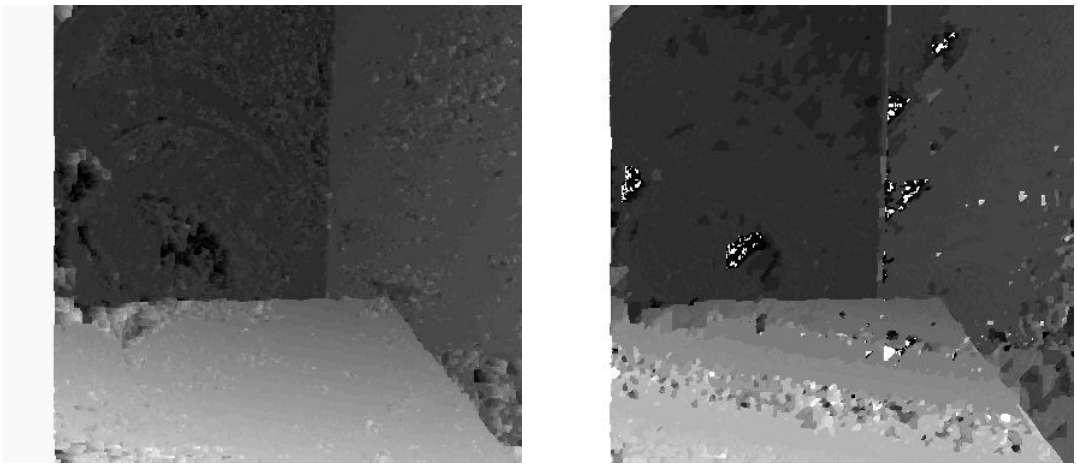
(a) 左画像　　(b) 右画像

図 6: 入力画像



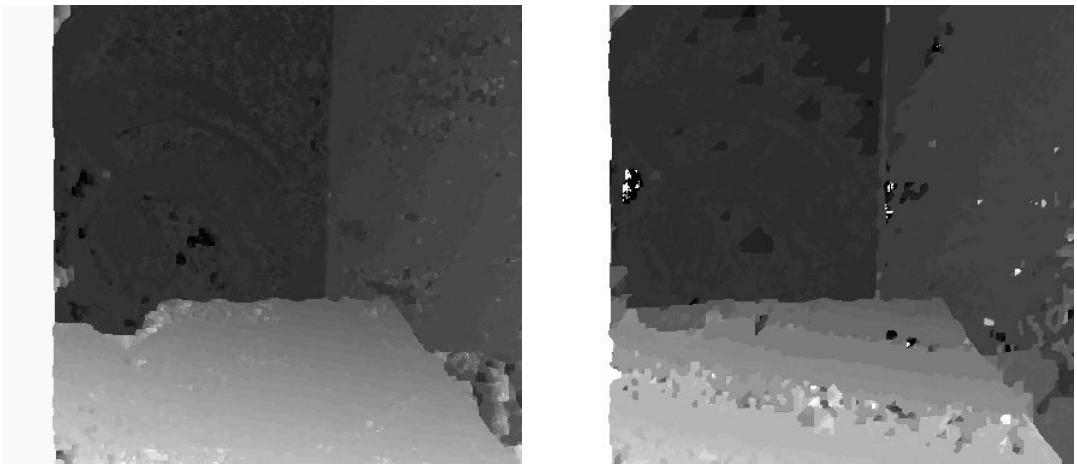
(a)α 拡張　　(b) 階層グラフカット

図 7: $W = 1$ の処理結果



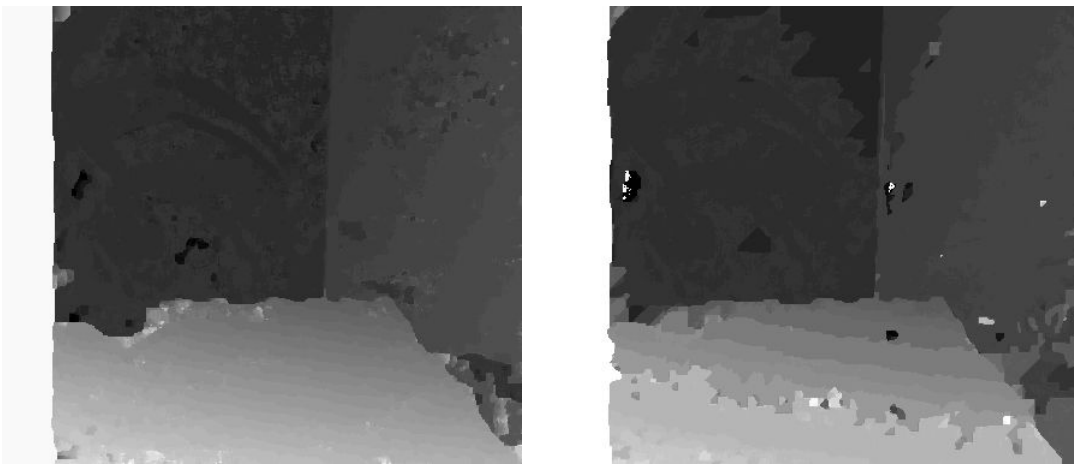
(a)α 拡張　　(b) 階層グラフカット

図 8: $W = 3$ の処理結果



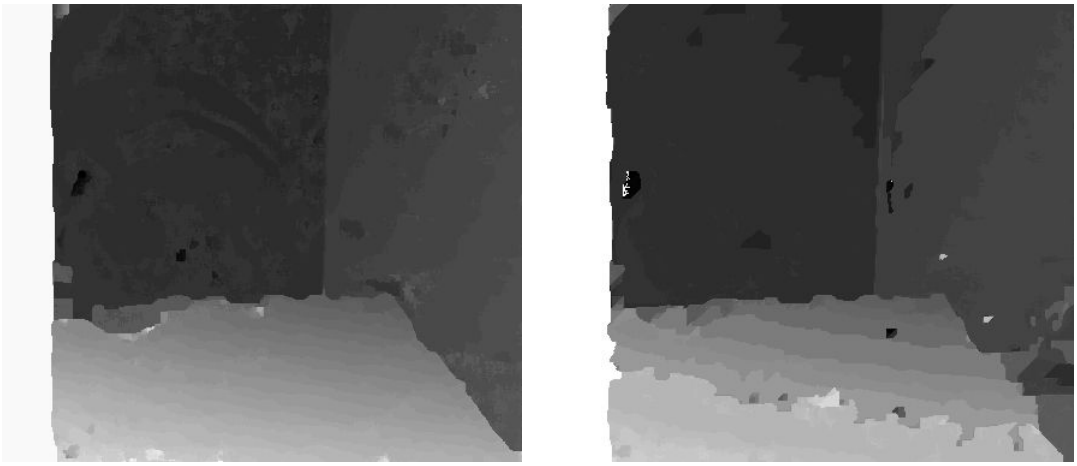
(a)α 拡張　　(b) 階層グラフカット

図 9: $W = 5$ の処理結果



(a)α 拡張　　(b) 階層グラフカット

図 10: $W = 7$ の処理結果



(a)α 拡張　　(b) 階層グラフカット

図 11: $W = 9$ の処理結果

	α 拡張 [s]	階層グラフカット [s]
W = 1	14.21	5.23
W = 3	11.99	6.45
W = 5	25.82	7.61
W = 7	40.74	11.2
W = 9	61.01	23.92

考察

どのウィンドウサイズのパターンでも α 拡張より階層グラフカットの方が処理時間が短くなっているのが分かる．大体 2　3 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大きい方がノイズのない滑らかな結果が得られた．

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，精度についてはコスト関数の工夫を加え改善していく必要があることが考えられる．今現在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分かった．