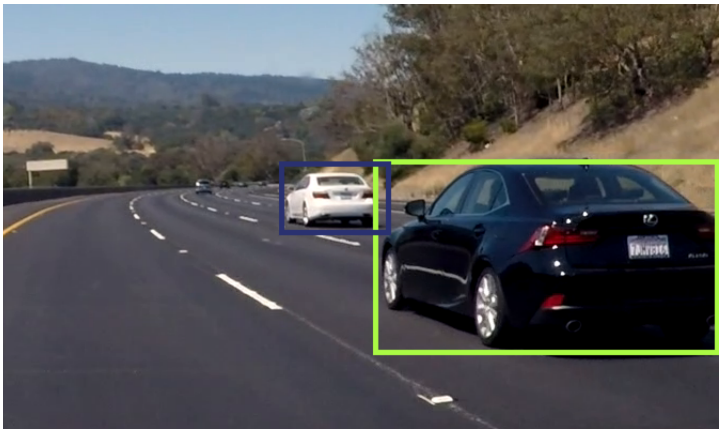
Сопровождение объектов в видео

Никита Лисин, Владимир Лютов, Влад Шахуро

Graphics & Media Lab
Vision Group

В этом задании вы научитесь отслеживать объекты с помощью нейросетевого детектора и простого метода ассоциации обнаружений. Максимальная оценка за задание — 10 баллов. Для удобства выполнения задание разделено на пункты.



1. Получение обнаружений (1 балл)

На первом этапе мы будем использовать Single Shot Detector (SSD) для получения обнаружений на каждом кадре видео независимо. Обнаружение состоит из 5 значений — координат, задающих положение ограничивающего прямоугольника, и степени уверенности детектора. В `detection.py` реализуйте функции:

- `extract_detections(frame, min_confidence)` — для заданного кадра возвращает обнаружения со степенью уверенности больше `min_confidence`;
- *Опционально*: `draw_detections(detections)` — возвращает изображение с визуализацией обнаружений.

Проверьте реализацию с помощью юнит-теста:
`$ ./run.py unittest detection`

2. Метрика IoU (1 балл)

Для сопоставления обнаружений между кадрами мы будем использовать метрику IoU. В `metrics.py` реализуйте функцию `iou_score(bbox1, bbox2)`.

Проверьте реализацию с помощью юнит-теста:
`$ ./run.py unittest iou`

3. Сопоставление обнаружений (2 балла)

Сопоставление производится в несколько этапов:

1. Получение обнаружений для первого кадра и присвоение каждому обнаружению уникального id;
2. Для каждого последующего кадра:
   (a) Получение обнаружений с предыдущих кадров — для каждого id по одному обнаружению;
   (b) Вычисление IoU между всеми обнаружениями с текущего кадра и всеми обнаружениями с предыдущего этапа;
   (c) Каждому обнаружению с текущего кадра присваивается id того обнаружения, IoU с которой максимально и больше порога;
   (d) Несопоставленным обнаружениям присваивается новый уникальный id.

В `tracking.py` реализуйте следующие методы класса `Tracker`:

- `init_tracklet(self, frame)` — получает обнаружения для первого кадра, присваивает каждой уникальный id и возвращает результат;
- `prev_detections(self)` — возвращает список обнаружений с последних `lookup_tail_size` кадров из `detection_history`, при этом для каждого id возвращается только одно обнаружение, принадлежащее кадру с наибольшим номером;
- `bind_tracklet(self, detections)` — сопоставляет по метрике IoU обнаружения с текущего кадра и обнаружения, полученные с помощью `prev_detections`. Присваивает каждому обнаружению с текущего кадра соотвествующий id и возвращает результат.

Проверьте реализацию с помощью юнит-теста:
```
$ ./run.py unittest tracker
```

4. Метрика MOTP (2 балла)

Для оценки качества сопровождения в базовой части используется метрика MOTP — среднее значение IoU по всем успешным сопоставлениям истинных и предсказанных обнаружений. Этапы вычисления MOTP (для каждого кадра):

1. Для каждого успешного сопоставления с предыдущего кадра: если на текущем кадре присутствуют истинная и предсказанное обнаружение с теми же id, и IoU больше порога, то сопоставление между этими обнаружениями считается успешным;
2. Оставшиеся истинные и предсказанные обнаружения сопоставляются в порядке убывания IoU.

В `metrics.py` реализуйте функцию `motp(obj, hyp, threshold)`. Проверьте реализацию с помощью тестов:
```
$ ./run.py unittest motp
$ ./run.py unittest baseline
```

5. Нормализированная кросс-корреляция

Так как для применения детектора к каждому кадру видео требуется много времени, мы будем применять детектор к каждому N кадру, а на промежуточных кадрах находить обнаружения с помощью нормализированной кросс-корреляции:

$$current\_bbox = \arg\max_{bbox} cross\_correlation(bbox, prev\_bbox) * P_{N(center_{prev\_bbox}, shape_{prev\_bbox})}(center_{bbox}),$$

$P_{N(mean,std)}$ – нормальное распределение.

Для кросс-корреляции используется template matching.

В `tracking.py` реализуйте следующий метод класса `CorrelationTracker`: `bind_tracklet(self, frame)` — возвращает результат применения нормализированной кросс-корреляции к текущему кадру.

6. Метрики MOTP и MOTA (4 балла)

Одними из основных метрик для задачи трекинга являются MOTP и MOTA. Прочитайте часть статьи и реализуйте соотвествующие метрики. В `metrics.py` реализуйте функцию `motp_mota(obj, hyp, threshold)`.

Проверьте реализацию с помощью тестов:

```
$ ./run.py unittest motp_mota
$ ./run.py unittest tracking
```

Multiple Object Tracking Performance Metrics and Evaluation in a Smart Room Environment

Keni Bernardin, Alexander Elbs, Rainer Stiefelhagen
Institut für Theoretische Informatik
Interactive Systems Lab
Universität Karlsruhe, 76131 Karlsruhe, Germany
keni@ira.uka.de, alex@segv.de, stiefel@ira.uka.de

Abstract

Simultaneous tracking of multiple persons in real world environments is an active research field and several approaches have been proposed, based on a variety of features and algorithms. Recently, there has been a growing interest in organizing systematic evaluations to compare the various techniques. Unfortunately, the lack of common metrics for measuring the performance of multiple object trackers still makes it hard to compare their results.
In this work, we introduce two intuitive and general metrics to allow for objective comparison of tracker characteristics, focusing on their precision in estimating object locations, their accuracy in recognizing object configurations and their ability to consistently label objects over time.
We also present a novel system for tracking multiple users in a smart room environment using several cameras, based on color histogram tracking of person regions and automatic initialization using special object detectors.
This system is used to demonstrate the expressiveness of the proposed metrics through a sample performance evaluation using real test video sequences of people interacting in the smart room.

1 Introduction and Related Work

The tracking of multiple persons in camera images is a very active research field with applications in many domains. These range from video surveillance, over automatic indexing, to intelligent interactive environments. Especially in the last case, a robust person tracking module can serve as a poweful building block to support other techniques, such as gesture recognizers, face identifiers, head pose estimators [10], scene analysis tools, etc. In the last few years, more and more approaches have been presented to tackle the problems posed by unconstrained, natural environments and bring person trackers out of the laboratory environment and into real world scenarios.

In recent times, there has also been a growing interest in performing systematic evaluations of such tracking tools with common databases and metrics. Examples are the CHIL project, funded by the EU [17], the VACE project in the U.S. [18], but also a growing number of workshops (PETS [19], EEMCV [20], etc). However, there is still no general agreement on a principled evaluation procedure using a common set of objective and intuitive metrics for measuring the performance of multiple object trackers. Because of this lack of metrics, some researchers present their tracking systems without any quantitative evaluation of their performance (e.g. [1, 7, 15]). On the other hand, a multitude of isolated measures were defined in individual contributions to validate trackers using various features and algorithms (see e.g. [2, 3, 5, 11, 13]), but no common agreement on a best set of measures exists.

To remedy this, this paper proposes a thorough procedure to detect all types of errors produced by a multiple object tracking system and introduces two novel metrics, the Multiple Object Tracking Precision ($MOTP$), and the Multiple Object Tracking Accuracy($MOTA$), that intuitively express a tracker's characteristics and could be used in general performance evaluations.

Perhaps the work that most closely relates to ours is that of Smith et al. in [4]. The authors also attempt to define an objective procedure and measures for multiple object tracker performance. However, key differences to our contribution exist: In [4], the autors introduce a large number of metrics: 5 for measuring object configuration errors, and 4 for measuring inconsistencies in object labeling over time. Some of the measures are defined in a dual way for trackers and for objects (e.g. MT/MO, FIT/FIO, TP/OP). This could make it difficult to gain a clear and direct understanding of the tracker's overall performance. Moreover, under certain conditions, some of these measures can behave in a non-intuitive fashion (such as the CD, as the authors state, or the $\overline{FP}$ and $\overline{FN}$, as we will demonstrate later). In comparison, we introduce just 2 overall performance measures that allow a clear and intuitive insight into the main tracker characteristics: its precision in estimating object positions, its ability to determine the number of objects and their configuration, and its skill at keeping consistent tracks over time. In addition, we offer an experimental validation of the presented theoretical framework by performing sample evaluation runs on 2 variants of a multiple person tracker, using real data recorded in a smart room environment. A demonstration run on simulated data is also performed to better illustrate the expressiveness of the proposed metrics.

The system used in our evaluations is a 3D multiple per-

son tracker developed for use in our smart room. It initializes automatically using special person detectors, performs color histogram tracking of body parts on several camera views and intelligently fuses the 2D information to produce a consistent set of 3D hypotheses. It is used as a proof of concept for the introduced $MOTP$ and $MOTA$ metrics, which are used to measure its accuracy on datasets of varying degrees of difficulty.

The remainder of the paper is organized as follows: Section 2 presents the new metrics, the $MOTP$ and the $MOTA$ and a detailed procedure for their computation. Section 3 briefly introduces the developed multiperson tracker which will be used in the evaluations. In Section 4, the sample performance measurements are shown and the usefulness of the metrics is discussed. Finally, Section 5 gives a summary and a conclusion.

2 Performance Metrics for Multiple Object Tracking

To help better understand the proposed evaluation metrics, we first explain what qualities we expect from an ideal multiple object tracker. It should at all points in time find the correct number of objects present and estimate the position of each object as precisely as possible (Note that properties such as the size, contour, orientation or speed of objects are not considered here). It should also keep consistent track of each object over time: Each object should be assigned a unique track ID which stays constant throughout the sequence (even after temporary occlusion, etc). This leads to the following design criteria for performance evaluation metrics:

- They should allow to judge the tracker's precision in determining exact object locations.
- They should reflect its ability to consistently track object configurations through time, i.e. to correctly trace object trajectories, producing exactly one trajectory per object.

Additionally, we expect useful metrics

- to have as few free parameters, adjustable thresholds, etc, as possible to help make evaluations straightforward and keep results comparable.
- to be clear, easily understandable and behave according to human intuition, especially in the occurence of multiple errors of different types or of uneven repartition of errors throughout the sequence.
- to be general enough to allow comparison of most types of trackers (2D, 3D trackers, object centroid trackers or object area trackers, etc).
- to be few in number and yet expressive, so they may be used e.g. in large evaluations where many systems are being compared.

Based on the above criteria, we propose a procedure for systematic and objective evaluation of a tracker's characteristics. Assuming that for every time frame t a multiple object tracker outputs a set of hypotheses $\{h_1 \dots h_m\}$ for a set of visible objects $\{o_1 \dots o_n\}$, the evaluation procedure comprises the following steps:
For each time frame t,

- Establish the best possible correspondence between hypotheses h_j and objects o_i
- For each found correspondence, compute the error in the object's position estimation.
- Accumulate all correspondence errors:
  - Count all objects for which no hypothesis was output as misses.
  - Count all tracker hypotheses for which no real object exists as false positives
  - Count all occurences where the tracking hypothesis for an object changed compared to previous frames as mismatch errors. This could happen, e.g., when two or more objects are swapped when they pass close to each other, or when an object track is reinitialized with a different ID, after it was previously lost because of occlusion.

Then, the tracking performance can be intuitively expressed in two numbers: The "tracking precision" which expresses how well exact positions of persons are estimated, and the "tracking accuracy" which shows how many mistakes the tracker made in terms of misses, false positives, mismatches, failures to recover tracks, etc. These measures will be explained in detail in the latter part of this section.

2.1 Establishing Correspondences Between Objects and Tracker Hypotheses

As explained above, the first step in evaluating the performance of a multiple object tracker is finding a continuous mapping between the sequence of object hypotheses $\{h_1 \dots h_m\}$ output by the tracker in each frame and the real objects $\{o_1 \dots o_n\}$. This is illustrated in Fig. 1. Naively, one would match the closest object-hypothesis pairs and treat all remaining objects as misses and all remaining hypotheses as false positives. A few important points need to be considered, though, which make the procedure less straightforward.

2.1.1 Valid Correspondences

First of all, the correspondence between an object o_i and a hypothesis h_j should not be made if their distance $dist_{i,j}$ exceeds a certain threshold T. There is a certain conceptual boundary beyond which we can no longer speak of an error in position estimation, but should rather argue that the tracker has missed the object and is tracking something else. This is illustrated in Fig. 2(a). For object area trackers (i.e.

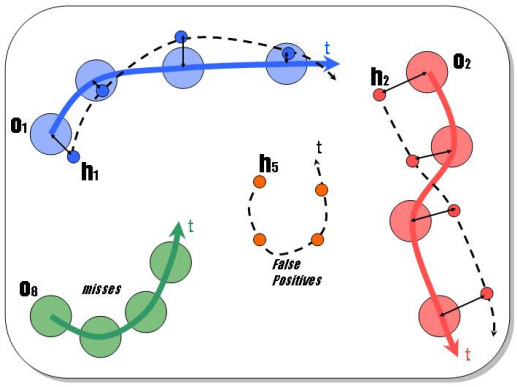


Figure 1: Mapping tracker hypotheses to objects. In the easiest case, matching the closest object-hypothesis pairs for each time frame t is sufficient

trackers that also estimate the size of objects or the area occupied by them), distance could be expressed in terms of the overlap between object and hypothesis, e.g. as in [2], and the threshold T could be set to zero overlap. For object centroid trackers, one could simply use the Euclidian distance, in 2D image coordinates or in real 3D world coordinates, between object centers and hypotheses, and the threshold could be, e.g., the average width of a person in pixels or cm. In the following, we refer to correspondences as *valid* if $dist_{i,j} < T$.

2.1.2 Consistent Tracking over Time

Second, to measure the tracker's ability to label objects consistently, one has to detect when conflicting correspondences have been made for an object over time. Fig. 2(b) illustrates the problem. Here, one track was mistakenly assigned to 3 different objects over the course of time. A mismatch can occur when objects come close to each other and the tracker wrongfully swaps their identities. It can also occur when a track was lost and reinitialized with a different identity. One way to measure such errors could be to decide on a "best" mapping (o_i, h_j) for every object o_i and hypothesis h_j, e.g. based on the initial correspondence made for o_i, or the most frequently made correspondence (o_i, h_j) in the whole sequence. One would then count all correspondences where this mapping is violated as errors. In some cases, this kind of measure can however become non-intuitive. As shown in Fig. 2(c), if, for example, the identity of object o_i is swapped just once in the middle of the tracking sequence, the time frame at which the swap occurs drastically influences the value output by the error measure.
This is why we follow a different approach: only count mismatch errors once at the time frame where a change in object-hypothesis mappings is made and consider the correspondences in intermediate segments as correct. Especially in cases where many objects are being tracked and mismatches are frequent, this gives us a more intuitive and expressive error measure.

To detect when a mismatch error occurs, a list of object-

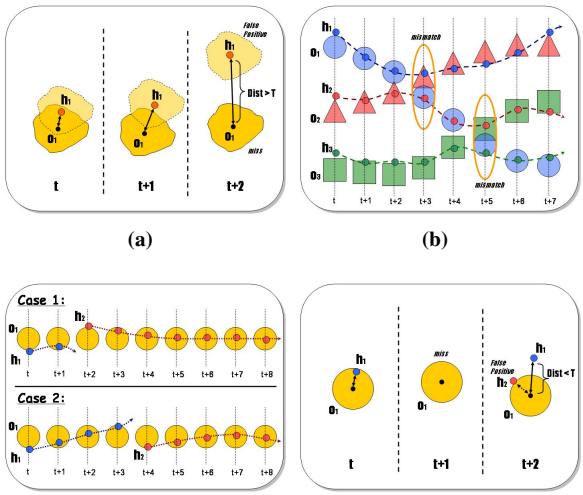


Figure 2: Optimal correspondences and error measures. Fig. 2(a): When the distance between o_1 and h_1 exceeds a certain threshold T, one can no longer make a correspondence. Instead, o_1 is considered missed and h_1 becomes a false positive. Fig. 2(b): Mismatched tracks. Here, h_2 is first mapped to o_2. After a few frames, though, o_1 and o_2 cross paths and h_2 follows the wrong object. Later, it wrongfully swaps again to o_3. Fig. 2(c): Problems when using a sequence-level "best" object-hypothesis mapping based on most frequently made correspondences. In the first case, o_1 is tracked just 2 frames by h_1, before the track is taken over by h_2. In the second case, h_1 tracks o_1 for almost half the sequence. In both cases, a "best" mapping would pair h_2 and o_1. This however leads to counting 2 mismatch errors for *case 1* and 4 errors for *case 2*, although in both cases only one error of the same kind was made. Fig. 2(d): Correct reinitialization of a track. At time t, o_1 is tracked by h_1. At $t+1$, the track is lost. At $t+2$, two valid hypotheses exist. The correspondence is made with h_1 although h_2 is closer to o_1, based on the knowledge of previous mappings up to time $t+1$

hypothesis mappings is constructed. Let $M_t = \{(o_i, h_j)\}$ be the set of mappings made up to time t and let $M_0 = \{\}$. Then, if a new correspondence is made at time $t+1$ between o_i and h_k which contradicts a mapping (o_i, h_j) in M_t, a mismatch error is counted and (o_i, h_j) is replaced by (o_i, h_k) in M_{t+1}.

The so constructed mapping list M_t can now help to establish optimal correspondences between objects and hypotheses at time $t+1$, when multiple valid choices exist. Fig. 2(d) shows such a case. When it is not clear, which hypothesis to match to an object o_i, priority is given to h_o with $(o_i, h_o) \in M_t$, as this is most likely the correct track. Other hypotheses are considered false positives, and could have occured because the tracker output several hypotheses for o_i, or because a hypothesis that previously tracked another object accidentally crossed over to o_i.

2.1.3 Mapping Procedure

Having clarified all the design choices behind our strategy for constructing object-hypothesis correspondences, we summarize the procedure:

Let $M_0 = \{\}$. For every time frame t,

1. For every mapping (o_i, h_j) in M_{t-1}, verify if it is still valid. If object o_i is still visible and tracker hypothesis h_j still exists at time t, and if their distance does not exceed the threshold T, make the correspondence between o_i and h_j for frame t.

2. For all objects for which no correspondence was made yet, try to find a matching hypothesis. Allow only one to one matches. Start by matching the pair with the minimal distance and then go on until the threshold T is exceeded or there are no more pairs to match. If a correspondence (o_i, h_k) is made that contradicts a mapping (o_i, h_j) in M_{t-1}, replace (o_i, h_j) with (o_i, h_k) in M_t. Count this as a mismatch error and let mme_t be the number of mismatch errors for frame t.

3. After the first two steps, a set of matching pairs for the current time frame is known. Let c_t be the number of matches found for time t. For each of theses matches, calculate the distance d_t^i between the object o_i and its corresponding hypothesis.

4. All remaining hypotheses are considered false positives. Similarly, all remaining objects are considered misses. Let fp_t and m_t be the number of false positives and misses respectively for frame t. Let also g_t be the number of objects present at time t.

5. Repeat the procedure from step 1 for the next time frame. Note that since for the initial frame, the set of mappings M_0 is empty, all correspondences made are initial and no mismatch errors occur.

In this way, a continuous mapping between objects and tracker hypotheses is defined and all tracking errors are accounted for.

2.2 Performance Metrics

Based on the matching strategy described above, two very intuitive metrics can be defined.

1. The *M*ultiple *O*bject *T*racking *P*recision ($MOTP$).

$$MOTP = \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t}$$

It is the total position error for matched object-hypothesis pairs over all frames, averaged by the total number of matches made. It shows the ability of the tracker to estimate precise object positions, independent of its skill at recognizing object configurations, keeping consistent trajectories, etc.

2. The *M*ultiple *O*bject *T*racking *A*ccuracy ($MOTA$).

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}$$

where m_t, fp_t and mme_t are the number of misses, of false positives and of mismatches respectively for time t. The $MOTA$ can be seen as composed of 3 error ratios:

$$\overline{m} = \frac{\sum_t m_t}{\sum_t g_t},$$

the ratio of misses in the sequence, computed over the total number of objects present in all frames,

$$\overline{fp} = \frac{\sum_t fp_t}{\sum_t g_t},$$

the ratio of false positives, and

$$\overline{mme} = \frac{\sum_t mme_t}{\sum_t g_t},$$

the ratio of mismatches.

Summing up over the different error ratios gives us the total error rate E_{tot}, and $1 - E_{tot}$ is the resulting tracking accuracy. The $MOTA$ accounts for all object configuration errors made by the tracker, false positives, misses, mismatches, over all frames. It is similar to metrics widely used in other domains (such as the Word Error Rate (WER), commonly used in speech recognition) and gives a very intuitive measure of the tracker's performance at keeping accurate trajectories, independent of its precision in estimating object positions.

Remark on Computing Averages: Note that for both $MOTP$ and $MOTA$, it is important to first sum up all errors across frames before a final average or ratio can be computed. The reason is that computing ratios r_t for each frame t independently before calculating a global average $\frac{1}{n}\sum_t r_t$ for all n frames (such as, e.g., for the $\overline{FP}$ and $\overline{FN}$ measures in [4]), can lead to non-intuitive metric behavior. This is illustrated in Fig. 3. Although the tracker consistently missed most objects in the sequence, computing ratios independently per frame and then averaging would still yield only 50% miss rate. Summing up all misses first and computing a single global ratio, on the other hand, produces a more intuitive result of 80% miss rate.

3 A 3D Multiperson Tracker using Color and Object Detectors

In order to experimentally validate the new metrics introduced here, we performed an example evaluation of an indoor multiperson tracking system developed for our smartroom [16]. This system will now be briefly presented.