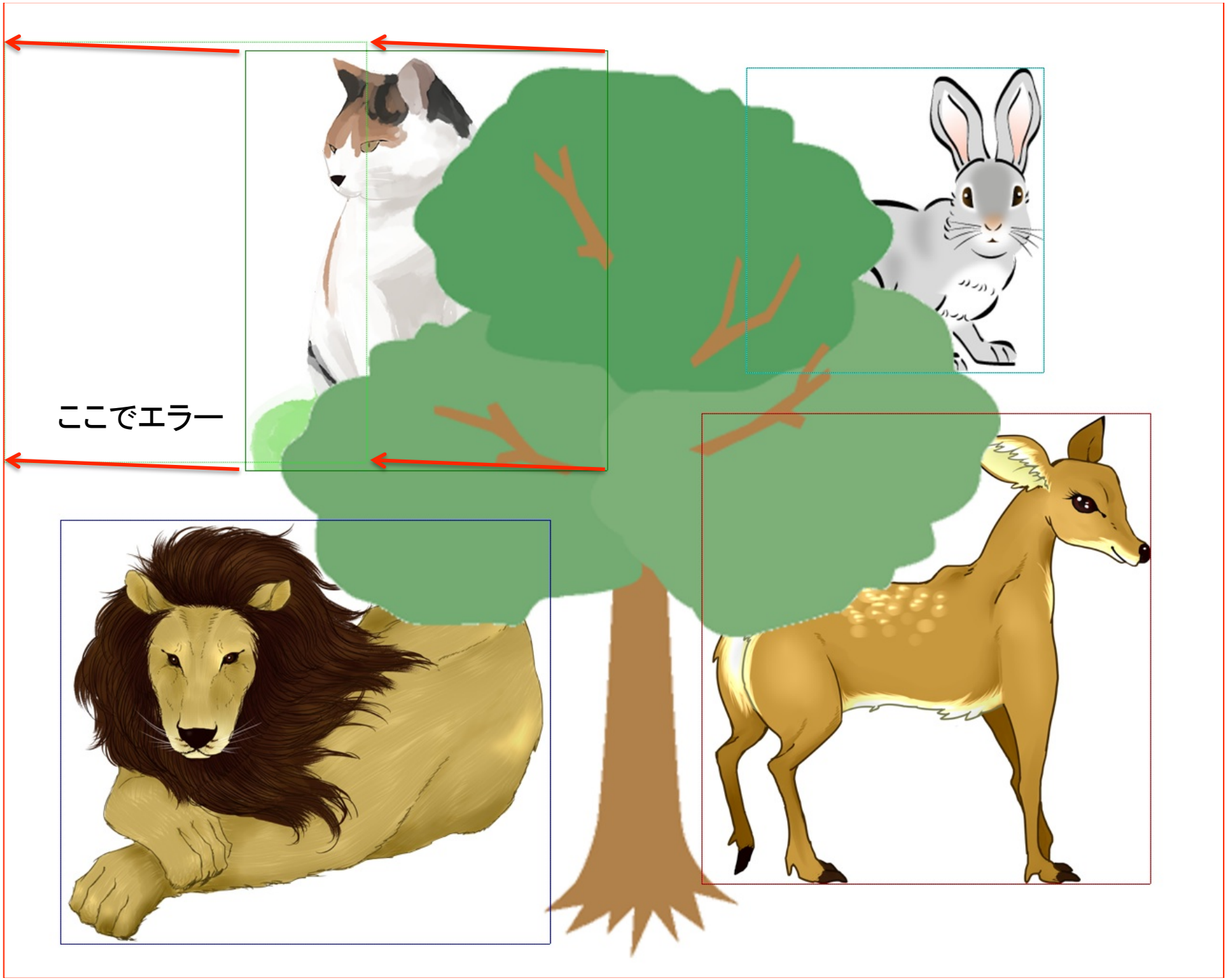
アルコン2013サンプルについて

使用されているアルゴリズム

- 単純なテンプレートマッチング
  - まず，画像をグレイスケールに変換
  - cvMatchTemplateで照合
    - CV_TM_SQDIFF : 画素値の差の二乗和
- 結果
  - おおむねうまく動いている
  - 隠れの大きいキャラクターの照合に失敗


ここでエラー

サンプルの性能

- Level1 のみ
  - 01 : 75%
  - 02 : 25%
  - 03 : 100%
  - 04: 49.83%
  - 05 : 66.67%

問題点

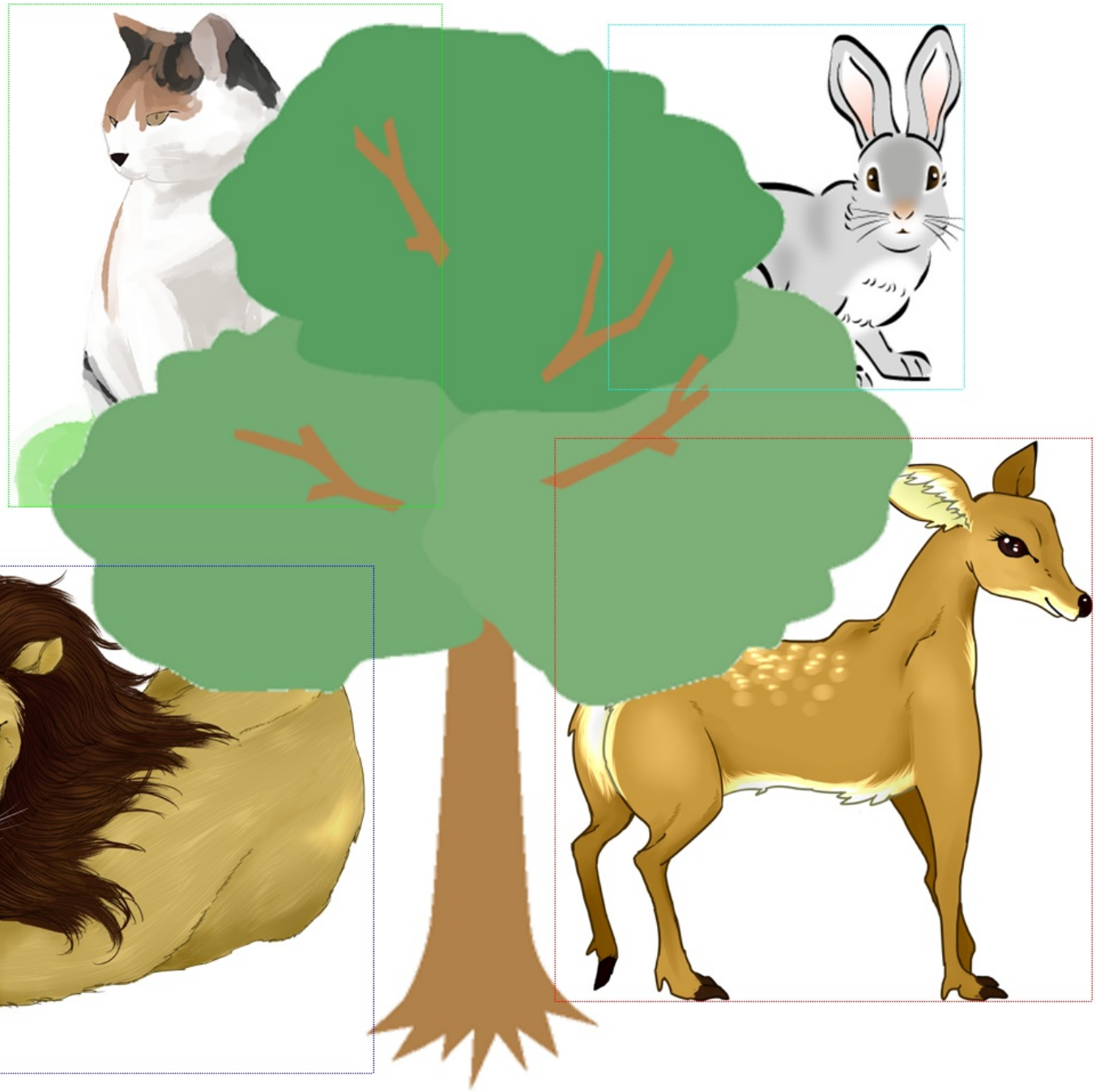
- 隠れの大きい部分のエラーが影響
  - 木の部分を避けて，背景の白い部分が一致するような位置のほうが誤差が少ない
  - 二乗和を用いていることも原因か

作りなおしてみた

- 画素値の一致を数え上げるアルゴリズム
  - テンプレート内で座標値をランダムに生成
    - 現在，300個
  - 画素値が一致するとカウントする
  - 一致した画素数が最も多いところを出力

```
// --- begin user code ---
int width  = image.width;
int height = image.height;

for (int i=0; i < reference.num; i++ ) {
  int x, y, tx, ty, mx, my, maxmatch;
  int t_width  = reference.sample[i].image.width;
  int t_height = reference.sample[i].image.height;

  printf("template %d\n", i);

  maxmatch = 0;
  for(ty = 0; ty < height - t_height + 1; ty++) {
    for(tx = 0; tx < width - t_width + 1; tx++) {
      int n, match = 0;

      for(n = 0; n < 300; n++) {
        x = rand() % t_width;
        y = rand() % t_height;
        if(reference.sample[i].image.data[(x+y*t_width)*3  ] == image.data[((tx+x)+(ty+y)*width)*3  ] &&
           reference.sample[i].image.data[(x+y*t_width)*3+1] == image.data[((tx+x)+(ty+y)*width)*3+1] &&
           reference.sample[i].image.data[(x+y*t_width)*3+2] == image.data[((tx+x)+(ty+y)*width)*3+2]) {
          match++;
        }
      }
      if(match > maxmatch) {
        mx = tx;
        my = ty;
        maxmatch = match;
      }
    }
  }

  set_result ( friends, mx, my, t_width, t_height, reference.sample[i].label );
}
// --- end user code ---
```

うまくいった



性能

- Level1 のみ
  - 01 : 100%
  - 02 : 99.78%
  - 03 : 80%
  - 04: 99.52%
  - 05 : 99.65%
- 乱数の代わりに，10画素飛ばしにしても
  あまり変わらない

問題点

- 遅い
  - OpenCV のテンプレートマッチングが速い
    （フーリエ変換を用いて実装してあるため）
- 乱数の個数
  - 増やすと遅くなる
- 背景の問題
  - Level1 の 03 では背景でマッチしてしまう

よりよい方針

- スピードアップ
  - まずは荒く探索する
  - それにより見つかった候補周辺で，より細かく探索する
- 安定化
  - RANSAC のような，ノイズにロバストなマッチングを行う

荒く探索する方法

- 解像度を下げる
  - 縮小比率の４乗で効いてくる
- アクティブ探索を用いる
  - ヒストグラムを生成する必要あり
- SIFTのような方法
  - 画像から特徴的な部分を探す
    - 例えば，20x20画素ぐらいの領域内で分散が大きい部分を抽出するなど