Vol.2013-CVIM-188 No.17
2013/9/2

協調的アルゴリズムによるステレオビジョン

福田 保彦[1,a)]　元島 晃伸[1]　小栗 清[1]　柴田 裕一郎[1]

概要：D. Marr は自身の著書において協調的アルゴリズムを用いてランダムドットステレオグラムの解読ができることを示したが，この方法は自然なステレオ画像においてうまく作動することは示されたことがないと述べている．本論文ではピクセル単位で計算を行い表面を検出する稠密ステレオビジョンにおいて，グラフカットの代わりに D. Marr の協調的アルゴリズムを用いても十分に正しい表面が検出できることを述べる．

1. はじめに

近年，従来の特徴ベース (feature-based) および領域ベース (area-based) のステレオビジョンに代わって，稠密ステレオビジョン (Dense Stereo Vision) の研究が盛んである．van der Mark らの研究 [1] ではインテリジェント車両 (Intelligent Vehicle: IV) のためのリアルタイムで動作する稠密ステレオのフレームワークについて検討を行なっている．このフレームワークでは対象ピクセルの周囲の SAD(Sum of Absolute Difference) の計算を行うことによってサブピクセル単位で視差を求める．また，Intel x86 アーキテクチャの CPU において実装されている SSE2(Streaming SIMD Extensions 2) を用いることによって CPU 上で高速に処理を行うことができる．このように稠密な距離データを高速に得ることは IV に限らず様々な応用分野で重要になっていくものと考えられる．

以上で述べた [1] は視差を求めるための手法であり，対応という概念が残っている．しかしながら，対応点を求めることなしに 3 次元空間上の位置を求めることは可能である．小栗らのステレオビジョンの研究 [2] では表面存在仮説点 (HSP: Hypothetical Surface Point) と呼ばれる点を三次元空間上に配置し，その点に表面が存在するもっともらしさ (尤度) をその点に対応する左右の単一ピクセルの輝度差によって与える．次に，表面存在仮説点の集合をグラフとして表現し，そのグラフの最小カットを求めること (グラフカット) によって物体の表面を検出することができるということを示した．また，他にもグラフカットによって表面を検出する研究として Vogiatzis らの研究 [3] が挙げられる．Vogiatzis らのマルチビューステレオの研究では空間上をボクセル (voxel) に分割し，ボクセルの集合をグラフとして表現してグラフカットを行い物体表面を検出する．これらの研究は 2 次元画像上の処理ではなく，3 次元空間上における処理で物体の表面を得ていることでは一致している．

ここで，これら 2 つの研究における立体の検出の高速性について考える．小栗らの研究においては表面存在仮説点 $129 \times 129 \times 81$ 個から構築されるグラフのカットに 3 秒を要し，Vogiatzis らの研究においては空間を $560 \times 460 \times 360$ のボクセルに分割し，36 枚の画像を使用したとき，およそ 40 分を要する．小栗らの研究で用いられた Kolmogorov らのグラフカットアルゴリズム [4] の最悪の場合の時間計算量は n をノードの数，m をエッジの数，そして $|C|$ を最小カットのコストとすれば，$O(mn^2|C|)$ である．また，ステレオで用いられるグラフでは $m \propto n$ であることが多いため，$O(n^3|C|)$ と考えても良い．一方で，Kolmogorov らは時間計算量は Ford-Fulkerson のアルゴリズムや Push-Relabel アルゴリズム [5] と比較して小さいわけではないが，ビジョンの分野で用いられるような疎グラフ (sparse graph) ではそれらのアルゴリズムより高速に処理を行うことができると述べている．しかしながら，これらは検出された物体の緻密さにおいては有利な手法であるが，依然として処理のリアルタイム性の側面から考えると不利な手法であると言える．例えば，動画像では 30fps 以上のフレームレートであるものが多い．表面検出をリアルタイムで処理することを考えると，フレーム間の時間間隔はフレームレートを 30fps とすると $1/30 = 33[ms]$ であるから，それ以下の時間で処理を行わなければならないことになる．以上で述べた 2 つの研究はいずれもこの制約を満たさない．

リアルタイムに処理を行うためには，より時間計算量の小さいアルゴリズムを用いてグラフをカットする必要がある．

[1] 長崎大学大学院工学研究科
[a)] bb52113327@cc.nagasaki-u.ac.jp

Ford-Fulkerson のアルゴリズム，Push-Relabel アルゴリズム，Kolmogorov らのグラフカットアルゴリズムなどではグラフの最小カットが求まることが保証されている．しかしながら，必ずしも最小カットが求まる必要はないのではないだろうか．最小カットが求まることが保証されていなくとも，より高速にグラフのカットを求めることができれば，IV はもとより様々な応用が可能になるはずである．そこで，本研究では協調的アルゴリズムと呼ばれるアルゴリズムを用いてグラフのカットを行う．協調的アルゴリズムについては，D. Marr の著書 [6] でランダムドットステレオグラムを解く方法として紹介されている．そして，D. Marr はランダムドットステレオグラム以外の自然な画像に対して作動することは示されたことがないと述べた．しかしながら，ピクセル単位で計算を行い表面を検出する稠密ステレオビジョンにおいて協調的アルゴリズムがグラフカットアルゴリズムに対してどの程度有効であるかをもう一度確認することには意味があるものと思われる．このために複数の画像に対してグラフカットの最小カットと協調的アルゴリズムで得られるカットと比較した．また，協調的アルゴリズムは並列化に向いたアルゴリズムであるため，マルチコアプロセッサや GPU(Graphics Processing Unit) への実装はもとより FPGA (Field Programmable Gate Array) などのハードウェアへの実装も容易である．このことが本研究の動機となっている．

本論文の構成は以下の通りである．第 2 章でグラフカットおよび協調的アルゴリズムで解くべき共通のグラフについて述べる．そして，第 3 章では協調的アルゴリズムの動作および計算量について述べる．さらに，第 4 章では本研究の実験の結果について述べ，第 5 章では本研究の考察および今後の課題について述べる．

2. 処理対象となるグラフ

2.1 表面存在仮説点

カットの対象となるグラフは 3 次元空間上に配置された表面存在仮説点の存在尤度から求めることができる．また，表面存在仮説点はホロプタ (Horopter) と呼ばれる**図 1** のような左右のレンズの中心と原点を通る円の上に配置されなければならない．そして原点上にある表面存在仮説点から放たれた光は左右のカメラの撮像面の中心，すなわち左右の画像の中心のピクセルに届くようにする．ここでは 2 つのカメラの撮像面は球面上にあると仮定する．ここでホロプタが空間上にどのように定義されているかを説明する．まず，議論を簡単にするためにホロプタ座標 $\boldsymbol{H}$ を定義しておく．$\boldsymbol{H}$ は $a, b, c \in \boldsymbol{Z}$(整数) とすると以下の式 1 のように定義される．

$$\boldsymbol{H}(a,b,c) = \{\boldsymbol{v} = (u,v,w) \in \boldsymbol{Z}^3 \mid u \in [-a,a], v \in [-b,b], w \in [-c,c]\} \quad (1)$$

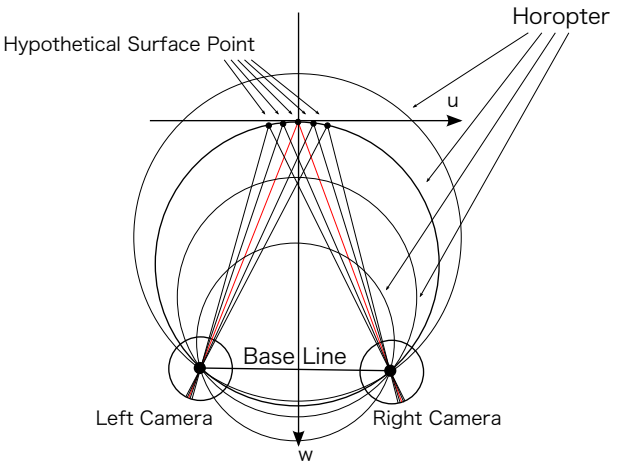


図 1 ホロプタ
Fig. 1 Horopter

また，ホロプタ座標を定義できたので，表面存在仮説点の集合 S は以下の式 2 のように定義することができる．

$$S = \{s(\boldsymbol{v}) \mid \boldsymbol{v} \in \boldsymbol{H}\} \quad (2)$$

ホロプタ座標とは表面存在仮説点がホロプタ上のどの位置にあるのかを示すものである．座標値は整数でかつ範囲が決められている．u はホロプタ上の位置を表しており，u の値はホロプタ上を半時計回りに進むほど小さくなり，時計回りに進むほど大きくなる．v はホロプタの高さを表していて，v の値が大きいほど高い位置を表している．w はホロプタの奥行きである．カメラに近いホロプタほど w は小さい値となる．逆にカメラから遠いホロプタは値が大きくなる．ここで例を一つ示すと原点上にある表面存在仮説点のホロプタ座標は $(0,0,0)$ である．

まず，原点を通るホロプタ上の表面存在仮説点，つまり $w = 0$ の場合についてであるが，例えば，$u < 0$ を満たす表面存在仮説点から放たれた光は左のカメラでは画像の中心より右のピクセルに届き，右のカメラでは画像の中心より左のピクセルに届く．$u > 0$ の場合だと，放たれた光は左のカメラでは画像の中心より左のピクセルに届き，右のカメラでは画像の中心より右のピクセルに届くことになる．次に $(0,0,-1)$ の表面存在仮説点について考える．放たれた光は左のカメラでは画像の中心から 1 ピクセル左に届き，右のカメラでは画像の中心から 1 ピクセル右に届くことになる．逆に $(0,0,1)$ であれば，左のカメラでは画像の中心から 1 ピクセル右に届き，右のカメラでは画像の中心から 1 ピクセル左に届く．さらに v 軸について考えると，$(0,1,0)$ では左右とも画像の中心から上に 1 ピクセルずれた場所に届く．

端的に言うと，表面存在仮説点から放たれた光は左右のカメラの決まったピクセルに届くということが予めわかっているため，それらのピクセル同士の類似度から表面存在仮説点の存在尤度を求めることができる．

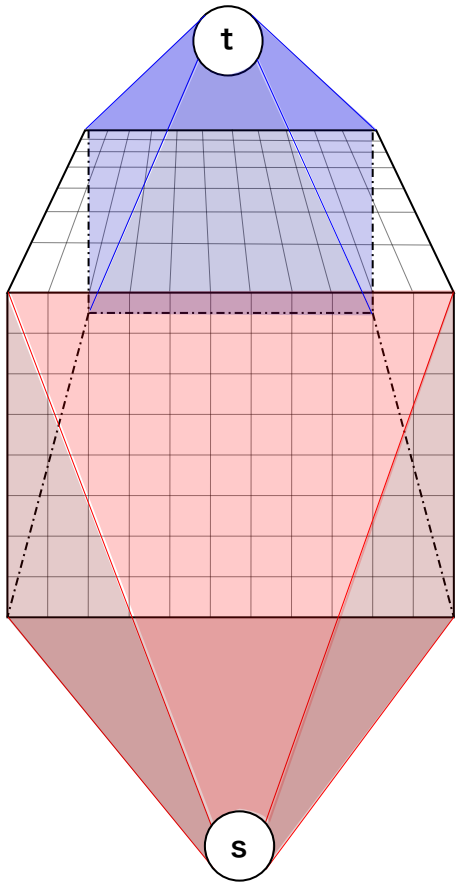


図 2 表面存在仮説点の尤度から生成されたグラフ
Fig. 2 A graph which is generated by likelihood of HSPs

2.2 グラフの生成

各表面存在仮説点の尤度はグラフのエッジのコストとして表すことができる．ノードの集合 V およびエッジの集合 E で構成される有向グラフを $G(V,E)$ とする．$G(V,E)$ は**図 2** のように 3 次元のグリッドグラフとなる．表面存在仮説点の尤度はエッジのコストとして表現される．これらはソースからシンクへの方向に平行なエッジである．ソースは一番手前のホロプタ，つまりホロプタ座標において $w=-c$ を満たす表面存在仮説点を表すエッジを介して図 2 の最も手前の面のノードに接続している．また，シンクは $w=c$ を満たす表面存在仮説点を表すエッジを介して図 2 の最も奥の面のノードから接続されている．そしてさらにグラフにソースからシンクへの方向に対して垂直なエッジをソースとシンクを除いた各ノード間に導入する．それらのエッジのコストはペナルティ (定数) であり表面の激しい凸凹を抑制する働きをもつ．このグラフをカットすると，カットの境界は面となる．小栗らの研究 [2] ではこの面を物体の表面として利用する．

ホロプタ座標 $\boldsymbol{H}(1,0,2)$ とグラフがどのように対応しているかを示す．まず，$\boldsymbol{H}(1,0,2)$ のホロプタは**図 3** のように描ける．このとき，各表面存在仮説点の尤度は既に分かっているものとする．図 3 のホロプタをグラフとして表

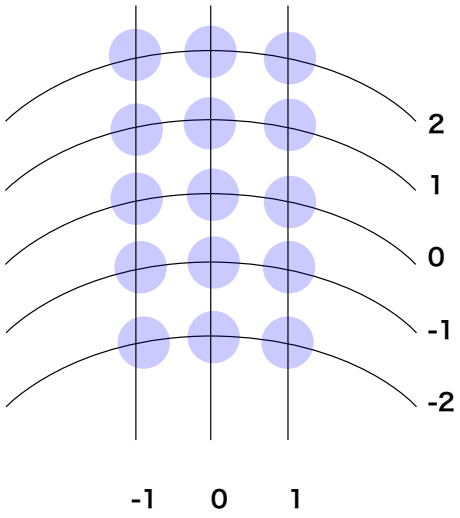


図 3 ホロプタの例
Fig. 3 An example of Horopter

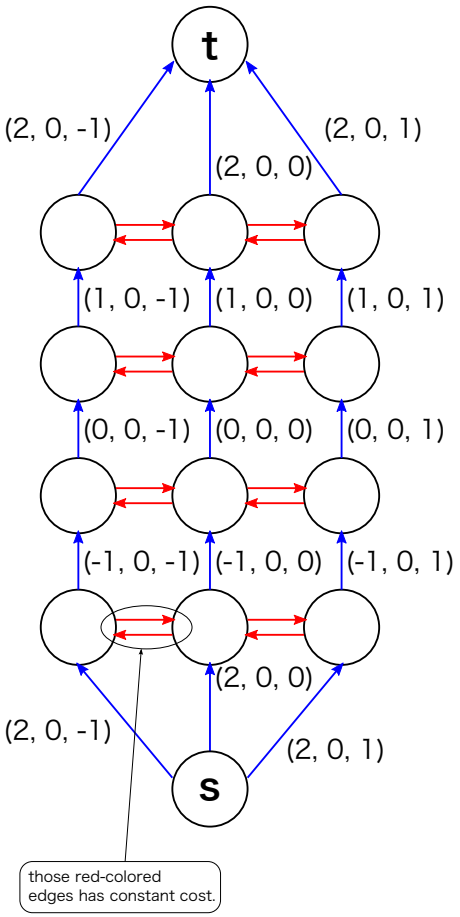


図 4 図 3 のホロプタのグラフ表現
Fig. 4 A graphical representation of the Horopter

現すると**図 4** のようになる．図 4 のグラフでは以上で説明したようにソースからシンクへの方向に対して平行なエッジがホロプタ上の表面存在仮説点の尤度をグラフのコストとして表していて，ソースからシンクの方向に対して垂直

なエッジがペナルティのコストとなる．

3. 協調的アルゴリズム

3.1 協調的アルゴリズムによるグラフのカット

本研究ではグラフのカットに協調的アルゴリズムを用いる．具体的には既にグラフのカット (最小カットではない) が与えられている場合を考える．グラフにおけるカットされたエッジとその周辺のエッジから評価関数を求め，その評価関数に従って局所的にカットを変更する．この操作をグラフ全体に適用し，カットのコストが収束するまで反復する．このアルゴリズムを擬似コードとして表すと以下のようになる．

```
initialize surface_to_cut;
while (cost of cut is changing) {
    for (e in E_surface) {
        if (φ_next(e) < φ_prev(e)) {
            move(surface_to_cut);
        }
    }
}
```

最初に与えるグラフのカット (以下初期カットという) は任意のものでかまわないが，カットの選び方によって最終的に到達する解が異なるため，慎重にそれを検討する必要がある．本研究では平面および以下に説明するカットで実験を行った．まず，ホロプタの奥行き方向で最も尤度が高いもの，つまりホロプタ座標 (u, v, z) において u, v を固定し，w のみを変化させた時の尤度の最大値を返す 2 引数関数 $max(u, v)$ を定義する．このとき，すべての u, v で $w = max(u, v)$ を計算し，(u, v, w) でカットを行う．例として図 3 のホロプタにおいて，**図 5** のように尤度が定義されているとする．そのときのグラフのカットは**図 6** のようになる．図 6 でわかることは，表面が凸凹しているほどソースからシンクへの方向に対して垂直なエッジ (以下ペナルティエッジと呼ぶ) を多く通らなければならないということである．ペナルティエッジを多く通るということはそれだけカットのコストが増大するという事であるから，カットのコストを小さくするためにはペナルティエッジをできる限り通らないようにすることが重要である．したがってグラフカットで最小カットを求めることは表面の凸凹を抑制することにつながることが分かる．

協調的アルゴリズムで表面の凸凹を抑制するためには，適切な評価関数を定める必要がある．

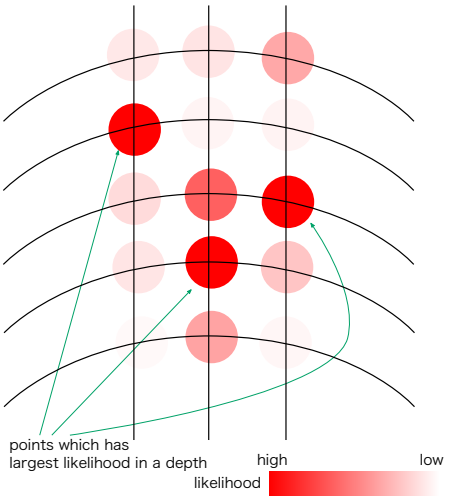


図 5 ホロプタ上の表面存在仮説点の尤度
Fig. 5 Likelihood of HSPs in the Horopter

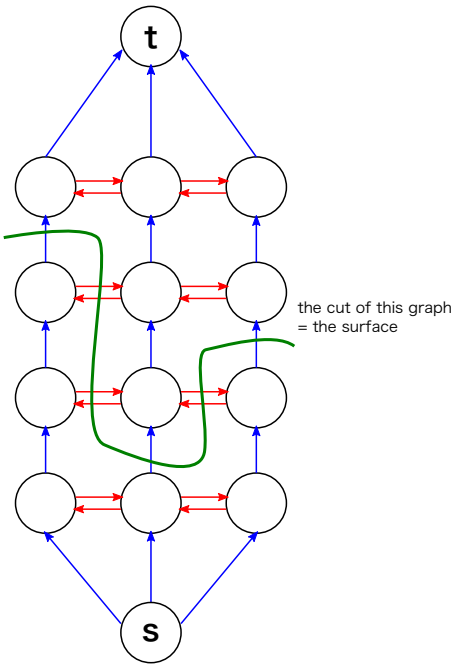


図 6 図 5 におけるグラフのカット
Fig. 6 a cut of the graph

今回用いた評価関数は

$$\phi_1(\boldsymbol{v}) = p\left\{\sum_{\boldsymbol{n}\in S_n(s(\boldsymbol{v}))} |s(\boldsymbol{n}) - s(\boldsymbol{v})|\right\} + i_1(s(\boldsymbol{v}), s(\boldsymbol{n})) + c(s(\boldsymbol{v})) \quad (3)$$

$$\phi_2(\boldsymbol{v}) = p\left\{\sum_{\boldsymbol{n}\in S_n(s(\boldsymbol{v}))} \{s(\boldsymbol{n}) - s(\boldsymbol{v})\}^2\right\} + i_2(s(\boldsymbol{v}), s(\boldsymbol{n})) + c(s(\boldsymbol{v})) \quad (4)$$

の 2 つである．まず，ϕ_2 を評価関数としてカットの変更を繰り返す．この評価関数 ϕ_2 は表面をなめらかにする性質をもっている．カットのコストが収束すれば，次は ϕ_1 を評価関数としてカットの変更を繰り返す．その時カットのコストが収束した時点でアルゴリズムは終了となる．ここで $S_n(s(\boldsymbol{v}))$ は近傍を表す表面存在仮説点の集合で，

$$\begin{aligned} S_n(s(\boldsymbol{v})) = \{&s(u-1, v, max(u-1, v)),\\ &s(u+1, v, max(u+1, v)),\\ &s(u, v-1, max(u, v-1)),\\ &s(u, v+1, max(u, v+1))\} \end{aligned} \quad (5)$$

となり，p はペナルティエッジのコストを示している．関数 $i_n(s(\boldsymbol{v}), s(\boldsymbol{n}))$ は，左右のカメラから見えるはずのない表面の傾きを禁止するための関数である．大きな傾きに対してコストを付加することによって，傾きを小さくする方向への動きを促す働きをもつ．$\boldsymbol{v} = (v_u, v_v, v_w)$，$\boldsymbol{n} = (n_u, n_v, n_w)$ とし，q を禁止エッジの大きさとすると以下のように表される．

$$i_n(s(\boldsymbol{v}), s(\boldsymbol{n})) = \begin{cases} q|n_w - u_n|^n & (|n_w - v_n| > d) \\ 0 & (otherwise) \end{cases} \quad (6)$$

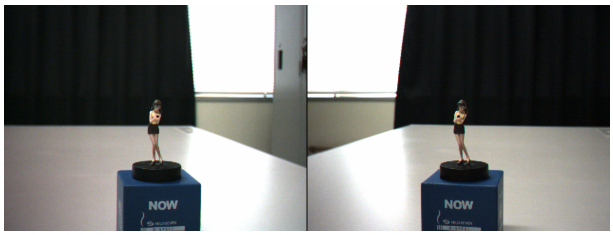
そして，$c(s(\boldsymbol{v}))$ は表面存在仮説点 $s(\boldsymbol{v})$ の尤度から導出されたグラフにおけるコストである．この評価関数を $s(u, v, max(u, v))$，$s(u, v, max(u, v) - 1)$，$s(u, v, max(u, v) + 1)$ で計算し，評価関数の小さい方を新しいカットとする．

3.2 計算量

本手法の計算量であるが，グラフ全体を見て計算を行うのではなく，あくまで事前に与えられたカット，つまり検出された表面の周辺のみを見て計算を行うので計算量はホロプタ座標 $\boldsymbol{H}(a, b, c)$ のとき反復回数を n とすれば，$O(nab)$ となる．対して Kolmogorov らのグラフカットアルゴリズムの最悪の場合の時間計算量は $O(mn^2|C|)$ である．ホロプタ座標 $\boldsymbol{H}(a, b, c)$ とすれば，計算量は $O((abc)^3|C|)$ とすることができる．

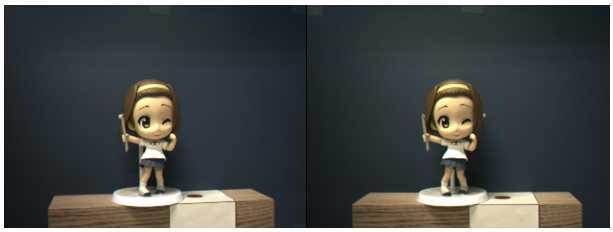
4. 実験

本研究における実験について以下に述べる．



(a)

(b)



(c)

図 7 実験で用いた画像の一部
Fig. 7 A part of an image which was used in this experiment

4.1 実験の概要

実験では小栗らによるグラフカットによる表面検出 [2] と本研究における協調的アルゴリズムを用いた手法の 2 つを 43 組のステレオ画像で比較した．比較の対象は処理時間とカットのコストである．加えて，協調的アルゴリズムでは最初に与えるカットによる解の違い，および反復回数の検証を行った．今回の実験で用いた画像の一部を**図 7**(a),(b),(c) に示す．

4.2 実験環境

実験で用いたマシンは以下の通りである．

- CPU: Intel Core i7 920
- メモリ: 12GB DDR3
- OS: Ubuntu 12.04 LTS (64bit)

4.3 実験結果

4.3.1 カットのコスト

ホロプタの奥行き方向で最も尤度の高い表面存在仮説点を選択して初期カットを求めたときのカットのコストの推移を**図 8** に示す．図 8 は反復回数を変化させた時にカットのコストがどのように変化するかを表したグラフである．グラフ中では 43 組のステレオ画像の中から 10 組を選択

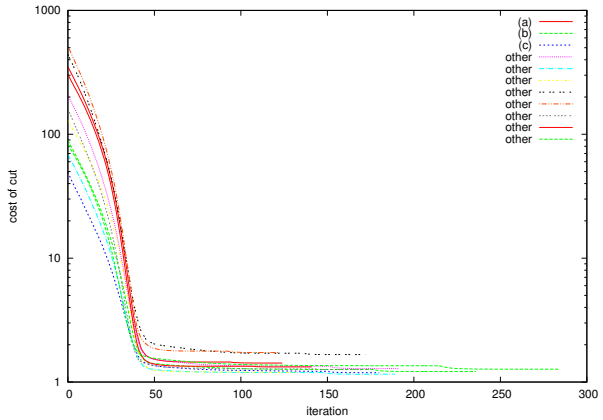


図 8 カットのコストの推移 (奥行き方向で最も尤度の高い点を選択)
Fig. 8 The transition of the cost of cut (best likelihood in depth)

し，それらのカットのコストの推移を表示している．また，うち 3 組は図 7 に示された画像である．このグラフの横軸は反復回数であり，縦軸は各画像の最小カットのコストを 1 としたときのカットのコストである．図 8 を見ると，初期カットでは最小カットのコストから 40-500 倍のコストであったが，急激に減少し，50 ステップを過ぎるとほとんどの画像において 2 倍程度となった．さらにカットのコストが収束するまで続けると，1.2 倍程度までカットのコストが小さくなっていることが分かった．カットのコストが収束した直後の図 7 (a),(b),(c) の処理結果を**図 9** に示す．

図 9 では 1 つの検出された表面を 2 つの方向から眺めている図である．奥行き方向における尤度の高さの情報を使うことによって，ある程度表面を検出できることが分かった．しかし，背景部分が波打つような曲面に変形している点がグラフカットで検出される面とは異なり，改善が必要である．

一方で，初期カットが平面の場合は**図 10** のような推移になる．初期カットのコストはほとんどの画像において 2 倍未満であったものの，反復回数が大きくなってもコストはほとんど減少していないことが分かった．そして収束するまでの反復回数は初期カットで奥行き方向で最も尤度の高い点を選択する場合と比較して小さくなっていることが確認できた．また，カットのコストが収束した直後の図 7 (a),(b),(c) の処理結果を**図 11** に示す．

平面から協調的アルゴリズムの操作を始めるとステレオ画像の左右が重なっているように見えたり，表面が検出できていないことが分かった．また，図 9 では表面を検出できていることから，協調的アルゴリズムでは単一ピクセルの比較からほぼ正しい表面が得られる場合以外では正しい表面を検出することは難しいものと考えられる．

4.3.2 処理時間

43 組の画像における協調的アルゴリズムおよびグラフカットの処理時間の平均および標準偏差を初期カットの設定ごとに**表 1**，**表 2** に示す．単位はすべて sec である．加えて，協調的アルゴリズムでは収束ステップ数の平均および標準偏差についても表記してある．

表 1，表 2 によると，初期カットにおいて奥行き方向で最も尤度の高い点を選択する場合にグラフカットに対して約 60 倍の速度向上が見られた．初期カットで平面を与える場合はさらに高速になり，約 170 倍の速度向上となった．

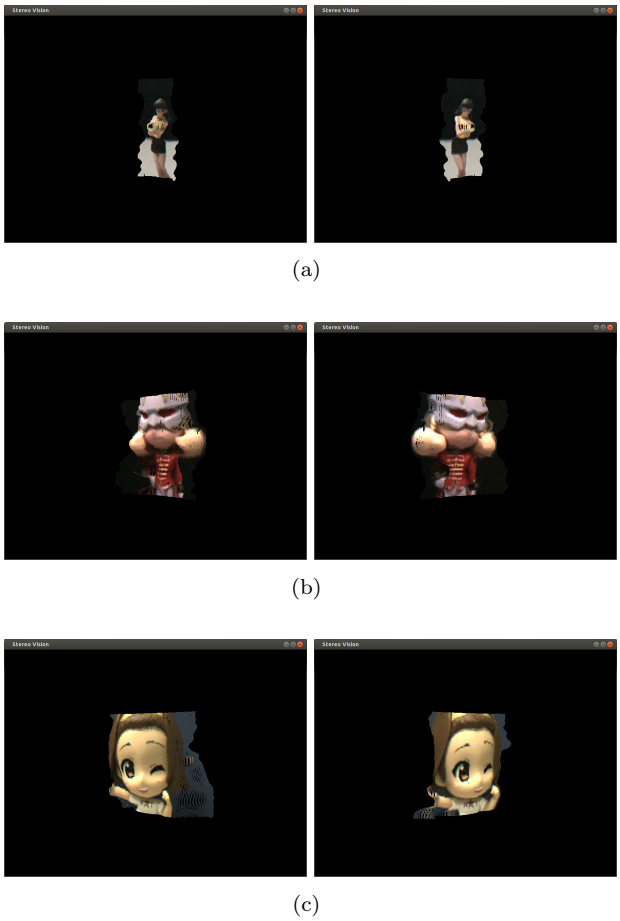


図 9 処理結果 (奥行き方向で最も尤度の高い点を選択)
Fig. 9 Results of the processing (best likelihood in depth)

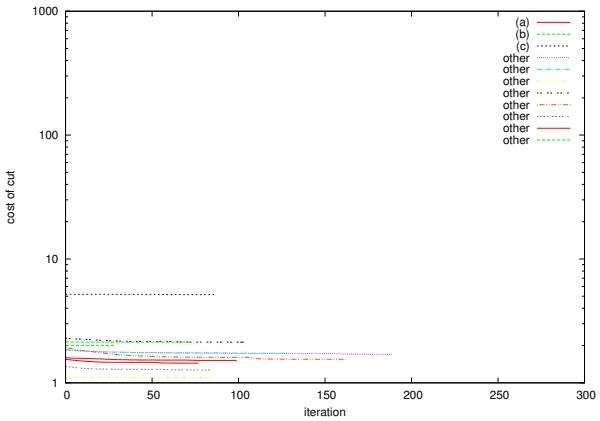
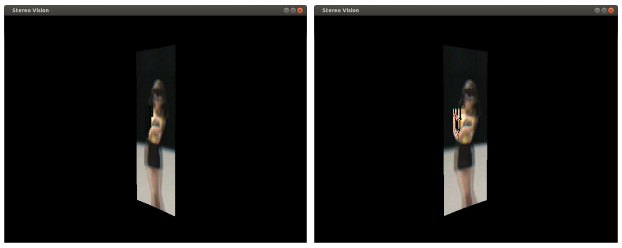
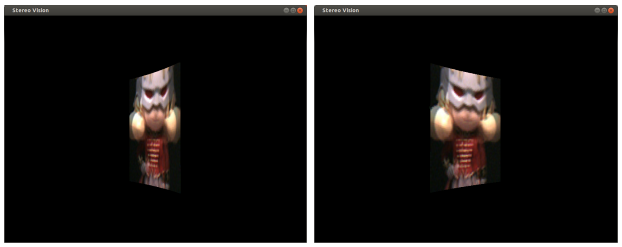


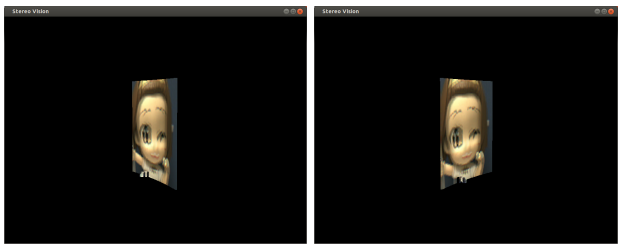
図 10 カットのコストの推移 (平面)
Fig. 10 The transition of the cost of cut (planar surface)



(a)



(b)



(c)

図 11 処理結果 (平面)
Fig. 11 Results of the processing (planar surface)

表 1 処理時間 (奥行き方向で最も尤度の高い点を選択)
Table 1 processing time (best likelihood in depth)

	協調的アルゴリズム	グラフカット
平均	0.462 (177.2 steps)	27.4243
標準偏差	0.136 (54.76 steps)	24.9659

表 2 処理時間 (平面)
Table 2 processing time (planar surface)

	協調的アルゴリズム	グラフカット
平均	0.163 (73.47 steps)	27.60
標準偏差	0.136 (50.71 steps)	25.11

5. まとめ

本研究では，ステレオビジョンにおける 3 次元空間中の表面検出を協調的アルゴリズムによって行う手法をグラフカット手法と比較した．協調的アルゴリズムでは事前に与えられた表面をグラフのカットとして表現し，その表面の一部を評価関数に従って局所的に変更していくことでよりよい表面を検出する．協調的アルゴリズムによる表面検出はグラフカットによるものと比較して 1/10 以下の時間で行えることが分かった．しかし，検出された表面は前景部分においてあるはずのない突起が見られたり，背景の乱れなど多少誤った部分が見られた．今後の課題としては Simulated Annealing(SA) 手法の導入やグラフカットの協調的並列動作実装を行いたい．

参考文献

[1] van der Mark, W. and Gavrilla, D. M.: Real-time dense stereo for intelligent vehicles, *IEEE Trans. Intelligent Transportation Systems*, Vol. 7, No. 1, pp. 38–50 (2006).
[2] 小栗　清，柴田裕一郎，濱田　剛：対応という概念を使わないステレオビジョン，CVIM 研究報告 (2013).
[3] Vogiatzis, G., Torr, P. H. S. and Cipolla, R.: Multi-view stereo via volumetric graph-cuts, *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 2, pp. 391–398 vol. 2 (2005).
[4] Boykov, Y. and Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 26, No. 9, pp. 1124–1137 (2004).
[5] Goldberg, A. V. and Targan, R. E.: *Journal of the Association of Computing Machinery*, Vol. 35, No. 4, pp. 921–940 (1988).
[6] Marr, D.: ビジョン 視覚の計算理論と脳内表現，産業図書 (1987).