Vol.2019-HPC-170 No.15
2019/7/25

ベクトルプロセッサを用いた三次元位置追跡

井手口 裕太[1,a)]　大野 善之[1,b)]　石坂 一久[1,c)]

概要：画像から物体の三次元位置を計測し，追跡することは重要である．近年では，カメラから人や車などの物体の三次元位置の計測や追跡などがハイエンドサーバー上でも行われている．このような，三次元位置追跡では，さまざまな画像処理を行う必要がある．画像処理には，メモリへのアクセスが多い処理や，メモリへのアクセスがランダムアクセスとなる処理が多く存在する．このような処理においては多くの場合，メモリネックな処理となる．したがって，メモリ帯域の高いハードウェアを用いることで，三次元位置追跡をより高速にすることが可能となる．本稿では，三次元位置追跡に必要な処理（特徴点抽出，トラッキング，色変換，積分画像生成，二次元フィルタ）をメモリ帯域が大きいベクトルエンジン（VE）を用いて実装した．その結果，関数により異なるが，GPU に対して平均で約 3.5 倍の高速化ができることが確認できた．また，三次元位置追跡を VE（2Core）を用いて実装した結果，CPU+GPU より約 1.5 倍の高速化ができることが分かった．

キーワード：ベクトルプロセッサ，SX-Aurora TSUBASA，画像処理，コンピュータビジョン

1. はじめに

画像から物体の三次元位置を計測し，追跡することは重要である．近年では，図 1（実際には実画像）に示すように，カメラから人や車などの物体の三次元位置の計測や追跡などがハイエンドサーバー上でも行われている．このような，三次元位置追跡では，物体認識，二次元トラッキング，三次元位置計測などのさまざまな処理を行う必要があり，これらの処理の多くは画像処理である．このような画像処理を行うことができるライブラリとして OpenCV [1] があり，OpenCV を用いることで，これらの画像処理を簡単に実装することができる．OpenCV を高速に実行するために，OpenCV には元の関数とは別に GPU 用の関数が用意されており，その関数を利用することでより高速に処理することも可能である．したがって，GPU を利用することで，三次元位置追跡の高速化が可能である．

しかしながら，画像処理には，メモリへのアクセスが多い処理や，メモリへのアクセスがランダムアクセスとなる処理が多く存在する．このような処理においては多くの場合，メモリネックな処理となる．したがって，メモリ帯域の大きなハードウェアを用いることで，三次元位置追跡を

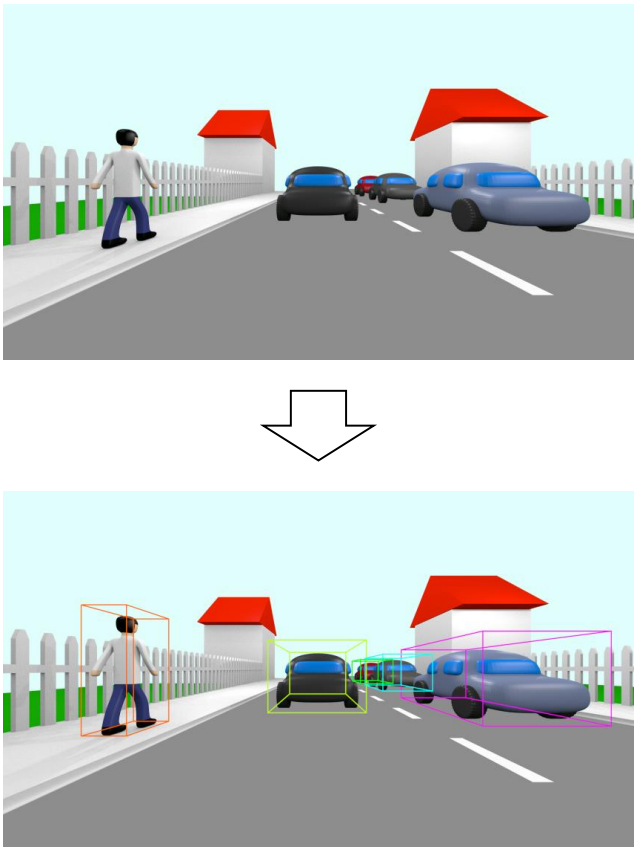


図 1　三次元位置追跡の出力結果イメージ
※実際には実画像

[1] NEC データサイエンス研究所
神奈川県川崎市中原区下沼部 1753, 211–8666
[a)] y-ideguchi@cj.jp.nec.com
[b)] y-ohno@ji.jp.nec.com
[c)] k-ishizaka@ay.jp.nec.com

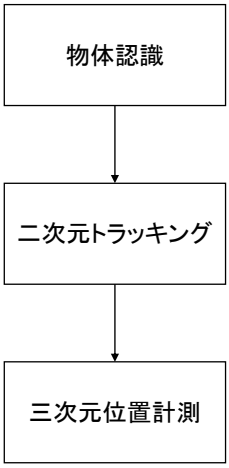


図 2 三次元位置追跡システム

より高速にすることが可能となる．

近年，ベクトルプロセッサであるベクトルエンジン（VE）が搭載されたコンピュータ（SX-Aurora TSUBASA [2]）が発売された．VE は，メモリ帯域が大きいという特性があるため，メモリネックな処理においては，GPU 以上の高速化が期待される．

本研究では，ベクトルプロセッサを用いて三次元位置追跡を実装することで，三次元位置追跡の高速化を目指す．OpenCV には，非常に多くの処理が実装されているが，本稿では，三次元位置追跡に必要な処理（特徴点抽出，トラッキング，色変換，積分画像生成，二次元フィルタ）をベクトルプロセッサを用いて実装することで，三次元位置追跡の高速化を目指す．本稿では，ベクトルプロセッサを用いて三次元位置追跡を実装した結果を示す．

2. ベクトルエンジン（VE）

ベクトルプロセッサは，複数のデータをレジスタに格納し，同時に計算を行なうことで，複数のデータに対して数値演算を高速に実行できるプロセッサである．そのため，レジスタの数が多く，これらのデータをメモリから多く読み書きするため，メモリバンド幅も大きい．ベクトルプロセッサは，一つのベクトル命令で複数のデータを一括処理できるため，大規模・複雑な計算を高速処理することができ，気象，航空宇宙，環境，流体解析，物性計算などに適している．従来，ベクトルプロセッサはスーパーコンピュータ用に開発されていた．近年，発売された SX-Aurora TSUBASA [2] では，タワー型のモデルも発売され，より使用用途の拡大が見込まれる．また，SX-Aurora TSUBASA は，高速メモリを搭載したベクトルプロセッサであるベクトルエンジン（VE）を搭載している．SX-Aurora TSUBASA に搭載されている VE（Type B）は，1.2 TB/s の大きいメモリバンド幅を持っており，GPU（NVIDIA Tesla V100: 0.9 TB/s）よりも大きい．そのため，メモリネックな処理は，VE を用いることでより高速化できると考えられる．本稿では，三次元位置追跡に使われる画像処理を VE で実装することで，三次元位置追跡の高速化を目指す．

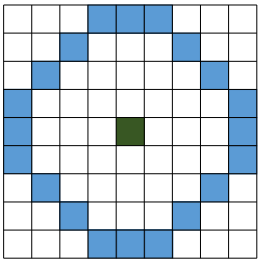


図 3 特徴点の計算: アクセスするピクセル

3. システム概要

本稿では，図 2 に示すような構成の三次元位置追跡を対象とする．出力結果の例を図 1 に示す．このシステムは，物体認識，二次元トラッキング，三次元位置計測の三つのモジュールから成り立っている．このシステムは，三つのモジュールがパイプライン的に動作するため，システムの速度は，三つのモジュールの内の最も遅いモジュールの処理時間に律速する．したがって，すべてのモジュールを均等に高速化する必要がある．二次元トラッキング，三次元位置計測では，多くの画像処理を行う必要がある．本稿では，その中でも，特徴点抽出，トラッキング，色変換，積分画像生成，二次元フィルタについて言及する．以下，これらの処理について OpenCV [1] の実装を基に説明する．

3.1 特徴点抽出

特徴点抽出の処理速度の速い関数として，FastFeatureDetect がある．FastFeatureDetect では，図 3 に示すように，対象とするピクセルに対して，周囲の 16 ピクセルにアクセスし，対象とするピクセルが特徴点として利用できるかどうかの有無を判断する．その後，特徴点として利用できるピクセルに対しては，その度合いを計算し，数値化する．これをすべてのピクセルに対して処理を行うことで画像全体の特徴点を抽出する．

この処理では，度合いを計算の時間が長いため，数値化しないピクセルに対しては，処理時間が非常に短くなる．したがって，この処理を VE で行う場合には，特徴点として利用できるかの判定までと，度合いの計算のベクトル長を変えることで無駄な処理を減らすことができる．この場合，特徴点として利用できると判断されたピクセルの ID を配列に詰めていき，その後，その配列の ID のピクセルに対してのみ度合いを計算する．このとき，実際に度合いを計算するピクセルの位置は離散的な位置になるため，後半の処理のメモリへのアクセスはランダムアクセスとなる．また，一つの特徴点の度合いを計算するために 16 ピクセルにアクセスする必要があるが，度合いを計算する際の演算数も多い．したがって，この処理は，度合いを計算する

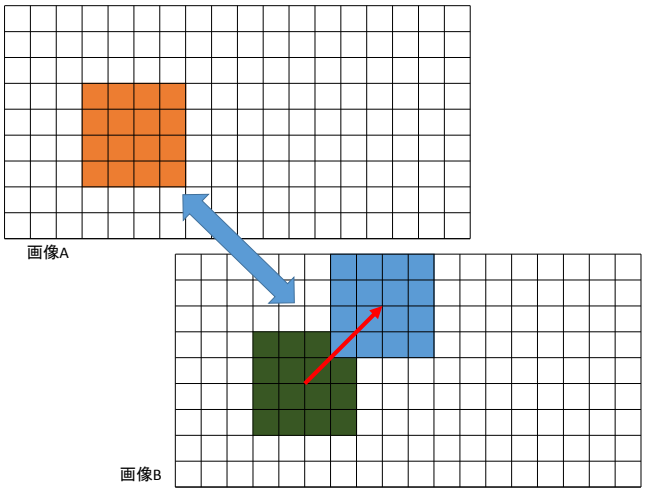


図 4　トラッキング：一点をトラッキングする際にアクセスする範囲

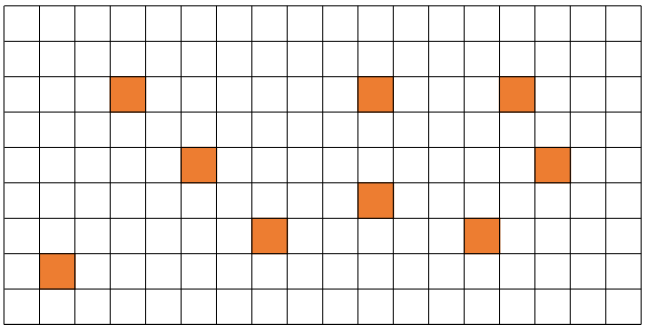


図 5　トラッキング：トラッキングする各特徴点の位置

際の演算数によって，演算ネックかメモリネックになる．

3.2　トラッキング

トラッキングの処理速度の速い関数として，OpticalFlow-PyrLK がある．OpticlFlow では，画像 A で特徴点抽出された特徴点に対して，画像 B 上のどの位置に移動したかを計算する．図 4 に示すように，まず画像 A と画像 B の特徴点の周囲のピクセルにアクセスし，比較する．その後，画像 B でアクセスする位置を変えながら，同じになる位置を探す．実際には，各画像の縮小画像や，微分画像，二階微分した画像を事前に計算し，それらを利用することで効率的に位置を変える．

この手法は，画像 B でアクセスする位置を変えながら比較を行い，誤差がしきい値以下になるまで演算を繰り返すため，特徴点によって収束までの演算回数が異なる．したがって，特徴点抽出と同様に，ベクトル長を変えることで無駄な処理を減らすことができる．また，各特徴点は図 4 に示すように，ランダムな位置に存在するため，この処理はランダムなアクセスとなり，メモリネックな処理となる．

3.3　色変換

色変換にはさまざまな関数があるが，特に演算数が多い色変換の関数として，カラー画像を Lab 色空間に変換する，cvtColorBGR2Lab がある．カラー画像を Lab 色空間に変換するには，RGB 色空間を一度 XYZ 色空間に変換した後，Lab 色空間に変換する．これをすべての画素に対して処理する．

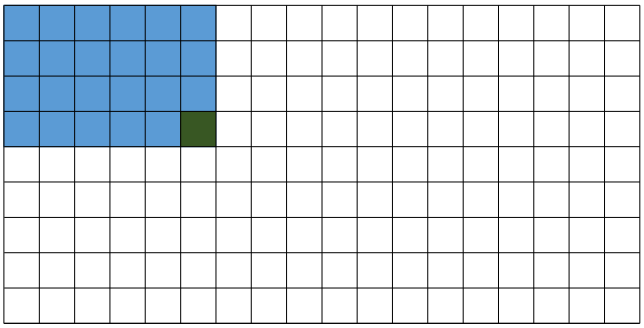
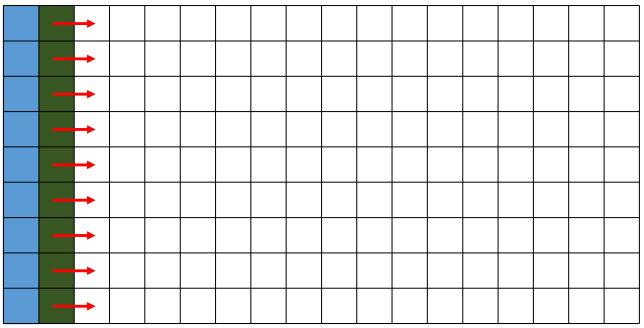
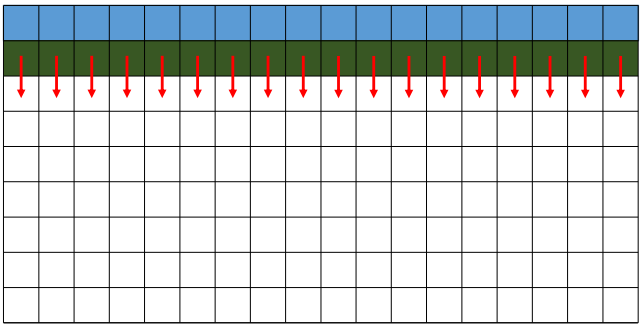


図 6　積分画像生成



(a) 右方向への加算



(b) 下方向への加算

図 7　積分画像生成手法

この処理は，XYZ 色空間から Lab 色空間への処理において演算数が非常に多い．そのため，この処理は演算ネックな処理となる．しかしながら，RGB 色空間の値域は各色 0〜255 であるため，事前に色の対応を計算し，メモリに保存しておくことで，実際の色変換をメモリアクセスのみにすることが可能である．この場合にはメモリネックな処理となる．

3.4　積分画像生成

積分画像生成の関数として，integral がある．積分画像生成では，図 6 に示すように，対象とするピクセルに，そのピクセルより上および左に位置するすべてのピクセルの総和を計算する．この処理は以下の処理によって代替する

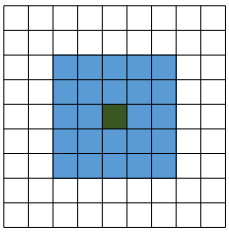


図 8　フィルタ処理：アクセスするピクセル

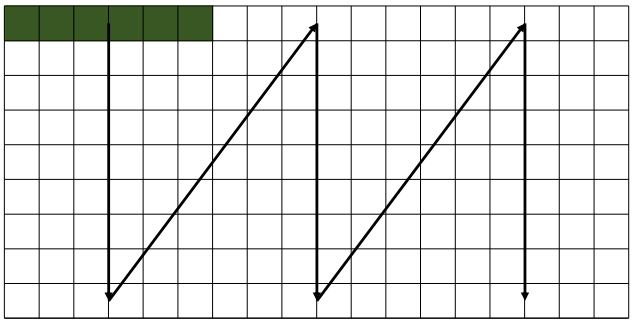


図 9　フィルタ処理の順番

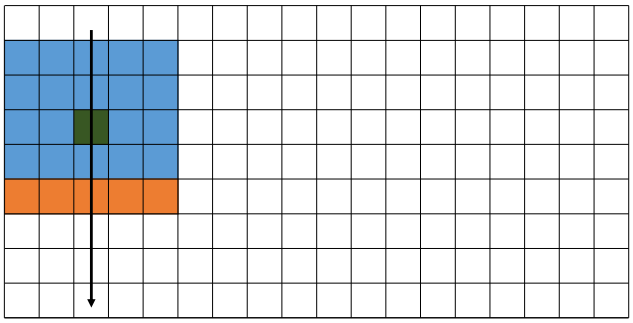


図 10　フィルタ処理のレジスタの使い回し

ことが可能である．まず，図 7 (a) に示すように，左に位置するピクセルの値を加算する処理を，左から右に行う．その後，図 7 (b) に示すように，上に位置するピクセルの値を加算する処理を，上から下に行う．これにより，積分画像を生成できる．

この処理では，図 7 (a) はストライドロード，ストライドストアとなり，図 7 (b) は連続ロード，連続ストアとなる．この処理は，演算の数が比較的少なく，メモリへのアクセス数が多いためメモリネックな処理となる．

3.5　二次元フィルタ

二次元フィルタの関数としては，filter2D がある．フィルタ処理では，図 8 に示すように，対象とするピクセル周囲のピクセルにアクセスし演算を行う．これを画像内のすべてのピクセルに対して行う．

この処理は，演算の数が比較的少なく，メモリへのアクセス数が多いためメモリネックな処理となる．また，この処理の場合，図 9 に示すような順番で処理を行うことで，図 10 に示すように，レジスタ上のデータを使い回すことができる．

表 1　画像一枚あたりの処理時間

処理	GPU [ms]	VE [ms]
Feature Detect	0.982	0.978
Optical Flow	2.520	0.962
cvtColorBGR2Lab	0.534	0.550
integral	1.357	0.303
filter2D	0.542	0.052

4.　実験

実験では，要素処理を VE で実装した際の GPU との速度性能の比較を行った．また，二次元トラッキングと三次元位置計測を VE を用いて実装した際の，各モジュールの速度性能を CPU+GPU の場合と比較した．

4.1　要素処理評価実験

要素処理評価実験では，GPU を搭載したサーバと SX-Aurora TSUBASA で処理速度を比較した．GPU は NVIDIA TITAN V(13.8 TFlops, 0.65 TB/s) を利用した．SX-Aurora TSUBASA はオンサイトモデルの VE（Type B: 8 Core, 4.3 TFlops, 1.2 TB/s) [2] を利用した．OpenCV の関数を利用し，VE では C++で実装した関数を利用した．OpenCV には，OpenCV-3.3.1 を使用した．GPU および VE で 800 枚処理した際の 1 枚あたりの平均処理時間を表 1 に記す．GPU で実行した際の処理時間を基準に，VE で実行した際の高速化の倍率を図 11 に示す．

特徴点抽出では，FastFeatureDetect 関数の GPU 版を使用した．入力画像は，1920×1080 のグレー画像とした．

トラッキングでは，OpticalFlowPyrLK 関数の GPU 版を使用した．入力画像は，1920×1080 のグレー画像とした．パラメータは，縮小画像生成の数を 9 枚，ウィンドウサイズを 4×4 とした．

色変換では，cvtColorBGR2Lab 関数の GPU 版を使用した．入力画像は，1920×1080 のカラー画像とした．

積分画像生成では，Integral 関数の GPU 版を使用した．入力画像は，1920×1080 のグレー画像とした．

二次元フィルタでは，Filter2D 関数を対象として，より高速なフィルタ関数である NPP [3] の nppFilter 関数を利用した．入力画像は，1920×1080 のグレー画像とした．パラメータは，フィルタサイズを 7×7 とした．

特徴点抽出では，図 11 を見ると，GPU と同等の処理性能であることが確認できる．Feature Detect の後半の処理はランダムアクセスとなるが，度合いを計算する際の演算数も多いため，度合いを計算する際の演算数によって，演算ネックかメモリネックになる．この結果から，今回実装した特徴点抽出の関数である Feature Detect を高速化するためには，ハードウェアの演算性能とメモリ帯域の両方が重要であることが分かる．

トラッキングでは，図 11 を見ると，GPU に対して，約

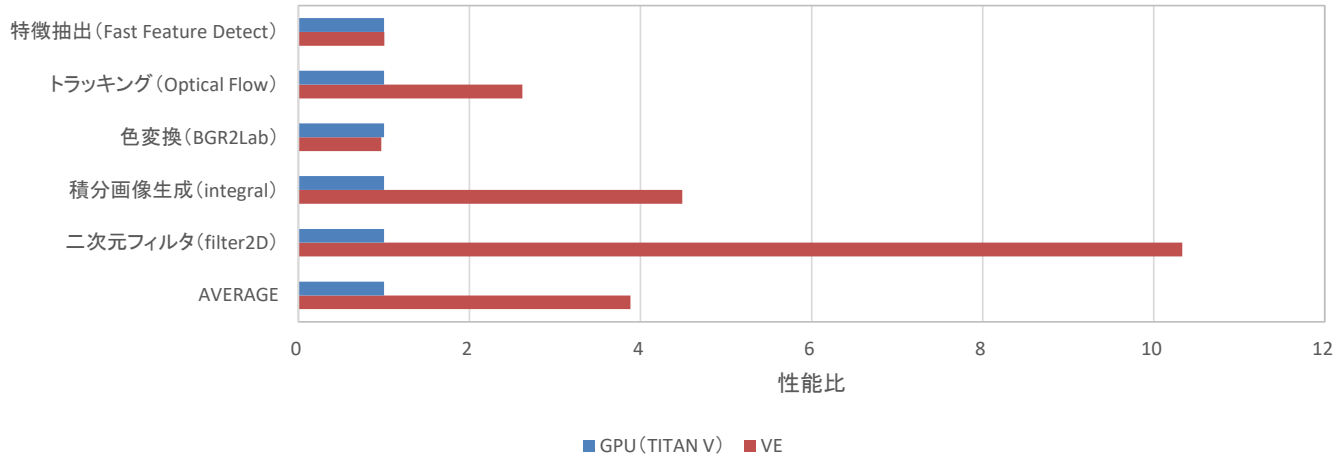


図 11 各画像処理の GPU との性能比較結果

2.5 倍の高速化ができていることが確認できる．Optical Flow は，ランダムアクセスになるため，メモリネックな処理となることが予想される．この結果からも，この処理はメモリネックな処理であることが確認できる．

色変換では，図 11 を見ると，GPU と同等の処理性能であることが確認できる．BGR から Lab への色変換は，演算数が非常に多いため，演算ネックな処理となる．しかしながら，事前に色の対応を計算することで，メモリネックな処理にできる．GPU では演算ネックな手法を，VE ではメモリネックな手法を実装することで，同等の処理性能結果となったと考えられる．

積分画像生成では，図 11 を見ると，GPU に対して，約 4.5 倍の高速化ができていることが確認できる．この処理は，演算の数が比較的少なく，メモリアクセスが多いためメモリネックな処理となることが予想される．この結果からも，この処理はメモリネックな処理であることが確認できる．

二次元フィルタでは，図 11 を見ると，GPU に対して，約 10 倍の高速化ができていることが確認できる．この処理は，演算の数が比較的少なく，メモリへのアクセス数が多いためメモリネックな処理となることが予想される．この結果からも，この処理はメモリネックな処理であることが確認できる．

これらの結果より，今回実装した関数において GPU に対して平均で約 3.5 倍程度の高速化ができることが確認できた．また，三次元位置追跡で使われている画像処理において，メモリネックな処理が多いことも確認できた．また三次元位置追跡で使われている画像処理を VE で実装することでより高速化することができると考えられる．

4.2 モジュール評価

保証対象外の構成ではあるが，今回はテストケースとして，図 12 に示すように，2 Socket Xeon サーバに GPU と VE を導入することで，CPUx2 (Xeon Gold 6126: 1.0 TFlops, 0.13 TB/s)，GPU（Quadro P2000: 3.0 TFlops, 0.14 TB/s)，VE（Type C: 8 Core, 4.3 TFlops, 0.75 TB/s）の構成のサーバを作成し，このサーバで速度評価を行った．評価実験では，VE は各モジュールで 1Core のみを利用した．

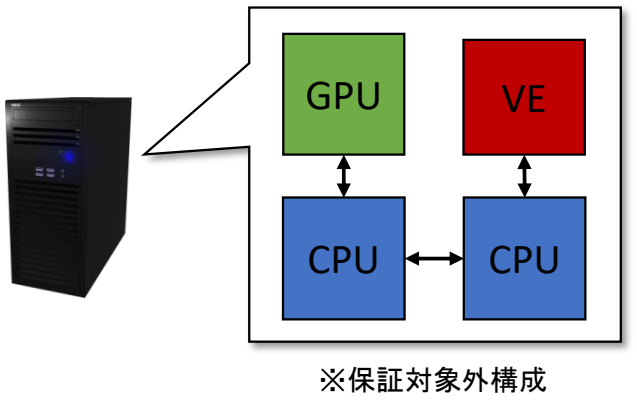


図 12 2 Socket Xeon サーバに GPU（Quadro P2000）と VE（Type C）を導入した構成

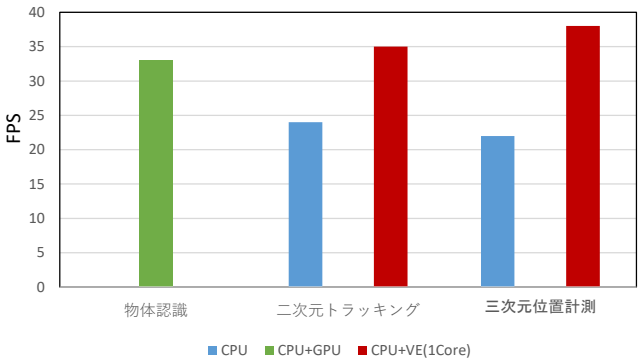


図 13 三次元位置追跡の速度向上

本実験では，物体認識として YOLOv2 [4] を用いて GPU 上で動作させ，二次元トラッキングおよび三次元位置計測を CPU および VE で実装し，速度を評価した．VE の実装では，モジュール内の画像処理に必要なデータをまとめ

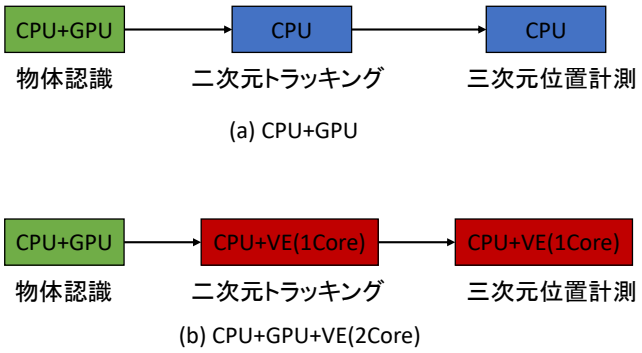


図 **14** 全体動作

て VE に転送し，画像処理の結果を CPU に転送する構成とした．評価結果を図 13 に示す．この結果から，二次元トラッキングでは CPU 比約 1.5 倍，三次元位置計測では CPU 比約 1.7 倍の高速化ができていることが確認できる．今回は各モジュールで VE の 1Core を用いたが，複数コアを用いることでより高速化が可能である．また，今回の実験では，画像処理部分のみを VE で処理するため，CPU と VE 間で通信が発生するが，今後，すべての処理を VE で処理することで，通信部分がなくなるため，より高速化することができると考えられる．

図 14 に示すように，CPU+GPU を利用する場合の (a) と，CPU+GPU+VE を利用する場合の (b) でそれぞれ速度を評価する場合，それぞれのモジュールの最も遅いモジュールに速度が律速される．したがって，今回の結果より，(a)(b) それぞれの場合で三次元位置追跡を実装した場合における性能は，(a) は 22fps，(b) は 33fps となることが予想される．したがって，三次元位置追跡を VE（2Core）を用いることで，CPU+GPU に比べて約 1.5 倍の高速化できることが分かる．

5. おわりに

本研究では，ベクトルプロセッサを用いて三次元位置追跡を実装することで，三次元位置追跡の高速化を目指す．本稿では，三次元位置追跡に必要な処理（特徴点抽出，トラッキング，色変換，積分画像生成，二次元フィルタ）をベクトルプロセッサを用いて実装した．その結果，関数により異なるが，GPU に対して平均で約 3.5 倍の高速化ができることが確認できた．この結果から，画像処理が VE に適していることが分かった．さらに，三次元位置追跡を VE（2Core）を用いて実装することで，CPU+GPU より約 1.5 倍の高速化ができることが分かった．今後は，今回実装していない処理においても VE で実装することで，更なる高速化を目指す．

参考文献

[1] OpenCV. https://opencv.org/.
[2] NEC Corporation. SX-Aurora TSUBASA. http://jpn.nec.com/hpc/sxauroratsubasa/.
[3] NVIDIA. NVIDIA Performance Primitives. https://developer.nvidia.com/npp.
[4] J. Redmon and A. Farhadi. YOLOv2. https://pjreddie.com/darknet/yolov2/.