「マルチメディア，分散，協調とモバイル
(DICOMO2021)シンポジウム」 令和3年6月-7月

ユーザ視線検出を用いた効率的な動的映像デコード手法

河野 翔太[1]　中村 朋生[1]　門本 淳一郎[1]　入江 英嗣[1]　坂井 修一[1]

概要：情報化の進展により，身の回りに多くの計算機が存在する社会がもたらされたが，それらの中には計算資源あるいは使用可能な電力が潤沢でないエッジデバイスも多い．また，人間中心の情報化社会が成立するにつれて，結果の正確性よりも，リアルタイムでの応答性に比重が置かれた計算の需要が増加している．ここで，時間とともに変化する人間の状態に応じて，求められる正確性は変化する．この背景から，従来よりも積極的な近似計算を行いつつ，ユーザの状態を利用して近似の積極性を動的制御することで，計算の処理効率を向上させることができる．本研究では，前述した状況を表す例として，透過型ヘッドマウントディスプレイにおいて映像を再生する状況を取り上げる．そして，ユーザの視線情報のフィードバックを受けつつその周囲のみをデコードする近似デコード手法を提案する．提案手法を実装し，実際にデコードされる領域の割合を調査する実験を行った結果，デコードされる面積は 56.7% まで削減されることが分かった．また，近似計算の積極性を評価する実験を行い，提案手法の近似計算の積極性はある時点で過剰になるものの，ユーザの満足度を維持しつつ効率的な映像のデコードを行うことができることが分かった．

Efficient Dynamic Video Decoding Using User Gaze Detection

SHOUTA KOUNO[1]　TOMOKI NAKAMURA[1]　JUNICHIRO KADOMOTO[1]　HIDETSUGU IRIE[1]
SHUICHI SAKAI[1]

1. はじめに

近年，情報化が進展するにつれて，計算需要が高まり，それとともに我々の身の回りに存在する計算機の数は増加した．それらの中には，ウェアラブルデバイスやスマート家電といった，使用可能な計算資源あるいは電力が潤沢でないエッジデバイスの数も少なくない．また，情報化社会の中心は計算機から人間へと変化している．これにより，計算品質を追求することよりも，人間の待ち時間を少なくするためにリアルタイム性に比重が置かれた計算の需要が増加している．加えて，人間の状態は時々刻々と変化するため，要求される正確性もそれに応じて変化する．この背景から，以下の 2 つの技術を用いることで，より効率的な計算を達成する．1 つめは，積極的な近似計算である．計算の出力が正確さに欠けるとしても，その出力対象が人間である場合，人間の認識が完璧でないことにより，多少の不正確さは許容される．この許容される不正確さの部分を最大限に生かし，従来よりも積極的な近似計算を行う．2 つめは，ユーザの状態をフィードバックした，近似計算の積極性の動的制御である．ユーザにとって計算結果が持つ意味は必ずしも一定ではなく，ユーザあるいはユーザ周囲の状態に応じて時間とともに変化する．したがって，近似計算の積極性が常に一定である場合，ある時点ではユーザが計算品質に不満を覚え，またある時点では計算品質を保つためにシステムの用いた計算資源が無駄になる．これらのことから，従来よりも積極的な近似計算を行いつつ，ユーザの状態をフィードバックしてその積極性を動的制御するシステムにより，ユーザの満足度を維持しつつより効率的な計算が実現できる．

本研究では，ユーザの視線をフィードバックしてその周囲のみをデコードすることにより，近似の積極性を動的制御するデコード手法を提案する．提案手法の実装に際して，ユーザの状態をフィードバックできるデバイスとしては透過型ヘッドマウントディスプレイを，動作させるアプリケーションとしては映像再生アプリケーションを選択した．これは，この組み合わせがユーザに返すための出力をリアルタイムに計算し，かつシステムがユーザの情報を利

[1] 東京大学大学院情報理工学系研究科

用することができるデバイス及びシステムとなることによる．透過型ヘッドマウントディスプレイでは，ユーザは周囲の状況を知覚することができるため，周囲の状況によるユーザの状態の変化が生じる．映像は時間的な広がりを持つため，単なる時間経過によってもユーザの状態は変化する．そして，その変化するユーザの状態を，ヘッドマウントディスプレイが検出するユーザの視線情報として映像デコーダへとフィードバックする．

提案手法をアプリケーションとして実装し，ユーザが使用した場合に実際にデコードされる領域の割合を調査する実験を行った．その結果，デコードされる面積は 56.7% まで削減されることが分かった．また，提案手法の近似計算の積極性を評価する実験により，提案手法の近似計算の積極性はある時点で過剰になる場合があることが分かった．しかし，同時に，動的な近似計算により，ユーザの満足度を維持しつつ効率的な映像デコードを行うことができることが分かった．

以降，第 2 章では本論文の背景技術を，第 3 章では提案手法を，第 4 章では提案手法の実装を，第 5 章では実装を用いた実験を，第 6 章では関連研究を，そして第 7 章では本論文のまとめを述べる．

2. 背景技術

映像の近似デコードを行うために，その基盤となるエンコード方式について説明する．この節では，特に MPEG-1 映像エンコード形式 (ISO/IEC 11172-2) について，以下の 3 つを説明する．1 つめは，MPEG-1 映像におけるフレームの形式についてである．2 つめは，MPEG-1 映像におけるフレーム同士の依存関係についてである．3 つめは，MPEG-1 映像ビットストリームの階層構造についてである．

2.1 MPEG-1 映像におけるフレーム

MPEG-1 映像は，フレーム間の時間的な冗長性を効果的に排除する動き補償フレーム間予測を用いてエンコードされるが，この技術を用いてエンコードされたフレームは，デコードする際に前後のフレームの情報が必要となる．このため，映像中のほぼ全てのフレームに対してこのエンコードを適用すると，映像のランダムアクセス性に乏しくなる．これは，映像のシーク及び途中からの再生を行う度に比較的大きな計算コストが必要となることを意味する [3]．これを回避するために，MPEG-1 映像エンコード形式におけるフレームには，異なるエンコード方法を持つ 3 種類のフレームが存在する．以下でそれら 3 種類のフレームについての詳細を述べる．

2.1.1 I フレーム

3 種類のフレームの中で，唯一他のフレームを利用しないエンコード方法によりエンコードされるものが I フレームである．I フレームは，他のフレームを利用した動き補償フレーム間予測によるエンコードがなされないことから，3 種類のフレームの中では最も圧縮効率が低い．しかし，他のフレームの情報を利用する必要がないため，それ単体でデコードが可能である．

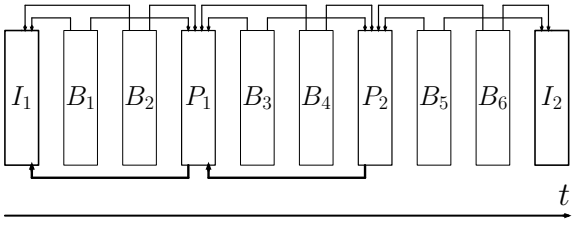


図 1: MPEG-1 映像におけるフレームの依存関係
[3, p.52] を参考に作成

2.1.2 P フレーム

P フレームは，I フレームのエンコード方法に加え，直前の I フレームあるいは P フレームの情報を用いた動き補償フレーム間予測によるエンコード方法を選択できるフレームである．エンコーダはそれぞれのマクロブロックについて，2 種類のエンコード方法から一方を選択して適用し，エンコードを行う．ここで，MPEG-1 映像におけるマクロブロックとは，フレーム中のエンコードの単位となる縦横それぞれ 16 ピクセルの領域である．このエンコード方式でエンコードされた領域をデコードするには，直前の I フレームあるいは P フレームの参照される領域がデコードされている必要がある．

2.1.3 B フレーム

B フレームは，I フレームのエンコード方法に加え，直前及び直後の I フレームあるいは P フレームの情報を用いた動き補償フレーム間予測によるエンコード方法を選択できるフレームである．それぞれのマクロブロックについて，過去のフレームのみ，未来のフレームのみ，あるいはその両方を用いた動き補償フレーム間予測によるエンコードに加え，I フレームと同様のエンコードの中から 1 つを選択してエンコードを行う [3]．時間的に後のフレームの情報を用いたエンコードがなされていることにより，B フレームをデコードする際にはまず直後の I フレームあるいは P フレームをデコードし，その情報に加えて直前の I フレームあるいは P フレームの情報を用いてデコードを行う必要がある．

2.2 フレームの依存関係

図 1に，典型的な MPEG-1 映像エンコード形式の映像のフレーム列を示す．図中で，それぞれのフレームに記すアルファベットはそのフレームの種類を，フレームから出る矢印はそのフレームがどのフレームに依存するかを表し，

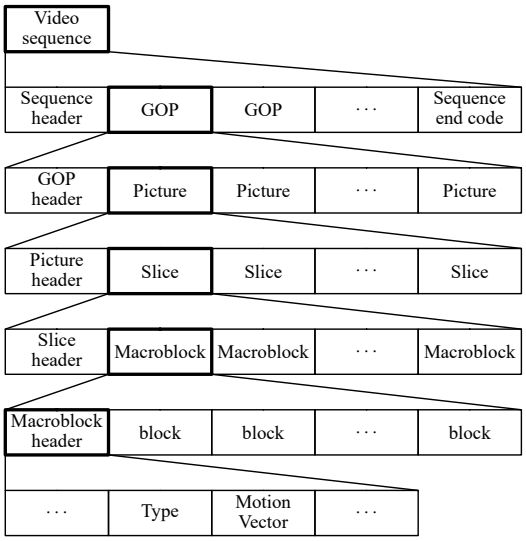


図 2: MPEG-1 映像ビットストリームの階層構造
[4, Fig.11.5] 及び [5] を参考に作成

右にあるフレームが時間的に後のフレームとなる．この映像を再生する際には，フレームの依存関係を満たす必要があるため，左のフレームから順序通りにデコードすることはできない．B フレームをデコードする際に先読みが必要となり，$I_1, P_1, B_1, B_2, P_2, B_3, B_4, I_2, B_5, B_6$ の順にデコードされる．

2.3 MPEG-1 映像ビットストリームの階層構造

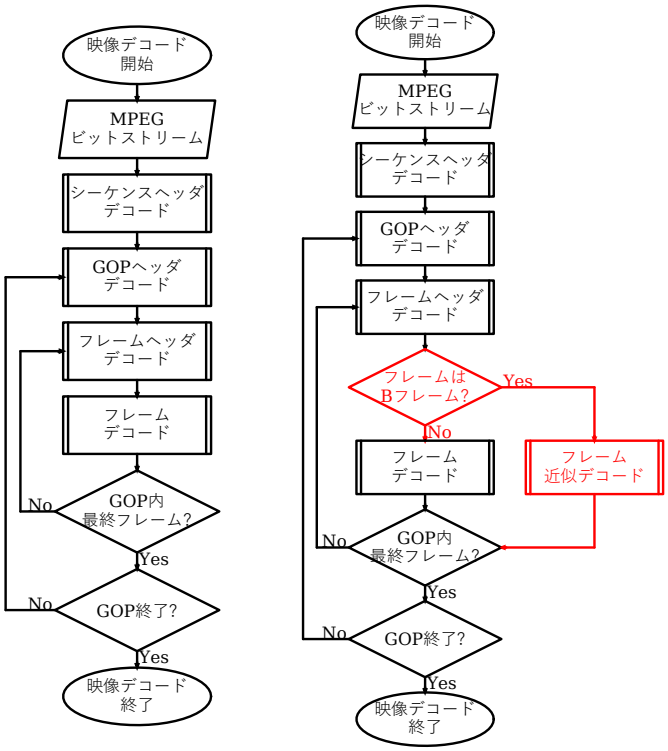
図 2に，MPEG-1 映像ビットストリームの階層構造を示す [4,5]．本論文中では一貫して「フレーム」と表現したが，MPEG-1 映像エンコードの仕様に従い，図 2では Picture として示す．図中の GOP は Group Of Pictures を意味し，これは 1 つ以上の I フレームを含むフレームの列である．Slice は画像中で同じ行にあるマクロブロックの集まりである [6]．2.1で述べた通り，P フレーム及び B フレーム中に含まれるマクロブロックは，エンコード方法に選択肢が存在する．そのため，マクロブロックヘッダにはエンコード方法を示す Type フィールドが含まれる．したがって，MPEG-1 デコーダは，マクロブロックヘッダを確認した後に適切な方法でデコードを行う．

3. 提案手法

3.1 提案手法の概要

本研究では，デコードしようとするフレームの種類に応じて，通常のデコードパスと近似デコードパスとを使い分ける MPEG-1 映像デコーダを提案する．近似デコードパス中では，ユーザの視線情報を利用して映像中の一部の領域のデコードを省略する．ここで，近似デコードパスによる処理の対象は，MPEG-1 映像中に存在するフレームのうちの B フレームのみとする．これは，B フレームは他のフレームによって参照されない [3] ため，近似デコードの影響が他のフレームに伝播しないからである．

3.2 デコードパスの切り替え



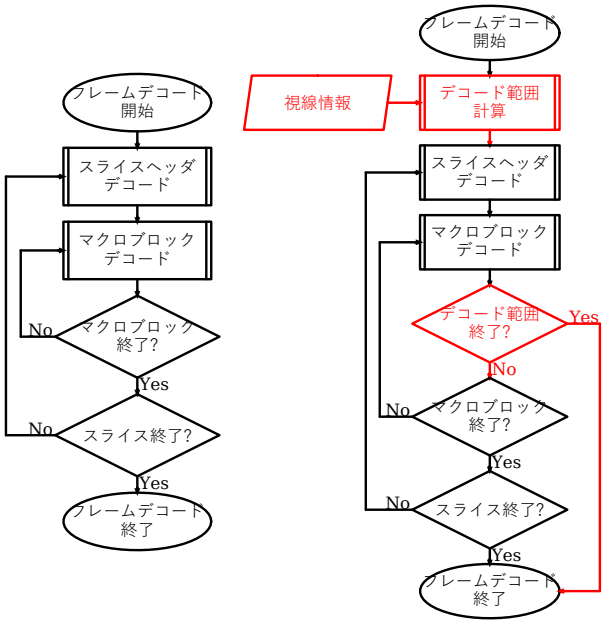
(a) 通常のデコーダ　　(b) 提案手法におけるデコーダ

図 3: ビットストリームのデコード処理のフロー

図3(a)に，MPEG-1 映像のデコード処理の通常のフローを示す．図 2に示す MPEG-1 映像ビットストリーム中の階層構造に従い，シーケンス，GOP，フレームの順にデコードを行う．本研究では，図3(b)に示す通り，フレームを処理する直前にフレームの種類による分岐を挿入する．分岐を行う時点でフレームヘッダは既にデコードされているため，分岐を行う際にデコードしようとしているフレームの種類の情報を利用することは可能である．この分岐により，I 及び P フレームは通常通りデコードされるが，B フレームは近似デコードされる．

3.3 B フレームの近似デコード

図4(a)に，通常のフレームのデコード処理の詳細なフローを示す．これは，図3(a)及び図3(b)中に処理「フレームデコード」として示す部分に対応する．フレーム中に存在するすべてのスライスについて，それぞれに含まれるマクロブロックをすべてデコードする．



(a) 通常のデコーダ　(b) 提案手法におけるデコーダ

図 4: フレームのデコード処理のフロー

本研究で行う，視線情報を利用した B フレームの近似デコード処理の詳細なフローを，図4(b)に示す．これは，図3(b)中に処理「フレーム近似デコード」として示す部分に対応する．このフローでは，ユーザの視線情報を利用してデコード範囲を計算し，視線から離れた一部のマクロブロックのデコードを省略する．デコードを省略したマクロブロックについては過去のフレームのデータで代替する．

3.4 B フレーム中で通常のデコードを行う範囲

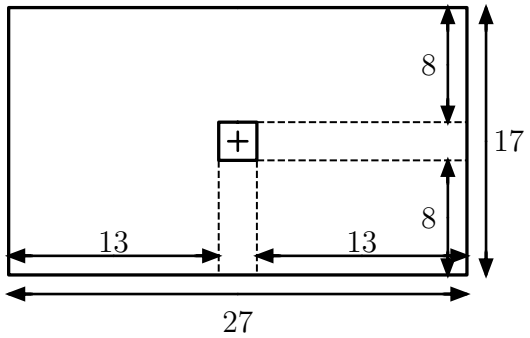


図 5: デコードを行うマクロブロックの範囲

B フレーム中で通常のデコードを行う範囲を図 5に示す．図 5に示すように，デコード範囲は，視線の存在するマクロブロックから水平方向に 13 ブロック以内，鉛直方向に 8 ブロック以内にある合計 459 個のマクロブロックである．この値は，人間の有効視野が視線を中心として左右にそれぞれ 15°，上下にそれぞれ 10° である [10] ことを用い，アプリケーションウィンドウからユーザの目までの距離が 35 cm であることを仮定して算出された．視線が画面から完全に外れた場合でも，視線から最も近い画面内の位置に視線があるとして計算が行われる．

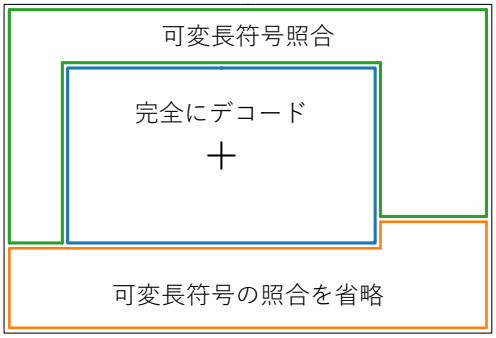


図 6: デコードの種類によるフレーム領域の分類

MPEG-1 ビットストリームは可変長符号の連続により表現されるため，近似デコードを行うフレーム領域の右下を超えるまでは，後続する符号との整合のために，可変長符号の照合を行う必要がある．近似デコードを行うフレーム領域の右下よりも後のマクロブロックからは，可変長符号の整合を保つ必要がなくなり，開始コードの検索のために 1 バイトずつビットストリームをシークできる．このことにより，近似デコードが行われるフレームのビットストリームは 2 つの領域に分けられる．1 つめは，可変長符号が照合される部分である．2 つめは，可変長符号の照合が省略される部分である．可変長符号が照合される部分は，照合した可変長符号を用いて実際にデコードを行うかでさらに 2 つに分けられるため，近似デコードが行なわれるフレームは，3 つの領域に分類される．1 つめは，完全にデコードされる領域である．2 つめは，可変長符号は照合されるが，照合した符号は使用されない領域である．3 つめは，可変長符号の照合が省略される領域である．デコード後のフレームにおけるそれぞれの領域を，図 6に示す．

4. 実装

4.1 実装したアプリケーションの構成

実装したアプリケーションの構成を図 7に示す．アプリケーションはデコーダ，バッファ，及び UI の 3 要素からなり，デコーダ及びバッファは C# 言語，UI は C# 言語及び XAML により記述される．これらをユニバーサル Windows プラットフォーム (UWP) アプリケーションとしてパッケージングし，HoloLens 2 上に配置する．

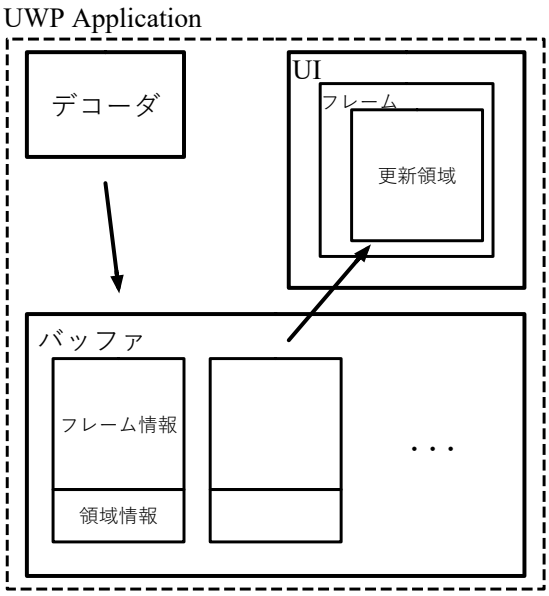


図 7: アプリケーションの構成

デコーダと UI はそれぞれ異なるスレッドで並列に処理を行うが，バッファを介して通信する．言い換えると，デコーダはフレーム情報とその画面上での位置の情報の組となるデコード結果をバッファに格納し，UI は格納されたデコード結果を取得して画面上での位置の情報を利用して以前のフレームに上書き描画を行う．

4.2 バッファの仕様

MPEG-1 映像中の P フレーム及び B フレームはデコードの際に他のフレームを参照するため，MPEG-1 映像をデコードするためにはフレームバッファが必要となる．また，MPEG-1 映像のフレームレートは最低でもおよそ 24 fps であるため，すべてのフレームを保持することはできず，描画済みかつ参照しないフレームのデータは削除する必要がある．これらの機能を満たすため，フレームバッファは，配列にアクセスする際にその添え字を配列長の剰余とすることによるリングバッファとして実装を行った．しかし，リングバッファは配列を再利用するため，デコードが描画よりも高速に行われた場合，まだ描画が行われていないバッファに書き込みが行われる．これを抑制するため，バッファに条件変数を導入した．

ソースコード 1は，条件変数によるスレッド同期を行う処理の一部である．まず, 3 行目でデコーダスレッドがバッファのロックを取得する．ここで，Buffer クラスのメンバ変数 Lock は排他ロックのトークンとなるオブジェクトであり，それ自体に意味はない．次に，5 行目から 8 行目で，そのバッファが描画可能な状態である間，デコーダスレッドはバッファのロックを開放し待機する．バッファの描画が行われた場合，UI スレッドにより条件変数が変更され，

ソースコード 1: 条件変数によるスレッド同期処理

```
1  Buffer buffer = buffers[index % length];
2
3  lock (buffer.Lock)
4  {
5      while (buffer.State == BufferState.DrawReady)
6      {
7          Monitor.Wait(buffer.Lock);
8      }
9      buffer.Write(...);
10     Monitor.PulseAll(buffer.Lock);
11 }
```

デコーダスレッドに通知される．これにより，デコーダスレッドは 9 行目から 10 行目のバッファへの書き込み及び他のブロックされているスレッドの解放を行うことができる．

4.3 デコーダの同期

視線を利用しない場合，デコーダは描画と非同期にデコードを行うことができる．しかし，デコード処理に視線情報を利用すると，デコーダが描画に大きく先行できないという制約が生じる．デコーダが大きく先行すると，フレームがデコードされてから描画されるまでの時間が増加することにより，フレームがデコードされてから実際に描画されるまでの視線の移動距離がより大きくなる．これは，ユーザの立場からすると，過去に見ていた点を中心として映像がデコードされるように見える．したがって，デコーダの先行を抑え，描画と同期する機構が必要となる．この機構は，バッファに実装される条件変数によるスレッド同期を利用して実現できる．条件変数によるスレッド同期により，デコーダはバッファ長を超えて先行することができないため，バッファ長を制限することでデコーダの過剰な先行を防ぐことができる．

4.4 提案アプリケーションによってデコードされたフレームの例

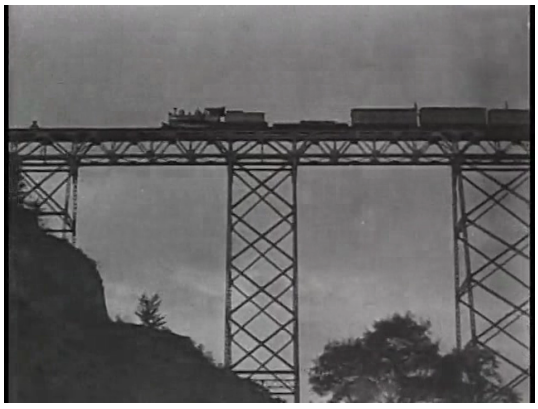


図 8: 通常の B フレーム

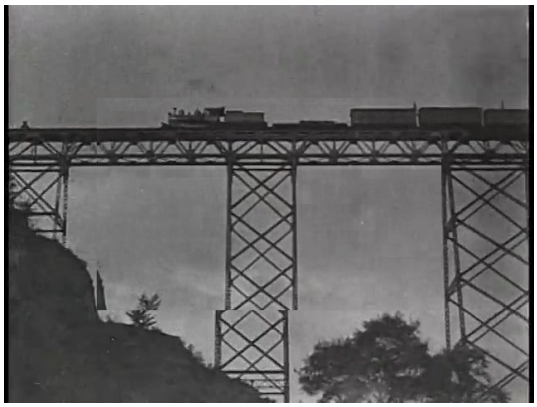


図 9: 近似デコードを行った B フレーム

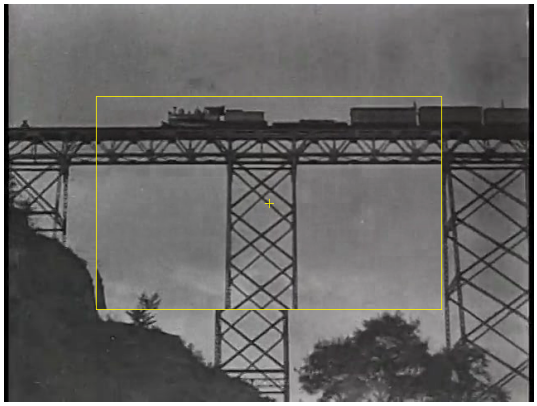


図 10: B フレーム中のデコードされた部分

図 8及び図 9に，映像中のある同一の B フレームに対し，通常のデコード及び近似デコードを行った結果を示す．また，図 10に，図 9中における視線位置及び正確なデコードを行った部分を示す．MPEG-1 映像において B フレームの挿入される割合には幅があるが，図 1に示すフレーム列を仮定すると，3 フレームあたりに 2 フレームの割合で B フレームが存在することにより，過去のフレームの再利用を行う部分での事実上のフレームレートはもとのフレームレートの 3 分の 1 となる．

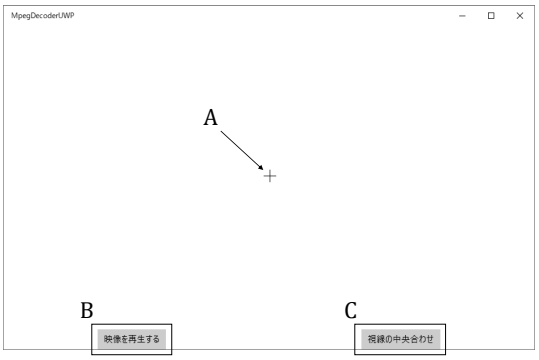


図 11: 実験 1 で用いたアプリケーションの UI (視線調整時)

図 12: 実験 1 で用いたアプリケーションの UI (映像再生時)

5. 実験

5.1 実験 1：デコードされるマクロブロックの数の調査

表 1: 実験 1 で使用した映像の情報

項目	値	備考
幅 [ピクセル]	640	
高さ [ピクセル]	480	
フレーム当たりのマクロブロック数	1200	
I フレーム	125	全体の 8.5%
P フレーム	366	全体の 24.9%
B フレーム	978	全体の 66.6%
総フレーム数	1469	
総マクロブロック数	1762800	

ユーザが使用した際に実際にデコードされるマクロブロックの数を調査するため，被験者 5 名を対象とした実験を行った．

5.1.1 実験に使用する映像

実験に用いた映像の情報を表 1に示す．映像中には 3 フレームあたりに 2 フレームの割合で B フレームが存在するため，過去のフレームの再利用を行う部分でのフレームレートは，もとの値の 3 分の 1 となる．B フレーム中でデコードされるマクロブロックの数は，最大で図 5に示す 459 個である．これはフレーム中の 38.25% の面積を占める．

5.1.2 実験に使用するアプリケーション

実験に用いたアプリケーションの UI を図 11及び図 12に示す．

図 11は映像を再生する前の UI の状態である．この画面はユーザの視線を校正する目的を持つ．UI は，映像を再生する代わりに画面の中央部に十字を表示する．これは図中の A と対応する．ユーザはこの十字に視線を合わせた状態で，図中に C で示すボタンを押す．これにより，システムはユーザの視線の原点の情報を得る．視線が調整された後に図中に B で示すボタンを押すことにより，映像の再生が開始される．

映像の再生が開始された後の UI の状態を図 12に示す．

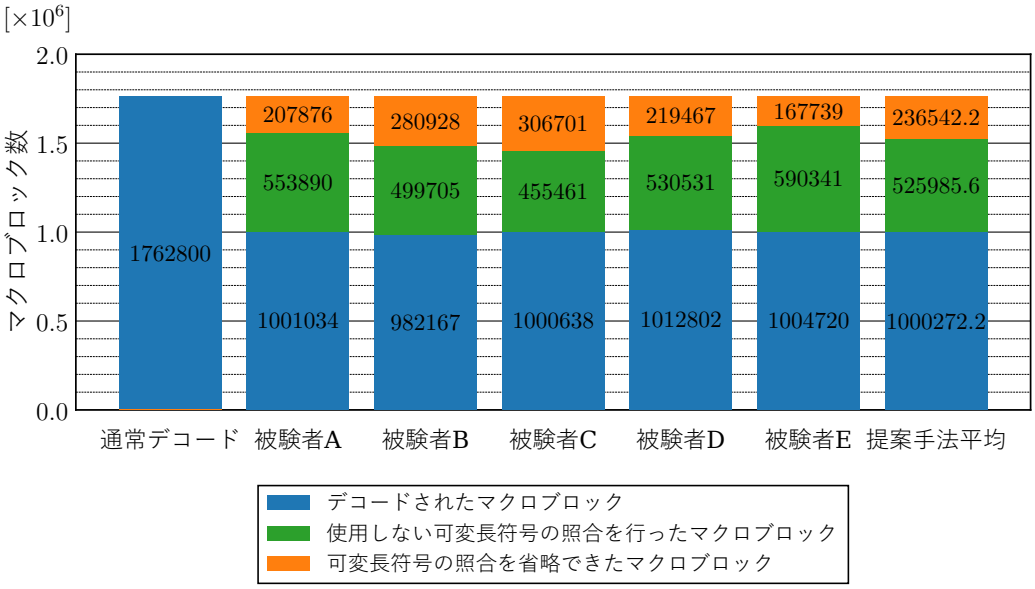


図 13: デコードされたマクロブロックの数

この画面では，提案手法によりデコードされた映像が中央に表示される．

5.1.3 実験の手順

実験手順は以下の通りである．

(1) 被験者はヘッドマウントディスプレイを装着し，視線認識機能の校正を行う．
(2) 被験者は，アプリケーションウィンドウを実測で顔から 35 cm 前方に調整する．
(3) 被験者は，視線をアプリケーション中央の十字に合わせた状態で，図 11に C で示すボタンを押す．
(4) 被験者は，図 11に B で示すボタンを押し，再生される映像を見る．
(5) アプリケーションは，被験者の視線情報を取得して近似デコードを行いつつ，そのときの視線位置，通常のデコードを行ったマクロブロック，可変長符号の読み出しのみ行ったマクロブロック，可変長符号の読み出しを省略できたマクロブロックを数える．
(6) アプリケーションは，視線位置の配列に加え，数えた 3 つの値を映像再生の終了時にファイルに出力する．

5.1.4 実験の結果

図 13に，実際にデコードされたマクロブロック，使用しない可変長符号の照合を行ったマクロブロック，及び，可変長符号の照合を省略したマクロブロックそれぞれの数を，被験者ごとに示す．5 名の平均をとると，デコードされたマクロブロックの数は 1000272.2 で全体の 56.7%，使用しない可変長符号の照合を行ったマクロブロックの数は 525985.6 で全体の 29.8%，可変長符号の照合を省略できたマクロブロックの数は 236542.2 で全体の 13.4% であった．ここで，図中の棒は，図 6に示すフレーム中の領域と対応する．

図 14に，B フレームにおけるそれぞれの被験者の視線位置の分布を示す．縦軸及び横軸は，実際の映像における位置に対応する．

表 2に，視線が端に移動することによってデコード領域が見切れた B フレームの割合を示す．映像中に 978 枚存在する B フレームのうちデコード領域が見切れたものは，被験者 5 名の平均をとると 338.4 枚であり，これは B フレーム中の 34.6% にあたる．

図 15及び図 16に，ヘッドマウントディスプレイを装着した状態で，視線を画面上の左及び右に移動したときの同一の B フレームのデコードの様子を示す．それぞれの図では，画面の左及び右の部分のみがデコードされており，それ以外は過去のフレームの再利用となっている．

5.1.5 実験結果の考察

まず，図 15及び図 16から，実装したアプリケーションにより，視線を利用したデコード領域の動的な変更ができていることが確認できた．また，それに加え，以下の 2 つの点が分かった．

1 つめは，視線情報を動的にフィードバックしたデコードを行うことで，デコードを行う面積は 56.7% に削減できる点である．ただし，以下の理由により，さらに近似デコードの面積は削減できる可能性がある．この実験においては，被験者には動画を見せる際に特に指示はしていない．これにより，図 14に示すように，被験者は映像の比較的中央付近を注視した．中央付近に多く視線が分布したことによって，通常のデコードを行う領域が画面外にはみ出すことによる計算コストの減少はほぼ生じていない．このことは，描画される領域を画面外に出さないようにした場合にデコードされるマクロブロックの数が 1038102 であるのに対し，提案手法を用いた際に平均的にその 96.3% がデ

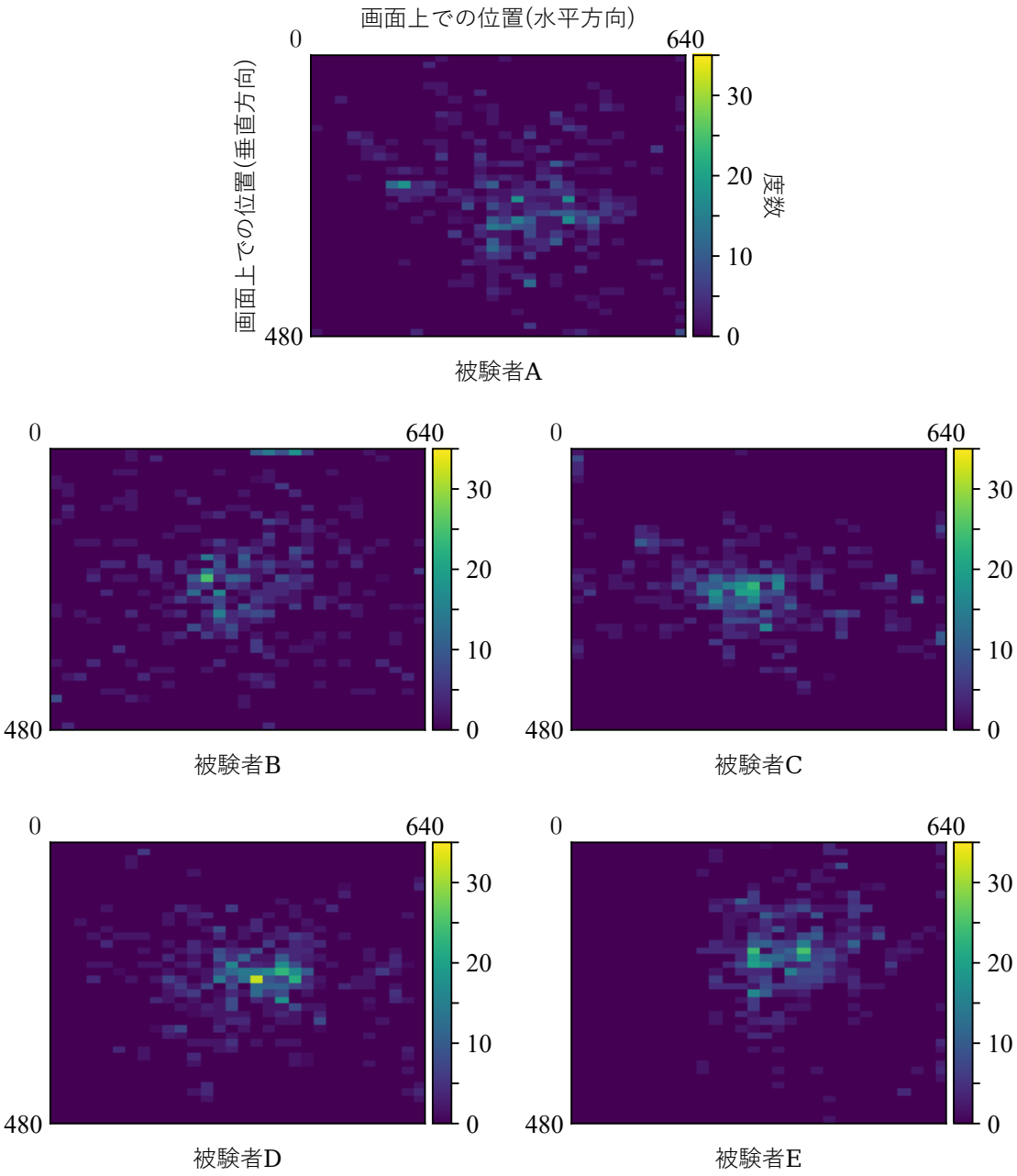


図 14: B フレームにおける被験者の視線位置の分布

表 2: B フレームのデコード領域が見切れた割合

被験者	A	B	C	D	E	平均
B フレームの数	978	978	978	978	978	978
デコード領域の一部が見切れた B フレームの数	388	422	291	258	333	338.4
デコード領域の一部が見切れた B フレームの割合	39.7%	43.1%	29.8%	26.4%	34.0%	34.6 %

図 15: 視線を左に移動した場合の実際のデコードの様子

図 16: 視線を右に移動した場合の実際のデコードの様子

コードされたことに加え，表 2に示す通り，デコード領域が一部でも見切れたマクロブロックの数が 34.6% であることからわかる．したがって，他の要因により被験者の注意が逸らされた場合，通常のデコードを行う領域が画面外にはみ出すことで近似の積極性が引き上げられ，さらなるデコード面積の減少が見込まれる．

2 つめは，使用しない可変長符号の照合を行う必要があったマクロブロックの数は，デコードを完全に省略できたマクロブロックより多い点である．使用しない可変長符号の照合は，可変長符号同士の依存関係から生じている．可変長符号を検索する際には，ビットストリームから 1 ビットずつ取り出して照合することになるため，計算コストが高い．可変長符号の依存関係をより小さい計算コストで解決することで，計算コストを大きく引き下げることができる．

5.2 実験 2：近似デコードの積極性の適切さの評価

表 3: 実験 2 で使用した映像の情報

映像	項目	値
映像 1	幅 [ピクセル]	1280
	高さ [ピクセル]	720
	総フレーム数	930
映像 2	幅 [ピクセル]	1280
	高さ [ピクセル]	720
	総フレーム数	942

表 4: 被験者が選択できるデコード領域の大きさ

段階	縦 [ピクセル]	横 [ピクセル]	面積 [%]
1	144	80	1.25
2	288	160	5.00
3	424	240	11.04
4	568	320	19.72
5	712	400	30.90
6	856	480	44.58
7	1000	560	60.76
8	1136	640	78.89
9	1280	720	100.00

提案手法の近似デコードの積極性が適切であるかを調査するため，被験者 3 名を対象とした実験を行った．

5.2.1 実験に使用する映像

実験に用いた 2 種類の映像の情報を表 3に示す．次節で説明するが，この実験では実験 1 と異なるアプリケーションを使用するため，フレームの種類に関しては記述していない．なお，実験では映像をスクリーン全画面に表示したため，映像の縦横はそれぞれ 3 倍に拡大された．

5.2.2 実験に使用するアプリケーション

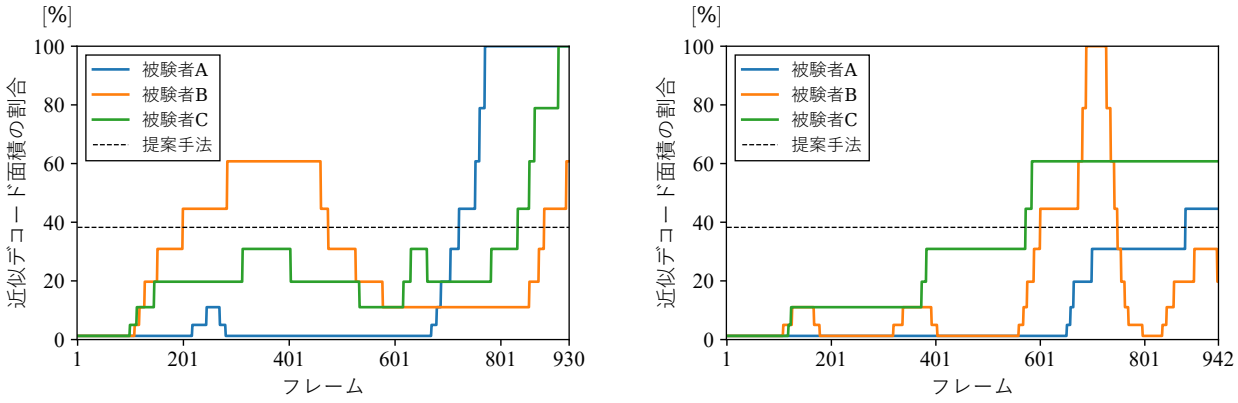
この実験には，提案手法による映像デコード結果をデスクトップ PC 上で再現するアプリケーションを用いた．アプリケーションは 3 フレーム毎に 1 フレームは全ての画面領域を更新するが，残りの 2 フレームは視線を中心とした画面領域のみを更新する．被験者の視線はデスクトップ PC に装着したアイトラッカーを通じて取得する．このアプリケーションは提案手法で設定した近似デコード範囲の適切さを調査するためのものであり，映像を通常通りにデコードした後，視線情報を利用してフレームを処理することで，提案手法による映像デコード結果を再現する．被験者が見ることになるフレームは提案手法による映像デコード結果と等しくなるが，被験者はデコードを行う領域の大きさをキー入力によって 9 段階に変更することができる．

表 4に，被験者が選択できるデコード領域の大きさを示す．実験開始直後には，デコード領域の大きさは 1 段階目に設定されているが，被験者はキー入力によってデコード領域の大きさを上または下に 1 段階ずつ変更することができる．

5.2.3 実験の手順

被験者は，以下の実験手順をそれぞれの映像について行う．

(1) アプリケーションが映像を再生する．
(2) 被験者は，再生される映像を見ながら，キー入力によってデコード範囲を調節する．
(3) アプリケーションは，被験者の視線情報を取得して近似デコード結果の再現を行いつつ，そのときのデコード領域の大きさを記録する．
(4) アプリケーションは，デコード領域の大きさの配列を



(a) 映像 1 についての結果　　(b) 映像 2 についての結果

図 17: デコード領域の大きさの時間変化

デコードの終了時にファイルに出力する．

5.2.4 実験の結果

図17(a)及び図17(b)に，それぞれの被験者が 2 つの映像に対してフレーム毎に選択したデコード領域の大きさを示す．図17(a)に示す通り，映像 1 の後半では全ての被験者がデコード領域を画面の全ての領域にまで広げた．これは，映像 1 が比較的動きのある映像であり，後半になるにつれて激しくなることによる．映像 2 は映像 1 よりも動きが穏やかであるものの，それでも図17(b)に示す通り映像後半で被験者は提案手法の領域よりも大きいデコード領域を選択した．

5.2.5 実験結果の考察

実験結果から，人間は有効視野の外側であっても動きを感知し違和感を覚えることが分かった．このことから，提案手法では人間の有効視野をもとに固定のデコード範囲を設定したが，映像の動きの激しさに応じてデコード範囲を動的に変化させることで，近似デコードの効率を引き上げつつ，ユーザの覚える違和感を減少させることができると予想している．提案手法では，現在の視線位置の情報だけを近似デコーダへのフィードバックとしていた．しかし，映像それ自体の情報もデコーダへとフィードバックすることで，さらに柔軟で積極的な近似計算を行うことができることが見込まれる．

6. 関連研究

この章では，本研究と関連する研究について述べる．

6.1 計算機科学における視線の応用

Duchowski [1] は，計算機科学における視線の応用例を，Selective Systems 及び Gaze-Contingent Displays の 2 つに分類した．

Selective Systems は，視線をポインティングデバイスとして用いることでシステムとのインタラクションを図る手法である．PC のマウスを視線により代替することを例として考える．まず，視線の移動速度は，眼球から 50 cm 前方に位置する 21 インチディスプレイの対角上を 150 ms 程度で移動できる高速性を有し [8,9]，これを有効に活用することができる．反面，視線の精度はディスプレイ上の 1 ピクセル単位を正確に動かすには不十分である．また，人間が 1 点を注視している場合でも視線は固視微動によって常に細かく振動しているため，入力の正確さに難がある [8]．加えて，マウスを完全に代替するためには，クリックを入力できる必要がある．クリック動作を代替するために，以下の 2 つの手法が挙げられている．1 つめの手法は，瞬きをクリックとみなす方法である．しかし，人間は定期的に瞬きをするうえ，瞬きを意図的に連続して行うのは難しい．2 つめの手法は，ある程度の時間の視線移動の停止をクリックとみなす方法である．とはいえ，前述の通り，視線移動は完全には停止しない．また，不適切な停止時間を設定すると，以下の問題が発生する．長すぎる停止時間は入力速度の低下をもたらし，短すぎる停止時間は Midas Touch Problem を引き起こす [1,7]．

対照的に，Gaze-Contingent Displays は事物を観察する視線の本来の役割に着目した手法である．この手法において，視線は人間の注目している点の情報に過ぎず，その情報を有効活用してシステムの描画コストを下げることを図る．具体的には，視線が注目している部分は詳細に描画するが，それ以外の部分については，使用する計算資源を節約した描画を行う．この手法はシステムの計算コストを低下させると同時に，ユーザの視聴体験を損なう可能性がある．計算コストの低下とユーザの視聴体験の減少とのバランスを見極めることが必要となる．

6.2 視線情報を利用した映像描画の高速化

Kaplanyan ら [2] は，以下の手順による映像描画の高速化を提案した．まず，フレームをエンコードする際に，

natural scene statistics に基づいて人間の視線位置を計算し，その情報をもとにフレームからピクセルをサンプリングする．サンプリングレートは，視線位置から計算される網膜の細胞密度配置に従って決定される．そして，サンプリングされたピクセルを，元の画像中における位置とともに符号化する．フレームをデコードする際には，符号中に含まれる情報のみではフレームを完全に復元できない．したがって，あらかじめ学習を行った Generative Adversarial Networks により不足するピクセルを推論し，これによりフレームを復元する．

この手法はユーザ体験を損なわずに描画コストを削減する [2] 一方で，以下の 3 つの欠点がある．1 つめは，人工的な映像に弱い点である．人工的な映像においては，natural scene statistics を利用した人間の視線位置の決定はできない．2 つめは，既存の映像ファイルは，ファイル形式を変換する必要がある点である．この手法におけるデコーダは，サンプリングされたピクセルの情報及びそのピクセルの元の画像中における位置を必要とする．したがって，別の形式によりエンコードされた映像ファイルは，一度各フレームをビットマップにデコードし，再エンコードする必要がある．3 つめは，エンコードする際に視線位置を決定するため，実行時の動的な近似精度の変更ができない点である．

7. まとめ

本研究では，ユーザの視線をフィードバックしてその周囲のみをデコードすることにより，近似の積極性を動的制御する近似デコード手法を提案した．そして，提案手法を実装，ヘッドマウントディスプレイに配置し，実際のデコード領域についての実験を行った．また，近似の積極性の適切さを調査する実験も行った．これらの実験からは，提案手法よりもさらに多くの計算コストを削減できる場合があること，及び，提案手法の近似計算の積極性が過剰となる場合があることが分かった．しかし，それと同時に，動的な近似デコードによりユーザの満足度を維持しつつ効率的な映像デコードを行うことができることが分かった．

今後の展望としては以下の 2 つが挙げられる．1 つめは，視線の現在位置だけでなく，映像それ自体の情報も用いることにより，より大きく積極性を変更できる手法への改良を行うことである．本手法では，近似計算の積極性を変更できる量に限界があるため，変化するユーザの状態に追従しきれない．2 つめは，近似計算を行うハードウェア及びアーキテクチャとの連携である．本研究で，ヘッドマウントディスプレイ上に配置したアプリケーションにおける近似計算はソフトウェアのみにより行われており，ハードウェアのサポートを受けていない．これにより，十分なデコード速度が確保できず，また，近似計算を行うことにより節約できる計算資源の量にも限界があった．より低いレイヤからの実装を行った場合，デコード速度が向上するとともに，近似計算のさらなる効率化が可能となる．これによって，さらに高効率かつ積極的な近似計算を達成できる．

参考文献

[1] Duchowski, A. T.: A breadth-first survey of eye-tracking applications, *Behavior Research Methods, Instruments, & Computers*, Vol. 34, No. 4, pp. 455–470 (2002).

[2] Kaplanyan, A. S., Sochenov, A., Leimkühler, T., Okunev, M., Goodall, T. and Rufo, G.: DeepFovea: Neural Reconstruction for Foveated Rendering and Video Compression Using Learned Statistics of Natural Videos, *ACM Trans. Graph.*, Vol. 38, No. 6 (online), doi:10.1145/3355089.3356557 (2019).

[3] Le Gall, D.: MPEG: A Video Compression Standard for Multimedia Applications, *Commun. ACM*, Vol. 34, No. 4, p. 46–58 (online), doi:10.1145/103085.103090 (1991).

[4] Li, Z.-N., Drew, M. S. and Liu, J.: *MPEG Video Coding: MPEG-1, 2, 4, and 7*, pp. 341–394 (online), doi:10.1007/978-3-319-05290-8_11, Springer International Publishing (2014).

[5] Marshall, D.: The MPEG Video Bitstream, available from ⟨https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node262.html⟩ (2003). Accessed: 2020-09-30.

[6] Marshall, D.: MPEG Video Layers, available from ⟨https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node257.html⟩ (2003). Accessed: 2020-09-30.

[7] Singh, H. and Singh, J.: Human eye tracking and related issues: A review, *International Journal of Scientific and Research Publications*, Vol. 2, No. 9, pp. 1–9 (2012).

[8] 大野健彦：視線を用いた高速なメニュー選択作業，情報処理学会論文誌， Vol. 40, No. 2, pp. 602–612 (1999).

[9] 大和正武, 門田暁人, 松本健一, 井上克郎, 鳥居宏次：一般的な GUI に適した視線・マウス併用型ターゲット選択方式，情報処理学会論文誌，Vol. 42, No. 6, pp. 1320–1329 (2001).

[10] 畑田豊彦ほか：眼・色・光 より優れた色再現を求めて，p. 9, (公社) 日本印刷技術協会 (2012).