H-015

複数の組み込みプロセッサと GPU を併用した俯瞰画像合成システムの開発

Development of Bird's-Eye View Synthesis System using Multiple Embedded Processors and GPU

田所　勇生[1)]　近藤　真史[2)]　瀬島　吉裕[1)]
Yusei　TADOKORO　Masafumi　KONDO　Yoshihiro　SEJIMA
佐藤　洋一郎[1)]　河本　崇幸[3)]　石原　洋之[3)]
Yoichiro SATO　Takayuki KOMOTO　Hiroyuki ISHIHARA

1　まえがき

近年，監視カメラが急速に普及しているが，スーパーマーケット等の商業施設では陳列棚などにより死角を排除できず，日本の万引き被害額は 4500 億円以上 (世界 2 位) に達している．これに対する有効な解決策の一つとして，広視野角・高解像度なカメラの導入が挙げられるが，これに代表される魚眼カメラはレンズが特殊であるため依然として高価であり，設置コストの観点から導入可能な台数に限界がある [1]．この問題に対して筆者らは，天井に格子状に配置した複数のネットワークカメラを用いて，任意の視点から死角のない動画像を合成する，いわゆる自由視点型の俯瞰合成システムの開発を進めている．その第一段階として，複数のカメラ付き組み込みプロセッサ (Raspberry Pi3) と画像処理ライブラリ OpenCV のパノラマ合成関数 (Sticher) とを併用した俯瞰画像合成システムの試作を行った [2]．しかしながら，この種の画像合成関数は演算負荷が高く，合成枚数の増加に伴ってフレームレートが著しく劣化することとなる．そこで本研究では，GPU を用いて画像合成関数の高速化を図るとともに，合成処理の一部をクライアント (組み込みプロセッサ) に譲渡することによるサーバへの負荷分散手法について検討を行う．

2　俯瞰画像合成システム

提案する俯瞰画像合成システムは，監視フロアの撮像を担うネットワークカメラをクライアント，俯瞰画像の合成を担う汎用 PC をサーバとするクライアントサーバモデルとして実現し，その具体的な構成を図 1 に示す．以下，クライアントによる撮像からサーバでの俯瞰画像の合成までの動作手順を示す．なお，“cv::”を付した関数は OpenCV の実装関数・クラスである．

手順 1．動画取り込み：cv::VideoCapture クラスを用いてカメラ (Pi Camera V2) を操作し，動画のフレーム画像 (以下，単にフレームという) を取得する．

手順 2．JPEG エンコード：ビットマップ (BMP) 画像であるフレームは，そのままではデータ量が大きいため cv::imencode 関数で JPEG に圧縮する．

手順 3．パケット送信：圧縮されたフレームをパケット単位に分割して送信する．なお，本システムでは，俯瞰画像の合成に係るフレームレートの向上，すなわちクライアント-サーバ間の通信速度を優先し，その通信プロトコルには通信速度に優れる UDP を採用する．

手順 4．無線通信：無線によりクライアント-ルータ間の通信を行う．Pi3 は無線通信機能 (IEEE 802.11 b/g/n) を

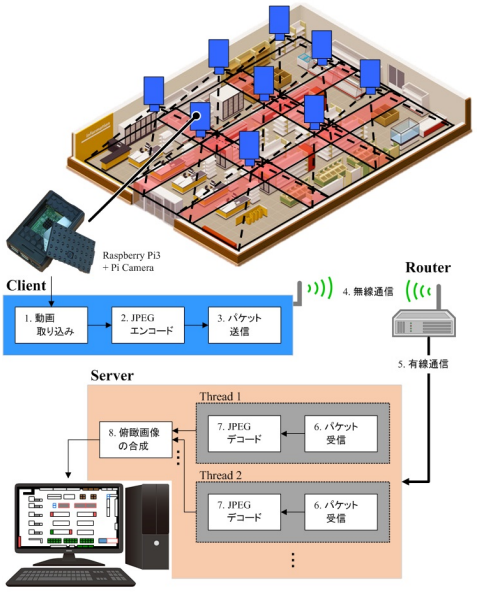

図 1　俯瞰画像合成システムの構成

内蔵しているが，一般的な動画像の送受信においても十分な通信帯域を備えている IEEE 802.11 n (最大伝送速度 150Mbps) を使用する．

手順 5．有線通信：複数のクライアントによる帯域負荷を軽減するため，ルータ-サーバ間については LAN ケーブル (最大伝送速度 1Gbps) による有線通信を行う．

手順 6．パケット受信：受信したパケットを結合して圧縮されたフレームの画像データを取得する．

手順 7．JPEG デコード：cv::imdecode 関数で手順 6 の画像データをデコードしてフレームを復元する．

手順 8．俯瞰画像の合成：各クライアントから得られたフレームを俯瞰画像に合成し，動画像としてモニタに出力・表示する．

なおここでは，各クライアントから送信される画像を随時合成する方針を採り，これを実現するため，クライアントと一対一で交わされる手順 6 および手順 7 を単位としてサブスレッドを生成して非同期通信を実施する．

3　俯瞰画像合成の高速化

OpenCV では，複数の画像を合成してパノラマ画像を生成する cv::Stitcher クラスが用意されており，これを利用することで手順 8 に係る処理を実現できる．Stitcher クラスでは，cv::estimateTransform 関数でカメラ位置を推定し，cv::ComposePanorama 関数 (以下，CP 関数) で画像合成を行う．ここで本研究では，各カメラの位置を固定して撮像を行うため，最初に estimateTransform 関数でカメラ位置を推定すれば，後はそれを基に CP 関数を順次

1）岡山県立大学, Okayama Prefectural Univ.
2）川崎医療福祉大学, Kawasaki Univ. of Medical Welfare
3）(株) システムズナカシマ, Systems Nakashima Co.,Ltd.

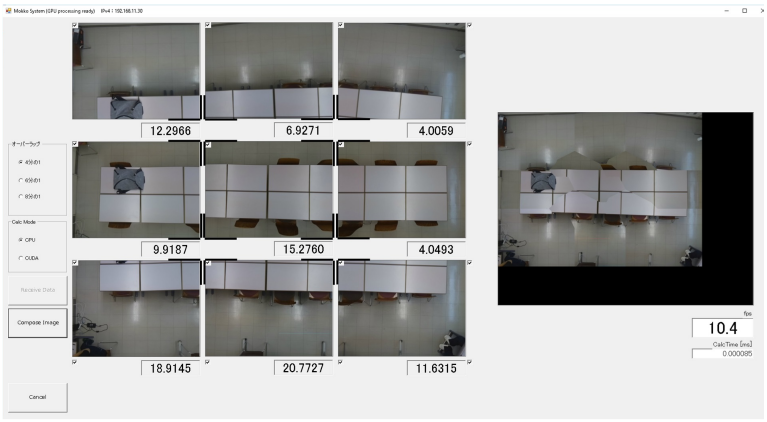

図 2　俯瞰画像合成システム (川崎医療福祉大学 8015 演習室)

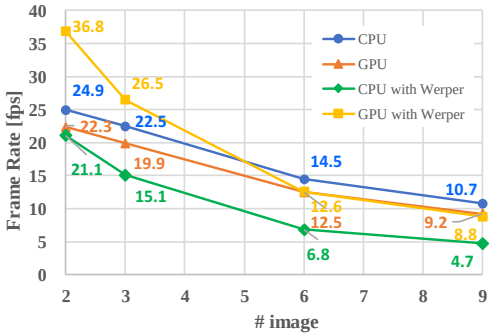

図 3　合成に係るフレームレート

実行すればよい．したがって，画像合成の速度は CP 関数のそれに依存することとなるが，OpenCV の関数群の中でもこれに係る演算コストは高く，一般的な CPU 処理では十分なフレームレートを得ることが困難である．

CP 関数の実体は以下の 3 つの関数であり，これらを順に実行することで画像合成を実現している．

Werper：カメラ位置に応じて幾何学変換 (変形) を施す．

Compensator：隣接する画像間の輝度変化を緩和するため露出補正を施す．

Blender：上記関数で得られた画像を合成する．

ここで文献 [3] では，CP 関数の OpenCV ソース (.cpp) から汎用的な四則演算部を抽出し，それを GPU による並列処理として CUDA ソース (.cu) に書き下すことで，CP 関数を完全に GPU で処理する手法を提案している．しかしこの実装では，球体形の投影面を前提としているため，本研究のように全てのカメラが直下を撮像する俯瞰画像の合成には対応できない．そこで本研究では，カメラ位置を表す平行移動行列 t を新たに定義し，これを適宜 cv::warp に与えることで投影面を平面形に拡張した画像合成を実現する (紙面の都合上，コード詳細は省略)．

4　幾何学変換の分散処理

本研究では，クライアントに組み込みプロセッサの使用を前提としていることから，サーバと協調してその処理の一部をクライアント側で実行することができる．例えば，演算負荷の高い Werper 関数をクライアント側で高速に処理すれば [4]，システム全体の負荷分散効果を期待できる．ここで，サーバ側で処理することになる Compensator 関数と Blender 関数は，隣接する画像間の重複領域を表すマスク画像を引数とする必要があるが，これはカメラ位置情報を基に Werper 関数から共通的かつ連続的に生成されるものであるため，この形態では使用できない．そこで本研究では，図 1 よりカメラ位置は規則的かつ静的に決まるものと仮定し，平面上のカメラ位置からマスク画像とその座標位置リストを新たに作成し，それを引数として合成を行う方針を採る．特にこの形態によればカメラ位置を任意に定められるため，本研究が目的とする自由視点型の画像合成にも応用できる．

5　実装と評価

システムの開発にあたっては豊富な GUI を備えた Microsoft .Net Framework(以下，単に.Net という) による開発を前提とし，CUDA，OpenCV，および.Net の各開発環境を横断的に記述可能な C++/CLI を開発言語としている．特に，これら各環境間のデータ型に係る変換ラッパーを構築することにより，CUDA ⇔ OpenCV ⇔ .Net 間のマルチプラットフォームとして開発を行った．俯瞰画像合成システムの動作例を図 1 に示す．システムの左部には各クライアントからの受信フレーム，右部には合成画像をそれぞれ併せて表示しており，フロア一面の俯瞰画像として適切に合成されていることを確認できる．

本システムにおける合成枚数とそれに係るフレームレートを図 3 に示す．クライアント側のカメラ解像度は 640 × 480，撮像フレームレートは 30 であり，サーバ側の実行環境は CPU:Core i7 4790K，RAM:16GB，GPU:Geforce 1060 6GB である．図 3 より，Werper 関数を含めて合成を行った場合，GPU を用いることで FPS が約 1.87 倍に改善している．一方，Compensator 関数と Blender 関数のみで合成を行った場合，CPU の方が約 1.16 倍高速に合成されている．この結果より，負荷の高い Warper 関数を GPU で処理することで大幅に高速化を図ることができるが，比較的負荷の低い Compensator 関数と Blender 関数のみでは GPU へのメモリアクセスにより FPS が劣化する可能性がある．したがって，Warper 関数をサーバとクライアントのいずれで処理するかに依って，GPU を使用するか否かの判断基準とすることができる．

6　あとがき

本研究では，複数の組み込みプロセッサと GPU を併用し俯瞰画像合成システムの開発を行った．今後の課題としては，クライアント間での段階的な画像合成，自由視点型画像合成への拡張，各クライアントからの受信フレームの同期化手法について検討を進める予定である．

謝辞　本研究の一部は，ウエスコ学術振興財団「平成 29 年度学術研究費助成」の支援を受けて実施された．

参考文献

[1] 高野 照久, 他, “視界支援用の車載カメラとして使用される魚眼カメラ画像列を用いた超解像の提案,” 生産研究, Vol.67, No.2, pp.99-104 (Jan.2015)

[2] 木香 里奈子, 近藤 真史, 他 “複数の組み込みプロセッサを用いた俯瞰画像の合成に関する基礎検討,” IEEE 広島支部学生シンポジウム論文集, pp.115-118 (Dec. 2017)

[3] 松岡 洋, “GPU 並列処理による OpenCV 関数高速化のポイント,” Interface, Vol.41, No.9, pp.118-126 (Sep. 2015)

[4] 田所 勇生, 近藤 真史, 他, “加算のみに帰着した高速射影変換法に基づくパノラマ画像合成システムの開発,” 電子情報通信学会総合大会, ISSS-SP-008 (Mar. 2018)