グラフィクスとCAD 66−10
(1993. 12. 16)

# テクスチャ・マッピングの一方法について

—フライト・シミュレータの模擬視界装置のためのパラレル・テクスチャ・マッピングについて—

梶 原 景 範
三菱プレシジョン株式会社
鎌倉市上町屋345
TEL 0467-46-7762 FAX 0467-46-3050

あらまし リアルタイムのコンピュータ・グラフィックス（CG）の最も進んだ一応用形態であるフライト・シミュレータの模擬視界装置（ビジュアル・システムという）においては高い画素処理能力を実現するために、一般に画素並列処理方式がとられる。従来は数十枚の基本的なパターンを混合してテクスチャ・マッピングのための模様を生成していたが、近年は具体的な実在の地域の地形地物を詳細に模擬するために、その地域の空中写真から作成したテクスチャ・マップを用いることが要求されるようになってきた。この要求を満たすためには従来方式では大容量のテクスチャ・マップ・メモリを各画素処理装置に重複して持つことがシステム構成上問題になってくる。本論文ではテクスチャ・マップ・メモリを共有しかつ画素を並列に処理するパラレル・テクスチャ・マッピングの方法を提案し、そのエミュレーション結果を示す。

和文キーワード ビジュアルシステム，リアルタイムコンピュータグラフィックス，テクスチャマッピング，MIPマップ

# A Method of Texture Mapping

—A Parallel Texture Mapping Technique for a Visual System of Flight Simulator—

Kagenori KAJIHARA
MITSUBISHI PRECISSION CO.,LTD
345 Kamimachiya, Kamakura, Kanagawa, Japan, 247

Abstract A visual system for flight simulator is one of the most advanced application of realtime computer graphics (CG) and realizes its high image generation capability by parallel processing of pixels. A conventional visual system generates desired texture pattern by mixing a few patterns of dozens of basic patterns stored in texture map memory. Recently, to simulate actual particular area precisely, visual systems are required to use geospecific textures made from the aerial photographcs of the tarfget areas. It is a big problem of cost that each pixel-processor provides large redundant texture map memory indivisually. This paper presents a method of mapping textures to multiple pixels from a common texture map memory in parallel and the resulting images by emulation.

英文 key words visual system, realtime computer graphics, texture mapping, Mipmap

# 1 はじめに

画素並列処理方式によるリアルタイムのコンピュータ・グラフィックス（ＣＧ）におけるテクスチャ・マッピングの一方法について述べる。具体的にはリアルタイムＣＧの最も進んだ一応用形態であるフライト・シミュレータ用の模擬視界装置（シミュレータの分野では一般にビジュアル・システムと呼ばれる）におけるパラレル方式のテクスチャ・マッピングについて述べる。

ビジュアル・システムでは限られた時間（1/30 sec 又は 1/60 sec）で、訓練に必要な多くの情報を含んだリアルな映像を発生するために、テクスチャ・マッピングが多用される。この傾向は軍用シミュレータにおいて特に強い。エアライン用ビジュアル・システムにおいても海面、田畑、市街地、滑走路等にそれぞれの質感を表すテクスチャをマッピングすることによって航空機の速度、高度、降下率等のキューをパイロットに効果的に与えることができる。軍用シミュレータにおいては、さらに木の間隠れに見える目標物や砂煙の向こうの戦車と云ったシーンが訓練上重要である。このような映像を実現するのに半透明物体、及び半透明テクスチャが用いられる。

テクスチャ・マッピングを行うために１画素当たりの計算量は多くなり、半透明物体、及び半透明テクスチャを実現するために１画素を多重（overwrite ratio という）に処理する必要が生じる。低空を飛ぶ戦闘機やヘリコプタ用のシーンをリアルに生成するには overwrite ratio は３～４程度必要であると云われている。このように高い処理性能を実現するために画素の並列処理が一般に行われている。

エアライン用シミュレータが離発着の訓練を主目的とするのに対して、軍用シミュレータは離発着に加えて模擬の対象となる航空機の担っている任務（例えば、空中戦、対地射爆撃、対潜水艦哨戒等）の遂行の訓練をする機能（例えば、レーダ測距機能、スレット・オカルト機能、赤外線等のセンサ映像の模擬機能等）が必要であり、ミッション・トレーニング用シミュレータと呼ばれている。軍用においては、最近それに加えてミッション・リハーサル用シミュレータというものが要求されるようになってきた。

これは、前者が比較的定形的なミッションの日常的な訓練を主目的とするのに対して、現実の突発的な特定のミッションの実際の予行演習を目的とするものである。このために、ミッション・リハーサル用ビジュアル・システムは、実在の特定地域の具体的な地形地物を識別できる程度のリアルな映像が必要であり、そのためデータベースを短時間で生成する機能及びそれをリアルタイムでレンダリングする性能が要求される。

具体的には、国土数値情報や空中写真等の既在のデータベースからビジュアル・システム用のデータベースを自動生成する。この場合、地表を表すテクスチャ（ＧＳＴ：Geospecific texture と云う）が必要になり、そのマップ数は当然大きなものになる（ミッション・トレーニング用の場合は前述のように典型的な模様を発生すればよいのでテクスチャ・マップは数十枚で十分であった）。

画素を単純に並列に処理する方式では、このような大量のＧＳＴを各画素処理装置にそれぞれ重複して格納し処理することになるので装置が大きくなる。この問題を解決するテクスチャ・マッピングの一方法を提案する

# 2 テクスチャ・マッピング

ビジュアル・システムのような動画像の生成においてはエイリアシングは現実感を著しく損なうので、テクスチャ・マッピングにおけるエイリアシングを除くために一般に Mip Map が用いられる。このときの処理の手順は次の通りである。

1. スクリーン座標系からテクスチャ・マップ座標系への座標変換

2. レベルの計算

3. テクセル・アドレスの計算

4. アンチタイル及びレベル補間

それぞれの処理の内容は式（1)、式（2)、式（3)、及び式（4）で与えられる。

視点を中心とし最大可視距離で前後左右を囲んだ領域をアクティブ・エリアという。次に、このアクティブ・エリアを覆うのに必要なＧＳＴのマップ数を計算する。高度$h$，ピッチ$\theta$で地点A$(x_a,0,0)$を見たときの地点Aのマップのレベルを考える（図1)。このとき地点Aをスクリーン上に投影した点及び地点Aの視点座標系における座標は式（5）で与えられる。地点Aのテクスチャ・マップのレベルは、式（2）に式（5）を適用して、式（6）で得られる。高度$h$　及びピッチ$\theta$　が変化するとき、$|\theta-\varepsilon|\leq\psi_{FOV}/2$ であるから、レベルの最大値は式（7）で与えられる。

視野角$\psi_{FOV}=45°$, 表示分解能 $n_{pxl}=1024$ pixel/scanline , マップの分解能を各レベル $256\times256$ texels/map とすると、レベル$L$が覆うべきエリアは $|x_a|\leq 2.83\times2^{-L}$ となる。従って、ＧＳＴマップをレベル8から0まで用意するものとすると、アクティブ・エリアを覆うのに必要なＧＳＴマップ数は 約 $(2\times3)^2\times9=324$ maps となる。

単純に画素を並列処理する方式においては、各画素処理装置は前述のマッピング計算回路、上記のＧＳＴマップメモリ及び汎用テクスチャ用のマップメモリを重複して持つことになる（図2)。

$$\boldsymbol{r}_m=T_{me}\boldsymbol{r}_s/z^{-1}+\boldsymbol{r}_{meye} \quad \cdots\cdots(1)$$

$\boldsymbol{r}_m$ ：マップ上の点 (単位　：1マップの幅)

$T_{me}=T_{mb}T_{bw}T_{ew}^{-1}$ ：座標変換マトリックス（視点 → マップ）

$\boldsymbol{r}_s=(x_s,y_s,k)^t$ ：スクリーン上の点（単位　：pixel)

$z^{-1}=\dfrac{(T_{eb}\boldsymbol{n}_b)^t\boldsymbol{r}_s}{k(\boldsymbol{n}_b^t\boldsymbol{r}_{beye}+d_b)}$ ：フェース上の点の視点座標系における奥行き

$\boldsymbol{n}_b,d_b$ ：フェースの法線ベクトル及び原点からの距離

$k=\dfrac{n_{pxl}}{2\tan(\psi_{FOV}/2)}$ ：$\psi_{FOV}$は水平視野角, $n_{pxl}$はスキャンライン当たりの pixel 数

$\boldsymbol{r}_{meye}=T_{mb}\{T_{bw}(\boldsymbol{r}_{eye}-\boldsymbol{r}_{wbo})-\boldsymbol{r}_{bto}\}$ ：マップ座標系における視点

サッフィックス $m,s,b,e,w$はそれぞれマップ，スクリーン，ボディ，視点，ワールド座標系を表す

$$l=-\log_2\{\max\{|\frac{\partial x_m}{\partial x_s}|,|\frac{\partial x_m}{\partial y_s}|,|\frac{\partial y_m}{\partial x_s}|,|\frac{\partial y_m}{\partial y_s}|\}\} \quad \cdots\cdots(2)$$

$$\frac{\partial\boldsymbol{r}_m}{\partial x_s}=T_{me}\boldsymbol{e}_1/z^{-1}-T_{me}\boldsymbol{r}_s\frac{\partial z^{-1}}{\partial x_s}/z^{-2}$$

$$\frac{\partial\boldsymbol{r}_m}{\partial y_s}=T_{me}\boldsymbol{e}_2/z^{-1}-T_{me}\boldsymbol{r}_s\frac{\partial z^{-1}}{\partial y_s}/z^{-2}$$

$$\left(\frac{\partial z^{-1}}{\partial x_s},\frac{\partial z^{-1}}{\partial y_s}\right)^t=\frac{T_{eb}\boldsymbol{n}_b z^{-1}}{(T_{eb}\boldsymbol{n}_b)^t\boldsymbol{r}_s}$$

$$\boldsymbol{a}_c = \lfloor 2^L \boldsymbol{r}_m \rfloor,\ \boldsymbol{a}_f = \lfloor 2^{L+1} \boldsymbol{r}_m \rfloor \quad \cdots\cdots(3)$$
$$L = \lfloor l \rfloor$$

$$t = (1-\gamma)t_c + \gamma t_f \quad \cdots\cdots(4)$$
$$\gamma = l - \lfloor l \rfloor$$
$$t_c = (1-\beta_c)\{(1-\alpha_c)t_L(\boldsymbol{a}_c) + \alpha_c t_L(\boldsymbol{a}_c + \boldsymbol{e}_1)\}$$
$$+ \beta_c\{(1-\alpha_c)t_L(\boldsymbol{a}_c + \boldsymbol{e}_2) + \alpha_c t_L(\boldsymbol{a}_c + \boldsymbol{e}_1 + \boldsymbol{e}_2)\}$$
$$t_f = (1-\beta_f)\{(1-\alpha_f)t_{L+1}(\boldsymbol{a}_c) + \alpha_f t_{L+1}(\boldsymbol{a}_f + \boldsymbol{e}_1)\}$$
$$+ \beta_f\{(1-\alpha_f)t_{L+1}(\boldsymbol{a}_f + \boldsymbol{e}_2) + \alpha_f t_{L+1}(\boldsymbol{a}_f + \boldsymbol{e}_1 + \boldsymbol{e}_2)\}$$
$$(\alpha_c, \beta_c)^t = 2^L \boldsymbol{r}_m - \lfloor 2^L \boldsymbol{r}_m \rfloor,\ \ (\alpha_f, \beta_f)^t = 2^{L+1} \boldsymbol{r}_m - \lfloor 2^{L+1} \boldsymbol{r}_m \rfloor$$

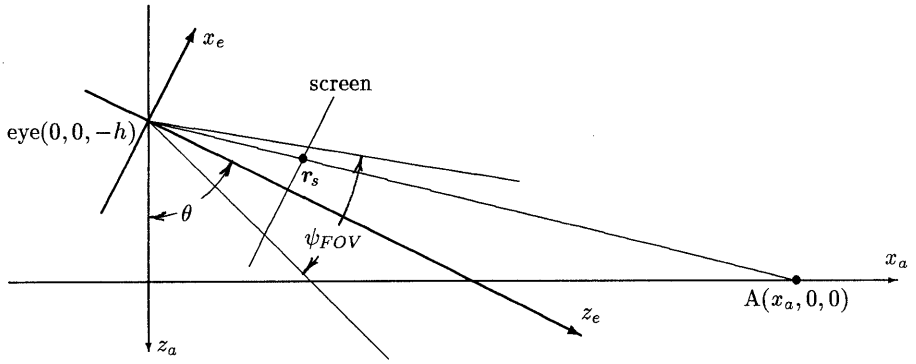



図 1: アクティブ・エリア

$$\boldsymbol{r}_s = kz^{-1}\boldsymbol{r}_e\ ,\ \ \boldsymbol{r}_e = (\cos\theta\, x_a - \sin\theta\, h,\ 0,\ \ sin\theta\, x_a + \cos\theta\, h)^t \quad \cdots\cdots(5)$$

$$l = -\log_2\left\{ \left| \frac{\partial x_m}{\partial x_s} \right| \right\} = \log_2\left\{ \frac{kh}{(x_a{}^2 + h^2)\cos^2(\theta - \varepsilon)} \right\} \quad \text{ただし } \varepsilon = \tan^{-1}\frac{x_a}{h} \quad \cdots\cdots(6)$$

$$l_{max} = \max\left\{ -\log_2\{ \left| \frac{\partial x_m}{\partial x_s} \right| \} \ :\ | \theta - \varepsilon | \leq \frac{\psi_{FOV}}{2} \right\} = \log_2 \frac{n_{pxl}}{2\sin\psi_{FOV}\, x_a} \quad \cdots\cdots(7)$$

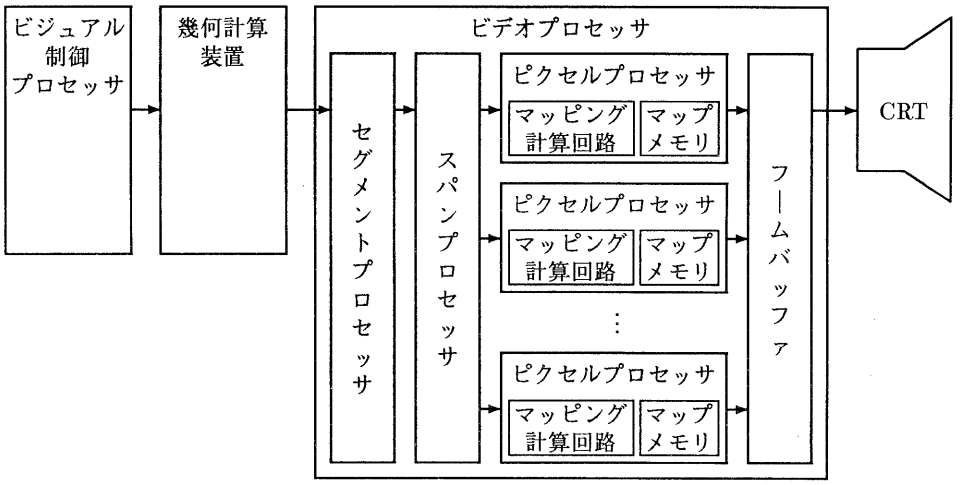



図 2: 従来方式のビジュアル・システムの構成

## 3 画素並列処理のためのテクスチャ・マッピング方法

並列に処理される画素の集合（例えば 2×2 pixels）を span と呼ぶことにする。共通のテクスチャ・マップ・メモリから span 内のピクセルに並列に同時にテクスチャをマッピングするパラレル・テクスチャ・マッピングについてのべる。ここで提案する方法は、span 内の画素をテクスチャ・マップ上に座標変換するとき、各画素のマップ上の像の位置は独立ではなく、その密集度はレベルにより決まり、また逆にレベルは画素の写像の密集度により決まるという性質に基づいている。

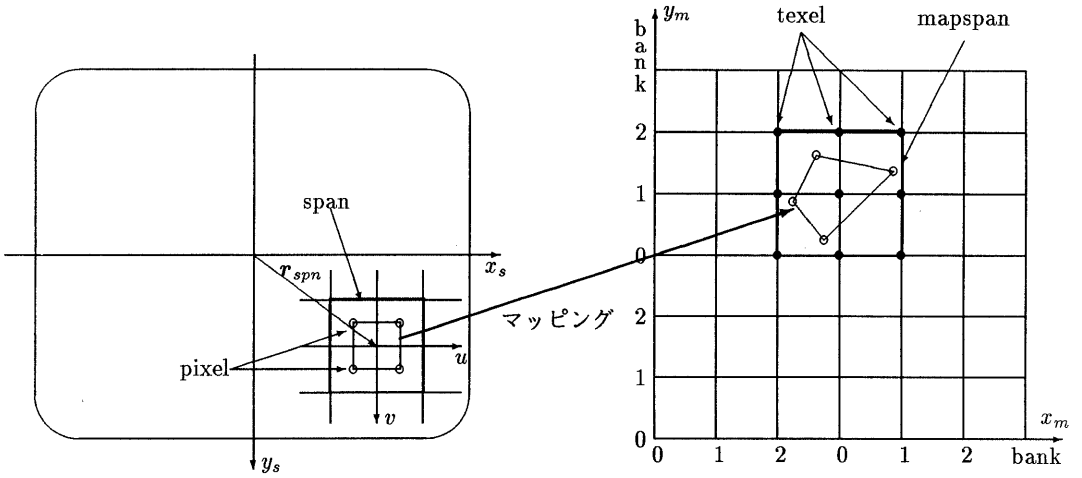
具体的には、テクスチャ・マップ上の texel の小集合（例えば $3 \times 3$ texels）を mspan(map span の意) と呼ぶことにすると、レベルを適当に選ぶと span は mspan 内に射影される。span 及び span 内の座標系を図 3(a) とし、レベルを式 9により計算すると span（正確にはその中の pixel) は、そのレベルのテクスチャ・マップの mspan 内に射影される（図 3)。

従って、テクスチャ・マップを $3 \times 3$ bank で構成することによって、各画素の処理を並列に同時に行うことができる（$2 \times 2$ pixels/span のとき)。このとき $(i, j)$ bank の読み出すべきアドレス $\boldsymbol{a}_{ij}$は式（10）で与えられる。

同時に読み出された $3 \times 3$ bank の中から、各画素のマップ上の像 $\boldsymbol{r}_m(u, v)$ を囲む $2 \times 2$ bank を選び前記の式（4）により各画素に対応したテクスチャ値を得ることができる。この方法に基づいたシステム構成を図 4に示す。このシステムのエミュレーションによる出力映像例を図 5に示す。この映像の背景は北海道のある地域の格子状の標高データから自動生成した地形モデルにその地域の空中写真から生成したＧＳＴを上記の方法でマッピングしたものである。航空機や樹木等の汎用テクスチャについても同様の方法を用いた。

## 4 終わりに

図 5に示すように、本方式で従来方式とほぼ同程度の詳細度の出力映像を得ることができた。一方処理の複雑さは、テクセル・アドレスの計算がやや複雑になり、$3 \times 3$ texel から $2 \times 2$ texel を選択する処理が必要であるが、テクスチャ・マップ・メモリは 4 分の 1 になり、メモリの bank 数はシ




(a) スクリーン　　(b) テクスチャ・マップ

図 3: span のマッピング

$$\boldsymbol{r}_m(u,v) = T_{me}\boldsymbol{r}_s(u,v)/z^{-1}(u,v) + \boldsymbol{r}_{meye} \quad \cdots\cdots(8)$$

$$\boldsymbol{r}_s(u,v) = \boldsymbol{r}_{spn} + (u,v)^t$$

$$z^{-1}(u,v) = z^{-1}_{spn} + \left(\frac{\partial z^{-1}}{\partial x_s}, \frac{\partial z^{-1}}{\partial y_s}\right)\begin{pmatrix} u \\ v \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \end{pmatrix} \in span = \left\{\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 0.5 \\ -0.5 \end{pmatrix}, \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix}, \begin{pmatrix} -0.5 \\ -0.5 \end{pmatrix}\right\}$$

$$l = -\log_2\{\max\{|\, x_{mmax} - x_{mmin}\,|, |\, y_{mmax} - y_{mmin}\,|\}\} \quad \cdots\cdots(9)$$

$$x_{mmax} = \max\{x_m(u,v) \ : (u,v)^t \in span\}$$

$$x_{mmin} = \min\{x_m(u,v) \ : (u,v)^t \in span\}$$

$$y_{mmax} = \max\{y_m(u,v) \ : (u,v)^t \in span\}$$

$$y_{mmin} = \min\{y_m(u,v) \ : (u,v)^t \in span\}$$

$$\boldsymbol{a}_{ijc} = \lfloor 2^L \boldsymbol{r}_{mspn}/3 \rfloor + \left(\delta(i, 2^L x_{mspn}),\ \delta(j, 2^L y_{mspn})\right)^t \quad \cdots\cdots(10)$$

$$\boldsymbol{a}_{ijf} = \lfloor 2^{L+1} \boldsymbol{r}_{mspn}/3 \rfloor + \left(\delta(i, 2^{L+1} x_{mspn}),\ \delta(j, 2^{L+1} y_{mspn})\right)^t$$

$$\boldsymbol{r}_{mspn} = (x_{mmin}, y_{mmin})^t$$

$$\delta(i,x) = \begin{cases} 1 & : \ i < \lfloor x \rfloor \ (\mathrm{mod}\ 3) \\ 0 & : \ other \end{cases}$$

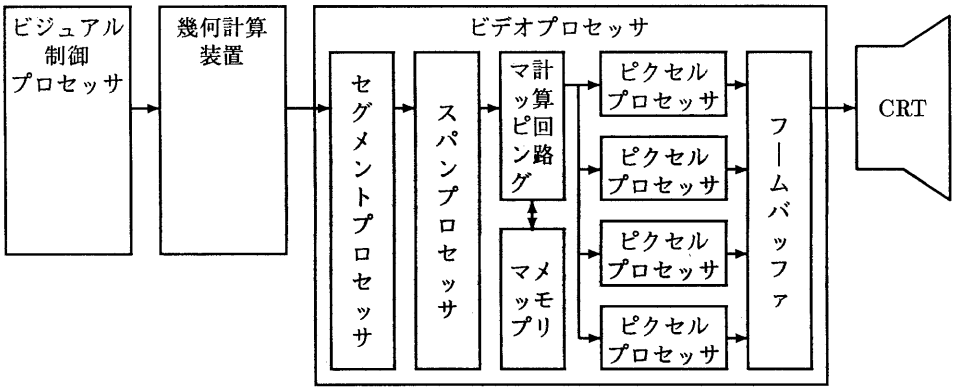



図 4: パラレル・テクスチャ・マッピングを用いたビジュアル・システムの構成

ステム全体で 4 × 4 から 3 ×3 bank になり、レベル計算は非常に簡単になる。また、各画素の座標変換において span の像からの変位分を計算すればよい部分があり回路の簡略化が行える。

厳密にいうと従来方式に比較して本方式ではいくらかぼける。この原因は従来方式に比べてレベルが低くなることによるものと思われる。これはメモリの bank を 3×3 にせず 4×4 にすることによって避けることができる。また、今回は確認はしなっかったが、画素のテクスチャ・マップ上の像の近傍の 3 × 3 texel が利用可能でありアンチタイルの処理として式（4）の線形補間より高次の方式を用いることが可能であることがわかる。これによって同一のメモリ容量でより忠実なテクスチャ・マッピングが実現できるものと思われる。

本研究を行うに当たり、当社画像機器事業部の藤野参事及び緒方主幹には有意義な討議及び助言をいただいた。また、エミュレーションのためのプログラム及びデータベースの作成は若い技術者伊藤、菊川、木村、越塚、船矢及び須原の各君の努力に全面的に負うものである。ここに謝意を表する。

## 参考文献

[1] B.J.Schachter : “Computer Image Generation” , John Wiley & Sons, New York(1983)

[2] 麻生，梶原：“航空機用シミュレータ”，シミュレーション第 9 巻第 4 号，pp259-270,（Feb. 1990）

[3] 梶原：“フライト・シミュレータにおけるコンピュータグラフィックス”，情報処理，第 29 巻第 10 号（1988）

[4] 梶原：“リアルタイム・レンダリング”，PIXEL No.92(1990)

[5] 梶原，藤野，緒方：“次世代ビジュアル・システムのアルゴリズム，アーキテクチャ”，三菱プレシジョン技報　Vol.1 pp70～ 77(May. 1992)

[6] 梶原，他：“テクスチャ・マッピングの方法及びテクスチャ・マッピング装置”，特願平　5－１１９２７５

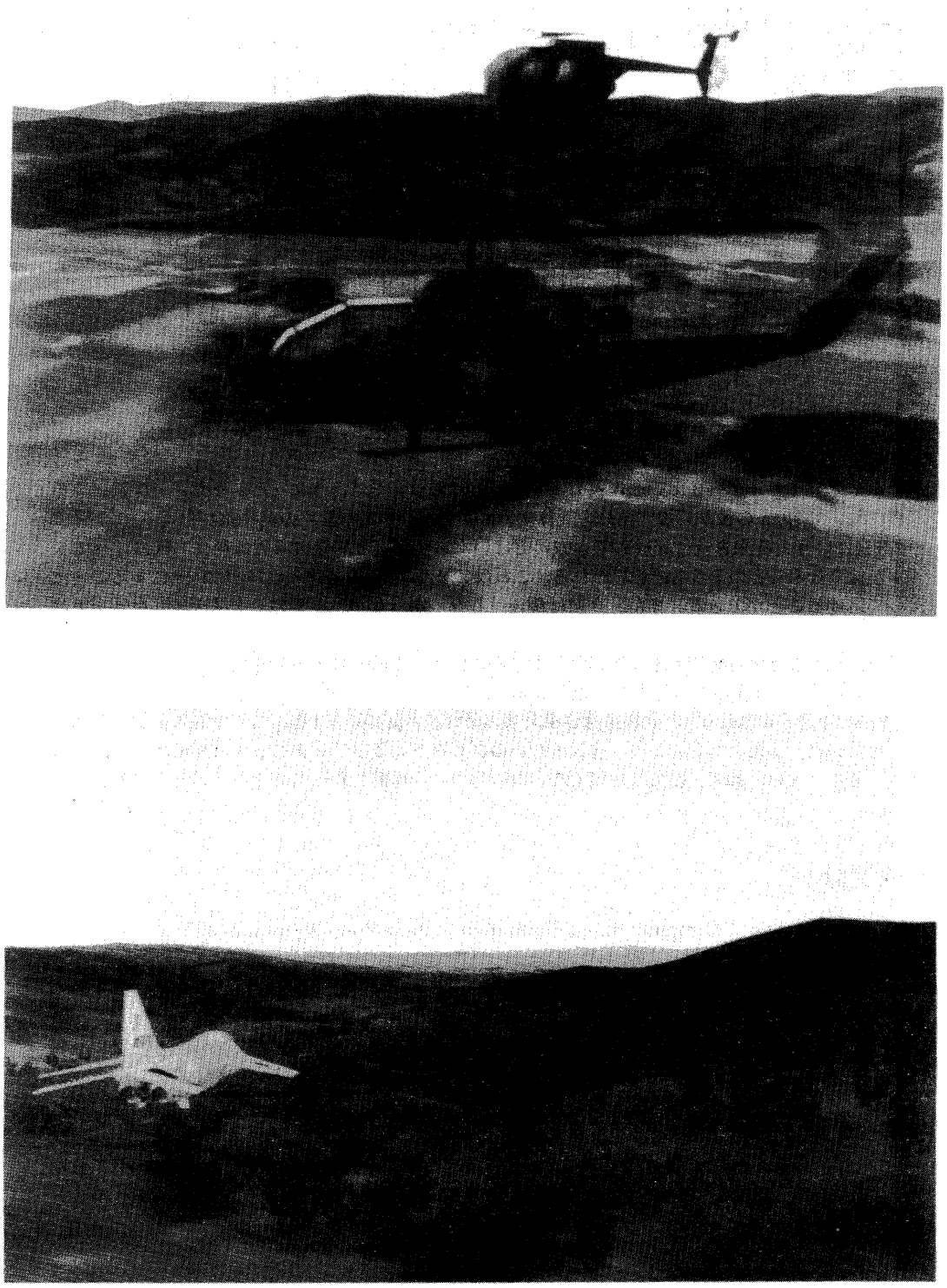


図 5: パラレル・テクスチャ・マッピングを用いた出力映像例