

# 続 クロマキー

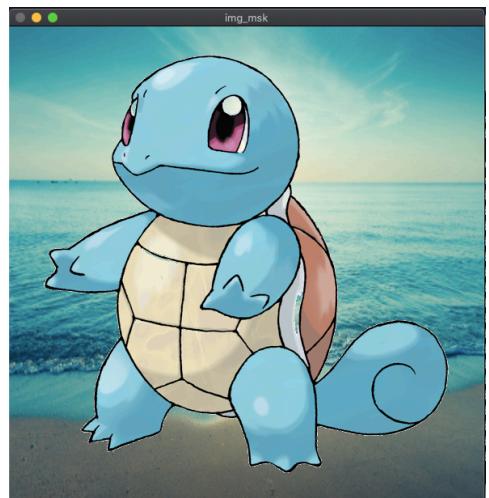
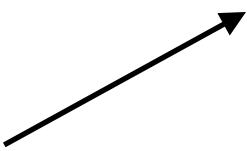
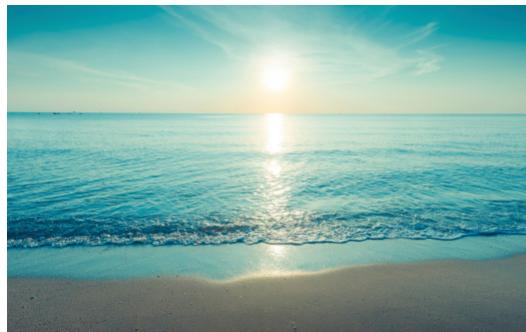
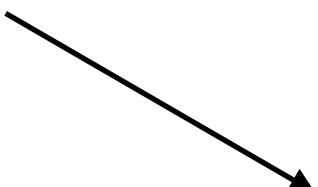
## 課題

合成対象物が写っている画像に背景があった場合、綺麗な合成画像を作成することができない。

例1 うまく合成できる場合



背景が白系統

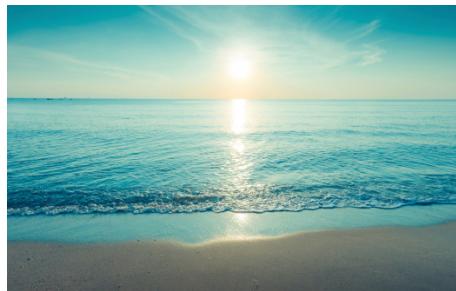
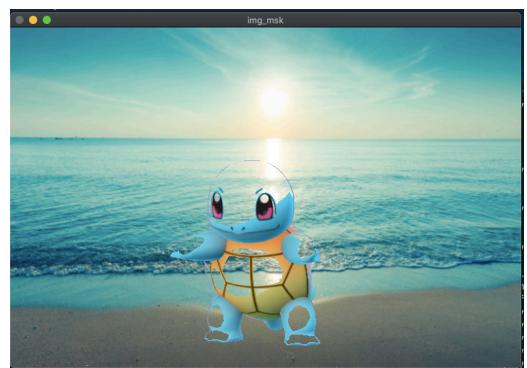
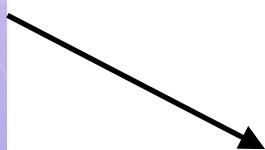


若干の乱れはあるものの、  
合成画像は作成されている

## 例2 背景に色がついている場合



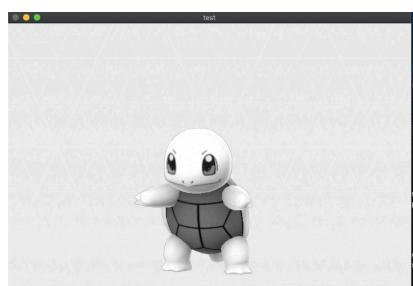
背景に色がある



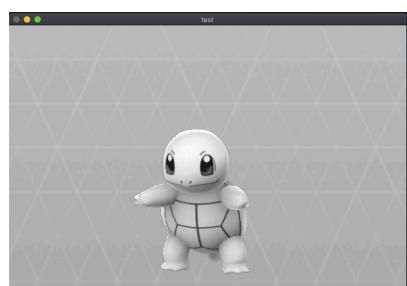
うまく合成できていない

今回はRGB画像をグレースケール変換して切り取り用の方を作成していました。グレースケール画像は輝度のみを表す画像であるため、色は異なっていてもグレースケール変換後の値が同じであることもあります。上記の例では、背景の薄紫と対象物の水色部分の輝度が同じであったため、背景と共に消されてしまったと考えました。

以下に、RGBの各要素に分けた画像を示します。



ブルーの画素値のみの画像



グリーンの画素値のみの画像

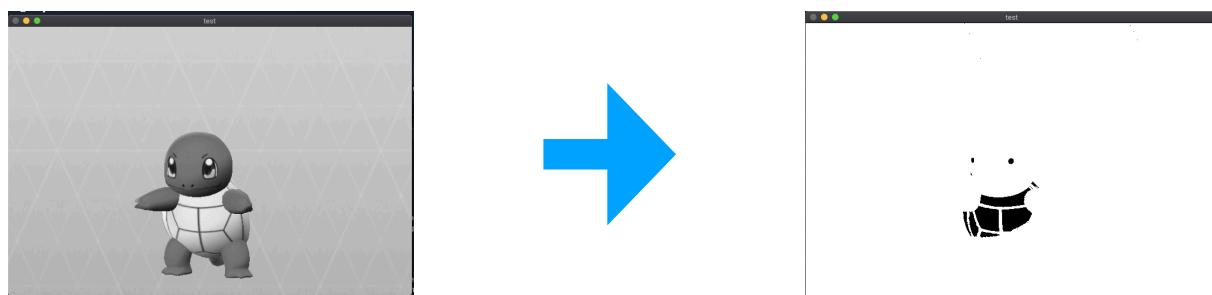
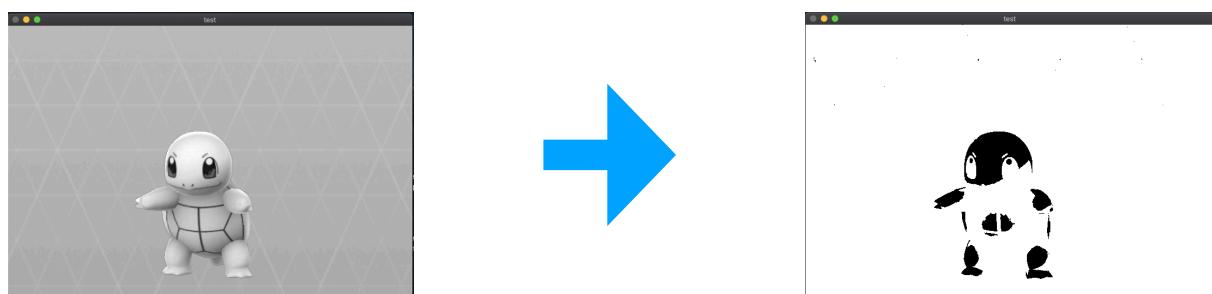
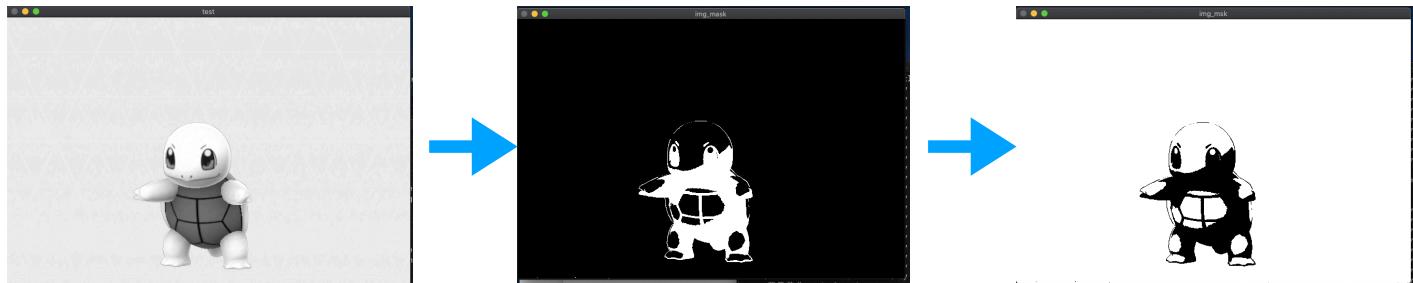


レッドの画素値のみの画像

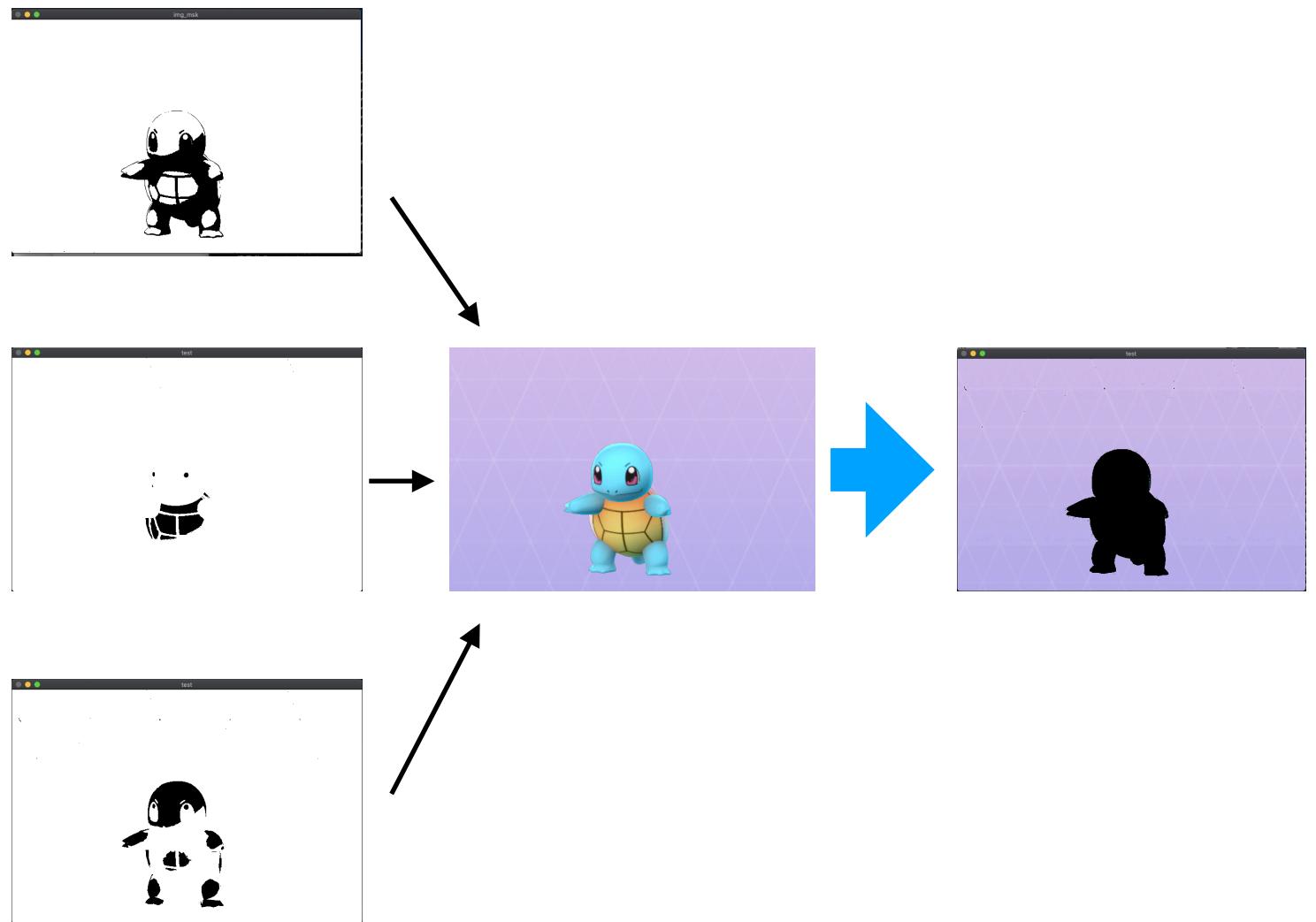
上記の結果から、RGBの各画素値で分けて処理を行えば精度が高い合成画像が作成できるのではないかと考えました。

## 方法1 物体の抽出

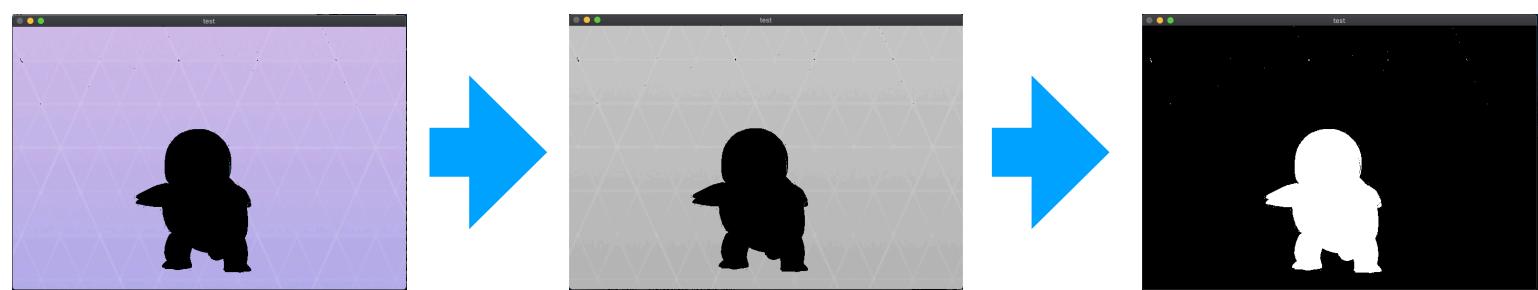
- ① RGBの各画素値で分けた画像をそれぞれ二値化する。この時、背景部分が黒く表示される場合は反転させる。



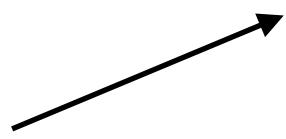
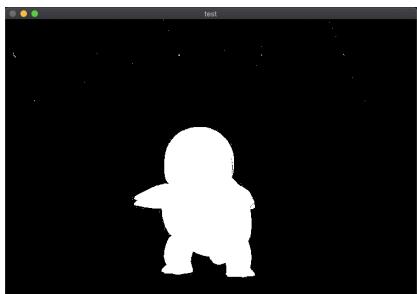
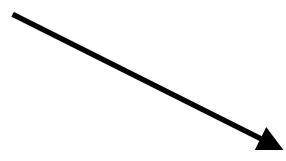
② ①で作成した画像と対象物が映った入力画像を使い、マスク画像の素材を作成



③ ②で作成したマスク画像の素材を元にマスク画像を作成

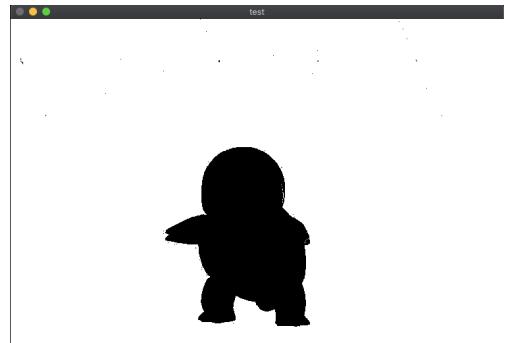
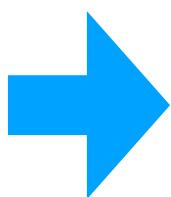
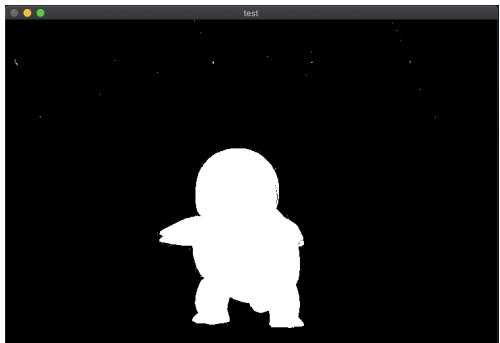


④ マスク画像を使用し対象物を切り抜く

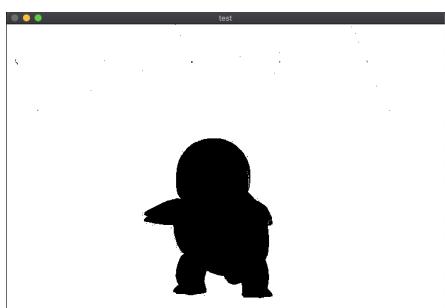
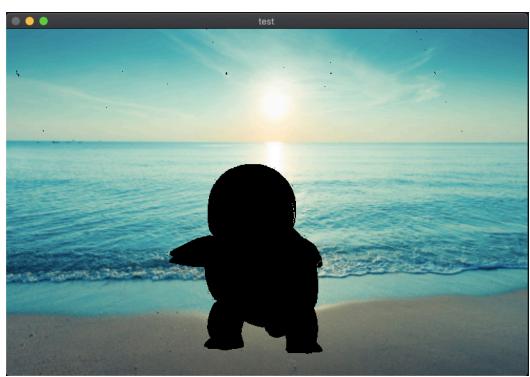
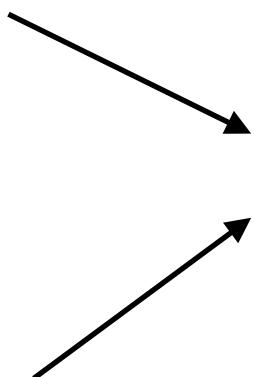
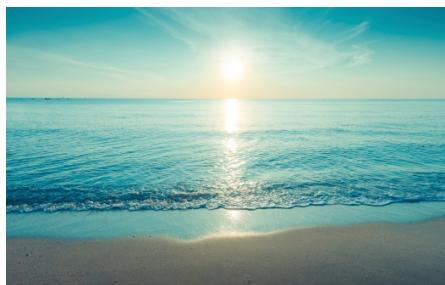


## 方法2 背景からの切り出し

① マスク画像を反転させる

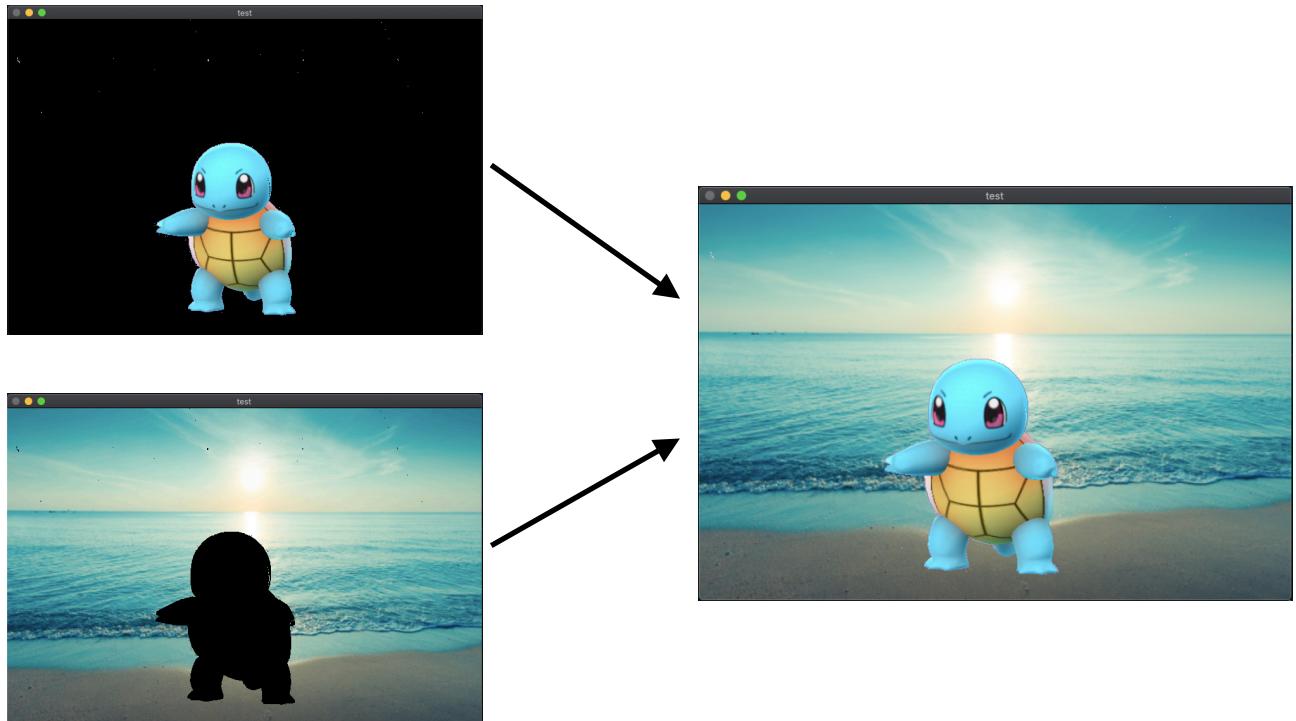


② 背景画像から背景用マスク画像分だけ切り出す

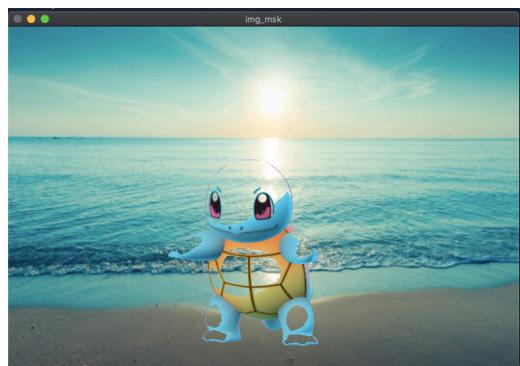


### 方法3 画像の合成

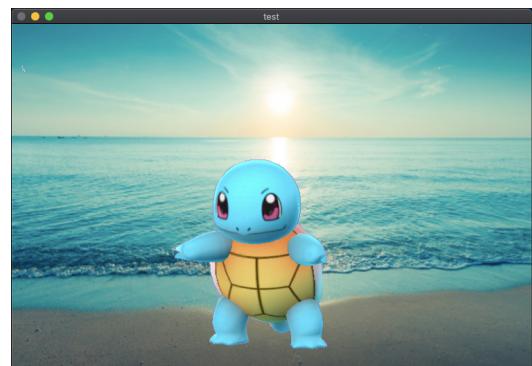
方法1、2から作成した物体のみの画像と物体を入れたい部分を切り抜いた画像を合成



RGB画像をマスク画像として使用した場合とRGBの3要素に分けてマスク画像を作成し使用した場合とで比較すると、より綺麗な合成画像ができていることがわかります。



以前作成した合成画像



今回作成した合成画像

#### 課題

二値化する際の最適な閾値は画像によって異なり、入力画像により閾値を変える必要が出てきます。今後、どのようにして汎化性能を上げていくかが課題になってくると考えられます。

また、方法1 物体抽出における①において、「背景部分が黒く表示される場合は反転させる。」と記述しました。しかし、これはあくまで人が出来上がった二値化画像を見て判断しているため、どのようにコンピュータに上記の処理を行わせるかという問題も出てくると考えられます。

```
import cv2
import math
import numpy as np

img_src = cv2.imread("test.jpg")
img_src2 = cv2.imread("back.jpg")
height = img_src.shape[0]
width = img_src.shape[1]
img_src2 = cv2.resize(img_src2, (int(width), int(height)))

#RGB画像を色素によって分ける
img_bgt = cv2.split(img_src)

#RGBの各要素で分けて二値化を行う
img_bgtR = cv2.threshold(img_bgt[0], 231, 255, cv2.THRESH_BINARY_INV)[1]
img_bgtG = cv2.threshold(img_bgt[1], 196, 255, cv2.THRESH_BINARY_INV)[1]
img_bgtB = cv2.threshold(img_bgt[2], 214, 255, cv2.THRESH_BINARY_INV)[1]

#背景が黒くなっているものは、反転させる
img_bgtR = cv2.bitwise_not(img_bgtR)

#マスク画像の素材を作成する
#(この時、cv2.merge((img_bgtR, img_bgtG, img_bgtB)))とするとRGB画像となってしまうため別々に行う)
img_mskR = cv2.merge((img_bgtR, img_bgtR, img_bgtR))
img_mskG = cv2.merge((img_bgtG, img_bgtG, img_bgtG))
img_mskB = cv2.merge((img_bgtB, img_bgtB, img_bgtB))

#マスク画像の素材を使い切り抜く
img_msk = cv2.bitwise_and(img_src, img_mskR)
img_msk = cv2.bitwise_and(img_msk, img_mskG)
img_msk = cv2.bitwise_and(img_msk, img_mskB)

#マスク画像を作成
img_msk = cv2.cvtColor(img_msk, cv2.COLOR_BGR2GRAY)
img_msk = cv2.threshold(img_msk, 12, 255, cv2.THRESH_BINARY_INV)[1]
img_msk = cv2.merge((img_msk, img_msk, img_msk))

#入力画像から対象物を切り抜く
img_s1m = cv2.bitwise_and(img_src, img_msk)

#マスク画像を反転させる
img_msksn = cv2.bitwise_not(img_msk)

#背景画像から対象物の入る部分を切り抜く
img_s2m = cv2.bitwise_and(img_src2, img_msksn)

#合成させる
img_sdt = cv2.bitwise_or(img_s1m, img_s2m)
cv2.imshow("test", img_sdt)

cv2.waitKey(0)
cv2.destroyAllWindows()
```