ウォールディスプレイを用いたインタラクティブフォトモザイク
Interactive Photomosaic Using Wall Display

奈良先端科学技術大学院大学　　○ 藤澤 誠，加藤博一
Makoto Fujisawa and Hirokazu Kato
Nara Institute of Science and Technology

Abstract A photomosaic is a type of decorative art and is made up of various other pictures. This paper presents a method for fast generating the photomosaic, and propose an interactive recursive photomosaic system. The users can operate it by using a wall display with touch input and its appearance will be changed dynamically.

1 はじめに

バーチャルリアリティを用いたインタラクティブなアプリケーションにおいて，ユーザに提示する情報のほとんどはCRT, 液晶などのディスプレイを通して与えられる．ディスプレイは，近年，VGA(640×480)，SXGA(1280×1024) からハイビジョン (1920×1080) へと高解像度化が進んでおり，さらに，今後，4k(4096×2160)，そして，スーパーハイビジョン (7680×4320) へと映像の高精細化が勧められると考えられる．そのため，そのコンテンツもこの高解像度を生かしたものが必要である．

これまでにも，高解像度ディスプレイを利用したコンテンツとして，デジタルプラネタリウム，デジタルミュージアム，シミュレーションの視覚化などが提案されてきた．これらは映像の高精細化による臨場感の向上，提示できる情報の拡大を実現している．一方，映像の高精細化は視聴距離の変化に対応できるという特徴を持つ．例えば，従来はディスプレイに近づくと映像のドットが目立ってしまうような場合でも，高解像度になると近くて見ても映像が判別できる．本研究では，視聴する距離に対して印象が変化するコンテンツとして，フォトモザイクに注目し，高解像度のフォトモザイクを GPU を用いた並列化によりリアルタイムに生成する手法を開発し，さらに，タッチパネル機能付きのウォールディスプレイを用いることで，直感的なインタラクションを加えたインタラクティブフォトモザイクアプリケーションを提案する．

フォトモザイクは写真を要素としたモザイク画の一種であり，芸術作品として広く知られている (図 1 参照)．元の画像をタイルと呼ばれる小さな領域で分割し，そこにデータベースに格納された多くの写真 (タイル画像) をあるルールに従って当てはめることで，新しい画像を生成する．ルールを例えば，色や模様の近似とすることで，元の画像を近似することができる．これまでのフォトモザイクは，特にコンピュータを用いて作成したとき，タイルを構成しているタイル画像を画面上で確認することは解像度の問題から難しかった．そのため，タイルを元画像の輪郭に合わせて変形したり，大きさを適用的に変えたりするなどの手法がとられてきた．しかし，ハイビジョンや 4k などの高解像度ディスプレイならば，要素画像を視覚的に確認することも可能となる．

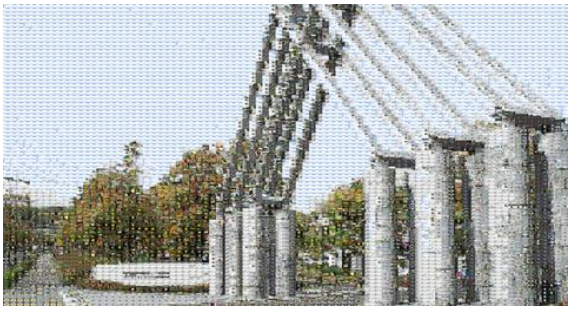

図 1: フォトモザイクの例

我々の高解像度インタラクティブフォトモザイクは絵の中にさらなる絵が隠されているというフォトモザイクの再帰性を拡張し，タイルを構成する画像もまたフォトモザイクであるという特徴を持たせる．これにより，例えば，写真間の接続関係を利用し，人と人，人と物，物と物との関係性などを表現することも可能となる．そして，高解像度ディスプレイの利用により，ユーザはタイル画像をある程度識別でき，ウォール型ディスプレイによりタッチ操作で見たいタイルを選択し，拡大することで，新しい情報を得ることが出来る．これを実現するためには，高解像度フォトモザイクの高速な生成が必要となる．我々は，GPU を用いて高解像度フォトモザイクを生成する方法を提案する．提案手法では，CPU による画像選択と GPU による画像合成を組み合わせることで，オンラインでの画像生成を可能とする．

2 Related Work

絵画やデジタルアートの分野において，異なる画像の集合で別の画像を表現する技法として，16 世紀後半に Arcimboldo が人物画を果物や動物，花などの集合で表した [9]．また，写真をタイルとして用いたフォトモザイクは，Silvers[8] によって開発された．Silvers は矩形のタイルを格子状に配置し，元の画像の一部分と色分布がにている画像を当てはめていくことで，写真の基本的な形状である矩形を維持したまま，元の画像をよく再現したモザイク画を生成した．また，Silvers はタイルを構成する写真の集合に，元の写真との関連性を持たせることで，より人々の興味を引く作品にしている．

ノンフォトリアリスティックレンダリングの分野では，モザイク画の一種であるタイル画 [4] や Arcimboldo の作品のような様々な形状の集合画 [5, 2] などを再現する手法が提案されている．これらの方法は，画像から得られた色の境界に合わせて，別の形状を配置する．グリッド状にタイルを配置するフォトモザイクの生成手法として，Tran[10] はタイルとそれを配置するグリッドを比較し，各ピクセルの RGB 成分の差の合計を L_1 距離として，その最小値を画像群の中から検索することで，元の画像に類似したフォトモザイクを生成した．しかし，この手法では，タイルの全ピクセルを参照する必要があり，リアルタイムでの計算には不向きである．より高速なフォトモザイクの生成のために，Di Blasi ら [3] はタイル画像を 9 分割 (3×3) し，各分割領域の平均色を計算することで得られる各画像の特徴を表す 27 次元のベクトルを用いた．彼らは元画像をグリッド状に分割した矩形領域においても同様のベクトル量を求め，ベクトル空間における最近傍画像を探索しあてはめていくことで，フォトモザイクを生成した．

モザイク画をインタラクティブなアプリケーションに応用する研究として，Kim ら [6] はモザイク画を多数の動画のナビゲーションに用いた．Klein ら [7] は動画を用いたモザイク手法を提案した．本研究ではウォールディスプレイを用いたタイルの選択，高解像度フォトモザイク画像の再帰的な生成により，新しいフォトモザイクのアプリケーションを提案する．

3 フォトモザイク

本研究で用いるフォトモザイクは，元の画像 (サイズ $w_o \times h_o$) を格子状に分割し，各格子 (タイル) 領域 (サイズ $w_t \times h_t$) の色特徴ベクトルから類似画像を検索し，あてはめていくことで合成される．色特徴ベクトルにはタイルを 4 分割 (2×2) したときの各分割領域の平均色 $T_r^{ij}, T_g^{ij}, T_b^{ij}$ $(i, j = 0, 1)$ からなる 12 次元ベクトルを用いる．

フォトモザイク生成の全体の流れを図 2 に示す．まず，前処理として，データベースに登録された画像の色特徴ベクトルを計算し，それを検索を高速化するためのツリー構造に格納する．さらに，タイルの大きさがあらかじめ決まっている場合，画像をタイルの大きさにリサイズする．リサイズされた画像は 3.1 節に示すように 1 枚の画像としてテクスチャに転送される．

フォトモザイクを生成するときは，元の画像を読み込んだ後，その画像をグリッド分割，各グリッドの色特徴ベクトルを計算し，前処理で構築されたツリーを用いて，最近傍探索により類似画像を取得する．最後に，GPU を用いた並列処理によりフォトモザイク画像を合成する．この章では，これらの各処理について詳しく述べる．

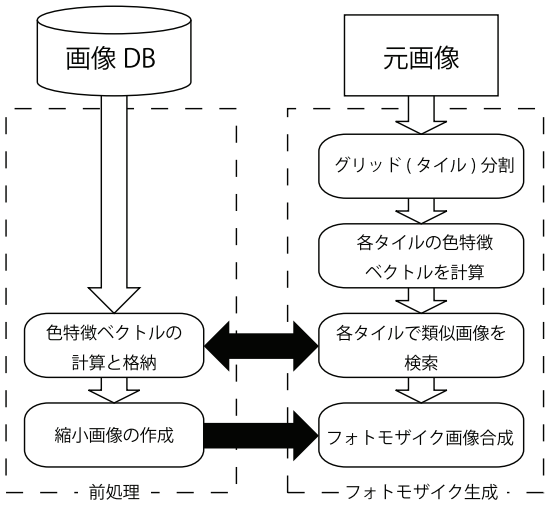

図 2: フォトモザイク生成手順

3.1 前処理

タイルとして用いる画像群に対して，4 分割した各領域の平均色からなる色特徴ベクトル $(T_r^{ij}, T_g^{ij}, T_b^{ij}$ $(i, j = 0, 1))$ とタイルの大きさにリサイズした画像を求める．各領域の平均色は RGB で表されるので，画像の色特徴量は 12 次元のベクトルとして表される．

色特徴ベクトルの値は近傍探索のために，ANN (Approximate Nearest Neighbor) KD 木 [1] に格納される．フォトモザイクでは類似画像の完全性は必要でないことから，近似的なアルゴリズムではあるが，高次元においても高速に処理可能な近傍探索手法である ANN を用いた．また，フォトモザイク処理における計算コストがかかる部分である，画像のリサイズ処理もあらかじめ行っ

ておく．リサイズされた画像は，画像合成時のアクセスのしやすさを考慮し，図 3 に示すように画像を一次元配列に再配置することで，データベース内のすべてのリサイズ画像を 1 枚の画像として格納し，さらに，GPU でのモザイクが合成のために，テクスチャメモリに転送する．図 3 のような格納方式にしたのは，データベース内の画像枚数に関係なく，効率的に画像を格納するためである．タイルに用いる画像の枚数を N とすると，リサイズ画像を格納するテクスチャのサイズは，$w_t h_t \times N$ となる．

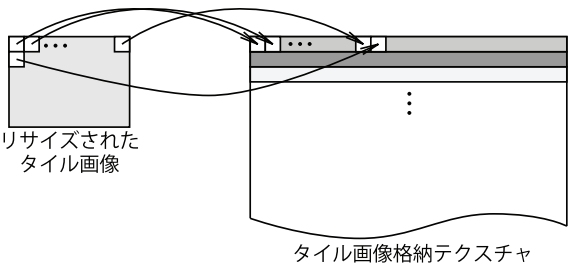

図 3: タイルに用いるリサイズ画像の格納

3.2 マッチング画像探索

前処理により，タイル画像の色特徴ベクトル場とリサイズ画像格納テクスチャが作成されたので，これらを用いて元画像に当てはめるべき画像のマッチングを行う．まず，元になる画像を読み込む．元画像はタイルサイズに対して，縦横ともに整数倍の大きさでなければならない．つまり，

$$w_o = nw_t \tag{1}$$

$$h_o = mh_t \tag{2}$$

となる自然数の組 (n, m) が存在しなければならない．そのため，元画像をあらかじめリサイズしておくか，読み込んだときにその大きさを変更する．

読み込まれた画像を $n \times m$ に分割し，各分割領域における色特徴ベクトルを計算し，その値から近傍探索により類似画像を探索する．近傍探索には 3.1 節でも述べたように，ANN を用いる．ANN のようなツリー型探索は GPU 上での実行に適していないため，ANN による画像マッチングは CPU 上で行う．ANN を用いた場合，CPU 上でもインタラクティブに実行するための十分な速度が得られる．また，なるべく多くの種類の画像をモザイクに用いたいため，最近傍 1 つではなく N 個の画像を探索し，その中からランダムに画像を決定する．マッチング結果は，メインメモリに格納され，フォトモザイク生成後も保持される．これは高解像度ディスプレイの特性を生かしたインタフェースとして，4 章で述べるようにタイルをユーザが選択出来るようにするためである．

3.3 GPU によるモザイク画像生成

マッチング結果とリサイズされたタイル画像からフォトモザイクを生成する．生成されたフォトモザイクは GPU を用いて画面表示するため，メインメモリ上で CPU を用いてモザイク生成した場合，特に高解像度画像の場合，メインメモリから GPU への転送時間が増大し，リアルタイムでの生成が難しくなる．さらに，我々は再帰的な画像選択とそのズームアニメーションのために，選択された画像とその周囲 8 画像のモザイクを生成・描画するため，転送時間は 9 倍になる．そのため，生成された画像を GPU に転送するのではなく，GPU 内でモザイク画を生成する．

GPU の基本的な構造を図 4 に示す．図中点線の枠で示したピクセルシェーダでフォトモザイク生成を行う．シェーダは複数存在し (NVIDIA GeForce GTX285 で最大 240 個)，描画画面上の各ピクセルごとに並列に処理することができるため，各ピクセルを画像の各画素に割り当てることで高速な並列計算が可能である．しかし，GPU は汎用計算用に作られておらず，プログラミング言語で一般的に用いられている配列などのデータ構造に格納したデータを直接渡すことはできない．その代わりに多くのデータを一度に渡すことが出来るテクスチャを用いる．図 4 のようにピクセルシェーダはラスタライザからのデータとテクスチャデータを受け取る．メインメモリに格納されたマッチング結果を $n \times m$ の大きさのテクスチャデータとして転送することで，ピクセルシェーダに値を渡すことができる．マッチング結果は $[1, N]$ の自然数であるため，OpenGL 拡張である `GL_EXT_texture_integer` を用いて，整数値を変換無しに直接転送する．

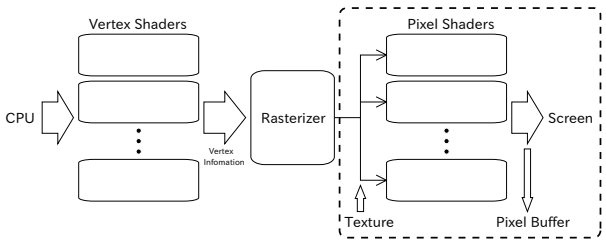

図 4: GPU の構造

リサイズされたタイル画像群を格納したテクスチャは，前処理の段階で GPU に転送しているため，モザイク生

成ステップではマッチング結果のみを受け取る．マッチング結果は分割数と同じピクセル数の画像として転送されるため，その転送コストは画像そのものを転送するのに比べて非常に小さい．

図 5 にピクセルシェーダを用いたフォトモザイク画像の合成手法を示す．ピクセルシェーダはピクセルごとに処理されるため，モザイクを行う元の画像と同じ大きさのビューポートを設定し，矩形ポリゴンを描画することで対応するテクスチャ座標から元画像上のピクセル位置 (i, j) を得る．ピクセル (i, j) において，タイル座標 (t_x, t_y) は

$$t_x = \mathrm{floor}(i/w_t) \tag{3}$$

$$t_y = \mathrm{floor}(j/h_t) \tag{4}$$

である．ここで，floor は小数点以下切り捨てを表す．マッチング結果を格納したテクスチャの (t_x, t_y) の値を参照することで対応する画像の番号を取得する．また，タイル内のローカルピクセル位置 $(t_i, t_j) = (i - t_x w_t, j - t_y h_t)$ からタイル画像格納テクスチャの x 座標値を計算し，最終的なピクセル色を得る．これらの処理はピクセルシェーダを用いて，ピクセルごとに並列に実行される．GPU 内で描画すべきフォトモザイク画像を生成することで，メインメモリから GPU 上のビデオメモリへの転送の必要性がなくなり，特に高解像度画像を用いた場合の高速化が可能である．

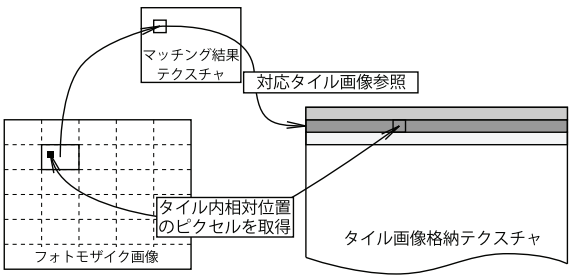

図 5: フォトモザイク合成

4 再帰的なフォトモザイク

高解像度フォトモザイクをナビゲーションなどに応用するために，再帰的なフォトモザイクアプリケーションを提案する．このアプリケーションでは，高解像度ディスプレイに表示されたフォトモザイクから，ユーザが任意の画像を選択する．高解像度ディスプレイにより各タイル画像をユーザはある程度判別することが可能である．選択された画像を元に，新たにフォトモザイクをダイナミックに生成し，画像を最大化する．このとき，ズームするようなエフェクトをかけることで，ユーザはあたかもすべてのタイル画像がフォトモザイクで構成されている感覚を得る．また，拡大するときに周囲 1 周りの画像も含めてフォトモザイクを生成しておくことで，拡大時の不自然さを軽減する．

図 6: ウォールディスプレイを用いたインタラクティブなフォトモザイクの操作

この再帰的なフォトモザイクはそのまま，エンターテイメントとして利用するほかに，ユーザが興味のあるタイルを選択していくことで，必要な情報を最終的に得るための，ナビゲーションの作業にエンターテイメント性を持たすことや，子供の教育のために用いるなどの応用が考えられる．

5 結果

再帰的なフォトモザイクを実装し，ウォールディスプレイに表示した結果を示す．ウォールディスプレイとして，タッチパネル機能のための赤外線イメージセンサを持つ，3 台のプロジェクタによるバックプロジェクションによる 250 インチのディスプレイを用いた．1 台のプロジェクタの解像度は 1400×1050 であり，3 台を水平方向に並べてあるため，総解像度は 4200×1050 である．実験では 2 台のプロジェクタを用い，2800×1050 の解像度の表示領域にハイビジョン解像度 (1920×1080) の画像を描画した．図 6 はインタラクティブなフォトモザイクの操作の様子を示している．実験で用いたウォールディスプレイは赤外線イメージセンサによるタッチパネル機能を有しており，フォトモザイク中のタイルを触ることにより，そのタイルが拡大され，再帰的にフォトモザイクが描画される．タイル画像を選択したときにダイナミックにフォトモザイクを生成することで，この再帰的なフォトモザイクを可能としている．フォトモザイクからフォトモザイクへの拡大処理の間のフレームを図 8 に示す．

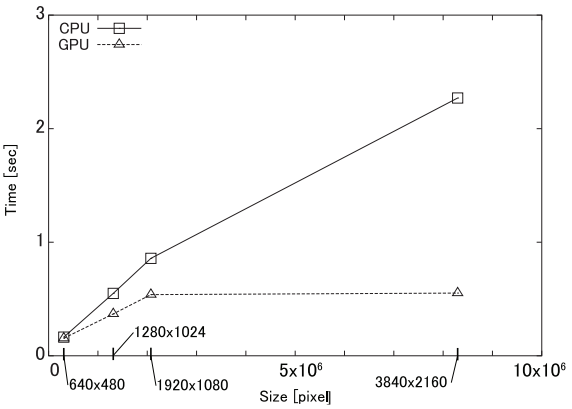

図 7: フォトモザイク生成時間の比較

選択からズーミングの間にフォトモザイクの生成時間が遅いとユーザは違和感を感じてしまう．そのために GPU による高速化を行った．CPU と GPU による計算時間の比較結果を図 7 に示す．Intel Core 2 Duo 3.16GHz, ビデオカードとして NVIDIA GeForce GTX 285 (ビデオメモリ 1GB) を搭載した PC 上で実行した．画像によく用いられる解像度 (640×480(VGA), 1280×1024(SXGA), 1920×1080(HD), 3840×2160(HD の縦横 2 倍)) における画像を選択してから拡大を始めるまでの処理時間を 10 回計測し，その平均をグラフに示した．拡大するときに周囲の画像も含めてフォトモザイクを生成するため，合計 9 枚のモザイク画像生成にかかった時間である．解像度が低い場合は，計算時間に対する画像の読込み時間の割合が大きく GPU 処理のアドバンテージは小さい．しかし，HD を超える解像度になると画像のテクスチャへの転送時間の割合が大きくなり，転送すべき情報が少なくすむ GPU 処理と CPU 処理の差が大きくなっている．GPU 処理では，HD 解像度やその 4 倍の解像度においても，1 枚のフォトモザイク生成時間は約 0.06 秒であり，ほとんどタイムラグを感じることはないが，9 枚の処理では約 0.5 秒であり，さらなる高速化が必要である．

6 おわりに

本研究では，タッチパネル機能付きのウォールディスプレイを用いたインタラクティブフォトモザイクを提案した．高解像度ディスプレイにおける視聴距離が小さいときの映像の視認性の向上を利用し，視聴距離に対して印象が変化するコンテンツとしてフォトモザイクに注目し，高解像度のフォトモザイクを高速に生成する手法を開発した．高速化に GPU を用いたモザイク合成処理の並列処理を導入することにより，高解像画像の GPU への転送速度の遅さの解消，解像度の変化に対するロバスト性の向上，ハイビジョンの 4 倍の解像度においても高速なフォトモザイク生成を成し遂げた．しかしながら，拡大処理において，視覚的な不自然さを解消するために，9 枚のフォトモザイクをリアルタイムで生成するためにはさらなる高速化が必要である．今後の研究としては，フォトモザイクの写真間の接続関係を利用し，人と人との関係性などを表現することに応用する，動画像に対するフォトモザイク生成などがあげられる．

参考文献

[1] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[2] G. Di Blasi, G. Gallo, and M. P. Petralia. Puzzle image mosaic. In *Proc. of IASTED/VIIP2005*, 2005.

[3] G. Di Blasi, G. Gallo, and M. P. Petralia. Smart ideas for photomosaic rendering. In *Proceedings of Eurographics Italian Chapter Conference*, 2006.

[4] Alejo Hausner. Simulating decorative mosaics. In *Proceedings of SIGGRAPH 2001*, pages 573–580, New York, NY, USA, 2001. ACM.

[5] Junhwan Kim and Fabio Pellacini. Jigsaw image mosaics. In *Proceedings of SIGGRAPH 2002*, pages 657–664, New York, NY, USA, 2002. ACM.

[6] Kihwan Kim, Irfan Essa, and Gregory D. Abowd. Interactive mosaic generation for video navigation. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 655–658, New York, NY, USA, 2006. ACM.

[7] Allison W. Klein, Tyler Grant, Adam Finkelstein, and Michael F. Cohen. Video mosaics. In *NPAR 2002: Second International Symposium on Non Photorealistic Rendering*, pages 21–28, June 2002.

[8] Robert Silvers. *Photomosaics*. Henry Holt and Co., Inc., New York, NY, USA, 1997.

[9] Claudia Strand. *Hello, Fruit Face!: The Paintings of Guiseppe Arcimboldo*. Prestel, 1999.

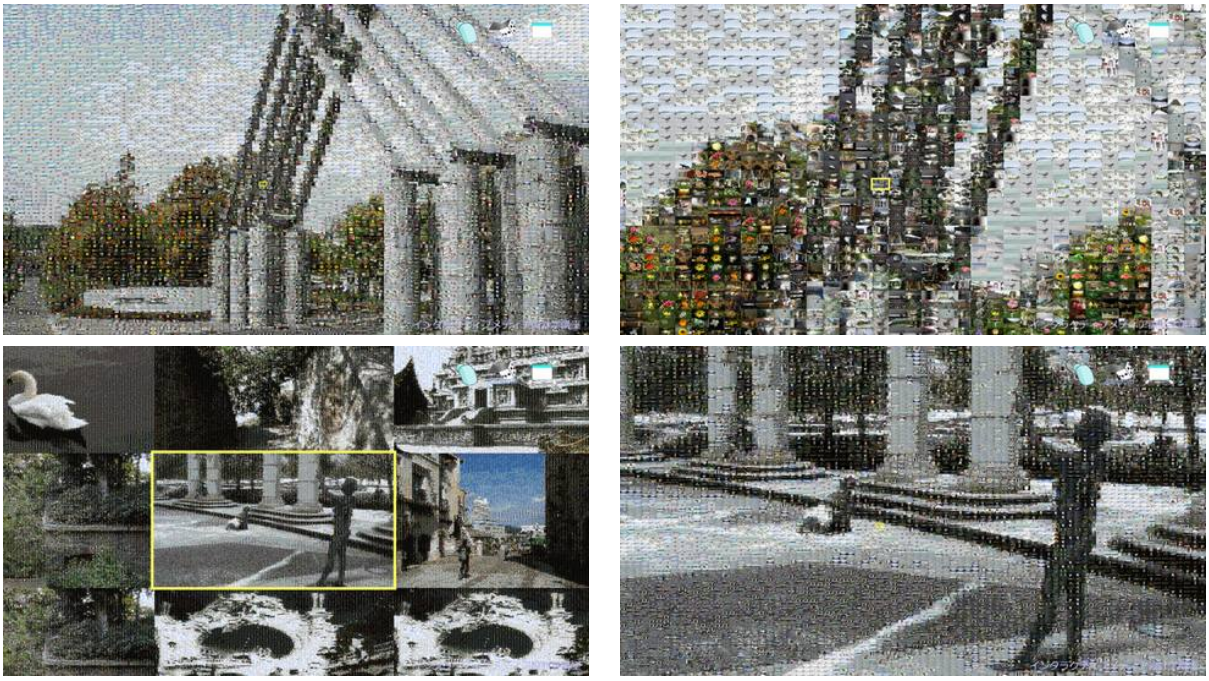

図 8: 再帰的なフォトモザイクによるズーム処理．周囲の写真も変換することにより，拡大時にあたかもすべての写真がフォトモザイクであるように見せることが出来る．

[10] Nicholas Tran. Generating photomosaics: an empirical study. In *SAC '99: Proceedings of the 1999 ACM symposium on Applied computing*, pages 105–109, New York, NY, USA, 1999. ACM.