Листинг 1 – Процесс обнаружения

def run(self) -> None:  
 self.\_\_init()  
 self.\_health\_checker.beat(  
 type=HealthDataType.DATA,  
 data='START'  
 )  
 try:while self.\_continue\_work():  
 if self.\_\_image\_queue.empty():  
 time.sleep(self.\_\_sleep\_time\_attempting\_get\_task)  
 continuecap\_image: CapturedImage = self.\_\_image\_queue.get()  
 image = cap\_image.get\_image()  
 predictions = self.\_\_net.image\_detection(image)  
 result = DetectionResult(  
 cap\_image=cap\_image,  
 predictions=predictions  
 )  
 self.\_\_result\_queue.put(result)except Exception as ex:  
 self.\_health\_checker.beat(  
 type=HealthDataType.EXCEPTION,  
 data=traceback.format\_exc()  
 )  
 self.\_health\_checker.beat(  
 type=HealthDataType.STOPPED,  
 data='FINISH'  
 )  
 self.\_health\_checker.stop()  
 self.\_health\_checker.join()

Листинг 2 – Процесс обработки обнаруженных объектов

def \_\_handle\_predictions(self):   
 predictions: list[DetectionResult] = self.\_\_get\_predictions()  
 self.\_\_handle\_results(predictions)  
 self.\_\_save\_results()

def \_\_handle\_results(self, predictions: list[DetectionResult]):  
 for res in predictions:   
 video, video\_dt = self.\_\_videos\_writer.set\_frame(res.cap\_image)  
 self.\_\_eventor.set\_data(res.cap\_image.meta.source,  
 res.cap\_image.meta.dt,  
 video,  
 video\_dt,  
 res.predictions)

Листинг 3 – Определение погрузки

def set\_data(  
 self,  
 source: str,  
 src\_dt: datetime.datetime,  
 video\_name: str,  
 video\_time: datetime.timedelta,  
 predictions: list[Prediction]  
):  
 *# Если это новый источник и его данные не были проинициализированы, то ставим значение по умолчанию* if self.\_\_areas\_interest.get(source, None) is None:  
 self.add\_or\_update\_interest\_area(source, None)  
 if not self.\_\_time\_without\_ladle.get(source, None):  
 self.add\_or\_update\_bucket\_loss\_time(source, self.\_TIME\_WITHOUT\_LADlE\_DEFAULT)  
  
 predictions = self.get\_needed\_prediction(predictions)  
 is\_arrived = (any(  
 (  
 prediction.name in self.\_\_interest\_objs  
 and self.\_\_check\_area(source, prediction)  
 )  
 for prediction in predictions  
 ))  
 value = TractorStates.ARRIVED if is\_arrived else TractorStates.DEPARTED  
  
 self.\_\_determine\_event(source, src\_dt,  
 video\_name, video\_time,  
 value)

def \_\_determine\_event(  
 self,  
 source: str,  
 src\_dt: datetime.datetime,  
 video\_name: str,  
 video\_dt: datetime.timedelta,  
 state: TractorStates  
):  
 if not (current\_state := self.\_\_states.get(source, None)):  
 current\_state = State(  
 value=state.value,  
 dt=src\_dt  
 )  
 self.\_\_states[source] = current\_state  
 self.\_\_add\_new\_state(current\_state, source, video\_name, video\_dt)  
 return  
  
 if current\_state.value == TractorStates.DEPARTED.value:  
 if state == TractorStates.DEPARTED:  
 return  
 current\_state.value = state.value  
 current\_state.dt = src\_dt  
 self.\_\_add\_new\_state(current\_state, source, video\_name, video\_dt)  
 return  
  
 if state == TractorStates.ARRIVED:  
 current\_state.dt = src\_dt  
 return  
 if (src\_dt - current\_state.dt) >= self.\_\_time\_without\_ladle[source]:  
 current\_state.value = state.value  
 current\_state.dt = src\_dt  
 self.\_\_add\_new\_state(current\_state, source, video\_name, video\_dt)

def \_\_determine\_event(  
 self,  
 source: str,  
 src\_dt: datetime.datetime,  
 video\_name: str,  
 video\_dt: datetime.timedelta,  
 state: TractorStates  
):  
 if not (current\_state := self.\_\_states.get(source, None)):  
 current\_state = State(  
 value=state.value,  
 dt=src\_dt  
 )  
 self.\_\_states[source] = current\_state  
 self.\_\_add\_new\_state(current\_state, source, video\_name, video\_dt)  
 return  
  
 if current\_state.value == TractorStates.DEPARTED.value:  
 *# Если установленное состояние "Уехал" и пришло такое же состояние* if state == TractorStates.DEPARTED:  
 *# то ничего не меняем* returnelse:   
 *# Если установленное состояние "Приехал" и пришло такое же состояние* if state == TractorStates.ARRIVED:  
 *# то обновляем время приезда* current\_state.dt = src\_dt  
 return  
   
 *# если состояние "Приехал" не обновлялось менее N времени* if (src\_dt - current\_state.dt) < self.\_\_time\_without\_ladle[source]:  
 *# то состояни на "Уехал" не меняем* return

current\_state.value = state.value  
 current\_state.dt = src\_dt  
 self.\_\_add\_new\_state(current\_state, source, video\_name, video\_dt)