توضیح کلاس Tube:

این کلاس جهت ذخیره سازی اطلاعات مربوط به تیوب ها مورد استفاده قرار میگیرد.

* self.id = tube\_id

شناسه مربوط به تیوب را مشخص میکند.

* self.f\_start = f\_start

شماره فریم ابتدای تیوب

* self.f\_end = f\_start

شماره فریم انتهای تیوب. در ابتدا برابر با شماره ابتدا قرار داده می شود و در طی زمان به روز رسانی می گردد.

* self.f\_start\_shift = 0

شماره فریم ابتدای تیوب پس از اعمال تغییرات جهت خلاصه سازی

* self.f\_end\_shift = self.f\_end - self.f\_start

شماره فریم انتهایی تیوب پس از اعمال تغییرات جهت خلاصه سازی

* self.boxes = []

مختصات باکس ها مربوط به یک آبجکت در طول فریمها در این فیلد نگهداری میشود.

این اعداد به صورت (x,y,w,h) میباشد.

* self.cropped = []

در این فیلد تصاویر بریده شده مربوط به آبجکت ها نگهداری می شود

* self.motion = []
* self.speed = 0

سرعت هر آبجکت با محاسبه فاصله اقلیدسی بین باکس اول و آخر محاسبه می گردد

و در این فیلد ذخیره میشود

* self.history\_speed = []

بعد از انجام فرایند چسباندن تیوب ها به همدیگر سرعتهای تیوب ها با هم ادغام میشوند

و در این لیست نگهداری می شوند

* self.alpha\_decrease = []

در بعضی مواقع بعد از چیدن تیوبها بازهم تعدادی از آنها با هم تداخل دارند. برای نمایش بهتر سعی شده

تا آبجکت بزرگتر به صورت transparent نمایش داده شود. برای این منظور فریم هایی که باید آبجکت

transparent شود در این لیست نگهداری میشود.

**(tubes\_concatenation(tube\_list, overlap\_ignoring, transparency\_overlap\_ignoring**

این تابع برای چیدن تیوب ها برای انجام خلاصه سازی بعد از به پایان رسیدن یک بافر مورد استفاده قرار میگیرد. آرگومان اول شامل لیست تیوب های ایجاد شده و ادغام شده در یک بافر می باشد. آرگومان دوم میزان نادیده گرفتن همپوشانی دو باکس را در نظر مشخص میکند. برای مثال اگر این مقدار 0.5 باشد. در زمانی که دو باکس با هم همپوشانی دارند، اگر باکس بزرگتر کمتر از 50 درصد از باکس کوچکتر را پوشانده باشد، الگوریتم تداخل بین آنها را نادیده میگیرد. آرگومان سوم نیز برای transparent کردن مورد استفاده قرار میگیرد. اگر میزان تداخل دو باکس کمتر از این مقدار باشد، آبجکت بزرگتر transparent نخواهد شد.

**are\_locations\_overlapped(box1, box2)**

این تابع دو باکس را به عنوان ورودی میگیرد و شرایط تداخل آنها را بررسی می نماید.

**check\_conflict(tube1, tube2)**

این تابع برای بررسی تداخل دو تیوب با هم به کار می رود. برای این منظور تداخل باکسهای اول دو تیوب با همدیگر بررسی می شوند.

**find\_intersection(tube1, tube2, f\_start\_new\_tube1, f\_end\_new\_tube1)**

در حالتی که دو تیوب با هم تداخل داشته باشند برای پیدا کردن محل دقیق تداخل از این تابع استفاده خواهد شد. این تابع از روشی شبیه جستجوی دودویی استفاده میکند. فرض کنید که در ویدیو اصلی tube 1 زودتر وارد فیلم شده است و حالا با tube2 تداخل دارد. باکس وسط tube 1 با باکس اول tube2 تداخل بررسی میشود، اگر تداخل وجود نداشت باکس اول tube2 با باکس وسط نیمه اول(نیمه سمت راست) tube1 مقایسه میشود. در غیر این صورت با باکس وسط نیمه دوم(نیمه سمت چپ) tube1 مقایسه میگردد. به همین ترتیب در این تابع بازگشتی آنقدر tube1 نصف میشود تا به شرط پایانی تابع یعنی یک شدن سایز tube1 برسیم. بدین ترتیب محل دقیق تداخل به دست آید.

**is\_overlap\_bigger\_than\_threshold(box1, box2)**

دو باکس را از لحاظ تداخل با هم بررسی می کند. اگر میزان پوشانیده شدن باکس کوچکتر از میزان پارامتر Overlap\_ignoring (که در setting.json میباشد) از تداخل آنها چشم پوشی می شود.

is\_checking\_conflicts\_required(tube1, tube2, index\_tube\_1, index\_tube\_2)

اگر یک تیوب در مکانی ثابت باشد و تیوب دیگر در حال حرکت باشد ، تداخل بین آنها در نظر گرفته نمیشود. در اصل تنها زمانی که دو تیوب در حال حرکت باشند یا هر دو ساکن باشند شرایط تداخلی آنها بررسی میگردد.

**find\_conflicts\_in\_interval(tube\_list)**

زمانی که ترتیب قرار گیری تیوب ها مشخص شد، تعدادی از تیوب ها به دلیل استفاده از پارامتر Overlap\_ignoring در برخی باکس ها با هم تداخل خواهند داشت. این تابع باکس هایی که با هم تداخل دارند را شناسایی می نماید و باکس های بزرگتر را در لیست مربوط به آلفاهای تیوب قرار میدهد. در هنگام ساختن ویدیوی نهایی، زمانی که نوبت ایجاد این باکس ها در فریم های نهایی برسد، این باکس ها به صورت trnasparent نمایان خواهند شد.