拡散モデルを用いた食事変換 AR

唐澤 香梨菜[†1,a)]　柳井 啓司[†1,b)]

概要：食欲は人間の最大の欲求の一つであり，食事制限や病気により食べられないものが存在する場合，生活の質である QOL が低下してしまう．一方で，味は視覚にも影響されるクロスモーダル効果が発生する．この効果を利用し，食事変換 AR により味を操作することで，食事制限をしている人でも好きなものが食べられるようになる．従来の食事変換 AR は GAN を使用しており，画像の品質の低さが課題として残った．そこで本研究では AR 食事体験における没入感を高めるため，食事領域をセグメントし，よりリアルな画像変換が可能な拡散モデルを使用することで没入感のある食事変換 AR を作成することを目的とする．システムとして，Unity と GPU サーバ間で通信を行い，セグメントには DeepLabV3，変換には StreamDiffusion を用いた．実験では変換結果や実行時間，プロンプト，マスク拡張の効果を検証し，従来より柔軟な食事変換が可能となった．

1. はじめに

食欲は生命維持に不可欠であり，好物や美味しいもの，贅沢品を味わうことは貴重な体験である．しかし，食物アレルギーや病気，食事制限などにより望む食事ができない状況では，生活の質（QOL）が低下してしまう．

一方で，味覚は視覚，聴覚，嗅覚，触覚などの感覚にも影響されるクロスモーダル効果が発生する．これを利用し食べ物の見た目を変えることで味覚をある程度操作することが可能である．またディープラーニングを用いた画像変換の手法について，以前は敵対的生成ネットワーク（GAN）が利用されていたが，現在はより高品質な画像を生成できる拡散モデルも多く使用されている．拡散モデルは GAN と比較して生成に時間がかかるという課題があるが，近年は高速化も進んでおり，リアルタイムに近い速度で生成できるモデル StreamDiffusion [1] も登場している．

先行研究である DeepTaste [2] では VR ゴーグルから取得した画像を StarGAN [3] で高速画像変換し，それをヘッドマウントディスプレイ（HMD）に表示させることで，クロスモーダル効果により味覚を操作する食事変換 AR を作成した．一方で本研究では GAN の代わりに拡散モデルと StreamDiffusion を使用することで，高品質な画像を生成する食事変換 AR の作成を目的とする．

[†1] 現在，電気通信大学
Presently with The University of Electro-Communications, Chofu, Tokyo 182-8585, Japan
[a)] karasawa-k@mm.inf.uec.ac.jp
[b)] yanai@cs.uec.ac.jp

2. 関連研究

2.1 食事 AR

プロジェクションマッピングを食事に投影する研究 [4] では適切な彩度と部分的な色の変更，ハイライトを追加することで食品をよりおいしそうに見せることができた．また FoodChangeLens [5] では，HoloLens と CNN(Conditional CycleGAN) [6] を利用し，実際の形状を考慮した 10 種類の食事カテゴリ変換 AR を作成した．しかしこれらは対象となる食事への彩度や明度の微調整，食事のカテゴリが変わった場合には追加学習が必要である．さらに実際に食べるなどして食品の形が変わった場合は対応できず，静的な変換を行うものであった．

Meta Cookie+ [7] では，AR マーカを印刷したプレーンクッキーをチョコや紅茶クッキーに変換し，さらに嗅覚ディスプレイにより匂いを提示する疑似味覚ディスプレイを提案した．クッキーの形に合わせて変換でき，被験者の 80%がクッキーの味が変化したと答えたが，AR マーカが印刷されたクッキーしか変換することができなかった．

2.2 拡散モデル（Diffusion Model）

拡散モデルとは，画像に徐々にノイズを加えていく拡散過程を学習することで，ノイズを除去しながらデータを修正していく逆拡散過程を通し画像を生成する学習モデルである．高品質な画像を生成できる点で，GAN や VAE などの生成モデルより優れているが，逆拡散過程において何度もノイズ除去処理を行うため生成に時間がかかるという課

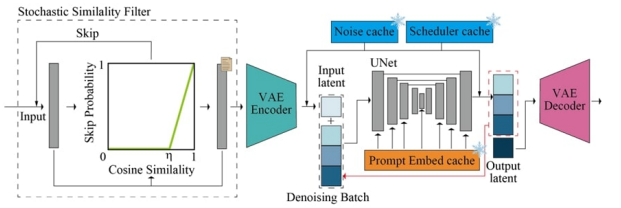


図 1: StreamDiffusion の推論概要図 (図は [1] から引用)

題がある．

2.3 StreamDiffusion

近年，拡散モデルの生成を速める Latent Consistency Models（LCM） [8] や Latent Denoising Diffusion GAN（LDDGAN） [9] などの研究が盛んに行われており，本研究ではそのうちのひとつである StreamDiffusion [1] を使用する．StreamDiffusion とは Hugging Face が開発した拡散モデルライブラリ Diffusers[*1]内の StableDiffusionPipeline モデルをサポートしたリアルタイム拡散パイプラインである．StreamDiffusion の概要図を図 1 に示す．

StreamDiffusion の戦略の一つである StreamBatch とは，データ効率化のため拡散モデルにおけるノイズ除去をバッチ処理することである．拡散モデルはノイズ除去を繰り返すことで画像を生成するため，ステップ数と U-Net の処理時間が比例する．しかし高品質な画像を生成するためステップ数を減らすことは避けたい．そこで逐次的なデノイズ処理をバッチ処理に再構成することで，各ノイズ除去ステップの後に次の入力画像を受け入れ処理している．このようにすることでステップ数と U-net の処理時間が比例することなく，待機時間を最小限に抑え高速な処理を可能にした．

また入力したプロンプトの効果を高めるため，プロンプトとネガティブプロンプトのベクトルの差を用いる分類器フリーガイダンス（Classifier-Free Guidance，CFG） [10] という手法がある．この CFG ではネガティブプロンプトによる負の条件付き残差ノイズを計算するため，各入力の潜在変数と負の条件付き埋め込みを推論時に U-Net に渡す必要がある．これに対し，StreamDiffusion では Residual Classifier-Free Guidance（RCFG）という手法を新たに提案した．従来の CFG の計算量が $2n$ 回であるのに対し，提案した Self-Negative RCFG と Onetime-Negative RCFG はそれぞれ n 回と $n+1$ 回で済み，計算量を減らすことができる．

さらに連続入力画像の類似度を計算し，類似度が高い場合は処理をスキップする確率的類似フィルタ（Stochastic Similarity Filter，SSF）の使用や，前もってサンプリングしたガウスノイズやプロンプト埋め込みをキャッシュに保存することで，高速処理のためのキャッシュ戦略を最適化する．

以上の戦略により，StreamDiffusion では拡散モデルを用いながらも高速で画像を生成することができる．

[*1] https://huggingface.co/docs/diffusers/ja/index

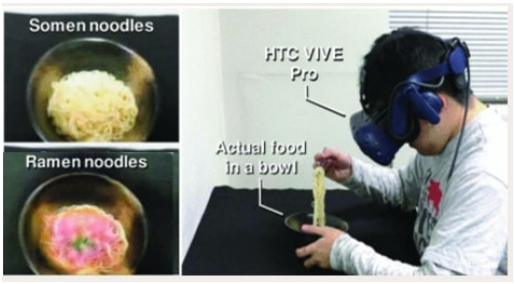


図 2: DeepTaste の入出力画像と使用例 (図は [2] から引用)

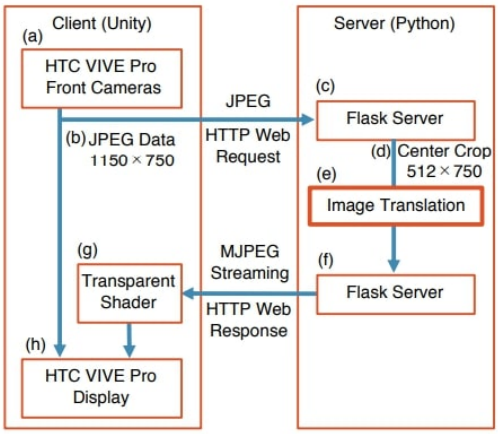


図 3: DeepTaste のシステム構想図 (図は [2] から引用)

2.4 DeepLabV3

セマンティックセグメンテーションに Deep Convolutional Neural Networks（DCNNs）を適用した場合，従来は二つの課題があった．一つ目はプーリング層や畳み込みストライドによる空間解像度の低下であり，DeepLabV3 [11] ではダウンサンプリング層をアップサンプリング層に変更することで対処した．二つ目は物体の大きさの多様性により特徴抽出が困難である点で，これには異なる間隔でアトラス畳み込みを適用する Atrous Spatial Pyramid Pooling（ASPP）を使用した．これらにより DeepLabV3 では高精度かつ効率的なセグメンテーションを実現した．

本研究では食事領域をセグメンテーションに，食事データでファインチューニングした DeepLabV3 を使用する．

3. 既存システム:DeepTaste

本研究のもととなった DeepTaste [2] では，StarGAN [3] とクライアントーサーバ通信により動的に食品の外観を変更できる味覚操作 AR システムを作成した．このシステムの構想図を図 3 に示す．

まず，クライアントである Unity で HTC VIVE Pro のフロントカメラからの画像を取得し，その画像を HTTP

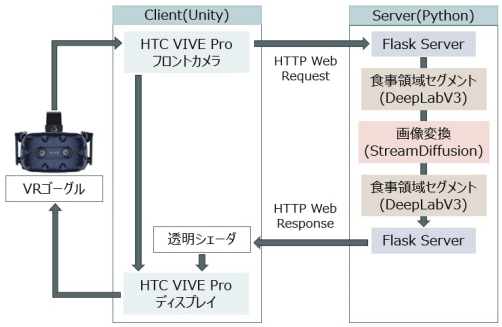


図 4: 提案システムの概要図

Web Request によりノート PC 上に立てた Python の Flask サーバに送信する．サーバでは StarGAN により画像変換を行い，変換した部分を入力画像と合成する．合成画像は HTTP Web Response によりクライアント側に送信され，透明シェーダでグラデーション上に透明度が調整される．それを最初にフロントカメラで取得した画像と合わせ，HTC VIVE Pro のディスプレイに表示する仕組みである．

クライアントは HTC VIVE Pro を接続したノート PC 上で動作する．HMD の左右のフロントカメラは 30 fps で 1150×750 の画像をそれぞれキャプチャする．しかしサーバで左右の 1150×650 サイズの画像を 30 fps で処理すると動作が遅くなってしまうため，左側の画像のみを JPEG 形式で 6 fps でサーバに送信している．変換画像を受け取った後，Unity では透明シェーダを利用して元画像と合成される．

画像変換では UECFOOD-100 [12] と Twitter（現 X）から収集した追加画像で学習した StarGAN を使用し，10 種類の食事に変換が可能である．

この研究では味覚を操作できた一方で，システム遅延や変換後の画像の品質の低さ，食事の種類が限られているなどの課題が残った．

4. 提案システム:DeepTasteDM

本研究の提案システムの 概要図を図 4 に示す．先行研究の DeepTaste [2] と基本設計は同様であるが，サーバはノート PC に搭載されている GPU ではなくポートフォワーディングを用いて接続された GPU 上に構築され，食事領域をセグメンテーションし拡散モデルで画像変換を行う点が異なる．

本研究のサーバでの一連の処理の例を図 5 に示す．ここで，s とはセグメンテーション時の画像サイズである．

まず取得した画像に対し，DeepLabV3 [11] で食事領域のセグメンテーションを行う．その結果に基づき，食事領域はフロントカメラから取得した画像，背景は黒となる画像を生成する．この生成画像を StreamDiffusion の入力画像とすることで，食事領域のみの変換を行う．そして変換画像に対し再度 DeepLabV3 で食事領域のセグメンテーションを行い，フロントカメラから取得した画像と $\alpha = 0.6$ でアルファブレンディングを行う．ここでアルファブレンディングを採用した理由は，$\alpha = 1.0$ の場合は変換結果の彩度が高く馴染まなかったためであり，$\alpha = 0.6$ という数字は経験的な調整結果である．

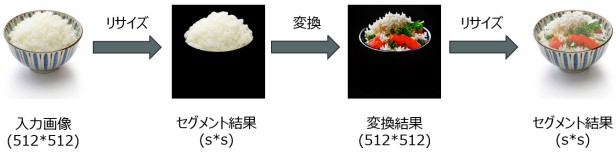


図 5: サーバで行うセグメンテーションと画像変換の処理の例

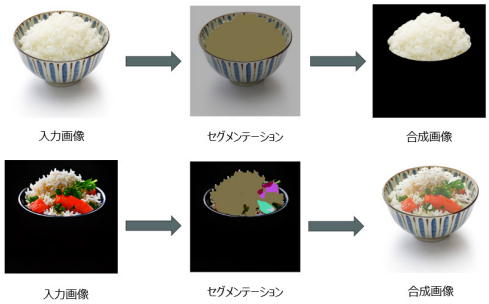


図 6: セグメンテーション結果に基づくマスク画像の生成方法

このように食事領域のみの画像変換を行うことで背景や食器は影響を受けず，没入感を高めることができる．

4.1 セグメンテーション

セグメンテーションに基づく合成画像の生成方法を図 6 に示す．食事領域のセグメンテーションを行う際に利用する DeepLabV3 はセマンティックセグメンテーションを行うモデルであり，その結果に基づいて画像を合成する際はピクセル毎の処理を行う．当然画像サイズが大きくなれば処理するピクセルの数も多くなるため、処理に時間がかかってしまう．そこで本研究では図 5 に示すように，サイズ $s \times s$ $(s = 128, 256, 512)$ にリサイズした取得画像をセグメンテーションし，その画像を StreamDiffusion で変換することで 512×512 の画像を生成し，再度 $s \times s$ にリサイズすることでセグメンテーションを行っている．

4.2 使用するモデル

DeepTaste では StarGAN を用いていたが，変換後の画像の品質が低いという問題があった．そこで本研究では StreamDiffusion でサポートされている StableDiffusion-Pipeline モデルのうち，リアルな画像を生成するため今回の食事変換 AR の目的に適していると考えた Stability AI の SD-Turbo と Stable Diffusion v1.5 [13] を使用する．

(a) 入力画像　(b) [10，15]　(c) [32，45]

図 7: t_index_list の違いによる変換結果の影響

4.3 StreamDiffusion のパラメータ

StreamDiffusion のパラメータの一つである t_index_list では推論に使用されるパイプラインの数，深さを指定する．パイプラインの数が多いほど品質は良くなるが，実行に時間がかかる．またパイプラインが浅いほどプロンプトの影響が強くなるが入力画像から形が変わりやすく，深いほど入力画像に近いがプロンプトを反映しない結果となる．モデルを SD v1.5，プロンプトを “seafood rice bowl，slices of salmon on rice，delicious，high quality”，ネガティブプロンプトを “bad quality，blurry” とし，t_index_list を [10，15] と [32，45] とした時の例を図 7 に示す．

図 7（b）では海鮮丼らしく刺身のようなものが現れているが画像が少しぼやけており，茶碗の柄が消えてしまっている．一方で図 7（c）では茶碗の柄がそのままで白米の粒感が残っているものの，プロンプトの影響はなく刺身らしきものは見えない．このように変換結果は t_index_list の値に大きく影響されており，本研究では経験的な調整により [10，30] と指定する．

5. 実験

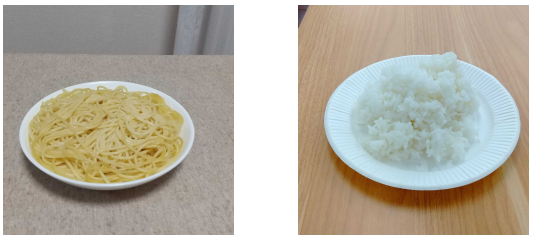
実験として実行時間を計測し，モデルによる違い，プロンプトによる違い，セグメンテーション時のマスク拡張による違いを調べた．

5.1 画像の変換結果

DeepTaste による StarGAN と StreamDiffusion による SD-Turbo，Stable Diffusion v1.5 で入力を図 8 とした時の食事を変換した結果を図 9 に示す．StarGAN の画像生成サイズは 256 × 512 であったため，食事を中心に切り出している．SD-Turbo と SD v1.5 は 256 × 256 の出力サイズそのままである．そのため，同じような位置から撮影しても StreamDiffusion での結果は食事が小さく写ってしまっている．また SD-Turbo と SD v1.5 のプロンプトは “spaghetti bolognese”，“fried rice”，ネガティブプロンプトは “bad quality，blurry”，t_index_list = [10，30] である．

5.2 実行時間

変換時のモデル，セグメンテーション時の画像サイズを変更した場合の処理時間の平均を表 1 に示す．セグメント 1 とは背景を黒にする処理，セグメント 2 は変換結果と元画像を合成する処理，全体とは画像を受け取ってから送り返すまでの時間である．



(a) 麺類の入力画像　(b) ご飯類の入力画像

図 8: 食事変換時の入力画像

さらにクライアントにおいて，画像を取得してから変換画像を表示するまでの時間であるレイテンシを計測した結果は表 2 のようになった．計測を開始してからサーバとの通信が始まるまで時間がかかってしまうため，初めのデータは取り除く必要がある．しかしサーバとの通信が始まるまでの時間は決まっておらず，取り除くべき値を判断する条件が難しかったため，最後から 50 個のデータを使用し平均レイテンシを計算した．

5.3 プロンプトによる違い

SD-Turbo に対し画像サイズ 512 × 512 で醤油ラーメン，豚骨ラーメン，黒ラーメンとラーメンの種類を指定した結果を図 10 に，DeepTaste には無い食事の変換結果を図 13 に示す．ただし，図 13 の麺の入力画像は図 11，食パンの入力画像は図 12 である．

5.4 マスク拡張による違い

今まで食事領域のセグメンテーション結果に対し合成画像を生成し，StreamDiffusion の入力としていた．しかしそれでは変換した際に形が変わってしまい，元画像の食事領域より変換画像の食事領域が小さくなってしまっていた．そこで，StreamDiffusion に入力する合成画像の食事領域を広げるマスク拡張を行った．その概要を図 14 に示す．

まず入力画像に対しセグメンテーションを行い，その結果を二値化する．それを OpenCV の findContours 関数を使用して輪郭を検出しマスクを拡張している．この拡張したマスク結果を入力画像と合成することで，食事領域より広い部分まで StreamDiffusion の入力としている．このようにマスクを拡張した場合と，食事領域を入力した場合での生成結果の比較を図 15，16 に示す．なおマスク拡張なしの画像は図 9 と同じであり，マスク拡張した際の麺の入力画像は図 11 である．

またマスク拡張を行った際のサーバでの処理平均時間を表 3 に示す．

図 9: 各モデルの食事変換結果

表 1: モデル・画像サイズ・GPU の違いによるサーバでの処理平均時間

モデル	サイズ	セグメント 1(s)	変換 (s)	セグメント 2(s)	全体 (s)
SD-Turbo	128	0.0248	0.0941	0.0229	0.1419
SD-Turbo	256	0.0277	0.0923	0.0249	0.1453
SD-Turbo	512	0.0464	0.0874	0.0475	0.1828
SD v1.5	128	0.0256	0.0966	0.0228	0.1452
SD v1.5	256	0.0269	0.0979	0.0245	0.1497
SD v1.5	512	0.0462	0.0887	0.0471	0.1835
StarGAN			0.0153		0.0161

表 2: モデル・GPU の違いよるレイテンシ平均時間

モデル	サイズ	レイテンシ (s)
SD-Turbo	256	0.3164
SD v1.5	256	0.3063
StarGAN		0.2162

6. 考察

6.1 実験結果について

画像の変換結果について，DeepTaste の StarGAN では食事に色が付く一方，本研究は材料の質感や具材が現れる事でよりリアルな結果となった．また DeepTaste ではカメラから取得した画像の中心を切り取り StarGAN への入力としているため，机や壁なども色が変わってしまっていた．しかし本研究ではセグメンテーションを実行したため，壁や机を変更することなく食事領域のみ変換することができた．

実行時間について，NVIDIA GeForce RTX A4000 を使用した GPU サーバ上で StarGAN を動かした時の処理速度を 1.0 とした時の SD-Turbo，SD v1.5 の時の処理速度

(a) “shoyu ramen”

(b) “tonkotsu ramen”

(c) “black ramen”

図 10: ラーメンの種類を指定した結果

表 3: マスク拡張した場合のサーバでの処理平均時間

モデル	サイズ	セグメント 1（s）	変換（s）	セグメント 2（s）	全体（s）
SD-Turbo	256	0.0302	0.0947	0.0216	0.1470
SD v1.5	256	0.0299	0.1003	0.0213	0.1520

図 11: 麺類の二つ目の入力画像

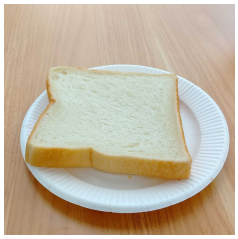


図 12: 食パンの入力画像

表 4: モデル・画像サイズ・GPU の違いによるサーバでの処理速度比

モデル	サイズ	変換	全体
SD-Turbo	128	6.15	8.81
SD-Turbo	256	6.03	9.02
SD-Turbo	512	5.71	11.4
SD v1.5	128	6.31	9.02
SD v1.5	256	6.49	9.30
SD v1.5	512	5.80	11.4
StarGAN(3060)		3.94	4.25
StarGAN(4700)		1.0	1.0

比，また DeepTaste で使用された NVIDIA GeForce RTX 3060 で StarGAN を実行した時の処理速度比を表 4 に示す．

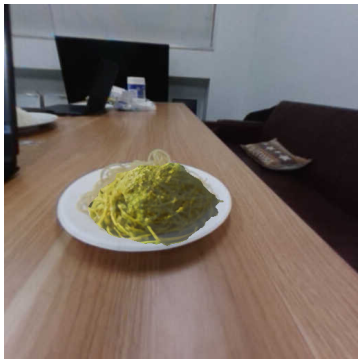
この表 4 の StarGAN（3060）と StarGAN（4700）より，GPU による処理速度は約 4 倍異なることが分かった．それを踏まえたうえで同じ GPU 上で実行している StarGAN（4700）と SD-Turbo，SD v1.5 を比較すると，変換に約 6 倍，全体の処理としては 9〜11 倍の時間がかかっており，GAN と拡散モデルの生成速度の違いが顕著に表れる結果となった．表 1 から SD-Turbo，SD v1.5 のどちらにおいても入力画像サイズに関わらず画像変換にかかる時間は 0.09 秒の約 11 fps であった．またセグメンテーションについては画像サイズ 512 の時は約 0.05 秒かかるのに対し，画像サイズ 128，256 の時はどちらも約 0.03 秒であり画像サイズと処理時間が比例する結果とはならなかった．

一方でレイテンシについて，表 1 の全体の処理にかかった時間と表 2 のレイテンシとの差を調べると SD-Turbo は 0.1711 秒，SD v1.5 は 0.1566 秒，StarGAN（3060）は 0.1478 秒，StarGAN（4700）は 0.1463 秒であった．このことからサーバを立てる場所に関わらず画像の送信などにかかる時間は約 0.15 秒であり，FPS を向上させるにはサーバでの処理時間を短くすることが重要であると考えられる．

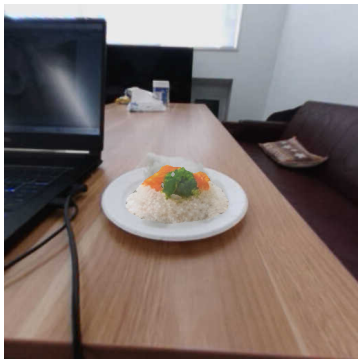
プロンプトを変更した結果について，ミートスパゲッティである “spaghetti bolognese” や炒飯である “fried rice” と入力した場合はそれらの料理らしい変換結果を得られた．また DeepTaste ではラーメンに変換する際，1 種類のラーメンしか変換できなかったが，本研究では醤油ラーメン，豚骨ラーメン，ブラックラーメンなどラーメンの味を変更することができた．さらに麺やご飯のみだけでなく，食パンに対しても食事変換することができた．これらの結果から，より柔軟な食事変換が可能となった．

マスク拡張を行った場合は，マスク拡張を行わなかった場合よりも食事領域全体に対して変換結果を合成することができた．表 1 と表 3 よりマスク拡張にかかる時間は約 0.003 秒であり，少ない実行時間で変換結果を向上させることができた．しかし図 15 の curry のように食事領域よりも広く変換してしまったり，図 16 の fried rice のように箸らしきものが映ってしまったりという課題も残った．

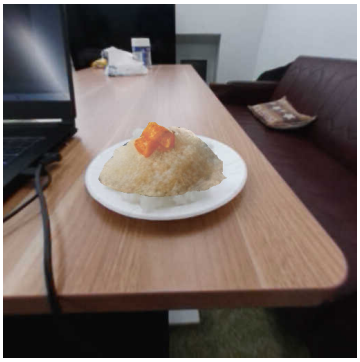
(a) “white, spaghetti alla carbonara, guanciale”



(b) “Genoese pesto pasta”



(c) “seafood rice bowl, many slices of salmon on rice, delicious, high quality”



(d) “chestnut rice”

(e) “pizza toast”

(f) “strawberry paste on toast”

図 13: DeepTaste にはない食事の変換結果

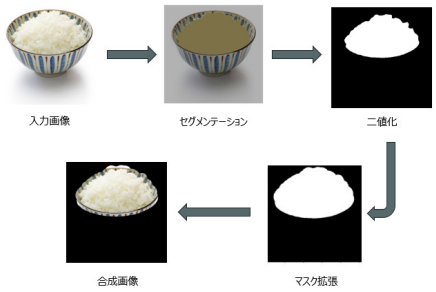


図 14: マスク拡張の概要

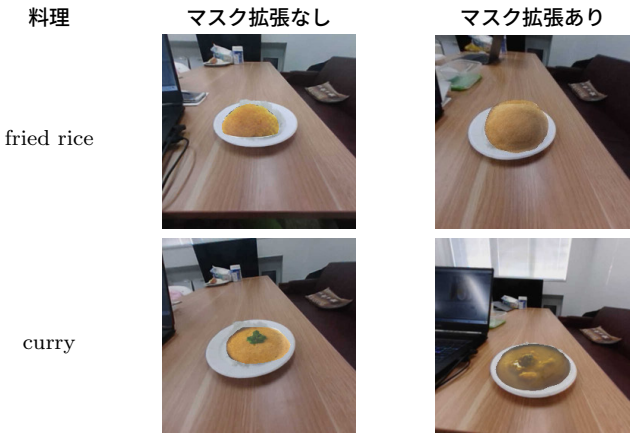


図 15: SD-Turbo におけるマスク拡張有無の違い (ご飯類)

図 16: SD v1.5 におけるマスク拡張有無の違い (ご飯類)

6.2 課題

実際に食事を持ち上げた際，持ち上げた部分に対し変換がうまくいった場合とうまくいかなかった場合があった．その例を図 17 に示す．DeepTaste ではフロントカメラから取得した画像をそのまま入力しているため，食事を動かしてもその形に合わせて変換することができた．しかし本研究ではセグメンテーションを行うため，食事が細かくなってしまうとうまく変換されないという事が分かった．

また本研究では変換画像の一貫性を考慮していないため，少しでも動くと同じ種類ではあるが全く別の料理が表示さ

(a) セグメンテーション失敗例

(b) セグメンテーション成功例

図 17: 食事を持ち上げた際のセグメンテーション失敗例と成功例

図 18: 連続入力画像に対する変換結果

れる問題があることが分かった．例として SD v1.5 でマスク拡張ありでカレーに変換した結果を図 18 に示す．左上が最初の変換結果，右下が最後の変換結果である．

このように食事に対して一貫性がなく，これは AR 食事体験における没入感を損なう大きな要因となる．そのため，拡散モデルに入力する際の初期ノイズを固定するのなどの解決策が必要である．

7. おわりに

本研究では，クライアントーサーバ通信を用いて，フロントカメラから取得した画像を食事領域のみ拡散モデルでリアルタイム画像変換し，それをディスプレイに表示するシステムの構想を説明した．食事領域のセグメンテーションには DeepLabV3 を，拡散モデルには StreamDiffusion を使用することで高速画像変換を行った．このように食事領域のみ変換することで没入感を高めることができ，また拡散モデルの text-to-image モデルを使用することで，学習時の食事の種類に制限されることのない柔軟な変換が可能になった．先行研究である DeepTaste の StarGAN に比べ画像変換処理に約 2 倍の時間がかかるものの，ノート PC の StarGAN と GPU サーバの StreamDiffusion のレイテンシの差は 0.1 秒であった．また食事領域のみを StreamDiffusion の入力とすると，入力画像に比べ変換画像の食事領域が小さくなってしまうという課題があったが，マスクを拡張することで効果的に対処することができた．

しかし食事を持ち上げた際にセグメンテーションがうまくいかず変換されない，少しでも動くと別の料理が表示されてしまうという課題も残った．これらは食事変換 AR において重大な課題である．今後はさらなる画像変換の高速化やセグメントの精度向上，食事のフレーム間での一貫性の制御を目指す．

参考文献

[1] Kodaira, A., Xu, C., Hazama, T., Yoshimoto, T., Ohno, K., Mitsuhori, S., Sugano, S., Cho, H., Liu, Z. and Keutzer, K.: StreamDiffusion : A Pipeline-level Solution for Real-time Interactive Generation, *arXiv:2312.12491* (2023).

[2] Nakano, K., Horita, D., Sakata, N., Kiyokawa, K., Yanai, K. and Narumi, T.: DeepTaste : Augmented Reality Gustatory Manipulation with GAN-Based Real-Time Food-to-Food Translation, *Proc. of 2019 IEEE International Symposium on Mixed and Augmented Reality*, pp. 212–223 (2019).

[3] Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S. and Choo, J.: StarGAN : Unified generative adversarial networks for multi-domain image-to-image translation, *Proc. of IEEE Computer Vision and Pattern Recognition*, pp. 8789–8797 (2018).

[4] Fujimoto, Y.: Projection Mapping for Enhancing the Perceived Deliciousness of Food, *IEEE Access*, Vol. 6, pp. 59975–59985 (2018).

[5] Naritomi, S., Tanno, R., Ege, T. and Yanai, K.: FoodChangeLens: CNN -based food transformation on HoloLens, *Proc. of IEEE International Conference on Artificial Intelligence and Virtual Reality*, pp. 197–199 (2018).

[6] Mirza, M. and Osindero, S.: Conditional Generative Adversarial Nets, *arXiv:1411.1784* (2014).

[7] Narumi, T., Nishizaka, S., Kajinami, T., Tanikawa, T. and Hirose, M.: Meta cookie+: an illusion-based gustatory display, *Virtual and Mixed Reality-New Trends: International Conference, Virtual and Mixed Reality 2011, Held as Part of HCI International*, pp. 260–269 (2011).

[8] Luo, S., Tan, Y., Huang, L., Li, J. and Zhao, H.: Latent consistency models: Synthesizing high-resolution images with few-step inference, *arXiv:2310.04378* (2023).

[9] Trinh, L. T. and Hamagami, T.: Latent Denoising Diffusion GAN: Faster sampling, Higher image quality, *IEEE Access*, Vol. 12, pp. 78161–78172 (2024).

[10] Ho, J. and Salimans, T.: Classifier-free diffusion guidance, *arXiv:2207.12598* (2022).

[11] Chen, L.-C.: Rethinking atrous convolution for semantic image segmentation, *arXiv:1706.05587* (2017).

[12] Matsuda, Y., Hoashi, H. and Yanai, K.: Recognition of multiple-food images by detecting candidate regions, *IEEE International Conference on Multimedia and Expo*, pp. 25–30 (2012).

[13] Rombach, R., Blattmann, A., Lorenz, D., Esser, P. and Ommer, B.: High-Resolution Image Synthesis With Latent Diffusion Models, *Proc. of IEEE Computer Vision and Pattern Recognition*, pp. 10684–10695 (2022).