「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

同世界放送：リアルタイム映像の収集と合成を伴う分散型インターネットライブ放送システム

牧田 航輝[1] 川上 朋也[1] 松本 哲[2] 義久 智樹[2] 寺西 裕一[3,2] 下條 真司[2]

概要：2019 年末に発生した新型コロナウイルス感染症（COVID-19）の影響により，テレワークやオンライン授業，エンターテインメント業界など，様々な場面でリアルタイム映像配信サービスの需要が高まっている．複数の参加者を同時に画面上に表示する場合，画面を分割して決まった領域に各参加者を配置するものが多くみられるが，それでは各参加者が孤立した感覚となる．全参加者が 1 つの空間内にいるかのような表示ができれば，より現実感のあるライブ放送が実現できる．本研究では，複数の遠隔地で撮影された参加者がまるで同じ空間に存在するかのようなライブ放送を「同世界放送」と呼び，そのシステム提案する．同世界放送では多数の映像処理が必要となり，それらの処理を特定のノードに集中させると，処理が完了するまでの遅延が大きくなるという問題が発生し得る．処理遅延を小さくするため，中継ノード上で映像を合成しつつ収集する方法を考案した．同世界放送システムを実装し，実装したシステムを用いて各合成方法における処理遅延時間を評価した．

1. はじめに

YouTube Live などのインターネットライブ放送サービスが，近年広く普及している．配信者は，このようなサービスを利用して映像や音声を配信する．2019 年末に発生した新型コロナウイルス感染症（COVID-19）により外出自粛を求められ，テレワークやオンライン授業など，日常生活の中でリアルタイム映像配信サービスが広く利用される状況となった [1], [2]．映像配信では，個人が撮影した単一の映像を配信することだけでなく，複数の配信者によるコラボレーションもあり得る．その場合，多くの映像配信サービスでは，画面を分割して決められた領域内に各配信者による撮影対象を表示する方法がとられている．しかし，それでは各撮影対象が孤立しており，配信者にとっても視聴者にとっても満足できる配信とは言い難い．配信者側では，複数の映像が分離されているために他の配信者とのコミュニケーションが取りづらい．また，視聴者側では，通常のテレビ放送における中継のように遠隔地でのやり取りをしているような感覚となってしまう．このように，双方にとってコラボレーションが行われている感覚が希薄となる問題がある．画面を分割することなく，撮影対象が同じ空間に存在するかのような表示ができれば，高いエンターテインメント性を生み出したり，テレワークやオンライン授業などの際に生じ得る違和感や不快感を軽減したりする効果が期待できる．

本研究では，全ての撮影対象がまるで同じ空間に存在するかのようなライブ放送を「同世界放送」と呼び，合成によって生成する映像を「同世界映像」と呼ぶ．図 1 が従来の放送，図 2 が同世界放送のイメージである．図 2 のように，複数の配信者による撮影対象全てをある一つの背景上に合成する方法が考えられる．そうすることで，全ての撮影対象がまるで同じ空間に存在するような，現実感のあるライブ放送が実現できる．同世界放送の適用例としては，参加者が仮想空間内に自身や物体を配置した遠隔ミーティングなど，特定の対象の遠隔での存在感を高めることが考えられる．また，COVID-19 の影響で中止となった「仙台すずめ踊り」について，各演者が自宅で撮影した映像を募り 1 つの映像に合成することで，仮想的にイベントを実現した例がある [3]．この例は，参加者から録画された映像を募集し事後的に編集したものであり，リアルタイムに映像の合成をしている訳ではない．しかし，これをリアルタイムに合成することでさらなる現実感を生み出す可能性がある．また，演劇のオンライン化の動きもみられる [4]．このように，異なる場所にいる出演者のリアルタイム映像を用いた演劇や踊りなどのパフォーマンスでは，視聴者は出演

[1] 福井大学
University of Fukui
[2] 大阪大学
Osaka University
[3] 情報通信研究機構
National Institute of Information and Communications Technology

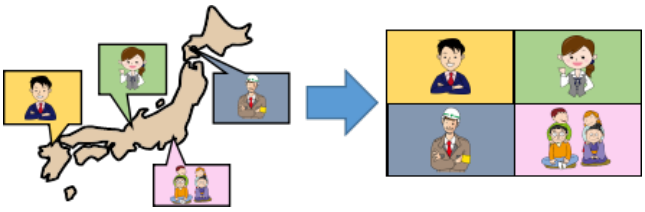

図 1 従来の放送

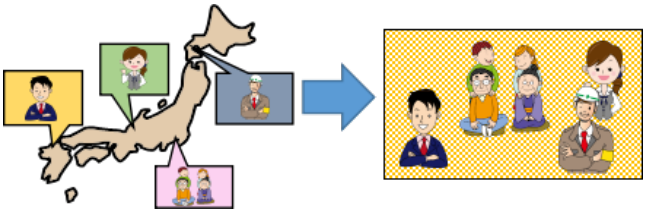

図 2 同世界放送

者が同じ空間で共演しているかのような感覚が得られ，高い臨場感や現実感を同世界映像から得られると考えられる．

同世界放送では，多数のリアルタイム映像をインターネット経由で収集し，低遅延に映像の合成を行い視聴者へ配信する必要がある．そのため，既存の収集方式や処理割り当て方式を用いると，「撮影者（収集対象の映像）の増加に伴い，特定のノードに映像の収集と合成の処理の負荷が集中し，処理の完了までの遅延が大きくなる」という問題が生じ得る．この問題によってノード上の処理や通信の遅延が大きくなると，同世界映像が視聴端末で再生されるまでの待ち時間や視聴中の再生途切れ時間も増大し，視聴者と配信者双方の満足度が低下する．この問題を解決するため，分散コンピューティング環境における「逐次合成処理を伴う分散型リアルタイム映像収集」を考案した．

以下，2 章で関連研究の紹介をし，3 章で本研究で提案する同世界放送システムについて述べる．4 章で実装した同世界放送システムの説明を行い，5 章で実装したシステムを用いた実験と評価を述べる．最後に，6 章で本研究のまとめと今後の課題について述べる．

2. 関連研究

リアルタイム映像の合成に関する類似研究がある．例えば，超高臨場感通信技術 Kirari![5], [6] では，あたかも目の前に別の空間にいる人が存在するかのような超高臨場感リアルタイム中継を目指している．そのためには，高い解像度と広い視野角を持つサラウンド映像を生成する必要があり，複数の 4K カメラで撮影した映像をリアルタイムに合成し，遠隔地へと伝送する技術を提案している [7], [8]．また，文献 [9] では，自由視点映像をリアルタイムに配信するための映像合成技術と，5G 網を用いた低遅延な配信システムを提案している．文献 [10] では，交差点付近で複数車両が撮影した映像をリアルタイムに車車間通信で交換し合成することで鳥瞰映像を生成するシステムを提案している．また，複数カメラから得た映像を合成するのではなく，実写映像と CG を合成させる研究もある．文献 [11] では，ハンディカメラの動きを計測し，その動きを CG 空間内の仮想カメラに反映させることによって実写映像と CG をリアルタイムに合成させるためのハイブリッドセンサを開発している．これらの研究は，ある特定の地点を撮影した映像を用いて合成しているのだが，本研究のように複数地点で撮影された映像をインターネット経由で収集し合成することは想定されていない．

また，遠隔での存在感や臨場感に着目した類似研究もされている．文献 [12] では，異なる場所の参加者や家具をヘッドマウントディスプレイ（HMD）に表示し，まるで参加者や物体が付近に存在するかのように臨場感を高めている．文献 [13] では，ユーザと対話相手を同じ映像内に合成した時に，「同じ空間にいる」という感覚（ソーシャルテレプレゼンス）を強化させることを目的としている．文献 [14] では，カメラとプロジェクタを用いてリアルタイムに異なる部屋にいる人を実物大で映し出すことによって，存在感を高める技術を提案している．これらの研究はいずれも，遠隔での存在感や臨場感を高めることを目的としているのだが，多数の映像の収集や合成，撮影対象者以外の視聴者への映像配信は想定されていない．

Microsoft Teams の Together mode[15] では，参加者の上半身が切り抜かれ，最大 49 人までの参加人数に応じて各撮影対象を最適なサイズに変更し，共通の背景に自動的に配置される．同じ空間内に参加者を表示することによって，より自然なコミュニケーションをとることが可能になるとされている．また Apple は，被写体が複数の自撮りを，全員がカメラの前に集まることなく撮影できるように，各々の端末で撮影した画像を合成させる特許を取得している [16]．撮影した後にも，各画像の位置を好きな位置に調整できるようになっている．

3. 同世界放送システム

同世界放送の構成は図 3 に示すものとなっており，分散映像のリアルタイムでの収集と処理を行い視聴者へと配信することを想定している [17]．また，特定のノードに処理負荷や通信負荷が集中し映像の合成が完了するまでの遅延が大きくなることを避けるため，中継ノード上で映像を合成しつつ収集する方法を採る．合成の例を図 4 に示す．図の例では，7 つの映像を合成することを考える．まず，B，D，E に注目すると，D，E からは自身の映像を B へと転送し，B では受け取った D，E の映像と自身の映像を合成した 1 つの映像を生成する．そして，その生成した映像を次の A に転送する．C，F，G でも同様の処理を行い，合成映像を A に転送する．そして，A で 7 つ全ての映像を合成するという流れになっている．このように，中継ノード上で複数の映像を合成し，合成後の映像のみを次のノード

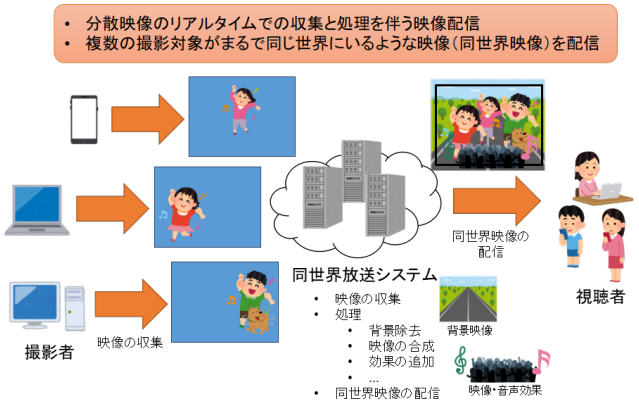

図 3　同世界放送の構成

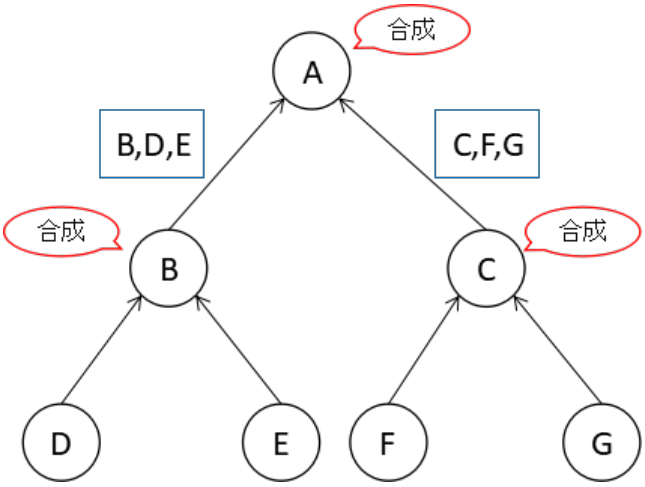

図 4　映像を合成しつつ収集する例

へ転送することで，特定ノードに対する通信負荷の削減や合成処理の分散を行える．

4. 実装

同世界放送システムの実装をした．これまでに述べたように，同世界放送ではすべての参加者が同じ空間に存在するかのような映像を生成するために，様々な処理が考えられる．

まず，必須の処理として，共通の背景に全参加者を表示するために，各参加者の背景を除去することが挙げられる．背景については，あらかじめ用意した画像を用いることや，ある 1 人の参加者に限って背景除去を行わずに，その参加者の背景を利用することも考えられる．本実装では，前者の方法で背景を選択することとした．また，背景除去の方法については，移動物体を検出する背景差分法 [18] や，機械学習や深層学習を用いた被写体抽出技術 [19], [20]，特定の色を透過するクロマキーなど様々な方法がある．撮影者としては，特別な設備などを必要とせず一般的なカメラで撮影するだけで処理されることが望ましいが，本実装では，実装の簡単さと合成映像の美しさの観点から，クロマキーを採用した．グリーンバックやブルーバックを用意する必

図 5　実装したシステムのスクリーンショット

要はあるが，比較的手間のかからない方法で，複雑な処理を必要とせず綺麗に被写体を抽出し合成可能である．

その他の処理としては，映像のサイズや表示位置の変更，実際に撮影したもの以外の映像・音声効果の追加などが考えられる．本実装では，「画面内の任意の位置に表示する」「拡大・縮小をする」「回転させる」の 3 つを主な機能として追加した．これらは，各撮影者が自身の映像に対してのみ操作可能となっている．また，各撮影者が自由に表示位置を変更させると，2 人以上の参加者が同じ位置に表示させようとしたときに，「どの映像が前面に表示されるのか」という問題が発生する．そこで，「重なり順を変更する」操作も可能とした．

図 5 は，実装した同世界放送システムのスクリーンショットである．図は，講義室の風景を背景画像に設定し，7 人の参加者が椅子に座っている様子を再現したものである．左側に合成された同世界映像が，右側に映像に対して操作を行うためのパネルが表示されており，参加者各々がこのような画面を持っている想定である．表示位置については，左の画面内の任意の位置をクリックすることでその位置へと移動させることができるようになっている．また，赤いパネルの scale で拡大・縮小の倍率を指定でき，青いパネルの angle で回転角度を指定し，rotate で指定した角度に回転，reset で元の角度に戻すことができる．緑のパネルの depth で重なり順（前面に表示するか背面に表示するか）を設定できるのだが，これは数値を大きく設定したユーザほど前面に表示されるようになっている．depth の値による重なり順の違いについて，図 6 に例を示す．図の例では，A，B，C という 3 人の映像を合成する場合を考えている．図の左側にあるような各参加者の映像を集めて合成し，図の右側のような同世界映像を生成する．この時，depth の値が，A が 30，B が 10，C が 50 の場合，値は C>A>B となるので，C が前面，A が中間，B が背面に表示されることとなり，結果として図中の同世界映像 1 が生成される．また，depth の値が，A が 10，B が 50，C が 30 の場合，値は B>C>A となるので，この場合は B が前面，C が中間，A が背面に表示されることとなり，図中の同世界映像 2 が生成されるということになる．

また，映像の収集と合成方法については，図 7 に示すよ

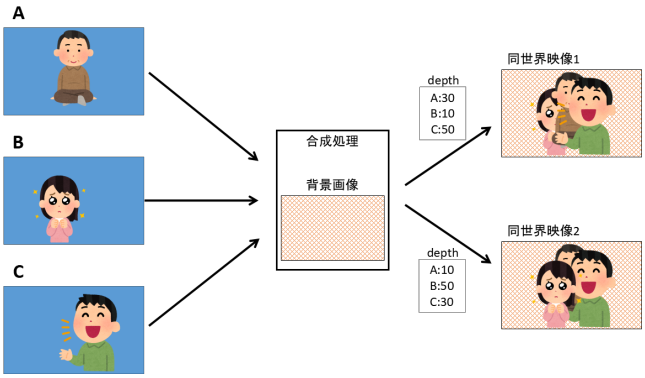

図 6 depth の値による各映像間の重なり順の例

うな特定のノードのみに映像を集めて合成処理をする「集中合成」，図 8 に示すような合成しつつ収集する「分散合成」を実装した．それぞれの方法における映像の合成の流れについて説明する．まず，集中合成については，撮影した映像を送信するのみのノードと，自身の映像と受信した映像の合成処理を行うノードの 2 種類がある．送信のみをするノードは，ブルーバックの下で撮影した映像を，合成を担当するノードへと転送する．また，映像のみでなく，指定した表示位置の座標，拡大倍率，回転角度，深度（前面に表示される優先度の値），送信側の IP アドレスの情報も送信する．映像の転送は FFmpeg[21] によって行われ，その他の情報は mqtt によって通信される．合成を担当するノードは，受信した映像に対して，まず深度情報を確認し，背面に表示されるべきものから合成処理をするように並べ替える．そして，クロマキーにより背景を除去した後に拡大・縮小と回転処理を行い，あらかじめ用意された背景画像に指定された座標へ貼り付ける．この処理を受信した映像の数だけ行うことになる．なお，動画の読み込みや各処理は，コンピュータ・ビジョン・ライブラリの OpenCV[22]，画像処理ライブラリの Pillow[23] によって行われる．次に，分散合成については，集中合成の 2 種類のノードに加え，自身の映像と受信した映像を合成し，その合成映像を次のノードへと転送する中継ノードが存在する．中継ノードでの処理手順は，合成を担当するノードの処理と同じ流れになるが，背景画像に青の単色を使用する点が異なる．青い背景に合成して転送することで，次のノードも同様の処理でクロマキーによる背景除去が可能となる．また，中継ノードから転送された映像については，それ自体を新たな 1 つの映像とみなすため，拡大等の処理を施すと合成映像全体に対する処理となり，それまでに合成された各映像に対する処理は不可能である．よって，中継ノードから転送された映像に対する処理は，背景除去と合成のみとなる．

5. 実験と評価

4 章で紹介した集中合成と分散合成による，映像合成までの処理時間の違いを確認するために，LAN 環境と情報

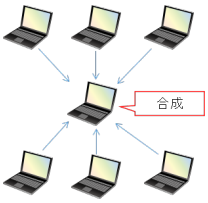

図 7 集中合成

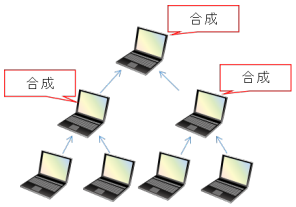

図 8 分散合成

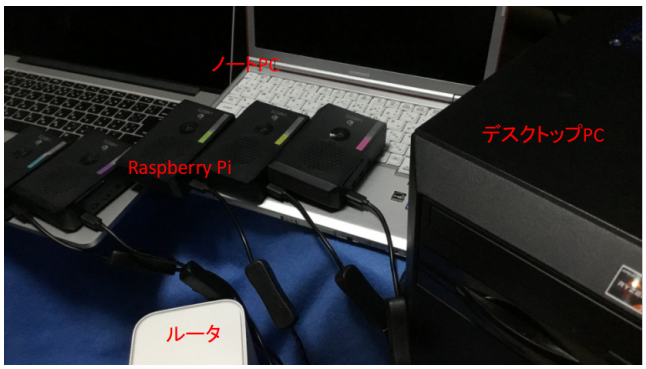

図 9 実験に用いた機器

通信研究機構（NICT）のテストベッド JOSE[24] を利用した環境でそれぞれ実験を行った．

5.1 LAN 環境

5.1.1 実験環境

まず，LAN 環境で実験を行った．分散合成では，どのようにして収集と合成処理を分散させるかが問題となるが，本実験では 2 分木を構成した．実験には，図 9 のように，ノートパソコン 2 台，デスクトップパソコン 1 台，Raspberry Pi5 台，ルータを用いた．各端末の詳細を表 1 に示す．集中合成では，デスクトップパソコンがすべての映像を受信し合成処理をする．分散合成では，最終的な合成をするノードにデスクトップパソコン，中継ノードにノートパソコン，末端のノードに Raspberry Pi を用いた．Raspberry Pi は 5 台用意したが，1 台は合成にかかる時間を計測するために使用したため，実際に映像の合成に利用したものは 4 台のみである．

4 章で述べたように，背景除去はクロマキー合成により行うため，ブルーバックを用いて撮影する必要があるのだが，全ての映像をそのような環境でリアルタイムに撮影することが困難であったため，あらかじめブルーバックの下で撮影した動画ファイルを用いた．映像サイズは，各端末からの送信時は 640x480，合成される同世界映像は 960x540 となっている．mqtt ブローカーには時間の計測に利用した Raspberry Pi を設定した．通信はルータを介して、デスクトップは有線（ケーブル：CAT5e)、それ以外は無線（IEEE 802.11ac）で行われる。実験で行った処理は映像の合成のみであり，映像の拡大や表示位置の変更といった操作は一切行わなかった．

実験は，集中合成と分散合成の各合成方法について，1フレームの合成にかかる処理時間を計測した．用意できた端末の数の都合上，合成数は 4 から 7 までの範囲で計測した．合成数 3 以下では，集中合成と分散合成の合成方法が同じであるため計測しなかった．処理時間は，映像の送信のみを行う末端のノードから最終的な合成が完了するまでの時間を計測し，経路数が複数存在する場合には，各経路ごとにかかった時間の平均をとることとした．また，ネットワーク帯域幅の違いにる影響を確認するため，ネットワーク帯域を制限しない場合と 10Mbps に制限した場合の 2 通りで実験を行った．

5.1.2 実験結果

図 10 は，帯域制限がない場合の実験結果を示している．横軸は合成した映像の数，縦軸は 1 フレームの合成にかかった処理時間を表している．また結果は 5 回の計測の平均値をとっている．図 10 を見ると，集中合成では合成数が増加すると処理時間もわずかに増加しており，分散合成では合成数が 5，6 の時に若干処理時間が短くなり，合成数 7 の時が最も処理時間がかかる結果となった．分散合成において，途中で処理時間が短くなっているが，これは使用した端末の処理速度の違いや通信状況によるものと考えられる．2 分木では映像の送信のみを行うノードから最終的な合成を行うノードまでの経路数について，合成数が 4 の時には 1 つ，合成数が 5 の時には 2 つ存在する．結果の詳細を確認したところ，合成数 5 の時に追加された経路での合成にかかる時間が短くなっていたため，平均をとった結果処理時間が短くなったものと言える．また，集中合成と分散合成を比較すると，どの合成数においても集中合成の方が短い時間で合成処理が完了していることが分かる．これは，「分散合成によって処理遅延を短くすることができる」という仮定に反しているが，その原因に合成数の少なさが挙げられる．処理時間は，「各ノードで映像の合成処理にかかる時間」と，画像のエンコードにかかる時間やノード間で情報を転送する時間などが含まれる「合成処理以外にかかる時間」に分けられる．前者は処理負荷が集中することによって増加し，後者は通信負荷や最終的な合成を担当するノードまでのホップ数が大きくなることにより増加すると考えられる．集中合成では合成数に関わらず，

表 1 実験に用いた機器の詳細

端末	詳細
デスクトップ PC	OS：Windows10, CPU：AMD Ryzen 7 2700, 8 core, 3.2GHz, Memory：32 GB
ノート PC(1)	OS：Windows10, CPU：Intel Core i5-7200U, 2 core, 2.5GHz, Memory：8GB
ノート PC(2)	OS：OS X El Capitan, CPU：Intel Core i7, 4 core, 2.8GHz, Memory：16GB
Raspberry Pi	OS：Raspbian10, CPU：ARM Cortex-A72, 4 core, 1.5GHz, Memory：4GB

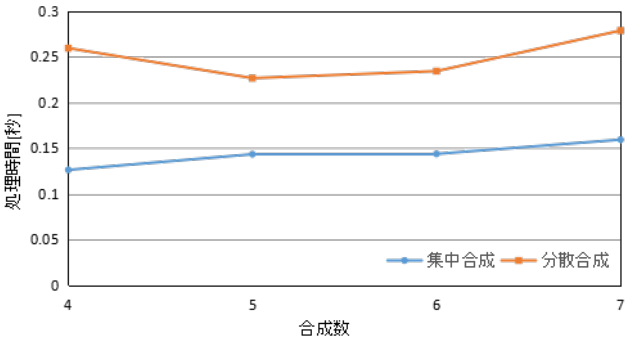

図 10 帯域制限がない場合の実験結果

常に 1 ホップで合成が完了する．2 分木では合成数が 3 までは 1 ホップで合成が完了するが，合成数が 4 から 7 までの間は木の高さが 2 となり合成の完了までに 2 ホップ必要となる．集中合成の結果を見て分かる通り，合成数が少なければ，集中合成であってもそれほど処理遅延が発生しないため，1 ホップ通信が増えてしまった分散合成の方が処理時間が長くなる結果となったと言える．しかし，合成数が 10，20，..... と増加する場合を考えると，集中合成では特定のノードがその全ての合成処理をしなければならず，処理負荷が大きくなっていくと考えられる．一方，2 分木では高さ n で $\sum_{k=0}^{n} 2^k$ 個のノードを持つことができ，合成ノードが取得する映像は最大でも 2 つであるので，合成数が増えるほど各ノードの処理負荷を抑えつつホップ数も抑えることができるため，全体の処理時間の増加も抑えられると考えられる．

図 11 は，10Mbps の帯域制限を設けた場合の実験結果を示している．図 10 と同様に，横軸が合成数，縦軸が処理時間を表している．集中合成では合成数が増加すると処理時間も大きく増加しており，分散合成ではほとんど処理時間が変化していない．分散合成では帯域制限がなかった場合の結果と変わらず，約 0.25 秒の処理時間がかかっているのに対し，集中合成ではどの合成数においても帯域制限がなかった場合に比べ大きく処理時間が増加していることが分かる．これは，実験中に転送した映像 1 つ当たりが約 5Mbps の帯域幅を必要としていたことが原因であると考えられる．分散合成では 2 分木を構成した結果，どのノードにおいても受信する映像の最大数は 2 であるが，集中合成では合成する映像数が増加するたび受信する映像数も増加する．受信する映像数が 2 であれば，帯域幅が 10Mbps に限定されたとしても影響はないが，集中合成ではネットワーク輻輳が発生した結果，処理時間が大幅に増加したと考えられる．

以上の結果より，合成数が少なくネットワーク帯域が十分広い場合には，分散合成より集中合成の方が優れていることが分かった．しかし，ネットワーク帯域が狭い場合には合成数が少なかったとしても分散合成の方が優れている

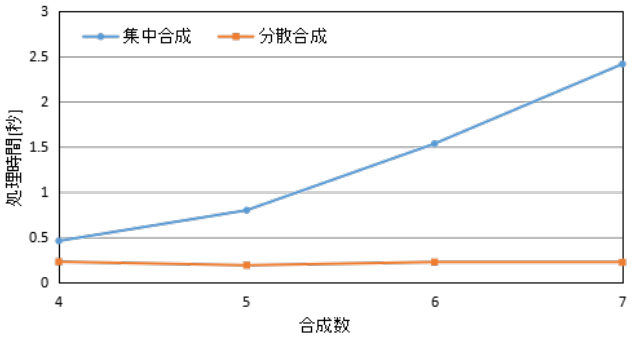

図 11　10Mbps の帯域制限がある場合の実験結果

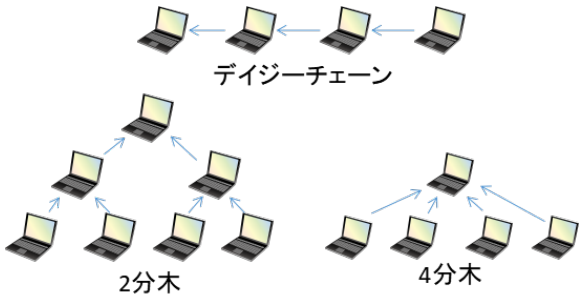

図 12　分散方法

と言える．

5.2　テストベッド環境

5.2.1　実験環境

合成数を増加させ，実際の使用で想定される環境に近い環境で評価を行うため，情報通信研究機構（NICT）のテストベッド JOSE[24] を利用した実験と評価を行った．LAN 環境での実験と同様に，集中合成と分散合成による比較を行ったのだが，分散合成については 2 分木のみでなく，図 12 のようにデイジーチェーンと 4 分木を追加した 3 パターンで実験を行った．

実験には 20 台の仮想マシンを用いた．使用した仮想マシンに 1〜20 までの番号，物理マシンに 1〜10 までの番号を付け，各仮想マシンと物理マシンとの対応は表 2 のようになっている（物理マシンの番号が同じものは同一の物理マシンを表す）．また，IP アドレスが 10.9.x.x と 10.88.x.x では異なるリージョン（北陸と横須賀）となっている．テストベッド内の通信環境の目安として，ping 値を測定した．測定の結果，同一の物理マシン上にある仮想マシン間では約 0.55ms，同一リージョン内にあり異なり物理マシン上にある仮想マシン間では約 0.7ms，リージョンを跨いだ仮想マシン間では約 19ms であった．

LAN 環境と同様に，1 フレームの合成にかかる処理時間を計測した．利用可能な仮想マシンは上記の通り 20 台であるが，そのうち同世界放送のために用いたものは，表 2 中の 1〜18 までの 18 台のみである．19 は正しく合成されていることを確認するために生成された同世界映像を動画ファイルに書き出すために用い，20 は処理時間の計測に用いた．本実験環境では，リアルタイムに撮影した映像を用いることが困難であったため，あらかじめ撮影した動画ファイルで代用した．また，映像のサイズについては，送信時には 540x960，生成される同世界映像は 960x540 となっている．映像の合成については，各映像を 270x480 にリサイズし，すべて画面の中央に合成するようにあらかじめ表示位置を指定した．処理時間の計測中には，表示位置の変更や拡大などの操作は一切行わなかった．

構成したデイジーチェーン，2 分木，4 分木は，それぞれ図 13，図 14，図 15 のようになっている．図中のノード番号は，表 2 に示した仮想ノードの番号に対応している．また，実験で合成数を増加させる場合には，特定の部分木のみが大きくなることがないように，均等に子を追加していった．具体的には，2 分木の場合，図 14 のノード番号で 1，2，3，4，6、5，7，8，12，10，14，9，13，11，15，16，18，17 の順で追加し，4 分木の場合，図 15 のノード番号で 1，2，3，4，5，6，10，13，16，7，11，14，17，8，12，15，18，9 の順で追加した．

LAN 環境における実験の結果，「処理負荷が小さいと集中合成の方が短い処理時間で合成が完了する」という傾向があったので，処理負荷の大小による違いを見るため，「単純に合成のみをする」場合と「合成する際に映像を 2 倍に拡大してから合成する」場合の 2 パターンを実験した．

5.2.2　実験結果

図 16 に単純に合成のみを行った場合の実験結果を示す．

表 2　実験に使用したノード情報

仮想マシン	IP アドレス	物理マシン
1	10.9.0.3	1
2	10.9.0.4	1
3	10.9.0.5	1
4	10.9.0.6	2
5	10.9.0.7	2
6	10.9.0.8	2
7	10.9.0.9	3
8	10.9.0.10	3
9	10.9.0.11	4
10	10.9.0.12	4
11	10.88.0.2	5
12	10.88.0.3	5
13	10.88.0.4	6
14	10.88.0.5	6
15	10.88.0.6	7
16	10.88.0.7	7
17	10.88.0.8	8
18	10.88.0.9	8
19	10.88.0.10	9
20	10.88.0.11	9

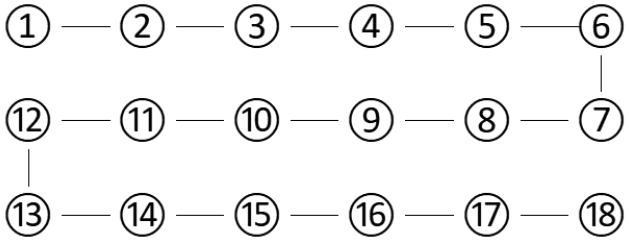

図 13 デイジーチェーン

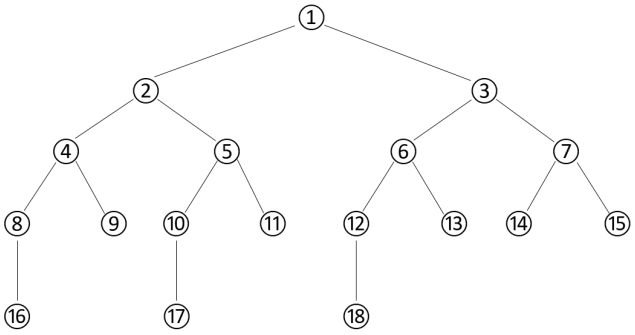

図 14 2 分木

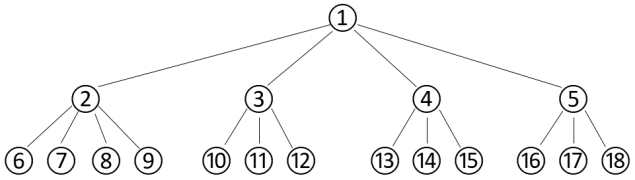

図 15 4 分木

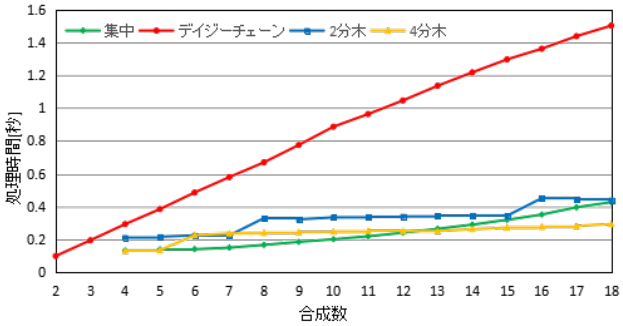

図 16 実験結果

横軸が合成数，縦軸が 1 フレームの合成にかかった処理時間を表している．また，処理時間については，各合成数で 10 回の計測を行いその平均値をとっている．

各合成方法を比較すると，合成数が 12 までは集中合成が最も短い処理時間で合成できており，それ以降では 4 分木の方が処理時間が短くなっている．また，分散合成の 3 パターンの中では，合成数に関係なく 4 分木が常に一番良い結果となっている．続いて，各合成方法における合成数と処理時間の関係について詳細に見る．集中合成については，合成数が増加するにつれて処理時間も増加しており，その増え幅も次第に大きくなっている．デイジーチェーン

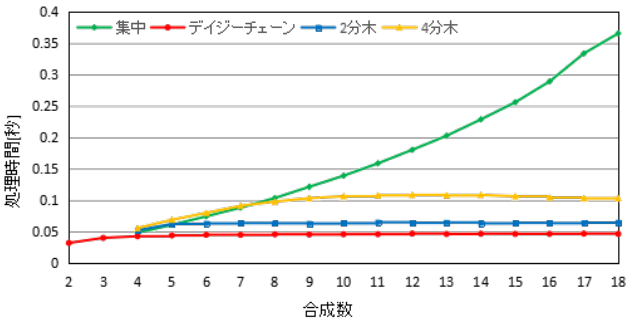

図 17 映像の合成処理にかかる時間

では，合成数の増加に伴いほぼ一定の割合で処理時間も増加している．2 分木および 4 分木では，特定のタイミングで処理時間が増加し，それ以外では合成数が増えても処理時間はほぼ変化していない．処理時間が増加するタイミングに注目すると，2 分木では合成数が 7 から 8，15 から 16 に増える時となっており，4 分木では合成数が 5 から 6 に増える時となっていることが分かる．これは，ちょうど木の高さが高くなるタイミングに等しい．木の高さが高くなると通信が 1 ホップ増えるので，その影響で処理時間が増加すると言える．また，木の高さが変化しない限りは処理時間の変化もないということになるが，1 ノードあたりが処理すべき合成映像数は（自身の映像も含めて），2 分木では最大でも 3，4 分木では最大でも 5 であり，合成処理は並列的に行われるためであると考えられる．さらに，デイジーチェーン，2 分木，4 分木では，処理時間が増加するときにはすべてにおいて約 0.1 秒の増加がみられることから，末端（合成処理を行わずに自身の映像の送信のみを行うノード）から最終的な合成映像を生成するノードまでのホップ数が 1 増加するごとに処理時間が約 0.1 秒増加すると考えられる．一方，集中合成では合成数の増加によるホップ数の変化はなく，代わりに特定の 1 ノードが処理すべき映像数が増加することになる．

以上の結果より，処理時間が増加する要因として，集中合成では映像の合成処理にかかる時間が大きく，分散合成ではノード間の通信等にかかる時間が大きくなっていると考えられる．これを確認するため，映像の合成処理にかかる時間のみに注目した結果を図 17 に示す．

集中合成では，全ての映像を集約させているノードに注目し，分散合成では木構造の根に当たるノードに注目している．図より，集中合成では合成数が増加するほど処理時間が増加していることが分かり，このグラフの形状は図 16 の集中合成と同じような形となっている．よって，集中合成では，映像の合成処理にかかる時間が全体の処理時間に大きく影響していると言える．デイジーチェーン，2 分木については，ほぼ変化が見られない．しかし，4 分木については，合成数が 4 から 9 までの間で処理時間の増加がみられる．これは，4 章で述べた実装方法が関係していると

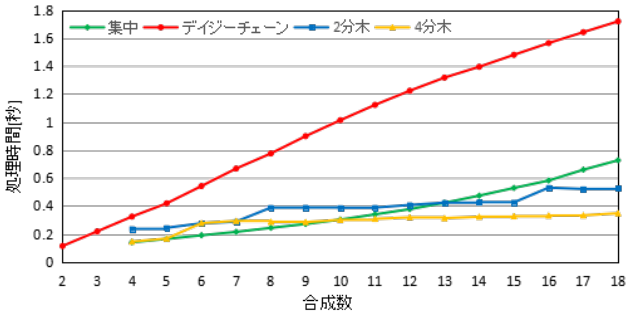

図 18　処理負荷を大きくした実験結果

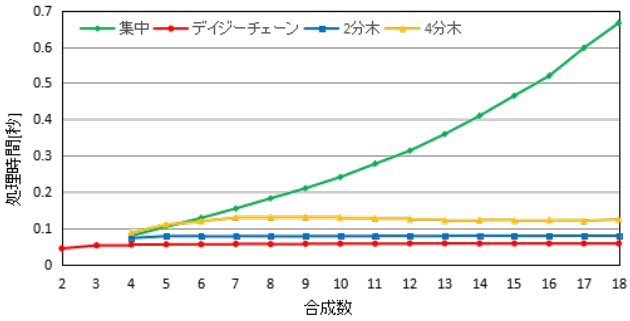

図 19　処理負荷を大きくした場合の映像の合成処理にかかる時間

考えられる．4 分木の場合，合成数 5 までは集中合成と同じような形となり，中継ノードが存在しない．そのため，受信した映像をリサイズしてから合成することになる．合成数が 6 以上になると，根ノードは孫を持つようになり，中継ノードが存在する．中継ノードから転送された映像はリサイズすることなくそのまま合成するため，合成数 5 までとは異なる処理をすることになる．4 つの子の内，合成数 6 では 4 つが中継ノード，合成数 7 では 2 つが中継ノード..... というようになっているため，合成数 9 までは処理時間の変化があったと考えられる．しかし，その変化も約 0.05 秒程度であるので，分散合成では，映像の合成処理にかかる時間が全体の処理時間に及ぼす影響は小さいと言える．

図 18 に，映像を拡大する処理を追加して処理負荷を大きくした実験結果を示す．図 16 と同様に，横軸に合成数，縦軸に処理時間をとっており，10 回の計測の平均値をとっている．図 16 の結果と比較する．各合成方法における合成数と処理時間の増加の関係は同様である．また，各合成方法を比較すると，分散合成より集中合成の方が優れているのは合成数が 9 までとなっており，合成数 14 以降では 2 分木よりも長い処理時間がかかる結果となった．こちらについても，映像の合成処理のみにかかる時間に注目し，その結果を図 19 に示す．

全体の傾向としては，図 17 と変わらないが，集中合成にかかる時間が大幅に増加していることが分かる．合成数が 4 から 18 に増加した時に，どれだけ処理時間が長くなった

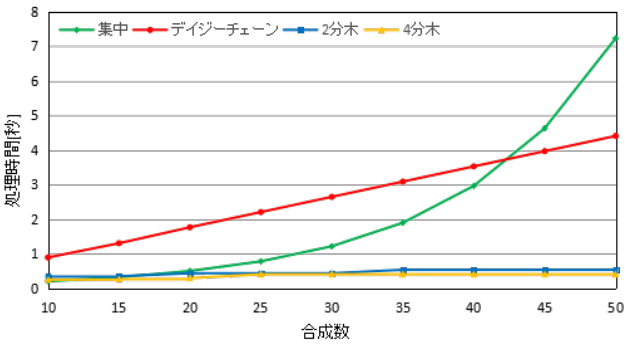

図 20　合成数が増加した場合の処理時間の予測

のかを見ると，図 17 では約+0.317 秒であったのに対し，図 19 では約+0.586 秒となっている．しかし，分散合成ではどちらの場合もほぼ映像の合成処理にかかる時間の増加がないことが分かる．

以上の結果より，映像の合成処理にかかる負荷が大きくなるほど，集中合成より分散合成の方が低遅延に合成するために効果的であることが分かった．また，本実験では合成数が最大でも 18 までの結果しか得られなかったが，合成数がさらに増加した場合にも，今回得られたような処理時間の増加傾向が続くとすれば，図 20 のような結果が得られると予測される．合成数 20 程度では，集中合成と分散合成の違いがあまり大きく見られなかったが，30 や 40 にもなると差が顕著に表れると考えられる．

6.　まとめと今後の課題

本研究では，ライブ放送において全ての撮影対象がまるで同じ空間に存在するかのような同世界放送を提案した．同世界放送では，視聴者と配信者双方にとって快適なものにするため，多数のリアルタイム映像を低遅延に収集し処理する必要がある．その際，特定のノードに処理負荷や通信負荷が集中し遅延が大きくなることを避けるため，映像を中継ノード上で合成しつつ収集する方法を考案した．同世界放送を簡易的に実装し，LAN 環境やテストベッド環境で 1 フレーム当たりの合成にかかる時間を計測したところ，合成すべき映像数や各映像に対する処理量が多く，ネットワーク帯域が十分に広くない環境においては，2 分木や 4 分木を構成し処理を分散させることで，より短時間で同世界映像を生成できることを確認した．

しかし，本稿では集中合成とデイジーチェーン，2 分木，4 分木の 3 パターンに限った分散合成の違いを確認したに過ぎない．そのため，2 分木などの単純な構造以外での分散方法を検討する必要がある．また，実験結果で述べた「1 ノードあたりの処理負荷が大きい」ことや「ネットワーク帯域が広くない」ことなどは，使用するノードや通信状況によって異なる．よって，仮に 2 分木を構成する場合でも，どのノードをどこに配置すべきかを考慮する必要があ

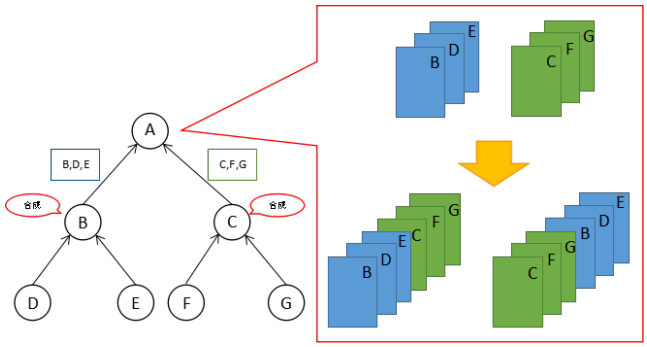

図 21 映像の収集順と重なり順の関係

る．例えば，処理性能が高くネットワーク帯域も十分であるノードを節に，そうでないノードは葉に配置させるというのが単純な考えである．しかし，同世界映像を生成する際には，映像を収集する順によって表示される重なり順が制限されるという問題がある．図 21 は映像の収集順と重なり順の関係を示している．

図では，B，C で 3 つの映像を合成し，その映像を転送先の A で合成することを考えている．B では背面から順に E，D，B と重なった映像を，C では背面から順に G，F，C と重なった映像を転送した場合，A での合成は G，F，C，E，D，B または E，D，B，G，F，C の順に限られる．よって，例えば B，C，D，E，F，G という順に表示したければ，木構造において C と E の位置を入れ替えるといったことを考えなければならない．ただし，中継ノードで重なり順を考慮して，合成することなく別の映像として次のノードに転送することを考えれば，必ずしも転送順に制限がかかるわけではない．しかし，いずれの場合も重なり順を把握することが必要となり，重なり順については同世界放送中に各ユーザによって変更されることが想定され，また参加者が増減することも起こり得るので，映像の収集木を動的に構成することが必要となる．

本稿では評価していなかった「表示される映像間で同期がとれているか」も重要な点であり，その評価も行う必要がある．また，実験結果では，合成数が増加するほど分散合成の方が処理時間が短く優れているという結果が得られたが，合成映像の品質に問題があった．分散合成ではホップ数が増加するため映像の圧縮が繰り返し行われることとなり，根に近いノードの撮影対象が鮮明であり葉に近いノードほど粗くなる．その結果，生成される同世界映像内に表示されている映像ごとに品質が異なるという問題点がある．一方で，集中合成は常に 1 ホップであるため，全ての映像が同じ品質となっており見栄えが良い．よって，映像の品質についても考慮する必要があると考える．

謝辞

本研究の一部は G-7 奨学財団研究開発助成事業および福井大学研究育成経費，JSPS 科研費 18K11316 の助成による成果である．

参考文献

[1] 河合豊明. オンライン授業の取り組み. 新地理, Vol. 68, No. 2, pp. 13–16, 2020.

[2] 小豆川裕子. BCP とテレワーク：業務を継続するための環境整備. 情報の科学と技術, Vol. 70, No. 9, pp. 447–451, 2020.

[3] NHK. 中止の”すずめ踊り”合成動画作りました. https://www.nhk.or.jp/sendai-blog/social/429730.html.（参照 2021-02-14）.

[4] 萩原健. コロナ禍を受けたオンライン（と）演劇、その展開. 演劇学論集 日本演劇学会紀要, Vol. 71, pp. 35–50, 2020.

[5] 高田英明. イマーシブテレプレゼンス技術 “Kirari!”. 日本画像学会誌, Vol. 56, No. 4, pp. 366–373, 2017.

[6] 阿久津明人, 南憲一, 日高浩太. 超高臨場感通信技術 Kirari! Beyond 2020. NTT 技術ジャーナル, Vol. 30, No. 10, pp. 12–15, October 2018.

[7] 山口徹也, 佐藤孝子, 小野正人, 外村喜秀, 難波功次, 菊地由実, 星出高秀, 森住俊美, 小野朗, 南憲一. 高臨場感ライブビューイングサービスのためのサラウンド映像合成・同期伝送システム. 映像情報メディア学会誌, Vol. 74, No. 2, February 2020.

[8] 佐藤孝子, 難波功次, 小野正人, 菊池由実, 山口徹也, 小野朗. 競技空間全体の高臨場ライブ中継に向けたサラウンド映像合成・同期伝送技術. NTT 技術ジャーナル, Vol. 29, No. 10, pp. 19–23, October 2017.

[9] 野中敬介, 渡邊良亮, 塚本航平. 5G 網を利用した自由視点映像リアルタイム配信技術. 映像情報メディア学会誌, Vol. 74, No. 1, pp. 180–186, 2020.

[10] 小谷和也, 中村正人, 木谷友哉, 孫為華, 柴田直樹, 安本慶一, 伊藤実. 複数のカメラ映像の合成によるリアルタイム鳥瞰映像提示システム. マルチメディア通信と分散処理ワークショップ論文集, pp. 109–110, September 2009.

[11] 加藤大一郎, 武藤一利, 三ッ峰秀樹. ハイブリッドセンサを用いたハンディカメラによるバーチャルスタジオの開発と実用化. 映像情報メディア学会誌, Vol. 72, No. 1, pp. J13–J22, 2018.

[12] Mohammad Keshavarzi, Allen Y. Yang, Woojin Ko, and Luisa Caldas. Optimization and manipulation of contextual mutual spaces for multi-user virtual and augmented reality interaction. *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, March 2020.

[13] 田中一晶, 加藤良治, 中西英之. 鏡型ビデオ会議における視触覚相互作用によるソーシャルテレプレゼンスの強化. 情報処理学会論文誌, Vol. 58, No. 5, pp. 946–954, May 2017.

[14] Tomislav Pejsa, Julian Kantor, Hrvoje Benko, Eyal Ofek, and Andrew Wilson. Room2Room: Enabling life-size telepresence in a projected augmented reality environment. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW 2016)*, pp. 1716–1725, February 2016.

[15] Jaron Lanier. How to get the most from together mode. https://techcommunity.microsoft.com/t5/microsoft-teams-blog/how-to-get-the-most-from-together-mode/ba-p/1509496.（参照 2021-02-14）.

[16] Apple Inc., Jean-Francois M Albouze, and Santa Cruz. Generating synthetic group selfies, US pat. 10,672,167, 2020.

[17] 牧田航輝, 川上朋也, 松本哲, 義久智樹, 寺西裕一, 下條真司. リアルタイム映像の収集と合成を伴う同世界放送システムの検討. 第 28 回情報処理学会マルチメディア通信と

分散処理ワークショップ (DPSWS2020) 論文集, デモ発表, pp. 186–192, November 2020.
[18] 森田順也, 岩井儀雄, 谷内田正彦. 室内における背景と物体の分離. Technical Report 34(2002-CVIM-133), 大阪大学基礎工学研究科, 大阪大学基礎工学研究科, 大阪大学基礎工学研究科, May 2002.
[19] 柿沼弘員, 長尾慈郎, 宮下広夢, 外村喜秀, 長田秀信, 日高浩太. 機械学習を用いた任意背景リアルタイム被写体抽出技術. NTT 技術ジャーナル, Vol. 30, No. 10, pp. 16–20, October 2018.
[20] Soumyadip Sengupta, Vivek Jayaram, Brian Curless, Steve Seitz, and Ira Kemelmacher-Shlizerman. Background matting: The world is your green screen, 2020.
[21] FFmpeg. https://ffmpeg.org/. （参照 2021-02-14）.
[22] OpenCV. https://opencv.org/. （参照 2021-02-14）.
[23] Pillow. https://python-pillow.org/. （参照 2021-02-14）.
[24] 寺西裕一, 齋藤祐貴, 室野栄, 西永望. 大規模 IoT サービスの実証を可能とするオープンテストベッド JOSE. 情報通信研究機構研究報告, Vol. 61, No. 2, pp. 145–153, 2015.