Spatial Transformer Network for Matching Players: Find the Easier Matches

Mengyao Zhai
SFU
mzhai@sfu.ca

Lei Chen
SFU
chenleic@sfu.ca

Abstract

Tracking is a fundamental problem in Computer Vision. In the context of tracklet alignment problem, the common approach is to build an affinity model and then optimize the association globally. Unlike global association optimization which has been paid wide attention to, the affinity model has received little attention. Many people use multiple cues extracted from the whole trackelts to align tracklets, while we consider this problem in a different way. Instead of take the whole tracklets into consideration, we prefer to look at tracklet segments which are easier to match. To accomplish this, we build a STN/Siamese network architecture, which aims at automatically regress the desired segments. This affinity model can be applied to many applications, such as multi-object tracking (MOT), tracklet alignment, player re-id in sports videos, etc.

1. Motivation

In this project, we focus on player association task in sport scenarios, more specifically in hockey videos. Alignment of multiple objects in the sport court is very different from standard MOT. The first challenge is that there are frequent shot switches and actions zooming in and out. The second challenge is that all objects are dressed in same uniform which makes the appearances not as diverse as standard MOT. The third challenge is that sometimes objects are moving very fast and result in severe motion blur. To address the above problem, we need a strong affinity model which use better features to distinguish among objects in the scene. We come up with a deep learning based affinity model. The CNN contains two architectures. The first architecture is a siamese network which is trained to discriminate short tracklets as *same objects* or *different objects*. The second architecture adopts spatial transformer networks (STN) to localize possible matching regions.

2. Approach

2.1. Siamese Network

To align tracklets belonging to same objects, we use the siamese architecture which aiming at building an embedding space in which similar feature vectors are near to each other and dissimilar feature vector are far away. The siamese network contains a twin networks sharing weights and processes pairs of tracklets. Contrasive loss is used to train the siamese network.

2.2. Spatial Transformer Network

Most previous approaches align pairs of tracklets by using multiple cues extracted from whole tracklets. While we solve the association problem differently. The affinity model could automatically find tracklet segments which are similar to each other and tracklets are associated given the similarity measures. To accomplish this, we use the Spatial Transformer Network (STN) to perform affine transformation. The STN contains 3 parts: *localisation network*, *grid generator*, and *samplor*.

Localization Network The localization network takes the input feature map $U \in \mathbb{R}^{N \times C \times H \times W}$, where N is the trajectory length, H is the height, W is the width and C is the channel, and outputs the parameters θ of the transformation $\mathscr{T}_\theta$. The architecture of localization network can take any form, we simply use two fully connected layers as localization network. Since we want to produce parameters for affine transformation, the size of θ is 2×3.

Parameterised Sampling Grid In this phase, affine transformation is applied on the pixels of the regular grid in the output feature map $V \in \mathbb{R}^{N' \times C \times H' \times W'}$. Let output pixels be (x_i^o, y_i^o), the source pixels (x_i^s, y_i^s) can be computed by:

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{pmatrix} \begin{pmatrix} x_i^o \\ y_i^o \\ 1 \end{pmatrix} \quad (1)$$

And we use normalized coordinates such that $-1 \leq x_i^o, y_i^o \leq 1$ when within spatial-temporal bounds of outputs

and $-1 \leq x_i^s, y_i^s \leq 1$ when within spatial-temporal bounds of inputs.

Sampler After the source coordinates are calculated, the value of V_i^C is computed by doing bilinear sampling.

$$\begin{aligned} V_i^C = &\sum_n^H \sum_m^W U_{nm}^C max(0, 1 - |x_i^s - m|) \\ &max(0, 1 - |y_i^s - n|)) \end{aligned} \tag{2}$$

To backpropagate through the network, the gradients can be computed using the following equation:

$$\begin{aligned} \frac{\partial V_i^C}{\partial U_{nm}^C} = &\sum_n^H \sum_m^W max(0, 1 - |x_i^s - m|) \\ &max(0, 1 - |y_i^s - n|)) \end{aligned} \tag{3}$$

$$\begin{aligned} \frac{V_i^C}{x_i^s} = &\sum_n^H \sum_m^W U_{nml}^C max(0, 1 - |y_i^s - n|) \cdot \\ &\begin{cases} 0 & |m - x_i^s| \geq 1 \\ 1 & m \geq x_i^s \\ -1 & m < x_i^s \end{cases} \end{aligned} \tag{4}$$

and similarly we can get $\frac{V_i^C}{y_i^s}$.

Since x_i^s can be calculated as:

$$x_i^s = \theta_{11} x_i^o + \theta_{12} y_i^o + \theta_{13} \tag{5}$$

The gradients with respect to θ can be easily derived.

3. Experiments

We designed several baselines to compare against.

baseline-1 Our first baseline is a stn-free model, which is a siamese network. Each arm is an AltexNet. Given a tracklet, features extracted from each frame of the tracklet are concatenated as the feature descriptor.

baseline-2 Our second baseline is a stn-free model, which is a siamese network. Each arm is an AltexNet. Given a tracklet, instead of concatenating features extracted from each frame of the tracklet, we use LSTMs to model the temporal information.

baseline-3 Our third baseline is STN/siamese network with simple CNN structure: we keep the AlexNet layers up to 'conv3'. We use spatial transformer neural network parameterised for attention.

baseline-4 Our fourth baseline is a STN/siamese network with relative complex CNN structure: each arm of the siamese network is an Alexnet, while other settings are the same as the third baseline.

our approach And our final version contains two stn components in each siamese arms. The spatial transformer neural network is not parameterised for attention anymore, but arbitrary affine transformation.

For testing, given a tracklet starting in frame t, we find the aligned tracklet starting from frame $t + 6$ which is the nearest neighbor of the tracklet. If the aligned tracklets belong to same player, we call it correct alignment. Otherwise, it is an incorrect alignment.