Transfer Learning Based Compressive Tracking

Shu Tian, Xu-Cheng Yin, Xi Xu, and Hong-Wei Hao

Abstract—Although existing online tracking algorithms can solve the problems of scene illumination changes, partial or full object occlusions, and pose variation, there are still two weaknesses, inadequacy of training data and drift problem. Considering these, Compressive Tracking algorithm (CT) [1] extracts features from compressed domain, and classified object and background via a naive Bayes classifier with online update. To further solve the problems of drift and inadequacy of training data, we introduce transfer learning into CT to take full advantage of prior information and propose a self-training-like transfer learning algorithm. It selects training samples from samples collection to update classifier by the conduction of the classifier constructed at first frame. Eventually we introduce self-training-like transfer learning algorithm into CT to construct a novel tracking algorithm called Transfer Learning based Compressive Tracking (TLCT). Experimental results on 17 publicly available challenging sequences have shown the effectiveness and robustness of our algorithm.

I. INTRODUCTION

VISUAL tracking is an important area in computer vision. Video surveillance, robotics, video indexing and human-computer interaction are benefit from the solutions of visual tracking [2]. According to whether algorithm uses transfer learning to transfer results in previous frame or not tracking algorithms are categorized into offline tracking and online tracking. Off-line algorithms train object model offline and never update, such as EigenTracking [3], SVT [4], and Frag-Track [5]. As not updated online, they cannot deal with scene illumination changes and pose variation. Moreover, it is necessary to know much information about object before tracking and collect a large number of samples of various views as the algorithms train template offline. Thus, online tracking algorithms attract much attention these years [6]-[13]. Online trackers have the ability to adapt to the appearance changes of object. Ross et al. [10] proposed Incremental Visual Tracking (IVT) algorithm in which low-dimensional subspace representation is learned to represent object and updated incrementally when object localization of new frame is obtained. Grabner et al. [11] formulated tracking problem as a classification problem. They used an adaboost classifier to discriminate object from background and updated it by the tracking result. Tang et al. [12] proposed a novel approach in which recognition object from background and updating classifiers on-line are in a co-training framework. Though online tracking algorithm has many advantages, there are two important problems, namely, drift and inadequacy of training data. Focusing on robust means of updating appearance model, Babenko et al. [13] utilized online Multiple Instance Learning (MIL) to reduce drift. To alleviate drift, Kalal et al. [14] proposed P-N learning algorithm to ensure the high confidence of data used to update classifier. There are many works on drift, but drift problem has not been completely solved yet. Online tracking algorithms assume that no information other than object location in first frame is known, so there are few training data available.

Transfer learning is a new framework to address classification problem [15]. It is a major assumption in traditional machine learning algorithm that training and test data are in the same feature space and follow the same distribution. However, it does not always hold. Transfer learning could solve this problem to some extent. Its critical idea is that transfer knowledge from source domain to target domain. We follow the notations and definitions in [15].

Track using transfer learning attracts some attention in recent years. Luo et al. [16] exploited transfer learning framework TrAdaBoost [17] to track object. The main idea is to transfer information from previous frame to current data. Wang et al. [18] collected generic real-world images which is represented by overcomplete dictionary and then transferred prior information by sparse coding and multiscale max pooling. Finally, they used an online updated classifier to discriminate object and background and particle filter to estimate object location. Rosenstein [19] carried out many experiments to show that transfer learning can also hurt performance when sources of data are too dissimilar, which is called negative transfer, the main problem in transfer learning.

Compressive Tracking algorithm (CT) [1] is a novel online tracking algorithm whose focus is object representation. It extracts features from compressed domain and uses a sparse matrix to compress these features. Then a naive Bayes classifier is used to recognize object and updated frame by frame. Focusing on update of classifier, we propose a new tracking algorithm called Transfer Learning based Compressive Tracking (TLCT). To further alleviate problems of drift and inadequacy of training data, we introduce transfer learning to CT. Classifier encounters drift when noise and object appearance change occur. Transferring some prior information is benefit to revise of classifier. Online tracking algorithms assume that nothing is known before tracking,
while offline tracking algorithms assume that a large amount of information about object is known before tracking. Actually, we know a little rather than nothing or anything about object in many situations. For example, we track only people or vehicle in visual surveillance, but we do not know all views of these people or vehicle. Taken a little prior information into consideration, we may at tune the tracker to the particular scenario. In this paper, we assume that a little information of tracked object is known in advance, so we collect some visual prior as source domain data of transfer learning to ensure that sources of data are not too dissimilar. At first frame, a classifier is initialized by CT. We save it as initial classifier and then use transfer learning to update the classifier. The classifier is updated when a new tracking result is obtained in each frame. To alleviate drift, we use initial classifier to get the similarity between tracking result and object in first frame. If the similarity is high, we use transfer learning to update classifier again. This step makes tracker more robust.

The rest of the paper is organized as follows. In Section 2, we introduce TLCT. Experimental results are demonstrated in Section 3. And conclusion is made in Section 4.

II. COMPRESSION TRACKING

CT formulates tracking as classification. It assumes only the location and size of object in first frame is known. The classifier is constructed at the first frame and updated at others. At each frame, the samples used to construct and update classifier are sampled from current frame. Positive samples are sampled near current object location, while negative samples are sampled far away from current object location. Due to the assumption that object location is not changed notably, the object location at the next frame is searched around the object location at current frame. The location with the maximal confidence is considered as the tracking result.

A. Object Representation

Haar-like features [20] are popular object representation method. There often exist a large number of Haar-like features. To reduce computational load, efficient dimensional reduction is done with a very sparse matrix. We just describe the procedures of CT without theoretical analysis due to the limit of the article length. At the first frame, rectangles to calculate Haar-like features are generated randomly and never changed. Each sample is represented as a vector \( \mathbf{v} = \{v_1, v_2, ..., v_m\}^T \in \mathbb{R}^m \). Each element of vector is a Haar-like feature which is calculated via certain rectangles and weights. The dimension of vector could be a small number which could be even smaller than one hundred. The compressive sensing theories [21][22] ensure that the vector contains almost all information of original Haar-like features.

B. Classifier construction and update

After dimension reduction, each sample is represented as a vector \( \mathbf{v} = \{v_1, v_2, ..., v_m\}^T \in \mathbb{R}^m \). A naive Bayes classifier is used to categorize samples as object or background. Define \( y = 0, 1 \) as label variable to represent object and background respectively and assume \( p(y = 0) = p(y = 1) \). Then, classifier is defined as

\[
H(\mathbf{v}) = \log\left(\frac{\sum_{i=1}^{m} p(v_i|y = 1)p(y = 1)}{\sum_{i=1}^{m} p(v_i|y = 0)p(y = 0)}\right)
= \sum_{i=1}^{m} \log\left(\frac{p(v_i|y = 1)}{p(v_i|y = 0)}\right)
\]

(1)

The sample is classified as object when \( H(v) \) is larger than a threshold. Diaconis and Freedman [23] have shown theoretically that the most projections are Gaussian or approximate Gaussian under some conditions. Thus \( p(v_i|y = 0) \) and \( p(v_i|y = 1) \) are assumed to be Gaussian distribution with the parameters \( \langle \mu_i^0, \sigma_i^0, \mu_i^1, \sigma_i^1 \rangle \), where

\[
p(v_i|y = 0) \sim N(\mu_i^0, \sigma_i^0),
p(v_i|y = 1) \sim N(\mu_i^1, \sigma_i^1).
\]

(2)

The parameters \( \langle \mu_i^0, \sigma_i^0, \mu_i^1, \sigma_i^1 \rangle \) are updated incrementally by

\[
\begin{align*}
\mu_i^j & \leftarrow \lambda \mu_i^j + (1 - \lambda) \mu_i^j \\
\sigma_i^j & \leftarrow \sqrt{\lambda (\sigma_i^j)^2 + (1 - \lambda)(\sigma_i^j)^2 + \lambda(1 - \lambda)(\mu_i^j - \mu)^2}
\end{align*}
\]

(3)

where \( j = 0 \) or \( 1 \), \( \lambda > 0 \) is a learning rate,

\[
\sigma_i^j = \frac{1}{q_j} \sum_{k=0}^{q_j-1} \sqrt{(v_i(k) - \mu_j)^2},
\]

(4)

\[
\mu_i^j = \frac{1}{q_j} \sum_{k=0}^{q_j-1} v_i(k),
\]

and \( q_j \) is the number of samples which are sampled in one frame and categorized to class \( j \).

Algorithm 1 Compressive Tracking [1]

**Input:** t-th video frame

1: Sample a set of image patches, \( D^\gamma = \{z||I(z) - l_{-1}|| < \gamma\} \) where \( l_{-1} \) is the tracking location at the (t-1)-th frame, and extract the features with low dimensionality

2: Use classifier H to each feature vector \( \mathbf{v}(z) \) and find the tracking location \( l_t \) with the maximal classifier response

3: Sample two sets of image patches \( D^\alpha = \{z||I(z) - l_t|| < \alpha\} \) and \( D^\beta = \{z|z < ||I(z) - l_t|| < \beta\} \) with \( \alpha < \zeta < \beta \)

4: Extract the features with these two sets of samples and update the classifier parameters according to (3)

5: **Output:** Tracking location \( l_t \) and classifier parameters
III. TRANSFER LEARNING BASED COMPRESSIVE TRACKING

CT focuses on object representation. Classifier used to discriminate object from background is naive Bayes classifier with a small modification on updating. We adopt the same framework of CT and introduce transfer learning to CT for further improvement of classifier performance.

A. Overview

Algorithm flowchart of classifier updating is shown in Fig. 1. In updating step, object location has already been estimated by the classifier updated in previous frame. Then update classifier by (3). If the object location estimated get a high response by initial classifier, transfer learning is done to update classifier again. As the classifier has been updated many times, noise has affected the precision of classifier. We transfer some prior information to revise the classifier when the object appearance at current frame is very similar to that at first frame, because images collected in advance are just similar to object in first frame.

We assume that a little information about object is known before tracking. Thus, we can collect some positive images which are similar to object but not object, and some negative images which are very dissimilar to object from the real world. As a result of object change in the process of tracking, being similar to object means that positive images are similar to object at first frame. It is often a reasonable assumption. In practice, we often track persons, cars, or toys from the frame in which they are frontal to camera.

After rectangles and weights used to calculate Haar-like features are generated randomly, positive images and negative images are represented as vectors whose generation is the same as that in CT. These vectors consist of source domain for transfer learning. The source domain is divided into two parts, negative source data and positive source data according to whether the image is dissimilar to object or not. Note that rectangles must be scaled according to the ratio of the size of images collected in advance to the size of object tracked. Then we introduce transfer learning to CT. Thanks to the conduction of initial classifier, classifier is updated via transfer learning only when the positive source data is similar to object recognized by the classifier. This process alleviates drift and supplements the data for training.

B. Proposed Transfer learning algorithm

We propose a transfer learning algorithm called self-training-like algorithm (STL) whose steps are summarized in Algorithm 2. Just as its name suggests, it is similar to self-training model.

The motivation of self-training-like algorithm is that f was trained on inadequate or incorrect data, and therefore we collect some similar and dissimilar data to supplement and revise dataset. However, the assumption of self-training-like algorithm is that its own predictions are highly confidence ones with a little bias. Thus it is still necessary to use f to predict labels for source data.

Target domain is not emerged explicitly in STL, but classifier trained on target domain contains its information. STL is categorized as inductive transfer learning, based on different situations between the source and target domains and tasks, and instance-based transfer learning, based on what to transfer[15]. The specific formula of step 4 in TLCT is described below.

Algorithm 2 Self-training-like

Input: positive source data \( \{(x_i, 1)\}_{i=1}^n \), negative source data \( \{(x_i, 0)\}_{i=1}^n \), classifier f trained on target data

1: Initially, let \( T = \phi \)
2: Apply f to all instances in P and N
3: Add instances classified correctly to T
4: Use T to update f by one means or another
5: Output: updated classifier f

C. Update via transfer learning

At first frame, initial classifier \( f_{\text{init}} \) (To distinguish TLCT from CT, we use f to represent classifier in TLCT) is constructed and saved just the same as CT. The classifier is the most confidential one because of high confidence of its training samples. A naive Bayes classifier f is used to categorize. The parameters \((\mu^j_0, \sigma^j_0, \mu^1, \sigma^1)\) of f is updated via (3) firstly, and then if the response of initial classifier \( f_{\text{init}}(v) \) is larger than a threshold, f is updated via STL with parameters: positive source data, negative source data and f.

The formula of step 4 in TLCT is

\[
\sigma^j = \frac{1}{s^j} \sum_{k=0}^{s^j-1} \sqrt{(v_i(k) - \mu^j)^2},
\]

and \( s^j \) is the number of samples which are in source domain and categorized to class j.

The equation (3) in CT is reasonable, so we use the same equation to update mean value in (5). The equation to update variance in (5) is different from (3) in CT because the dataset is different. In CT, data for updating is assumed to be sampled from the same distribution as data of previous frame. But in TLCT, \( P(X) \) of source domain is assumed to be different from \( P(X) \) in target domain. \( \sigma^j \) is often small because the data collected in advance is selected deliberately. Thus, equation (5) weights more \( \sigma^j \) and is more effective.

The prediction step of TLCT is the same as CT. And the whole flowchart of TLCT is shown in Algorithm 3. The main difference between TLCT and CT are step 5 and 6.
Simplicity is the main characteristic of TLCT. We propose a simple transfer learning algorithm STL and introduce it to a simple tracking algorithm CT, but simplicity does not affect the robustness and effectiveness of TLCT. TLCT is different from other transfer learning based tracking algorithms. Recently proposed algorithm TLB [16] regards data sampled from previous frames as source domain. If it encounters drift at previous frames, transfer learning would hurt performance instead of improving. The algorithm proposed by [18] collects prior visual information arbitrarily which can be very dissimilar to object, and results in negative transfer easily. It is a more effective strategy that prior visual information is exploited under certain condition which is used in TLCT.

D. Discussion

We compare our algorithm on 17 sequences with 2 methods. TLCT is able to track any type of objects. For convenience, our experiments focus on only head tracking. We use head tracking datasets used by CT to evaluate our algorithm. They are all from [13], including David indoor, Girl, Occluded face and Occluded face 2. For stronger persuasion, we conduct experiments on 14 other sequences from [24], in which seq.mb is the same as Girl from [13]. In order to avoid repetition, we eliminate Girl and then the total number of sequences is 17. The total number of frames is 4131. All images are converted to gray scale. The object is modeled as a rectangle. The ground truth of David indoor, Occluded face and Occluded face 2 is given. The other 14 sequences are manually annotated by us. Sequence seq_villains1 includes some frames in which object is out of

IV. EXPERIMENT

A. Dataset

We compare our algorithm on 17 sequences with 2 methods. TLCT is able to track any type of objects. For convenience, our experiments focus on only head tracking. We use head tracking datasets used by CT to evaluate our algorithm. They are all from [13], including David indoor, Girl, Occluded face and Occluded face 2. For stronger persuasion, we conduct experiments on 14 other sequences from [24], in which seq.mb is the same as Girl from [13]. In order to avoid repetition, we eliminate Girl and then the total number of sequences is 17. The total number of frames is 4131. All images are converted to gray scale. The object is modeled as a rectangle. The ground truth of David indoor, Occluded face and Occluded face 2 is given. The other 14 sequences are manually annotated by us. Sequence seq_villains1 includes some frames in which object is out of
Algorithm 3 Transfer learning based compressive tracking

**Input:** positive source data \( \{(x_i, 1)\}_{i=1}^p \), negative source data \( \{(x_i, 0)\}_{i=1}^n \), classifier \( f_{t-1} \), initial classifier \( f_{\text{init}} \)

1. Sample a set of image patches \( D^\gamma = \{z||I(z) - l_{t-1}|| < \gamma\} \) where \( l_{t-1} \) is the tracking location at the (t-1)-th frame, and extract the features with low dimensionality
2. Use classifier \( H \) to each feature vector \( v(z) \) and find the tracking location \( I_t \) with the maximal classifier response
3. Sample two sets of image patches \( D^\alpha = \{z||I(z) - l_t|| < \alpha\} \) and \( D^\zeta^\beta = \{z|\zeta < ||I(z) - l_t|| < \beta\} \) with \( \alpha < \zeta < \beta \)
4. Extract the features with these two sets of samples and update the classifier parameters according to (3)
5. Extract the feature vector \( v_t \) of sample with location \( I_t \)
6. If \( f_{\text{init}}(v_t) \) is larger than a threshold, use STL with parameters \( P, N \) and \( f_t \) to update \( f_t \) according to (5)

**Output:** Tracking location \( I_t \) and updated classifier \( f_t \)

**B. Evaluation metrics**

Similar to [1], we use 2 quantitative metrics to evaluate algorithms, namely, precision (P) and average center location error (ACLE). Define overlap rate (OR) and center location error (CLE) as

\[
OR = \frac{\text{area}(\text{ROI}_T \cap \text{ROI}_G)}{\text{area}(\text{ROI}_T \cup \text{ROI}_G)},
\]

\[
CLE = \frac{\left\| c_T - c_G \right\|_2}{\text{area}(\text{ROI}_T)},
\]

where \( \text{ROI}_T \) is the result of tracker, \( \text{ROI}_G \) is ground truth, \( c_T \) is the center coordinate of tracking result, \( c_G \) is the center coordinate of ground truth and \( \| \cdot \|_2 \) is Euclidean distance. The tracking result is considered right when OR is larger than 0.25. P and ACLE are defined as

\[
P = \frac{\#_{\text{right}}}{\#_{\text{frames}}},
\]

\[
ACLE = \frac{\sum \text{sum}}{\#_{\text{frames}}}
\]

where \#_{\text{right}} is the number of frames in which the tracking result is right, \#_{\text{frames}} is the total number of frames and sum is the sum of CLE for all frames.

**C. Implementation**

Face recognition is one of the most successful applications of image analysis and understanding [25], so there are many benchmark datasets on face recognition and it is convenience for us to use these datasets as source domain in transfer learning to do head tracking. CBCL face dataset\(^3\) is chosen to do the experiments in this paper.

\(^1\)http://vision.ucsd.edu/~bbabenko/project_flisttrack.shtml
\(^2\)http://www.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html
\(^3\)http://cbcl.mit.edu/software-datasets/FaceData2.html

Our method is compared with both CT [1]\(^4\) and tracking learning detection (TLD) [26]\(^5\). As using random number in TLCT and CT, we run these algorithms 10 times and report the average metrics. Parameters in CT and TLD remain default in source code provided by authors. TLCT parameters included in both TLCT and CT, are equal to CT parameters and the other TLCT parameters are set experientially. The experimental results are a little sensitive to the parameters. But parameter values are suitable for the similar situations, so we can determine the value of parameters conveniently by learning from a typical situation.

**D. Experimental results**

TABLE I and TABLE II show the experimental results. In TABLE I, Red fonts mean the best. Blue fonts mean the second best. TLD tracker often loses the object, so ACLE of TLD in Table II is only calculated for the frames that TLD could track. Thus, only ACLE of TLCT and CT are compared. Red fonts in Table II mean the best among TLCT and CT. The ACLE of TLD is just a reference. Fig. 2 show some images with tracking results. As shown in TABLE I, TLCT achieves the best performance on 12 sequences. TLCT outperforms CT on 10 sequences and achieves 100% on 3 sequences as the same as CT. Performances of TLCT on david\_indoor and faceocc are just a little bit worse than CT. CT and TLCT update classifier at each frame to deal with appearance change, but sometimes drift emerges. TLCT can further alleviate drift via transfer learning. As TLD cannot deal with out-of-plane rotation of object, its performances on first 10 sequences are not very good. Sequences seq\_ms and seq\_sb are different from other sequences. The object is not out-of-plane rotated at most frames. Moreover, it is occluded completely at some frames by an obstacle similar to object. Thus, CT and TLCT encounter drift but TLD redetect the object after losing track for a few frames. TABLE II shows the similar situation to TABLE I. In most cases, ACLE is smaller when P is higher. TLCT still achieves the best performance on 12 sequences.

Our method spends a large amount of time on calculating features of source data at the first frame. This procedure prevent the method from real time application. To overcome the problem, we plan to use parallel computing in the future.

**V. Conclusions**

In this paper, we propose a novel transfer learning algorithm called self-training-like algorithm and apply it to compressive tracking to further alleviate drift and extend training dataset. Experimental results show the effectiveness of our algorithm.

**ACKNOWLEDGEMENTS**

The research was supported by National Natural Science Foundation of China (61105018, 61175020).

\(^4\)http://www4.comp.polyu.edu.hk/~csli/A/CT/CT.htm
\(^5\)http://info.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html
Fig. 2. Tracking results of some frames

REFERENCES

### TABLE I

P(\%) FOR ALL SEQUENCES

<table>
<thead>
<tr>
<th>Sequence</th>
<th>TLCT</th>
<th>CT</th>
<th>TLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>seq_bb</td>
<td>96.1</td>
<td>94.7</td>
<td>41.2</td>
</tr>
<tr>
<td>seq_cubicle</td>
<td>94.1</td>
<td>89.4</td>
<td>70.6</td>
</tr>
<tr>
<td>seq_dlb</td>
<td>100.0</td>
<td>100.0</td>
<td>31.4</td>
</tr>
<tr>
<td>seq_dt</td>
<td>71.2</td>
<td>64.1</td>
<td>49.0</td>
</tr>
<tr>
<td>seq_dk</td>
<td>100.0</td>
<td>100.0</td>
<td>64.7</td>
</tr>
<tr>
<td>seq_dp</td>
<td>80.0</td>
<td>73.2</td>
<td>44.0</td>
</tr>
<tr>
<td>seq_div</td>
<td>94.0</td>
<td>97.7</td>
<td>76.3</td>
</tr>
<tr>
<td>seq_last</td>
<td>92.3</td>
<td>86.9</td>
<td>78.4</td>
</tr>
<tr>
<td>seq_mb</td>
<td>98.4</td>
<td>85.7</td>
<td>78.3</td>
</tr>
<tr>
<td>seq_ms</td>
<td>95.7</td>
<td>87.1</td>
<td>67.7</td>
</tr>
<tr>
<td>seq_set</td>
<td>54.9</td>
<td>62.2</td>
<td>93.7</td>
</tr>
<tr>
<td>seq_scr</td>
<td>75.6</td>
<td>70.4</td>
<td>90.9</td>
</tr>
<tr>
<td>seq_villains1</td>
<td>94.1</td>
<td>83.2</td>
<td>12.5</td>
</tr>
<tr>
<td>seq_villains2</td>
<td>96.0</td>
<td>83.6</td>
<td>87.5</td>
</tr>
<tr>
<td>david_indoor</td>
<td>98.9</td>
<td>99.7</td>
<td>97.0</td>
</tr>
<tr>
<td>faceocc</td>
<td>6.3</td>
<td>7.6</td>
<td>9.4</td>
</tr>
<tr>
<td>faceocc2</td>
<td>29.3</td>
<td>32.1</td>
<td>25.2</td>
</tr>
<tr>
<td>Average</td>
<td>91.7</td>
<td>86.9</td>
<td>80.0</td>
</tr>
</tbody>
</table>

### TABLE II

MACLE (PIXELS) FOR ALL SEQUENCES

<table>
<thead>
<tr>
<th>Sequence</th>
<th>TLCT</th>
<th>CT</th>
<th>TLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>seq_bb</td>
<td>3.9</td>
<td>5.5</td>
<td>38.8</td>
</tr>
<tr>
<td>seq_cubicle</td>
<td>16.7</td>
<td>16.2</td>
<td>18.9</td>
</tr>
<tr>
<td>seq_dlb</td>
<td>9.8</td>
<td>6.9</td>
<td>49.9</td>
</tr>
<tr>
<td>seq_dt</td>
<td>29.3</td>
<td>32.1</td>
<td>25.2</td>
</tr>
<tr>
<td>seq_dp</td>
<td>6.3</td>
<td>7.6</td>
<td>9.4</td>
</tr>
<tr>
<td>seq_div</td>
<td>21.6</td>
<td>21.6</td>
<td>49.8</td>
</tr>
<tr>
<td>seq_last</td>
<td>8.5</td>
<td>7.1</td>
<td>10.6</td>
</tr>
<tr>
<td>seq_ms</td>
<td>9.0</td>
<td>18.7</td>
<td>7.0</td>
</tr>
<tr>
<td>seq_scr</td>
<td>4.6</td>
<td>14.2</td>
<td>13.1</td>
</tr>
<tr>
<td>seq_villains1</td>
<td>5.5</td>
<td>9.6</td>
<td>28.5</td>
</tr>
<tr>
<td>seq_villains2</td>
<td>24.5</td>
<td>21.2</td>
<td>11.1</td>
</tr>
<tr>
<td>david_indoor</td>
<td>12.3</td>
<td>20.4</td>
<td>14.1</td>
</tr>
<tr>
<td>faceocc</td>
<td>98.9</td>
<td>99.7</td>
<td>97.2</td>
</tr>
<tr>
<td>faceocc2</td>
<td>100.0</td>
<td>100.0</td>
<td>82.2</td>
</tr>
<tr>
<td>Average</td>
<td>10.5</td>
<td>14.8</td>
<td>17.0</td>
</tr>
</tbody>
</table>


