

TECHNIQUES FOR RESTORING ROOM IMAGES FROM VIRTUAL BACKGROUND IMAGES

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





Name :Ryusei Ishikawa (石川 琉聖)

Branch :Ritsumeikan University

Twitter:@ryusei_ishika

Achievements :

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)



Name :Satoki Tsuji (辻 知希)

Branch :NEET (Nagoya University?)

Twitter:@satoki00

Achievements :

GoogleVRP\$5000

CVE...

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 is a tool that extracts color information from the boundary between a virtual background and an object, such as a body, and automatically restores the room hidden behind it.

ZOOMG



The screenshot shows the GitHub repository page for `Tsuku43/zoomg`. The repository has 0 stars and 0 forks. The code tab is selected, showing a commit history from yesterday. A yellow arrow points to the 'Code' dropdown menu. Another yellow arrow points to the 'Star' button, which is highlighted with a yellow box. The commit history shows several commits, including 'add github setting files' and 'add README' for various files like '.github', 'avtokyo', 'images', 'resources', 'sample', 'zoomg', and '.gitignore'. The repository also includes an Octree view, a 'About' section with project details, releases (version 1.0), and packages.

File	Action	Time
.github	add github setting files	yesterday
avtokyo	add README	27 minutes ago
images	move gif	2 hours ago
resources	add README	27 minutes ago
sample	add README	27 minutes ago
zoomg	add README	27 minutes ago
.gitignore	add .gitignore	yesterday
ALGORITHM.md	add README	27 minutes ago
LICENSE	Update LICENSE	yesterday
README.md	add README	27 minutes ago

Latest Development Repositories

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

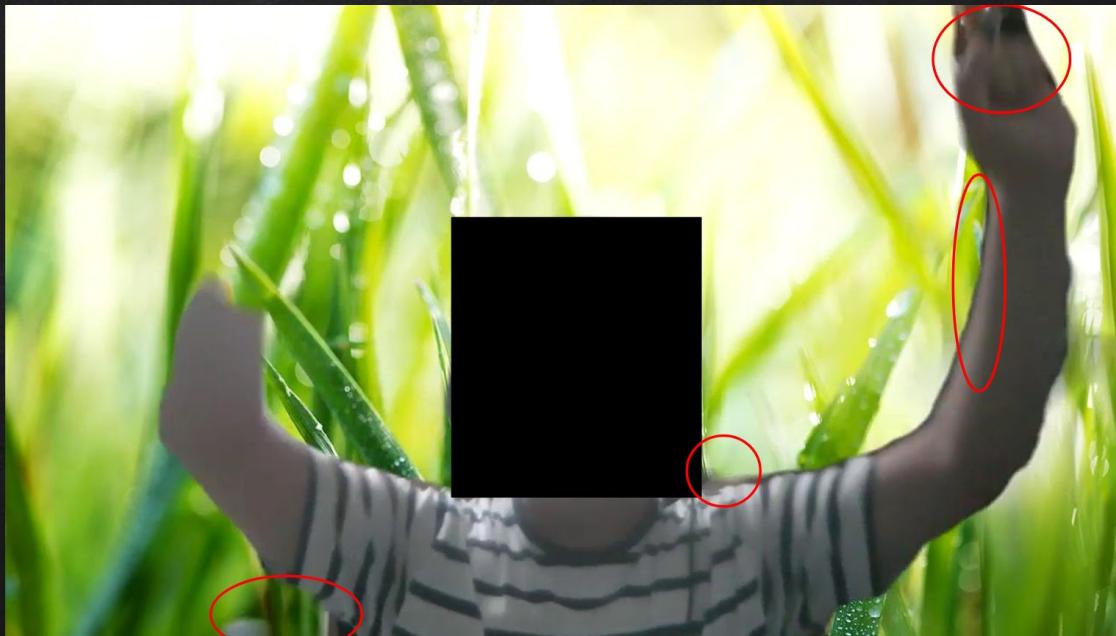


MANUALLY

手動で復元するおはなし

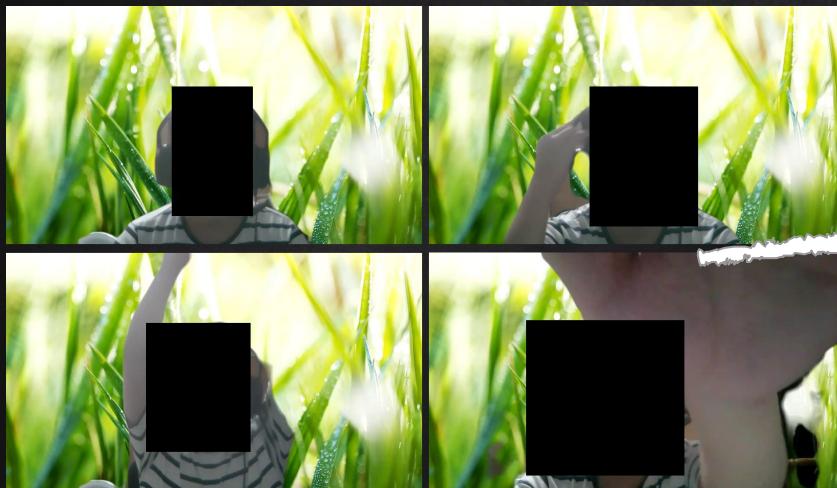


Areas used for restoration

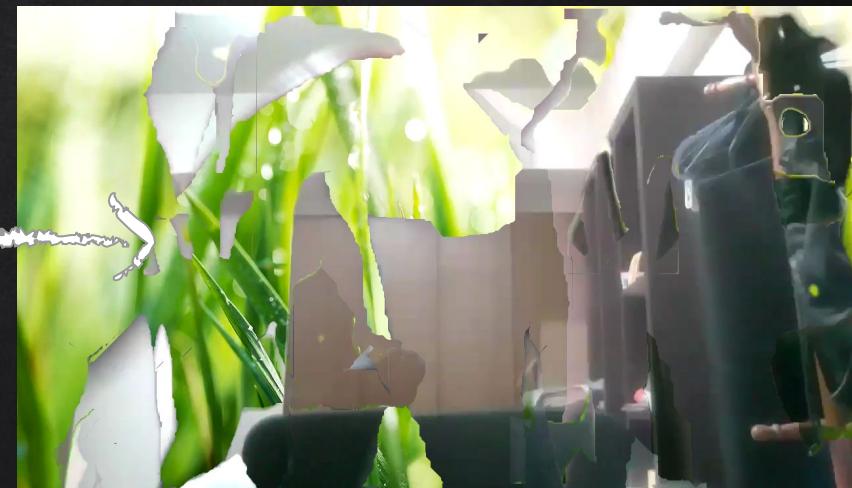




Restoration



BEFORE



AFTER



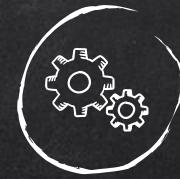
AUTOMATICALLY

自動で復元するおはなし



Idea of restoration

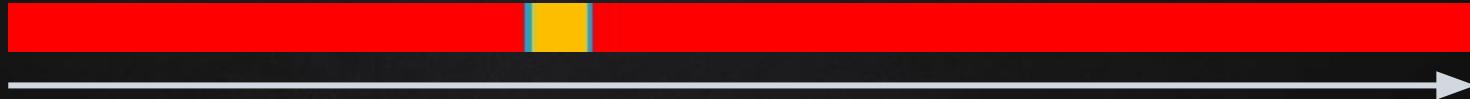




Algorithm

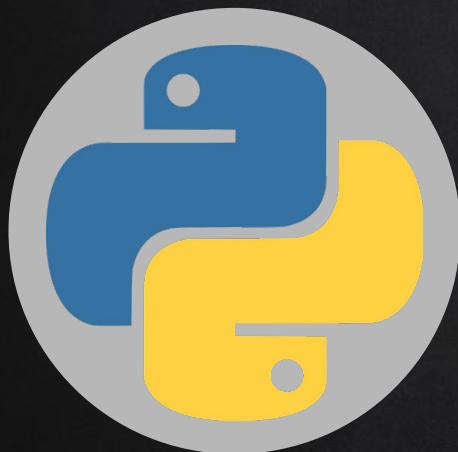


1px





Coding1 – Technology & Language –



pybind11



Coding2 – Initialization –

Python Script

```
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. Creating the array for the restored image

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

2. Creation of the array for frequency management

```
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
```



Coding2 – Add Images –

Python Script

```
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. Create a tree to manage the frequency of each pixel

```
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++};
            }
        }
    }
}
```



Coding2 – Generate Image –

Python Script

```
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. Sort each pixel in order of frequency and timing
5. Search for values cosine similarity less than the param
6. Use that pixel in the reconstructed image

```
// 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
```



Coding2 – Getting a generated image –

Python Script

```
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. Get a restored image

```
return image;
```



Coding3 – Computational cost –

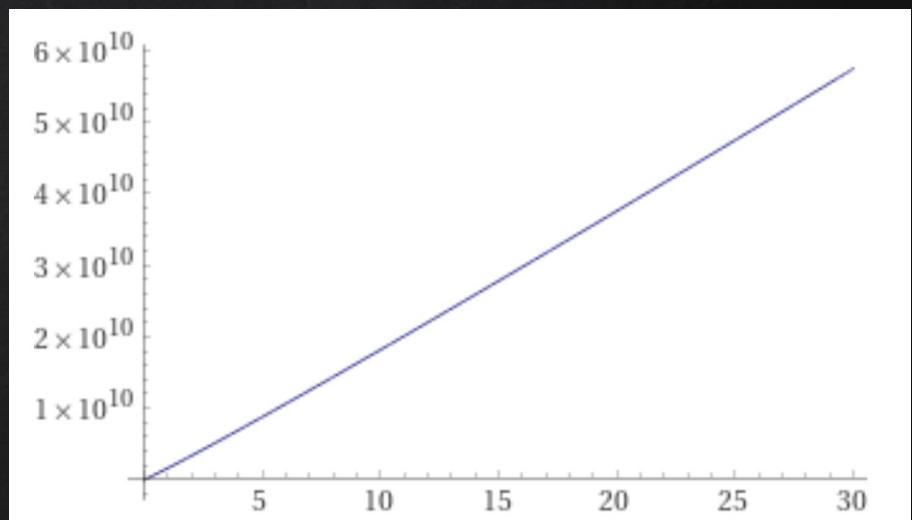
Time complexity

F := frames of the video

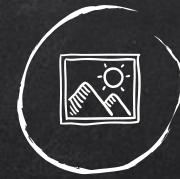
H := height of the image

W := width of the image

$O(FHW \log_2 (FHW))$



Estimated number of calculations for video playback time
X : Time [sec], Y : Estimated number of operations [times]



Evaluation

BACKGROUND



RECONSTRUCTIO



3.

DEFENSE

復元を防御するおはなし



Idea of defensive techniques

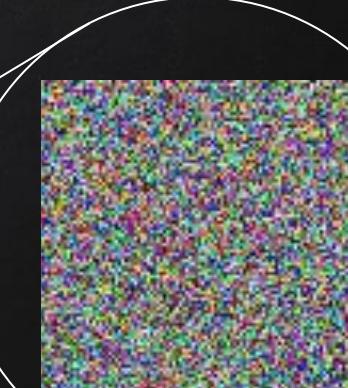
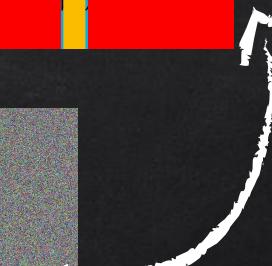
Wouldn't that be better on video?



- ✗ Easily available videos
- ✗ Short, looping videos

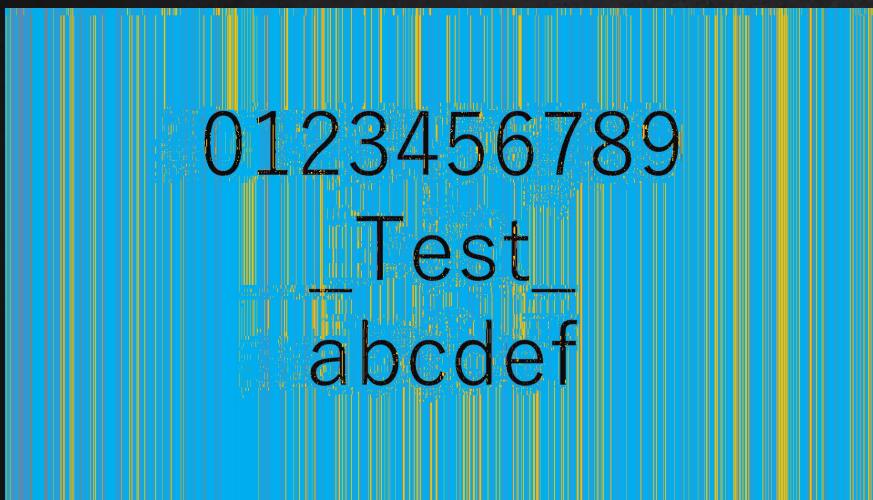


Defense1 – frame noises –

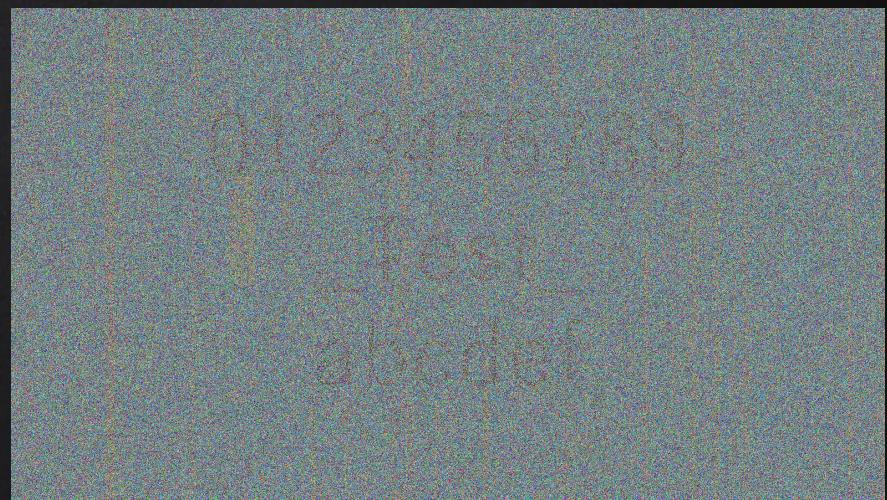




Defense1 – frame noises –



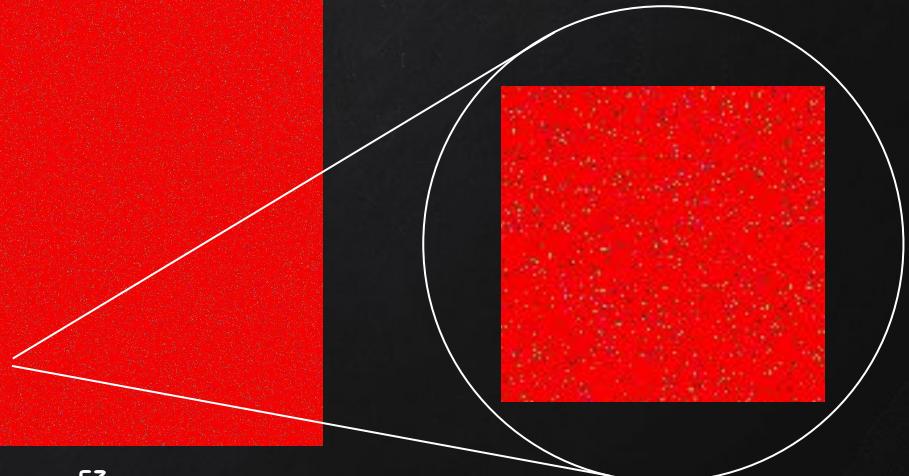
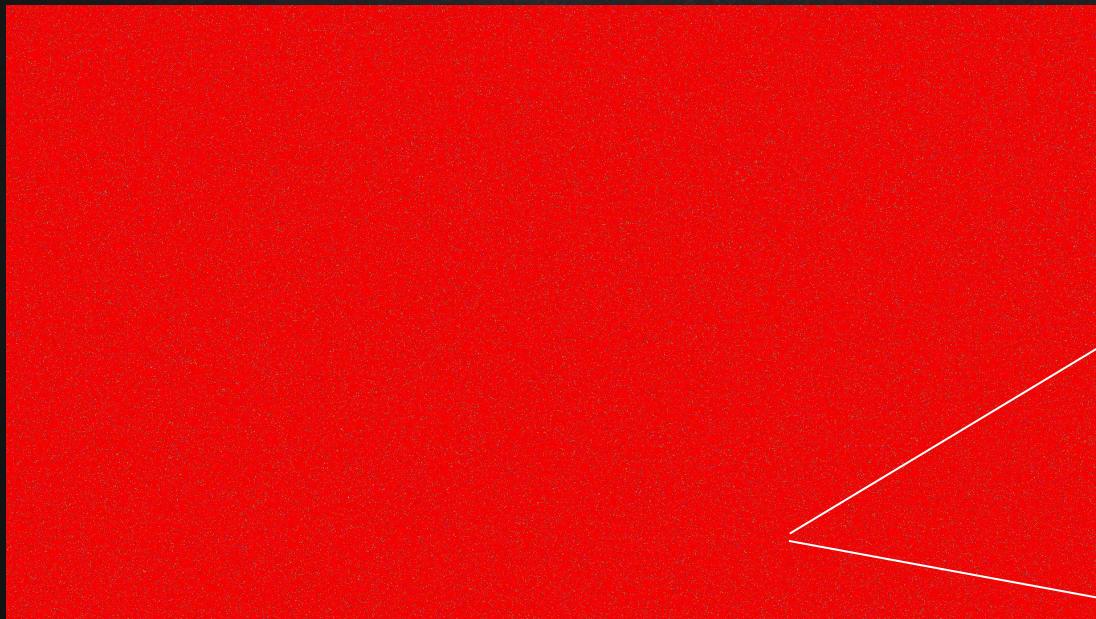
BEFORE



AFTER

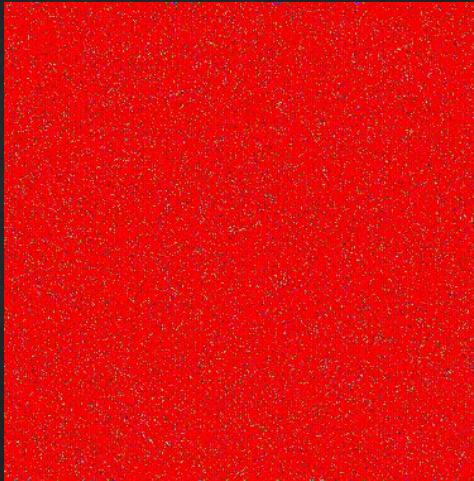


Defense2 – pixel noises –





Defense2 – pixel noises –





The Best Defense Method



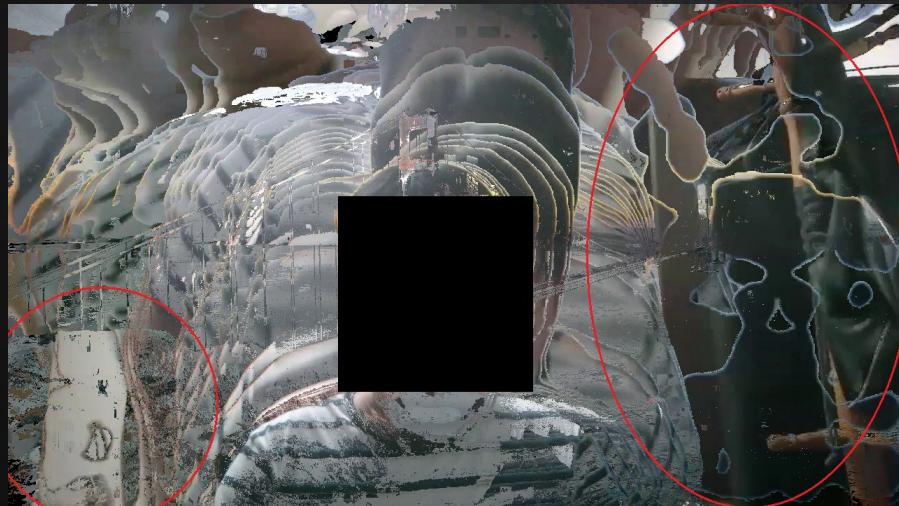
4.

PRACTICALLY

実際のおはなし



Restoring room image from virtual background images



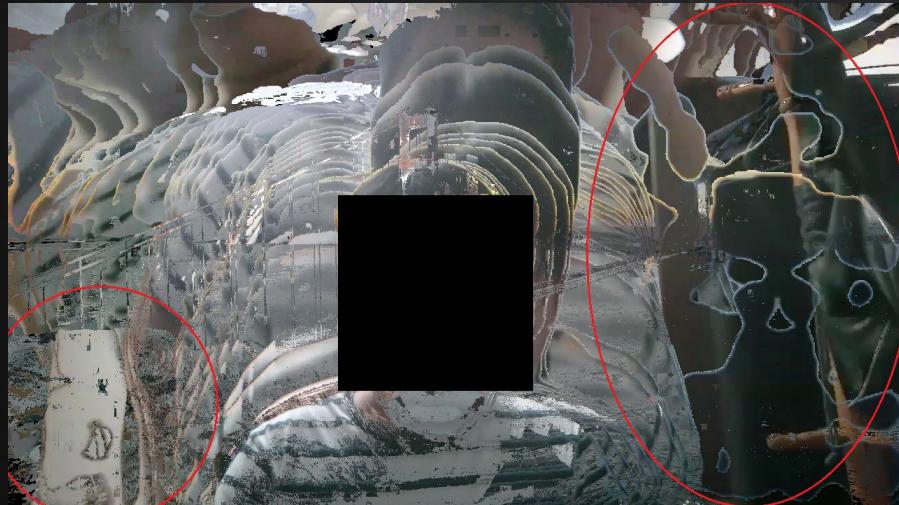
Restored Room



Real Room



Pixel noises defense



Restored without noises



Restored with noises



Capture The Flag !!!



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