

魚眼ステレオカメラを用いた路面平面と障害物高さ推定手法の GPU 実装

GPU Accelerated Estimation Method of Road Surface Shape and Object Height Using Fisheye Stereo Camera

佐久田朝優 (中大) 〇学 新井健斗 (中大)
Sarathak Pathak (中大) 正 梅田和昇 (中大)

Tomoyu SAKUDA, Chuo University, sakuda@sensor.mech.chuo-u.ac.jp
Kento ARAI, Chuo University
Sarathak PATHAK, Chuo University
Kazunori UMEDA, Chuo University

In this paper, we implement a novel method to estimate road surface shape and object height using a fisheye stereo camera on GPU for acceleration. Our method consists of three parts, preprocessing, road surface shape estimation and object clustering. Among them, the road surface shape estimation algorithm takes 20 seconds to process because the algorithm is recursive. This is not feasible for real application. Therefore, we change the algorithm to be able to be processed in parallel. We conducted experiment to confirm the difference between the accuracy and the processing time between the previous CPU implementation and the proposed GPU implementation of the approach. It was found that the proposed GPU implementation resulted in 8-times faster processing with no loss of accuracy.

Key Words: Fish-eye Stereo Camera, Autonomous Vehicle, Image Processing, Object Detection, Drive-able Area

1 緒言

近年、運転支援システムに関する技術の研究や開発が盛んに行われている。運転支援システムでは、LiDAR やステレオカメラなどの距離センサにより計測された周囲の 3 次元情報をもとに運転を支援する。必要な支援が必要か判断をする際には車周囲の環境を把握する必要がある。車の周囲環境を把握する為に使用されている距離センサの中でも特にステレオカメラは、色情報を取得可能であることや、高い計測密度で計測が出来ることから周囲環境を詳しく把握する為に用いられており、様々な環境把握の手法が提案されている [1]~[6]。

ステレオカメラを用いた環境把握の手法は幾何学的な条件や 3 次元点群を利用して路面領域を推定し障害物領域を抽出する手法と単眼カメラの色情報から障害物の位置を検知する手法が用いられる。路面領域を推定し、障害物を抽出する手法として射影行列を用いた手法 [1] や uv-disparity を用いる手法 [2][3] が上げられる。しかし、これらの手法は前提としている路面形状への制約が強く、路面の傾斜の変化などに対応していない。色情報から障害物を抽出する手法では主に深層学習を用いた手法が提案されている [4]~[6]。しかし、これらの手法では手法の精度が学習データに依存してしまうことや不具合の原因を知ることが難しく説明可能性が低いことが課題である。また、これらの手法 [1]~[6] に用いられている一般的なステレオカメラは視野角が狭く、周囲環境の把握が可能な領域が狭いことも課題である。

我々は魚眼ステレオカメラという広範囲の撮影が可能な距離画像センサを使用し、距離画像にたいして複数平面をフィッティングすることにより路面形状の制約が少ない路面平面と障害物高さの推定手法を提案した [7]。しかし、この手法は処理時間が約 22 s かかってしまい、周囲環境から支援の判断までの時間としての応答面で課題が残されている。そこで、本研究では CUDA による GPU 実装を行い、提案手法の高速化を試みる。

2 提案手法概要

提案手法の概要を図 1 に示す。手法は 3 次元計測、路面平面推定、障害物識別の 3 ステップにより構成される。本研究では、GPU 実装の際に並列可能な処理にするため、路面平面推定のアルゴリズムの変更を行った。

3 3 次元計測

3 次元計測では、飯田らの提案した魚眼ステレオカメラを使用し 3 次元計測を行う擬似バイラテラルフィルタを用いた計測手法を使用する [8]。この計測手法では、左右のカメラから取得される時系列の画像から 3 種類の Structure from Motion(SfM) を行い、疎ではあるが高精度な 3 次元情報を取得する。また、領域ベースの 2 眼ステレオカメラにより得られた密な 3 次元情報を元にこれらの情報を融合することにより、密で高精度な 3 次元計測を実現している。

4 路面平面推定

4.1 前処理

前処理では魚眼ステレオカメラから得られた 3 次元情報からはずれ値を除去する。各方位角に対して図 2 に示すような仰角-視差空間においてしきい値処理を行う。視差の変化量の大きさがしきい値 $\Delta\lambda_{th}$ 以上ある場所で区切られた領域の内、仰角の大きさが ϕ_{th} 以下である領域をはずれ値として除去する。

4.2 平面推定

平面推定では、路面の傾斜が途中で変化するなどの路面が一つの平面で近似できない環境に対応する為に複数の平面フィッティ

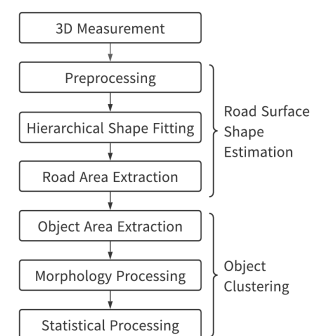


Fig.1 The flow of the proposed method

ングを行う。平面フィッティングでは、複数の平面をフィッティングするが、その際に平面推定する領域の大きさを決める必要がある。平面推定を行う領域が小さいと3次元計測のばらつきの影響を受けてしまい、平面推定を行う領域が大きいと障害物領域と路面領域両方が含まれる領域において平面フィッティングが行われる可能性が高くなる。そこで、平面推定を行う領域の大きさを変更し、複数回平面推定を行う。領域を変更する前と後に推定されたパラメータが近い場合に平面推定を行うのに適した大きさの領域であると仮定して以下に手順を示す手法を提案する。

- Step1) 視差画像を正方形領域のブロックに分割する。ただし、正方形の一辺の長さを 2^n ($n \in \mathbb{N}$) とする。
- Step2) 各ブロックに対し図3にある点が中心となるように $2^{n-1} \times 2^{n-1}$ の大きさのメディアンフィルタを9箇所適用し、代表点を抽出する。
- Step3) Step2 で得られた代表点を用いて最小2乗法により平面パラメータ (a, b, c) を求める。
- Step4) 各ブロックの長さを半分に分割し Step2~3 を行い、Step5 を行う。
- Step5) 分割の前後での平面パラメータの類似度を算出し、類似度が高い場合 Step6, 低い場合 Step7 を行う。
- Step6) 得られている平面パラメータは適切な分割において推定されたとし平面パラメータを確定する。
- Step7) Step4 を行う。試行回数が一定以上超えた場合はその場所を平面パラメータ推定不可とする（障害物と路面の境界など）。

Step3) の平面パラメータの算出には、

$$\Delta\lambda = a \cos \phi \sin \lambda + b \tan \phi \cos \lambda + c \cos^2 \lambda \quad (1)$$

を用いる。ただし、(a, b, c) は平面パラメータ、 λ は方位角、 ϕ は仰角、 $\Delta\lambda$ は視差である。Step2) において最小2乗法に用いる代表点を抽出しているが、領域の平均値を代表点にした場合、環境におけるエッジ情報がぼけてしまうため、中央値を用いて代表点を抽出している。

上記手法で3次元情報に対して複数の平面をフィッティングしたが、この複数の平面には路面以外に当てはめられたものも含まれる。そこで、路面領域にフィッティングされた平面のみ選択す

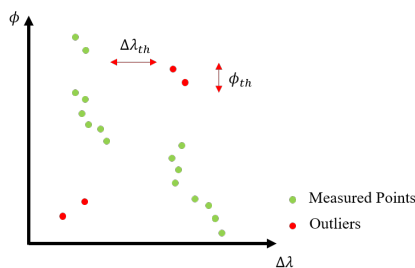


Fig.2 Disparity-elevation angle space

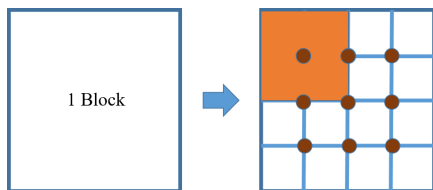


Fig.3 How to split a block in disparity image

る。図4に示すように路面が写る可能性がある領域に対してシードを設定し、近傍の平面パラメータの類似度が高ければ膨張を行う制限付き膨張を行う。複数のシードから膨張された領域の内、面積が最大の領域を路面領域とする。

5 障害物の抽出

5.1 モルフォロジー処理

4章にて得られた路面に当てはまっている平面群を用いて障害物の領域を抽出する。障害物領域に対応する箇所など路面平面が定まっていない箇所は近傍の路面平面パラメータの平均をその箇所における路面平面としてパラメータを補完する。路面平面と視差の差にしきい値を設け、しきい値より大きい箇所を1、それ以外を0とする2値画像を生成する。この画像に対して、ノイズと穴を除去するため、オープニング・クロージング処理を行う。その後、連結した領域を1つのクラスとするクラスターリングを行う。

5.2 距離の度数に着目したクラスターリング

5.1節において2値画像上で障害物のクラスターリングを行った。しかし、2値に圧縮した空間でクラスターリングを行っているため、オクルージョン等で異なる障害物が同一のクラスとして認識されてしまう。そこで、各クラスの度数分布を用いて離散的な距離の分布がある個所を探し、離散的な領域で区切られている領域ごとに再度クラスを振り分ける。

6 GPUでの実装

前処理では視差において周辺点との値の差が大きい領域を探す処理を並列化し、処理を行う。その後、視差の周辺点との差が大きい領域に対して仰角の幅を探し、小さかった場合に除去する処理をGPUにおいて行う。

平面推定後の路面抽出において、CPUでの実装では、図4に示すように路面領域での平面の制約を利用した制限付き膨張を行っていた。しかし、膨張は再帰的な処理であり、GPUでの実装において高速化の効果が薄い。そこで、図5に示す並列な処理が可能なアルゴリズムに変更する。はじめに、路面と障害物の境界を抽出する。各平面において隣接する平面との類似度を比較し、類似度が低い領域があれば1無ければ0となる2値画像を生成する。これは1が路面と障害物の境界となる画像となる。この区切られた領域の内、路面が存在する可能性が高い画像の下側に領域を持ち、面積が最大の領域を路面平面領域とする。

障害物クラスターリングにおいては、オープニング・クロージング処理をGPUで実装した。

7 精度評価実験

7.1 実験条件

本実験ではCPU実装とGPU実装の違いを評価した。高さ0.5mの段ボールを障害物とし、提案手法によって3次元情報から抽出された障害物の高さや障害物までの距離の精度と処理速度を評価した。撮影環境は傾斜の角度が変化しない平らな環境である。環境の模式図を図6に示す。これは撮影の様子を横から見た図であり、赤い線が障害物、緑の線が路面を表している。黒い2つのカメラは魚眼ステレオカメラを表しており、右から左へ1frame

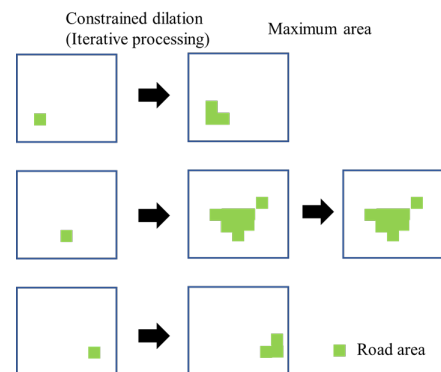


Fig.4 Previous road area extraction in disparity image

で 0.15 m 平行移動している様子を表している。カメラは 30 路面方向に傾けて、障害物までの距離を 1 m, 2 m, 3 m として各 10 回撮影を行った。前処理のしきい値は視差方向に 0.004 rad, 仰角方向に 15 pixel とした。路面平面の際に用いる類似度は正規化された平面のパラメータとし、しきい値を 0.98 とした。境界抽出では、正規化された平面パラメータの内積がしきい値 0.95 以上であり、近傍の平面群において画像上で隣接している箇所における点の距離平均が 0.35m 以下である領域を境界として抽出した。平面から 0.02 rad 以上離れている点を障害物であるとして抽出した。クラスタリングにおいては、頻度値が 40 以上の点がある領域を障害物とし、障害物が存在している領域の間隔が 0.3 以上ある場合に分離して別の障害物とした。

7.2 実験結果

実験の結果を図 7～図 10, 表 1 に示す。図 7 は正距円筒画像と視差画像である。図 8 は GPU 実装を行ったクラスタリングの結果を 3 次元点群表示したものであり、この図から障害物を正しく抽出出来ていることが分かる。また、画像下部にて障害物の誤抽出があるが、これは魚眼ステレオカメラ特有の偏差誤差が原因であり、CPU 実装時にも同様の誤差が見られた。図 9 は CPU 実装における手法の障害物の高さや障害物とカメラ間の距離の誤差を示しており、図 10 は GPU 実装における手法の障害物の高さや障害物とカメラ間の距離の誤差を示している。どちらも同等な誤差であり、CPU での実装と GPU での実装で精度の差は見られない。表 1 は処理時間を表にまとめたものである。前処理で約 5 倍、路面平面推定で約 12 倍、障害物クラスタリングで約 2 倍の速さで処理を実現しており、全体の処理時間を 2.49 s まで縮めることが出来た。

Table 1 Processing time of GPU and CPU

	GPU[s]	CPU[s]
Preprocessing	0.13±0.01	0.69±0.02
Road surface shape estimation	1.69±0.04	20±1
Object clustering	0.67±0.03	1.36±0.01
Whole process	2.49±0.09	22±1

8 結言

本研究では、魚眼ステレオカメラを使用し得られた 3 次元情報を用いた路面平面推定と障害物高さ推定手法の GPU 実装を行っ

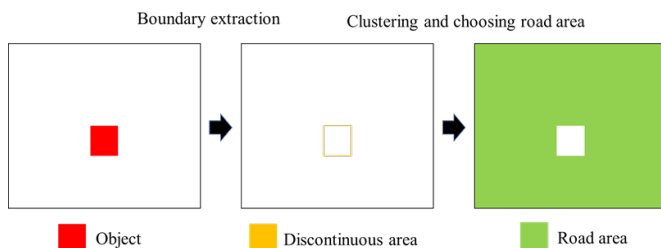


Fig.5 Parallel processed road area extraction in disparity image

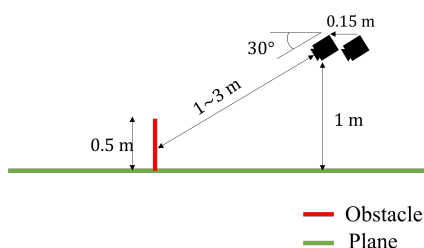
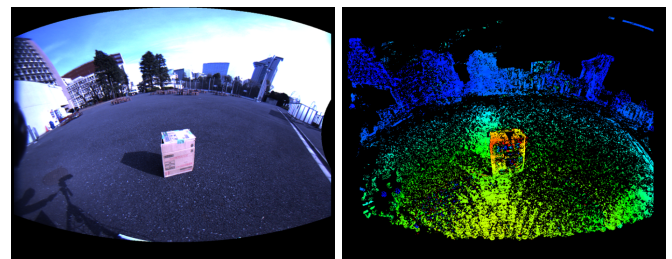


Fig.6 Experimental condition

た。実験の結果、CPU による実装と同程度の精度を保ったまま全体として約 1/8 倍の処理時間で処理を行うことが出来た。しかし、処理時間は全体として 2.49 s であり、運転支援を考えたとき高速に動く物体に対して十分に対応できない。そこで、処理時間が長い路面平面推定のアルゴリズムをさらに改良して高速化を行う必要がある。

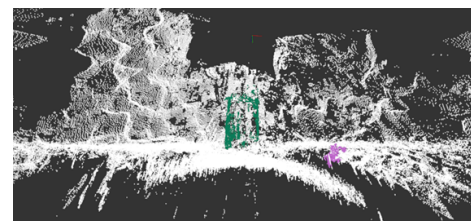
参考文献

- [1] 関 晃仁 他, “ステレオ動画像を利用した平面領域抽出による障害物検出”, 情報処理学会誌, vol.45, no.SIG 13, pp.17-24, 2004.
- [2] M. Liu et al., “Stereo Vision Based Road Free Space Detection,” 2016 9th International Symposium on Computational Intelligence and Design(ISCID), pp.272-276, 2016.
- [3] W. Song et al., “Real-Time Obstacles Detection and Status Classification for Collision Warning in a Vehicle Active Safety System,” IEEE Transactions on Intelligent Transportation Systems(ITSS), vol.19, no.3, pp.758-773, 2018.
- [4] P. Li, X. Chen and S. Shen, “Stereo R-CNN Based 3D Object Detection for Autonomous Driving”, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7636-7644, 2019.
- [5] B. Li et al., “GS3D: An Efficient 3D Object Detection Framework for Autonomous Driving”, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
- [6] X. Chen, et al., “3D object proposals using stereo imagery for accurate object class detection”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- [7] 佐久田 朝優 他, “魚眼ステレオカメラを用いた視差画像における分割スケールに着目した路面平面と障害物高さ推定”, 第 39 回日本ロボット学会学術講演会予稿集, 3K1-02, 2021.

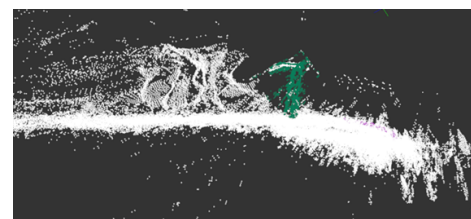


(a)Equirectangular Image (b)Disparity Image

Fig.7 Images with the target obstacle at 1 m



(a)The front view



(b)The side view

Fig.8 Point clouds with the target obstacle at 1 m

- [8] 飯田 浩貴 他, “魚眼ステレオカメラの 2 眼ステレオとモーションステレオの融合による距離画像計測”, 日本機械学会論文集, vol.85, no.875, p.19-00069, 2019.

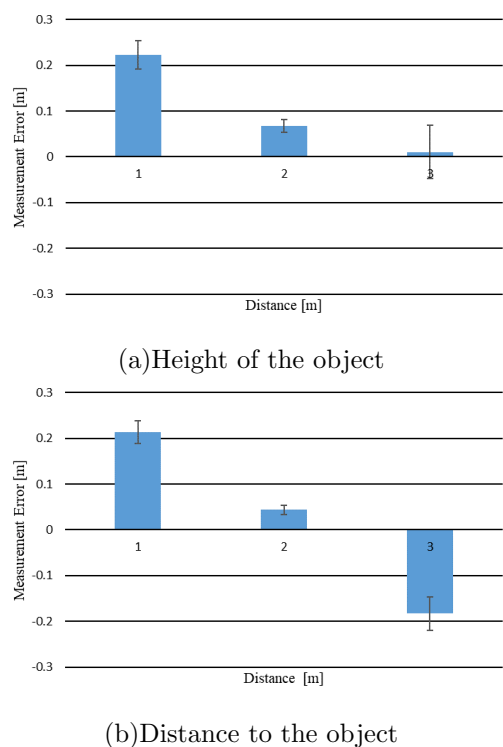


Fig.9 Measurement error of CPU implementation

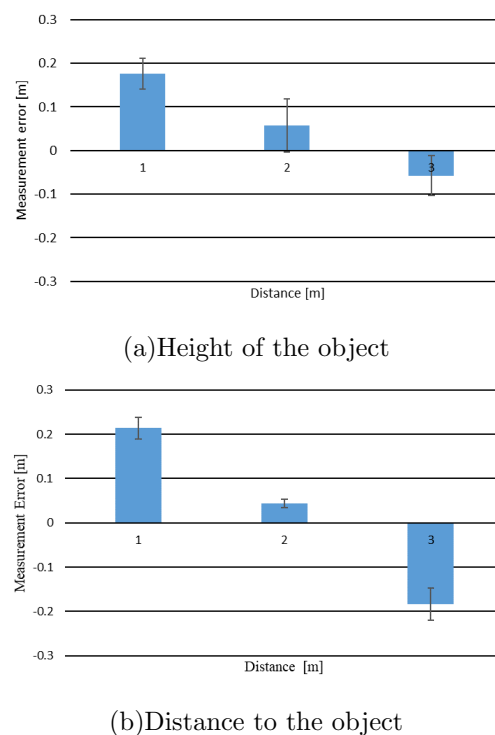


Fig.10 Measurement error of GPU implementation