進捗報告

1　今週やったこと

- 追加損失関数の検討
- 評価方法の検討
- マッチング実験

2　今週の収穫

2.1　追加損失関数の検討

先週のゼミで, 前回の損失関数は 3 つ以上の星座があるときにしか課されないため限定的かつ 3 つ以上星座を検出しないほうが有利に働いてしまう恐れがあるということだったので, 新たな損失関数を考案した. 図 1 に考案した損失関数の概要図を示す.

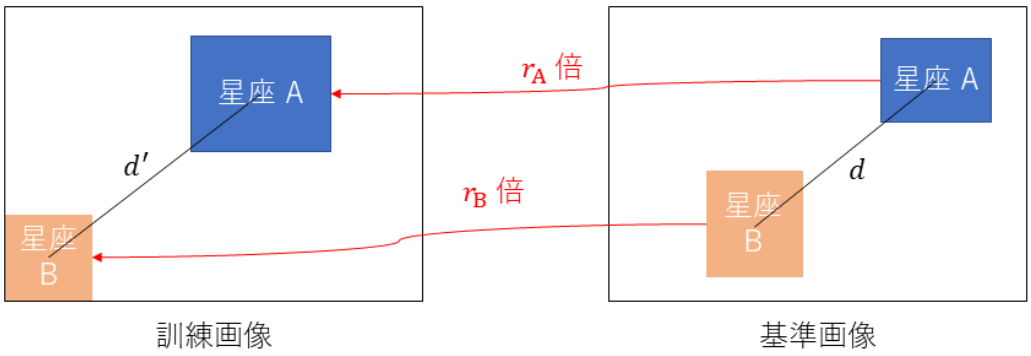


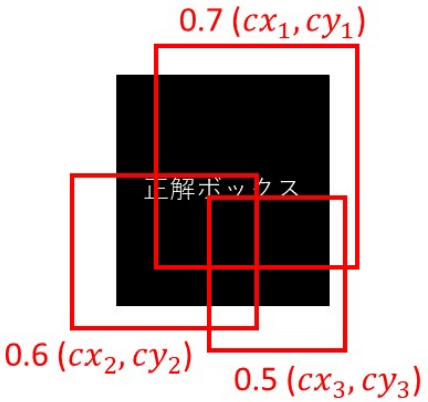
図 1: 損失関数の説明

一部しか写っていない星座があったとしても全体が写っている星座が一つでもあればそのサイズを使って本来の星座間距離を算出する. このとき星座の面積の求め方は図 2 に示す各星座の中心を求める手法と同じ方法をとった. (この方法が妥当かどうかは検討する必要があると考えている.)

また以下に数式も示す.

$$
\begin{aligned}
L(x,c,l,g,o) &= \frac{1}{N}(L_{\text{conf}}(x,c) + \alpha L_{\text{loc}}(x,l,g)) + \underline{\beta L_{\text{dis}}(x,l,g,o)} \\
L_{\text{conf}}(x,c) &= -\sum_{i\in Pos}^{N} x_{ij}^{p} \log \hat{c}_i^p - \sum_{i\in Neg}^{N} x_{ij}^{p} \log \hat{c}_i^0 \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \\
L_{\text{loc}}(x,l,g) &= \sum_{i\in Pos}^{N} \sum_{m\in cx,cy,w,h} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m) \quad \text{where} \quad \text{smooth}_{\text{L1}} = \begin{cases} 0.5x^2 & (|x| < 1) \\ |x| - 0.5 & (\text{otherwise}) \end{cases} \\
\underline{L_{\text{dis}}(x,l,g,o)} &= \sum_{a,b\in p}^{n} \left(\sum_{i\in Pos}^{N} x_{ij}^{k} \left(rd\left(\sum o_i^a l_i^a, \sum o_i^b l_i^b \right) - d\left(\hat{g}_j^a, \hat{g}_j^b \right) \right)^2 \right)
\end{aligned}
$$

このオリジナル損失関数を実装した実験は現在回している最中である. 結果が出次第報告します.



$$(X, Y) = \frac{0.7}{0.7+0.6+0.5}(cx_1, cy_1) + \frac{0.6}{0.7+0.6+0.5}(cx_2, cy_2) + \frac{0.5}{0.7+0.6+0.5}(cx_3, cy_3)$$

図 2: 各星座の中心を求める手法

2.2 評価方法の検討

検出実験において評価指標がないと有効性を示しづらいと考え, 評価指標を導入することを検討した. マッチングをするためには一つの星座だけ正確に検出できればいいが, 星座検出そのものの精度比較でも問題ないと考えたため, mAP (Mean Average Precision) を採用することを考えている.

試しにオリジナル損失関数なしのバージョンでの mAP を算出してみた. 図 3 に各星座の AP と mAP をあらわしたものを示す.

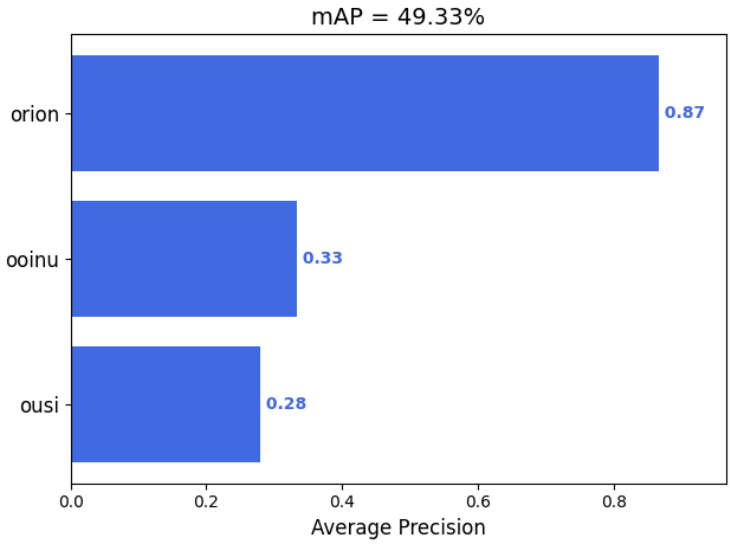


図 3: 各星座の AP と mAP

今回はふたご座が写っている星座がなかったため 3 つの星座のみの結果となっている. オリジナル損失関数ありのバージョンの実験が修了したらその場合においても mAP を算出して有効性を確認したいと考えている.

2.3　マッチング実験

先週のゼミで, マッチングはうまくいったものの若干のズレが見られる, という報告をしたが原因はカメラに使われている広角レンズであると考えられる. 広角レンズは広い範囲を撮ることができる反面画面の端が伸びるため歪みが生じやすくなってしまう.

ここで, 広角レンズの歪みを直す方法があったのでそれを試してみた. なおこの方法はチェスボードを数十枚撮影し, そこからカメラの内部パラメータや歪み係数などを測定→その値を用いて画像を修正する, という方法である. 図 4 に 100 枚の写真で修正した画像を示す.

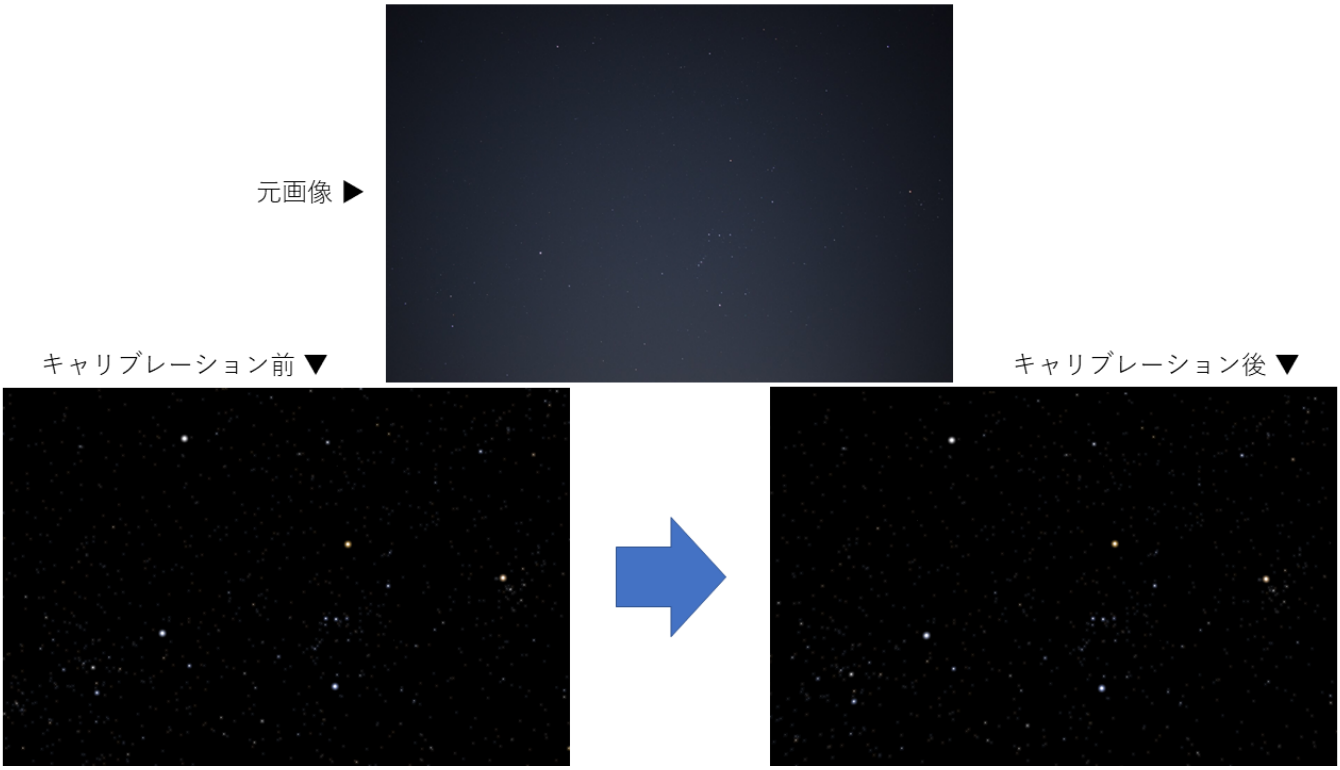


図 4: カメラキャリブレーション結果

若干の修正はされているものの, いまだズレが残る結果となった. ちなみに 50 枚の写真でもやってみたが 100 枚の方が修正量は多かった. 画像枚数を増やせばさらに修正ができるだろうがそれはアプリケーションとしてはあまり機能していないような気もする. どこまで修正する必要があるかは検討していきたいと考えている.

またアプリ化するにあたり, 実行時間も必要であると考えたため実行時間も測定してみた. 計測したところ, 全マッチングの実行時間が 2.12 秒, 類似画像の生成まで含めたプログラム全体の実行時間は 3.98 秒であった.

2.4　今後の方針

オリジナル損失関数を入れた場合の mAP を算出し, その有効性を確認する. うまくいくようであれば論文を執筆してみたい.