Supplementary Document : *Deep Model Predictive Control for Visual Servoing*

1 Overview

In this document, we present a detailed analysis of few crucial components of our approach to demonstrate their efficacy and establish the rationale behind the improvement in performance claimed in this work. We also present an additional quantitative and qualitative study of our pipeline design along with few more comparisons with state-of-the-arts in addition to the ones present in the main manuscript.

Section 3 contains the ablation studies of the proposed architecture with respect to varying the LSTM sequence length 3.1 and varying the number of the LSTM units 3.2. Section 4 presents an in-depth analysis of the flow loss $\mathscr{L}_{flow}$ used in our approach.

We conduct extensive experimentation on Habitat [1] which allows us to evaluate our controller in a photorealistic 3D simulation and gauge generalization capability across environments. Our controller achieves state of the art results in this rich realistic simulator. We report some more qualitative results of our architecture on the test benchmark in section 7, which showcases the efficiency of our controller when compared to the other state of the art visual servoing techniques. We further describe the training procedure of DDFlow [2] in section 5.2 that we utilise for an extension to a purely unsupervised architecture. Next, we evaluate and compare a reinforcement learning baseline on our benchmark in section 6. We also report our findings by incorporating actuation noise in section 2 to expose our controller to the stochasticity of the real world. We achieve convergence on our test scenes, showing promise of transference to real world platforms.

2 Generalisation to Real-Life Robots

The strict lockdown policies that are being followed at our university due to COVID-19 made it difficult to have access to any of the experimental hardware platforms. As we are unable to demonstrate the efficacy of our approach in a real world setup on a robotic platform, we try to prove that our approach can be tested in a real world setting without any modification or additional tuning.

As all of our experiments were performed in simulation environments, replicating them in a real-world setting on a real robot poses two additional challenges. Firstly, in a real-world setting, the visual inputs to the robot has additional noise. Secondly, in a real-world setting, the robotic hardware suffers with a noise in actuation. We have tried to emulate both of these conditions by adding noises and we show that our pipeline adapts quite well.

In a real-world setup, robot commands are noisy. To simulate this, we add a Gaussian noise in the Habitat environment with mean=0.0 and standard deviation=0.1m in all 6-DoF to the control commands. We perform test on the *Mesic* environment from the simulation benchmark and show that our pipeline is able to converge to a photometric error < 500 in 683 iterations in the presence of such large noise, as compared to 504 iterations without noise. This showcases the ability of our approach to handle actuation noise and generalize in real world environments.

Seq	Iters	PE	Time
5	**531**	476	**0.8**
10	694	**450**	2.2
20	757	486	4.7
50	1000	8456	5.1

(a) Effect of varying seq length

LSTMs	Iters	PE	Time
1	1000	1693	**0.4**
3	913	**369**	0.5
5	**531**	476	0.8
10	757	486	1.3

(b) Effect of varying number of LSTM units.

3 Ablation Studies

3.1 Varying the sequence length

The sequence length is a tuning parameter that balances between speed and accuracy. To identify a suitable value of sequence length, we evaluated our approach for various sequence lengths on the *Stokes* scene from our benchmark. It could be seen from table 1a that for smaller sequence lengths, our approach is able to converge. However, the recurrent network faces issues for larger sequence lengths due to unstable gradients, which is a general issue of recurrent networks. Thus for our other experiments, we empirically select a sequence length of 5, optimising over computation time.

3.2 Varying the depth of the network

To design the architecture of our control network, we select a stack of LSTMs, due to their efficient backpropagation of gradients through time. For identifying the number of LSTM units required to achieve optimal performance, we evaluate our approach with a variable number of LSTM units. It could be seen from table 1b, that a single LSTM is not sufficient to achieve a precise alignment. As we increase the number of LSTMs to 3 units, the network is able to converge but takes larger number of iterations to converge. An LSTM stack with 5 units gives us the correct balance between speed and convergence and thus, we select this number for the rest of our experiments. Here again, we see that having a deeper network not only increases the speed but also reduces precision.

3.3 Stability Control - A comparison with [3]

In this experiment we try changing the $\lambda(0.01)$ reported in [3] by dividing it by 2, 5 and multiplying it by 2. We have found that their [3] controller performance depends a lot on the value of λ and hampers the convergence of their pipeline. When we change λ by dividing it by 2 the pipeline takes more time to converge, dividing it by 5 made it converge faster and when we multiply it by 2 it fails to converge as shown towards the right side in Fig. 1. This is in stark contrast to our approach where the Deep Controller learns the velocity scale by itself.

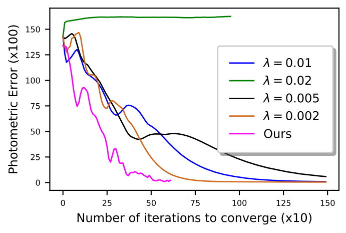

Figure 1: **Photometric error (vs) Number of Iterations**: - Plotting different *lambda* values from [3] hard benchmark scenes

4 Discussion on the Flow Loss

In our approach, flow loss $\mathcal{L}_{flow}$ acts as the primary guiding principle driving the entire pipeline to convergence. This plays a pivotal role in our approach and in order to showcase its significance and provide a deeper understanding of this principle, we plot the evolution of photometric error(the conditional parameter) and the flow loss(the guiding parameter) with time. We can comprehend from Fig. 2 that the reduction in photometric error leads to a simultaneous reduction in flow loss, thereby demonstrating the ability of our loss function to implicitly capture the photometric error.

We notice that the flow loss and the photometric error shows almost similar trends with time initially. It is also interesting to note that towards the later part of the evolution timeline, both the quantities show indistinguishable behaviour which validates our choice of using $\mathcal{L}_{flow}$ as the loss function.

5 Training Details

5.1 Online Training with Flownet-2.0 [4]

The input to the Deep MPC network is a randomly initialized 6-DoF vector which signifies 3 translational and 3 rotational velocities. We compute flow between the initial image and the desired image using Flownet 2.0 [4] denoted by $\mathscr{F}(I_{init}, I*)$. This target flow is used to train our control network online in a supervised fashion. In this work we have empirically selected our control network as a stack of 5 LSTM units each having dimensionality 6. We train our network for 100 iterations in a stateful manner as our batch size is 1 during each MPC step. Thus, for subsequent MPC steps the network is not reinitialised from random values. We observed a faster convergence with this strategy.

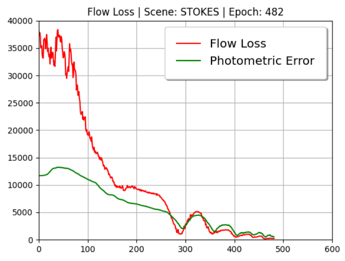

Figure 2: Plot showing evolution of flow loss and photometric error with time

After the inner MPC loop is trained for 100 iterations, we now take the velocity predicted by our network and apply it to the agent to get the next image I_t. Subsequently, we repeat this process of training the network for another 100 iterations before moving the agent again. This process of online training is continued till we reach convergence (photometric error less than 500).

5.2 Online Training with Unsupervised Flow [2]

To showcase that our network is agnostic to the flow encoder and could also be trained in a fully unsupervised manner with just images, we trained an unsupervised flow network [2] on a couple of scenes from the simulation benchmark using the curriculum learning approach. We used the pretrained FlyingChairs model of [2] to provide a warm start and fine tune over the scenes. Since we do not provide any ground truth while training, it would be difficult for the model to learn the flow between frames that do not have a significant overlap. If we were to train on frames that were remarkably far apart, the model would not be able to learn much. Hence, we trained DDflow [2] 'step-by-step' in a curriculum learning fashion and we consistently kept decreasing the overlap between consecutive frames.

We first trained on consecutive image frames that had an appreciable overlap between them, for 16k iterations. We then took the 'next-step' in the curriculum pipeline and decreased the overlap, and trained it again for 16k iterations. Hence, the network is able to predict the optical flow even for larger camera transformations by continuously learning.

In the first stage, we trained for 16k iterations for frame pairs [$\mathscr{F}_i$, $\mathscr{F}_{i+1}$], [$\mathscr{F}_{i+1}$, $\mathscr{F}_{i+2}$] and so on. We then skipped a frame in between while going up the curriculum pipeline and trained the model for 16k iterations for frame pairs [$\mathscr{F}_i$, $\mathscr{F}_{i+2}$], [$\mathscr{F}_{i+2}$, $\mathscr{F}_{i+4}$] and so on. Had we done the latter without the former, it would have been difficult to learn the flow. But since the network had already learned weights for consecutive frames, it became easier for it to learn the features for subsequent frames.

6 Comparison with Reinforcement Baseline

Here we provide a comparison with Reinforcement Learning (RL) Baseline [5] visual navigation approach. We report comparison with Zhu et al. [5] in isolation since RL methods only report results in 2-DoF (surge, heading). On the other hand, through our approach, we were able to achieve an accurate 6-DoF pose. Another issue is the resolution of action space; while Zhu et al. [5] take large translation actions of $0.25m$ and 10^o, we report a precise alignment of under $0.03m$, 0.6^o which results in continuous action and state space. RL approaches have high sampling space and large trajectories to explore based on the size of the action space. We evaluate [5] with three different setting varying state space resolution $(0.1m, 0.01m)$ and dimensionality of the action space (2-Dof, 3-Dof). We train [5] the scene *Ballou* from our benchmark for 200K iterations and evaluate the trained policy for 100 episodes on the same environment. The agent is spawned at a random pose within a translation error of $1m$. We use a similar reward as described by [5], while only assuming collision with the environment's boundary (the environment boundary is kept just $1m^3$, as we are only interested in a local policy). The results are shown in table 2. It can be seen from the table the the policy easily reaches the goal in case of 2-Dof and large discretization of the state space, while it fails to learn to reach the desired goal image in case of 3-DoF, as the sampling space grows exponentially. This showcases the limitations of existing end-to-end model free RL approaches.

Test #	Dimensionality	Resolution	Convergence	Avg. reward	Avg. steps
1	2-Dof	0.1m,3^o	89%	8.72	9
2	2-Dof	0.01m,1^o	72%	6	41
3	3-Dof	0.01m	0%	-	-

Table 2: **Performance of RL baseline for visual navigation**: Model free RL baselines face challenges while navigating in larger continuous state space in higher DoFs. Whereas we showcase precise convergence in 6-DoF.

7 Additional qualitative results

Qualitative results for rest of the scenes in the benchmark are shown in figure 3. It can be seen from the figure that all the variants of our approach are capable of easily converging on all the scenes of the benchmark. The first two rows show the initial and desired image that the agent needs to navigate to. The subsequent rows show the difference in the desired image and the final image attained by the approach.

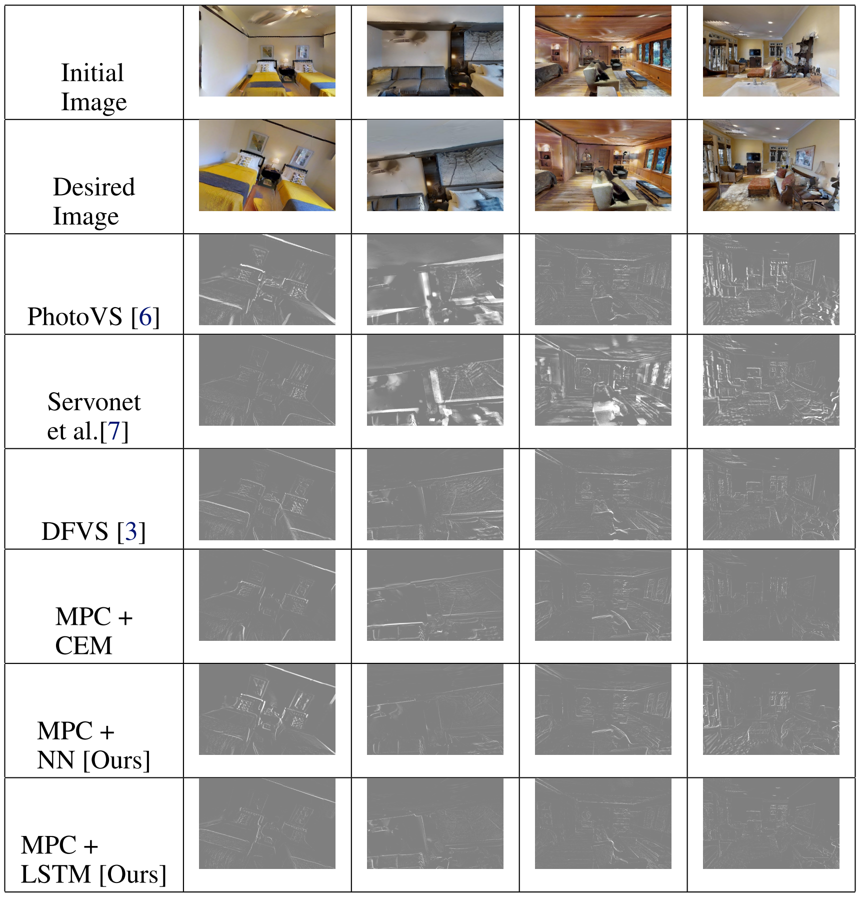

Figure 3: **Additional Qualitative Comparison**: Here, we show qualitative results for rest of the scenes from the benchmark. It can be seen from the error images (gray color denotes zero error in pixels) that while classical approaches like PhotoVS [6] and supervised learning based approaches such as Servonet [7] do not converge, DFVS and all variants of our approach (NN, CEM and LSTM) do converge. Note that, using CEM with a predictive model was also proposed by [8]. However, their predictive model was different and required accurate odometry while training. On the contrary, we present CEM results with our predictive model instead.

References

[1] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[2] P. Liu, I. King, M. R. Lyu, and J. Xu. Ddflow: Learning optical flow with unlabeled data distillation, 2019.

[3] Y. V. S. Harish, H. Pandya, A. Gaud, S. Terupally, S. Shankar, and K. M. Krishna. Dfvs: Deep flow guided scene agnostic image based visual servoing. In *IEEE ICRA*, pages 3817–3823. IEEE, 2020.

[4] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks, 2016.

[5] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning, 2016.

[6] C. Collewet and E. Marchand. Photometric visual servoing. *IEEE TRO*, 27(4):828–834, 2011.

[7] A. Saxena, H. Pandya, G. Kumar, A. Gaud, and K. M. Krishna. Exploring convolutional networks for end-to-end visual servoing. In *IEEE ICRA*, pages 3817–3823. IEEE, 2017.

[8] C. Finn and S. Levine. Deep visual foresight for planning robot motion, 2016.