Vol.2018-CVIM-212 No.6
2018/5/10

GPU を用いた Trilateral Upsampling の並列実装

門脇奨[†1]　和田俊和[†1]

概要：遠赤外線カメラや TOF カメラなどの特殊カメラは、撮像素子や周辺回路の特殊さ故に高解像度画像を得ることが難しく，高価格である．この一方で，RGB カメラは通常低価格かつ高解像度である． Trilateral Upsampling は、RGB 画像をガイドとすることで特殊カメラの画像を高解像度化する手法である。しかし、Trilateral Upsampling は、画素値に依存してフィルタが変化するため，計算時間が長いという問題がある。この問題を解決するために，本報告では 1) Trilateral Upsampling を高次元空間中での線形フィルタとして表現し，2)これを GPU を用いて並列実装することによって高速化する方法を提案する。単純な CPU 実装では約 3 時間を要した Trilateral Upsampling の計算に本手法を用いることで同じ計算が約 50ms で行えることを確認した．

キーワード：トライラテラルアップサンプリング，クロスモーダル超解像処理，GPU を用いた並列化，Bilateral Grid，温度画像

GPU Parallelization of Trilateral Upsampling

Susumu KADOWAKI[†1]　Toshikazu WADA[†1]

Abstract: Special cameras, such as far infrared cameras or time of flight cameras, are expensive while most of their spacial resolutions are low. On the other hand, standard RGB cameras are much cheaper and their spacial resolutions are high. Trilateral Upsampling can generate high resolutional special camera images by utilizing RGB images as guide information. However, the computational time is unbearably long, because the filter varies depending on the pixel values. This report presents 1) a method transforming the original upsampling to a equivalent linear filtering in an augmented space and 2) a parallel implementation of this filtering using GPU. Through the experiments, we confirmed that our accelerated method consumes 50ms for producing one high-resolutional image, while the original Trilateral Upsampling consumes almost three hours.

Keywords: Trilateral Upsampling，Super resolution using cross-modal，GPU Parallelization，Bilateral Grid，Thermal image

1. はじめに

自動運転では，道路上の歩行者などを検知するために遠赤外線画像や深度画像を用いることが有効である．このような画像が撮影可能な特殊カメラの中には，用途に合った解像度がない，ノイズが多い，そのうえ価格が高いといった欠点を持つものがある．

例えば，光の飛行時間を計測し奥行きを計算する Time of Fright（TOF）型の深度カメラは，自動運転などへの応用が期待されている．しかし，その解像度は QVGA（320 × 240）のものが標準である．この理由は，変調をかけた投影光の反射波を受光し，それと変調信号の相関計算を行う素子が特殊であり，入射光量の制限も相まって，撮像素子の高解像度化が行えないためである．また，屋外環境下では太陽光の強い照射があるため，ノイズが多く含まれる画像が得られる．また，非冷却の micro bolometer 型の遠赤外線カメラも，最高解像度は TOF カメラと同程度であり、撮像素子のばらつきや温度のドリフトによって，ノイズが発生しやすい．加えて価格は RGB カメラの数十倍程度になる．通常の RGB カメラの場合には，Full-HD（1920 × 1080）程度の解像度の高品位な画像を撮影できるものが，10,000 円程度で入手できることを考えればこれらの特殊カメラの解像度は低く，価格は高いと言える．

この問題を解決するために，低解像度の特殊カメラで撮影した対象画像を超解像処理することによって，安価に高解像度画像を作成する方法が考えられる．この方法には，以下に述べる 3 つのアプローチがある．

① 個々の対象画像に対して超解像処理を行う Image Hallcination [1]や Deep Neural Networks（DNN）を用いた手法 [2]など，大量の画像を用いた学習に基づくアプローチ．

② 同じ対象を連続的に撮影して得られた，複数の対象画像間の対応付けを行い，その結果を用いて画像の解像度を上げるアプローチ [3].

③ 対象画像とは別のガイド画像を参照し，画像の解像度を上げる Joint Bilateral Upsampling [4]などのアプローチ．

①のアプローチは，他の大量の画像を参照し，低解像度画像と高解像度画像の相関関係から高解像度画像を推定する手法であり，基本的に信頼性の高い結果は得られない．しかし，並列化を行えばリアルタイム処理の可能性はある．

②のアプローチは，画素の追跡を複数フレーム間で行い，

†1 和歌山大学 システム工学部
Wakayama University, Faculty of Systems Engineering

巨大な連立方程式を解くことで画像の超解像処理を行う方法である．このため，安定に画素の追跡が行える平面上の物体についての超解像処理は行え，結果も信頼できる．しかし，比較的近距離の凹凸のある物体については安定して適用できず，原理的にリアルタイム処理はできない．

③のアプローチの代表的な手法は，Joint Bilateral Upsampling [4]である．しかし，この手法には，図 1 に示すように，ガイド画像と対象画像のエッジが一致しない場合に，正常な超解像が行えないという問題がある．

図 1　Joint Bilateral Upsampling の失敗例：対象画像では確認できる黒とグレーの境目が超解像後はぼやける．

この問題に対して, Aleksandar ら [5]はガイド画像と対象画像の両方を用いてフィルタの重みを計算する Trilateral Upsampling を提案し，問題の緩和を試みた．この手法は，直射光などによってガイド画像のテクスチャがなくなっても通常の Bilateral Filter になるだけで，ロバスト性も兼ね備えている．しかし，この手法では，低解像度の対象画像もフィルタ計算に含めているため，解像度が十分上がらずエッジ付近にジャギーが発生しやすいという問題点がある．

Aleksandar らは，さらにこの問題を解決するために，ガイド画像と対象画像のテクスチャ間に共起性がある場合には，ガイド画像の影響を強く受け，類似していない場合には対象画像の影響を強く受ける Local Trilateral Upsampling という手法を提案している．この中で，ガイド画像と対象画像のテクスチャの共起性は局所相互情報量によって評価している．

これらの Upsampling 手法は計算量が多いため，通常の CPU を用いた計算では数分以上の時間がかかる．自動運転用のセンサーとして位置づけた場合，リアルタイム処理ができなければならないため，高速化は必須の条件である．

この問題を解決するため，Bilateral Filter の高速化手法を Local Trilateral Upsampling に適用し，GPU による並列化を行うことで，リアルタイム処理に近づけることが考えられる．しかし，Local Trilateral Upsampling のリアルタイム処理は, 局所相互情報量の計算と Trilateral Upsampling の計算の両方を高速に処理する必要があり，複雑である．そこで本研究では, Trilateral Upsampling をターゲットとして GPU を用いた並列処理によって高速化を行う．

以下では，これら高速化の対象となる Trilateral Upsampling, Local Trilateral Upsampling および，局所相互情報量などについて述べる．

1.1 Trilateral Upsampling

Aleksandar ら [5]は，Joint Bilateral Upsampling の問題点であった図 1 のような場合の超解像を可能にした新しい手法を提案している．

Trilateral Upsampling は, ガイド画像の輝度値$\boldsymbol{I}$から計算される重みG_{σ_r}と，空間的なガウシアンの重みG_{σ_s}によって表される Joint Bilateral Upsampling の重みに，対象画像$\boldsymbol{O}$から計算される重みG_{σ_o}を掛け合わせたものをフィルタの重みとして式(1)のように計算される．

$$\boldsymbol{O}_{\boldsymbol{p}}^{TF} = \frac{1}{W_{\boldsymbol{p}}^{TF}} \sum_{\boldsymbol{q}\in S} \{G_{\sigma_s}(\|\boldsymbol{p}-\boldsymbol{q}\|) G_{\sigma_r}(\|\boldsymbol{I}_{\boldsymbol{p}}-\boldsymbol{I}_{\boldsymbol{q}}\|) \times G_{\sigma_o}(\|\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{p}}-\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{q}}\|)\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{q}}\} \quad (1)$$

ただし，$\boldsymbol{T}(\boldsymbol{O})$は，対象画像をガイド画像に対応づける幾何学的変換を表しており，

$$W_{\boldsymbol{p}}^{TF} = \sum_{\boldsymbol{q}\in S} \{G_{\sigma_s}(\|\boldsymbol{p}-\boldsymbol{q}\|) G_{\sigma_r}(\|\boldsymbol{I}_{\boldsymbol{p}}-\boldsymbol{I}_{\boldsymbol{q}}\|) \times G_{\sigma_o}(\|\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{p}}-\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{q}}\|)\}$$

$$G_{\sigma_o}(x) = \exp\left(-\frac{x^2}{2\sigma_o^2}\right)$$

である．

こうして，計算された画像は，図 2 のようになり，ガイド画像にエッジが現れない場合でも，対象画像のエッジが保持されることがわかる．

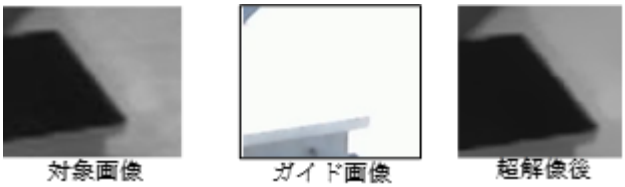

図 2　Trilateral Upsampling の出力例

この手法は，フィルタの重み$G_{\sigma_r}(\|\boldsymbol{I}_{\boldsymbol{p}}-\boldsymbol{I}_{\boldsymbol{q}}\|)G_{\sigma_o}(\|\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{p}}-\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{q}}\|)$が対象画像，ガイド画像，それぞれにおけるフィルタリング空間S内の輝度値$\boldsymbol{I}_{\boldsymbol{q}}$，$\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{q}}$に依存しているため，画素ごとにフィルタの重みを再計算する必要があり，計算が遅い．

1.2 Local Trilateral Upsampling

Trilateral Upsampling は，ガイド画像にエッジが現れない場合であっても対象画像のエッジが保持されるが，そのエッジ付近にジャギーが発生するという問題がある．Aleksandar ら [5]は，Trilateral Upsampling の輝度値に関するパラメータであるσ_r，σ_oを局所相互情報量に基づき，動的に変更することでこの問題を解決している．

Local Trilateral Upsampling は，局所相互情報量が低い場合には対象画像による輝度値の重みの影響を小さくする．逆に局所相互情報量が高いとき場合にはガイド画像による輝度値の重みの影響を小さくする. このようにすることで，エッジ付近では Joint Bilateral Upsampling のような出力画

像，それ以外の場所では Trilateral Upsampling のような出力画像となり，ジャギーが低減される．

局所相互情報量によって変更される輝度値に関するパラメータを$\sigma_{o,i}$，$\sigma_{r,i}$，点$\boldsymbol{p}$における局所相互情報量を$\mu_{YX}(\boldsymbol{p})$とすると Local Trilateral Upsampling は，式(2)で表される．

$$O_{\boldsymbol{p}}^{TF} = \frac{1}{W_{\boldsymbol{p}}^{TF}} \sum_{\boldsymbol{q}\in S} \{G_{\sigma_s}(\|\boldsymbol{p}-\boldsymbol{q}\|)\, G_{\sigma_{r,i}}(\|\boldsymbol{I}_{\boldsymbol{p}}-\boldsymbol{I}_{\boldsymbol{q}}\|) \times G_{\sigma_{o,i}}(\|\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{p}}-\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{q}}\|)\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{q}}\} \quad (2)$$

ただし，

$$W_{\boldsymbol{p}}^{TF} = \sum_{\boldsymbol{q}\in S} \{G_{\sigma_s}(\|\boldsymbol{p}-\boldsymbol{q}\|) G_{\sigma_{r,i}}(\|\boldsymbol{I}_{\boldsymbol{p}}-\boldsymbol{I}_{\boldsymbol{q}}\|) \times G_{\sigma_{o,i}}(\|\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{p}}-\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{q}}\|)\}$$

$$\sigma_{o,i} = \sigma\sqrt{1+(\mu_{YX}(\boldsymbol{p})^2)}$$

$$\sigma_{r,i} = \sigma\sqrt{\frac{1+\mu_{YX}(\boldsymbol{p})^2}{\mu_{YX}(\boldsymbol{p})}}$$

Local Trilateral Upsampling の計算結果は，図 3 に示すようになり，ジャギーが低減されている．

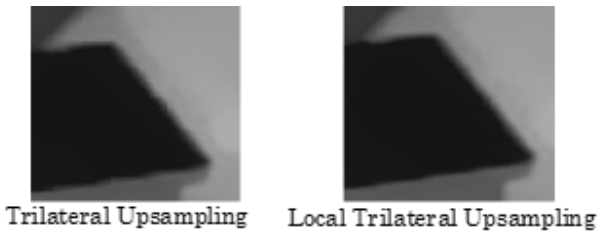

図 3　Trilateral Upsampling と Local Trilateral Upsampling の実行結果の比較（文献 [5]より抜粋）

1.2.1 局所相互情報量

局所相互情報量とは，対応する画像対の局所領域内で相互情報量を計算したものである．

局所相互情報量の計算式は式(3)のようになり，この式を用いて局所相互情報量を計算し，可視化したものが図 4 である．

$$\mu_{YX} = \sum_{x\in S}\sum_{y\in S} P_{X,Y}(x,y)\log\left(\frac{P_{X,Y}(x,y)}{P_X(x)P_Y(y)}\right) \quad (3)$$

このときのSは Bilateral Filter などの場合と異なり，X，Yから作られる 2 次元ヒストグラムの空間を意味する．また，$P_{X,Y}$は事象X，Yの同時確率を意味し，P_X，P_Yは事象X，Yの周辺確率を意味する．

図 4　局所相互情報量の一例（文献 [5]より抜粋）：白に近いほど局所相互情報量が高い

2. Bilateral Grid

Paris ら [6]は，Bilateral Filter を次元を上げた同次座標系で表現し直すことで Bilateral Filter の式が，線形になることに着目した Bilateral Grid と呼ばれる高速計算法を提案している．彼らは画像空間を拡張しx軸，y軸に加え，輝度を表すr軸を持つ 3 次元空間内の曲面として画像を表すことにより，Bilateral Filter をフィルタカーネルが場所によって変化しない線形フィルタとして表現している．また，Chen ら [7]は，Bilateral Grid を RGB 画像に適用し，5 次元空間での線形フィルタリングを GPU を用いて並列実装することにより，リアルタイム処理を実現する方法を提案している．

Bilateral Filter の結果は割り算を含む非線形の式で表される． Bilateral Grid では，この計算を分子と分母の 2 つに分けた同次座標系内のフィルタリングで表し，それぞれのフィルタリング後に，分子を分母で割る計算を行う．

このフィルタリング計算では，画像を$x-y-r$の 3 次元空間内の曲面として表現しているため，輝度値に関するフィルタの重みが輝度値によらず一定になる．これが， 線形フィルタリングになる理由である．

1 次元データに対する Bilateral Grid を図示したものが図 5 である．この図では空間方向をx，輝度値方向をζで表している．まず，図 5 の一番上に示されている 1 次元データが入力される輝度値であり，空間x，輝度ζの直積空間において表現すると 2 段目の結果が得られる．このときの$x-\zeta$直積空間で表したデータを Bilateral Grid と呼ぶ．図は，左側が分子，右側が分母を表している．次に，Bilateral Grid をガウス関数で畳み込むと 3 段目の結果が得られる．その後，分子を分母で割ると 4 段目の結果が得られる．最後に，4 段目にオレンジ色で示されている(2 段目の分子と同じ)場所からデータを取り出し，色の濃淡を輝度として元の次元に復元(スライシング)することで 5 段目に示す Bilateral Filter の計算結果を得ることができる．

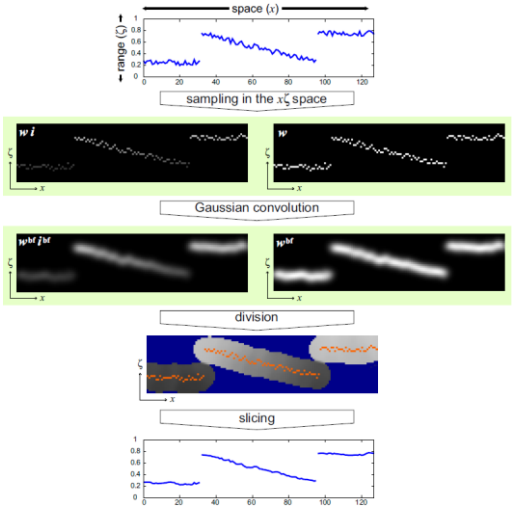

図 5　1 次元データに Bilateral Grid を適用した場合の例（文献 [6]より抜粋）

3. Trilateral Grid

Trilateral Upsampling も，Bilateral Filter と同じく，重みの計算が周辺の輝度値に依存する非線形フィルタであるため計算速度が遅い．したがって，Bilateral Grid と同様の方法で，高速化が可能であると考えられる．

本章では，Chen らの Bilateral Grid の考え方をもとにした，Trilateral Upsampling の GPU を用いた高速化手法である Trilateral Grid を提案する．

Trilateral Grid は，以下の 3 つの手順に分けることができる．

① 対象画像およびガイド画像を用いて，専用のデータ構造を作成するグリッド作成
② 作成したデータ構造とフィルタを畳み込む平滑化
③ 平滑化後のデータ構造から計算結果の画像を作成するスライシング

Trilateral Grid は，この手順を上から順番に実行することで実現される．

以降では，これらの手順を詳細に説明する．

3.1 グリッド作成

グリッド作成は，図 6 に示すように対象画像とガイド画像をもとに図 7 に示すような S（空間）× R（ガイド画像の輝度値）× E（対象画像の輝度値）の直積空間で表現される対象画像のデータ構造を作成する処理である．このとき作成されるデータ構造を Trilateral Grid と呼ぶ．

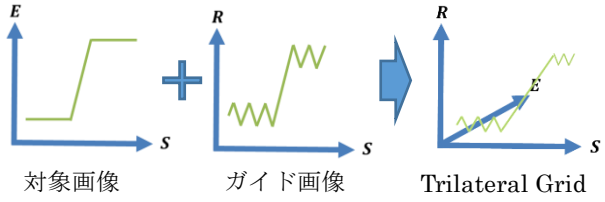

図 6 グリッド作成のイメージ

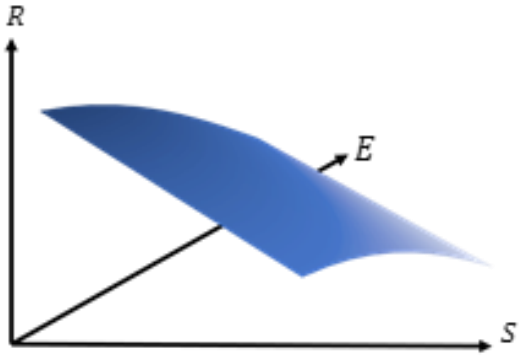

図 7 Trilateral Grid の概念図：青い曲面は Trilateral Grid 上で表現された対象画像である．

まず，通常の空間$\boldsymbol{S}$と対象画像の輝度値$\boldsymbol{R}$，ガイド画像の輝度値$\boldsymbol{E}$を軸に持ち，要素を 2 つ持つベクトル配列を作成し，式(4)に示すように分母分子に対応する要素を0で初期化する．

$$\boldsymbol{\Gamma}(\boldsymbol{S},\boldsymbol{R},\boldsymbol{E}) = (0,0) \tag{4}$$

次に，空間$\boldsymbol{S}$が同じである点の輝度値を対象画像とガイド画像からそれぞれ取り出す．

最後に，取り出した 2 つの輝度値と空間$\boldsymbol{S}$の値を座標とする点に対象画像から取り出した輝度値と 1 の 2 つの値を与える(式(5))．これを各画素について行う．

$$\boldsymbol{\Gamma}(\boldsymbol{S},\boldsymbol{R},\boldsymbol{E}) = \begin{cases} (\boldsymbol{E},1), if\ \boldsymbol{I_S} = \boldsymbol{R}\ and\ \boldsymbol{T}(\boldsymbol{O})_S = \boldsymbol{E} \\ (0,0), \qquad otherwise \end{cases} \tag{5}$$

3.2 平滑化

平滑化は作成した Trilateral Grid に$S \times R \times E$の直積空間で表現したフィルタの重みを畳み込む処理である(図 8).

数式で表すと式(6)になる．

$$\hat{\boldsymbol{\Gamma}} = g_{\sigma_s,\sigma_r,\sigma_e} \otimes \boldsymbol{\Gamma} \tag{6}$$

この計算は，Trilateral Grid 全体に行う必要はなく，第 2 要素が 0 でない点のみに行えばよい．なぜなら，次に説明するスライシングで利用される点は第 2 要素が 0 でないからである．

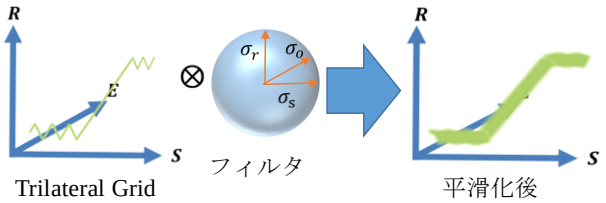

図 8 平滑化のイメージ

3.3 スライシング

スライシングは図 9 に示すように，平滑化を行った Trilateral Grid から 2 次元の画像データを作成する処理である．図 9 では，ぼかした Trilateral Grid から赤で示している元データに対応する点から数値を取り出すことで結果を計算している．

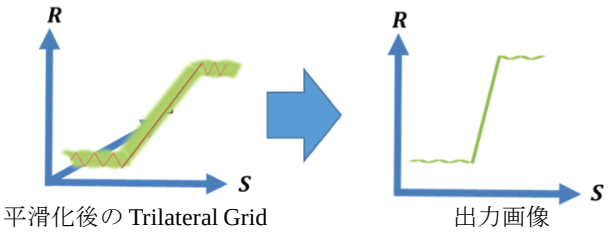

図 9 スライシングの概念図

この処理では，Trilateral Grid から式(7)に示すようにグリッド作成時にデータを代入した点からデータを取り出し，第 1 要素を第 2 要素で割ることで対応する輝度値を計算する(式(8)).

このとき，データを取り出す点がグリッド作成の時に参照した点とまったく同じ点であるため，W^{TG}は必ず 0 よりも大きい値をとる．

$$(W^{TG}\boldsymbol{O}^{TG}, W^{TG}) = \hat{\boldsymbol{\Gamma}}(\boldsymbol{S}, \boldsymbol{I}(\boldsymbol{S}), \boldsymbol{T}(\boldsymbol{O}(\boldsymbol{S}))) \quad (7)$$

$$\boldsymbol{O}^{TG}(\boldsymbol{S}) = \frac{W^{TG}\boldsymbol{O}^{TG}}{W^{TG}} \quad (8)$$

3.4 ダウンサンプリングによる高速化

実際に 3.1 から 3.3 までに説明した方法で実装すると，データ量が対象画像とガイド画像の輝度値の分だけ増加するため，計算が遅くなる．この問題を回避するために，Chen らの手法の場合と同様にデータを空間方向と対象画像の輝度値方向，ガイド画像の輝度値方向にダウンサンプリングする．この場合，サンプリングレートをそれぞれS_s，S_r，S_oとすると，グリッド作成は式(9)で表される．

$$\boldsymbol{\Gamma}\left(\frac{\boldsymbol{S}}{S_s}, \frac{\boldsymbol{R}}{S_r}, \frac{\boldsymbol{E}}{S_o}\right) += \begin{cases} (\boldsymbol{E}, 1), & if\ \boldsymbol{I}(\boldsymbol{S}) = \boldsymbol{R}\ and\ \boldsymbol{T}(\boldsymbol{O}(\boldsymbol{S})) = \boldsymbol{E} \\ (0,0), & otherwise \end{cases} \quad (9)$$

また，スライシング前のアップサンプリングは，Bilateral Grid と同じように空間$\boldsymbol{S}$，対象画像の輝度値$\boldsymbol{T}(\boldsymbol{O})_{\boldsymbol{p}}$，ガイド画像の輝度値$\boldsymbol{I}_{\boldsymbol{p}}$から求められる点の近傍点を用いた線形補間で行う．

3.5 平滑化の高速化

文献 [6]では，Bilateral Grid は，スライシングの処理がボトルネックになるとされている．しかし，Trilateral Grid の場合，平滑化の処理がボトルネックになる．これは，Bilateral Grid の軸にさらに対象画像の輝度値による軸が追加されているためである．

そこで，本手法では，平滑化を$w \geq 1$の点でのみ行う．このようにすることで，スライシングに関係しない点を中心としたフィルタの計算が行われなくなり，より高速になる．しかし，ダウンサンプリングによる高速化と併用する場合，線形補間に用いられる点の一部が計算されなくなるため，結果に若干の誤差が発生する．

4. GPU による実装

本研究では，Trilateral Grid の並列処理の方針として，データを分割し 1 つの命令を複数の分割されたデータに対して並列適用するデータレベル並列処理を用いる．ただし，Trilateral Grid の計算は手順が図 5 に示したように 3 段階に分かれているため，並列実装された各段階の処理を順番に実行することが必要になる．

以下，各段階の並列処理について詳細に述べる．

4.1 グリッド作成

グリッド作成の並列処理は，対象画像，ガイド画像，Trilateral Grid の 3 つを各スレッドに分割することで，データレベル並列処理をするが，その実装方法としては次の 2 つが考えられる．（図の参照順序を揃える）

① Trilateral Grid を図 10 に示すように矩形領域に分割したものを各スレッドに与え，各スレッドから，対象画像およびガイド画像を参照する方法．

② 対象画像およびガイド画像を図 11 と同じように格子状に分割したものを各スレッドに与え，対象画像およびガイド画像から，各スレッドに対応する Trilateral Grid 上の点を計算する方法．

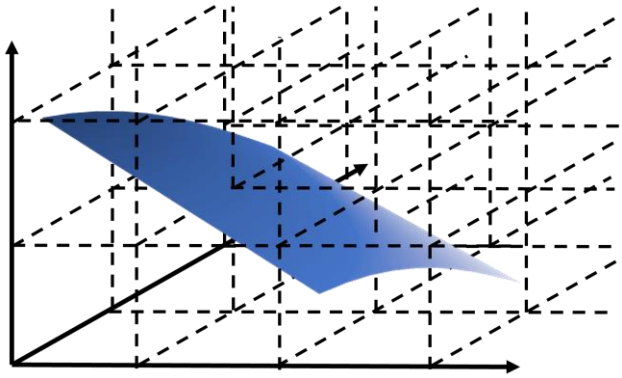

図 10　Trilateral Grid の分割方法の概念図

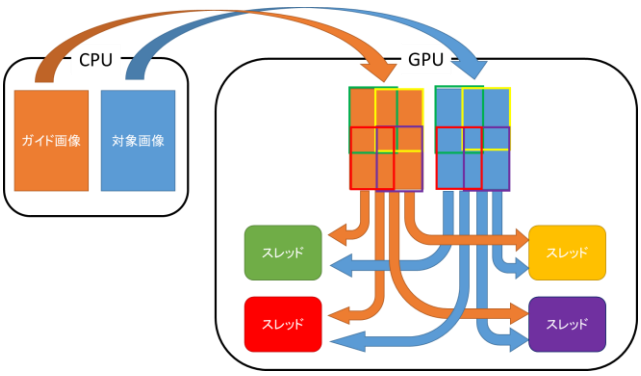

図 11　Trilateral Upsampling の GPU による並列化の概念図

①の方法を用いれば，容易に並列処理が実装できる．しかし， Trilateral Grid は，対象画像を通常の空間，対象画像の輝度値，ガイド画像の輝度値を軸に持つ空間で表現したものであり，対象画像はその空間内で曲面として表される．そのため，本来何も参照する必要のない曲面に含まれない Trilateral Grid 上の点からも対象画像とガイド画像を参照する．それに対して，②の方法は，対象画像とガイド画像から Trilateral Grid 上の点を計算し，その点のみにアクセスするため，曲面に含まれない点へのアクセスが発生しない．したがって並列化の効果が高くなるというメリットがあるため，本研究では②の方法を用いる．ただし，この方法を用いる場合，ダウンサンプリングの並列実行時に発生する

同一メモリへのアクセス競合を回避することが必要になる．

②の方法で Trilateral Grid を実装する場合，対象画像とガイド画像を空間に基づいて分割し各スレッドに転送する．そして各スレッドで Trilateral Grid 内の点を計算し，その点に対象画像の輝度値を保存する．ダウンサンプリングを用いる場合は，サンプリングレートに応じて Trilateral Grid 内のデータを加算して保存し、第 2 要素には加算したデータの個数を保存する．この加算の際のアクセス競合を回避するために CUDA の atomic 関数と呼ばれる関数群を使用する．

4.2 平滑化

Trilateral Grid は$S \times R \times E$の直積空間として表現できる．この空間内での平滑化に用いるガウシアンカーネルは，各軸上のガウシアンの積で表現できる．このため，各軸上でのガウシアンフィルタを並列実装し，その結果を掛け合わせることで平滑化結果を得る．この際に，フィルタカーネルを GPU 上のコンスタントメモリに保存しておくことで計算の高速化を図っている．

CUDA では，GPU 上のコンスタントメモリは 64kBしか利用できないため，単純にフィルタの重みを保存するのでは粗いサンプリングになってしまう．この問題は，ガウシアンフィルタの対称性を利用し，フィルタの重みを半分にすることで解決できる．

4.3 スライシング

スライシングを並列実装する方法は以下の 2 つが考えられるが，グリッド作成の並列処理と同様の理由から，本研究では②の方法を用いる．

① 平滑化後の Trilateral Grid を分割し各スレッドに配置して行う方法
② 出力画像，対象画像，ガイド画像を分割し各スレッドに配置して行う方法

②の方法で並列実装する場合，はじめに対象画像とガイド画像の空間および輝度からスライシングの対象となる点を求める．次に，対象となる点の周辺の点を取り出し線形補間を行い$W_p^{TF} I_p^{TF}$とW_p^{TF}の 2 つのデータを復元する．最後に割り算を行うことで出力画像を得る．

GPU で線形補間を行う場合，CUDA にはデータを 3 次元以下の実数型データ構造にタイリングすることで利用できるハードウェア線形補間機能が存在する．しかし，ハードウェア線形補間を利用するためにはデータのサイズ制限があるため，本手法では，ハードウェア線形補間は利用しない．

5. 実験

本章では，可視光画像をガイド画像として遠赤外線画像のアップサンプリングを行う際の性能比較実験とリアルタイム性の確認実験の 2 種類を行う．まず，共通の実験環境についての説明を行い，その後，各実験について述べる．

5.1 実験環境

本研究で用いる PC 環境を表 1 に示す．本研究の実験はすべてこの環境で行ったものである．

表 1 実験に用いた PC 環境

CPU	Intel core i7 6700K
GPU	NVIDIA Geforce GTX980 Ti
メモリ	8GB Dual Chanel
OS	ubuntu 16.04 LTS

次に，実験に用いる撮影機材を表 2 に示す．本研究の実験に用いる画像はすべてこの機材で撮影したものである．

表 2 撮影機材

可視光カメラ	logicool HD Pro ウェブカメラ C920
遠赤外線カメラ	FLIR社 Tau2 640 13mm

5.2 性能比較実験

図 12 に示す可視光画像と遠赤外線画像のペアを CPU で Trilateral Upsampling を純朴に実装したものと Trilateral Grid を GPU で並列実装したものにそれぞれ入力し，それぞれの出力画像を水平微分ログヒストグラムおよび，PSNR で比較する．また，このときの実行時間も記録し比較する．

それぞれの方法により生成された画像は図 13 図 14 のようになった．しかし，ダウンサンプリングなどによる近似の影響で同じパラメータで計算した画像には見た目に違いが生じている．そのため，見た目が近くなるように Trilateral Grid のパラメータを調整し，Trilateral Upsampling の出力画像の見た目に近づけた結果，図 15 のような結果を得ることができた．また，同じ画像対を Joint Bilateral Upsampling で高解像度化した場合，図 16 のような出力が得られた．

可視光画像　　遠赤外線画像

図 12 入力画像

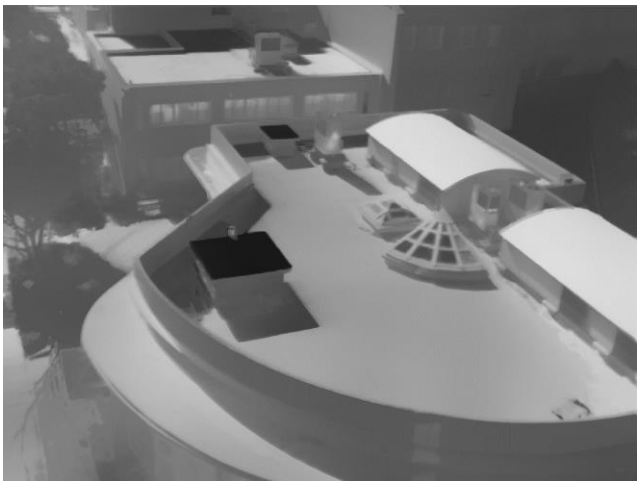

図 13　Trilateral Upsampling による生成画像（$\sigma_s = 80$, $\sigma_r = \sigma_o = 32$の場合）

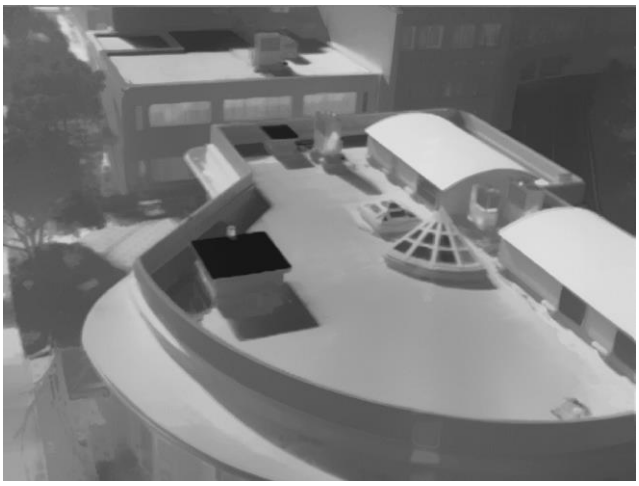

図 14　Trilateral Grid による生成画像（$\sigma_s = 80$，$\sigma_r = \sigma_o = 32$，$S_s = 40$，$S_r = S_o = 16$の場合）

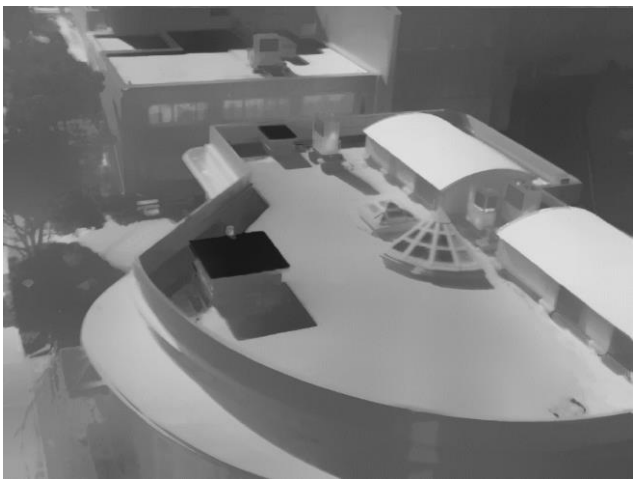

図 15　Trilateral Grid による生成画像（$\sigma_s = 80$，$\sigma_r = \sigma_o = 64$，$S_s = 40$，$S_r = S_o = 16$の場合）

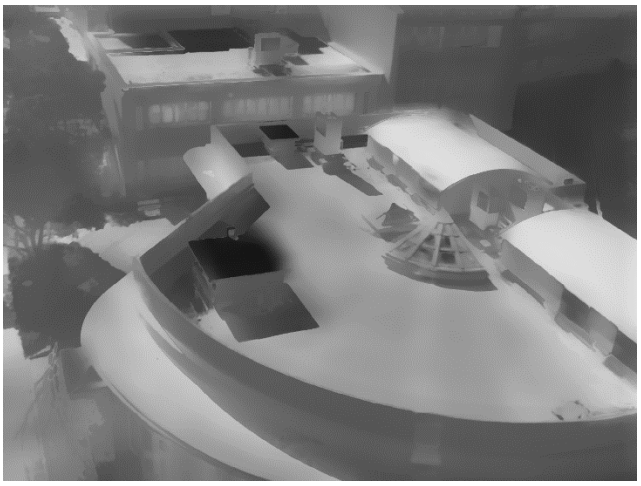

図 16　Joint Bilateral Upsampling による生成画像（$\sigma_s = 80$, $\sigma_r = 32$の場合）

図 17 は，図 13 と図 15 の水平微分ログヒストグラムのグラフである．

水平微分ログヒストグラムは，頂点の角度がノイズ除去の程度を示しており，横軸に最も近い部分の横幅がテクスチャの残り具合を示している．

図 17 を見ると頂点の角度や横軸に最も近い部分の横幅にそれほど大きな差はなくノイズ除去やテクスチャの残り具合はほぼ同じであることがわかった．

図 13 と図 15 で PSNR を計算した場合 35.25[dB]であった．

PSNR は，画像の再現性を評価する際によく用いられる手法で一般的に 30[db]以上の値をとることが良いとされている．したがって，Trilateral Grid は Trilateral Upsampling をよく再現していると言える．

実行時間の評価は同じパラメータを用いた場合で行った．

実行時間は表 3 に示すように大きな差があり，Trilateral Grid を用いて GPU による並列化を行うことで大幅な高速化が行えることがわかった．

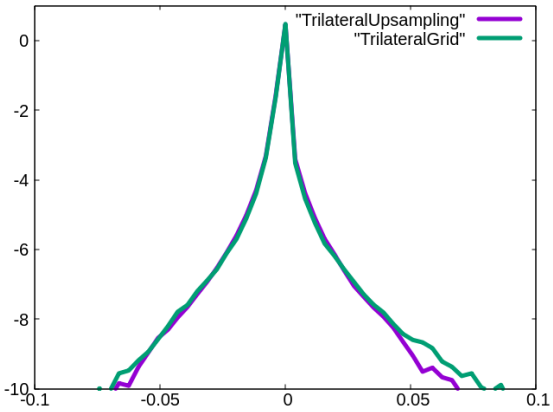

図 17　図 13 図 15 の水平微分 log ヒストグラムによる性能比較

表 3　実行時間の比較

Trilateral Upsampling	約2.7時間
Trilateral Grid	約52ミリ秒

5.3 実行時間の計測実験

遠赤外線カメラの上部に可視光カメラを取り付け，これらのカメラによって撮影される動画像をそれぞれ対象画像，ガイド画像として Trilateral Grid を実行する．このとき，可視光カメラの解像度を変更しながら複数回実験を行い実行速度の比較を行う．

可視光画像の解像度毎の実行時間は表 4 のようになった．この結果から，可視光カメラの解像度が1280×720までならリアルタイム処理が可能であるということがわかった．

表 4　可視光カメラの解像度毎の実行時間

解像度（横×縦）	総画素数[pixel]	実行時間[ms]
640×480	307200	33
800×600	480000	41
960×720	691200	53
1280×720	921600	65
1600×896	1433600	79
1920×1080	2073600	101

ガイド画像の解像度が1600×896である時の Trilateral Grid の各手順の実行時間比をグラフ化したものが図 18 である．これを見ると，処理時間の約 80%がぼかしの処理に費やされていることがわかった．

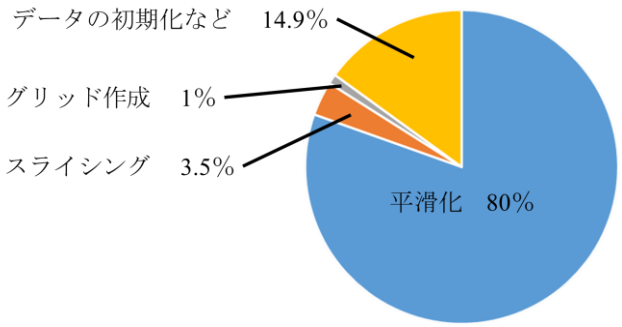

図 18　可視光カメラの解像度が1600×896の時の Trilateral Grid の実行時間

6. まとめ

本研究では，Joint Bilateral Upsampling の高速化手法である Bilateral Grid の考え方を Trilateral Upsampling に取り入れることで画像の再現性をほとんど落とさずに高速に実行できることを示した．

実行時間の計測実験の結果から，平滑化の処理を改善することで，より高速な計算が可能になると考えられる．

本研究では実装まで至らなかったが，より高品質な高解像度画像を得るためには Local Trilateral Upsampling に本手法を適用し，高速化を行う必要がある．Local Trilateral Upsampling は，フィルタの計算に加え各点での局所相互情報量の計算が必要になり，計算がより遅くなる．従って，Local Trilateral Upsampling のリアルタイム処理には ぼかしと局所相互情報量の高速計算が必要になる．

今後は，平滑化計算のさらなる高速化と，Local Trilateral Upsampling の高速化を行う．

参照文献

[1] J. Sun, N.-N. Zheng, H. Tao and H.-Y. Shum, "Image hallucination with primal sketch priors," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, no. 2, pp. II-729, 2003.

[2] C. Dong, C. Loy, K. He and X. Tang, "Learning a deep convolutional network for image super-resolution," *European Conference on Computer Vision*, pp. 184-199, 2014.

[3] S. C. Park, M. K. Park and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *IEEE signal processing magazine*, vol. 3, no. 20, pp. 21-36, 2003.

[4] J. Kopf, M. F. Cohen, D. Lischinski and M. Uyttendaele, "Joint bilateral upsampling," *ACM Transactions on Graphics (ToG)*, vol. 3, no. 26, p. 96, 2007.

[5] C. Aleksandar and T. Wada, "LOCAL TRIRATERAL UPSAMPLING FOR THERMAL IMAGE," in *ICASSP*, 2017.

[6] S. Paris and F. Durand, "A Fast Approximation of the Bilateral Filter using Signal Processing Approach," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 24-52, 2009.

[7] J. Chen, S. Paris and F. Durand, "Real-time Edge-Aware Image Processing with the Bilateral Grid," ACM SIGGRAPH conference proceedings, 2007.