IBR に基づいた仮想都市のリアルタイムレンダリング およびデータ圧縮の効率化手法の提案

佐藤　亮† 三上　武志† 川崎　洋† 小野晋太郎†† 池内　克史††

† 埼玉大学 〒 338-8570　埼玉県さいたま市桜区下大久保 255
†† 東京大学 〒 153-8505　東京都目黒区駒場 4-6-1
E-mail: †{satou,mikami,kawasaki}@cgv.ics.saitama-u.ac.jp, ††{onoshin,ki}@cvl.iis.u-tokyo.ac.jp

あらまし　近年，都市のような広域な空間をコンピュータに取り込み仮想空間を生成することが，コンピュータビジョンやコンピュータグラフィックスの分野で注目されている．この仮想空間を実現する手法として，実写画像ベースで描く Image Based Rendering(IBR) がある．IBR は写実的な再現が行える反面，大量の画像データベースから必要なデータを選択し合成する手法であるため，広域空間のレンダリングにはデータサイズの観点から大きな問題がある．一方で，IBR におけるデータは冗長性が高いことから，多くの圧縮手法が提案されている．特に，少ない固有画像に圧縮する手法は，多次元画像解析の１つとしても盛んに研究されている．本稿では，近接フレームの対応をブロックマッチングを用いてより正確に行なうことで，固有画像による圧縮率を上げる手法を提案する．また，圧縮されたデータからの任意画像の復元を GPU で行なうことで，CPU に負荷をかけることなく高速なレンダリングを実現する手法もあわせて提案する．本手法の有効性を示すために，実際に車載カメラで撮影した都市の映像を用いて実験を行なったところ，良好な結果を得ることが出来た．

キーワード　仮想広域空間，IBR，KL 展開，シェーダ

Real-time Image Based Rendering Technique and Effcient Data Compression Method for Virtual City Modeling

Ryo SATO†, Takeshi MIKAMI†, Hiroshi KAWASAKI†, Shintaro ONO††, and Katsushi IKEUCHI††

† Saitama University 255 Shimo-Okubo, Sakura-ku, Saitama City, Saitama 338-8570, JAPAN
†† Tokyo University 4-6-1 Komaba Meguro-Ku, Tokyo 153-8505, JAPAN
E-mail: †{satou,mikami,kawasaki}@cgv.ics.saitama-u.ac.jp, ††{onoshin,ki}@cvl.iis.u-tokyo.ac.jp

Abstract　Construction of a large scale virtual scene like a city into a computer become an imporotant topic for Computer Vision(CV) and Computer Graphics(CG). One of the promising technique for the purpose is Image Based Rendering(IBR). IBR methods are effective for realistic rendering, but their huge data sets and restrictions on interactivity pose serious problems for an actual application. On the other hand, there are many compression methods proposed because IBR data sets are tedious. Especially, the compression method using eigen image is researched actively as one of the multi-dimensional image analyses. Thus, we propose a method for raising a compression ratio using eigen image, which uses blockmacthing technique to achieve a better correspondences between adjacent frames. We also propose a method to restore the compressed data using Graphic Processor Unit(GPU), which can achieve high-speed rendering without raising a load of CPU. We conducted an experiment using real images to verify the compression ratio of input data and effectiveness of GPU implementation and confirmed the impromvement.

Key words　virtual large-scale scene, IBR, KL expansion, shader

1. はじめに

近年，都市空間を計算機内に仮想的に再現する研究が盛んに行われている．これを実現する方法として，Model Based Rendering(MBR) と IBR の二つが知られている．MBR は，ポリゴン等の 3 次元データとその表面のテクスチャ情報を用いることで，あらゆる見えを容易に生成することができる．しかし，都市空間において，街路樹や電柱など細かい形状やテクスチャ等の情報を全てモデル化することは極めて難しい．

一方 IBR では，撮影した画像を直接利用するため，モデル化の難しい複雑な物体であっても容易に描画することが可能であるが，再現する空間すべての画像データを保持せねばならず，データの容量的な問題がある．また，IBR で用いるデータは同じ物体をあらゆる方向から撮影した画像群であるため，一般に冗長である．そこで，この冗長性を考慮し，効率よく画像を圧縮することが出来れば，データ容量を押さえて現実感の高い広域空間の描画が可能となる．

これを実現する手法として，車載カメラで撮影した全方位画像を対象とし，EPI(EpipolarPlane Image) を利用することで効率よく，KL 展開により画像を圧縮する手法 [1] が提案されている．KL 展開は微妙なズレ（誤差）などにより圧縮率が悪くなることが知られている．そこで，本稿ではブロックマッチングを適用し，より正確なトラッキングを行うことで圧縮率を向上する手法を提案する．また，線形演算による画像の復元を GPU で処理することで，高速にレンダリングする手法も合わせて提案する．これにより IBR による広域空間の再現を一般の PC 上でもリアルタイムで表示することができる．

2. 広域空間再現システムの概要

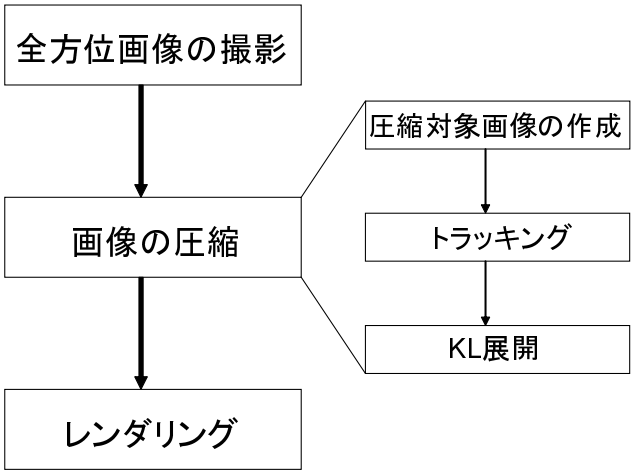


図 1　広域空間再現システムの流れ
Fig. 1　Overview of city modeling system.

本稿における広域空間再現システムは，全方位画像の撮影，全方位画像データの圧縮，圧縮データからのレンダリング，の３つの処理からなる (図 1)．以降では全方位画像の取得，画像の圧縮，レンダリングについて順に述べる．

2.1　全方位画像の取得

これまで IBR による広域空間のレンダリング手法が提案されている [2]〜[4]．その際，データ取得の効率化のために全方位画像がよく用いられている．全方位画像の取得方法として，鏡面型全方位カメラを用いる方法 [5] と，複数カメラを用いる方法 [6] がある．

鏡面型全方位カメラは，一度に水平方向 360 °の画像を撮影することができる利点がある反面，解像度が低く，現在の CCD 密度では現実感の高い都市のモデリングに十分な解像度を得ることが難しい．一方複数カメラを用いれば，解像度の問題は解決できるが，画像を統合することによる歪みが問題となる．しかし，この歪みについては多くの解決方法が提案されており，大きな問題とはならなくなりつつある [7]〜[9]．そこで，本稿においてもデータの撮影には図 2 のように複数カメラを車載して計測を行った．撮影された全方位画像の例を図 3 に示す．空間的な整合性をとりながら広い範囲が撮影できていることが分かる．

ところで，全方位画像は広い範囲を高解像度で撮影しているためデータサイズが大きい．さらに，IBR によるレンダリングの場合，わずかな空間を再現するだけでも非常に多くのデータが必要である．このため視点が移動する場合などにはメモリ上に次々と膨大なデータを読み込む必要があり，このため CPU や IO の負荷が大きく描画速度自体が低下してしまう問題があった．そこで，全方位画像の隣接するフレームでは変化が少なく，冗長性が高いという性質を利用して，画像を圧縮することを考える．同時にリアルタイムレンダリングのために，高速に復元可能な圧縮形式であることを考慮する必要がある．

図 2　車載カメラ
Fig. 2　Data capturing car.

3. 画像圧縮手法

3.1　概　　要

圧縮手順は図 1 に示したように，圧縮対象画像の生成，トラッキング，KL 展開の順に行い，トラッキングは EPI によるグローバルなトラッキングとブロックマッチングによる詳細なトラッキングを組み合わせた手法となっている．

図 3　全方位画像
Fig. 3　Omni-directional image.

3.2　圧縮対象画像

圧縮には，図 4 のような全方位画像を撮影経路に対して鉛直横向きに投影したものを用いる．これを用いるのは，3.3 節で述べるトラッキングにおいてオブジェクトの推移が EPI 上で直線となって表れるため，速度の推定が直線検出におきかえられ，トラッキングが容易になるためである．

図 4　圧縮対象画像
Fig. 4　Sample image for compression.

3.3　トラッキング

トラッキングはグローバルなトラッキングとその後のローカルなトラッキングに分けて行う．グローバルなトラッキングは EPI を用いた時空間画像解析により実現し，ローカルなトラッキングはブロックマッチングにより実現する．

3.3.1　時空間画像解析によるグローバルなトラッキング

等速直線運動する移動体に，カメラを進行方向に対して鉛直横向きに設置して撮影した連続画像を，時間軸方向に積層することで，時空間ポリゴンを作ることが出来る．この時空間ポリゴンをエピポーラ線に平行に切断したものが EPI（図 5）である．EPI に表れる直線の傾きは撮影地点と建物の距離によって変化する．本稿では，都市の建物が一定の面に乗っていると仮定し，複数ある EPI の直線の傾きの中から代表的な傾きを選択することでオブジェクトのグローバルなトラッキングを行う．このため，Canny オペレータを用いて代表的な傾きのみを求め，他の直線は閾値調整で除外し，この 2 値化画像に対してハフ変換を行う．図 7 は Canny オペレータによって図 5 の代表的な傾き（図 6）を検出した例を示したものである．

図 5　EPI 画 像
Fig. 5　EPI.

図 6　Canny オペレータによって EPI の直線のエッジを検出
Fig. 6　Edge detected by Canny operater.

図 7　Canny オペレータによって検出された直線
Fig. 7　Detected line by Canny operater.

EPI において直線の傾きはオブジェクトの奥行きと車速を表しているため，このパターンを抽出できればオブジェクトのトラッキングが可能となる．そこで EPI の直線の傾きをハフ変換を用いて検出する．求めた傾きを θ，EPI の画像座標を (u, v) とすると，

$$\frac{\delta u}{\delta v} = -\tan\theta \tag{1}$$

である．EPI 上では v 軸が時間に相当するため，1 フレームでのオブジェクトの u 方向の移動量は $-\tan\theta$ である．これがグローバルなトラッキングパラメータとなる．

3.3.2　ブロックマッチングによるローカルなトラッキング

図 8 を用いて手法を説明する．1 フレームでのオブジェクトの u 方向の移動量は式 (1) で求められているため，誤差などが無ければフレーム f_1 で黒枠で囲まれたブロック画像は時空間画像内でフレームが f_2, f_3 と進むに従い，黒破線枠で示されるように移動する．実際には，車の振動や速度の微小変化によりオブジェクトの移動量は u からずれるため，前後のフレーム間でブロックマッチングを行い，前フレームのブロック画像との相関値が最も高い画像（図 8 黒太線枠）を獲得していく．この作業により，EPI に基づいたオブジェクトの推移（破線）のみよりも正確にトラッキングが行え，より類似した画像群を獲得できる．

ブロックマッチングで用いる相関値（エネルギー関数）としては，線形相関係数を用いている．任意の二つの画像 X, Y の座標 (i, j) の画素値を x_{ij}, y_{ij}，それらの平均値を $\bar{x}, \bar{y}$ で示したときの相関係数 r は

$$r = \frac{\sum_i \sum_j (x_{ij} - \bar{x})(y_{ij} - \bar{y})}{\sqrt{\sum_i \sum_j (x_{ij} - \bar{x})^2} \sqrt{\sum_i \sum_j (y_{ij} - \bar{y})^2}} \tag{2}$$

であり，すべての画素値が同じ画像（すなわち全ての i, j において $x_{ij} = y_{ij}$）において 1 となる．

3.4　KL 展開による次元圧縮

まず，第 1 フレームを図 8 のように $m \times n$ のブロック単位に分割する．それぞれのブロック画像に対して，第 d フレームまで 3.3 節のトラッキングを適用し，類似ブロック集合を得る．こうして生成したブロック集合を行列にし，その行列に対して KL 展開をすることで k 次元の固有ベクトル（k 枚の固有画像）

に圧縮することができる．これを $m \times n$ のすべてのブロック画像に対して行うことで画像列を圧縮することができる．このとき，類似ブロック集合はブロックマッチングにより相関値が高い画像群が得られているため，高い圧縮率が期待できる．一般的な動画圧縮技術である MPEG なども適用することは可能だが，ランダムアクセスが必要な本手法には不向きである．

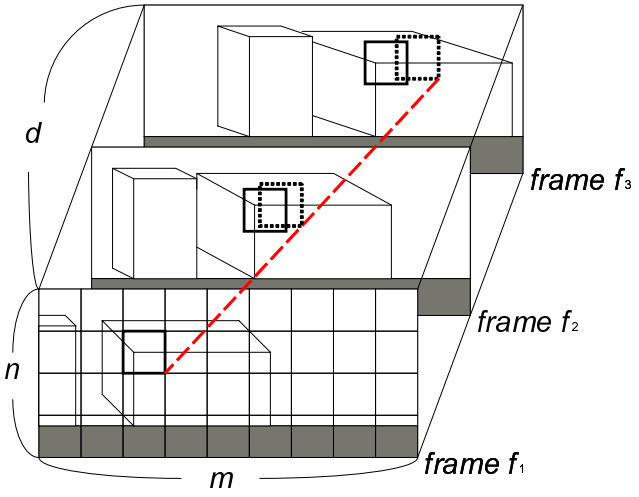


図 8　ブロックマッチングを適用したトラッキング
Fig. 8　Tracking with EPI and blockmatching.

4. レンダリング

4.1 画像の復元

画像の復元は KL 展開によって得られた固有画像を用いて，オリジナルの i フレーム目の画像は以下に示す線形和により復元することが出来る（式 (3)）．ここで，V_k は，k 番目の固有画像，w_{ik} は，フレーム i の重み係数である．また，r は，ある累積寄与率達成するために必要な主成分の枚数である．

$$X_i = \sum_{k=1}^{r} (w_{ik} * V_k) \tag{3}$$

4.2 レンダリングアルゴリズム

ここでは，圧縮したデータを用いたレンダリング手法について図 9 を用いて述べる．ある視点 P におけるレンダリングは，撮影経路の A，B，C において撮影された画像群の中から視点 P と壁面中心との成す角に対応するテクスチャをそれぞれ貼ることで実現される [3]．このとき撮影された画像群は圧縮され固有画像となっているので，例えば撮影地点 A におけるテクスチャは，ある累積寄与率を満たす固有画像 k 枚と θ_1 に対応するパラメータとの積和演算で復元することによって得られる [1]．本論文では，$m \times n$ のブロック画像に分割し KL 圧縮を行っているので，レンダリングは $m \times n$ のブロック画像それぞれを全て復元する必要がある．

4.3 GPU 処理による圧縮データの復元

原画像を復元する手法を図 10 を用いて説明する．従来手法では，はじめに固有画像をメモリに読み込んでおき，これと

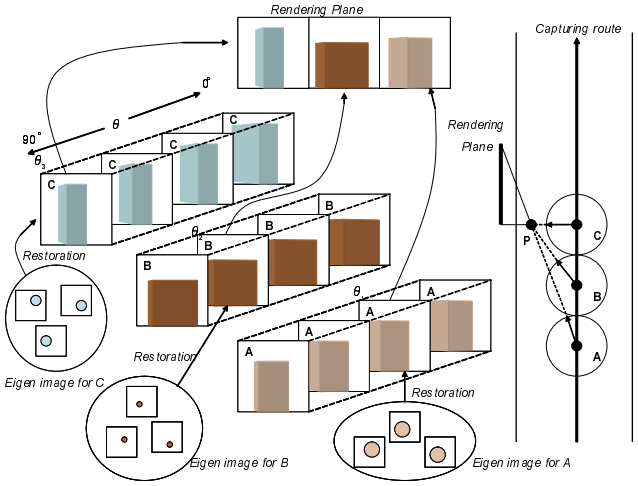
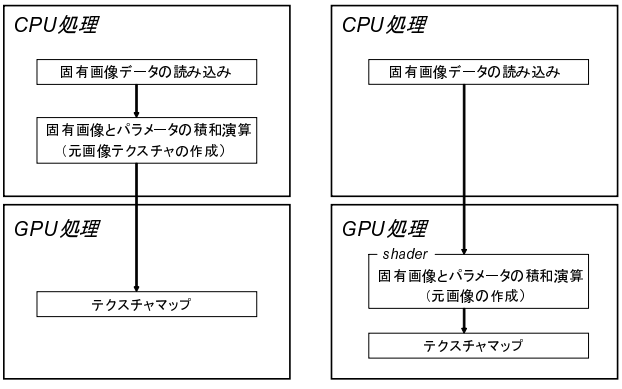


図 9　レンダリングアルゴリズム
Fig. 9　Rendering algorithm.

原画像に対応するパラメータとの積和演算を CPU で行う（図 10(a)）．CPU では，1 ピクセル単位での積和演算を順番に各ピクセル全てで行うことでレンダリング画像を生成する．こうして得られた画像をテクスチャとして GPU に渡し，テクスチャマップする．テクスチャサイズを $n \times n$ とすると計算量が $O(n^2)$ であるため，テクスチャのサイズが大きくなると CPU の負荷が急激に増加する．

一方，提案手法である GPU による圧縮の復元手法は固有画像を読み込んだあと，これをテクスチャとしてあらかじめ GPU に渡しておき，パラメータとの積和演算は GPU 内におけるピクセルシェーダのみで行われる（図 10(b)）．ピクセルシェーダは高速な並列プロセッサのためフレームレートでの処理が実現できる．これにより CPU に負荷をかけることなく原画像の復元が可能となる．またテクスチャのサイズが大きくなっても，GPU の処理能力の範囲内であれば計算負荷を一定とすることができる．



(a)　(b)

図 10　(a) 従来手法（CPU による演算），(b) 提案手法（GPU による演算）
Fig. 10　(a)CPU method and (b)GPU method.

5. 実　　験

5.1 トラッキング結果

各全方位画像を 16×16 に分割し，70 フレームの画像列を用いて圧縮の実験をおこなった．図 11,13 は，EPI だけを用いたトラッキングで得られたブロック画像，図 12,14 は EPI とブロックマッチングを用いたトラッキングで得られたブロック画像の結果である．図 11 において上下左右方向の揺れが見られるが，図 12 ではその揺れが見られないことが分かる．ただし，ここで示したように実際の建物などは三次元情報を持つため，視点方向が変わるに従い見えが変化するため，累積寄与率が高くなるに従い圧縮率の改善率は減少する．図 15 は，ブロック画像とその固有値（正規化処理後）の関係のグラフであり，従来手法に比べ提案手法が第 1,2,3 主成分において上回っており，第 1 主成分の固有値の重み付けが高くなっている．これにより，より類似したブロック画像群を獲得することができたことが分かる．また，図 16 は，固有画像に対する累積寄与率を示している．特に少ない枚数のときに，提案手法において急速に立ち上がっており，少ない枚数で画像を再現できていることが分かる．表 1 は，これらの圧縮に関してまとめたものであり，トラッキング精度の向上でより類似したブロック画像群が獲得でき，その結果圧縮率が向上したことが分かる．

図 11　EPI のみに基づいたトラッキング結果画像 1
Fig. 11　Result of tracking1(EPI).

図 12　ブロックマッチングを用いたトラッキング結果画像 1
Fig. 12　Result of tracking1(blockmatching).

5.2 レンダリング処理時間の比較

従来手法と今回提案する手法とのレンダリング処理時間の比較を行った．実験方法は，オリジナル画像 70 枚を順番に復元するという過程を 100 回繰り返す処理をしたときにかかる時間を計測する方法で，画像のサイズを変更しながらその処理時間の変化の様子をプロットした．用いた PC のスペックは Intel Core2(2.66GHz)，GPU は Quadro FX 550 である．図 17 の

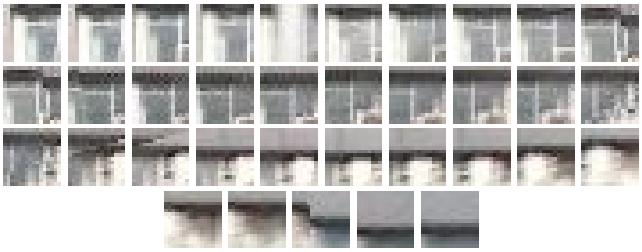


図 13　EPI のみに基づいたトラッキング結果画像 2
Fig. 13　Result of tracking2(EPI).

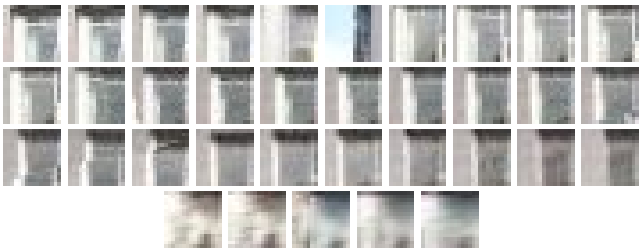


図 14　ブロックマッチングを用いたトラッキング結果画像 2
Fig. 14　Result of tracking2(blockmatching).

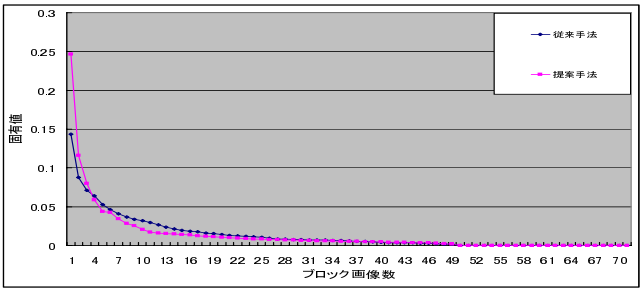


図 15　ブロック画像と固有値の関係
Fig. 15　Eigen value.

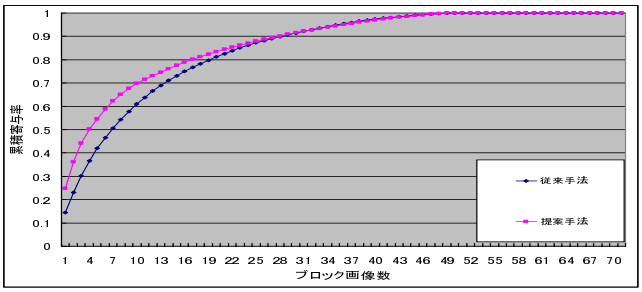


図 16　累積寄与率
Fig. 16　Cumulative propotion.

グラフがその実験の結果であり，横軸が画像のサイズ，縦軸が処理時間（実験 10 回の平均値）となっている．64×64 サイズ以下の画像までは，どちらもほぼ同じ処理時間であったが，画像サイズが大きくなるにつれて従来手法（CPU）における処理時間が急激に増加しているのが分かる．一方，提案手法による演算処理は，画像のサイズに関係なく一定の処理時間を保っている．この結果から GPU を用いたレンダリング手法により，高速かつ安定したレンダリングが実現できることが示された．

例 1	従来手法	提案手法	改善率
前後相関値の平均	0.756491	0.936779	-
累積寄与率 50 %（データサイズ圧縮率）	7.3 枚（40/280KB）	4.3 枚（36/280KB）	41 %（10 %）
累積寄与率 70 %（データサイズ圧縮率）	14.0 枚（60/280KB）	10.6 枚（48/280KB）	24 %（20 %）
累積寄与率 90 %（データサイズ圧縮率）	28.6 枚（128/280KB）	28.3 枚（124/280KB）	1 %（3 %）

例 2	従来手法	提案手法	改善率
前後相関値の平均	0.842592	0.929996	-
累積寄与率 50 %（データサイズ圧縮率）	7.0 枚（36/280KB）	5.0 枚（36/280KB）	29 %（0 %）
累積寄与率 70 %（データサイズ圧縮率）	15.0 枚（72/280KB）	12.0 枚（60/280KB）	20 %（17 %）
累積寄与率 90 %（データサイズ圧縮率）	30.3 枚（132/280KB）	28.0 枚（120/280KB）	8 %（9 %）

表 1　圧 縮 結 果
Table 1　Result of compression.

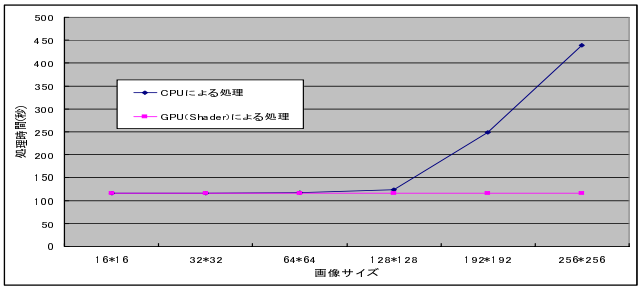


図 17　処理時間の比較実験結果
Fig. 17　Comparison of processing time.

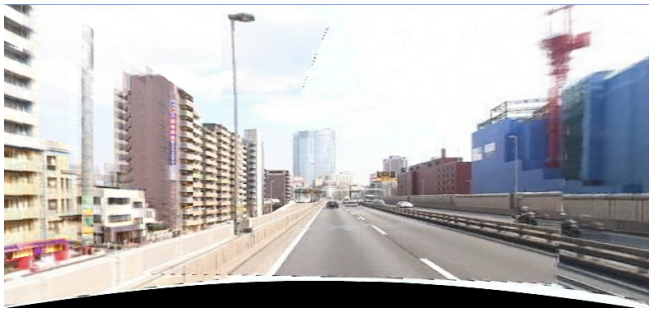
5.3　レンダリング結果

提案手法を用いて 4.2 節のレンダリングアルゴリズムにより実際に広域空間のレンダリングを行った．結果を図 18 示す．どちらも累積寄与率 75 %を満たす固有画像を用いてレンダリングした結果である．約 20 %の圧縮率の改善を実現しているにも関わらず，GPU を用いた処理により CPU とほとんど区別の無いレンダリングがされていることが分かる．

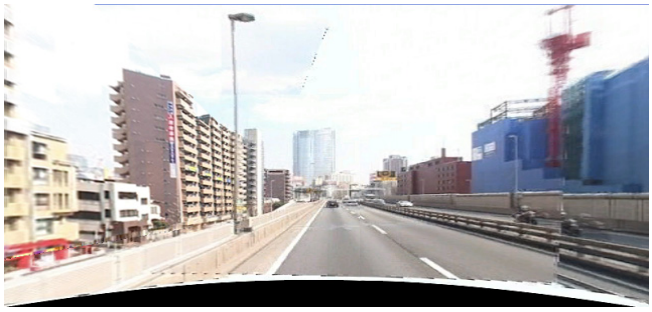
6.　ま　と　め

本稿では，全方位画像列の KL 展開による圧縮率を向上する手法および，圧縮されたデータを GPU を用いて高速に復元する手法について述べた．KL 展開による次元圧縮はシフトなどのズレに弱いため，ブロックマッチングを用いたトラッキングを行うことで，より類似したブロック画像群を得て，圧縮率を向上する手法を提案した．

圧縮されたデータは固有画像とその重み係数の積和演算で復元することができる．この演算はすべて線形演算であり，その演算をピクセルシェーダで処理することでより高速で安定したレンダリングを実現した．これは，固有画像と重み係数をシェーダにあらかじめ渡しておき，GPU 内でその固有画像と重み係数を積和演算することで CPU に負荷をかけることなく演算することで実現される．



(a)



(b)

図 18　レンダリング結果（(a) 従来手法，(b) 提案手法）
Fig. 18　Result of rendering.

文　　献

[1] 尾池，川崎，大澤，小野，池内：”固有画像を用いたイメージベースによる仮想広域空間の実時間レンダリング手法”，画像の認識・理解シンポジウム,July 2005.

[2] C.J.Taylor：”Video plus”，IEEE Workshop on Omnidirectional Vision,pp.3-11(2000).

[3] H.Kawasaki，K.Ikeuchi，and M.Sakauchi：”Light field rendering for image-scale scenes”，CVPR,Vol.2,Kauai,Hawaii,USA,pp.64-71(2001).

[4] 堀，神原，横矢：”被写体距離を考慮した Image-Based Rendering による広域屋外環境のステレオ画像生成”，信学技報，IEICE Technical Report，PRMU2006-185(2007).

[5] Y.Onoue, K.Yamasawa, H.Takemura, and N.Yokoya：”Telepresence by realtime view-dependent image generation from omnidirectional video streams”，In Computer Vision and Image Understanding, 第 71 巻,pp.154-165,Aug(1998).

[6] 山本，棚橋，桑島，丹羽：”実環境センシングのための全方位ステレオシステム (SOS)”，電学論 C　電子・情報・システム部門誌,Vol.121-C,pp.876-881(2001).

[7] 三上：”球面時空間画像解析による揺れの無い全方位画像列の自動生成手法”，埼玉大学大学院修士論文 (2006 年度).

[8] 小野，川崎，池内，坂内：”EPI 解析による複数ビデオカメラの画像統合”，Computer Vision and Image Media 2003(2003).

[9] P.G.R.Inc.:Ladybug：`"http://ptgrey.com/."`