DMM-Net: Differentiable Mask-Matching Network for Video Object Segmentation Supplementary Material

Xiaohui Zeng[1, 2*] Renjie Liao[1, 2, 3*] Li Gu[1] Yuwen Xiong[1, 2, 3]
Sanja Fidler[1, 2, 4] Raquel Urtasun[1, 2, 3, 5]
University of Toronto[1] Vector Institute[2] Uber ATG Toronto[3]
NVIDIA[4] Canadian Institute for Advanced Research[5]
{xiaohui, rjliao, yuwen, fidler}@cs.toronto.edu li.gu@mail.utoronto.ca urtasun@uber.com

1. Proof of Theorem 1

In this section, we prove our main result, *i.e.*, Theorem 1, based on the established convergence results of *Dykstra's algorithm* and the inexact projected gradient method for the general constrained convex minimization [2].

1.1. Convergence of Dykstra's Algorithm

First, we state the convergence result of *Dykstra's algorithm* from [1] without proof as Lemma 1 below.

Lemma 1. *Suppose K is the intersection of a finite number of closed half-spaces in a Hilbert space X. Starting with any point $x \in X$, there exists some constants $\rho > 0$ and $0 \leq c < 1$, such that the sequence of iterates $\{x_n\}$ generated by Dykstra's algorithm satisfied the inequality,*

$$\|x_n - \mathcal{P}_K(x)\| \leq \rho c^n \tag{9}$$

for all n, where $\mathcal{P}_K(x)$ is the nearest point in K to x.

1.2. Convergence of Inexact Projected Gradient Method

Now we turn to the inexact projected gradient method as described in [2] which aims at solving the following general constrained convex minimization,

$$\min_{x \in \mathbb{R}^n} \quad f(x) \qquad \text{s.t.} \quad x \in \mathcal{C}, \tag{10}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex and $\mathcal{C}$ is a general closed convex set. We only state the results which are relevant to our case, *i.e.*, f is differentiable and has Lipschitz continuous gradient with constant L_f. In particular, we have,

$$\|\nabla f(x) - \nabla f(y)\| \leq L_f \|x - y\| \qquad \forall x, y \in \mathbb{R}^n \tag{11}$$

The inexact projected gradient method is stated as below. Note that each projection step is inexact as it only requires

Algorithm 2 : Inexact Projected Gradient Method for General Constrained Convex Minimization

1: **Input:** $\delta > 0, N, x_0$
2: **For** $k = 1, 2, \ldots, N-1$:
3: $\quad z_k = x_{k-1} - \frac{1}{L_f} \nabla f(x_{k-1})$
4: $\quad$ Compute x_k such that $\|x_k - \mathcal{P}_{\mathcal{C}}(z_k)\| \leq \delta$
5: **Return** $\frac{1}{N} \sum_{k=0}^{N-1} x_k$

δ-approximation. Therefore, our proposed Alg. 1 can be regarded as a special instance of this general algorithmic framework of projected gradient method. We state the convergence result of this method from [2] without proof as Lemma 2 below.

Lemma 2. *Assuming the objective f is differentiable and has Lipschitz continuous gradient with constant L_f. Let $\{x_k\}$ be the sequence generated by the inexact projected gradient method. Given inner accuracy $\delta > 0$ and any $k \geq 0$, assuming $\|\nabla f(x^*)\| \leq L_f r_0$, then the following sublinear estimates for feasibility and suboptimality in $\hat{x}_k = \frac{1}{k} \sum_{i=0}^{k} x_i$ hold:*

$$\begin{aligned} &\|\hat{x}_k - \mathcal{P}_{\mathcal{C}}(\hat{x}_k)\| \leq \delta, \\ &f(\hat{x}_k) - f(x^*) \geq -\|\nabla f(x^*)\| \delta, \\ &f(\hat{x}_k) - f(x^*) \leq \frac{2 L_f r_0^2}{k} + 5 L_f r_0 \delta + 5 k L_f \delta^2, \end{aligned} \tag{12}$$

where $r_0 = \|x_0 - x^*\|$.

Note that the sublinear convergence rate is obtained provided that $\delta = O(\frac{1}{k})$.

1.3. Our Result

Now we are ready to prove our main result. Recall that our objective is $f(X) = \mathrm{Tr}(CX^\top)$. The gradient is

*Equal contribution.

$\nabla f(X) = C$. Therefore, any $L_f \geq 0$ is a Lipschitz constant satisfying Eq. (11). Also, our constraint set $\mathcal{C}$ is a closed convex set.

Theorem 1. *Let $r_0 = \|X^0 - X^*\|_F$ where X^0 and X^* are the initial and optimal assignment matrices respectively. Let the learning rate $0 < \alpha < \min(15r_0, r_0/\|C\|_F)$. There exists some constants $0 \leq c < 1$ and $\rho > 0$ such that the at any outerloop iteration i of Alg. 1, the error of projection $\|X^i - \mathcal{P}_{\mathcal{C}}(X^i)\|_F \leq \delta = \rho c^{N_{proj}}$ where X^i and $\mathcal{P}_{\mathcal{C}}(X^i)$ are the assignment matrix and its correct projection onto $\mathcal{C}$ respectively. Moreover, for any $0 < \epsilon < 1$, there exists a $N_{proj} \geq \log_{1/c}\left(\rho\sqrt{\frac{15K}{\alpha\epsilon}}\right)$ such that,*

$$\delta \leq \frac{\alpha\epsilon}{15r_0}$$

and after at most K iterations where

$$K = \left\lceil \frac{6r_0^2}{\alpha\epsilon} \right\rceil,$$

the output of Alg. 1 $\hat{X}$ is ϵ-optimal, i.e., *$\|\hat{X} - \mathcal{P}_{\mathcal{C}}(\hat{X})\|_F \leq \epsilon$ and $|\operatorname{Tr}\left(C\hat{X}^\top\right) - \operatorname{Tr}\left(CX^{*\top}\right)| \leq \epsilon$.*

Proof. At outer step i of Alg. 1, we run *Dykstra's algorithm* for N_{proj} inner steps. We know from Lemma 1 that there exists a $\rho_i > 0$ and $0 \leq c_i < 1$ such that,

$$\|X^i - \mathcal{P}_{\mathcal{C}}(X^i)\| \leq \rho_i c_i^{N_{\text{proj}}} \tag{13}$$

Denoting $\delta_i = \rho_i c_i^{N_{\text{proj}}}$, $\rho, c = \operatorname{argmax}_{(\rho_i, c_i), i=1,\ldots,N_{\text{grad}}} \delta_i$, and $\delta = \rho c^{N_{\text{proj}}}$, we have,

$$\max_{i=1,\cdots,N_{\text{grad}}} \|X^i - \mathcal{P}_{\mathcal{C}}(X^i)\|_F \leq \delta, \tag{14}$$

where $\rho > 0$ and $0 \leq c < 1$.

Since δ is monotonically decreasing w.r.t. N_{proj}, there exists $N_{\text{proj}} \geq \log_{1/c}\left(\rho\sqrt{\frac{15K}{\alpha\epsilon}}\right)$, for any $0 < \epsilon < 1$, such that,

$$\delta \leq \frac{\alpha\epsilon}{15r_0}. \tag{15}$$

Our objective function $f(X) = \operatorname{Tr}(CX^\top)$ is linear and constraint set $\mathcal{C}$ is closed and convex. Therefore, our Alg. 1 is an instance of the inexact projected gradient method. Moreover, since any $L_f \geq 0$ is a Lipschitz constant satisfying Eq. (11) in our case, we set $L_f = \frac{1}{\alpha}$ where α is is the learning rate and $0 < \alpha < \min(15r_0, \frac{r_0}{\|C\|_F})$. Now since all assumptions of Lemma 2 are satisfied, we apply it to our algorithm and obtain that,

$$\begin{aligned}
&\|\hat{X} - \mathcal{P}_{\mathcal{C}}(\hat{X})\|_F \leq \delta, \\
&f(\hat{X}) - f(X^*) \geq -\|C\|_F\delta, \\
&f(\hat{X}) - f(X^*) \leq \frac{2r_0^2}{K\alpha} + 5\frac{r_0}{\alpha}\delta + 5\frac{K\delta^2}{\alpha},
\end{aligned} \tag{16}$$

where the K-step output of our Alg. 1 is $\hat{X} = \frac{1}{K}\sum_{i=0}^{K-1} X^i$ and we use the fact that $\nabla f(X^*) = C$.

If $K \geq \frac{6r_0^2}{\alpha\epsilon}$, then we have,

$$\frac{2r_0^2}{K\alpha} \leq \frac{\epsilon}{3}. \tag{17}$$

Due to Eq. (15), we have,

$$5\frac{r_0}{\alpha}\delta \leq \frac{\epsilon}{3}. \tag{18}$$

Since $N_{\text{proj}} \geq \log_{1/c}\left(\rho\sqrt{\frac{15K}{\alpha\epsilon}}\right)$, we have,

$$\begin{aligned}
5\frac{K\delta^2}{\alpha} &= 5\frac{K}{\alpha}(\rho c^{N_{\text{proj}}})^2 \\
&= 5\frac{K}{\alpha}\left(\sqrt{\frac{\alpha\epsilon}{15K}}\right)^2 \\
&\leq \frac{\epsilon}{3}.
\end{aligned} \tag{19}$$

Therefore, with Eq. (17), Eq. (18) and Eq. (19), we prove that $f(\hat{X}) - f(X^*) \leq \epsilon$.

For the lower bound, from Eq. (16) we have,

$$\begin{aligned}
f(\hat{X}) - f(X^*) &\geq -\|C\|_F\delta \\
&\geq -\|C\|_F\frac{\alpha\epsilon}{15r_0} \\
&\geq -\frac{\epsilon}{15} \\
&\geq -\epsilon,
\end{aligned} \tag{20}$$

We prove the ϵ-optimality w.r.t. the objective function.

Again, from Eq. (16), we have,

$$\begin{aligned}
\|\hat{X} - \mathcal{P}_{\mathcal{C}}(\hat{X})\|_F &\leq \delta \\
&\leq \frac{\alpha\epsilon}{15r_0} \\
&\leq \epsilon.
\end{aligned} \tag{21}$$

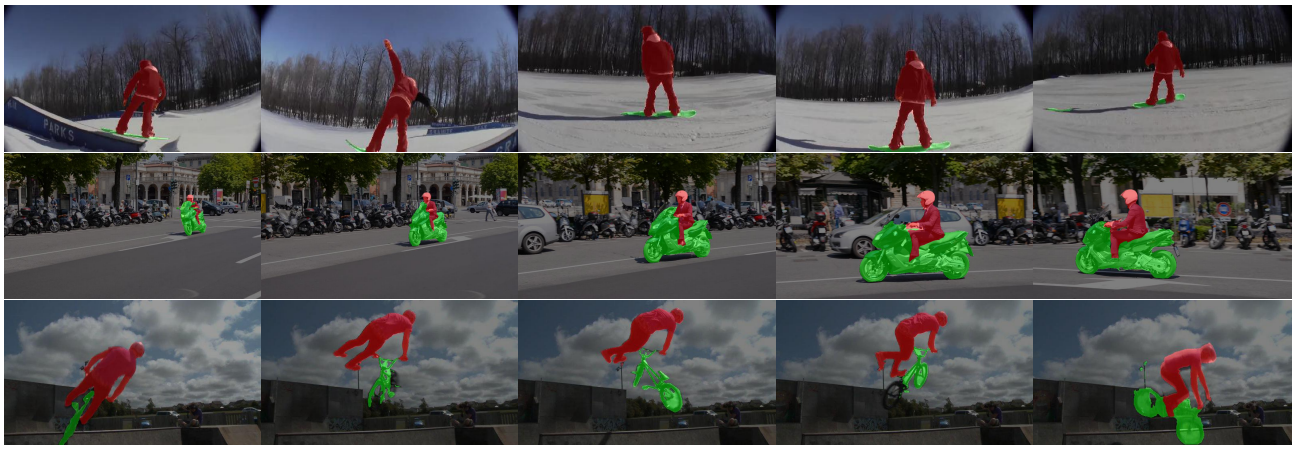
We prove the ϵ-optimality w.r.t. the feasibility.

□

Note that the constants (*e.g.*, 6, 15) in Theorem 1 do not matter that much as the inequality still holds by properly changing the constants in Lemma 2 as discussed in [2].

2. Additional Experiments

We show the hyperparameters of our matching algorithm on random cost matrices in Fig. 5. The random cost matrices have 5 rows and 100 columns, where the values are sampled from a uniform distribution over $[0, 1)$. Three random cost matrices are generated for the experiment. We plot the objective function value of the outer loop and the



(a) 0% (b) 25% (c) 50% (d) 75% (e) 100%

Figure 4. Visualization of our results on YouTube-VOS, DAVIS 2017 and SegTrack v2 at different time steps (percentage w.r.t. the whole video length). The three rows correspond to the YouTube-VOS, DAVIS 2017 and SegTrack v2 datasets respectively.

projection error of the inner loop under different sets of hypperparameters including number of steps of outer loop N_{grad}, number of steps of inner loop N_{proj} and the learning rate α. We choose the configuration which gives the best trade-off between the computational cost and the convergent rate: $N_{\text{grad}} = 400$, $N_{\text{proj}} = 50$, and $\alpha = 0.01$. From the figure, one can see that the objective value decreases as the number of outer loops increases and it reaches the minimum at about $N_{grad} = 400$ under different setting and different cost matrices. The inner projection error drops in a faster rate and in most of the case it is able to reach almost zero after about 50 rounds of projection. The figure also shows that the learning rate plays an important role for the optimization, *i.e*., learning rate 0.01 generally leads to a faster convergence than learning rate 0.005. For the experiments on video object segmentation dataset, we start with $N_{\text{grad}} = 400$, $N_{\text{proj}} = 50$ for both training and inference. However, we reduce them to 40 and 5, respectively, for the trade off of performance and speed.

We show additional visual examples of our method on three datasets in Fig. 4.

3. Example Code

We show the example PyTorch code (less than 50 lines) of our differentiable matching algorithm as below.

References

[1] F. Deutsch and H. Hundal. The rate of convergence of dykstra's cyclic projections algorithm: The polyhedral case. *Numerical Functional Analysis and Optimization*, 15(5-6):537–565, 1994. 1

[2] A. Patrascu and I. Necoara. On the convergence of inexact projection primal first-order methods for convex minimization. *IEEE Transactions on Automatic Control*, 63(10):3317–3329, 2018. 1, 2

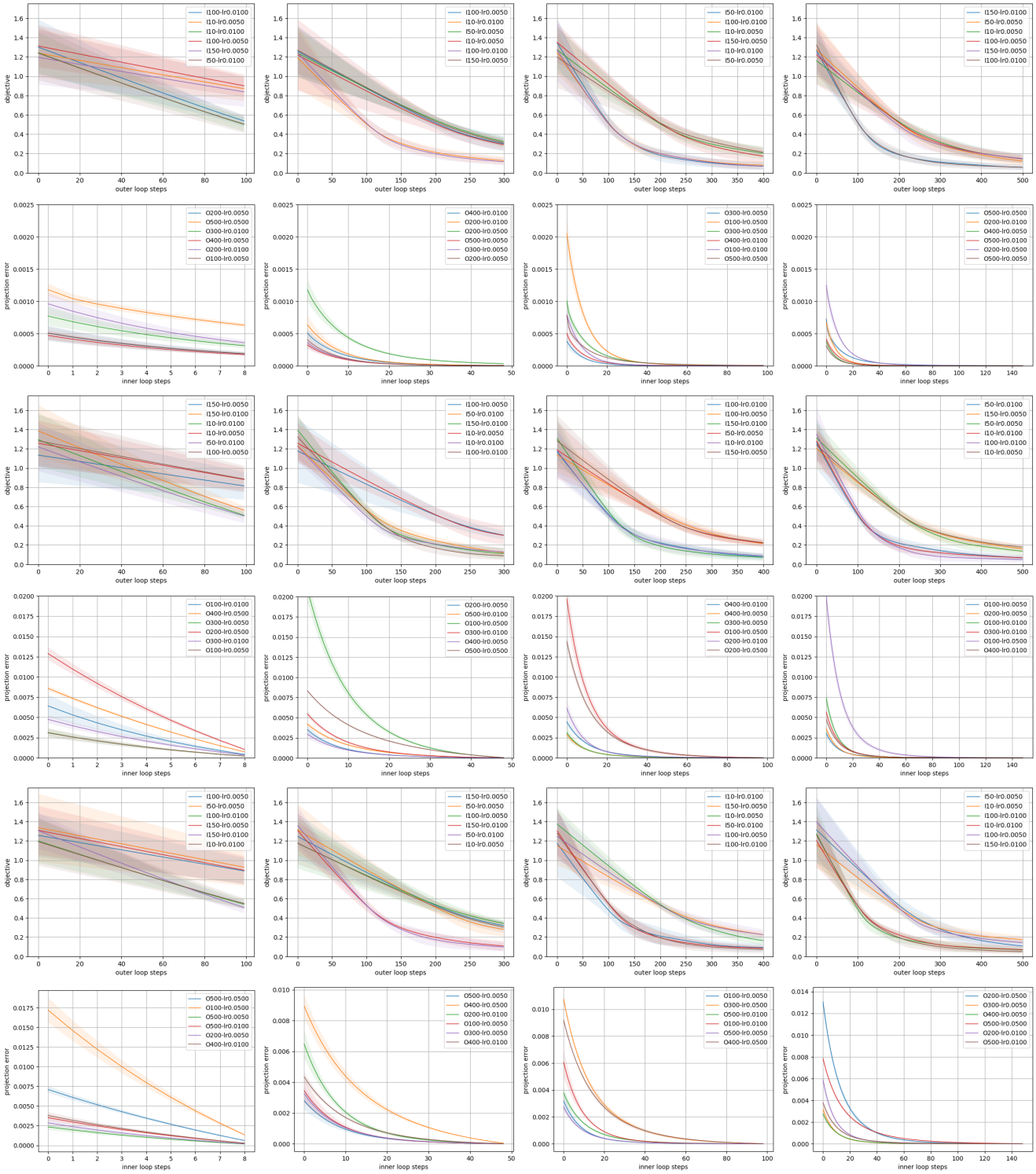


Figure 5. Hyperparameters (number of steps of outer loop N_{grad}, number of steps of inner loop N_{proj}, and learning rate α) tuning of our differentiable matching layer. We test with three random cost matrices, each of which corresponds to two consecutive rows in the figure, *e.g.*, 1-st and 2-nd rows correspond to the 1-st cost matrix. Within the two consecutive rows, the first one shows how the objective function varies with the number of steps in outer loop. The second one shows how projection error varies with the number of steps in inner loops. For each setting of hyperparameters, we perform 10 different random initialization of the assignment matrix and plot the mean and variance of the curves.

```python
def project_row(X):
    """
        p(X) = X - 1/m (X 1m - 1n) 1m^T
        X shape: n x m
    """
    X_row_sum = X.sum(dim=1, keepdim=True) # shape n x 1
    one_m = torch.ones(1, X.shape[1]).to(X.device) # shape 1 x m
    return X - (X_row_sum - 1).mm(one_m) / X.shape[1]

def project_col(X):
    """
        p(X) = X  if X^T 1n <= 1m else X - 1/n 1n (1n^T X - 1m^T)
        X shape: n x m
    """
    X_col_sum = X.sum(dim=0, keepdim=True)      # shape 1 x m
    one_n = torch.ones(X.shape[0], 1).to(X.device) # shape n x 1
    mask = (X_col_sum <= 1).float()
    P = X - (one_n).mm(X_col_sum - 1) / X.shape[0]
    return X * mask + (1 - mask) * P

def relax_matching(C, X_init, max_iter, proj_iter, lr):
    X = X_init
    P = [torch.zeros_like(C) for _ in range(3)]
    X_list = [X_init]

    for i in range(max_iter):
        X = X - lr * C # gradient step

        # project C onto the constrain set
        for j in range(proj_iter):
            X = X + P[0]
            Y = project_row(X)
            P[0] = X - Y

            X = Y + P[1]
            Y = project_col(X)
            P[1] = X - Y

            X = Y + P[2]
            Y = F.relu(X)
            P[2] = X - Y

            X = Y

        X_list += [X]
    return torch.stack(X_list, dim=0).mean(dim=0)
```

Listing 1. PyTorch example code