COL780 Assignment 2
Template Tracking

Aditi Khandelwal (2018EE10434) Shubham Mittal (2018EE10957)

Introduction

In this assignment, we discuss the traditional non-learning-based template tracking techniques to track specific templates/objects in a video sequence. The systems we describe here need no prior training and they may not see the template prior to tracking.

The evaluation metric used to judge the goodness of the algorithms is mIOU (mean Intersection-over-union) score which is given as:

$$mIOU = \frac{1}{N-1}\Sigma\frac{Area\ of\ overlap\ of\ prediction\ with\ the\ bounding\ box}{Area\ of\ union\ of\ prediction\ and\ bounding\ box}$$

The results presented in this report are for 3 datasets, namely BlurCar2, Bolt, and Liquor.

Note: there are two bounding boxes shown in this report. The red ones are the ground truths and the blue ones are the predictions by the specified model.

PART-1: Appearance Based Techniques

We implement block-based matching algorithm with SSD and NCC distance metrics. The mIOU scores are shown below:

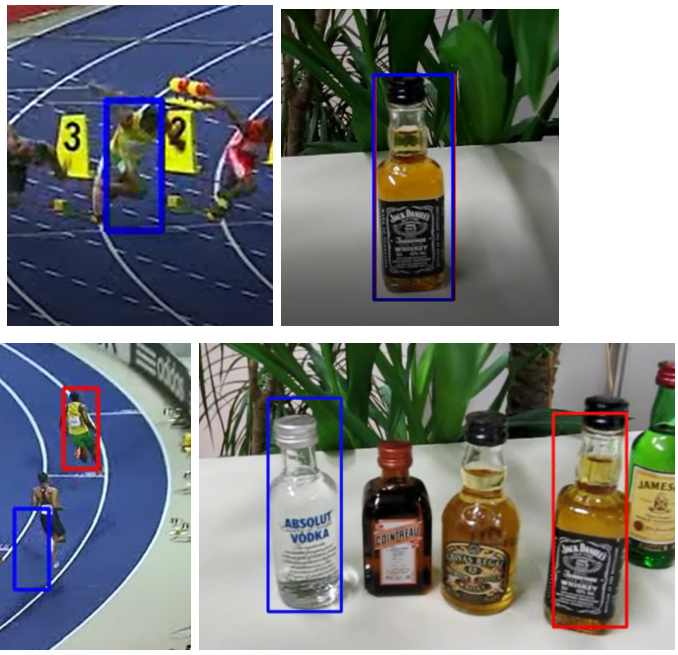
Dataset	SSD	NCC
BlurCar2	0.43	**0.57**
Bolt	0.04	**0.07**
Liquor	0.62	**0.64**

Discussion:

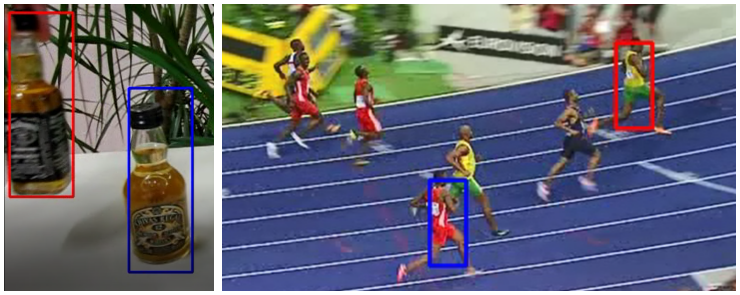
1. The empirical results clearly state that block-based matching with NCC is the best performing.
2. Since this method searches for the corresponding patch in the whole image, this method is robust to large motions.
3. The computation time was also reasonably less (~1min).

Problems observed in block-based matching:

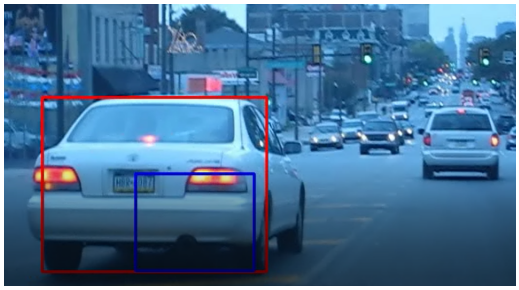
1. **Orientation variant:** Not robust to orientation changes of the object/template. The following images show that the model gave wrong predictions because the orientation of the ground-truth object changed. The top two images show the template used by the model for matching, and the bottom two images show the failed instance.



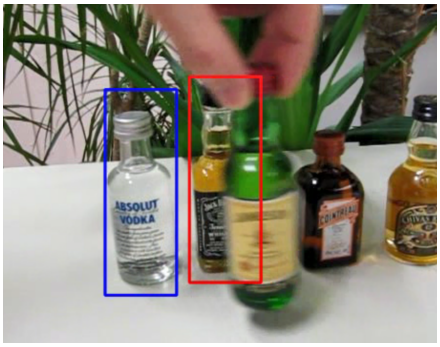
2. **ID Exchange (loosely):** When there are multiple objects in the image which look similar to that of the template, then this method starts to track other objects as shown below:



3. **Scale variant:** this method gives the wrong prediction when the scale of the concerned template/object changes i.e. it becomes big or small. This is so because the method uses a fixed size template for matching.




4. **Occlusion:** the model is also not robust to occlusion as shown in the below frame:



PART-2: Lucas-Kanade Algorithm

We implement the Lucas Kanade algorithm for template tracking. The mIOU scores are shown below:

Dataset	Translation	Affine	Projective
BlurCar2	**0.33**	0.13	0.34
Bolt	**0.15**	0.32	0.03
Liquor	**0.53**	0.53	0.45

Discussion:

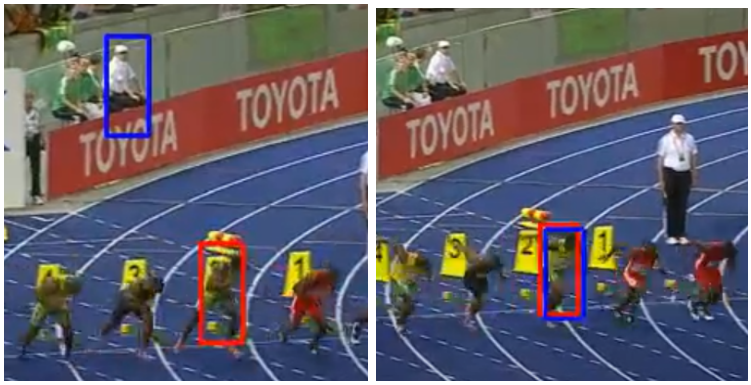
The empirical results clearly state that translation transformation outperforms the rest. We emphasize on the results of LK and state the problems with it (along with **methods tried**) in the following points:

1. **The problem of convergence**: the LK algorithm is very hard to converge. We used different metrics of convergence like:
    a. Norm of Δp
    b. SSD between the template and warped image (patch)

2. This problem of convergence is most likely the reason for less scores obtained using affine and projective transformations.
3. **Compositional implementation:** we also implemented a different update rule (as discussed in the lecture). Instead of "$p + \Delta p$", we use "$W(p).W(\Delta p)$" to find the predictions i.e. cascade or composition of transformations.
4. **Updating the template** with the last image also didn't help in convergence (and gave even lesser scores) so we use a fixed template.

Comparison with Part-1:

Visual comparison of part-1 (left) with part-2 (right):



1. Computation time is almost 5 times more than block-based matching. This stems from the iterations required per frame for convergence of the LK algorithm.
2. Visual inspection and the mIOU score comparison between Part-1 and Part-2 shows that LK is worse than block-based matching (except on Bolt). Theoretically, we don't expect this but this is because (in practice) the LK algorithm is finding it hard to converge on the given datasets.
3. The possible reasons for the failure of (convergence of) LK algorithm are:
    a. The initial estimate of p: we initialize p such that warp matrix is identity (for all three transformations)
    b. Temporal consistency: it is possible that the consecutive frames are not near in time (as required by the first-order approximation in Taylor series of LK algorithm)

PART-3: Pyramid based Lucas-Kanade Algorithm

We implement the Pyramid-based Lucas Kanade algorithm for template tracking. The mIOU scores comparison between all the parts of the assignment are shown below:

Dataset	Appearance-based technique	Lucas Kanade Algorithm	Pyramid-based LK algorithm
BlurCar2	**0.57**	0.33	0.34
Bolt	0.07	0.15	**0.17**
Liquor	**0.64**	0.53	0.54

Discussion:

1. The empirical results show that using the multi-scale approach the scores do improve as compared to just the LK algorithm.
2. Thus, the algorithm is closer to convergence but still far from being converged (as observed in plots of Δp).
3. **Optimal number of pyramid levels:** we found them to be 5 for all the datasets with the iterations for each level as [1,5,7,10,20].
4. Computational time is significantly reduced as compared to part-2 despite the total number of iterations in part-3 being three times that of part-2. This is because of the multi-scale approach. Also, part-3's computational time is comparable with that of part-1.

Comparison with Part-2:

Visual comparison of part-3 (left) with part-2 (right):

The mIOU score comparison and visual inspection between part-2 and part-3 shows that there are slight improvements in the prediction quality, and significant improvement in the computational time.

Part-4: Live Tracking

The live-tracking framework (livetracker.py) is submitted along with other deliverables (main.py, eval.py, utils.py)

References

1. Lucas-Kanade 20 Years On: A Unifying Framework