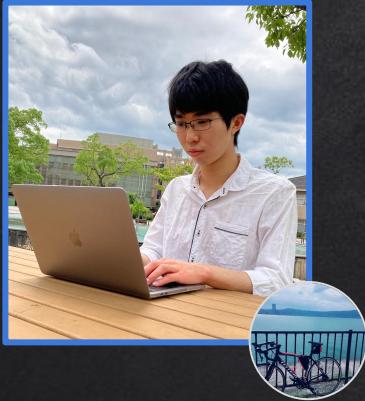


バーチャル背景適用済み映像から
部屋の画像を復元する手法





名前:石川 琉聖 (Ryusei Ishikawa)

所属:Ritsumeikan University

Twitter:@ryusei_ishika

実績:seccamp' 19

SecHack365' 20

ひとこと:素人質問する前に「僕は玄人です」と言ってください(ToT)



名前:辻 知希 (Satoki Tsuji)

所属:NEET (Nagoya University?)

Twitter:@satoki00

実績:GoogleVRP\$5000

CVEちょっと

ひとこと:よわよわなのでいちめないでください(ToT)

RESULT



ZOOMG

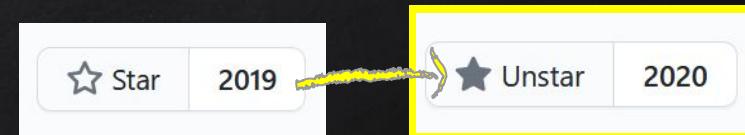


Zoomgとはバーチャル背景と体などのオブジェクトとの境界の色情報を抽出して、背後に隠れた部屋を自動で復元するツールである。

ZOOMG



A screenshot of the GitHub repository page for 'zoomg'. The repository was created by 'xryusei' and merged into the 'master' branch. It contains several files: '.github', 'avtokyo', 'images', 'resources', 'sample', 'zoomg', '.gitignore', 'ALGORITHM.md', 'LICENSE', and 'README.md'. The repository has 2 branches, 1 tag, 22 commits, and 3 stars. A red box highlights the star count (0) and the fork count (0). A yellow arrow points from the star count to the 'Star' button below the repository details.



最新の開発リポジトリ

<https://github.com/Tsuku43/zoomg>

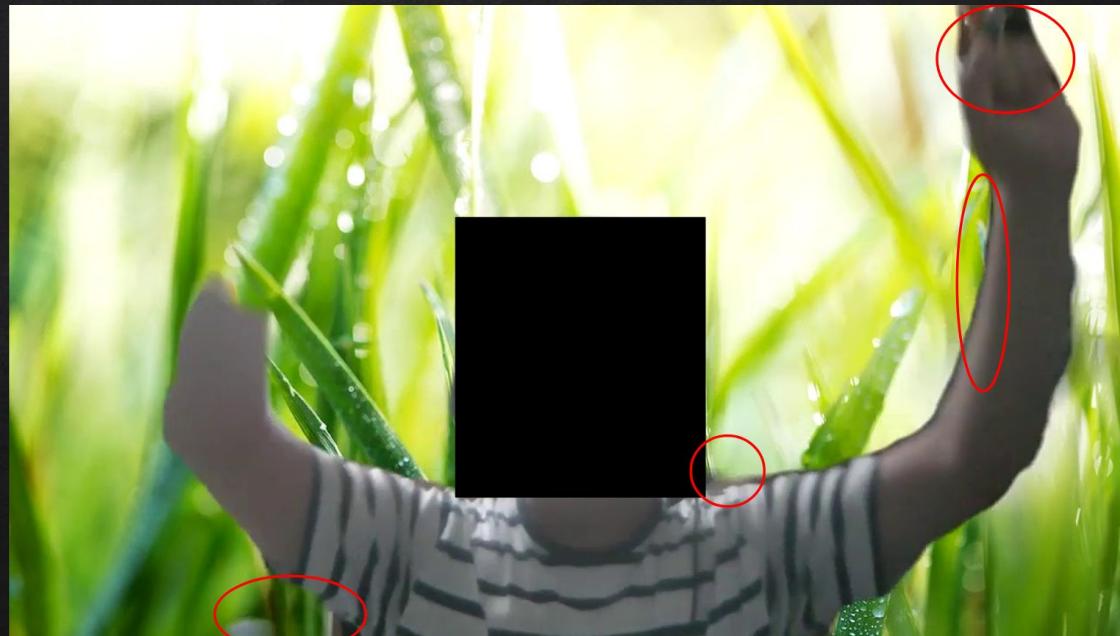


MANUALLY

手動で復元するおはなし



復元に使用する箇所





復元



BEFORE



AFTER



AUTOMATICALLY

自動で復元するおはなし



理想的な動画での復元

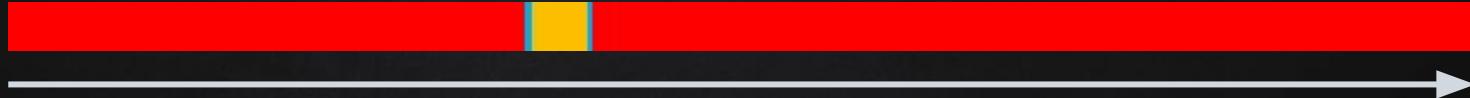




アルゴリズム

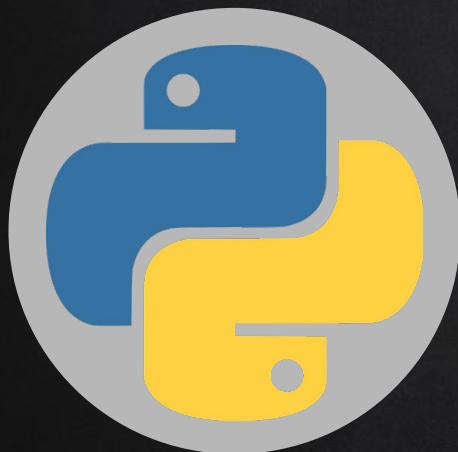


1px





実装1 - 使った技術・言語 -



pybind11



実装2 – 実装手法(初期化) –

Pythonスクリプト

```
import zoomg
zoom = zoomg.Zoomg(height, width) # 初期化
while frame:
    frame = frame_read(video) # 画像を取得
    zoom.add_image(frame) # 画像を追加
zoom.generate_image(param) # 画像を生成
image = zoom.get_image() # 生成画像を取得
savefig(image) # 画像を保存
```

1. 生成画像用配列の作成

```
using Pixel = std::vector<int>;
using Image = std::vector<vector<Pixel>>;
Image(height, std::vector<Pixel>(width)) image;
```

2. 出現頻度管理用配列の作成

```
std::vector<std::vector<
    std::map<Pixel, std::pair<int, int>>> freq;
// 各ピクセルで平衡二分木を作成する
// std::mapはキーを画素、値は個数と出現タイミングを表す
```



実装2 - 実装手法(画像を追加) -

Pythonスクリプト

```
import zoomg
zoom = zoomg.Zoomg(height, width) # 初期化
while frame:
    frame = frame_read(video) # 画像を取得
    zoom.add_image(frame) # 画像を追加
zoom.generate_image(param) # 画像を生成
image = zoom.get_image() # 生成画像を取得
savefig(image) # 画像を保存
```

3. 各画素の出現頻度を管理する木を作成

```
void add_image(Image img) {
    int order = 0; // 出現順を保持する
    for (int h = 0; h < height; ++h) {
        for (int w = 0; w < width; ++w) { // 全ての座標について
            Pixel p = img[h][w];
            // ある画素が以前出現したことがある場合
            if (freq[h][w].find(p) != freq[h][w].end()) {
                ++freq[h][w][p].first;
            } else { // ある画素が初めて出現した時
                freq[h][w][p] = {1, order++};
            }
        }
    }
}
```



実装2 - 実装手法(画像を生成) -

Pythonスクリプト

```
import zoomg

zoom = zoomg.Zoomg(height, width) # 初期化

while frame:
    frame = frame_read(video) # 画像を取得
    zoom.add_image(frame) # 画像を追加
    zoom.generate_image(param) # 画像を生成
    image = zoom.get_image() # 生成画像を取得
    savefig(image) # 画像を保存
```

4. 各画素を出現個数、出現タイミングの順でソート($=S$)
5. S のコサイン類似度がparam未満の値を探索($=p$)
6. p を復元画像の画素として使用

```
// 予めソートした配列 = most_commonを作成しておく
Pixel most = *most_common.rbegin(); // 最頻値を取得

for (auto p : most_common) {
    if (cos_sim(p, most) < param) { // コサイン類似度を計算
        image[h][w] = p; // 復元できた画素として特定
        return;
    }
}
image[h][w] = most; // 復元できなかった場合
```



実装2 - 実装手法(生成画像を取得) -

Pythonスクリプト

```
import zoomg
zoom = zoomg.Zoomg(height, width) # 初期化
while frame:
    frame = frame_read(video) # 画像を取得
    zoom.add_image(frame) # 画像を追加
zoom.generate_image(param) # 画像を生成
image = zoom.get_image() # 生成画像を取得
savefig(image) # 画像を保存
```

7. 復元画像 image を取得

```
return image;
```



実装3 - 計算量 -

時間計算量

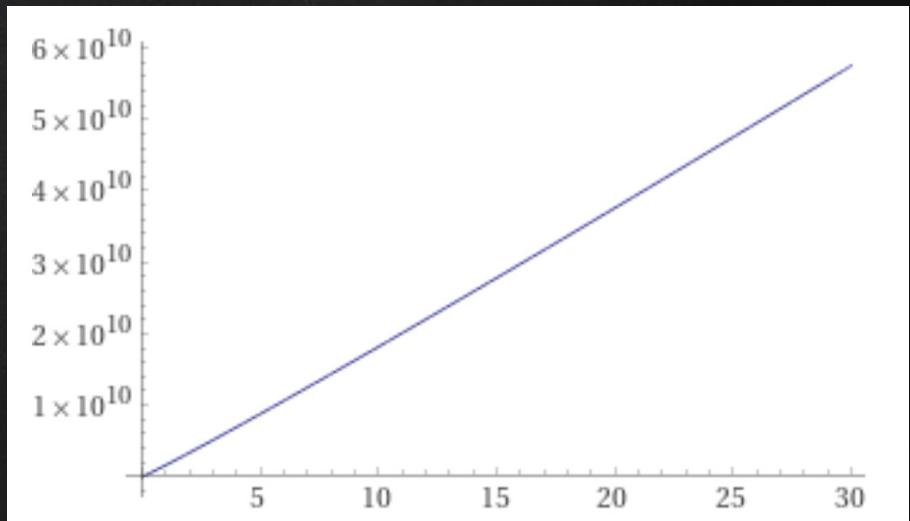
F := フレーム数

H := 画像の高さ

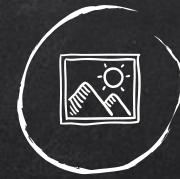
W := 画像の幅

とした時、

$O(FHW \log_2 (FHW))$

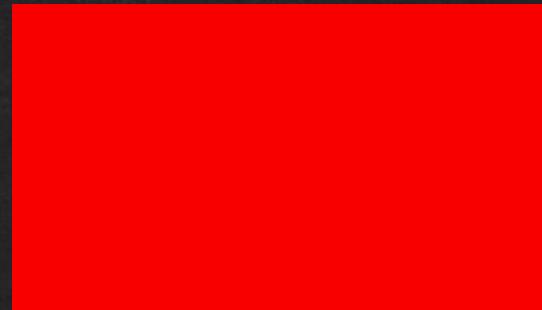


動画の再生時間に対する見積演算回数
横軸 : 時間[sec], 縦軸 : 見積演算回数[回]



アルゴリズムの評価

BACKGROUND



RECONSTRUCTION



3.

DEFENSE

復元を防御するおはなし



防御手法の考え方

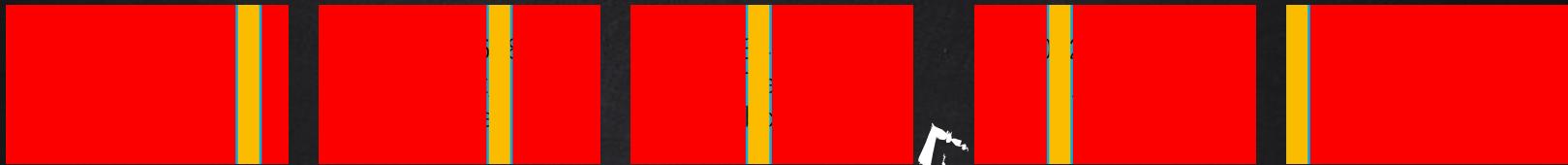
動画にすればよい



- ✖ 容易に手に入る動画
- ✖ 短時間でループする動画

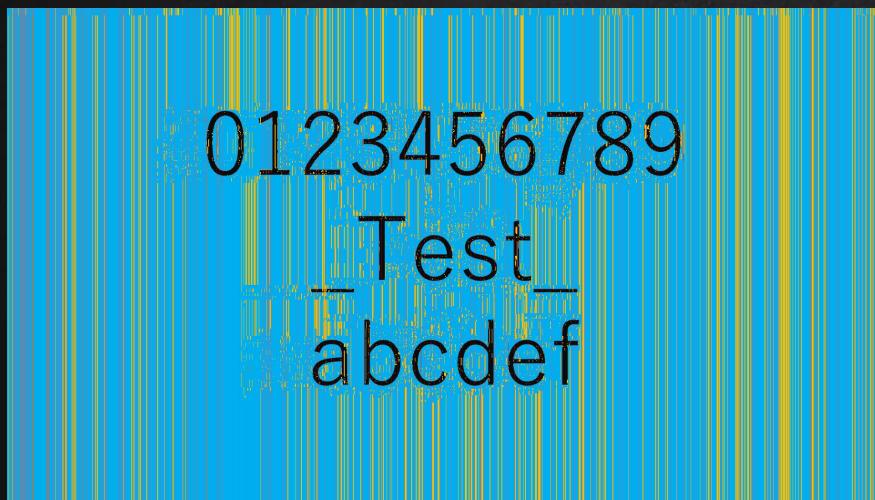


防御手法1 -フレーム単位でのノイズ-

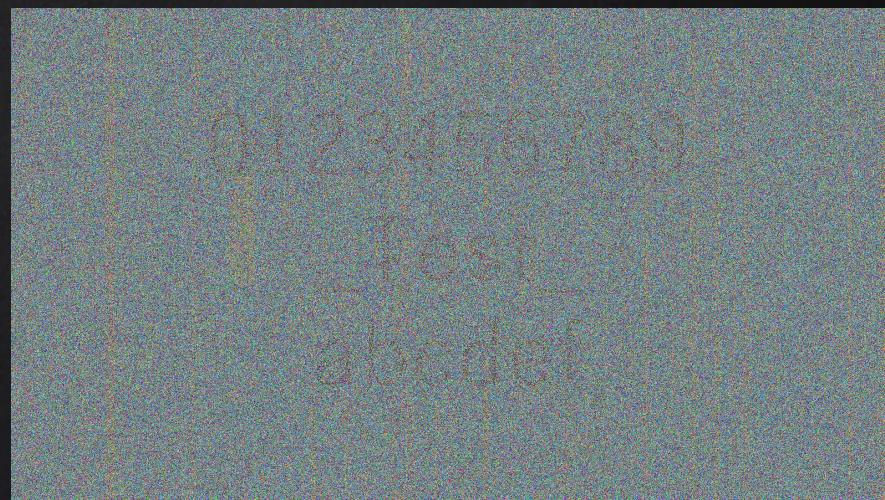




防御手法1 -フレーム単位でのノイズ-



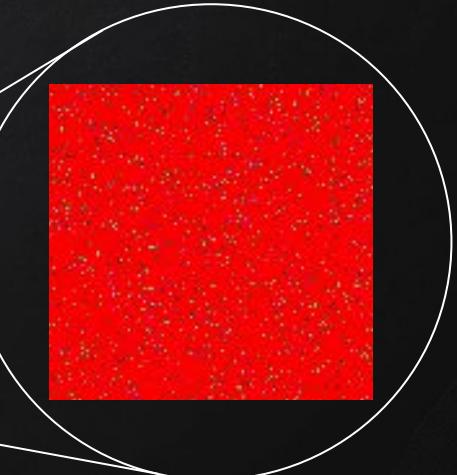
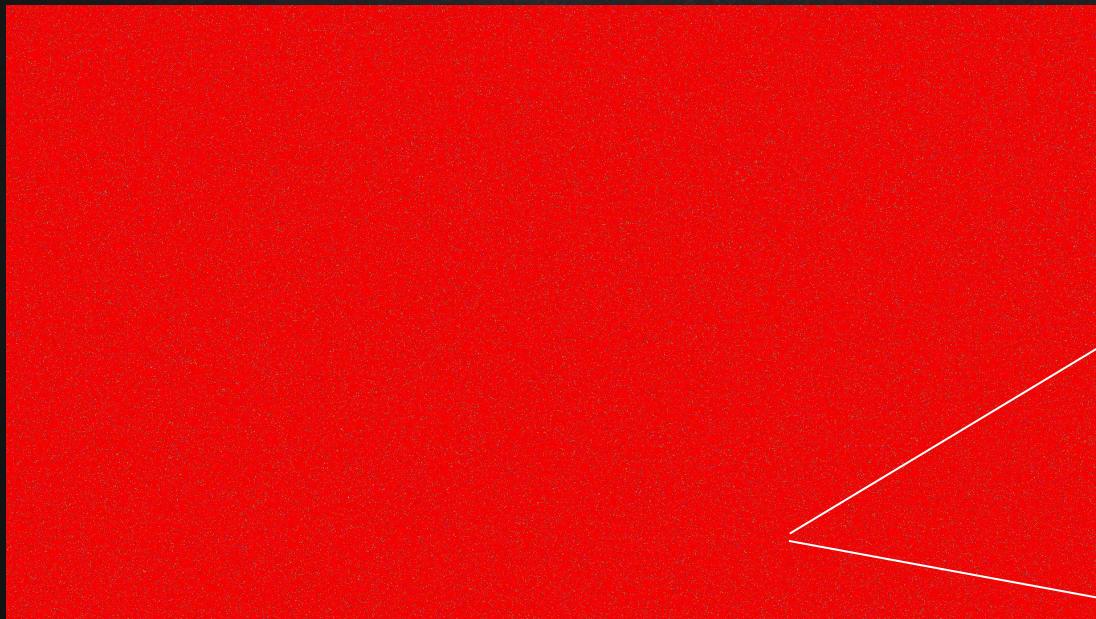
BEFORE



AFTER

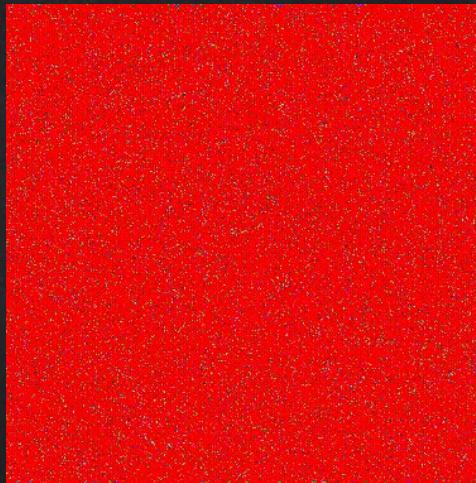


防御手法2 -ピクセル単位でのノイズ-





防御手法2 -ピクセル単位でのノイズ-





最強の防御手法



4.

PRACTICALLY

実際のおはなし



バーチャル背景適用済み映像からの復元



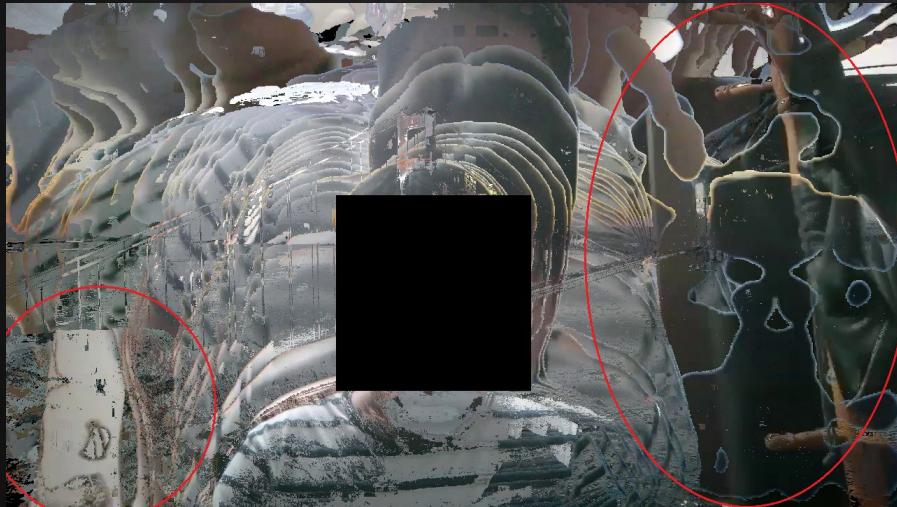
復元した部屋



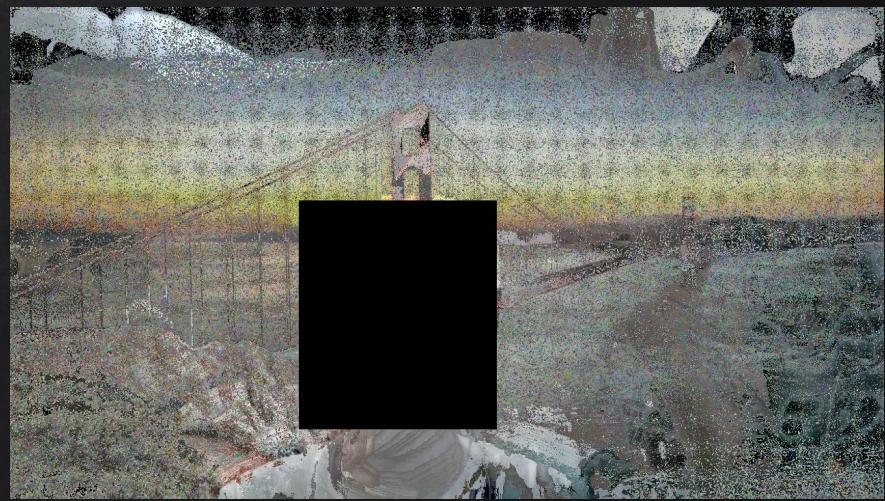
本当の部屋



ピクセル単位のノイズによる復元防止



ノイズなしで復元



ノイズありで復元



Capture The Flag !!!



<https://bit.ly/37TnyB6>