

## 概要

既存の AR アプリケーションの多くは物体のない平面を前提として設計されている。そのため、実物体と同じ位置に仮想物体を配置すると、本来見えるべきではない実物体の一部が見えてしまい、適切な結果を得ることができない。実物体が視覚的に消去され、仮想物体のみが表示される状態が理想的である。そこで本研究では、一般物体認識と画像処理を AR 技術と組み合わせることで、実物体が視覚的に消去され、実物体と同じカテゴリの仮想物体のみが表示されるシステムの開発を行った。本システムは、一般物体認識で除去対象のカテゴリと領域を取得し、画像処理で対象を消去した画像を AR の背景として用い、認識したカテゴリと同じカテゴリの仮想物体を配置するというものである。本システムは、スマートフォン 1 台で動作する。

# 目 次

<b>第 1 章 はじめに</b>	<b>1</b>
1.1 研究背景 . . . . .	1
1.2 DR 技術 . . . . .	1
1.3 関連研究 . . . . .	2
1.4 研究目標 . . . . .	3
<b>第 2 章 準備</b>	<b>4</b>
2.1 本論文で使用する用語 . . . . .	4
2.2 YOLOv3-Tiny のアルゴリズム . . . . .	5
2.3 Inpaintng 手法の事前実験 . . . . .	7
<b>第 3 章 提案手法</b>	<b>9</b>
3.1 ARCore を用いたカメラ画像の取得 . . . . .	10
3.1.1 RGB 色空間への変換 . . . . .	10
3.1.2 画像の転置 . . . . .	10
3.2 一般物体認識 . . . . .	11
3.3 Inpainting . . . . .	11
3.4 表示する仮想物体の決定 . . . . .	12
<b>第 4 章 実験と考察</b>	<b>13</b>
4.1 使用した仮想物体 . . . . .	13
4.2 実行結果 . . . . .	14
4.3 一般物体認識のメリット . . . . .	16
4.3.1 AR 表現の拡大 . . . . .	17
4.3.2 シーンの変化への対応 . . . . .	17
4.4 うまく動作しない場合の原因 . . . . .	17
4.4.1 物体の検出に失敗した . . . . .	18
4.4.2 バウンディングボックスが物体を完全に囲っていない . . . . .	18
4.4.3 背景が複雑 . . . . .	18
<b>第 5 章 まとめ</b>	<b>22</b>

# 第1章 はじめに

## 1.1 研究背景

近年、AR(Augmented Reality)の技術の発展が著しい。モバイルデバイスの発達に伴い、AR技術を用いたアプリケーションの需要は多分野において高まっている。ビジネスの分野でAR技術を用いたアプリケーションの1つが、インテリアの仮想モデルを配置し、購入を検討できるものである。これは、スマートフォンで平面を検出し、検出した平面に家具などの3次元モデルを配置することができる。そのため、製品カタログを眺めながら実際に配置した状況を想像するよりも、AR技術を用いた方が直感的である。

しかし、現在のAR技術では、大きさや形状の異なる既存物体と仮想物体を重なるように配置した場合、本来見えるべきではない既存物体の一部が見えてしまい、配置後の適切な検討ができないという問題がある。既存物体が視覚的に消去され、仮想物体のみが表示されることが理想的である。この問題は、配置場所を既存物体のない平面を前提として設計されているために発生する。

## 1.2 DR技術

この問題を解決する技術として、画像から既存物体を視覚的に消去する技術である Diminished Reality(DR) が着目されている。DR の手法には、静止画像中の除去対象に背景画像を合成する方法や、あらかじめ撮影された動画像に対して背景画像の合成を行うプリレンダリング方式による方法、リアルタイム動画像に対応したリアルタイムレンダリング方式による方法など様々な手法が存在する。リアルタイムレンダリング方式の DR 手法は、背景画像として使用する仮想世界の作成方法で大きく 2種類に分類できる。仮想世界をリアルタイムに作成しながら DR シミュレーションを行う手法と、DR シミュレーションを行う前に仮想世界を作成する手法である。

リアルタイムに背景画像として使用する仮想世界を作成する DR 手法としては、マーカーと複数台のカメラを使用する手法がある。しかし、本研究ではスマートフォンで動作することを目的としているため、カメラを複数台使用する方法は適さない。DR シミュレーションを行う環境の仮想世界をあらかじめ作成する手法としては、対象物のない状態の環境を深度センサなどを用いてあらかじめ作成しておき、シミュレーション時に背景として合成する方法がある。しかし、この方法は対象物の状況を作りだせる場合に限定されるため、対象物を移動させることができない場合には使えない。また、正確な仮想世界を作成するためには深度センサなどの特殊な装置が必要であるため、手軽な手法とはいえない。

そのほかのリアルタイムレンダリング方式の DR 手法として、除去対象の周辺画像から推定し、除去対象領域を埋めること方法がある。欠損した画像情報を周囲から内挿法などを用いて修復する画像処理手法は、一般に「Inpainting」と呼ばれている。この手法であれば、使用するカメラは

1台で済む。また、仮想世界をあらかじめ作成する必要はないため、何らかの原因で仮想世界があらかじめ作成できない場合でも、対応することができる。

### 1.3 関連研究

Inpaintingを用いたDR手法の先行研究として、Kawaiらの研究[2]がある。Kawaiらは、vSLAMからの点群を用いてシーンを複数の平面に分割し、それぞれの平面にアルゴリズムを適応し、物体を視覚的に消去する手法を提案した。シーンの空間構造を考慮して辺面を分割し、平面ごとにInpaintingを行うため、物体で隠れている平面の境界を再現できる。この手法では、除去対象領域のマスクをフレームごとに3次元的にトラッキングすることで作成している。そのため2次元画像を用いて除去対象をトラッキングするよりも堅牢である。しかし、トラッキングを開始するためにはユーザが初めに除去対象領域を指定する必要がある。そのため、動作中に消去対象を増やす場合や移動させる場合には対応することができない。

スマートフォンをデバイスとして用いたDR手法の先行研究として、Queguinerらの研究[3]がある。Queguinerらの手法は、DRシミュレーションを開始する前に対象物のない状態の3次元空間モデルを用意する。事前に3次元空間モデルを用意することで、DRシミュレーション時の環境光の変化や仮想物体を配置したときのオクルージョンに対応した。しかし、3次元空間モデルを作成する際に、詳細な3Dモデルが必要なため深度センサを用いている。そのためスマートフォン1台のみで実行することができない。

これらの先行研究は、いずれもリアルタイム動画像に対応したものである。先行研究から、本研究ではスマートフォン1台で動作可能であり、事前に3次元空間の取得を必要としないInpaintingを用いた手法に着目した。



図 1.1: 文献 [2] から引用



図 1.2: 文献 [3] から引用

## 1.4 研究目標

本研究の目的は、既存の実物体と仮想物体を同じ位置に重なるように配置した場合に、既存の実物体の一部が見えてしまうことを解決することである。この問題を一般物体認識と画像処理をARの背景動画像に応用することで解決する。DRの手法にあるように除去対象をトラッキングするのではなく、一般物体認識で除去対象のカテゴリと座標を取得する。そのため、除去対象と同カテゴリの仮想物体で置換することが可能になる。

既存の実物体の一部が見えてしまう問題の解決と物体の置換を同時に行えるため、ARの表現の拡大につながる。さらに、スマートフォン1台のみで動作するシステムを開発することで、手軽に導入できるシステムになる。

# 第2章 準備

本章では、本論文で使用している用語などについて説明する。まず2.1節で本論文で使用する用語について述べる。次に2.2節で一般物体認識で使用するYOLOv3-Tinyのアルゴリズムを述べる。最後に2.3節で除去対象領域を修復する処理において、本研究で使用する手法を決定するために行った事前実験について述べる。本論文で使用する用語について述べる。

## 2.1 本論文で使用する用語

### Unity

Unit社が開発したマルチプラットフォーム対応のゲームエンジンおよび開発環境。本研究ではUnity2018.3を使用した。

### ARCore

Google社が提供するARプラットフォーム。一般的なRGBカメラを搭載したスマートフォンを用いて、マーカレスARを実現することができる。主要機能は以下の通りである。

- 自己位置推定 (Motion Tracking)
- 平面認識 (Environmental Understanding)
- 明るさ推定 (Light Estimation)
- マーカー認識 (Augmented Image)
- 空間共有 (Cloud Anchor)
- 顔認識 (Augmented Faces)

### YOLO(You Only Look Once)

You Only Look Once(YOLO)[1]は、リアルタイムにバウンディングボックスとクラス確率を求める物体認識手法の1つである。既存の手法であったDPM(Deformable Part Model)やR-CNNは画像の領域推定と分類が別々であったため処理時間が長くなる問題があった。そこで、YOLOは画像の領域推定と分類を同時にを行うことでリアルタイムによる処理を実現した。画像全体を見て予測することができること、1つのネットワークで処理が完結することが特徴である。従来の手法は、物体候補をある程度探したうえでニューラルネットワークによる特徴抽出を行っていたため、背景を物体として誤検出してしまう欠点があった。一方、YOLOは画像の全範囲を学習に利用するため対象物体周辺の様子も同時に学習することができる。そのため、誤検出を従来手法であるFast R-CNNの半分以下に抑えている。本研究では、一般物体認識の処理で使用した。

## YUV 色空間

YUV 色空間は、輝度信号 Y と 2 つの色差信号 U(Cb), V(Cr) で表現される映像信号用の色空間である。色差とは、RGB の各色から輝度成分の Y を差し引いた信号のことである。U は Y と青色成分との差であり、V は Y と赤色成分との差である。この色空間は輝度と色差を使って色を表現する。これは、人の目は明るさには敏感であるが、色には敏感でないという、人の視覚特性を利用している。

## Google Pixel3a

Google 社が販売するスマートフォン。CPU は Snapdragon 670 であるため、2019 年 12 月時点で発売されている Android スマートフォンの中では中位の性能である。本研究でシステムの実行デバイスとして使用した。

## OpenCV

オープンソースのコンピュータ・ビジョン・ライブラリである。画像処理やコンピュータビジョン、汎用的な数学処理や機械学習に関するアルゴリズムが多数含まれている。BSD ライセンスであるため、学術用途だけでなく商用目的でも利用することができる。利用可能なプラットフォームは、Windows, Linux, Mac, IOS, Android などで、C, C++, Python, Java などの言語から利用できる。

## Inpainting

意図せずに写りこんでしまった物体など画像内の不要部分を除去し、除去した領域を違和感なく自動的に修復すること。

## 2.2 YOLOv3-Tiny のアルゴリズム

YOLOv3-Tiny は YOLO を改良した YOLOv3 において速度を重視したアルゴリズムである。YOLOv3 よりも検出精度は落ちるが、非常に高速で動作する。

YOLOv3-Tiny では、まずははじめに Darknet19 という独自の特微量検出器を用いて、入力画像の特微量を抽出する。特微量抽出後、入力画像を  $N \times N$  に分割し 3 D テンソルによって物体の検出を行う。これにより入力画像中の小さな物体も検出が可能になった。対象物体を検出する手順を以下に示す。

1. 入力画像を  $N \times N$  の領域に分割
2. 領域内の物体のクラス確率を計算
3. 領域ごとに予測したバウンディングボックスの信頼度とパラメータ  $(x, y, h, w)$  を計算
4. 物体のクラス確率とバウンディングボックスの信頼度の積が閾値以上であれば、そのバウンティボックスを検出結果とする

パラメータ  $(x, y)$  はバウンティボックスの中心座標であり、 $(h, w)$  はバウンティボックスの高さと幅である。検出するクラスの数を  $C$  とすると、バウンティボックスの信頼度とパラメータ  $(x, y, h, w)$  を 2 つのスケールで予測する。そのため  $N \times N \times (5 \times (2 + C))$  の 3 D テンソルを作成する。

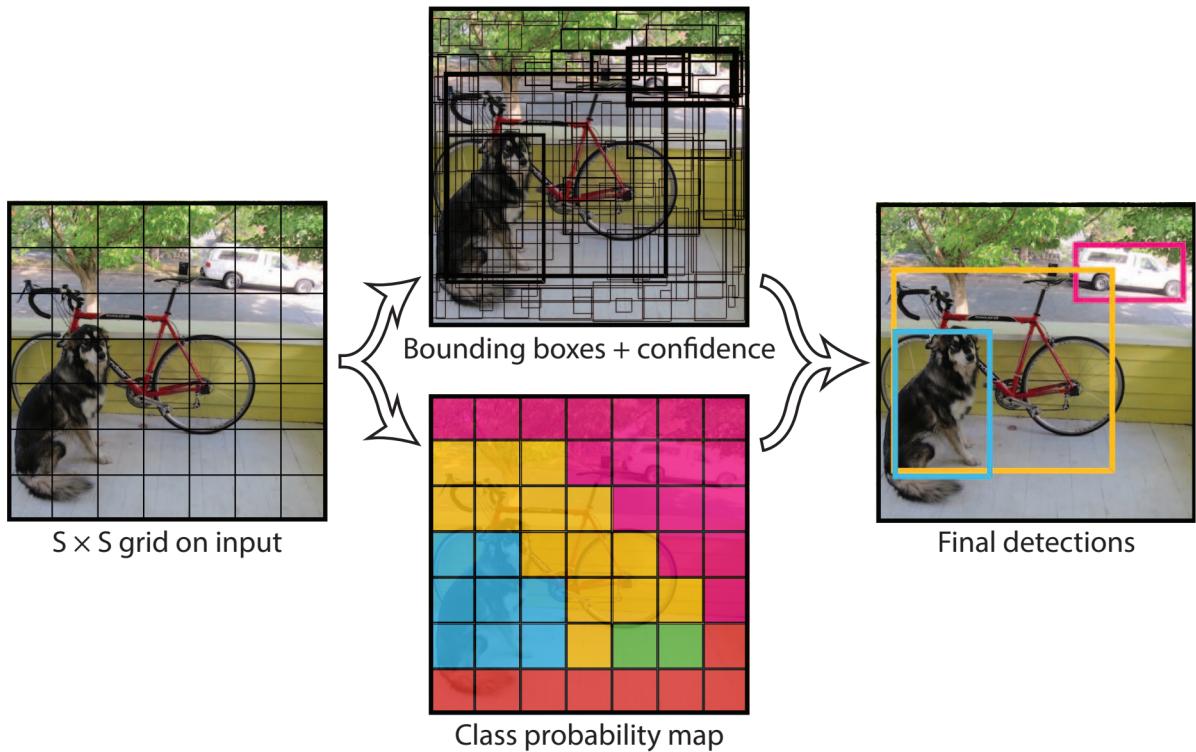


図 2.1: 物体検出手順 (文献 [1] から引用)

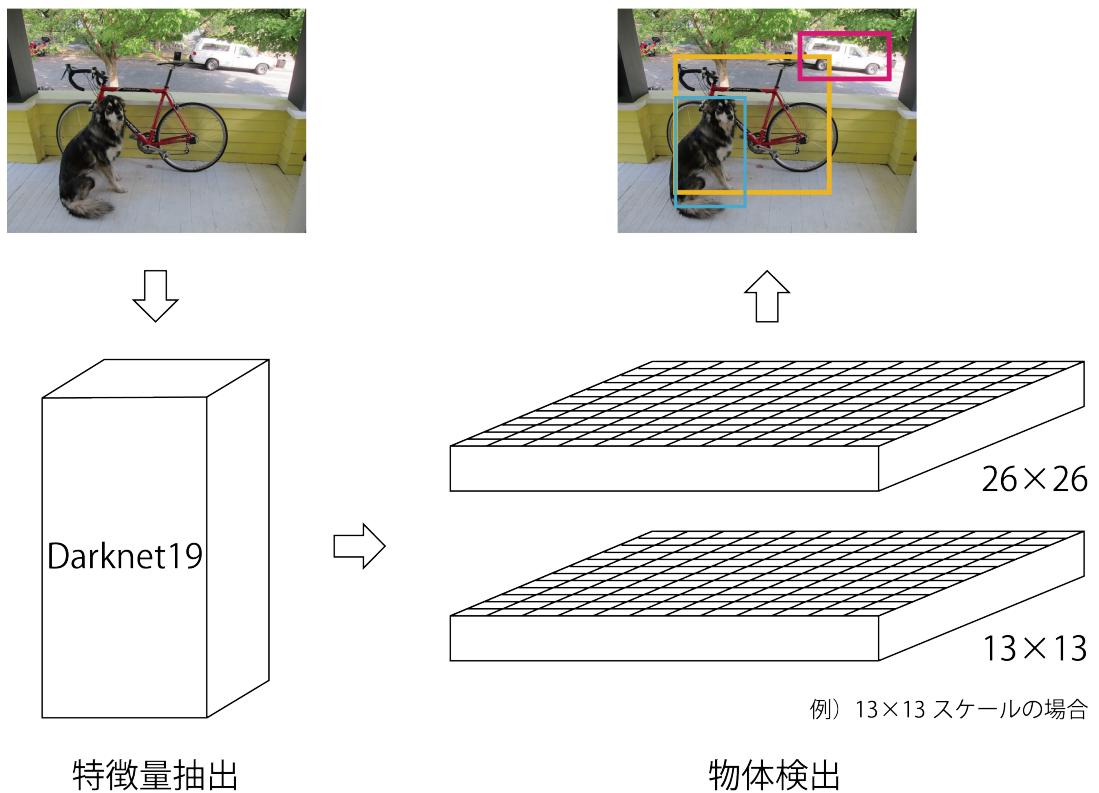


図 2.2: YOLOv3-Tiny のネットワーク構造

## 2.3 Inpaintng 手法の事前実験

次章で述べる提案手法において利用した Inpainting 手法を決定するために行った事前実験について述べる。

Inpainting の手法は、深層学習を使わない手法と深層学習を使う手法の 2 つに大別することができる。近年は深層学習を使用する手法が主流であり、深層学習を使わない場合に比べて修復精度が高い。しかし、深層学習を使う場合は学習用データの用意や学習用データのクレンジング、学習に時間がかかる。また、修復精度は学習に用いたデータに依存する欠点がある。深層学習を使わない場合、修復精度は入力画像に依存する。

そこで本研究では深層学習を用いない Inpainting の手法を持った。AR は室内だけでなく屋外など様々な環境で使用するため、深層学習を用いる手法ではそれに対応できる学習用データを準備することが困難であるためだ。様々な深層学習を用いない手法が提案されているが、本システムはスマートフォンで動作することを考慮し、RGB カメラ 1 つで実行でき実行速度が速い手法が望ましい。すなわち、入力画像が 1 枚の RGB 画像で実行できることが条件となる。

実験に用いた手法は、上記の条件を満たす [4][5][6][7] の手法である。これらは OpenCV とその拡張モジュールである OpenCV-Contrib に実装されている。そこで OpenCV を OpenCV-Contrib とともにビルドをし、4 つの手法を OpenCV の Inpaint() から呼び出して比較実験を行った。実験に使った PC は「OS:Windows10Pro 64bit」、「CPU:Intel(R) Core(TM) i7-6600U」、「RAM:16.0GB」である。

以下のコードで実行した。

---

```
1 cv::inpaint(input, mask, output, 2, cv::INPAINT_NS);
2 cv::inpaint(input, mask, output, 2, cv::INPAINT_TELEA);
3 xphoto::inpaint(input, mask, output, xphoto::INPAINT_SHIFTMAP);
4 ft::inpaint(input, mask, output, 3, ft::LINEAR, ft::MULTI_STEP);
```

---

入力画像のサイズは  $640 \times 480$  pixel である。修復する領域を指定するマスクは、物体全体を囲うような矩形のマスクを作成した。矩形のマスクを用いた理由は、本システムでは一般物体認識における認識結果のバウンディングボックスを除去対象領域とするためである。入力画像とマスクを図 2.3、実行結果を図 2.4 に示す。



図 2.3: 入力画像とマスク

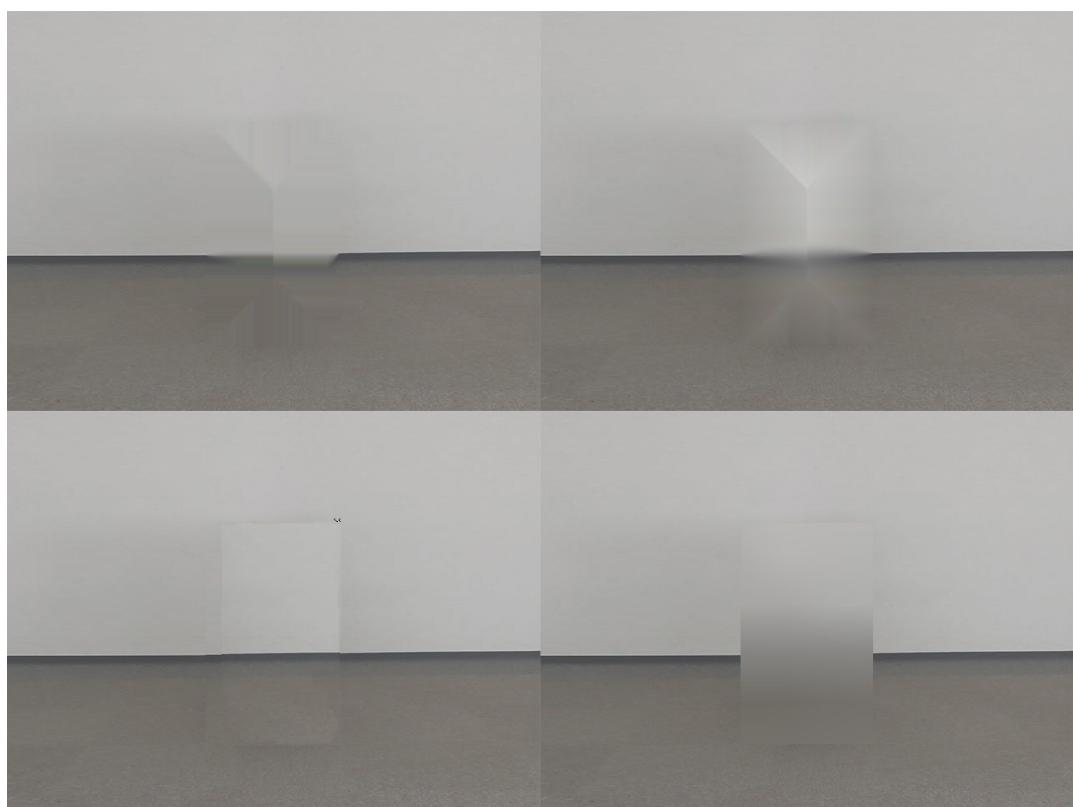


図 2.4: 左上 [4]:61.93ms, 右上 [5]:80.83ms, 左下 [6]:6379.01ms, 右下 [7]:958.76ms

## 第3章 提案手法

本章では、提案する AR システムの各段階で実行する処理について述べる。スマートフォンのリアカメラから動画像を取得し、フレームごとに一般物体認識と対象の除去を行う。また、一般物体認識で複数個の物体を検出した場合、仮想物体で置換するためにどの実物体が選択されているかを判断する必要がある。

カメラ画像の取得から AR の背景画像の作成と仮想物体での置換までの処理の流れを図 3.1 に示す。

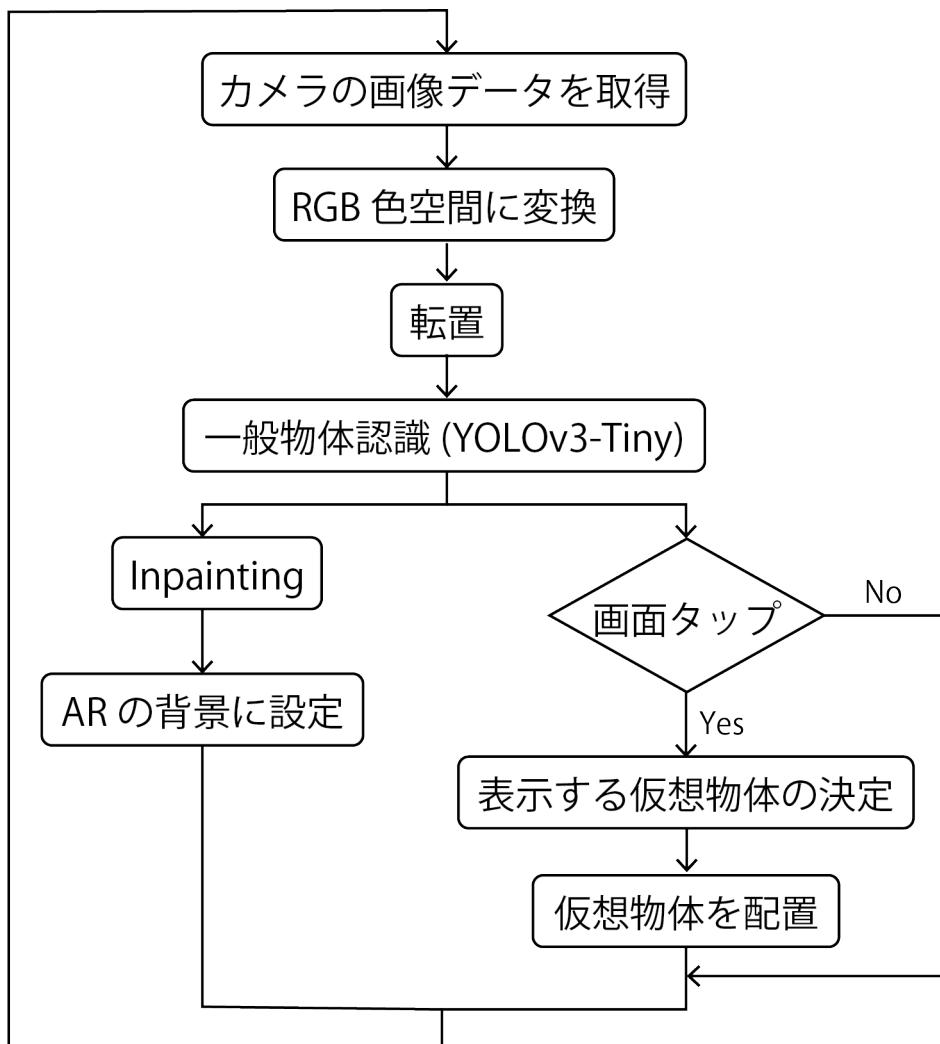


図 3.1: 処理の流れ

### 3.1 ARCore を用いたカメラ画像の取得

カメラ画像の取得には、ARCore の `AquireCameraImageBytes` メソッドを用いた。 `AquireCameraImageBytes` メソッドは戻り値として解像度が  $640 \times 480$  pixel のカメラ画像データを取得できる。このデータは左上を中心で転置された画像の YUV-420-888 形式の byte 配列である。このままでは次節で述べる一般物体認識のアルゴリズム YOLOv3-Tiny に入力として使用することができないため、入力できる形式に変換する必要がある。

本節では、取得したカメラ画像のデータを一般物体認識のアルゴリズムで入力として使用する画像データに変換する手法について述べる。

#### 3.1.1 RGB 色空間への変換

YUV 色空間から RGB 色空間への変換は、データ範囲が

$$\begin{cases} 0 \leq Y \leq 255 \\ -128 \leq U, V \leq 127 \\ 0 \leq R, G, B \leq 255 \end{cases}$$

のとき、ITU-R BT.601 規定では次の式で行う。

$$\begin{cases} R = 1.000Y + 1.402V \\ G = 1.000Y - 0.344U - 0.714V \\ B = 1.000Y + 1.772U \end{cases} \quad (3.1)$$

YUV420 フォーマットは、Y 成分の個数に対して U, V 成分がそれぞれ 4 分の 1 個であるフォーマットである。また、YUV-420-888 形式は色情報の格納方式により I420, NV21, NV12 に分かれる。色情報の格納方式はデバイスに依存するため、プログラム中に色情報の格納方式を調べる処理を追加し、それぞれに対応した変換処理を用意する必要がある。本研究で使用した Android スマートフォン Google Pixel3a の格納方式は NV21 であった。

NV21 のデータ構造を図 3.2 に示す。色が同じのメモリの Y, U, V の値を式 3.1 に代入する。

#### 3.1.2 画像の転置

一般に、一般物体認識のアルゴリズムへの入力画像は回転していない画像を用いる。回転させた画像を入力として与えた場合、認識性能が本来の性能よりも低下する可能性がある。そこで一般物体認識のモデル本来の性能を引き出すためには、事前に回転を修正する必要がある。

`AquireCameraImageBytes` メソッドで取得したデータは左上を中心で転置したものである。よって式 3.2 より、再度左上を中心で転置すればよい。

$$(A^T)^T = A \quad (3.2)$$

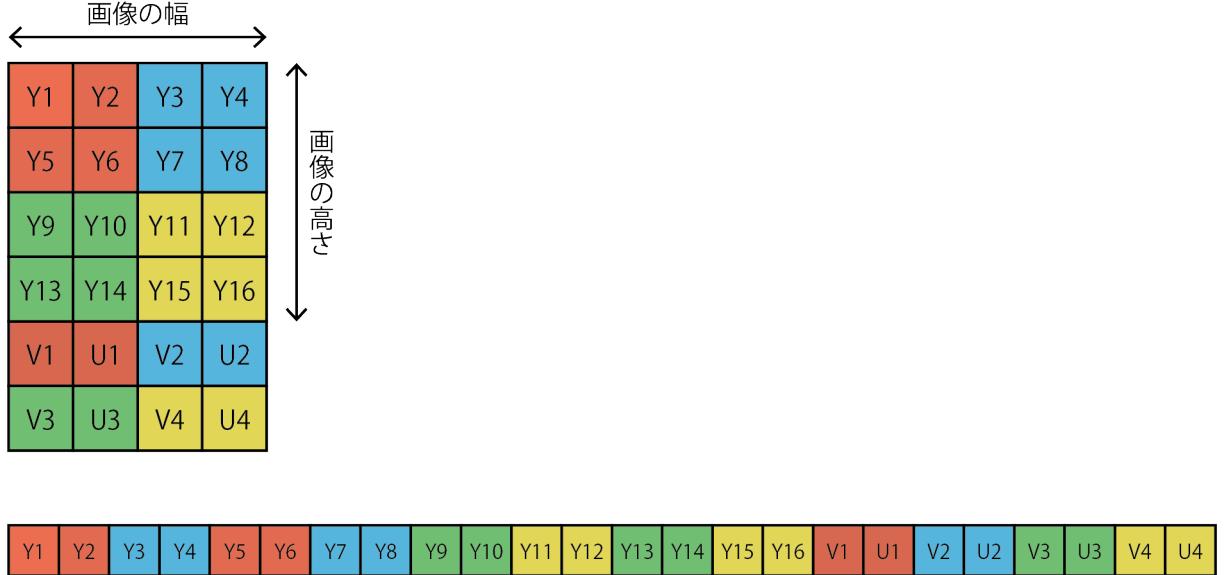


図 3.2: NV21 のデータ構造

### 3.2 一般物体認識

YOLOv3-Tiny を使用する場合, YOLOv3-Tiny のネットワークモデルデータ, そのモデルデータに対応する重み付けデータ, 検出したいクラスの名前データの 3 つが必要である. 本研究では, 公式サイトで配布されている学習済みのモデルデータである `yolov3-tiny.cfg` を用いた. `yolov3-tiny.cfg` は 80 個のクラスを識別できる.

YOLOv3-Tiny に前節で作成した RGB 画像を入力することで, 物体のクラス確率とバウンディングボックスの信頼度の積が閾値以上のバウンディングボックスの 4 つのパラメータとそのクラスを得る. バウンディングボックスの 4 つのパラメータは, (中心の  $x$  座標, 中心の  $y$  座標, 幅, 高さ) である. 本システムでは, バウンディングボックスの 4 つのパラメータを Inpainting の処理で使用し, クラス名を表示する仮想物体の決定する処理に使用する.

本研究では OpenCV の DNN モジュールから YOLOv3-Tiny を利用した. OpenCV の DNN モジュールを使用した場合, 4 つのパラメータは入力画像の大きさで正規化された値で出力される.

### 3.3 Inpainting

本システムでは, 2.3 節で述べた Inpainting 手法の事前実験の結果, 最も実行速度が速い手法 [4] を用いた. 手法 [4] は, Inapinting の基本的な手法である Navier-Stokes 方程式をベースとした手法を用いる. これは欠損領域の境界部分の輝度値が滑らかに連続することで違和感のない画像が生成されるという考え方に基づき, 欠損領域内の画素値を周囲から補間することで修復を行う手法である.

本システムでの欠損領域は, 3.2 節で取得したバウンディングボックスである.

### 3.4 表示する仮想物体の決定

本システムは画面をタップした位置にある実物体と同じカテゴリの仮想物体を置換する。一般物体認識で検出した置換対象が1つであれば、置換対象と同じカテゴリの仮想で置換すればよい。しかし、置換対象を複数個検出した場合、タップした位置にある実物体はどの仮想物体で置換しなければならないか判断する必要がある。タップした位置にある実物体を置換する仮想物体を決定する手順を以下に示す。

1. 各バウンディングボックスの中心座標をスマートフォンのスクリーンサイズに対応した座標に変換
2. タップした画面の座標を取得
3. 各バウンディングボックスの中心座標とタップした座標の距離を計算
4. 距離が最短であるバウンディングボックスのカテゴリに対応する仮想物体で置換する

# 第4章 実験と考察

本章では、提案手法に基づき開発したシステムを実行させた結果の記述と考察を行う。

## 4.1 使用した仮想物体

実験で使用した仮想物体を図 4.1 に示す。



©2017 Google



©2017 Google

図 4.1: 使用した仮想物体

## 4.2 実行結果

本システムを実行した結果を示す。ノートパソコンを置換した結果を図 4.2、イスを置換した結果を図 4.3、複数個の物体を置換した結果を図 4.4 に示す。各図の左の画像は従来の AR 技術を用いて仮想物体を実物体と重なるように配置した状態を示している。右の画像は本システムによって置換した結果を示している。

除去対象領域の背景が複雑でなく、一般物体認識で検出したバウンディングボックスが物体全体を囲うことに成功した場合、本システムはうまく動作する。本システムがうまく動作しない場合の原因と考察は 4.4 節で述べる。



図 4.2: ノートパソコンを置換した場合

壁と接した机の上に、ノートパソコンが一台あるシーンで実験を行った。従来の AR 技術（左の画像）では、現実のノートパソコンの上部と左部の一部が見えている。本システム（右の画像）では、現実のノートパソコンは消去され仮想物体のノートパソコンのみが表示されている。



図 4.3: イスを置換した場合

背景は壁と床のみで、イスが一つあるシーンで実験を行った。従来の AR 技術（左の画像）では、現実のイスの下部と右部の一部が見えている。本システム（右の画像）では、現実のイスは消去され仮想物体のイスのみが表示されている。

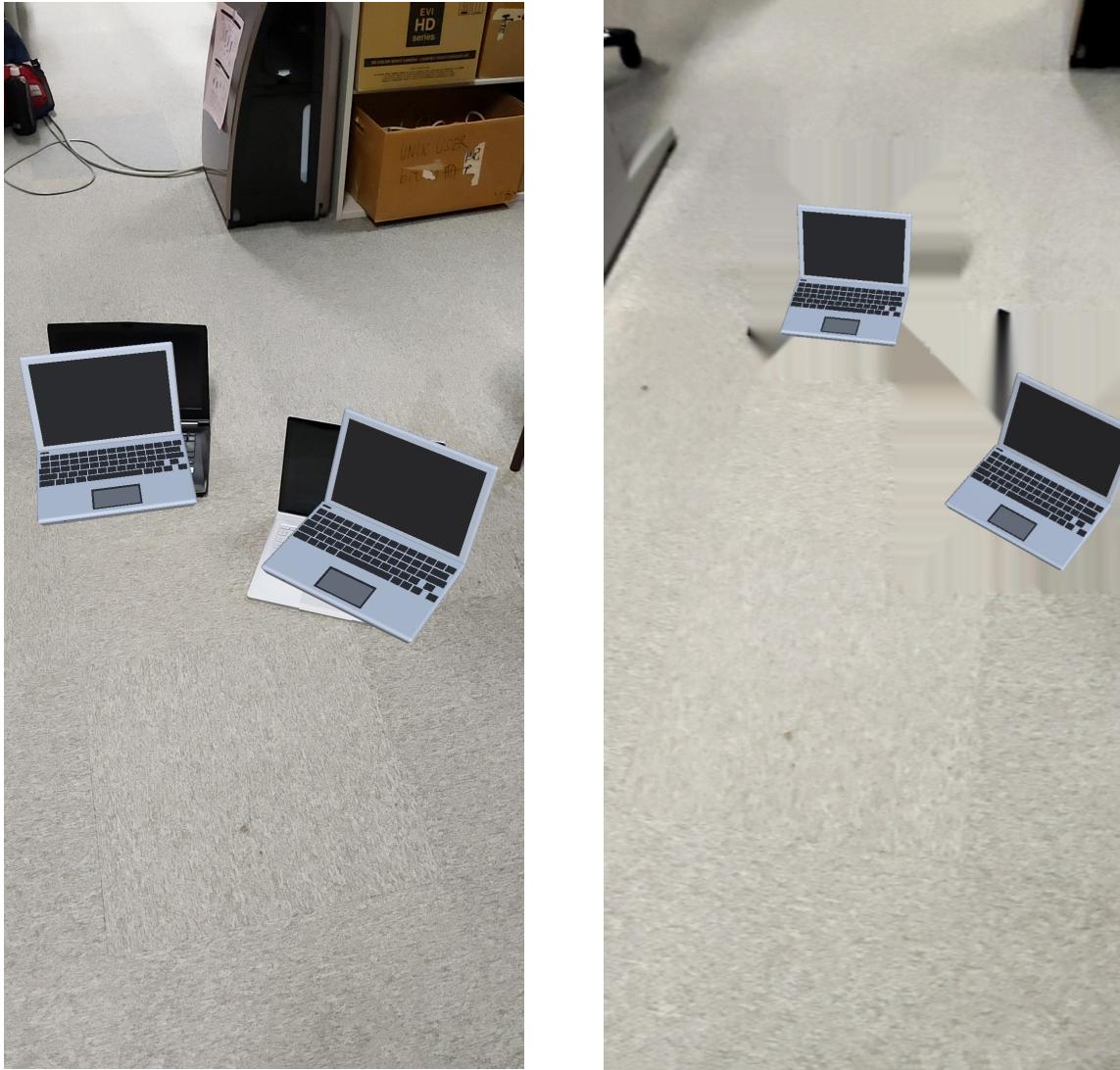


図 4.4: 複数の物体を置換した場合

床にノートパソコンが二つあるシーンで実験を行った。従来の AR 技術（左の画像）では、それぞれ現実のノートパソコンの一部が見えている。本システム（右の画像）では、それぞれ現実のノートパソコンは消去され仮想物体のノートパソコンのみが表示されている。

図 4.4 では同じカテゴリの物体を 2 つ置換したが、本システムでは異なるカテゴリの物体が複数個ある場合でも、同様に各カテゴリに対応する仮想物体で置換できる。

### 4.3 一般物体認識のメリット

本システムでは、除去対象領域の決定に一般物体認識のバウンディングボックスを用いた。DR では除去対象領域の決定に一般物体認識を使用することはない。除去対象領域を求めるだけであれば、より効率の良い方法があるからである。

しかし、AR 技術に DR 技術を組み合わせる場合、除去対象領域の決定に一般物体認識を用いることには次のメリットがある。

1. 除去対象領域に含まれる物体のカテゴリを取得できる
2. シミュレーション開始時に与えた除去対象領域に影響されない

#### 4.3.1 AR 表現の拡大

除去対象領域に含まれる物体のカテゴリを取得できるため、AR 表現の拡大につながる。本システムでは、除去対象領域に含まれる物体のカテゴリと同カテゴリの仮想物体に置換した。逆に、実物体のカテゴリに対して異なるカテゴリの仮想物体で置換することも可能である。

ARCore には事前に登録した画像を従来の AR マーカーのように利用することができる、マークー認識機能 (Augmented Image) がある。これは登録した画像とスマートフォンのカメラで撮影した画像が一致した場合に仮想物体を配置することができる機能だ。

一方、一般物体認識では形状や見た目が異なる物体であっても、カテゴリが同じならば同一のものとして扱える。すなわち、登録した画像と一致するかではなく、登録したカテゴリと一致するかで判断することができる。そのため、カテゴリが一致するという条件を用いた AR 表現を行える。

#### 4.3.2 シーンの変化への対応

従来の DR 手法のようにシミュレーション開始時に除去対象領域を与えるのではなく、一般物体認識のバウンディングボックスを用いることで、シーンの大きな変化に対応することができる。AR はユーザが自由に移動するため、視点が 360 度回転したりシミュレーション開始時から位置が大きく変化したりする。そのため、従来の DR 手法のようにシミュレーション開始時に除去対象領域を与えトラッキングを行うと、トラッキングに失敗する場合がある。また、シミュレーション開始時に与えた除去対象領域に含まれない物体を追加で消去することができない。

一方、一般物体認識はフレームごとに実行されるため、シミュレーション開始時に除去対象領域を与える必要はない。一般物体認識のバウンディングボックスを使用することで、シミュレーション開始時に与えた除去対象領域に依存しないため、除去対象が移動した場合や増減した場合にも対応することができる。

### 4.4 うまく動作しない場合の原因

本システムの出力結果は一般物体認識と Inpainting に大きく依存する。出力結果が目的を達成できない場合の原因是以下の通りである。

1. 物体の検出に失敗した
2. バウンディングボックスが物体を完全に覆えていない
3. 背景が複雑である

#### 4.4.1 物体の検出に失敗した

一般物体認識には物体を検出できない位置が存在する。そのため、本システムではフレームによっては Inpainting の処理が行われず、実物体の一部が仮想物体から見えててしまう。一般物体認識に失敗した場合を図 4.5 に示す。図 4.5 は一般物体認識が成功し仮想物体で置換した後に、一般物体認識が失敗したフレームを示している。右側のノートパソコンが検出されていないため、実物体の一部が見えている。

#### 4.4.2 バウンディングボックスが物体を完全に囲っていない

本システムで用いた Inpainting の手法は欠損領域内の画素値を周囲から補間することで修復する。そのためバウンディングボックスが完全に物体を囲うことができていない場合、図 4.6 のような結果となる。図 4.6 では物体の上部と右部の一部が除去対象領域の外側にある。上の画像は入力と除去対象領域を示し、下の画像は Inpainting した結果を示している。

Inpainting の結果、除去対象領域からはみ出した物体上部の影響を受け、白色になるべき箇所が灰色になっている。

#### 4.4.3 背景が複雑

本システムで用いた Inpainting 手法は、画像中の小さな欠損領域に適応するものである。そのため、バウンディングボックスのような大きな欠損領域では修復精度は高くない。また、除去対象領域が矩形であるため、欠損領域内の画素値を周囲から補間することで修復する手法では、本来修復で使用されるべきピクセルも除去対象領域に含まれてしまう問題がある。単純な背景であれば 4.2 節のような結果になる。しかし、複雑な背景であれば修復精度は低くなる。

複雑な背景の場合の Inpainting 結果を図 4.7 に示す。左の画像は入力画像と除去対象領域を示し、右の画像は Inpainting した結果を示している。除去対象領域上部の段ボールが引き延ばされたような修復結果となった。床になるべき箇所が段ボールの色になっている。



図 4.5: 物体の検出に失敗した場合

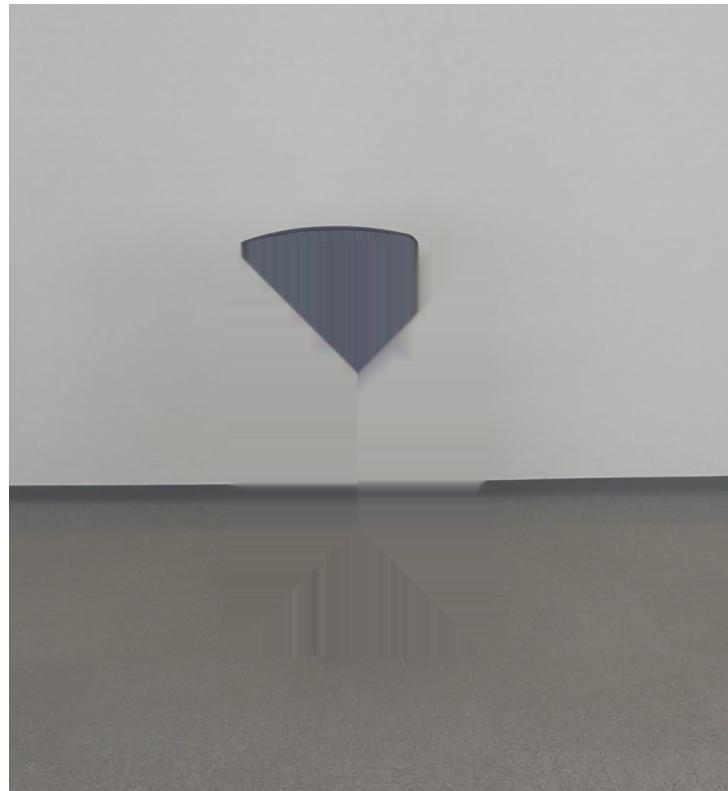


図 4.6: 使用した仮想物体



図 4.7: 背景が複雑な場合の Inpainting

## 第5章　まとめ

本研究では、AR技術に一般物体認識と画像処理を組み合わせて、実物体と仮想物体を重なるように配置した場合に、実物体と同カテゴリの仮想物体のみが表示されるシステムを開発した。まず一般物体認識で物体のカテゴリと位置を取得する。取得したカテゴリは仮想物体の置換処理で使用し、物体の位置を示すバウンディングボックスはInpaintingの除去対象領域の決定に使用した。

本研究では、従来のDR手法のようにシミュレーション開始時に指定した除去対象領域をトラッキングするのではなく、一般物体認識のバウンディングボックスを除去対象領域とした。一般物体認識を用いることで除去対象領域中の物体のカテゴリを取得できるため、カテゴリを活用したAR表現ができる。その結果、AR表現の拡大につながる。

本システムの一般物体認識とInpaintingの処理は、使用するデバイスのCPU性能に応じて任意のアルゴリズムに変更可能である。本研究の実験では、スマートフォンで動作させるために一般物体認識、Inpaintingのそれぞれの処理で軽量なアルゴリズムを使用した。CPU性能によっては、より精度の高い一般物体認識やInpaintingのアルゴリズムを用いることが可能であろう。

しかし、節で述べたように課題が残っている。これらの課題を解決し、より違和感なく実物体が視覚的に消去され仮想物体のみが表示されるシステムを開発し、本手法の有用性を検証する必要がある。

## 参考文献

- [1] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
- [2] Kawai, Norihiko, Tomokazu Sato, and Naokazu Yokoya. "Diminished reality based on image inpainting considering background geometry." IEEE transactions on visualization and computer graphics 22.3 (2015): 1236-1247.
- [3] Queguiner, Glen, Matthieu Fradet, and Mohammad Rouhani. "Towards mobile diminished reality." 2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct). IEEE, 2018.
- [4] Bertalmio, Marcelo, Andrea L. Bertozzi, and Guillermo Sapiro. "Navier-stokes, fluid dynamics, and image and video inpainting." Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. Vol. 1. IEEE, 2001.
- [5] Telea, Alexandru. "An image inpainting technique based on the fast marching method." Journal of graphics tools 9.1 (2004): 23-34.
- [6] He, Kaiming, and Jian Sun. "Statistics of patch offsets for image completion." European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2012.
- [7] Perfilieva, Irina, and Pavel Vlašánek. "Image reconstruction by means of F-transform." Knowledge-Based Systems 70 (2014): 55-63.
- [8] YUV フォーマット及び YUV と RGB の変換, <https://vision.kuee.kyoto-u.ac.jp/~hiroaki/firewire/yuv.html>