Vol.2011-HCI-142 No.15
2011/3/18

３Ｄディスプレイを用いたリアルタイムな局所立体視手法の提案とその試作

持田光将† 深井佑希† 伊藤永悟‡ 藤田光治‡
佐野勇司†,‡ 藤本貴之†,††

近年,3D 映像を上映するための様々な装置が存在する．現在,それら３D 装置の多くは,画面全体を立体化するためのシステムとなっている．
そのため,ディスプレイの一部だけを立体化するといったことの実現は難しい．
そこで,本論文では,特定のウィンドウを３D 化させる手法を提案する．本研究で提案する手法に基づき,3D ディスプレイと Windows の API を用いたプロトタイプシステムを構築した．

A proposal of Real-time solidification technique to use 3D display and construction of prototype system

Mitsumasa MOCHIDA† Yuki FUKAI† Eigo ITO‡ Koji FUJITA‡ Yuji SANO†,‡ Takayuki FUJIMOTO†,††

Recently, various devices to screen 3D image exist. Those 3D devices can make the entire screen 3D. However, it is difficult to make only a part of the display 3D.
In this paper, we proposed the technique to make one specific window 3D image. In this proposal, we constructed the prototype system that used API of 3D display and Windows.

1. 研究の概要

近年,三次元表示を可能とする３D ディスプレイが急速に普及している.極めて安価に３D コンテンツを楽しむ環境が整備されており,三次元表示の出力方法は,今日までに,実際に製品化されたものだけでも,実にさまざまな方法が開発されている．考案のみに留まっているものや実験段階レベルのものまで含めるとその数は定かではない.

３D ディスプレイの普及の試みは過去にも何度かあったが,すべて失敗に終わっているといえる．その原因の一つとして３D コンテンツの少なさが指摘されている.

３Dディスプレイを普及させるためには,３Dコンテンツの多種化とその装置の普及が並行して行わなければならないが,コンテンツ制作者にとっては制作した３D コンテンツの表示装置が必要であり,一方表示装置の開発者にしてみると表示させる３D コンテンツが必要となる.

これまでの傾向としては,表示装置が先行して開発されているために,コンテンツ制作者はある特定の条件のもとで制作を行わなければならない．そのため三次元表示技術の開発が必要である[１].

そのような３D ディスプレイ業界の動向および開発の現状に加え,現在の技術では,三次元表示を行なうコンテンツが原則として,「画面全体」であるという表現技法的な課題も指摘できる．局部的な箇所だけを「立体化」する場合は,事前にコンテンツ／ソフト自体がそうなるように加工しておく必要がある.すなわち,リアルタイムな映像で局部的な立体化を実現することは難しい.

そこで本研究では,３D ディスプレイを用いて,リアルタイムに画面全体ではなく,局区部的な箇所のみを立体視する技法およびシステムを提案し,試作した.

2. ３Ｄディスプレイの現在的課題

一般に３D ディスプレイを用いて立体視用を行なう場合,ディスプレイに映し出されたコンテンツは,そのまま全てが立体化する（図１）．例えば,３Ｄディスプレイの画面全体を立体視することは出来ても,特定の一部だけを立体にする製品あるいはサービスはしない.

現在,ディスプレイ内の一部だけを局部的に立体化するといったことは困難である.もちろん,画面の一部や必要箇所のみを立体化するように事前に３D 用映像として制

†東洋大学大学院工学研究科
Graduate School of Eng., Toyo University
‡東洋大学工学部
School of Eng., Toyo University
††東洋大学総合情報学部
Dep. of Information Science and Arts, Toyo University

作されたコンテンツであれば可能であるが,リアルタイムな映像や作業の中で,それを実現させる事は出来ない.

例えば,現在作業中のウィンドウやソフトウェアの部分だけを立体視さえ,その他を通常のフラットな画面状態にさせておく,といった３D 表示の運用を実現させるシステムや商品は著者らが知る限り存在していない.

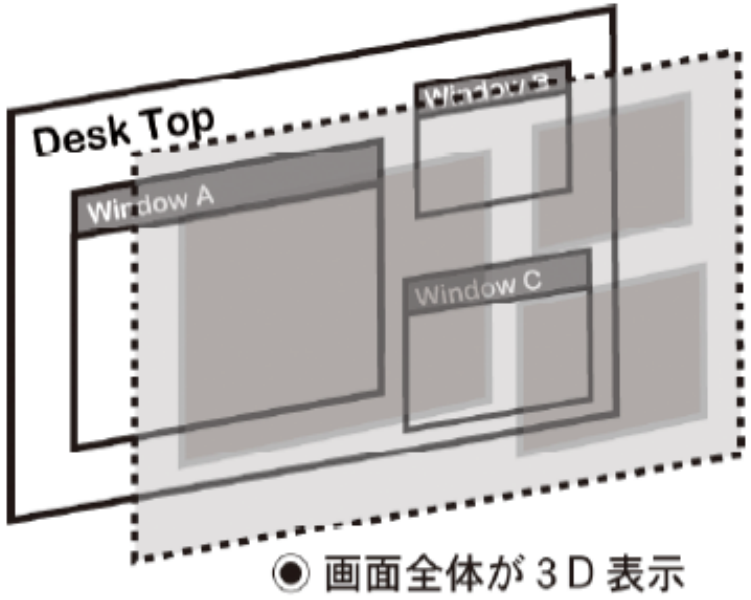

図 1 一般的な３D ディスプレイでの立体視

本研究で提案・試作したシステムでは,３Dディスプレイを用いてコンピュータ操作を行なう場合に,特定の作業ウィンドウ（あるいはソフトウェア）のみを立体視させることが可能である（図２）.

これにより,デスクトップ上に複数開いたウィンドウ（あるいはソフトウェア）のうち,映像のみを３D 表示にし,ワープロソフトやブラウザは通常のフラットな常時のままにしておくなど,３D 表示を計算機操作の効率化や表現性の拡張に利用することができる.

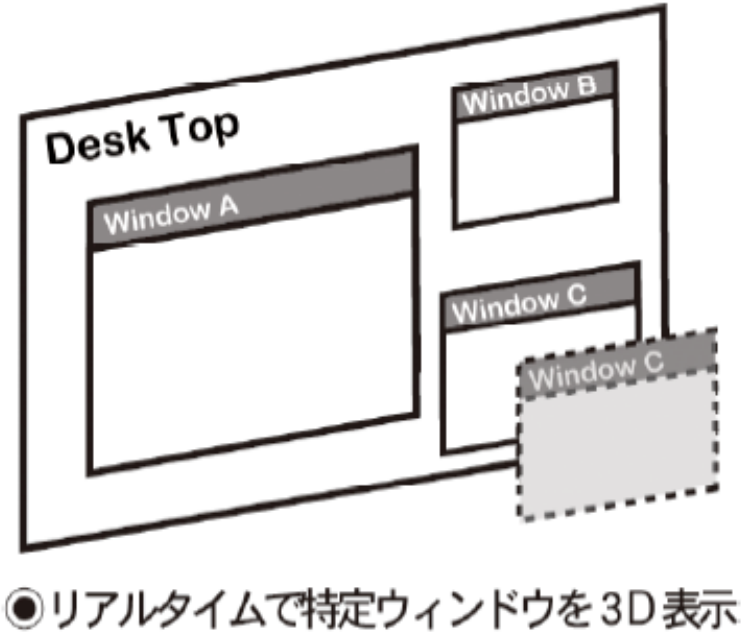

図 2 本研究で提案・実装する３D 表示

3. 立体視表示

３D ディスプレイを用いた立体映像の生成と取得には,一般に当該コンテンツを左右にズラして表示し,それをディスプレイと同期させた専用眼鏡で閲覧することで実現させる.

立体映像を生成するために必要となる元コンテンツの左右のズレに関しては,一般

に,快適な立体視を可能とするには,左右の映像のズレ量を２.３６ ㎝以下に,不快でない立体視を可能とするためのズレ量は４.４５cm 以下であるとされている（手前への立体視の場合）．また,３D ディスプレイと立体視用眼鏡の同期に関しては,現在市販されている一般的な３D システムの同期周波数は１２０Ｈｚである．

左右を１２０Hz で切り替えるため,片方の映像では６０Hz となる．すなわち,ディスプレイと眼鏡の同期間隔,左右映像それぞれ１ 秒間に６０ 回を交互に表示させることとなる．

よって本研究で提案・試作するシステムでは,Windows OS のデスクトップ上で立体視をしたい当該オブジェクトを左右２.３６cm 未満の最適な距離でズラして表示させつつ,その左右の表示を１秒間に６０ 回交互での表示とすることで,立体表示を実現する．

4. リアルタイムな２ つの映像の生成について

アクティブとなっているウィンドウ（あるいはソフトウェア）を２.３６cm 未満の適切距離にダブらせて表示させ,それを３D ディスプレイ内で専用眼鏡と同期させた状態で閲覧することで,当該ウィンドウ（あるいはソフトウェア）をリアルタイムに立体視させる．

アクティブウィンドウがズレた２つの映像のリアルタイムな生成し，その２つの映像を結合することで立体視可能な映像を生成する．そのためのフローを以下のステップで示す．

〔ステップ１〕デスクトップの設定

マルチディスプレイで複数のデスクトップを図３のように作成．

ここでは,デスクトップ A と B は全く同じ解像度・壁紙とし,アイコンなどは付加的なコンテンツは存在しないと仮定する．

デスクトップ C を通常の作業領域用のデスクトップ(アイコンやタスクバーなどがある)とする．

そのため，作業用領域を必要とせず，デスクトップ A と B のみで全く同じ解像度・壁紙とし,アイコンなどは付加的なコンテンツは存在しないを作り出せる場合は，デスクトップ C は不要．

図 3 デスクトップの設定

〔ステップ２〕アクティブウィンドウの表示１

アクティブウィンドウをデスクトップ A に表示（図４）．

アクティブウィンドウの位置はデスクトップ A の任意の位置とする．ただし，次のフローでデスクトップ B にデスクトップ A にあった位置より２.３６cm 以内程度ズラして表示アクティブウィンドウを表示するため，その移動によってデスクトップ外にアクティブウィンドウが表示されないように，デスクトップ A の画面端から適切な距離を離れた位置とする．

図 4 アクティブウィンドウの表示１

〔ステップ３〕アクティブウィンドウの表示２

アクティブウィンドウをデスクトップ B にデスクトップ A にあった位置よりも２.３６cm 以内程度ズラして表示（図５）.

図 5 アクティブウィンドウの表示２

〔ステップ４〕アクティブウィンドウの表示３

アクティブウィンドウをデスクトップ A に表示した後,デスクトップ B に２.３６cm 以下程度ズラして表示.

表示する位置は(２)のフローで表示した位置と同じとする.

〔ステップ５〕ズレた２つの映像の生成

上記２～４ のフローを交互に高速で繰り返して,アクティブウィンドウがズレた２つの映像を生成する（図６）.

〔ステップ６〕立体視可能な映像の生成

(５)のフローで生成された２つの映像を１つに合わせて，立体視可能な映像を生成する（図７） .

この生成した映像を３D ディスプレイで表示し，専用眼鏡で見ることで，アクティブウィンドウのみを立体視することができる.

図 6 ズレた２つの映像の生成

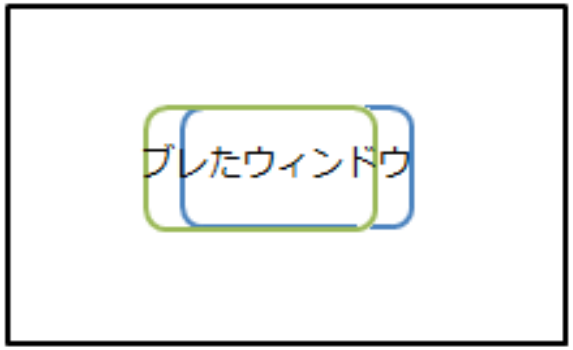

図 7 立体視可能な映像の生成

5. アクティブウィンドウの操作について

本研究で提案する立体視の手法では,同一のオブジェクトを左右にズラした形で表示させることで,その当該オブジェクト（今回の試作ではウィンドウ）を３D 表示させる.

同一ウィンドウを左右にズラして表示させるためには,アクティブになっているウ

ィンドウの位置情報の取得し,視覚的に適正な速度で高速移動させることで,実現させる.

そこで本研究では,その位置情報の取得と反映にはWindows のAPI を用いて操作しているため,特別な器具や環境あるいはシステムなどは用いる必要はなく,一般的なWindows コンピュータがあれば,容易に実現させることができる.

具体的には,C 言語を用いuser３２.dll の関数を利用する.

以下に本研究で用いた具体的な手順を示す.

〔手順１〕ハンドル取得

アクティブウィンドウの取得のために，GetForegroundWindow 関数を用いる.

GetForegroundWindow 関数はフォアグラウンドウィンドウ，すなわちユーザが操作しているウィンドウの中のアクティブウィンドウのハンドルを返す関数である.

GetForegroundWindow 関数を用いて,アクティブウィンドウのハンドルを取得する.

〔手順２〕アクティブウィンドウの移動１

アクティブウィンドウの移動にはSetWindowPos 関数を用いる.

SetWindowPos 関数は指定したウィンドウののサイズ，位置,およびZ オーダーを変更する関数である．SetWindowPos 関数を用いて，GetForegroundWindow 関数取得したハンドラによって移動させたいアクティブウィンドウを指定し，ディスプレイ A の任意の位置を移動先に指定してウィンドウを移動させる．また，アクティブウィンドウのサイズは変更しないため，パラメータSWP_NOSIZE を指定し，Zオーダーは先頭にするため，パラメータはHWND_TOP を指定する.

〔手順３〕同期処理１

同期処理を行うため，Sleep 関数とGetLocalTime 関数を用いる.

GetLocalTime 関数は現在のシステム時間を取得する関数である.この関数は現在の年，月，日，曜日，時，分，秒，ミリ秒を所得することができるが，今回はミリ秒の値のみを用いる.

Sleep 関数は指定した時間だけ処理を中断する関数であり，ミリ秒単位で指定できる.

GetLocalTime 関数でアクティブウィンドウの移動前と移動後の時間を取得し,その経過時間に応じて，同期処理に必要な待ち処理時間を求め，Sleep 関数でその時間を指定し，待ちの処理を行い,ディスプレイ同期させる.

〔手順４〕アクティブウィンドウの移動２

〔手順２〕で行なった操作と同様に，再度GetForegroundWindow 関数でハンドラを取得し,SetWindowPos 関数にそのハンドラと移動させるディスプレイB の位置設定し,ウィンドウを移動させる.

SetWindowPos 関数におけるその他のウィンドウののサイズやZ オーダーといったパラメータは〔手順２〕の指定と同様である.

〔手順５〕同期処理２

〔手順３〕の操作と同様に,〔手順３〕で取得した時間とGetLocalTime 関数を用いて,〔手順４〕の操作で行ったアクティブウィンドウの移動後の時間を取得し,その経過時間に応じて，同期処理に必要な待ち処理時間を求め，Sleep 関数でその時間を指定し，待ちの処理を行い,ディスプレイ同期させる.

〔手順６〕繰り返し

以上の〔手順１〕から〔手順５〕までを繰り返すことでそれぞれのディスプレイの位置に同じウィンドウが表示されている状態となる.

これによって，アクティブウィンドウがズレた２ つの映像を生成可能となる.

以上のように,Windows OS の一般的な環境を用いた実行が可能な手法およびシステムとなっている．システムの起動・終了も一般的なC 言語でのプログラムファイルの操作を同一である.

6. 試用実験と考察

本章では,実装したシステムを用いて行なった試用実験　について述べる.

まず,三次元表示や立体視に関する評価には,必ずしも厳密な指標があるわけではない．また,閲覧者／視聴者の主観や感じ方,あるいは「慣れ」なども立体表示の状態に大きく影響を受けるため客観的な基準を明示することが難しいためである.

例えば同一の３D 映像コンテンツやディスプレイ・システムなどを利用した場合でも,その「立体視のなり方」は一定ではなく,「ものすごく立体に見えた」という感じる人と,「単に映像がチラついていただけ」と感じる人が併存してしまうパターンは少なくない．また,仮に「立体に見えて」としても,それを不快に感じる人も多い.

そのため,本試用実験では,あくまでも,特別な知識や経験を有さない一般的な視聴者２０名を対象に行なった.

実験では,ディスプレイとの同期タイミングなどを考慮した結果,ウィンドウのズレ幅は２ドットから４ドット,左右間の高速移動の待ち時間は８ミリ秒で行なった.

試用実験に用いた環境を以下の表１に示す.

・プログラム実行用計算機：acer 社製ノート型計算機 ASPIRE AS３８１０T-H２２（Windows OS/Core２ Duo SU９４００ １.４GHz/３MB）
・搭載メモリ：DDR３ PC３-８５００ ２GB
・３D ディスプレイ：ZALMAN 社 ZM-M２１５W（２１.５inch／１６:９ ワイド）

表１．試用実験での利用環境

以上である．これらはいずれも一般的に入手可能な極めて安価な民生機である.

実験は極めてシンプルに,通常のディスプレイ状態とシステム起動時（特定ウィンドウの３D 表示時）の両方を見比べてもらうという方法で行なった.

その結果を以下の表２に示す.

完全に立体化している	立体化している	どちらとも言えない	ほとんど立体化していない	全く立体化していない
３人（１５%）	８人（４０%）	４人（２０%）	３人（１５%）	２人（１０%）
ポジティブ意見 １１人（５５%）			ネガティブ意見 ５人（２５%）	

表２．試用実験の結果（被験者総数２０人）

結果から,本研究で提案した手法／システムを利用した立体視に対して,ネガティブな意見は,「ほとんど立体化していない／全く立体化していない」を合わせても,２０人中５人（２５%）にとどまり,逆に,「完全に立体化している／立体化している」というポジティブな意見は,１１人（５５%）と過半数を超えた.「どちらとも言えない（４人（２０%））」をネガティブ意見に入れたとしても,概ね本提案システムでは,特定ウィンドウのみの立体視を実現することができたと考えられる.

本実験では,一般的な民生用のノート型計算機を利用したため,本システムを利用する上での十分な性能が確保できているわけない．よって,マシンスペック的な問題から,画像のチラつきや不自然感を完全に払拭できていなかった．しかしながら,計算機のスペックを挙げ,システム利用に十分な環境を確保でき,自然なプログラム実行をすることができれば,ポジティブな意見は拡大すると考えられる.

7. 今後の課題

試用実験から,本研究で提案した手法およびシステムが,概ね３D ディスプレイ上での局部的な立体視を実現させることができた.

よって,理論的／手法的な妥当性は概ねあると考えられる．しかしながら,試用実験を通して,実用化をめざす上での今後の課題や検討事項が明らかになった．本章では,本提案・システムを実用レベルへと向上させるための今後の課題および改善・検討事項について述べる.

まず第一に本試作システムの利用環境の問題である．実験に利用した試作システムでの三次元表示とは,現状では,アクティブなオブジェクトを高速での短距離移動をループさせることで擬似的にズレた状態を生成している．そのため,マシンスペックや利用環境によっては,それぞれのウィンドウにチラつきが発生する場合があり,十分な三次元表示を実現できないばかりか,不愉快感だけが増幅する危険性を有する．特殊な性能は必要ではないが,一般的な民生機であっても,少しでも処理性能や表示性能の高い計算機の利用を推奨する必要がある.

また,実験では,単独のウィンドウおみを用いての評価のみを行なったが,複数のウィンドウを利用する場合にスムーズな利用する場合の処理にも課題がある．今後の課題・検討事項としたい.

Vol.2011-HCI-142 No.15
2011/3/18

参考文献

[１] 羽倉弘之“３D 映像技術と事業化の先端動向”映像情報メディア学会誌技術報告, 33(4), 11-16, 2009-01-22, 2099
[２]狩野育史, “運動視差による立体映像:残像 式立体映像”, 三次元映像のフォーラム, 2006
[３]西川善司 , “各立体方式の基本技術とその課題,業界の流れを理解する 立体視対応ディス プレイのしくみと技術的問題”,インターフェース 37(1), 44-55, 2011-09-01, CQ 出版社